environments and various combinations thereof, including, by way of just a few examples: a general-purpose hardware microprocessor such as the Intel Pentium series; operating system software such as Microsoft Windows/CE, Palm OS, or Apple Mac OS (particularly for client devices and client-side processing), or Unix, Linux, or Windows/NT (the latter three particularly for network data servers and server-side processing), and/or proprietary information access platforms such as Microsoft's WebTV or the Diva Systems video-on-demand system.

2. Processing Methodology

The present invention provides a spoken natural language interface for interrogation of remote electronic databases and retrieval of desired information. A preferred embodiment of the present invention utilizes the basic methodology outlined in the flow diagram of FIG. **4** in order to provide this interface. This methodology will now be discussed.

a. Interpreting Spoken Natural Language Requests

At step **402**, the user's spoken request for information is initially received in the form of raw (acoustic) voice data by a suitable input device, as previously discussed in connection with FIGS. **1**–**2**. At step **404** the voice data received from the user is interpreted in order to understand the user's request for information. Preferably this step includes performing speech recognition in order to extract words from the voice data, and further includes natural language parsing of those words in order to generate a structured linguistic representation of the user's request.

Speech recognition in step **404** is performed using speech recognition engine **310**. A variety of commercial quality, speech recognition engines are readily available on the market, as practitioners will know. For example, Nuance Communications offers a suite of speech recognition engines, including Nuance 6, its current flagship product, and Nuance Express, a lower cost package for entry-level applications. As one other example, IBM offers the ViaVoice speech recognition engine, including a low-cost shrink-wrapped version available through popular consumer distribution channels. Basically, a speech recognition engine processes acoustic voice data and attempts to generate a text stream of recognized words.

Typically, the speech recognition engine is provided with a vocabulary lexicon of likely words or phrases that the recognition engine can match against its analysis of acoustical signals, for purposes of a given application. Preferably, the lexicon is dynamically adjusted to reflect the current user context, as established by the preceding user inputs. For example, if a user is engaged in a dialogue with the system about movie selection, the recognition engine's vocabulary may preferably be adjusted to favor relevant words and phrases, such as a stored list of proper names for popular movie actors and directors, etc. Whereas if the current dialogue involves selection and viewing of a sports event, the engine's vocabulary might preferably be adjusted to favor a stored list of proper names for professional sports teams, etc. In addition, a speech recognition engine is provided with language models that help the engine predict the most likely interpretation of a given segment of acoustical voice data, in the current context of phonemes or words in which the segment appears. In addition, speech recognition engines often echo to the user, in more or less real-time, a transcription of the engine's best guess at what the user has said, giving the user an opportunity to confirm or reject.

In a further aspect of step **404**, natural language interpreter (or parser) **320** linguistically parses and interprets the textual output of the speech recognition engine. In a preferred embodiment of the present invention, the natural-language interpreter attempts to determine both the meaning of spoken words (semantic processing) as well as the grammar of the statement (syntactic processing), such as the Gemini Natural Language Understanding System developed by SRI International. The Gemini system is described in detail in publications entitled "Gemini: A Natural Language System for Spoken-Language Understanding" and "Interleaving Syntax and Semantics in an Efficient Bottom-Up Parser," both of which are currently available online at http://www.ai.sri.com/natural-language/projects/arpa-sls/nat-lang.html. (Copies of those publications are also included in an information disclosure statement submitted herewith, and are incorporated herein by this reference). Briefly, Gemini applies a set of syntactic and semantic grammar rules to a word string using a bottom-up parser to generate a logical form, which is a structured representation of the context-independent meaning of the string. Gemini can be used with a variety of grammars, including general English grammar as well as application-specific grammars. The Gemini parser is based on "unification grammar," meaning that grammatical categories incorporate features that can be assigned values; so that when grammatical category expressions are matched in the course of parsing or semantic interpretation, the information contained in the features is combined, and if the feature values are incompatible the match fails.

It is possible for some applications to achieve a significant reduction in speech recognition error by using the natural-language processing system to re-score recognition hypotheses. For example, the grammars defined for a language parser like Gemini may be compiled into context-free grammar that, in turn, can be used directly as language models for speech recognition engines like the Nuance recognizer. Further details on this methodology are provided in the publication "Combining Linguistic and Statistical Knowledge Sources in Natural-Language Processing for ATIS" which is currently available online through http://www.ai.sri.com/natural-language/projects/arpa-sls/spnl-int.html. A copy of this publication is included in an information disclosure submitted herewith, and is incorporated herein by this reference.

In an embodiment of the present invention that may be preferable for some applications, the natural language interpreter "learns" from the past usage patterns of a particular user or of groups of users. In such an embodiment, the successfully interpreted requests of users are stored, and can then be used to enhance accuracy by comparing a current request to the stored requests, thereby allowing selection of a most probable result.

b. Constructing Navigation Queries

In step **405** request processing logic **300** identifies and selects an appropriate online data source where the desired information (in this case, current weather reports for a given city) can be found. Such selection may involve look-up in a locally stored table, or possibly dynamic searching through an online search engine, or other online search techniques. For some applications, an embodiment of the present invention may be implemented in which only access to a particular data source (such as a particular vendor's proprietary content database) is supported; in that case, step **405** may be trivial or may be eliminated entirely.

Step **406** attempts to construct a navigation query, reflecting the interpretation of step **404**. This operation is preferably performed by query construction logic **330**.

A "navigation query" means an electronic query, form, series of menu selections, or the like; being structured appropriately so as to navigate a particular data source of

interest in search of desired information. In other words, a navigation query is constructed such that it includes whatever content and structure is required in order to access desired information electronically from a particular database or data source of interest.

For example, for many existing electronic databases, a navigation query can be embodied using a formal database query language such as Standard Query Language (SQL). For many databases, a navigation query can be constructed through a more user-friendly interactive front-end, such as a series of menus and/or interactive forms to be selected or filled in. SQL is a standard interactive and programming language for getting information from and updating a database. SQL is both an ANSI and an ISO standard. As is well known to practitioners, a Relational Database Management System (RDBMS), such as Microsoft's Access, Oracle's Oracle**7**, and Computer Associates' CA-OpenIngres, allow programmers to create, update, and administer a relational database. Practitioners of ordinary skill in the art will be thoroughly familiar with the notion of database navigation through structured query, and will be readily able to appreciate and utilize the existing data structures and navigational mechanisms for a given database, or to create such structures and mechanisms where desired.

In accordance with the present invention, the query constructed in step **406** must reflect the user's request as interpreted by the speech recognition engine and the NL parser in step **404**. In embodiments of the present invention wherein data source **110** (or **210** in the corresponding embodiment of FIG. **2**) is a structured relational database or the like, step **406** of the present invention may entail constructing an appropriate Structured Query Language (SQL) query or the like, or automatically filling out a front-end query form, series of menus or the like, as described above.

In many existing Internet (and Intranet) applications, an online electronic data source is accessible to users only through the medium of interaction with a so-called Common Gateway Interface (CGI) script. Typically the user who visits a web site of this nature must fill in the fields of an online interactive form. The online form is in turn linked to a CGI script, which transparently handles actual navigation of the associated data source and produces output for viewing by the user's web browser. In other words, direct user access to the data source is not supported, only mediated access through the form and CGI script is offered.

For applications of this nature, an advantageous embodiment of the present invention "scrapes" the scripted online site where information desired by a user may be found in order to facilitate construction of an effective navigation query. For example, suppose that a user's spoken natural language request is: "What's the weather in Miami?" After this request is received at step **402** and interpreted at step **404**, assume that step **405** determines that the desired weather information is available online through the medium of a CGI-scripted interactive form. Step **406** is then preferably carried out using the expanded process diagrammed in FIG. **5**. In particular, at sub-step **520**, query construction logic **330** electronically "scrapes" the online interactive form, meaning that query construction logic **330** automatically extracts the format and structure of input fields accepted by the online form. At sub-step **522**, a navigation query is then constructed by instantiating (filling in) the extracted input format—essentially an electronic template—in a manner reflecting the user's request for information as interpreted in step **404**. The flow of control then returns to step **407** of FIG. **4**. Ultimately, when the query thus constructed by scraping is used to navigate the online data source in step **408**, the query effectively initiates the same scripted response as if a human user had visited the online site and had typed appropriate entries into the input fields of the online form.

In the embodiment just described, scraping step **520** is preferably carried out with the assistance of an online extraction utility such as WebL. WebL is a scripting language for automating tasks on the World Wide Web. It is an imperative, interpreted language that has built-in support for common web protocols like HTTP and FTP, and popular data types like HTML and XML. WebL's implementation language is Java, and the complete source code is available from Compaq. In addition, step **520** is preferably performed dynamically when necessary—in other words, on-the-fly in response to a particular user query—but in some applications it may be possible to scrape relatively stable (unchanging) web sites of likely interest in advance and to cache the resulting template information.

It will be apparent, in light of the above teachings, that preferred embodiments of the present invention can provide a spoken natural language interface atop an existing, non-voice data navigation system, whereby users can interact by means of intuitive natural language input not strictly conforming to the linear browsing architecture or other artifacts of an existing menu/text/click navigation system. For example, users of an appropriate embodiment of the present invention for a video-on-demand application can directly speak the natural request: "Show me the movie 'Unforgiven'"—instead of walking step-by-step through a typically linear sequence of genre/title/actor/director menus, scrolling and selecting from potentially long lists on each menu, or instead of being forced to use an alphanumeric keyboard that cannot be as comfortable to hold or use as a lightweight remote control. Similarly, users of an appropriate embodiment of the present invention for a web-surfing application in accordance with the process shown in FIG. **5** can directly speak the natural request: "Show me a one-month price chart for Microsoft stock" —instead of potentially having to navigate to an appropriate web site, search for the right ticker symbol, enter/select the symbol, and specify display of the desired one-month price chart, each of those steps potentially involving manual navigation and data entry to one or more different interaction screens. (Note that these examples are offered to illustrate some of the potential benefits offered by appropriate embodiments of the present invention, and not to limit the scope of the invention in any respect.)

c. Error Correction

Several problems can arise when attempting to perform searches based on spoken natural language input. As indicated at decision step **407** in the process of FIG. **4**, certain deficiencies may be identified during the process of query construction, before search of the data source is even attempted. For example, the user's request may fail to specify enough information in order to construct a navigation query that is specific enough to obtain a satisfactory search result. For example, a user might orally request "what's the weather?" whereas the national online data source identified in step **405** and scraped in step **520** might require specifying a particular city.

Additionally, certain deficiencies and problems may arise following the navigational search of the data source at step **408**, as indicated at decision step **409** in FIG. **4**. For example, with reference to a video-on-demand application, a user may wish to see the movie "Unforgiven", but perhaps the user can't recall name of the film, but knows it was

directed by and starred actor Clint Eastwood. A typical video-on-demand database might indeed be expected to allow queries specifying the name of a leading actor and/or director, but in the case of this query—as in many cases—that will not be enough to narrow the search to a single film, and additional user input in some form is required.

In the event that one or more deficiencies in the user's spoken request, as processed, result in the problems described, either at step **407** or **409**, some form of error handling is in order. A straightforward, crude technique might be for the system to respond simply "input not understood/insufficient, please try again." However, that approach will likely result in frustrated users, and is not optimal or even acceptable for most applications. Instead, a preferred technique in accordance with the present invention handles such errors and deficiencies in user input at step **412**, whether detected at step **407** or step **409**, by soliciting additional input from the user in a manner taking advantage of the partial construction already performed and via user interface modalities in addition to spoken natural language ("multi-modality"). This supplemental interaction is preferably conducted through client display device **112** (**202**, in the embodiment of FIG. **2**), and may include textual, graphical, audio and/or video media. Further details and examples are provided below. Query refinement logic **340** preferably carries out step **412**. The additional input received from the user is fed into and augments interpreting step **404**, and query construction step **406** is likewise repeated with the benefit of the augmented interpretation. These operations, and subsequent navigation step **408**, are preferably repeated until no remaining problems or deficiencies are identified at decision points **407** or **409**. Further details and examples for this query refinement process are provided immediately below.

Consider again the example in which the user of a video-on-demand application wishes to see "Unforgiven" but can only recall that it was directed by and starred Clint Eastwood. First, it bears noting that using a prior art navigational interface, such as a conventional menu interface, will likely be relatively tedious in this case. The user can proceed through a sequence of menus, such as Genre (select "western"), Title (skip), Actor ("Clint Eastwood"), and Director ("Clint Eastwood"). In each case—especially for the last two items—the user would typically scroll and select from fairly long lists in order to enter his or her desired name, or perhaps use a relatively couch-unfriendly keypad to manually type the actor's name twice.

Using a preferred embodiment of the present invention, the user instead speaks aloud, holding remote control microphone **102**, "I want to see that movie starring and directed by Clint Eastwood. Can't remember the title." At step **402** the voice data is received. At step **404** the voice data is interpreted. At step **405** an appropriate online data source is selected (or perhaps the system is directly connected to a proprietary video-on-demand provider). At step **406** a query is automatically constructed by the query construction logic **330** specifying "Clint Eastwood" in both the actor and director fields. Step **407** detects no obvious problems, and so the query is electronically submitted and the data source is navigated at step **408**, yielding a list of several records satisfying the query (e.g., "Unforgiven", "True Crime", "Absolute Power", etc.). Step **409** detects that additional user input is needed to further refine the query in order to select a particular film for viewing.

At that point, in step **412** query refinement logic **340** might preferably generate a display for client display device **112** showing the (relatively short) list of film titles that satisfy the user's stated constraints. The user can then preferably use a relatively convenient input modality, such as buttons on the remote control, to select the desired title from the menu. In a further preferred embodiment, the first title on the list is highlighted by default, so that the user can simply press an "OK" button to choose that selection. In a further preferred feature, the user can mix input modalities by speaking a response like "I want number one on the list." Alternatively, the user can preferably say, "Let's see Unforgiven," having now been reminded of the title by the menu display.

Utilizing the user's supplemental input, request processing logic **300** iterates again through steps **404** and **406**, this time constructing a fully-specified query that specifically requests the Eastwood film "Unforgiven." Step **408** navigates the data source using that query and retrieves the desired film, which is then electronically transmitted in step **410** from network server **108** to client display device **112** via communications network **106**.

Now consider again the example in which the user of a web surfing application wants to know his or her local weather, and simply asks, "what's the weather?" At step **402** the voice data is received. At step **404** the voice data is interpreted. At step **405** an online web site providing current weather information for major cities around the world is selected. At step **406** and sub-step **520**, the online site is scraped using a WebL-style tool to extract an input template for interacting with the site. At sub-step **522**, query construction logic **330** attempts to construct a navigation query by instantiating the input template, but determines (quite rightly) that a required field—name of city—cannot be determined from the user's spoken request as interpreted in step **404**. Step **407** detects this deficiency, and in step **412** query refinement logic **340** preferably generates output for client display device **112** soliciting the necessary supplemental input. In a preferred embodiment, the output might display the name of the city where the user is located highlighted by default. The user can then simply press an "OK" button—or perhaps mix modalities by saying "yes, exactly" —to choose that selection. A preferred embodiment would further display an alphabetical scrollable menu listing other major cities, and/or invite the user to speak or select the name of the desired city.

Here again, utilizing the user's supplemental input, request processing logic **300** iterates through steps **404** and **406**. This time, in performing sub-step **520**, a cached version of the input template already scraped in the previous iteration might preferably be retrieved. In sub-step **522**, query construction logic **330** succeeds this time in instantiating the input template and constructing an effective query, since the desired city has now been clarified. Step **408** navigates the data source using that query and retrieves the desired weather information, which is then electronically transmitted in step **410** from network server **108** to client display device **112** via communications network **106**.

It is worth noting that in some instances, there may be details that are not explicitly provided by the user, but that query construction logic **330** or query refinement logic **340** may preferably deduce on their own through reasonable assumptions, rather than requiring the use to provide explicit clarification. For example, in the example previously described regarding a request for a weather report, in some applications it might be preferable for the system to simply assume that the user means a weather report for his or her home area and to retrieve that information, if the cost of doing so is not significantly greater than the cost of asking the user to clarify the query. Making such an assumption

might be even more strongly justified in a preferred embodiment, as described earlier, where user histories are tracked, and where such history indicates that a particular user or group of users typically expect local information when asking for a weather forecast. At any rate, in the event such an assumption is made, if the user actually intended to request the weather for a different city, the user would then need to ask his or her question again. It will be apparent to practitioners, in light of the above teachings, that the choice of whether to program query construction logic **330** and query refinement logic **340** to make make particular assumptions will typically involve trade-offs involving user conveience that can be assessed in the context of specific applications.

3. Open Agent Architecture (OAA®)

Open Agent Architecture™ (OAA®) is a software platform, developed by the assignee of the present invention, that enables effective, dynamic collaboration among communities of distributed electronic agents. OAA is described in greater detail in co-pending U.S. patent application Ser. No. 09/225,198, which has been incorporated herein by reference. Very briefly, the functionality of each client agent is made available to the agent community through registration of the client agent's capabilities with a facilitator. A software "wrapper" essentially surrounds the underlying application program performing the services offered by each client. The common infrastructure for constructing agents is preferably supplied by an agent library. The agent library is preferably accessible in the runtime environment of several different programming languages. The agent library preferably minimizes the effort required to construct a new system and maximizes the ease with which legacy systems can be "wrapped" and made compatible with the agent-based architecture of the present invention. When invoked, a client agent makes a connection to a facilitator, which is known as its parent facilitator. Upon connection, an agent registers with its parent facilitator a specification of the capabilities and services it can provide, using a high-level, declarative Interagent Communication Language ("ICL") to express those capabilities. Tasks are presented to the facilitator in the form of ICL goal expressions. When a facilitator determines that the registered capabilities of one of its client agents will help satisfy a current goal or sub-goal thereof, the facilitator delegates that sub-goal to the client agent in the form of an ICL request. The client agent processes the request and returns answers or information to the facilitator. In processing a request, the client agent can use ICL to request services of other agents, or utilize other infrastructure services for collaborative work. The facilitator coordinates and integrates the results received from different client agents on various sub-goals, in order to satisfy the overall goal.

OAA provides a useful software platform for building systems that integrate spoken natural language as well as other user input modalities. For example, see the above-referenced co-pending patent application, especially FIG. **13** and the corresponding discussion of a "multi-modal maps" application, and FIG. **12** and the corresponding discussion of a "unified messaging" application. Another example is the InfoWiz interactive information kiosk developed by the assignee and described in the document entitled "InfoWiz: An Animated Voice Interactive Information System" available online at http://www.ai.sri.com/~oaa/applications.html. A copy of the InfoWhiz document is provided in an Information Disclosure Statement submitted herewith and incorporated herein by this reference. A further example is the "CommandTalk" application developed by the assignee for the U.S. military, as described online at http://www.ai.sri.com/~lesaf/commandtalk.html and in the following publications, copies of which are provided in an Information Disclosure Statement submitted herewith and incorporated herein by this reference:

"CommandTalk: A Spoken-Language Interface for Battlefield Simulations", 1997, by Robert Moore, John Dowding, Harry Bratt, J. Mark Gawron, Yonael Gorfu and Adam Cheyer, in "Proceedings of the Fifth Conference on Applied Natural Language Processing", Washington, D.C., pp. 1–7, Association for Computational Linguistics

"The CommandTalk Spoken Dialogue System", 1999, by Amanda Stent, John Dowding, Jean Mark Gawron, Elizabeth Owen Bratt and Robert Moore, in "Proceedings of the Thirty-Seventh Annual Meeting of the ACL", pp. 183–190, University of Maryland, College Park, Md., Association for Computational Linguistics

"Interpreting Language in Context in CommandTalk", 1999, by John Dowding and Elizabeth Owen Bratt and Sharon Goldwater, in "Communicative Agents: The Use of Natural Language in Embodied Systems", pp. 63–67, Association for Computing Machinery (ACM) Special Interest Group on Artificial Intelligence (SIGART), Seattle, Wash.

For some applications and systems, OAA can provide an advantageous platform for constructing embodiments of the present invention. For example, a representative application is now briefly presented, with reference to FIG. **6**. If the statement "show me movies starring John Wayne" is spoken into the voice input device, the voice data for this request will be sent by UI agent **650** to facilitator **600**, which in turn will ask natural language (NL) agent **620** and speech recognition agent **610** to interpret the query and return the interpretation in ICL format. The resulting ICL goal expression is then routed by the facilitator to appropriate agents—in this case, video-on-demand database agent **640**—to execute the request. Video database agent **640** preferably includes or is coupled to an appropriate embodiment of query construction logic **330** and query refinement logic **340**, and may also issue ICL requests to facilitator **600** for additional assistance—e.g., display of menus and capture of additional user input in the event that query refinement is needed—and facilitator **600** will delegate such requests to appropriate client agents in the community. When the desired video content is ultimately retrieved by video database agent **640**, UI agent **650** is invoked by facilitator **600** to display the movie.

Other spoken user requests, such as a request for the current weather in New York City or for a stock quote, would eventually lead facilitator to invoke web database agent **630** to access the desired information from an appropriate Internet site. Here again, web database agent **630** preferably includes or is coupled to an appropriate embodiment of query construction logic **330** and query refinement logic **340**, including a scraping utility such as WebL. Other spoken requests, such as a request to view recent emails or access voice mail, would lead the facilitator to invoke the appropriate email agent **660** and/or telephone agent **680**. A request to record a televised program of interest might lead facilitator **600** to invoke web database agent **630** to return televised program schedule information, and then invoke VCR controller agent **680** to program the associated VCR unit to record the desired television program at the scheduled time.

Control and connectivity embracing additional electronic home appliances (e.g., microwave oven, home surveillance system, etc.) can be integrated in comparable fashion. Indeed, an advantage of OAA-based embodiments of the present invention, that will be apparent to practitioners in light of the above teachings and in light of the teachings disclosed in the cited co-pending patent applications, is the relative ease and flexibility with which additional service agents can be plugged into the existing platform, immediately enabling the facilitator to respond dynamically to spoken natural language requests for the corresponding services.

4. Further Embodiments and Equivalents

While the present invention has been described in terms of several preferred embodiments, there are many alterations, permutations, and equivalents that may fall within the scope of this invention. It should also be noted that there are many alternative ways of implementing the methods and apparatuses of the present invention. It is therefore intended that the following appended claims be interpreted as including all such alterations, permutations, and equivalents as fall within the true spirit and scope of the present invention.

What is claimed is:

1. A method for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, wherein a data link is established between a mobile information appliance of the user and the one or more network servers, comprising the steps of:

(a) receiving a spoken request for desired information from the user utilizing the mobile information appliance of the user, wherein said mobile information appliance comprises a portable remote control device or a set-top box for a television;

(b) rendering an interpretation of the spoken request;

(c) constructing a navigation query based upon the interpretation;

(d) utilizing the navigation query to select a portion of the electronic data source; and

(e) transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user.

2. The method of claim **1**, wherein the step of rendering the interpretation of the spoken request is performed by the mobile information appliance.

3. The method of claim **1**, wherein the step of rendering the interpretation of the spoken request is performed by the mobile information appliance.

4. The method of claim **1**, further comprising the steps of soliciting additional input from the user, including user interaction in a modality different than the original request; refining the navigation query, based upon the additional input; and using the refined navigation query to select a portion of the electronic data source.

5. The method of claim **1**, wherein the data link includes a cellular telephone system.

6. The method of claim **1**, wherein steps (a)–(d) are performed with respect to multiple users.

7. The method of claim **1**, wherein the mobile information appliance is a wireless telephone.

8. The method of claim **1**, wherein the mobile information appliance is a portable computing device.

9. The method of claim **8**, wherein the portable computing device is a personal digital assistant.

10. A computer program embodied on a computer readable medium for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, wherein a data link is established between a mobile information appliance of the user and the one or more network servers, comprising:

(a) a code segment that receives a spoken request for desired information from the user utilizing the mobile information appliance of the user, wherein said mobile information appliance comprises a portable remote control device or a set-top box for a television;

(b) a code segment that renders an interpretation of the spoken request;

(c) a code segment that constructs a navigation query based upon the interpretation;

(d) a code segment that utilizes the navigation query to select a portion of the electronic data source; and

(e) a code segment that transmits the selected portion of the electronic data source from the network server to the mobile information appliance of the user.

11. The computer program of claim **10**, wherein the rendering of the interpretation of the spoken request is performed at the one or more network servers.

12. The computer program of claim **10**, wherein the rendering of the interpretation of the spoken request is performed by the mobile information appliance.

13. The computer program of claim **10**, further comprising a code segment that solicits additional input from the user, including user interaction in a modality different than the original request; a code segment that refines the navigation query, based upon the additional input; and a code segment that uses the refined navigation query to select a portion of the electronic data source.

14. The computer program of claim **10**, wherein the data link includes a wireless telephone system.

15. The computer program of claim **10**, wherein code segments (a)–(d) are executed with respect to multiple users.

16. The computer program of claim **10**, wherein the mobile information appliance is a wireless telephone.

17. The computer program of claim **10**, wherein the mobile information appliance is a portable computing device.

18. The computer program of claim **17**, wherein the portable computing device is a personal digital assistant.

19. A system for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, comprising:

(a) a mobile information appliance operable to receive a spoken request for desired information from the user, wherein said mobile information appliance comprises a portable remote control device or a set-top box for a television;

(b) spoken language processing logic, operable to render an interpretation of the spoken request;

(c) query construction logic, operable to construct a navigation query based upon the interpretation;

(d) navigation logic, operable to select a portion of the electronic data source using the navigation query, and

(e) electronic communications infrastructure for transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user.

20. The system of claim **19**, wherein the spoken language processing logic renders the interpretation of the spoken request at the one or more network servers.

21. The system of claim **19**, wherein the spoken language processing logic renders the interpretation of the spoken request at the mobile information appliance.

22. The system of claim **19**, further comprising user interaction logic operable to solicit additional input from the user, including user interaction in a modality different than the original request; and query refining logic operable to refine the navigation query based upon the additional input; wherein the navigation logic users the refined navigation query to select a portion of the electronic data source.

23. The system of claim **19**, wherein the data link includes a cellular telephone system.

24. The system of claim **19**, wherein the system operates with respect to multiple users.

25. The system of claim **19**, wherein the mobile information appliance is a wireless telephone.

26. The system of claim **19**, wherein the mobile information appliance is a portable computing device.

27. The system of claim **26**, wherein the portable computing device is a personal digital assistant.

* * * * *

US Patent & Trademark Office

US 6,757,718
USPTO Transaction Information*

SEQ.[δ]	DATE	DESCRIPTION
1	18 Jan 2017	File Marked Found
2	05 Apr 2016	File Marked Found
3	16 Aug 2006	ENTITY STATUS SET TO UNDISCOUNTED (INITIAL DEFAULT SETTING OR STATUS CHANGE)
4	29 Jun 2004	Recordation of Patent Grant Mailed
5	10 Jun 2004	Issue Notification Mailed
6	29 Jun 2004	Patent Issue Date Used in PTA Calculation
7	01 Jun 2004	Receipt into Pubs
8	27 May 2004	Application Is Considered Ready for Issue
9	06 May 2003	Issue Fee Payment Verified
10	06 May 2003	Workflow - Drawings Finished
11	06 May 2003	Workflow - Drawings Matched with File at Contractor
12	03 May 2004	Receipt into Pubs
13	29 Mar 2004	Receipt into Pubs
14	25 Mar 2004	Workflow - File Sent to Contractor
15	25 Mar 2004	Receipt into Pubs
16	25 Mar 2004	Receipt into Pubs
17	06 May 2003	Workflow - Drawings Received at Contractor
18	06 May 2003	Workflow - Drawings Sent to Contractor
19	06 May 2003	Issue Fee Payment Received
20	13 Mar 2003	Dispatch to Publications
21	11 Mar 2003	Mail Notice of Allowance
22	11 Mar 2003	Mail Formal Drawings Required
23	10 Mar 2003	Formal Drawings Required
24	10 Mar 2003	Notice of Allowance Data Verification Completed
25	10 Mar 2003	Case Docketed to Examiner in GAU
26	09 Jan 2003	Mail Examiner Interview Summary (PTOL - 413)
27	23 Dec 2002	Examiner Interview Summary Record (PTOL - 413)
28	08 Jan 2003	Date Forwarded to Examiner
29	06 Jan 2003	Response after Non-Final Action
30	04 Oct 2002	Mail Non-Final Rejection
31	01 Oct 2002	Non-Final Rejection
32	05 Sep 2002	Case Docketed to Examiner in GAU
33	29 Jul 2002	Information Disclosure Statement (IDS) Filed
34	29 Jul 2002	Information Disclosure Statement (IDS) Filed
35	18 Jul 2002	Date Forwarded to Examiner
36	18 Jul 2002	Response after Non-Final Action
37	18 Jul 2002	Request for Extension of Time - Granted
38	28 Mar 2002	Case Docketed to Examiner in GAU
39	26 Mar 2002	Case Docketed to Examiner in GAU
40	19 Feb 2002	Mail Non-Final Rejection
41	19 Feb 2002	Non-Final Rejection
42	11 Feb 2002	Date Forwarded to Examiner
43	08 Feb 2002	Request for Continued Examination (RCE)
44	11 Feb 2002	Disposal for a RCE / CPA / R129
45	08 Feb 2002	Request for Extension of Time - Granted
46	08 Feb 2002	Workflow - Request for RCE - Begin

* Document generated on 01/24/2017 by PATENTEC from official USPTO records, external to this file. Information deemed accurate, but not Certified.

[δ] Transaction Sequence Number (SEQ.) is unrelated to Paper Number in File Table of contents.

US Patent & Trademark Office

US 6,757,718
USPTO Transaction Information*

SEQ.[δ]	DATE	DESCRIPTION
47	28 Jan 2002	Mail Advisory Action (PTOL - 303)
48	28 Jan 2002	Advisory Action (PTOL-303)
49	17 Jan 2002	Date Forwarded to Examiner
50	10 Jan 2002	Response after Final Action
51	16 Jan 2002	Mail Examiner Interview Summary (PTOL - 413)
52	08 Jan 2002	Examiner Interview Summary Record (PTOL - 413)
53	10 Oct 2001	Mail Final Rejection (PTOL - 326)
54	09 Oct 2001	Final Rejection
55	01 Oct 2001	Date Forwarded to Examiner
56	21 Sep 2001	Response after Non-Final Action
57	01 Oct 2001	Change in Power of Attorney (May Include Associate POA)
58	01 Oct 2001	Correspondence Address Change
59	01 Oct 2001	Change in Power of Attorney (May Include Associate POA)
60	21 May 2001	Correspondence Address Change
61	30 Apr 2001	Information Disclosure Statement (IDS) Filed
62	30 Apr 2001	Information Disclosure Statement (IDS) Filed
63	24 Apr 2001	Mail Non-Final Rejection
64	20 Apr 2001	Non-Final Rejection
65	30 Jun 2000	Preliminary Amendment
66	30 Jun 2000	Preliminary Amendment
67	05 Mar 2001	Case Docketed to Examiner in GAU
68	30 Nov 2000	Case Docketed to Examiner in GAU
69	15 Nov 2000	Application Dispatched from OIPE
70	15 Nov 2000	Application Is Now Complete
71	01 Sep 2000	Notice Mailed--Application Incomplete--Filing Date Assigned
72	31 Aug 2000	Correspondence Address Change
73	24 Jul 2000	IFW Scan &amp; PACR Auto Security Review
74	30 Jun 2000	Initial Exam Team nn

* Document generated on 01/24/2017 by PATENTEC from official USPTO records, external to this file. Information deemed accurate, but not Certified.

[δ] Transaction Sequence Number (SEQ.) is unrelated to Paper Number in File Table of contents.

US Patent & Trademark Office

US 6,757,718
Assignment History*

Assignment: 1 / 1

Reel / Frame:	039857/0097	**Recorded:**	09/26/2016	**Pages in document:**	5
Conveyance:	ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS).				
Assignor:	Sri International			**Exec. Dt:**	05/20/2016
Assignee:	IPA TECHNOLOGIES INC. 600 ANTON BLVD. SUITE 1350 COSTA MESA, CALIFORNIA 92626				
Correspondent:	IPA TECHNOLOGIES INC. 600 ANTON BLVD. SUITE 1350 COSTA MESA, CA 92626				

* Document generated on 01/24/2017 by PATENTEC from official USPTO records, external to this file. Information deemed accurate, but not Certified.

United States Patent and Trademark Office

Office of the Commissioner for Patents

MOBILE NAVIGATION OF NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN INPUT

PATENT #	APPLICATION #	FILING DATE	ISSUE DATE
6757718	09608872	06/30/2000	06/29/2004

Payment Window Status

No maintenance fees are due.

WINDOW	STATUS	FEES
11.5 Year	Closed	Paid

Window	First Day to Pay	Surcharge Starts	Last Day to Pay	Status	Fees
3.5 Year	06/29/2007	01/01/2008	06/30/2008	Closed	Paid
7.5 Year	06/29/2011	12/30/2011	06/29/2012	Closed	Paid
11.5 Year	06/29/2015	12/30/2015	06/29/2016	Closed	Paid

Patent Holder Information

Customer #

Entity Status UNDISCOUNTED

Phone Number 4085055100

Address

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY, NJ 07702
UNITED STATES

Generated: 01/24/2017 09:43:59

ISSUE CLASSIFICATION	
Class	Subclass
709	218

PATENT NUMBER

6757718

U.S. UTILITY Patent Application

O.I.P.E.	PATENT DATE
SCANNED Q.A. A.G.	JUN 2 9 2004

APPLICATION NO.	CONT/PRIOR	CLASS	SUBCLASS	ART UNIT	EXAMINER
09/608872	D	~~704~~ 709	~~202~~ 218	~~2641~~ ~~2145~~ 2141	BACKER Jean

APPLICANTS
Christine Halverson
Luc Julia
Dimitris Voutsas
Adam Cheyer

TITLE
Mobile navigation of network-based electronic information using spoken input

PTO-2040
12/99

ISSUING CLASSIFICATION

ORIGINAL		CROSS REFERENCE(S)						
CLASS	SUBCLASS	CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)					
709	218	709	202	217	219	227		
INTERNATIONAL CLASSIFICATION		704	257					
G06F	15/16							
					□ Continued on Issue Slip Inside File Jacket			

□ TERMINAL DISCLAIMER	DRAWINGS			CLAIMS ALLOWED	
	Sheets Drwg.	Figs. Drwg.	Print Fig.	Total Claims	Print Claim for O.G.
	7	7	1A	27	1

□ The term of this patent subsequent to ____________ (date) has been disclaimed.

(Assistant Examiner) (Date)

NOTICE OF ALLOWANCE MAILED

3-11-03

□ The term of this patent shall not extend beyond the expiration date of U.S Patent. No. ____________

Frantz B Jean 3/7/03
(Primary Examiner) (Date)

ISSUE FEE

Amount Due	Date Paid
$1650.00	5/6/03

□ The terminal ____months of this patent have been disclaimed.

L. Godison 3-13-03
(Legal Instruments Examiner) (Date)

ISSUE BATCH NUMBER

WARNING:
The information disclosed herein may be restricted. Unauthorized disclosure may be prohibited by the United States Code Title 35, Sections 122, 181 and 368. Possession outside the U.S. Patent & Trademark Office is restricted to authorized employees and contractors only.

Form PTO-436A
(Rev. 6/99)

FILED WITH: □ DISK (CRF) □ FICHE □ CD-ROI
(Attached in pocket on right inside fl

ISSUE FEE IN FILE

(FACE)

PATENT APPLICATION

09608872

[redacted]

INITIALS 7/12/00 12

CONTENTS

		Date Received (Incl. C. of M.) or Date Mailed			Date Received (Incl. C. of M.) or Date Mailed
1.	Application papers.	8/3/00	42.		
2.	Fee Dec		43.		
3.	Dec Surchage	11-2-00	44.		
4.	Prel. Amdt A	6-30-00	45.		
5.	Prel. Amdt B	6-30-00	46.		
6.	Rejection 3months	4/24/01	47.		
7.	IDS	4-30-01	48.		
8.	Associate P/A	5-14-01	49.		
9.	Change of address	5-14-01	50.		
10.	Ext. of time ②	9-21-01	51.		
11.	Revocation & Power of Attorney	9-21-01	52.		
12.	Req. for reconsideration	9-21-01	53.		
13.	notice of acceptance	10-2-01	54.		
14.	10-19-01 Final Rejection 3months	10/10/01	55.		
15.	Interview Summary	1-16-02	56.		
16.	Amdt C	1-10-02	57.		
17.	2-28-02 Advisory Action	1-28-02	58.		
18.	Ext of time 1 month / Request for RCE	2-~~8~~-02	59.		
19.	Rejection 3months	2-19-02	60.		
20.	~~Interview Summary~~		61.		
21.	Ext of time (2 mos)	7/18/02	62.		
22.	Amdt D	7/18/02	63.		
23.	IDS	[illegible]	64.		
24.	Rejection 3months	10-4-02	65.		
25.	Response	1-6-03	66.		
26.	Interview Summary	01/08/03	67.		
27.	notice of allow.	3/11/03	68.		
28.	Formal Drawings (7 shts) set 1	05-06-03	69.		
29.			70.		
30.			71.		
31.			72.		
32.			73.		
33.			74.		
34.			75.		
35.			76.		
36.			77.		
37.			78.		
38.			79.		
39.			80.		
40.			81.		
41.			82.		

(LEFT OUTSIDE)

ISSUE SLIP STAPLE AREA (for additional cross references)

POSITION	INITIALS	ID NO.	DATE
FEE DETERMINATION		62814	7/12/00
O.I.P.E. CLASSIFIER		48	7/12/00
FORMALITY REVIEW		60874	8/3/00
RESPONSE FORMALITY REVIEW			
		60894	11/5/00

INDEX OF CLAIMS

✓ .... Rejected		N .... Non-elected	
= .... Allowed		I .... Interference	
— (Through numeral)... Canceled		A .... Appeal	
÷ .... Restricted		O .... Objected	

Claim		Date									
Final	Original										
	1										
	2										
	3										
	4										
	5										
	6										
	7										
	8										
	9										
	10										
	11										
	12										
	13										
	14										
	15										
	16										
	17										
	18										
	19										
	20										
	21										
	22										
	23										
	24										
	25										
	26										
	27										
	28										
	29										
	30										
	31										
	32										
	33										
	34										
	35										
	36										
	37										
	38										
	39										
	40										
	41										
	42										
	43										
	44										
	45										
	46										
	47										
	48										
	49										
	50										

Claim		Date									
Final	Original	4/9/01	10/4/01	2/10/02	9/29/02	3/7/03					
—	51					—					
—	52					—					
—	53					—					
—	54					—					
—	55					—					
1	56	✓	✓	✓	✓	=					
2	57										
3	58										
4	59										
5	60										
6	61										
7	62										
8	63										
9	64										
10	65										
11	66										
12	67										
13	68										
14	69										
15	70										
16	71										
17	72										
18	73										
19	74										
20	75										
21	76										
22	77										
23	78										
24	79										
25	80										
26	81										
27	82	✓	✓	✓	✓	=					
	83										
	84										
	85										
	86										
	87										
	88										
	89										
	90										
	91										
	92										
	93										
	94										
	95										
	96										
	97										
	98										
	99										
	100										

Claim		Date									
Final	Original										
	101										
	102										
	103										
	104										
	105										
	106										
	107										
	108										
	109										
	110										
	111										
	112										
	113										
	114										
	115										
	116										
	117										
	118										
	119										
	120										
	121										
	122										
	123										
	124										
	125										
	126										
	127										
	128										
	129										
	130										
	131										
	132										
	133										
	134										
	135										
	136										
	137										
	138										
	139										
	140										
	141										
	142										
	143										
	144										
	145										
	146										
	147										
	148										
	149										
	150										

If more than 150 claims or 10 actions
staple additional sheet here

(LEFT INSIDE)

SEARCHED

Class	Sub.	Date	Exmr.
709	202	4/6/01	F.B
709	218		
707	5 3 4		
704	257		
updated		9/30/02	
updated		3/7/03	
704	270.1 275 246 257	3/7/03	
709	217 219 227		

INTERFERENCE SEARCHED

Class	Sub.	Date	Exmr.
709	202 217 218 219 227	3/7/03	
704	257		

SEARCH NOTES
(INCLUDING SEARCH STRATEGY)

	Date	Exmr.
WEST SEARCH	4/10/01	F.B
West, Derwent TDBD, EPO, JPO	9/29/02 9/30/02	
consulted with David Wiley	9/30/02	
updated.	3/7/03	
West, Derwent TDBD, EPO, JPO, PG Pubs inventor search check for possible double patenting	3/7/03	
NPL	3/7/03	

(RIGHT OUTSIDE)

UNITED STATES PATENT AND TRADEMARK OFFICE

COMMISSIONER FOR PATENTS
UNITED STATES PATENT AND TRADEMARK OFFICE
WASHINGTON, D.C. 20231
www.uspto.gov

Bib Data Sheet

CONFIRMATION NO. 2382

SERIAL NUMBER	FILING DATE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
09/608,872	06/30/2000 RULE	709	2155	SRIlp037B

APPLICANTS

Christine Halversen, San Jose, CA;
Luc Julia, Menlo Park, CA;
Dimitris Voutsas, Thessaloniki, GREECE;
Adam Cheyer, Palo Alto, CA;

** CONTINUING DATA *************************

THIS APPLICATION IS A CON OF 09/524,095 03/13/2000
WHICH IS A CIP OF 09/225,198 01/05/1999
WHICH CLAIMS BENEFIT OF 60/124,718 03/17/1999
AND SAID 09/524,095 03/13/2000
CLAIMS BENEFIT OF 60/124,720 03/17/1999
AND CLAIMS BENEFIT OF 60/124,719 03/17/1999

** FOREIGN APPLICATIONS ********************

IF REQUIRED, FOREIGN FILING LICENSE GRANTED ** SMALL ENTITY **
** 08/31/2000

		STATE OR COUNTRY	SHEETS DRAWING	TOTAL CLAIMS	INDEPENDENT CLAIMS
Foreign Priority claimed	□ yes ☑ no	CA	7	27	3
35 USC 119 (a-d) conditions met	□ yes ☑ no □ Met after Allowance				
Verified and Acknowledged	Examiner's Signature Initials				

ADDRESS

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY ,NJ 07702

TITLE

mobile navigation of network-based electronic information using spoken input

FILING FEE RECEIVED 473	FEES: Authority has been given in Paper No. _________ to charge/credit DEPOSIT ACCOUNT No. _________ for following:	□ All Fees □ 1.16 Fees (Filing) □ 1.17 Fees (Processing Ext. of time) □ 1.18 Fees (Issue) □ Other _______________ □ Credit

Navigating Network-Based Electronic Information Using Spoken Natural Language Input with Multimodal Error Feedback

Background of the Invention

~~This is~~ Is a Continuation In Part of co-pending U.S. Patent Application No. 09/225,198, filed January 5, 1999, Provisional U.S. Patent Application No. 60/124,718, filed March 17, 1999, Provisional U.S. Patent Application No. 60/124,720, filed March 17, 1999, and Provisional U.S. Patent Application No. 60/124,719, filed March 17, 1999, from which applications priority is claimed and these application are incorporated herein by reference.

The present invention relates generally to the navigation of electronic data by means of spoken natural language requests, and to feedback mechanisms and methods for resolving the errors and ambiguities that may be associated with such requests.

As global electronic connectivity continues to grow, and the universe of electronic data potentially available to users continues to expand, there is a growing need for information navigation technology that allows relatively naïve users to navigate and access desired data by means of natural language input. In many of the most important markets -- including the home entertainment arena, as well as mobile computing -- spoken natural language input is highly desirable, if not ideal. As just one example, the proliferation of high-bandwidth communications infrastructure for the home entertainment market (cable, satellite, broadband) enables delivery of movies-on-demand and other interactive multimedia content to the consumer's home television set. For users to take full advantage of this content stream ultimately requires interactive navigation of content databases in a manner that is too complex for user-friendly selection by means of a traditional remote-control clicker. Allowing spoken natural language requests as the input modality for rapidly searching and accessing desired content is an important objective for a successful consumer entertainment product in a context offering a dizzying range of database content choices. As further examples, this same need to drive navigation of (and transaction with) relatively complex data warehouses using spoken natural language requests applies equally to surfing the Internet/Web or other networks for general information, multimedia content, or e-commerce transactions.

In general, the existing navigational systems for browsing electronic databases and data warehouses (search engines, menus, etc.), have been designed without navigation via spoken natural language as a specific goal. So today's world is full of existing electronic data navigation systems that do not assume browsing via natural spoken commands, but rather assume text and mouse-click inputs (or in the case of TV remote controls, even less). Simply recognizing voice commands within an extremely limited vocabulary and grammar -- the spoken equivalent of button/click input (e.g., speaking "channel 5" selects TV channel 5) -- is really not sufficient by itself to satisfy the objectives described above. In order to deliver a true "win" for users, the voice-driven front-end must accept spoken natural language input in a manner that is intuitive to users. For example, the front-end should not require learning a highly specialized command language or format. More fundamentally, the front-end must allow users to speak directly in terms of what the user ultimately wants -- e.g., "I'd like to see a Western film directed by Clint Eastwood" -- as opposed to speaking in terms of arbitrary navigation structures (e.g., hierarchical layers of menus, commands, etc.) that are essentially artifacts reflecting constraints of the pre-existing text/click navigation system. At the same time, the front-end must recognize and accommodate the reality that a stream of naïve spoken natural language input will, over time, typically present a variety of errors and/or ambiguities: e.g., garbled/unrecognized words (did the user say "Eastwood" or "Easter"?) and under-constrained requests ("Show me the Clint Eastwood movie"). An approach is needed for handling and resolving such errors and ambiguities in a rapid, user-friendly, non-frustrating manner.

What is needed is a methodology and apparatus for rapidly constructing a voice-driven front-end atop an existing, non-voice data navigation system, whereby users can interact by means of intuitive natural language input not strictly conforming to the step-by-step browsing architecture of the existing navigation system, and wherein any errors or ambiguities in user input are rapidly and conveniently resolved. The solution to this need should be compatible with the constraints of a multi-user, distributed environment such as the Internet/Web or a proprietary high-bandwidth content delivery network; a solution contemplating one-at-a-time user interactions at a single location is insufficient, for example.

SUMMARY OF THE INVENTION

Sub B3

The present invention addresses the above needs by providing a system, method, and article of manufacture for navigating network-based electronic data sources in response to spoken NL input requests. When a spoken natural language input request is received from a user, it is interpreted, such as by using a speech recognition engine to extract speech data from acoustic voice signals, and using a natural language parser to linguistically parse the speech data. The interpretation of the spoken natural language request can be performed on a computing device locally with the user or remotely from the user. The resulting interpretation of the request is thereupon used to automatically construct an operational navigation query to retrieve the desired information from one or more electronic network data sources, which is then transmitted to a client device of the user. If the network data source is a database, the navigation query is constructed in the format of a database query language.

Typically, errors or ambiguities emerge in the interpretation of the spoken NL request, such that the system cannot instantiate a complete, valid navigational template. This is to be expected occasionally, and one preferred aspect of the invention is the ability to handle such errors and ambiguities in relatively graceful and user-friendly manner. Instead of simply rejecting such input and defaulting to traditional input modes or simply asking the user to try again, a preferred embodiment of the present invention seeks to converge rapidly toward instantiation of a valid navigational template by soliciting additional clarification from the user as necessary, either before or after a navigation of the data source, via multimodal input, i.e., by means of menu selection or other input modalities including and in addition to spoken natural language. This clarifying, multi-modal dialogue takes advantage of whatever partial navigational information has been gleaned from the initial interpretation of the user's spoken NL request. This clarification process continues until the system converges toward an adequately instantiated navigational template, which is in turn used to navigate the network-based data and retrieve the user's desired information. The retrieved information is transmitted across the network and presented to the user on a suitable client display device.

In a further aspect of the present invention, the construction of the navigation query includes extracting an input template for an online scripted interface to the data source and using the input template to construct the navigation query. The extraction of the input template can include dynamically scraping the online scripted interface.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention, together with further advantages thereof, may best be understood by reference to the following description taken in conjunction with the accompanying drawings in which:

Figure 1a illustrates a system providing a spoken natural language interface for network-based information navigation, in accordance with an embodiment of the present invention with server-side processing of requests;

Figure 1b illustrates another system providing a spoken natural language interface for network-based information navigation, in accordance with an embodiment of the present invention with client-side processing of requests;

Figure 2 illustrates a system providing a spoken natural language interface for network-based information navigation, in accordance with an embodiment of the present invention for a mobile computing scenario;

Figure 3 illustrates the functional logic components of a request processing module in accordance with an embodiment of the present invention;

Figure 4 illustrates a process utilizing spoken natural language for navigating an electronic database in accordance with one embodiment of the present invention;

Figure 5 illustrates a process for constructing a navigational query for accessing an online data source via an interactive, scripted (e.g., CGI) form; and

Figure 6 illustrates an embodiment of the present invention utilizing a community of distributed, collaborating electronic agents.

Detailed Description of the Invention

1. System Architecture

a. Server-End Processing of Spoken Input

Figure 1a is an illustration of a data navigation system driven by spoken natural language input, in accordance with one embodiment of the present invention. As shown, a user's voice input data is captured by a voice input device 102, such as a microphone. Preferably voice input device 102 includes a button or the like that can be pressed or held-down to activate a listening mode, so that the system need not continually pay attention to, or be confused by, irrelevant background noise. In one preferred embodiment well-suited for the home entertainment setting, voice input device 102 is a portable remote control device with an integrated microphone, and the voice data is transmitted from device 102 preferably via infrared (or other wireless) link to communications box 104 (e.g., a set-top box or a similar communications device that is capable of retransmitting the raw voice data and/or processing the voice data) local to the user's environment and coupled to communications network 106. The voice data is then transmitted across network 106 to a remote server or servers 108. The voice data may preferably be transmitted in compressed digitized form, or alternatively --particularly where bandwidth constraints are significant-- in analog format (e.g., via frequency modulated transmission), in the latter case being digitized upon arrival at remote server 108.

At remote server 108, the voice data is processed by request processing logic 300 in order to understand the user's request and construct an appropriate query or request for navigation of remote data source 110, in accordance with the interpretation process exemplified in Figure 4 and Figure 5 and discussed in greater detail below. For purposes of executing this process, request processing logic 300 comprises functional modules including speech recognition engine 310, natural language (NL) parser 320, query construction logic 330, and query refinement logic 340, as shown in Figure 3. Data source 110 may comprise database(s), Internet/web site(s), or other electronic information repositories, and preferably resides on a central server or servers -- which may or may not be the same as server 108, depending on the storage and bandwidth needs of the application and the resources available to the practitioner. Data source 110 may include multimedia content, such as movies or other digital video and audio content, other various forms of entertainment data, or other electronic information. The contents of data source 110 are navigated -- i.e., the contents are accessed and searched, for retrieval of the particular information desired by the user -- using the processes of Figures 4 and 5 as described in greater detail below.

Once the desired information has been retrieved from data source 110, it is electronically transmitted via network 106 to the user for viewing on client display device 112. In a preferred embodiment well-suited for the home entertainment setting, display device 112 is a television monitor or similar audiovisual entertainment device, typically in stationary position for comfortable viewing by users. In addition, in such preferred embodiment, display device 112 is coupled to or integrated with a communications box (which is preferably the same as communications box 104, but may also be a separate unit) for receiving and decoding/formatting the desired electronic information that is received across communications network 106.

Network 106 is a two-way electronic communications network and may be embodied in electronic communication infrastructure including coaxial (cable television) lines, DSL, fiber-optic cable, traditional copper wire (twisted pair), or any other type of hardwired connection. Network 106 may also include a wireless connection such as a satellite-based connection, cellular connection, or other type of wireless connection. Network 106 may be part of the Internet and may support TCP/IP communications, or may be embodied in a proprietary network, or in any other electronic communications network infrastructure, whether packet-switched or connection-oriented. A design consideration is that network 106 preferably provide suitable bandwidth depending upon the nature of the content anticipated for the desired application.

b. Client-End Processing of Spoken Input

Figure 1b is an illustration of a data navigation system driven by spoken natural language input, in accordance with a second embodiment of the present invention. Again, a user's voice input data is captured by a voice input device 102, such as a microphone. In the embodiment shown in Figure 1b, the voice data is transmitted from device 202 to requests processing logic 300, hosted on a local speech processor, for processing and interpretation. In the preferred embodiment illustrated in Figure 1b, the local speech processor is conveniently integrated as part of communications box 104, although implementation in a physically separate (but communicatively coupled) unit is also possible as will be readily apparent to those of skill in the art. The voice data is processed by the components of request processing logic 300 in order to understand the user's request and construct an appropriate query or request for navigation of remote data source 110, in accordance with the interpretation process exemplified in Figures 4 and 5 as discussed in greater detail below.

The resulting navigational query is then transmitted electronically across network 106 to data source 110, which preferably resides on a central server or servers 108. As in Figure 1a, data source 110 may comprise database(s), Internet/web site(s), or other electronic information repositories, and preferably may include multimedia content, such as movies or other digital video and audio content, other various forms of entertainment data, or other electronic information. The contents of data source 110 are then navigated -- i.e., the contents are accessed and searched, for retrieval of the particular information desired by the user -- preferably using the process of Figures 4 and 5 as described in greater detail below. Once the desired information has been retrieved from data source 110, it is electronically transmitted via network 106 to the user for viewing on client display device 112.

In one embodiment in accordance with Figure 1b and well-suited for the home entertainment setting, voice input device 102 is a portable remote control device with an integrated microphone, and the voice data is transmitted from device 102 preferably via infrared (or other wireless) link to the local speech processor. The local speech processor is coupled to communications network 106, and also preferably to client display device 112 (especially for purposes of query refinement transmissions, as discussed below in connection with Figure 4, step 412), and preferably may be integrated within or coupled to communications box 104. In addition, especially for purposes of a home entertainment application, display device 112 is preferably a television monitor or similar audiovisual entertainment device, typically in stationary position for comfortable viewing by users. In addition, in such preferred embodiment, display device 112 is coupled to a communications box (which is preferably the same as communications box 104, but may also be a physically separate unit) for receiving and decoding/formatting the desired electronic information that is received across communications network 106.

Design considerations favoring server-side processing and interpretation of spoken input requests, as exemplified in Figure 1a, include minimizing the need to distribute costly computational hardware and software to all client users in order to perform speech and language processing. Design considerations favoring client-side processing, as exemplified in Figure 1b, include minimizing the quantity of data sent upstream across the network from each client, as the speech recognition is performed before transmission across the network and only the query data and/or request needs to be sent, thus reducing the upstream bandwidth requirements.

c. Mobile Client Embodiment

A mobile computing embodiment of the present invention may be implemented by practitioners as a variation on the embodiments of either Figure 1a or Figure 1b. For example, as depicted in Figure 2, a mobile variation in accordance with the server-side processing architecture illustrated in Figure 1a may be implemented by replacing voice input device 102, communications box 104, and client display device 112, with an integrated, mobile, information appliance 202 such as a cellular telephone or wireless personal digital assistant (wireless PDA). Mobile information appliance 202 essentially performs the functions of the replaced components. Thus, mobile information appliance 202 receives spoken natural language input requests from the user in the form of voice data, and transmits that data (preferably via wireless data receiving station 204) across communications network 206 for server-side interpretation of the request, in similar fashion as described above in connection with Figure 1. Navigation of data source 210 and retrieval of desired information likewise proceeds in an analogous manner as described above. Display information transmitted electronically back to the user across network 206 is displayed for the user on the display of information appliance 202, and audio information is output through the appliance's speakers.

Practitioners will further appreciate, in light of the above ,teachings, that if mobile information appliance 202 is equipped with sufficient computational processing power, then a mobile variation of the client-side architecture exemplified in Figure 2 may similarly be implemented. In that case, the modules corresponding to request processing logic 300 would be embodied locally in the computational resources of mobile information appliance 202, and the logical flow of data would otherwise follow in a manner analogous to that previously described in connection with Figure 1b.

As illustrated in Figure 2, multiple users, each having their own client input device, may issue requests, simultaneously or otherwise, for navigation of data source 210. This is equally true (though not explicitly drawn) for the embodiments depicted in Figures 1a and 1b. Data source 210 (or 100), being a network accessible information resource, has typically already been constructed to support access requests from simultaneous multiple network users, as known by practitioners of ordinary skill in the art. In the case of server-side speech processing, as exemplified in Figures 1a and 2, the interpretation logic and error correction logic modules are also preferably designed and implemented to support queuing and multi-tasking of requests from multiple simultaneous network users, as will be appreciated by those of skill in the art.

It will be apparent to those skilled in the art that additional implementations, permutations and combinations of the embodiments set forth in Figures 1a, 1b, and 2 may be created without straying from the scope and spirit of the present invention. For example, practitioners will understand, in light of the above teachings and design considerations, that it is possible to divide and allocate the functional components of request processing logic 300 between client and server. For example, speech recognition -- in entirety, or perhaps just early stages such as feature extraction -- might be performed locally on the client end, perhaps to reduce bandwidth requirements, while natural language parsing and other necessary processing might be performed upstream on the server end, so that more extensive computational power need not be distributed locally to each client. In that case, corresponding portions of request processing logic 300, such as speech recognition engine 310 or portions thereof, would reside locally at the client as in Figure 1b, while other component modules would be hosted at the server end as in Figures 1a and 2.

Further, practitioners may choose to implement the each of the various embodiments described above on any number of different hardware and software computing platforms and environments and various combinations thereof, including, by way of just a few examples: a general-purpose hardware microprocessor such as the Intel Pentium series; operating system software such as Microsoft Windows/CE, Palm OS, or Apple Mac OS (particularly for client devices and client-side processing), or Unix, Linux, or Windows/NT (the latter three particularly for network data servers and server-side processing), and/or proprietary information access platforms such as Microsoft's WebTV or the Diva Systems video-on-demand system.

2. Processing Methodology

The present invention provides a spoken natural language interface for interrogation of remote electronic databases and retrieval of desired information. A preferred embodiment of the present invention utilizes the basic methodology outlined in the flow diagram of Figure 4 in order to provide this interface. This methodology will now be discussed.

a. Interpreting Spoken Natural Language Requests

At step 402, the user's spoken request for information is initially received in the form of raw (acoustic) voice data by a suitable input device, as previously discussed in connection with Figures 1-2. At step 404 the voice data received from the user is interpreted in order to understand the user's request for information. Preferably this step includes performing speech recognition in order to extract words from the voice data, and further includes natural language parsing of those words in order to generate a structured linguistic representation of the user's request.

Speech recognition in step 404 is performed using speech recognition engine 310. A variety of commercial quality, speech recognition engines are readily available on the market, as practitioners will know. For example, Nuance Communications offers a suite of speech recognition engines, including Nuance 6, its current flagship product, and Nuance Express, a lower cost package for entry-level applications. As one other example, IBM offers the ViaVoice speech recognition engine, including a low-cost shrink-wrapped version available through popular consumer distribution channels. Basically, a speech recognition engine processes acoustic voice data and attempts to generate a text stream of recognized words.

Typically, the speech recognition engine is provided with a vocabulary lexicon of likely words or phrases that the recognition engine can match against its analysis of acoustical signals, for purposes of a given application. Preferably, the lexicon is dynamically adjusted to reflect the current user context, as established by the preceding user inputs. For example, if a user is engaged in a dialogue with the system about movie selection, the recognition engine's vocabulary may preferably be adjusted to favor relevant words and phrases, such as a stored list of proper names for popular movie actors and directors, etc. Whereas if the current dialogue involves selection and viewing of a sports event, the engine's vocabulary might preferably be adjusted to favor a stored list of proper names for professional sports teams, etc. In addition, a speech recognition engine is provided with language models that help the engine predict the most likely interpretation of a given segment of acoustical voice data, in the current context of phonemes or words in which the segment appears. In addition, speech recognition engines often echo to the user, in more or less real-time, a transcription of the engine's best guess at what the user has said, giving the user an opportunity to confirm or reject.

In a further aspect of step 404, natural language interpreter (or parser) 320 linguistically parses and interprets the textual output of the speech recognition engine. In a preferred embodiment of the present invention, the natural-language interpreter attempts to determine both the meaning of spoken words (semantic processing) as well as the grammar of the statement (syntactic processing), such as the Gemini Natural Language Understanding System developed by SRI International. The Gemini system is described in detail in publications entitled "Gemini: A Natural Language System for Spoken-Language Understanding" and "Interleaving Syntax and Semantics in an Efficient Bottom-Up Parser," both of which are currently available online at http://www.ai.sri.com/natural-language/projects/arpa-sls/nat-lang.html. (Copies of those publications are also included in an information disclosure statement submitted herewith, and are incorporated herein by this reference). Briefly, Gemini applies a set of syntactic and semantic grammar rules to a word string using a bottom-up parser to generate a logical form, which is a structured representation of the context-independent meaning of the string. Gemini can be used with a variety of grammars, including general English grammar as well as application-specific grammars. The Gemini parser is based on "unification grammar," meaning that grammatical categories incorporate features that can be assigned values; so that when grammatical category expressions are matched in the course of parsing or semantic interpretation, the information contained in the features is combined, and if the feature values are incompatible the match fails.

It is possible for some applications to achieve a significant reduction in speech recognition error by using the natural-language processing system to re-score recognition hypotheses. For example, the grammars defined for a language parser like Gemini may be compiled into context-free grammar that, in turn, can be used directly as language models for speech recognition engines like the Nuance recognizer. Further details on this methodology are provided in the publication "Combining Linguistic and Statistical Knowledge Sources in Natural-Language Processing for ATIS" which is currently available online through http://www.ai.sri.com/natural-language/projects/arpa-sls/spnl-int.html. A copy of this publication is included in an information disclosure submitted herewith, and is incorporated herein by this reference.

In an embodiment of the present invention that may be preferable for some applications, the natural language interpreter "learns" from the past usage patterns of a particular user or of groups of users. In such an embodiment, the successfully interpreted requests of users are stored, and can then be used to enhance accuracy by comparing a current request to the stored requests, thereby allowing selection of a most probable result.

b. Constructing Navigation Queries

In step 405 request processing logic 300 identifies and selects an appropriate online data source where the desired information (in this case, current weather reports for a given city) can be found. Such selection may involve look-up in a locally stored table, or possibly dynamic searching through an online search engine, or other online search techniques. For some applications, an embodiment of the present invention may be implemented in which only access to a particular data source (such as a particular vendor's proprietary content database) is supported; in that case, step 405 may be trivial or may be eliminated entirely.

Step 406 attempts to construct a navigation query, reflecting the interpretation of step 404. This operation is preferably performed by query construction logic 330.

A "navigation query" means an electronic query, form, series of menu selections, or the like; being structured appropriately so as to navigate a particular data source of interest in search of desired information. In other words, a navigation query is constructed such that it includes whatever content and structure is required in order to access desired information electronically from a particular database or data source of interest.

For example, for many existing electronic databases, a navigation query can be embodied using a formal database query language such as Standard Query Language (SQL). For many databases, a navigation query can be constructed through a more user-friendly interactive front-end, such as a series of menus and/or interactive forms to be selected or filled in. SQL is a standard interactive and programming language for getting information from and updating a database. SQL is both an ANSI and an ISO standard. As is well known to practitioners, a Relational Database Management System (RDBMS), such as Microsoft's Access, Oracle's Oracle7, and Computer Associates' CA-OpenIngres, allow programmers to create, update, and administer a relational database. Practitioners of ordinary skill in the art will be thoroughly familiar with the notion of database navigation through structured query, and will be readily able to appreciate and utilize the existing data structures and navigational mechanisms for a given database, or to create such structures and mechanisms where desired.

In accordance with the present invention, the query constructed in step 406 must reflect the user's request as interpreted by the speech recognition engine and the NL parser in step 404. In embodiments of the present invention wherein data source 110 (or 210 in the corresponding embodiment of Figure 2) is a structured relational database or the like, step 406 of the present invention may entail constructing an appropriate Structured Query Language (SQL) query or the like, or automatically filling out a front-end query form, series of menus or the like, as described above.

In many existing Internet (and Intranet) applications, an online electronic data source is accessible to users only through the medium of interaction with a so-called Common Gateway Interface (CGI) script. Typically the user who visits a web site of this nature must fill in the fields of an online interactive form. The online form is in turn linked to a CGI script, which transparently handles actual navigation of the associated data source and produces output for viewing by the user's web browser. In other words, direct user access to the data source is not supported, only mediated access through the form and CGI script is offered.

For applications of this nature, an advantageous embodiment of the present invention "scrapes" the scripted online site where information desired by a user may be found in order to facilitate construction of an effective navigation query. For example, suppose that a user's spoken natural language request is: "What's the weather in Miami?" After this request is received at step 402 and interpreted at step 404, assume that step 405 determines that the desired weather information is available online through the medium of a CGI-scripted interactive form. Step 406 is then preferably carried out using the expanded process diagrammed in Figure 5. In particular, at sub-step 520, query construction logic 330 electronically "scrapes" the online interactive form, meaning that query construction logic 330 automatically extracts the format and structure of input fields accepted by the online form. At sub-step 522, a navigation query is then constructed by instantiating (filling in) the extracted input format -- essentially an electronic template -- in a manner reflecting the user's request for information as interpreted in step 404. The flow of control then returns to step 407 of Figure 4. Ultimately, when the query thus constructed by scraping is used to navigate the online data source in step 408, the query effectively initiates the same scripted response as if a human user had visited the online site and had typed appropriate entries into the input fields of the online form.

In the embodiment just described, scraping step 520 is preferably carried out with the assistance of an online extraction utility such as WebL. WebL is a scripting language for automating tasks on the World Wide Web. It is an imperative, interpreted language that has built-in support for common web protocols like HTTP and FTP, and popular data types like HTML and XML. WebL's implementation language is Java, and the complete source code is available from Compaq. In addition, step 520 is preferably performed dynamically when necessary -- in other words, on-the-fly in response to a particular user query -- but in some applications it may be possible to scrape relatively stable (unchanging) web sites of likely interest in advance and to cache the resulting template information.

It will be apparent, in light of the above teachings, that preferred embodiments of the present invention can provide a spoken natural language interface atop an existing, non-voice data navigation system, whereby users can interact by means of intuitive natural language input not strictly conforming to the linear browsing architecture or other artifacts of an existing menu/text/click navigation system. For example, users of an appropriate embodiment of the present invention for a video-on-demand application can directly speak the natural request: "Show me the movie 'Unforgiven'" -- instead of walking step-by-step through a typically linear sequence of genre/title/actor/director menus, scrolling and selecting from potentially long lists on each menu, or instead of being forced to use an alphanumeric keyboard that cannot be as comfortable to hold or use as a lightweight remote control. Similarly, users of an appropriate embodiment of the present invention for a web-surfing application in accordance with the process shown in Figure 5 can directly speak the natural request: "Show me a one-month price chart for Microsoft stock" -- instead of potentially having to navigate to an appropriate web site, search for the right ticker symbol, enter/select the symbol, and specify display of the desired one-month price chart, each of those steps potentially involving manual navigation and data entry to one or more different interaction screens. (Note that these examples are offered to illustrate some of the potential benefits offered by appropriate embodiments of the present invention, and not to limit the scope of the invention in any respect.)

c. Error Correction

Several problems can arise when attempting to perform searches based on spoken natural language input. As indicated at decision step 407 in the process of Figure 4, certain deficiencies may be identified during the process of query construction, before search of the data source is even attempted. For example, the user's request may fail to specify enough information in order to construct a navigation query that is specific enough to obtain a satisfactory search result. For example, a user might orally request "what's the weather?" whereas the national online data source identified in step 405 and scraped in step 520 might require specifying a particular city.

Additionally, certain deficiencies and problems may arise following the navigational search of the data source at step 408, as indicated at decision step 409 in Figure 4. For example, with reference to a video-on-demand application, a user may wish to see the movie "Unforgiven", but perhaps the user can't recall name of the film, but knows it was directed by and starred actor Clint Eastwood. A typical video-on-demand database might indeed be expected to allow queries specifying the name of a leading actor and/or director, but in the case of this query -- as in many cases -- that will not be enough to narrow the search to a single film, and additional user input in some form is required.

In the event that one or more deficiencies in the user's spoken request, as processed, result in the problems described, either at step 407 or 409, some form of error handling is in order. A straightforward, crude technique might be for the system to respond simply *"input not understood / insufficient; please try again."* However, that approach will likely result in frustrated users, and is not optimal or even acceptable for most applications. Instead, a preferred technique in accordance with the present invention handles such errors and deficiencies in user input at step 412, whether detected at step 407 or step 409, by soliciting additional input from the user in a manner taking advantage of the partial construction already performed and via user interface modalities in addition to spoken natural language ("multi-modality"). This supplemental interaction is preferably conducted through client display device 112 (202, in the embodiment of Figure 2), and may include textual, graphical, audio and/or video media. Further details and examples are provided below. Query refinement logic 340 preferably carries out step 412. The additional input received from the user is fed into and augments interpreting step 404, and query construction step 406 is likewise repeated with the benefit of the augmented interpretation. These operations, and subsequent navigation step 408, are preferably repeated until no remaining problems or deficiencies are identified at decision points 407 or 409. Further details and examples for this query refinement process are provided immediately below.

Consider again the example in which the user of a video-on-demand application wishes to see "Unforgiven" but can only recall that it was directed by and starred Clint Eastwood. First, it bears noting that using a prior art navigational interface, such as a conventional menu interface, will likely be relatively tedious in this case. The user can proceed through a sequence of menus, such as Genre (select "western"), Title (skip), Actor ("Clint Eastwood"), and Director ("Clint Eastwood"). In each case --especially for the last two items -- the user would typically scroll and select from fairly long lists in order to enter his or her desired name, or perhaps use a relatively couch-unfriendly keypad to manually type the actor's name twice.

Using a preferred embodiment of the present invention, the user instead speaks aloud, holding remote control microphone 102, "I want to see that movie starring and directed by Clint Eastwood. Can't remember the title." At step 402 the voice data is received. At step 404 the voice data is interpreted. At step 405 an appropriate online data source is selected (or perhaps the system is directly connected to a proprietary video-on-demand provider). At step 406 a query is automatically constructed by the query construction logic 330 specifying "Clint Eastwood" in both the actor and director fields. Step 407 detects no obvious problems, and so the query is electronically submitted and the data source is navigated at step 408, yielding a list of several records satisfying the query (e.g., "Unforgiven", "True Crime", "Absolute Power", etc.). Step 409 detects that additional user input is needed to further refine the query in order to select a particular film for viewing.

At that point, in step 412 query refinement logic 340 might preferably generate a display for client display device 112 showing the (relatively short) list of film titles that satisfy the user's stated constraints. The user can then preferably use a relatively convenient input modality, such as buttons on the remote control, to select the desired title from the menu. In a further preferred embodiment, the first title on the list is highlighted by default, so that the user can simply press an "OK" button to choose that selection. In a further preferred feature, the user can mix input modalities by speaking a response like "I want number one on the list." Alternatively, the user can preferably say, "Let's see Unforgiven," having now been reminded of the title by the menu display.

Utilizing the user's supplemental input, request processing logic 300 iterates again through steps 404 and 406, this time constructing a fully-specified query that specifically requests the Eastwood film "Unforgiven." Step 408 navigates the data source using that query and retrieves the desired film, which is then electronically transmitted in step 410 from network server 108 to client display device 112 via communications network 106.

Now consider again the example in which the user of a web surfing application wants to know his or her local weather, and simply asks, "what's the weather?" At step 402 the voice data is received. At step 404 the voice data is interpreted. At step 405 an online web site providing current weather information for major cities around the world is selected. At step 406 and sub-step 520, the online site is scraped using a WebL-style tool to extract an input template for interacting with the site. At sub-step 522, query construction logic 330 attempts to construct a navigation query by instantiating the input template, but determines (quite rightly) that a required field -- name of city -- cannot be determined from the user's spoken request as interpreted in step 404. Step 407 detects this deficiency, and in step 412 query refinement logic 340 preferably generates output for client display device 112 soliciting the necessary supplemental input. In a preferred embodiment, the output might display the name of the city where the user is located highlighted by default. The user can then simply press an "OK" button -- or perhaps mix modalities by saying "yes, exactly" -- to choose that selection. A preferred embodiment would further display an alphabetical scrollable menu listing other major cities, and/or invite the user to speak or select the name of the desired city.

Here again, utilizing the user's supplemental input, request processing logic 300 iterates through steps 404 and 406. This time, in performing sub-step 520, a cached version of the input template already scraped in the previous iteration might preferably be retrieved. In sub-step 522, query construction logic 330 succeeds this time in instantiating the input template and constructing an effective query, since the desired city has now been clarified. Step 408 navigates the data source using that query and retrieves the desired weather information, which is then electronically transmitted in step 410 from network server 108 to client display device 112 via communications network 106.

It is worth noting that in some instances, there may be details that are not explicitly provided by the user, but that query construction logic 330 or query refinement logic 340 may preferably deduce on their own through reasonable assumptions, rather than requiring the use to provide explicit clarification. For example, in the example previously described regarding a request for a weather report, in some applications it might be preferable for the system to simply assume that the user means a weather report for his or her home area and to retrieve that information, if the cost of doing so is not significantly greater than the cost of asking the user to clarify the query. Making such an assumption might be even more strongly justified in a preferred embodiment, as described earlier, where user histories are tracked, and where such history indicates that a particular user or group of users typically expect local information when asking for a weather forecast. At any rate, in the event such an assumption is made, if the user actually intended to request the weather for a different city, the user would then need to ask his or her question again. It will be apparent to practitioners, in light of the above teachings, that the choice of whether to program query construction logic 330 and query refinement logic 340 to make make particular assumptions will typically involve trade-offs involving user conveience that can be assessed in the context of specific applications.

3. Open Agent Architecture (OAA®)

Open Agent Architecture™ (OAA®) is a software platform, developed by the assignee of the present invention, that enables effective, dynamic collaboration among communities of distributed electronic agents. OAA is described in greater detail in co-pending U.S. Patent Application No. 09/225,198, which has been incorporated herein by reference. Very briefly, the functionality of each client agent is made available to the agent community through registration of the client agent's capabilities with a facilitator. A software "wrapper" essentially surrounds the underlying application program performing the services offered by each client. The common infrastructure for constructing agents is preferably supplied by an *agent library*. The agent library is preferably accessible in the runtime environment of several different programming languages. The agent library preferably minimizes the effort required to construct a new system and maximizes the ease with which legacy systems can be "wrapped" and made compatible with the agent-based architecture of the present invention. When invoked, a client agent makes a connection to a facilitator, which is known as its *parent facilitator*. Upon connection, an agent registers with its parent facilitator a specification of the capabilities and services it can provide, using a high-level, declarative Interagent Communication Language (*"ICL"*) to express those capabilities. Tasks are presented to the facilitator in the form of ICL goal expressions. When a facilitator determines that the registered capabilities of one of its client agents will help satisfy a current goal or sub-goal thereof, the facilitator delegates that sub-goal to the client agent in the form of an ICL request. The client agent processes the request and returns answers or information to the facilitator. In processing a request, the client agent can use *ICL* to request services of other agents, or utilize other infrastructure services for collaborative work. The facilitator coordinates and integrates the results received from different client agents on various sub-goals, in order to satisfy the overall goal.

OAA provides a useful software platform for building systems that integrate spoken natural language as well as other user input modalities. For example, see the above-referenced co-pending patent application, especially Figure 13 and the corresponding discussion of a "multi-modal maps" application, and Figure 12 and the corresponding discussion of a "unified messaging" application. Another example is the InfoWiz interactive information kiosk developed by the assignee and described in the document entitled "InfoWiz: An Animated Voice Interactive Information System" available online at http://www.ai.sri.com/~oaa/applications.html. A copy of the InfoWhiz document is provided in an Information Disclosure Statement submitted herewith and incorporated herein by this reference. A further example is the "CommandTalk" application developed by the assignee for the U.S. military, as described online at http://www.ai.sri.com/~lesaf/commandtalk.html and in the following publications, copies of which are provided in an Information Disclosure Statement submitted herewith and incorporated herein by this reference:

- "CommandTalk: A Spoken-Language Interface for Battlefield Simulations", 1997, by Robert Moore, John Dowding, Harry Bratt, J. Mark Gawron, Yonael Gorfu and Adam Cheyer, in "Proceedings of the Fifth Conference on Applied Natural Language Processing", Washington, DC, pp. 1-7, Association for Computational Linguistics

- "The CommandTalk Spoken Dialogue System", 1999, by Amanda Stent, John Dowding, Jean Mark Gawron, Elizabeth Owen Bratt and Robert Moore, in "Proceedings of the Thirty-Seventh Annual Meeting of the ACL", pp. 183-190, University of Maryland, College Park, MD, Association for Computational Linguistics

- "Interpreting Language in Context in CommandTalk", 1999, by John Dowding and Elizabeth Owen Bratt and Sharon Goldwater, in "Communicative Agents: The Use of Natural Language in Embodied Systems", pp. 63-67, Association for Computing Machinery (ACM) Special Interest Group on Artificial Intelligence (SIGART), Seattle, WA

For some applications and systems, OAA can provide an advantageous platform for constructing embodiments of the present invention. For example, a representative application is now briefly presented, with reference to Figure 6. If the statement "show me movies starring John Wayne" is spoken into the voice input device, the voice data for this request will be sent by UI agent 650 to facilitator 600, which in turn will ask natural language (NL) agent 620 and speech recognition agent 610 to interpret the query and return the interpretation in *ICL* format. The resulting *ICL* goal expression is then routed by the facilitator to appropriate agents -- in this case, video-on-demand database agent 640 -- to execute the request. Video database agent 640 preferably includes or is coupled to an appropriate embodiment of query construction logic 330 and query refinement logic 340, and may also issue ICL requests to facilitator 600 for additional assistance -- e.g., display of menus and capture of additional user input in the event that query refinement is needed -- and facilitator 600 will delegate such requests to appropriate client agents in the community. When the desired video content is ultimately retrieved by video database agent 640, UI agent 650 is invoked by facilitator 600 to display the movie.

Other spoken user requests, such as a request for the current weather in New York City or for a stock quote, would eventually lead facilitator to invoke web database agent 630 to access the desired information from an appropriate Internet site. Here again, web database agent 630 preferably includes or is coupled to an appropriate embodiment of query construction logic 330 and query refinement logic 340, including a scraping utility such as WebL. Other spoken requests, such as a request to view recent emails or access voice mail, would lead the facilitator to invoke the appropriate email agent 660 and/or telephone agent 680. A request to record a televised program of interest might lead facilitator 600 to invoke web database agent 630 to return televised program schedule information, and then invoke VCR controller agent 680 to program the associated VCR unit to record the desired television program at the scheduled time.

Control and connectivity embracing additional electronic home appliances (e.g., microwave oven, home surveillance system, etc.) can be integrated in comparable fashion. Indeed, an advantage of OAA-based embodiments of the present invention, that will be apparent to practitioners in light of the above teachings and in light of the teachings disclosed in the cited co-pending patent applications, is the relative ease and flexibility with which additional service agents can be plugged into the existing platform, immediately enabling the facilitator to respond dynamically to spoken natural language requests for the corresponding services.

4. Further Embodiments and Equivalents

While the present invention has been described in terms of several preferred embodiments, there are many alterations, permutations, and equivalents that may fall within the scope of this invention. It should also be noted that there are many alternative ways of implementing the methods and apparatuses of the present invention. It is therefore intended that the following appended claims be interpreted as including all such alterations, permutations, and equivalents as fall within the true spirit and scope of the present invention.

CLAIMS

What is claimed is:

1. A method for utilizing spoken natural language for navigating an electronic data source, the electronic data source being located at one or more network servers located remotely from a user, comprising the steps of:

   (a) receiving a spoken natural language ("NL") request for desired information from the user;

   (b) rendering an interpretation of the spoken natural language request;

   (c) constructing at least part of a navigation query based upon the interpretation;

   (d) soliciting additional input from the user, including user interaction in a modality different than the original request;

   (e) refining the navigation query, based upon the additional input;

   (f) using the refined navigation query to select a portion of the electronic data source; and

   (g) transmitting the selected portion of the electronic data source from the network server to a client device of the user.

2. The method of claim 1, wherein the step of rendering an interpretation further includes deriving linguistic information by using a speech recognition engine and an NL parser.

3. The method of claim 1, wherein the step of constructing a navigation query further includes the steps of extracting an input template for an online scripted interface to the data source, and using the input template to construct the navigation query.

4. The method of claim 3, wherein the step of extracting an input template includes dynamically scraping the online scripted interface.

5. The method of claim 1, wherein the navigation query is constructed in the format of a database query language.

6. The method of claim 1, wherein the step of rendering an interpretation and the step of constructing a navigation query are performed, at least in part, on a computing device located locally with the user.

7. The method of claim 1, wherein the step of rendering an interpretation and the step of constructing a navigation query are performed, at least in part, on a network computing device located remotely from the user.

8. The method of claim 1, wherein the step of soliciting additional input is performed in response to one or more deficiencies encountered during the step of constructing a navigation query.

9. The method of claim 8, wherein the deficiencies include unresolved words of the spoken NL request.

10. The method of claim 8, wherein the deficiencies include one or more required elements of the navigational query ~~not~~ determinable from the interpretation of the spoken NL request.

11. The method of claim 1, wherein the step of soliciting additional input is performed in response to one or more deficiencies encountered after a first navigation of the data source using the navigation query constructed in step (c).

12. The method of claim 11, wherein the deficiencies include existence of more than one data record within the data source responsive to the navigation query.

13. The method of claim 11, wherein the deficiencies include failure to identify a single data record within the data source responsive to the navigation query.

14. The method of claim 1, wherein the input modality of step (d) includes selecting from a displayed option menu.

15. The method of claim 14, wherein the act of selecting from the displayed option menu is performed by speaking.

16. The method of claim 1, wherein the method is performed with respect to a plurality of simultaneous users and corresponding client devices.

17. The method of claim 1, further including the step of selecting the data source from among a plurality of candidate electronic data sources, in response to the interpretation of the spoken NL request.

18. The method of claim 1, wherein the electronic data source stores multimedia content including at least one of video content and audio content.

19. A system for utilizing spoken natural language to navigate an electronic data source, the electronic data source being located at one or more network servers located remotely from a user, the system comprising:

(a) a portable microphone operable to receive a spoken natural language ("NL") request for desired information from the user;

(b) spoken language processing logic, operable to render an interpretation of the spoken natural language request;

(c) query construction logic, operable to construct a navigation query in response to the interpretation of the spoken natural language request;

(d) user interaction logic, operable to solicit additional input from the user, including user interaction in a modality different than the original request;

(e) query refining logic, operable to refine the navigation query, based upon the additional input;

(f) navigation logic, operable to select a portion of the electronic data source using the navigation query; and

(g) electronic communications infrastructure for transmitting the selected portion of the electronic data source from the network server to a primarily stationary, display device located locally with the user.

20. The system of claim 19, wherein the spoken language processing logic includes speech recognition logic and an NL parsing logic for deriving linguistic information.

21. The system of claim 19, wherein the spoken language processing logic extracts an input template for an online scripted interface to the data source, and uses the input template to construct the navigation query.

22. The system of claim 21, wherein the spoken language processing logic dynamically scrapes the online scripted interface.

23. The system of claim 19, wherein the query construction logic constructs the query in the format of a database query language.

24. The system of claim 19, wherein at least a portion of the spoken language processing logic is hosted on a computing device located locally with the user, and wherein the portable microphone is electronically coupled to the local computing device.

25. The system of claim 19, wherein at least a portion of the spoken language processing logic is hosted on a network computing device located remotely from the user, and wherein the portable microphone sends data to the remote network computing device via the communications infrastructure.

26. The system of claim 19, wherein the user interaction logic solicits additional input in response to one or more deficiencies encountered during construction of the navigation query.

27. The system of claim 26, wherein the deficiencies include unresolved words of the spoken NL request.

28. The system of claim 26, wherein the deficiencies include one or more required elements of the navigational query not determinable from the interpretation of the spoken NL request.

29. The system of claim 19, wherein the user interaction logic solicits additional input in response to one or more deficiencies encountered after a first navigation of the data source performed by the navigation logic.

30. The system of claim 29, wherein the deficiencies include existence of more than one data record within the data source responsive to the navigation query.

31. The system of claim 29, wherein the deficiencies include failure to identify a single data record within the data source responsive to the navigation query.

32. The system of claim 19, wherein the user interaction logic displays an option menu.

33. The system of claim 32, wherein the act of selecting from the displayed option menu is performed by speaking.

34. The system of claim 19, wherein the navigation logic selects the data source from among a plurality of candidate electronic data sources, in response to the interpretation of the spoken NL request.

35. The system of claim 19, wherein the electronic data source stores multimedia content including at least ~~one of~~ video content and audio content.

36. The system of claim 19, wherein the display device receives data from the electronicdata source on the network servers via a communications box.

37. The system of claim 19, wherein the electronic communication infrastructure is a two-way infrastructure and is selected from among one or more of the following group: {coaxial cable, DSL, satellite, wireless/cellular, fiber-optic}.

38. An computer program embodied on a computer readable medium for utilizing spoken natural language for navigating an electronic data source, the

electronic data source being located at one or more network servers located remotely from a user, comprising:

(a) a code segment that receives a spoken natural language ("NL") request for desired information from the user;

(b) a code segment that renders an interpretation of the spoken natural language request;

(c) a code segment that constructs at least part of a navigation query based upon the interpretation;

(d) a code segment that solicits additional input from the user, including user interaction in a modality different than the original request;

(e) a code segment that refines the navigation query, based upon the additional input;

(f) a code segment that uses the refined navigation query to select a portion of the electronic data source; and

(g) a code segment that transmits the selected portion of the electronic data source from the network server to a primarily stationary, display device located locally with the user.

39. The computer program of claim 38, further comprising a code segment that derives linguistic information by using a speech recognition engine and an NL parser.

40. The computer program of claim 38, further comprising a code segment that extract an input template for an online scripted interface to the data source, and a code segment that uses the input template to construct the navigation query.

41. The computer program of claim 40, further comprising a code segment that dynamically scrapes the online scripted interface.

42. The computer program of claim 38, wherein the navigation query is constructed in the format of a database query language.

43. The computer program of claim 38, wherein rendering of the interpretation and the construction of the navigation query are performed, at least in part, on a computing device located locally with the user.

44. The computer program of claim 38, wherein the rendering of the interpretation and the construction of a navigation query are performed, at least in part, on a network computing device located remotely from the user.

45. The computer program of claim 38, wherein code segment that solicits additional input solicits the additional input in response to one or more deficiencies encountered during the constructing of the navigation query.

46. The computer program of claim 45, wherein the deficiencies include unresolved words of the spoken NL request.

47. The computer program of claim 45, wherein the deficiencies include one or more required elements of the navigational query not determinable from the interpretation of the spoken NL request.

48. The computer program of claim 38, wherein the code segment that solicits the additional input solicits the additional input in response to one or more deficiencies encountered after a first navigation of the data source.

49. The computer program of claim 48, wherein the deficiencies include existence of more than one data record within the data source responsive to the navigation query.

50. The computer program of claim 48, wherein the deficiencies include failure to identify a single data record within the data source responsive to the navigation query.

51. The computer program of claim 38, wherein code segment that solicits additional input displays an option menu.

52. The computer program of claim 51, wherein the act of selecting from the displayed option menu is performed by speaking.

~~53. The computer program of claim 38, wherein the code segments of the computer program operate with respect to a plurality of simultaneous users and corresponding client devices.~~

~~54. The computer program of claim 38, further comprising a code segment that selects the data source from among a plurality of candidate electronic data sources, in response to the interpretation of the spoken NL request.~~

~~55. The computer program of claim 38, wherein the electronic data source stores multimedia content including at least one of video content and audio content.~~

add a B4

NAVIGATING NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN NATURAL LANGUAGE INPUT WITH MULTIMODAL ERROR FEEDBACK

ABSTRACT OF THE INVENTION

A system, method, and article of manufacture are provided for navigating an electronic data source by means of spoken natural language. When a spoken natural language input request is received from a user, it is interpreted. Additional input is solicited from the user in a modality different than the original request and used to refine the navigation query. The resulting interpretation of the request is thereupon used to automatically construct an operational navigation query to retrieve the desired information from one or more electronic network data sources.

Post-it® Fax Note 7671	Date 6/29/00 # of pages ▶ 7
To Domenic Kotab	From K. Elarsmi
Co./Dept.	Co. SRI Intl.
Phone # 408-971-4660	Phone # 650-859-6631
Fax # 408-971-4660	Fax # 650.859-6420

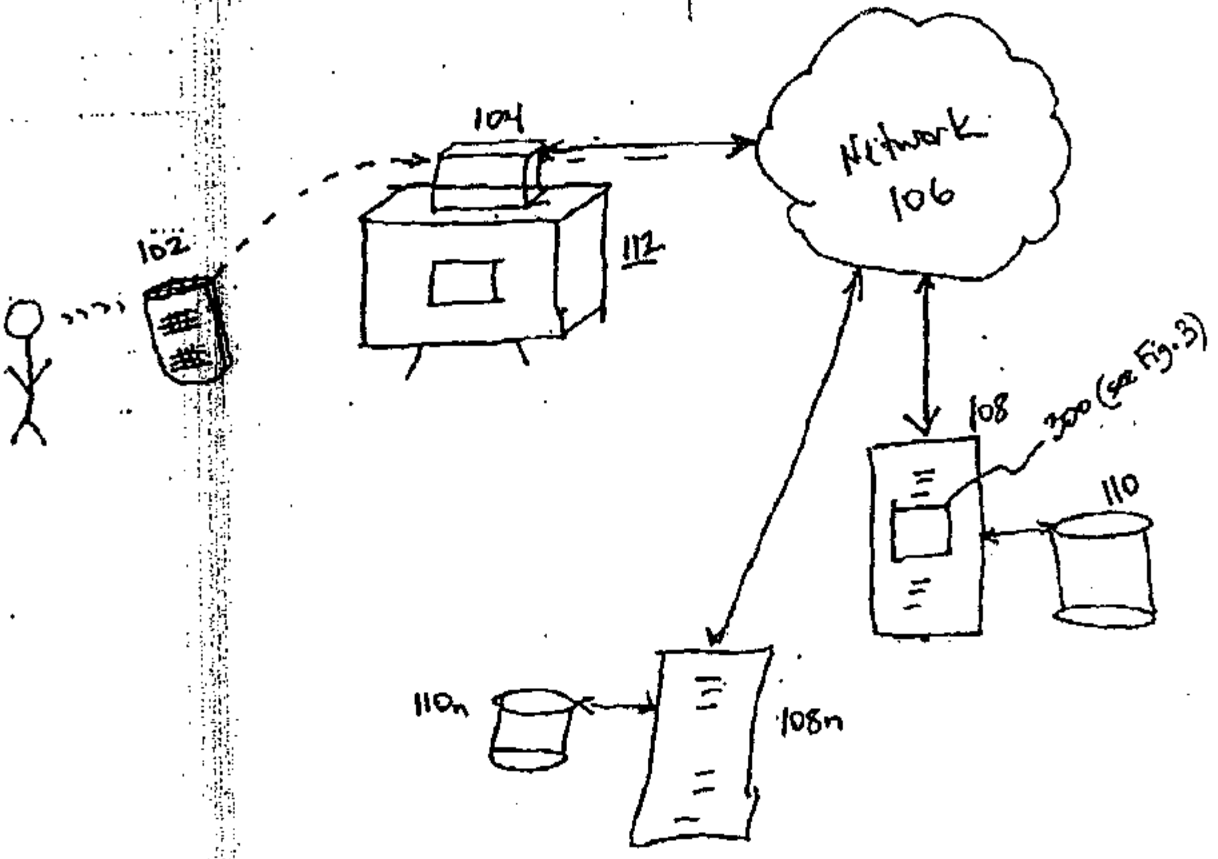


Fig. 1a

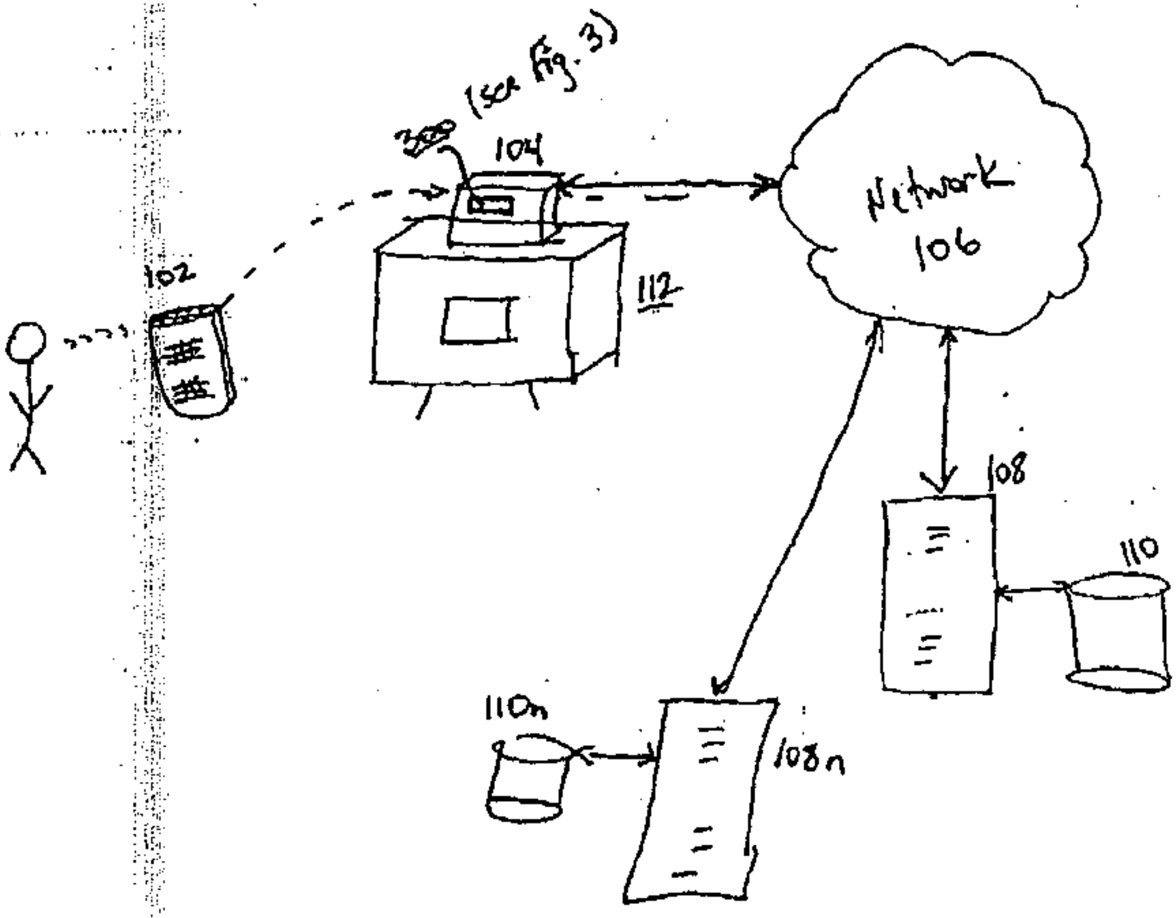


Fig. 1b

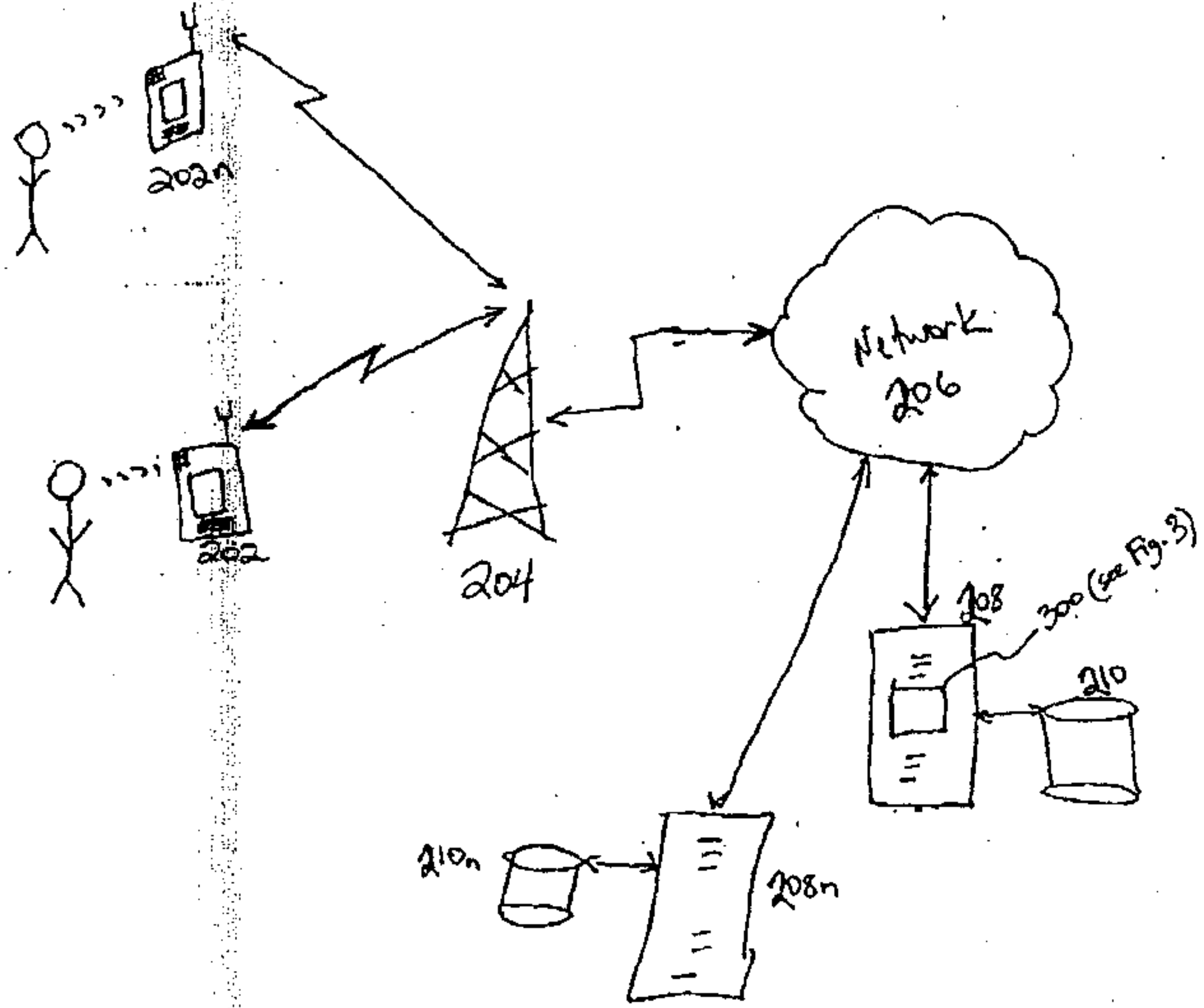


Fig. 2

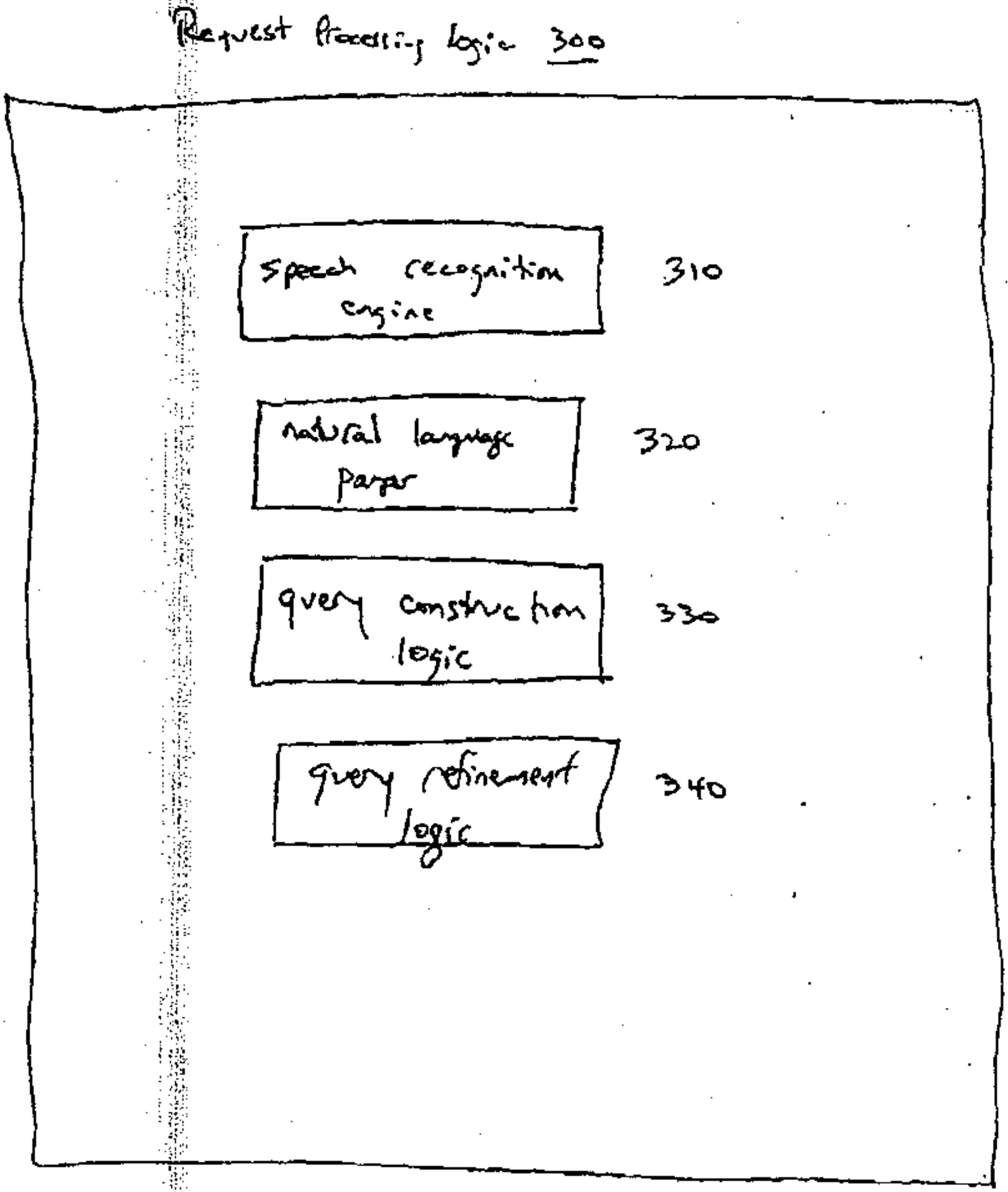

Request Processing logic 300
speech recognition engine
310
natural language parser
320
query construction logic
330
query refinement logic
340

Fig. 3

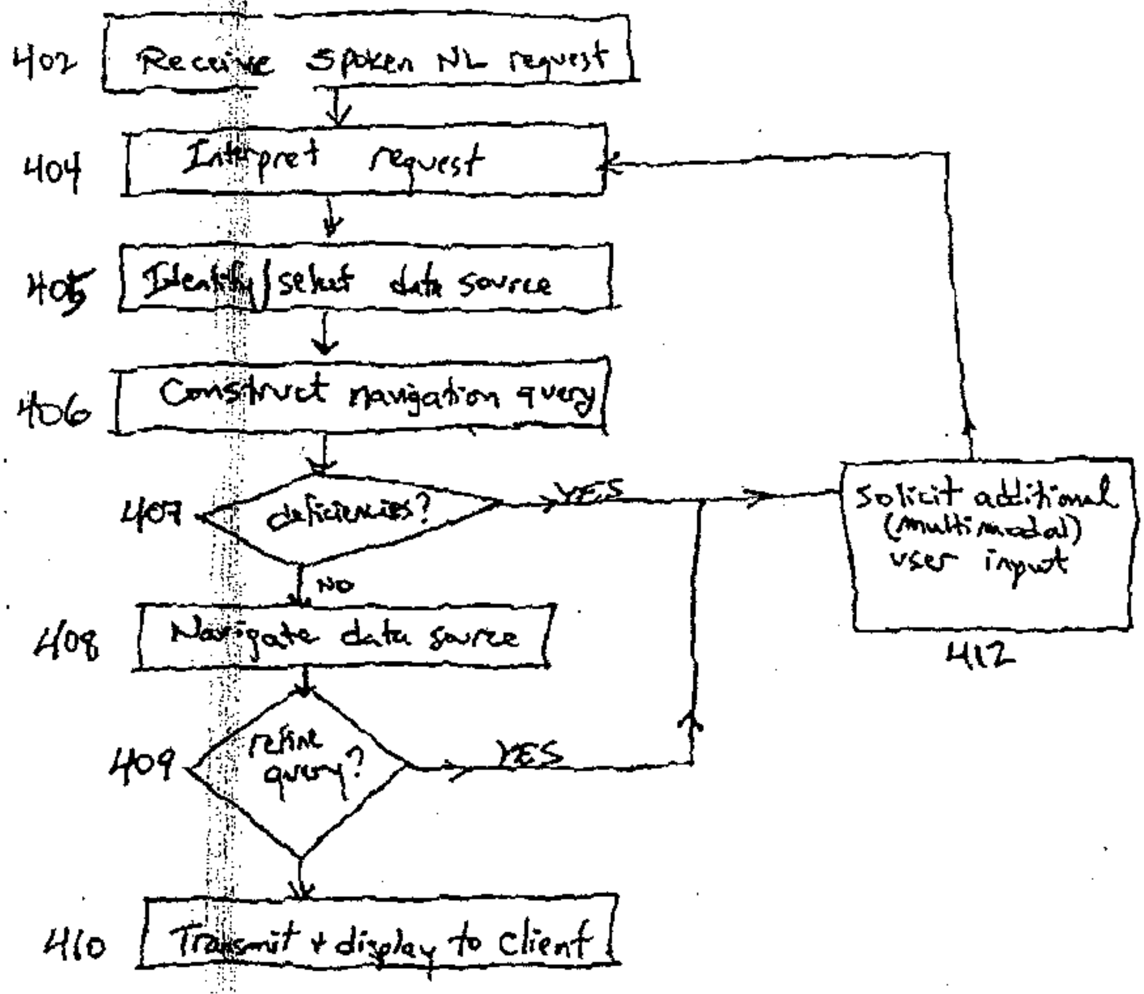

402
Receive spoken NL request
404
Interpret request
405
Identify/select data source
406
Construct navigation query
407
deficiencies?
YES
NO
Solicit additional (multimodal) user input
412
408
Navigate data source
409
refine query?
YES
410
Transmit + display to client

Fig. 4

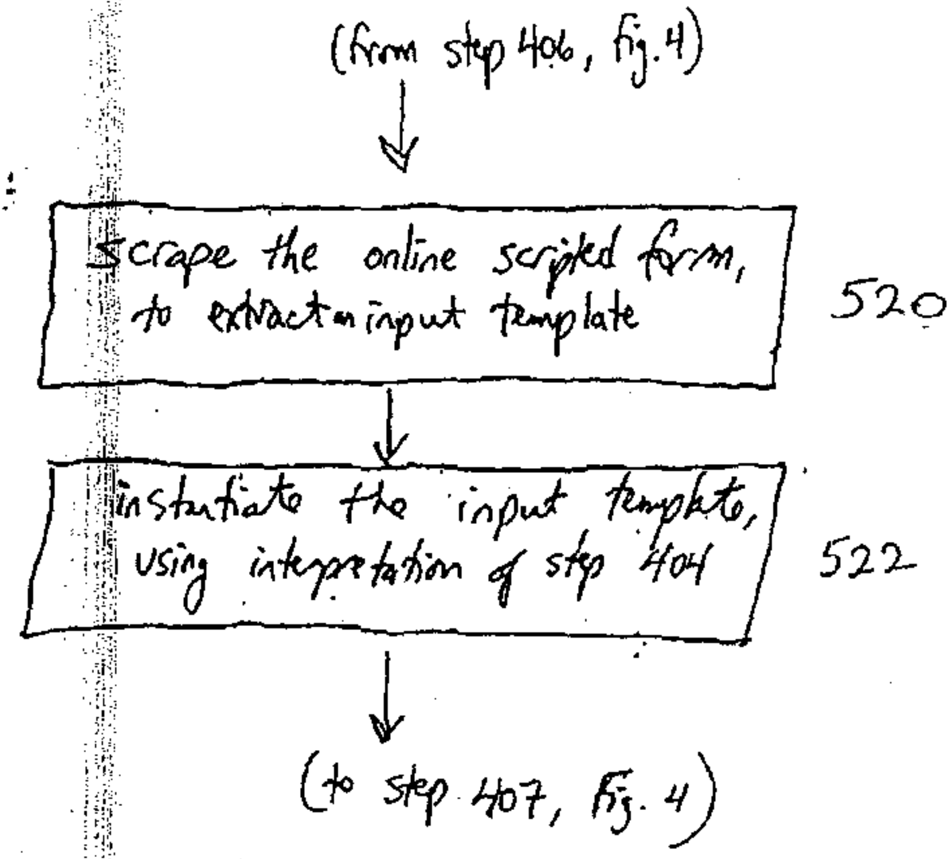

(from step 406, fig. 4)
Scrape the online scripted form, to extract an input template
520
instantiate the input template, using interpretation of step 404
522
(to step 407, Fig. 4)

Fig. 5

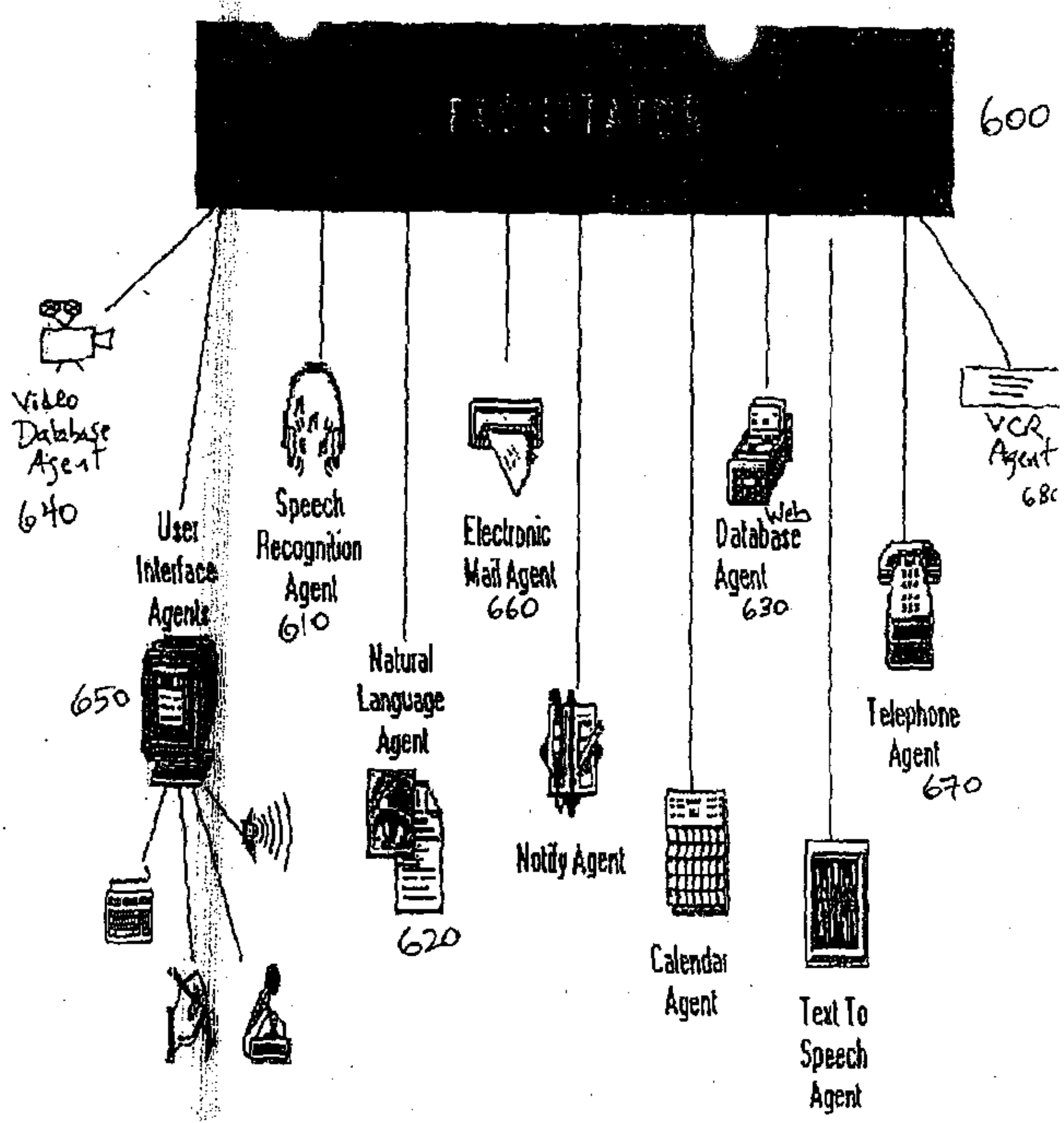

FACILITATOR
600
Video Database Agent
640
User Interface Agents
650
Speech Recognition Agent
610
Natural Language Agent
620
Electronic Mail Agent
660
Notify Agent
Calendar Agent
Web Database Agent
630
Text To Speech Agent
Telephone Agent
670
VCR Agent
68

Fig. 6

PRINT OF DRAWINGS
AS ORIGINALLY FILEI

2

Post-it® Fax Note 7671	Date 6/29/01	# of pages ▶ 7
To Domenic Rutab	From K. Eloisni	
Co./Dept.	Co. SRI Intl.	
Phone # 408-971-4660	Phone # 650-859-6631	
Fax # 408-971-4460	Fax # 650.859-6420	

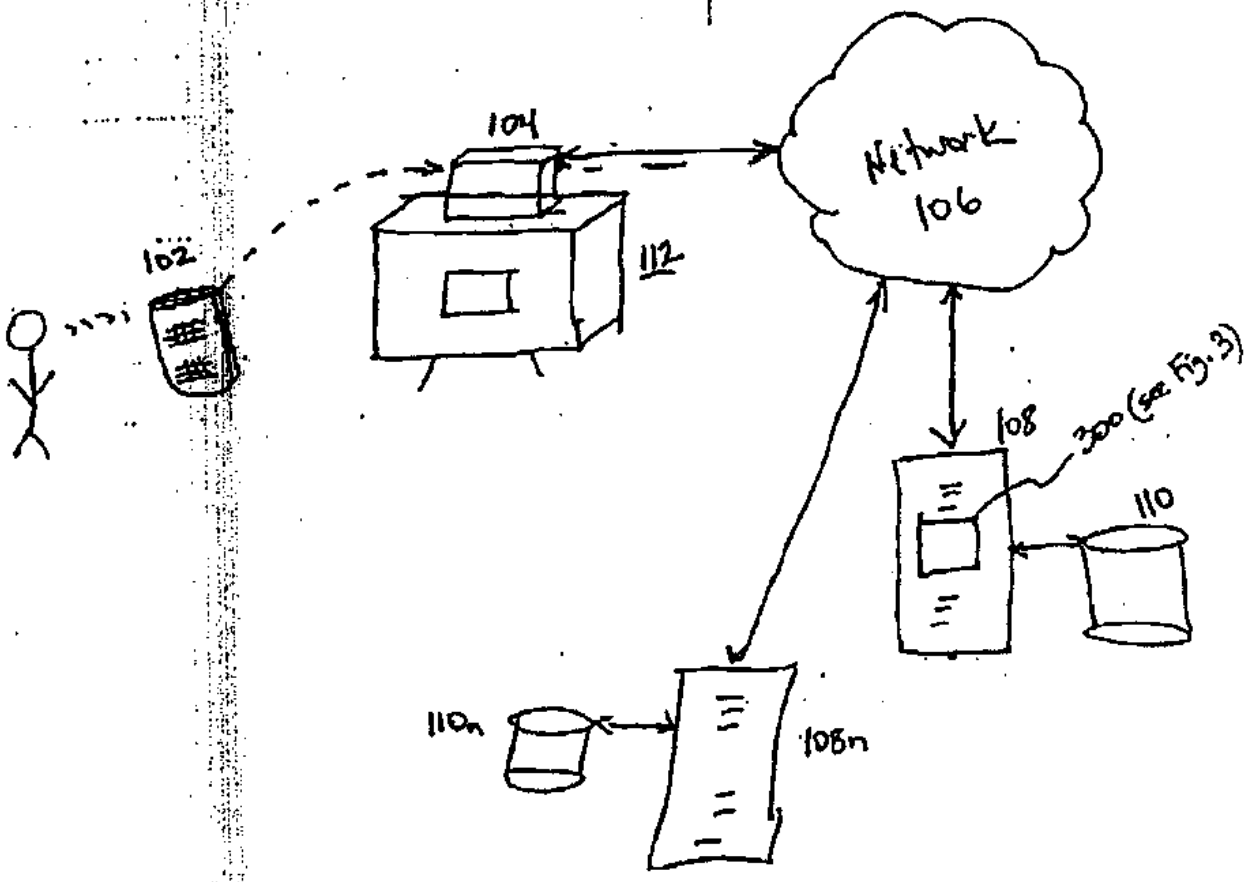


Fig. 1a

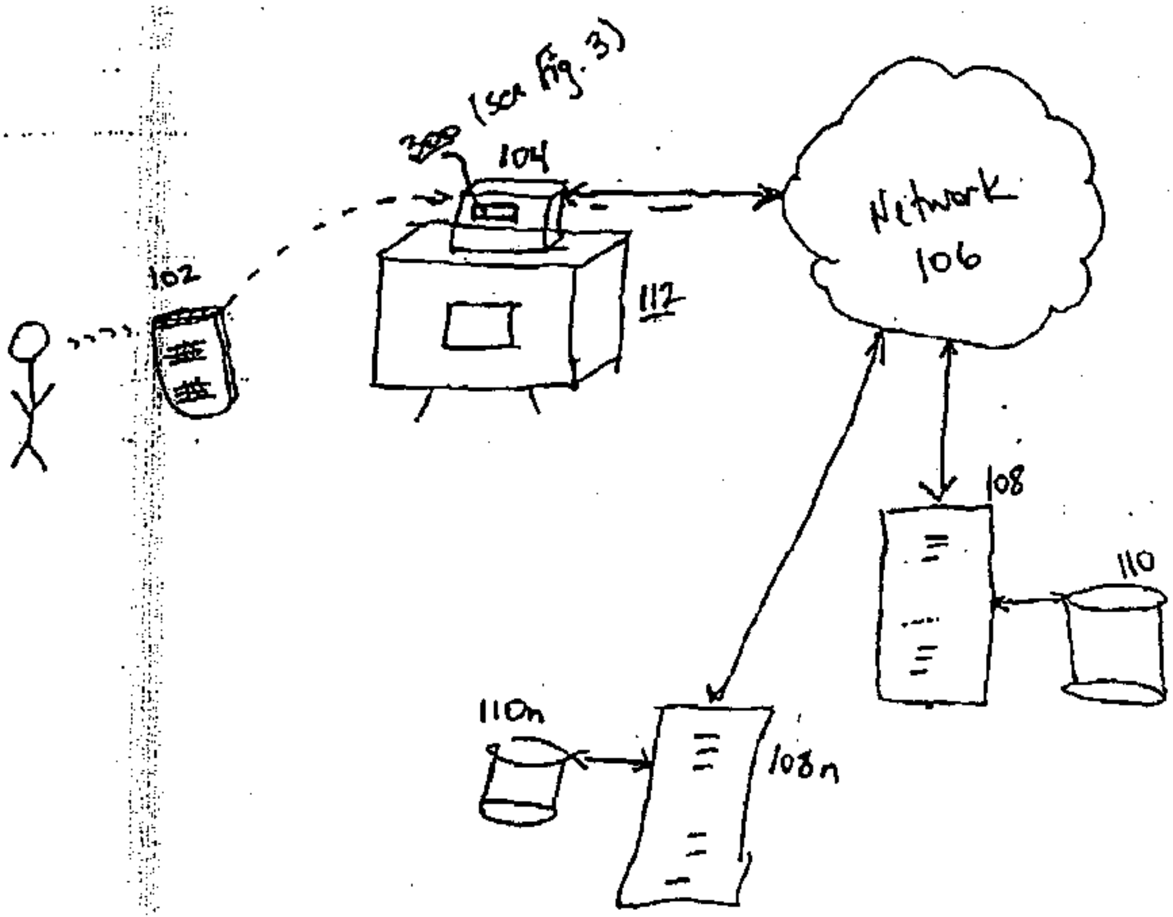


Fig. 1b

PRINT OF DRAWINGS
AS ORIGINALLY FILED

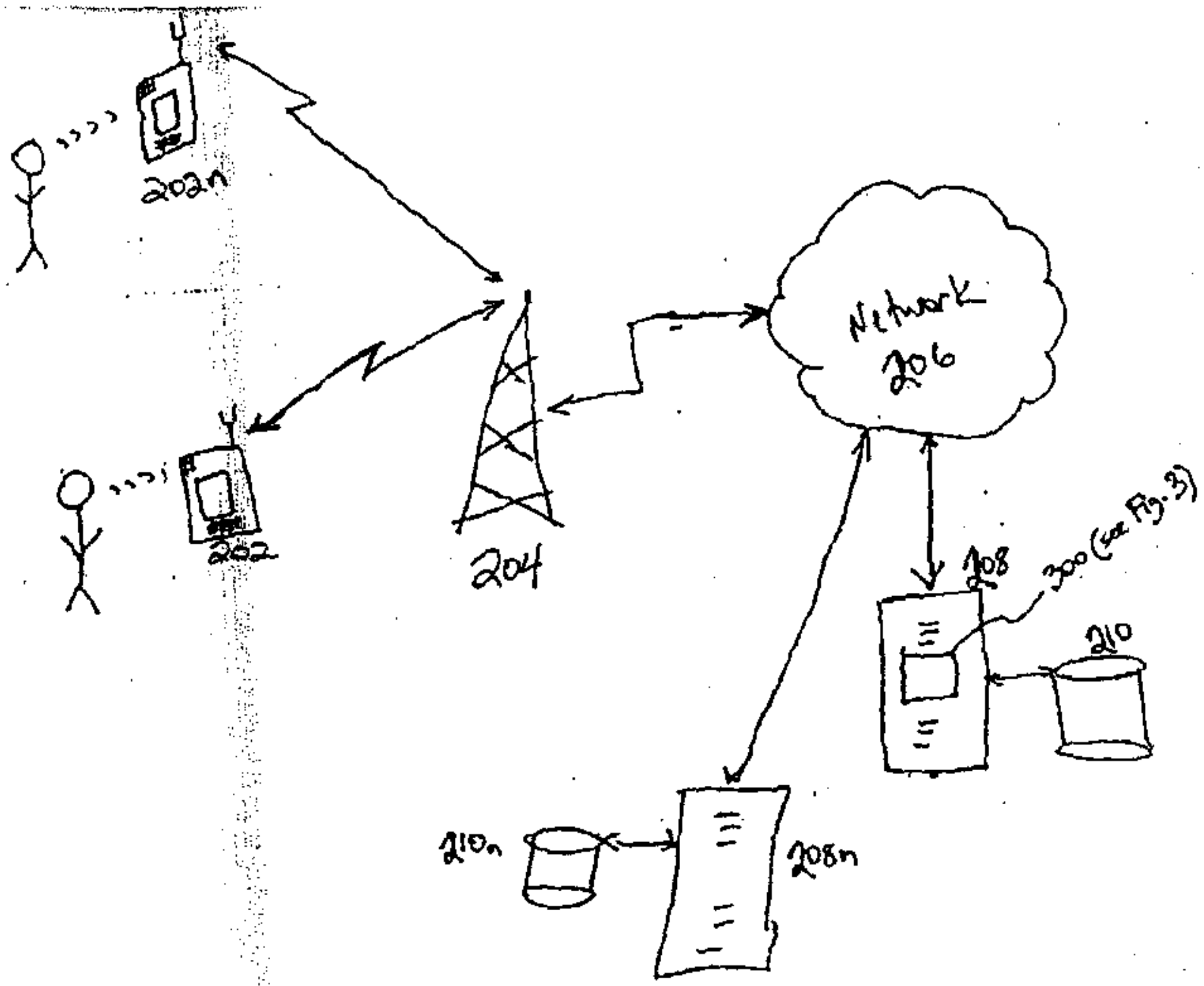


Fig. 2

PRINT OF DRAWINGS
AS ORIGINALLY FILED

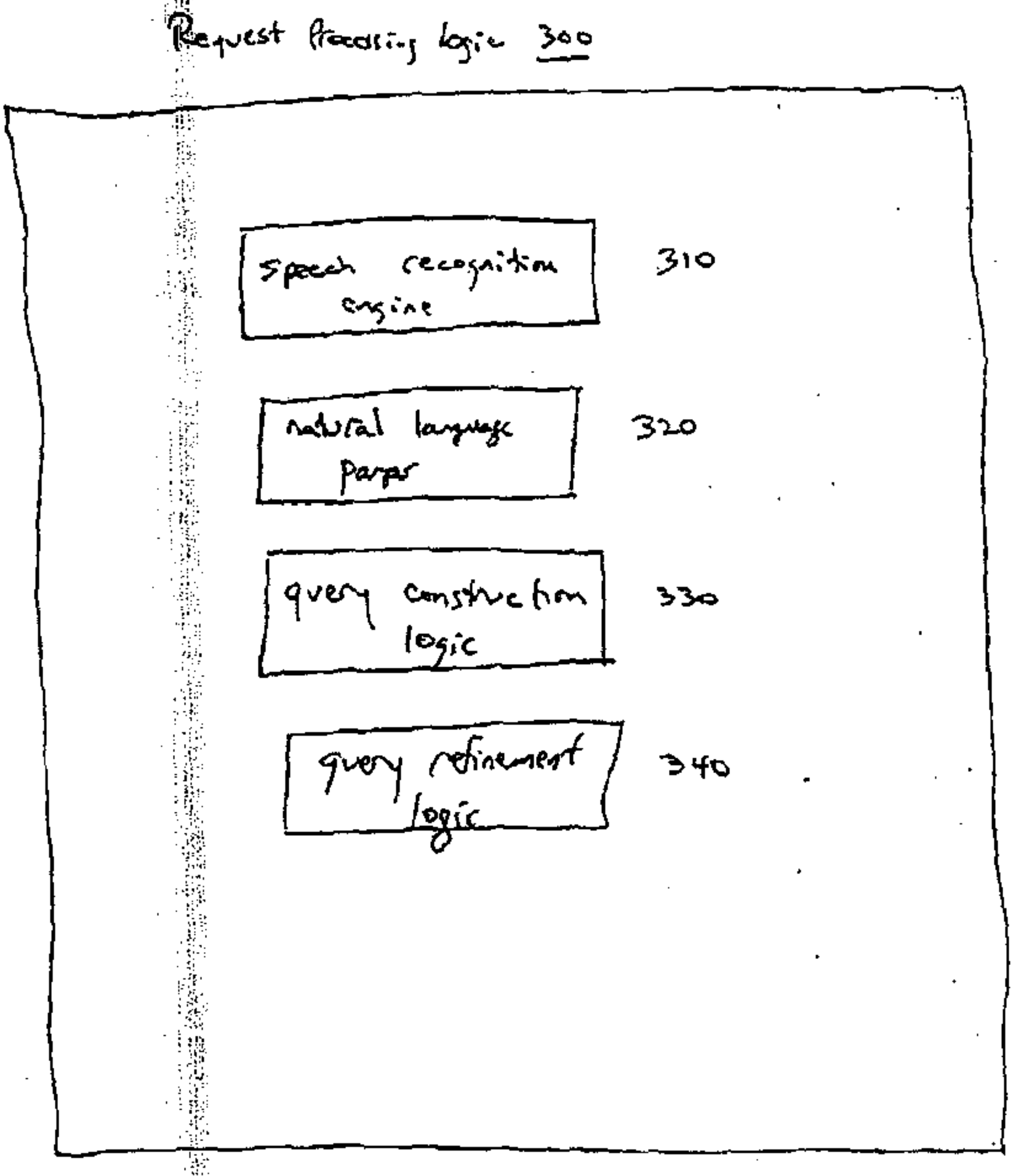


Fig. 3

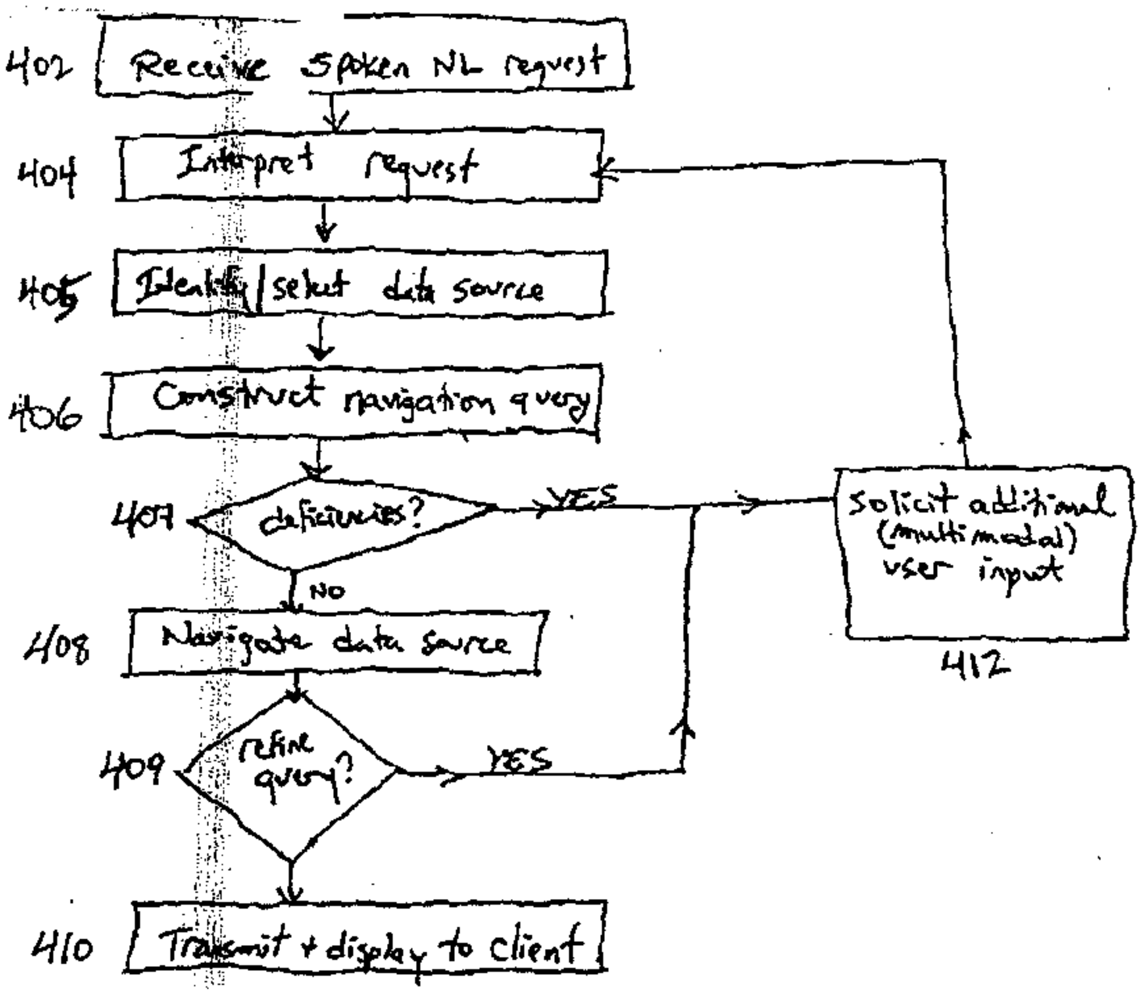

402
Receive spoken NL request
404
Interpret request
405
Identify/select data source
406
Construct navigation query
407
deficiencies?
YES
NO
Solicit additional (multimodal) user input
412
408
Navigate data source
409
refine query?
YES
410
Transmit + display to client

Fig. 4

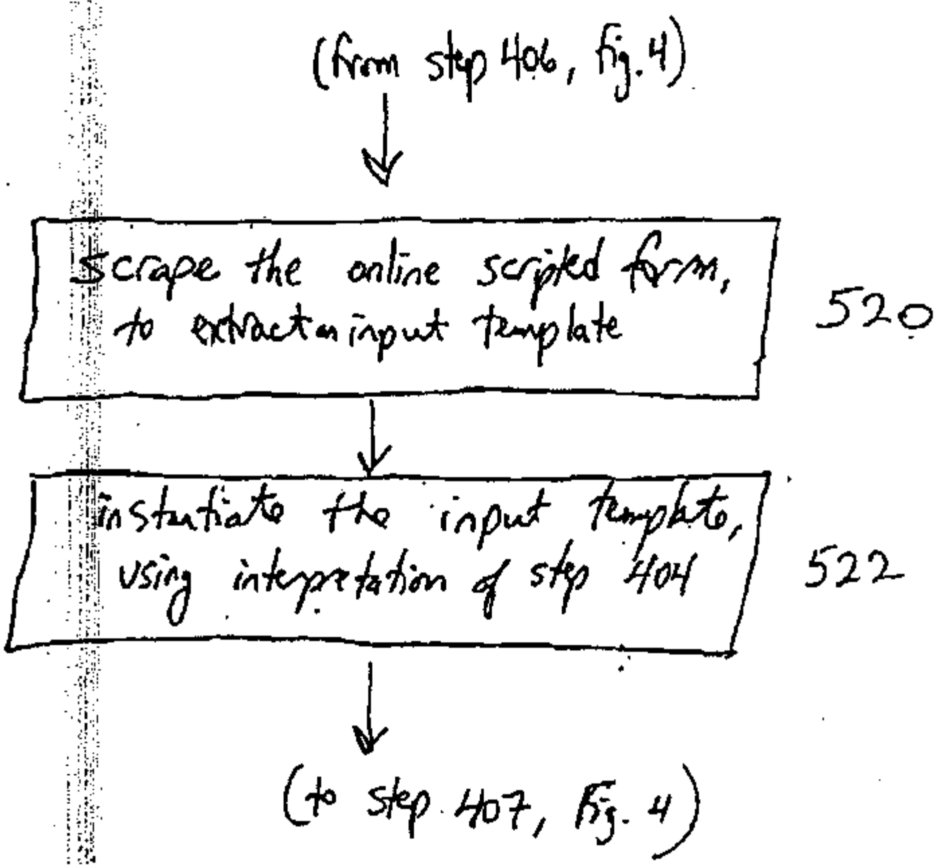

(from step 406, fig. 4)
Scrape the online scripted form, to extract an input template
520
instantiate the input template, using interpretation of step 404
522
(to step 407, Fig. 4)

Fig. 5

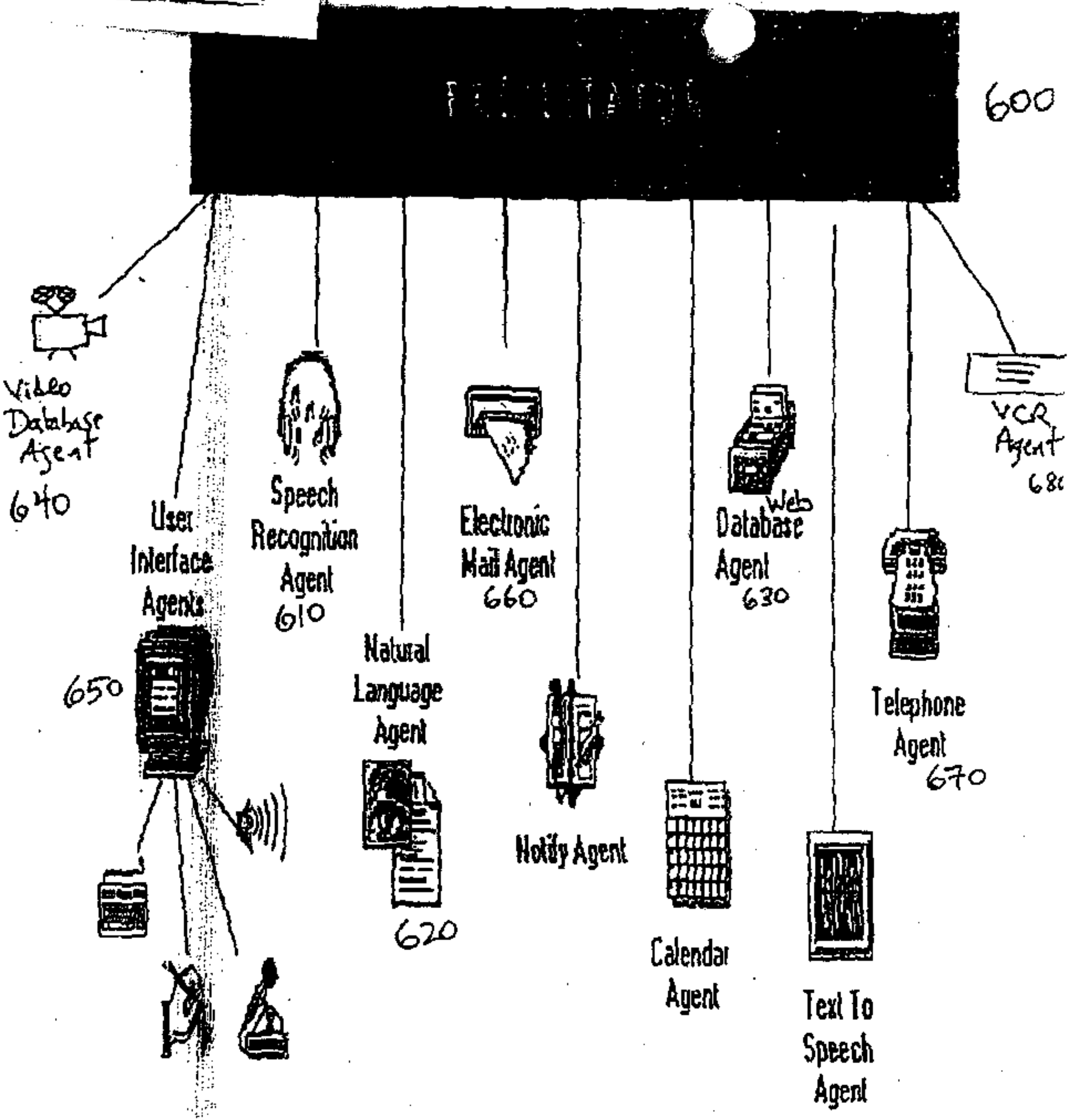

600
Video Database Agent
640
User Interface Agents
650
Speech Recognition Agent
610
Natural Language Agent
620
Electronic Mail Agent
660
Notify Agent
Calendar Agent
Web Database Agent
630
Text To Speech Agent
Telephone Agent
670
VCR Agent
68

Fig. 6

PATENT APPLICATION SERIAL NO. ____________________

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
FEE RECORD SHEET

```
07/12/2000 PALLEN   00000020 09608872

01 FC:201                      345.00 OP
02 FC:203                       63.00 OP
```

PTO-1556
(5/87)

*U.S. GPO: 1999-459-082/19144

UNITED STATES PATENT AND TRADEMARK OFFICE

COMMISSIONER FOR PATENTS
UNITED STATES PATENT AND TRADEMARK OFFICE
WASHINGTON, D.C. 20231
www.uspto.gov

APPLICATION NUMBER	FILING/RECEIPT DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NUMBER
09/608,872	06/30/2000	Christine Halversen	SR1Ip037B

Kevin J Zilka
P O Box 721030
San Jose, CA 95172-1030

FORMALITIES LETTER

OC000000005370740

Date Mailed: 09/01/2000

NOTICE TO FILE MISSING PARTS OF NONPROVISIONAL APPLICATION

FILED UNDER 37 CFR 1.53(b)

Filing Date Granted

An application number and filing date have been accorded to this application. The item(s) indicated below, however, are missing. Applicant is given TWO MONTHS from the date of this Notice within which to file all required items and pay any fees required below to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

- The oath or declaration is missing.
  A properly signed oath or declaration in compliance with 37 CFR 1.63, identifying the application by the above Application Number and Filing Date, is required.
- To avoid abandonment, a late filing fee or oath or declaration surcharge as set forth in 37 CFR 1.16(e) of $65 for a small entity in compliance with 37 CFR 1.27, must be submitted with the missing items identified in this letter.

- **The balance due by applicant is $ 65.**

A copy of this notice MUST be returned with the reply.

M. Middleton
Customer Service Center
Initial Patent Examination Division (703) 308-1202
PART 3 - OFFICE COPY

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the application of

Halverson et al.

Application No. 09/608,872

Filed: June 30, 2000

For: MOBILE NAVIGATION OF NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN INPUT

Examiner: Not Assigned

Art Unit: 2741

Atty. Docket No. SRI1P037B

Date: October 30, 2000

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231 on October 30, 2000.

Signed: Julie
Julie A. Curts

RESPONSE TO NOTICE TO FILE MISSING PARTS

Assistant Commissioner for Patents
Box: Missing Parts
Washington, D.C. 20231

Sir:

In response to the Notice to File Missing Parts of Application--Filing Date Granted dated September 1, 2000, Applicants hereby attach an original executed Declaration and Power of Attorney, and the copy of the Notice to be returned with this response.

Applicants are also attaching Check No. 238 for $65.00 in payment of the surcharge fee. The Commissioner is authorized to charge any other fees that may be due to our Deposit Account No. 50-1351 (Order No. SRI1P037B). A copy of this sheet is enclosed for this purpose.

Respectfully submitted,
SILICON VALLEY IP LAW GROUP

Kevin J. Zilka
Reg. No. 41,429

P.O. Box 721030
San Jose, CA 95172-1030
(408) 505-5100

DECLARATION AND POWER OF ATTORNEY FOR ORIGINAL U.S. PATENT APPLICATION

Attorney's Docket No. SRI1P037

As a below-named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe that I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled: NAVIGATING NETWORK-BASED ELECTRONIC INFORMATIN USING SPOKEN NATURAL LANGUAGE INPUT WITH MULTIMODAL ERROR FEEDBACK, the specification of which,

(check one) 1. □ is attached hereto.

2. ☒ was filed on March 13, 2000 as
U.S. Application Serial No. 09/524,095
and was amended on ______________.

3. □ was filed on ______________ as
International PCT Application Serial No. ______________
and was amended on ______________.

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, CFR § 1.56.

I hereby claim foreign priority benefits under Title 35, United States code, § 119(a)-(d) or § 365(b) of any foreign application(s) for patent or inventor's certificate, or § 365(a) of any PCT International application which designated at least one country other than the United States, listed below and have identified below, by checking the box, any foreign application for patent or inventor's certificate, or PCT International application having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)

			Priority Benefits Claimed?
(Appl. No.)	(Country)	(Filing Date)	□ Yes □ No
(Appl. No.)	(Country)	(Filing Date)	□ Yes □ No
(Appl. No.)	(Country)	(Filing Date)	□ Yes □ No

I hereby claim the benefit under 35 U.S.C. §119(e) of any United States provisional application(s) listed below:

______________ (Application Serial No.) ______________ (Filing Date)

______________ (Application Serial No.) ______________ (Filing Date)

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s), or § 365(c) of any PCT International application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

Prior U.S. Application(s)

(Application Serial No.) (Filing Date) (Status - patented, pending, abandoned)

(Application Serial No.) (Filing Date) (Status - patented, pending, abandoned)

And I hereby appoint the law firm of Hickman Stephens Coleman & Hughes, including Paul L. Hickman (Reg. No. 28,516); L. Keith Stephens (Reg. No. 32,632); Brian R. Coleman (Reg. No. 39,145); Michael J. Hughes (Reg. No. 29,077); Michael E. Melton (Reg. No. 32,276); Raymond E. Roberts (Reg. No. 38,597); Vidya R. Bhakar (Reg. No. 42,323); Larry B. Guernsey (Reg. No. 40,008); Douglas E. Mackenzie (Reg. No. 38,955); Michael D. Plimier (Reg. No. 43,004); Ronald B. Feece (Reg. No. P46,327); Stefanie M. Howell (Reg. No. P45,929); and Robert D. Hayden (Reg. No. 42,645) as my principal attorneys to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith:

Send Correspondence To:

HICKMAN STEPHENS COLEMAN & HUGHES, LLP
P.O. BOX 52037
Palo Alto, California 94303-0746

Direct Telephone Calls To: Raymond E. Roberts at telephone number (408) 558-9950

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

Typewritten Full Name of Sole or First Inventor: Christine Halverson Citizenship: USA

Inventor's signature: Christine Halverson Date of Signature: 6-16-00

Residence: (City) San Jose (State/Country) California/USA

Post Office Address: 1623 Fairorchard Avenue, San Jose, California 95125

Full Name of Second Joint Inventor (if any): Luc Julia Citizenship: ~~USA~~ FRANCE

Inventor's signature: Date of Signature: 6.21.00

Residence: (City) Menlo Park (State/Country) California/USA

Post Office Address: 607 Menlo Avenue, Menlo Park, California 94025

Full Name of Third Joint Inventor (if any): Dimitris Voutsas Citizenship: Greece

Inventor's signature: Date of Signature: 6/16/00

Residence: (City) Thessaloniki (State/Country) Greece

Post Office Address: 14 M. Pyrza Street, Neoi Epivates, Thessaloniki 57019, Greece

Attny Docket No. SRI1P037 Page 2 of 3

Full Name of Fourth Joint Inventor (if any): Adam Cheyer Citizenship: USA

Inventor's signature: Date of Signature: 6/22/00

Residence: (City) Palo Alto (State/Country) California /USA

Post Office Address: 757 Cereza Drive, Palo Alto, California 94306

SECTOR 8

#3

UNITED STATES PATENT AND TRADEMARK OFFICE

COMMISSIONER FOR PATENTS
UNITED STATES PATENT AND TRADEMARK OFFICE
WASHINGTON, D.C. 20231
www.uspto.gov

APPLICATION NUMBER	FILING/RECEIPT DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NUMBER
09/608,872	06/30/2000	Christine Halversen	SRIIp037B

Kevin J Zilka
P O Box 721030
San Jose, CA 95172-1030

FORMALITIES LETTER

OC000000005370740

Date Mailed: 09/01/2000

NOTICE TO FILE MISSING PARTS OF NONPROVISIONAL APPLICATION

FILED UNDER 37 CFR 1.53(b)

Filing Date Granted

An application number and filing date have been accorded to this application. The item(s) indicated below, however, are missing. Applicant is given TWO MONTHS from the date of this Notice within which to file all required items and pay any fees required below to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

- The oath or declaration is missing.
  A properly signed oath or declaration in compliance with 37 CFR 1.63, identifying the application by the above Application Number and Filing Date, is required.
- To avoid abandonment, a late filing fee or oath or declaration surcharge as set forth in 37 CFR 1.16(e) of $65 for a small entity in compliance with 37 CFR 1.27, must be submitted with the missing items identified in this letter.

- **The balance due by applicant is $ 65.**

A copy of this notice MUST be returned with the reply.

M. Middleton
Customer Service Center
Initial Patent Examination Division (703) 308-1202
PART 2 - COPY TO BE RETURNED WITH RESPONSE

07-03-00

A

#4/A
LDT
4-2001
entered

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF EXPRESS MAILING

I hereby certify that this paper and the documents and/or fees referred to as attached therein are being deposited with the United States Postal Service on June 30, 2000 in an envelope as "Express Mail Post Office to Addressee" service under 37 CFR §1.10, Mailing Label Number EK858788212US, addressed to the Assistant Commissioner for Patents, Washington, DC 20231.

Kevin J. Zilka

Attorney Docket No.: SRI1P037B

First Named Inventor:

HALVERSEN, Christine

UTILITY PATENT APPLICATION TRANSMITTAL (37 CFR. § 1.53(b))

(Continuation, Divisional or Continuation-in-part application)

Assistant Commissioner for Patents
Box Patent Application
Washington, DC 20231

□ Duplicate for fee processing

Sir: This is a request for filing a patent application under 37 CFR. § 1.53(b) in the name of inventors: Christine Halversen, Luc Julia, Dimitris Voutsas, Adam Cheyer

For: MOBILE NAVIGATION OF NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN INPUT

This application is a ☒ Continuation □ Divisional □ Continuation-in-part

of prior Application No.: **09/524,095**, from which priority under 35 U.S.C. §120 is claimed.

Application Elements:

☒ 33 Pages of Specification, Claims and Abstract

☒ 07 Sheets of Drawings

□ Declaration

□ Newly executed (original or copy)

□ Copy from a prior application (37 CFR 1.63(d) for a continuation or divisional). The entire disclosure of the prior application from which a copy of the declaration is herein supplied is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.

□ Deletion of inventors Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b).

Accompanying Application Parts:

□ Assignment and Assignment Recordation Cover Sheet (recording fee of $40.00 enclosed)

□ Power of Attorney

□ 37 CFR 3.73(b) Statement by Assignee

□ Information Disclo: ꞉tatement with Form PTO-1449 □ pies of IDS Citations

☒ Preliminary Amendment

☒ Return Receipt Postcard

□ Small Entity Statement(s) ☒ Statement filed in prior application. Status still proper and desired.

□ Other:

Claim For Foreign Priority

□ Priority of ________ Application No. __________ filed on __________ is claimed under 35 U.S.C. § 119.

□ The certified copy has been filed in prior application U.S. Application No. _______

□ The certified copy will follow.

Extension of Time for Prior Pending Application

□ A Petition for Extension of Time is being concurrently filed in the prior pending application. A copy of the Petition for Extension of Time is attached.

Amendments

□ Amend the specification by inserting before the first line the sentence: "This is a

□ Continuation □ Continuation-in-part □ Divisional

application of copending prior

□ Application No._________ filed on _____________,

□ International Application _________ filed on _________ which designated the United States,

the disclosure of which is incorporated herein by reference."

☒ Cancel in this application original claims 2-55 of the prior application before calculating the filing fee. (*At least one original independent claim must be retained.*)

Fee Calculation (37 CFR § 1.16)

	(Col. 1)	(Col. 2)	SMALL ENTITY		OR	LARGE ENTITY	
	NO. FILED	NO. EXTRA	RATE	FEE		RATE	FEE
BASIC FEE			$345	$ 345	OR	$690	$
TOTAL CLAIMS	27 -20 =	7	x09 =	$ 63	OR	x18 =	$
INDEP CLAIMS	3 -03 =	0	x39 =	$	OR	x78 =	$
[ ] Multiple Dependent Claim Presented			$130 =	$	OR	$260 =	$
* If the difference in Col. 1 is less than zero, enter "0" in Col. 2.			Total	$ 408	OR	Total	$

☒ Check No. 137 in the amount of $ 408.00 is enclosed.

☒ The Commissioner is autho to charge any fees beyond the amount sed which may be required, or to credit any overpayment, to Deposit Account No. 50-1351 (Order No. SRJ1P037B).

General Authorization for Petition for Extension of Time (37 CFR §1.136)

☒ Applicants hereby make and generally authorize any Petitions for Extensions of Time as may be needed for any subsequent filings. The Commissioner is also authorized to charge any extension fees under 37 CFR §1.17 as may be needed to Deposit Account No. 50-1351 (Order No. SRJ1P037B).

☒ Please send correspondence to the following address:

Kevin J. Zilka
P.O. BOX 721030
San Jose, California 95172-1030

Direct Telephone Calls To: **Kevin J. Zilka at telephone number (408) 505-5100**

Date: June 30, 2000

Kevin J. Zilka
Registration No. 41,429

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

#5/B
LDJ
4-20-01
entered

In re the application of)
Christine HALVERSEN et al.) Docket: SRI1P037B
Application No. 09/~~524,095~~ 608,872)
Filed: March 13, 2000) Date: June 30, 2000
For: NAVIGATING NETWORK BASED ELECTRONIC INFORMATION USING SPOKEN NATURAL LANGUAGE INPUT WITH MULTIMODAL ERROR FEEDBACK)

Preliminary Amendment

Assistant Commissioner for Patents
and Trademarks
Washington, DC 20231

Dear Sir:

In regard to the above-named patent application, please enter the following amendments.

IN THE TITLE: ✓

Please delete "NAVIGATING NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN NATURAL LANGUAGE INPUT WITH MULTIMODAL ERROR FEEDBACK", and insert therefore, --MOBILE NAVIGATION OF NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN INPUT--.

IN THE ABSTRACT: ✓

Please delete the Abstract and insert therefore --A system, method, and article of manufacture are provided for navigating an electronic data source by means of spoken language where a portion of the data link between a mobile information appliance of the user and the data

27

B

source utilizes wireless communication. When a spoken input request is received from a user who is using the mobile information appliance, it is interpreted. The resulting interpretation of the request is thereupon used to automatically construct an operational navigation query to retrieve the desired information from one or more electronic network data sources, which is transmitted to the mobile information appliance.--

B1 end

IN THE SPECIFICATION:

On page 1, line 5, please delete "This is" and insert therefore, --This application is a continuation of an application entitled NAVIGATING NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN NATURAL LANGUAGE INPUT WITH MULTIMODAL ERROR FEEDBACK which was filed on March 13, 2000 under serial number 09/524,095 and which is--.

B2

Please delete page 3, lines 3 to 32, and insert therefore, --The present invention addresses the above needs by providing a system, method, and article of manufacture for mobile navigation of network-based electronic data sources in response to spoken input requests. When a spoken input request is received from a user using a mobile information appliance that communicates with a network server via an at least partially wireless communications system, it is interpreted, such as by using a speech recognition engine to extract speech data from acoustic voice signals, and using a language parser to linguistically parse the speech data. The interpretation of the spoken request can be performed on a computing device locally with the user, such as the mobile information appliance, or remotely from the user. The resulting interpretation of the request is thereupon used to automatically construct an operational navigation query to retrieve the desired information from one or more electronic network data sources, which is then transmitted to a client device of the user. If the network data source is a database, the navigation query is constructed in the format of a database query language.

B3

Typically, errors or ambiguities emerge in the interpretation of the spoken request, such that the system cannot instantiate a complete, valid navigational template. This is to be expected occasionally, and one preferred aspect of the invention is the ability to handle such errors and ambiguities in relatively graceful and user-friendly manner. Instead of simply rejecting such input and defaulting to traditional input modes or simply asking the user to try again, a preferred embodiment of the present invention seeks to converge rapidly toward instantiation of a valid navigational template by soliciting additional clarification from the user as necessary, either before or after a navigation of the data source, via multimodal input, i.e., by means of menu selection or other input modalities including and in addition to spoken input. This clarifying, multi-modal dialogue takes advantage of whatever partial navigational information has been gleaned from the initial interpretation of the user's spoken request. This clarification process continues until the system converges toward an adequately instantiated navigational template, which is in turn used to navigate the network-based data and retrieve the user's desired information. The retrieved information is transmitted across the network and presented to the user on a suitable client display device.

B3 end

IN THE CLAIMS:

Please delete claims 1-55, and insert therefore the following claims 56 82 ~~1-27~~:

09524095 .032800

B4

Sub C1

56 ~~1.~~ ~~(New) A~~ method for speech-based navigation of an electronic ~~data source located~~ at one or more network servers located remotely from a user, wherein at least a portion of a data link between a mobile information appliance of the user and the one or more network servers utilizes wireless communication, comprising the steps of:

(a) receiving a spoken request for desired information from the user utilizing the mobile information appliance of the user;

(b) rendering an interpretation of the spoken request;

(c) constructing a navigation query based upon the interpretation;

(d) utilizing the navigation query to select a portion of the electronic data source; and

(e) transmitting the selected portion of the electronic data source from the network ~~server to the mobile information appliance of the user.~~

2 ~~57~~ ~~2.~~ (New) The method of claim 1 ~~56~~ ~~1~~, wherein the step of rendering the interpretation of the spoken request is performed at the one or more network servers.

3 ~~58~~ ~~3.~~ (New) The method of claim 1 ~~56~~ ~~1~~, wherein the step of rendering the interpretation of the spoken request is performed by the mobile information appliance.

4 ~~59~~ ~~4.~~ (New) The method of claim 1 ~~56~~ ~~1~~, further comprising the steps of soliciting additional input from the user, including user interaction in a modality different than the original request;

29

B

refining the navigation query, based upon the additional input; and using the refined navigation query to select a portion of the electronic data source.

5 ~~60~~ ~~5~~. (New) The method of claim 1 ~~56~~ ~~1~~, wherein the data link includes a cellular telephone system.

6 ~~61~~ ~~6~~. (New) The method of claim 1 ~~56~~ ~~1~~, wherein steps (a)-(d) are performed with respect to multiple users.

7 ~~62~~ ~~7~~. (New) The method of claim 1 ~~56~~ ~~1~~, wherein the mobile information appliance is a wireless telephone.

8 ~~63~~ ~~8~~. (New) The method of claim 1 ~~56~~ ~~1~~, wherein the mobile information appliance is a portable computing device.

9 ~~64~~ ~~9~~. (New) The method of claim 8 ~~63~~ ~~8~~, wherein the portable computing device is a personal digital assistant.

B4

Sub C2 ~~65~~ ~~10~~. ~~(New) A computer program embodied on~~ a computer ~~readable medium for~~ speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, wherein at least a portion of a data link between a mobile information appliance of the user and the one or more network servers utilizes wireless communication, comprising:

(a) a code segment that receives a spoken request for desired information from the user utilizing the mobile information appliance of the user;

(b) a code segment that renders an interpretation of the spoken request;

(c) a code segment that constructs a navigation query based upon the interpretation;

(d) a code segment that utilizes the navigation query to select a portion of the electronic data source; and

(e) a code segment that transmits the selected portion of the electronic data source from the network server to the mobile information appliance of the user.

30

B

Rule 1.126

11 ~~66~~ ~~11~~. (New) The computer program of claim 10 ~~65~~ ~~10~~, wherein the rendering of the interpretation of the spoken request is performed at the one or more network servers.

12 ~~67~~ ~~12~~. (New) The computer program of claim 10 ~~65~~ ~~10~~, wherein the rendering of the interpretation of the spoken request is performed by the mobile information appliance.

13 ~~68~~ ~~13~~. (New) The computer program of claim 10 ~~65~~ ~~10~~, further comprising a code segment that solicits additional input from the user, including user interaction in a modality different than the original request; a code segment that refines the navigation query, based upon the additional input; and a code segment that uses the refined navigation query to select a portion of the electronic data source.

B4

14 ~~69~~ ~~14~~. (New) The computer program of claim 10 ~~65~~ ~~10~~, wherein the data link includes a wireless telephone system.

15 ~~70~~ ~~15~~. (New) The computer program of claim 10 ~~65~~ ~~10~~, wherein code segments (a)-(d) are executed with respect to multiple users.

16 ~~71~~ ~~16~~. (New) The computer program of claim 10 ~~65~~ ~~10~~, wherein the mobile information appliance is a wireless telephone.

17 ~~72~~ ~~17~~. (New) The computer program of claim 10 ~~65~~ ~~10~~, wherein the mobile information appliance is a portable computing device.

18 ~~73~~ ~~18~~. (New) The computer program of claim 17 ~~72~~ ~~17~~, wherein the portable computing device is a personal digital assistant.

74 ~~19~~. ~~(New)~~ A system for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, comprising:

(a) a mobile information appliance operable to receive a spoken request for desired information from the user;

(b) spoken language processing logic, operable to render an interpretation of the spoken request;

31

B

Amd B3

(c) query construction logic, operable to construct a navigation query based upon the interpretation;

(d) navigation logic, operable to select a portion of the electronic data source using the navigation query; and

(e) electronic communications infrastructure for transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user, wherein at least a portion of a data link of the electronic communications infrastructure between a mobile information appliance of the user and the one or more network servers utilizes wireless communication.

B4

20 ~~75~~ ~~20~~. (New) The system of claim 19 ~~74~~ ~~19~~, wherein the spoken language processing logic renders the interpretation of the spoken request at the one or more network servers.

21 ~~76~~ ~~21~~. (New) The system of claim 19 ~~74~~ ~~19~~, wherein the spoken language processing logic renders the interpretation of the spoken request at the mobile information appliance.

22 ~~77~~ ~~22~~. (New) The system of claim 19 ~~74~~ ~~19~~, further comprising user interaction logic operable to solicit additional input from the user, including user interaction in a modality different than the original request; and query refining logic operable to refine the navigation query based upon the additional input; wherein the navigation logic users the refined navigation query to select a portion of the electronic data source.

23 ~~78~~ ~~23~~. (New) The system of claim 19 ~~74~~ ~~19~~, wherein the data link includes a cellular telephone system.

24 ~~79~~ ~~24~~. (New) The system of claim 19 ~~74~~ ~~19~~, wherein the system operates with respect to multiple users.

25 ~~80~~ ~~25~~. (New) The system of claim 19 ~~74~~ ~~19~~, wherein the mobile information appliance is a wireless telephone.

26 ~~81~~ ~~26~~. (New) The system of claim 19 ~~74~~ ~~19~~, wherein the mobile information appliance is a portable computing device.

32

B

27 ~~82~~ ~~27~~. (New) The system of claim 26 ~~81~~ ~~26~~, wherein the portable computing device is a personal digital assistant.

B4 end

In the event a telephone conversation would expedite the prosecution of this application, the Examiner may reach the undersigned at (408) 505-5100. If any fees are due in connection with the filing of this paper, then the Commissioner is authorized to charge such fees to Deposit Account No. 50-1351 (Order No. SRI1P037B). A duplicate copy of the transmittal is enclosed for this purpose.

Respectfully submitted,

Kevin J. Zilka
Registration No. 41,429

P.O. Box 721030
San Jose, CA 95172
Telephone: (408) 505-5100

33

B

UNITED STATES PATENT AND TRADEMARK OFFICE

COMMISSIONER FOR PATENTS
UNITED STATES PATENT AND TRADEMARK OFFICE
WASHINGTON, D.C. 20231
www.uspto.gov

Bib Data Sheet

SERIAL NUMBER	FILING DATE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
09/608,872	06/30/2000 RULE –	704	2741	SRIlp037B

APPLICANTS

Christine Halversen, San Jose, CA ;
Luc Julia, Menlo Park`, CA ;
Dimitris Voutsas, Thessaloniki, GREECE;
Adam Cheyer, Palo Alto, CA ;

** CONTINUING DATA ************************* yes FJ

THIS APPLICATION IS A CON OF 09/524,095 03/13/2000
WHICH IS A CIP OF 09/225,198 01/05/1999
WHICH CLAIMS BENEFIT OF 60/124,718 03/17/1999
WHICH CLAIMS BENEFIT OF 60/124,719 03/17/1999
WHICH CLAIMS BENEFIT OF 60/124,720 03/17/1999

** FOREIGN APPLICATIONS ******************** None FJ

IF REQUIRED, FOREIGN FILING LICENSE GRANTED ** 08/31/2000

** SMALL ENTITY **

		STATE OR COUNTRY	SHEETS DRAWING	TOTAL CLAIMS	INDEPENDENT CLAIMS
Foreign Priority claimed 35 USC 119 (a-d) conditions met Verified and Acknowledged	□ yes ☒ no □ yes ☒ no □ Met after Allowance FJ Examiner's Signature FJ Initials	CA	7	27	3

ADDRESS

24277

TITLE

Mobile navigation of network-based electronic information using spoken input

FILING FEE RECEIVED 473	FEES: Authority has been given in Paper No. ________ to charge/credit DEPOSIT ACCOUNT No. ________ for following:	□ All Fees □ 1.16 Fees (Filing) □ 1.17 Fees (Processing Ext. of time) □ 1.18 Fees (Issue) □ Other ________ □ Credit

6

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR		ATTORNEY DOCKET NO.
09/608,872	06/30/00	HALVERSEN	C	SRILP037B

024277
Kevin J. Zilka
PO Box 721030
San Jose CA 95172

TM02/0424

EXAMINER	
BACKER,F	
ART UNIT	PAPER NUMBER
2155	6

DATE MAILED: 04/24/01

Please find below and/or attached an Office communication concerning this application or proceeding.

Commissioner of Patents and Trademarks

Office Action Summary	**Application No.** 09/608,872	**Applicant(s)** HALVERSEN ET AL.	
	Examiner Firmin Backer	**Art Unit** 2155	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136 (a). In no event, however, may a reply be timely filed *after SIX (6) MONTHS from the mailing date of this communication.*
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 30 June 2000 .

2a)□ This action is **FINAL.** 2b)☒ This action is non-final.

3)□ Since this application is in condition for allowance except for fomal matters, prosecution as to the merits is *closed in accordance with the practice under Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 56-82 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)□ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 56-82 is/are rejected.

7)□ Claim(s) _____ is/are objected to.

8)□ Claims _____ are subject to restriction and/or election requirement.

Application Papers

9)□ The specification is objected to by the Examiner.

10)□ The drawing(s) filed on _____ is/are objected to by the Examiner.

11)□ The proposed drawing correction filed on _____ is: a)□ approved b)□ disapproved.

12)□ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. § 119

13)□ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)□ All b)□ Some * c)□ None of:

1.□ Certified copies of the priority documents have been received.

2.□ Certified copies of the priority documents have been received in Application No. _____ .

3.□ Copies of the certified copies of the priority documents have been received in this National Stage *application from the International Bureau* (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)□ Acknowledgement is made of a claim for domestic priority under 35 U.S.C. § 119(e).

Attachment(s)

15) ☒ Notice of References Cited (PTO-892)

16) □ Notice of Draftsperson's Patent Drawing Review (PTO-948)

17) □ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____ .

18) □ Interview Summary (PTO-413) Paper No(s). _____ .

19) □ Notice of Informal Patent Application (PTO-152)

20) □ Other:

DETAILED ACTION

This is in response to a letter for patent filed on June 30th, 2000 in which claims 56-82 are presented for examination. Claims 56-82 are pending in the letter.

Double Patenting

1. A rejection based on double patenting of the "same invention" type finds its support in the language of 35 U.S.C. 101 which states that "whoever invents or discovers any new and useful process ... may obtain a patent therefor ..." (Emphasis added). Thus, the term "same invention," in this context, means an invention drawn to identical subject matter. See *Miller v. Eagle Mfg. Co.*, 151 U.S. 186 (1894); *In re Ockert*, 245 F.2d 467, 114 USPQ 330 (CCPA 1957); and *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970).

A statutory type (35 U.S.C. 101) double patenting rejection can be overcome by canceling or amending the conflicting claims so they are no longer coextensive in scope. The filing of a terminal disclaimer cannot overcome a double patenting rejection based upon 35 U.S.C. 101.

2. Claims 56-82 are provisionally rejected under 35 U.S.C. 101 as claiming the same invention as that of claims 56-126 of copending Application No. 09/524,095. Although the conflicting claims are not identical, they are not patentably distinct. It would have been obvious to one of ordinary skill in the art to observed that the omission of the limitations "***soliciting additional input from the user, including user interaction in a modality different that the original request and, refining the navigation query, based upon the additional input***", of applicant claims 56-82 are already in the Co-pending application 09/524,095, as such they are obvious variation of the inventive concept defined in claims 56-126 of the Co-pending application 09/524,095. See In re Karlson, 136USPQ 184 (CCPA 1963). This is a provisional double patenting rejection since the conflicting claims have not in fact been patented.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in a patent granted on an application for patent by another filed in the United States before the invention thereof by the applicant for patent, or on an international application by another who has fulfilled the requirements of paragraphs (1), (2), and (4) of section 371(c) of this title before the invention thereof by the applicant for patent.

4. Claims 56-82 are rejected under 35 U.S.C. 102(e) as being anticipated by Levin et al. (U.S. Patent No. 6,173,279).

5. As per claim 56, Levin et al teach a method for speech-based navigation (information server, 110) of an electronic data source located at one or more network servers located remotely from a user, wherein at least a portion of a data link between a mobile information appliance of the user and the one or more network servers utilizes wireless communication (see abstract, fig 1, column 3 lines 5-35), comprising receiving a spoken request (*receive a natural language query*) for desired information from the user (user) utilizing the mobile information appliance (PC, 102) of the user; rendering an interpretation (*creating a semantic representation*) of the spoken request, constructing a navigation (*generating search)* query based upon the interpretation; utilizing the navigation query to select a portion of the electronic data source; and transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user. (see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22)

6. As per claim 57, 58, 62-64, Levin et al teach a method of rendering the interpretation of the spoken request is performed at the one or more network servers by the mobile information appliance including a wireless telephone, a portable computer that is a personal digital assistance (see abstract, fig 1, column 3 lines 5-35).

7. As per claim 59, Levin et al teach a method of soliciting additional input from the user, including user interaction in a modality different than the original request; refining the navigation query, based upon the additional input; and using the refined navigation query to select a portion of the electronic data source (see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22).

8. As per claim 60, Levin et al teach a method wherein the data link includes a cellular telephone system (see fig 1, column 2 line 61-67).

9. As per claim 61, Levin et al teach a method wherein steps (a)-(d) are performed with respect to multiple users (see abstract, fig 1, column 3 lines 5-35).

10. As per claim 65, Levin et al teach a computer system for speech-based navigation (information server, 110) of an electronic data source located at one or more network servers located remotely from a user, wherein at least a portion of a data link between a mobile information appliance of the user and the one or more network servers utilizes wireless communication (see abstract, fig 1, column 3 lines 5-35), comprising a code segment receiving a

spoken request (*receive a natural language query*) for desired information from the user (user) utilizing the mobile information appliance (PC, 102) of the user; a code segment rendering an interpretation (*creating a semantic representation*) of the spoken request, a code segment constructing a navigation (*generating search)* query based upon the interpretation; a code segment utilizing the navigation query to select a portion of the electronic data source; and a code segment transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user. (see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22)

11. As per claim 66, 67, 71-73, Levin et al teach a system of rendering the interpretation of the spoken request is performed at the one or more network servers by the mobile information appliance including a wireless telephone, a portable computer that is a personal digital assistance (see abstract, fig 1, column 3 lines 5-35).

12. As per claim 68, Levin et al teach a system of soliciting additional input from the user, including user interaction in a modality different than the original request; refining the navigation query, based upon the additional input; and using the refined navigation query to select a portion of the electronic data source (see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22).

13. As per claim 69, Levin et al teach a system wherein the data link includes a cellular telephone system (see fig 1, column 2 line 61-67).

14. As per claim 70, Levin et al teach a system wherein steps (a)-(d) are performed with respect to multiple users (see abstract, fig 1, column 3 lines 5-35).

15. As per claim 74, Levin et al teach a system for speech-based navigation (information server, 110) of an electronic data source located at one or more network servers located remotely from a user, wherein at least a portion of a data link between a mobile information appliance of the user and the one or more network servers utilizes wireless communication (see abstract, fig 1, column 3 lines 5-35), comprising receiving a spoken request (*receive a natural language query*) for desired information from the user (user) utilizing the mobile information appliance (PC, 102) of the user; rendering an interpretation (*creating a semantic representation*) of the spoken request, constructing a navigation (*generating search)* query based upon the interpretation; utilizing the navigation query to select a portion of the electronic data source; and transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user. (see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22)

16. As per claim 75, 76, 80-81, Levin et al teach a method of rendering the interpretation of the spoken request is performed at the one or more network servers by the mobile information appliance including a wireless telephone, a portable computer that is a personal digital assistance (see abstract, fig 1, column 3 lines 5-35).

17. As per claim 77, Levin et al teach a system of soliciting additional input from the user, including user interaction in a modality different than the original request; refining the

navigation query, based upon the additional input; and using the refined navigation query to select a portion of the electronic data source (see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22).

18. As per claim 78, Levin et al teach a system wherein the data link includes a cellular telephone system (see fig 1, column 2 line 61-67).

19. As per claim 79, Levin et al teach a system wherein steps (a)-(d) are performed with respect to multiple users (see abstract, fig 1, column 3 lines 5-35).

Conclusion

20. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. (6,192,338).

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Firmin Backer whose telephone number is 703-305-0624. The examiner can normally be reached on Mon-Thu 8:30-6:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Sheikh Ayaz can be reached on 703-305-9648. The fax phone numbers for the organization where this application or proceeding is assigned are 703-305-3718 for regular communications and 703-305-5352 for After Final communications.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is 703-305-3900.

Firmin Backer
Firmin Backer
April 9, 2001

AYAZ SHEIKH
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

PAGE 1 OF 1

FORM PTO-892	U.S. DEPARTMENT OF COMMERCE PATENT AND TRADEMARK OFFICE	SERIAL NO. 09/608,872	GROUP ART UNIT 2155 ~~2781~~	ATTACHMENT TO PAPER NO. 10
NOTICE OF REFERENCES CITED		APPLICANT(S) HALVERSEN ET AL.		

U.S. PATENT DOCUMENTS

*		DOCUMENT NO.	DATE	NAME	CLASS	SUB-CLASS	FILING DATE
	A	6,192,338	2/2001	Haszto et al	704	257	
	B	6,173,279	1/2001	Levin et al.	707	5	
	C						
	D						
	E						
	F						
	G						
	H						
	I						
	J						
	K						

FOREIGN PATENT DOCUMENTS

*		DOCUMENT NO.	DATE	COUNTRY	NAME	CLASS	SUB-CLASS
	L						
	M						
	N						
	O						
	P						
	Q						

OTHER REFERENCES (Including Author, Title, Date, Pertinent Pages, Etc.)

	R	
	S	
	T	
	U	

EXAMINER Firmin Backer	DATE April 9, 2001	Form892ccs2106b

* A copy of this reference is not being furnished with this office action.
(See Manual of Patent Examining Procedure, section 707.05(a).)

2155

PATENT

#7
LDS
5-8-01

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the application of:	
Halverson et al.	Group Art Unit: 2741
	Examiner: Unassigned
Application No. 09/608,872	
	Atty. Docket No. SRI1P037B/ 44454/03450
Filed: 06/30/2000	
For: MOBILE NAVIGATION OF NETWORK -BASED ELECTRONIC INFORMATION USING SPOKEN INPUT	Date: April 27, 2001

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, DC 20231 on April 27, 2001.

Signed: Erica L. Mann
Erica L. Mann

INFORMATION DISCLOSURE STATEMENT
UNDER 37 CFR §§ 1.56 AND 1.97(c)

Assistant Commissioner for Patents
Washington, DC 20231

Dear Sir:

The references listed in the attached PTO Form 1449, copies of which are attached, may be material to examination of the above-identified patent application. Applicants submit these references in compliance with their duty of disclosure pursuant to 37 CFR §§ 1.56 and 1.97. The Examiner is requested to make these references of official record in this application.

This Information Disclosure Statement is not to be construed as a representation that a search has been made, that additional information material to the examination of this application does not exist, or that these references indeed constitute prior art.

This Information Disclosure Statement is believed to be filed before the mailing date of a first Office Action on the merits. Accordingly, it is believed that no fees are due in connection with the filing of this Information Disclosure Statement. However, if it is determined that any fees are due, the Commissioner is hereby authorized to charge such fees to Deposit Account 03-0683 (Order No. 44454/03450/SRI1P037B).

Respectfully submitted,
CARLTON FIELDS

Dominic M. Kotab
Reg. No. 42,762

P.O. Box 721030
San Jose, CA 95172-1030
Telephone: (408) 271-2300

#7

Form 1449 (Modified) **Information Disclosure Statement By Applicant** (Use Several Sheets if Necessary)	Atty. Docket No. SRI1P037B Applicant: Halverson et al. Filing Date: 06/30/2000	Application No.: 09/608,872 Group Art Unit: ~~2741~~ 2155

U.S. Patent Documents

Examiner Initial	No.	Patent No.	Date	Patentee	Class	Sub-class	Filing Date
	A	6,026,388	02/15/00	Liddy et al.	707	1	08/14/96
	B	6,102,030	01/04/00	French- St. George et al.	704	275	04/21/98
	C	6,003,072	12/14/99	Gerritsen et al.	709	218	06/30/94
	D	5,890,123	03/30/99	Brown et al.	704	275	06/05/95
	E	5,855,002	12/29/98	Armstrong	704	270	06/11/96
	F	5,963,940	10/05/99	Liddy et al.	707	5	08/14/96
	G	5,805,775	09/08/98	Eberman et al.	395	12	02/02/96
	H	5,802,526	09/01/98	Fawcett et al.	707	10	04/18/96
	I	5,794,050	08/11/98	Dahlgren et al.	395	708	10/02/97
	J	5,774,859	06/30/98	Houser et al.	704	275	01/03/95
	K	5,748,974	05/05/98	Johnson	395	759	12/13/94

Foreign Patent or Published Foreign Patent Application

Examiner Initial	No.	Document No.	Publication Date	Country or Patent Office	Class	Sub-class	Translation Yes	Translation No
	L							
	M							
	N							
	O							
	P							

Other Documents

Examiner Initial	No.	Author, Title, Date, Place (e.g. Journal) of Publication
	R	Stent, Amanda et al., "The CommandTalk Spoken Dialogue System", SRI International
	S	Moore, Robert et al., "CommandTalk: A Spoken-Language Interface for Battlefield Simulations", October 23, 1997, SRI International
	T	Dowding, John et al., "Interpreting Language in Context in CommandTalk", February 5, 1999, SRI International

Examiner	Date Considered 9/22/02

Examiner: Initial citation considered. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

#7

Form 1449 (Modified) **Information Disclosure Statement By Applicant** (Use Several Sheets if Necessary)	Atty. Docket No. SRI1P037B Applicant: Halverson et al. Filing Date: 06/30/2000
	Application No.: 09/608,872 Group Art Unit: ~~2741~~ 2155

U.S. Patent Documents

Examiner Initial	No.	Patent No.	Date	Patentee	Class	Sub-class	Filing Date
JO	A	5,729,659	03/17/98	Potter	395	2.79	06/06/95
	B	5,721,938	02/24/98	Stuckey	395	754	06/07/95
	C	5,608,624	03/04/97	Luciw	395	794	05/15/95
	D	5,519,608	05/21/96	Kupiec	364	419.08	06/24/93
	E	5,434,777	07/18/95	Luciw	364	419.13	03/18/94
	F	5,386,556	01/31/95	Hedin et al.	395	600	[illegible]/23/92
JO	G	5,197,005	03/23/93	Shwartz et al.	364	419	05/01/89
	H						
	I						
	J						
	K						

Foreign Patent or Published Foreign Patent Application

Examiner Initial	No.	Document No.	Publication Date	Country or Patent Office	Class	Sub-class	Translation	
							Yes	No
	L							
	M							
	N							
	O							
	P							

Other Documents

Examiner Initial	No.	Author, Title, Date, Place (e.g. Journal) of Publication
JO	R	http://www.ai.sri.com/~oaa/infowiz.html, "InfoWiz: An Animated Voice Interactive Information System, May 8, 2000
JO.	S	Dowding, John, "Interleaving Syntax and Semantics in an Efficient Bottom-up Parser", SRI International
JO	T	Moore, Robert et al., "Combining Linguistic and Statistical Knowledge Sources in Natural-Language Processing for ATIS", SRI International

Examiner	Date Considered 9/27/02

Examiner: Initial citation considered. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

#7

Form 1449 (Modified) **Information Disclosure Statement By Applicant** (Use Several Sheets if Necessary)	Atty. Docket No. SRI1P037B Applicant: Halverson et al. Filing Date: 06/30/2000	Application No.: 09/608,872 Group Art Unit: ~~2741~~ 2155

U.S. Patent Documents

Examiner Initial	No.	Patent No.	Date	Patentee	Class	Sub-class	Filing Date
	A						
	B						
	C						
	D						
	E						
	F						
	G						
	H						
	I						
	J						
	K						

Foreign Patent or Published Foreign Patent Application

Examiner Initial	No.	Document No.	Publication Date	Country or Patent Office	Class	Sub-class	Translation Yes	Translation No
	L							
	M							
	N							
	O							
	P							

Other Documents

Examiner Initial	No.	Author, Title, Date, Place (e.g. Journal) of Publication
TO·	R	Dowding, John et al., "Gemini: A Natural Language System For Spoken-Language Understanding", SRI International
	S	
	T	

Examiner	Date Considered 9/27/02

Examiner: Initial citation considered. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

4B17

GAU-2155

Attorney Docket No.: SRI1P037B

#8

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF: HALVERSON, CHRISTINE
SERIAL NO.: 09/608,872
FILED: 6/30/00
TITLE: MOBILE NAVIGATION OF NETWORK-BASED ELECTRONIC INFORMATIONUSING SPOKEN INPUT

ASSOCIATE POWER OF ATTORNEY

Assistant Commissioner for Patents
Washington, DC 20231

Dear Sir:

I hereby appoint: C. Douglas McDonald (Reg. No. 26,659)

whose post office address is

Carlton Fields, P.A.
P. O. Box 3239
Tampa, Florida 33601-3239

as my associate attorney in the above-entitled application, to prosecute this application, to make alterations and amendments therein, and to transact all business in the Patent and Trademark Office connected therewith.

Please continue to address all future communications to:

Carlton Fields, LLP
P. O. Box 721030
San Jose, CA 95172-1030

Respectfully submitted

Date: MAY 7, 2001

Kevin J. Zilka (Reg. No. 41,429)
Dominic Kotab (Reg. No. 42,762)
Carlton Fields LLP
P.O. Box 721030
San Jose, CA 95172-1030
Telephone: (408) 271-2300
Fax: (408) 275-9579

TPA#1680358.01

#9

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION NO.: 09/608,872
INVENTOR: Halverson, Christine
TITLE: MOBILE NAVIGATION OF NETWORK-BASED ELECTRONIC INFORMATIONUSING SPOKEN INPUT

FILING DATE: 6/30/00
ATTORNEY DOCKET NO. SRI1P037B

NOTICE OF CHANGE OF CORRESPONDENCE ADDRESS

Assistant Commissioner for Patents
Washington, DC 20231

Sir:

Please change the correspondence address relating to the above-identified application as follows:

C. Douglas McDonald , Esq.
Carlton Fields, et al.
P.O. Box 3239
Tampa, FL 33601-3239

Respectfully submitted,

C. Douglas McDonald

Date: May 10, 2001

C. Douglas McDonald
Reg. No. 26,659
CARLTON FIELDS, P.A.
P.O. Box 3239
Tampa, FL 33601-3239
(813) 223-7000
Attorney of Record

TPA#1524975.01

U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a)	Docket Number (Optional) SRI 1P037B

#10
LDJ
10-01-0

In re Application of HALVERSON, et al

Application Number 09/608,872 | Filed June 30, 2000

For Mobile Navigation of Network-Based Electronic Information Using Spoken Input

Group Art Unit 2155 | Examiner F. Backer

This is a request under the provisions of 37 CFR 1.136(a) to extend the period for filing a response in the above identified application.

The requested extension and appropriate non-small-entity fee are as follows (check time period desired):

□ One month (37 CFR 1.17(a)(1))	$
☒ Two months (37 CFR 1.17(a)(2))	$390.00
□ Three months (37 CFR 1.17(a)(3))	$
□ Four months (37 CFR 1.17(a)(4))	$
□ Five months (37 CFR 1.17(a)(5))	$

☒ Applicant claims small entity status. See 37 CFR 1.27. Therefore, the fee amount shown above is reduced by one-half, and the resulting fee is: $ 195.00 .

☒ A check in the amount of the fee is enclosed.

□ Payment by credit card. Form PTO-2038 is attached.

□ The Commissioner has already been authorized to charge fees in this application to a Deposit Account.

☒ The Commissioner is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account Number 20-0782 .
I have enclosed a duplicate copy of this sheet.

I am the □ applicant/inventor.

□ assignee of record of the entire interest. See 37 CFR 3.71
Statement under 37 CFR 3.73(b) is enclosed. (Form PTO/SB/96).

☒ attorney or agent of record.

□ attorney or agent under 37 CFR 1.34(a).
Registration number if acting under 37 CFR 1.34(a). ______ .

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

September 19, 2001
Date

Signature

KIN-WAH TONG, Reg. No. 39,400
Typed or printed name

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below*.

□ *Total of ____ forms are submitted.

Burden Hour Statement: This form is estimated to take 0.1 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

SRI/4116-6

#11
LSS
10-01-01

IN THE UNITED STATES
PATENT AND TRADEMARK OFFICE

PATENT APPLICATION

Applicant(s):	**HALVERSON, et al.**	Atty. Docket No.	**SRI 1P037B**
Serial No.:	**09/608,872**	Group Art Unit:	**2155**
Filed:	**June 30, 2000**	Examiner:	**F. Backer**
Title:	**MOBILE NAVIGATION OF NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN INPUT**		

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

REVOCATION OF PREVIOUS POWER OF ATTORNEY AND NEW APPOINTMENT

The undersigned assignee of the above-identified application hereby revokes all previous Powers of Attorney and appoints the following attorneys with full power to prosecute the application, to make alterations and amendments therein, and to transact all business in the United States Patent and Trademark Office connected therewith and with full power of substitution and revocation:

Raymond R. Moser, Jr.; Reg. No. 34,682; Kin-Wah Tong, Reg. No. 39,400; Robert Brush, Reg. No. 45,710; Steven Weiner, Reg. No. 38,360; and Edward E. Davis, Reg. No. 35,112.

CHANGE OF CORRESPONDENCE ADDRESS

Please change the correspondence address for the above-identified application to:

Thomason, Moser & Patterson, LLP
595 Shrewsbury Avenue – Suite 100
Shrewsbury, New Jersey 07702

Please direct all telephone calls to: Kin-Wah Tong, telephone # (732) 530-9404

CERTIFICATE UNDER 37 C.F.R. § 3.73(B)

SRI International, a corporation of the State of California, certifies that it is the assignee of the entire right, title and interest in the patent application identified above by virtue of:

An Assignment from the inventor(s) of the parent patent application that is claimed as priority in the above-identified patent application. The Assignment was recorded in the United States Patent and Trademark Office, for which a copy thereof is attached.

The undersigned (whose title is supplied below) is empowered to act on behalf of the assignee.

Respectfully submitted,

Date: 9/11/01

~~EDWARD E. DAVIS, Assistant Secretary~~
STEVEN LORINER, VICE PRESIDENT

SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025
Telephone No.: 650-859-3115

ASSIGNMENT OF PATENT APPLICATION

(Not Accompanying Application)

Whereas I/we the undersigned inventor(s) have invented certain new and useful improvements as set forth in the patent application entitled:

NAVIGATING NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN NATURAL LANGUAGE INPUT WITH MULTIMODAL ERROR FEEDBACK

for which I/we have executed an application for a United States Letters Patent which was filed in the U.S. Patent and Trademark Office on March 13, 2000, and which bears the Application No. 09/524,095.

For good and valuable consideration, the receipt and sufficiency of which is hereby acknowledged, I/we the undersigned inventor(s) hereby:

1) Sell(s), assign(s) and transfer(s) to **SRI International**, a California non-profit corporation having a place of business at 333 Ravenswood Avenue, Menlo Park, California 94025, (hereinafter referred to as "ASSIGNEE"), the entire right title and interest in any and all improvements and inventions disclosed in, application(s) based upon, and Patent(s) (including foreign patents) granted upon the information which is disclosed in the above referenced application.

2) Authorize and request the Commissioner of Patents to issue any and all Letters Patents resulting from said application or any division(s), continuation(s), substitutes(s) or reissue(s) thereof to the ASSIGNEE.

3) Agree to execute all papers and documents and, entirely at the ASSIGNEE's expense, perform any acts which are reasonably necessary in connection with the prosecution of said application, as well as any derivative and applications thereof, foreign applications based thereon, and/or the enforcement of patents resulting from such applications.

4) Agree that the terms, covenants and conditions of this assignment shall inure to the benefit of the Assignee, its successors, assigns and other legal representative, and shall be binding upon the inventor(s), as well as the inventor's heirs, legal representatives and assigns.

5) Warrant and represent that I/we have not entered, and will not enter into any assignment, contract, or understanding that conflicts with this assignment.

Signed on the date(s) indicated beside my (our) signature(s).

1) Signature: ____________________ Date: 6-16-00.
Typed Name: Christine Halverson

2) Signature: ____________________ Date: ________
Typed Name: Luc Julia

3) Signature: ____________________ Date: 6/16/00
Typed Name: Dimitris Voutsas

4) Signature: ____________________ Date: 6/22/00
Typed Name: Adam Cheyer

Attny Docket No. SRI1P037

ASSIGNMENT OF PATENT APPLICATION

(Not Accompanying Application)

Whereas I/we the undersigned inventor(s) have invented certain new and useful improvements as set forth in the patent application entitled:

NAVIGATING NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN NATURAL LANGUAGE INPUT WITH MULTIMODAL ERROR FEEDBACK

for which I/we have executed an application for a United States Letters Patent which was filed in the U.S. Patent and Trademark Office on March 13, 2000, and which bears the Application No. 09/524,095.

For good and valuable consideration, the receipt and sufficiency of which is hereby acknowledged, I/we the undersigned inventor(s) hereby:

1) Sell(s), assign(s) and transfer(s) to **SRI International**, a California non-profit corporation having a place of business at 333 Ravenswood Avenue, Menlo Park, California 94025, (hereinafter referred to as "ASSIGNEE"), the entire right title and interest in any and all improvements and inventions disclosed in, application(s) based upon, and Patent(s) (including foreign patents) granted upon the information which is disclosed in the above referenced application.

2) Authorize and request the Commissioner of Patents to issue any and all Letters Patents resulting from said application or any division(s), continuation(s), substitutes(s) or reissue(s) thereof to the ASSIGNEE.

3) Agree to execute all papers and documents and, entirely at the ASSIGNEE's expense, perform any acts which are reasonably necessary in connection with the prosecution of said application, as well as any derivative and applications thereof, foreign applications based thereon, and/or the enforcement of patents resulting from such applications.

4) Agree that the terms, covenants and conditions of this assignment shall inure to the benefit of the Assignee, its successors, assigns and other legal representative, and shall be binding upon the inventor(s), as well as the inventor's heirs, legal representatives and assigns.

5) Warrant and represent that I/we have not entered, and will not enter into any assignment, contract, or understanding that conflicts with this assignment.

Signed on the date(s) indicated beside my (our) signature(s).

1) Signature: ______________ Date: 6-16-00.
Typed Name: Christine Halverson

2) Signature: ______________ Date: 6 · 20 . 00
Typed Name: Luc Julia

3) Signature: ______________ Date: 6/16/00
Typed Name: Dimitris Voutsas

4) Signature: ______________ Date: ________
Typed Name: Adam Cheyer

Attny Docket No. SRI1P037

ASSIGNMENT OF PATENT APPLICATIO

(Not Accompanying Application)

Whereas I/we the undersigned inventor(s) have invented certain new and useful improvements as set forth in the patent application entitled:

NAVIGATING NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN NATURAL LANGUAGE INPUT WITH MULTIMODAL ERROR FEEDBACK

for which I/we have executed an application for a United States Letters Patent which was filed in the U.S. Patent and Trademark Office on March 13, 2000, and which bears the Application No. 09/524,095.

For good and valuable consideration, the receipt and sufficiency of which is hereby acknowledged, I/we the undersigned inventor(s) hereby:

1) Sell(s), assign(s) and transfer(s) to **SRI International**, a California non-profit corporation having a place of business at 333 Ravenswood Avenue, Menlo Park, California 94025, (hereinafter referred to as "ASSIGNEE"), the entire right title and interest in any and all improvements and inventions disclosed in, application(s) based upon, and Patent(s) (including foreign patents) granted upon the information which is disclosed in the above referenced application.

2) Authorize and request the Commissioner of Patents to issue any and all Letters Patents resulting from said application or any division(s), continuation(s), substitutes(s) or reissue(s) thereof to the ASSIGNEE.

3) Agree to execute all papers and documents and, entirely at the ASSIGNEE's expense, perform any acts which are reasonably necessary in connection with the prosecution of said application, as well as any derivative and applications thereof, foreign applications based thereon, and/or the enforcement of patents resulting from such applications.

4) Agree that the terms, covenants and conditions of this assignment shall inure to the benefit of the Assignee, its successors, assigns and other legal representative, and shall be binding upon the inventor(s), as well as the inventor's heirs, legal representatives and assigns.

5) Warrant and represent that I/we have not entered, and will not enter into any assignment, contract, or understanding that conflicts with this assignment.

Signed on the date(s) indicated beside my (our) signature(s).

1) Signature: Christine Halverson Date: 6-16-00.
Typed Name: Christine Halverson

2) Signature: ______________ Date: ________
Typed Name: Luc Julia

3) Signature: ______________ Date: 6/16/00
Typed Name: Dimitris Voutsas

4) Signature: ______________ Date: ________
Typed Name: Adam Cheyer

Atty. Docket No. SRI1P037

#12
LDS
10-01-01

IN THE UNITED STATES
PATENT AND TRADEMARK OFFICE

PATENT APPLICATION

Applicant: **Halverson et al.**

Case: **SRI1P037B**

Serial No.: **09/608,872** Filed: **June 30, 2000**

Group Art Unit: **2155**

Examiner: **Firmin Backer**

Title: **MOBILE NAVIGATION OF NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN INPUT**

ASSISTANT COMMISSIONER FOR PATENTS
Box Non-Fee Amendment
Washington, D. C. 20231

SIR:

RESPONSE UNDER 37 C.F.R. § 1.111

This response addresses the Office Action dated April 24, 2001 (Paper No. 10).

REMARKS

In view of the following discussion, the Applicants submit that none of the claims now pending in the application are anticipated under the provisions of 35 U.S.C. § 102. Thus, the Applicants believe that all of these claims are now in allowable form.

I. REJECTION OF CLAIMS 56-82 UNDER DOUBLE PATENTING

The Examiner provisionally rejected claims 56-82 in Paragraphs 1-2 of the Office Action based on statutory type double patenting under 35 U.S.C. § 101 as claiming the same invention as that of claims 56-126 of copending Application No. 09/524,095. Applicants respectfully traverse the rejection.

First, the Examiner noted that "it would have been obvious to one of ordinary skill in the art to observe that the omission of the limitations '**soliciting additional input**

from the user, including user interaction in a modality different tha[n] the original request and, refining the navigation query, based upon the additional input'. After noting the differences between the scope of the claims between the two applications, the Examiner then concluded that claims 56-82 "are obvious variation of the inventive concept defined in claims 56-126 of co-pending application 09/524,095".

Applicants direct the Examiner's attention to the fact that there are two types of double patenting rejections: "statutory" and "non-statutory (obviousness-type)". MPEP 804 states that "[i]n determining whether a statutory basis for a double patenting rejection exists, the question to be asked is: Is the same invention being claimed twice?" "A reliable test for double patenting under 35 U.S.C. 101 is whether a claim in the application could be literally infringed without literally infringing a corresponding claim in the patent". Given the substantial differences between the claims of the two applications as noted by the Examiner, Applicants respectfully submit that applying the statutory double patenting test as promoted in the MPEP would not produce a statutory double patenting rejection in the present application. As such, Applicants submit that the present statutory double patenting rejection against claims 56-82 is inappropriate.

Second, it should be noted that the present application is a continuation of the co-pending application 09/524,095. As such, if and when these two applications mature into issued patents, both patents will have the same term. Thus, given the differences between the scope of the claims of both applications and the fact that both applications will expire at the same time (if issued), Applicants respectfully submit that statutory double patenting rejection against claims 56-82 is inappropriate.

II. REJECTION OF CLAIMS 56-82 UNDER 35 U.S.C. § 102

The Examiner has rejected claims 56-82 in Paragraphs 4-19 of the Office Action as being anticipated by the Levin et al. patent (US Patent 6,173,279 issued January 9, 2001, hereinafter referred to as Levin). The rejection is respectfully traversed.

Levin teaches "a method of using at least one natural language query to retrieve information from one or more data resources and further performing a requested action using the retrieved information is disclosed". (See Levin, Column 2, lines 15-18)

Namely, Levin teaches a method for using natural language query to obtain information, where upon receipt of the requested information, a desired action is executed based upon the requested information. To illustrate, Levin provides the example, where a user employs natural language to request the telephone number of a restaurant. Upon receipt of the telephone number, the telephone number is actually dialed for the user. (See Levin, Column 3 line 62 to Column 4, line 1)

In contrast, Levin fails to teach or suggest the novel concept of speech-based navigation where the method receives spoken request for desired information from the user utilizing the mobile information appliance of the user and where, in turn, the selected electronic data source from the network server is transmitted to the mobile information appliance of the user. Specifically, Applicants' independent claims 56, 65 and 74 positively recite:

> 56. A method for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, wherein at least a portion of a data link between a mobile information appliance of the user and the one or more network servers utilize wireless communication, comprising the steps of:
>
> (a) receiving a spoken request for desired information from the user utilizing the mobile information appliance of the user;
>
> (b) rendering an interpretation of the spoken request;
>
> (c) constructing a navigation query based upon the interpretation;
>
> (d)utilizing the navigation query to select a portion of the electronic data source; and
>
> (e) transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user. (emphasis added)
>
> 65. A computer program embodied on a computer readable medium for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, wherein at least a portion of a data link between a mobile information appliance of the user and the one or more network servers utilizes wireless communication, comprising:
>
> (a) a code segment that receives a spoken request for desired information from the user utilizing the mobile information appliance of the user;
>
> (b) a code segment that renders an interpretation of the spoken request.
>
> (c) a code segment that constructs a navigation query based upon the

interpretation;

(d) a code segment that utilizes the navigation query to select a portion of the electronic data source; and

(e) a code segment that transmits the selected portion of the electronic data source from the network server to the mobile information appliance of the user. (emphasis added)

74. A system for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, comprising:

(a) a mobile information appliance operable to receive a spoken request for desired information from the user;

(b) spoken language processing logic, operable to render an interpretation of the spoken request;

(c) query construction logic, operable to construct a navigation query based upon the interpretation;

(d) navigation logic, operable to select a portion of the electronic data source using the navigation query, and

(e) electronic communications infrastructure for transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user, wherein at least a portion of a data link of the electronic communications infrastructure between a mobile information appliance of the user and the one or more network servers utilizes wireless communication. (emphasis added)

Applicants' invention teaches a novel method and apparatus for speech-based navigation where the method receives spoken request for desired information from the user utilizing the **mobile information appliance** of the user and where, in turn, the selected electronic data source from the network server is **transmitted to the mobile information appliance** of the user. Specifically, Applicants address the criticality of providing speech-based navigation via a mobile, i.e., wireless communication, approach in addition to spoken natural language. It has been noted that with the proliferation of various mobile appliances, it would be advantageous to allow these mobile appliances to access the same vastness of electronic data sources that are available to hard-wired appliances like a desktop computer. However, the very essence of a mobile appliance is its portability, small size and ease of use. As such, unlike hard-wired appliances, mobile appliances are not equipped with large bulky input devices. In fact, even if the mobile appliance is equipped with extensive input devices, most users would still find these "shrunken" input devices to be cumbersome and difficult to use, e.g., an electronic representation of a keyboard on a PDA and the like.

To further exacerbate the problem, obtaining information from an electronic data source may require extensive and complex interaction between the user's mobile appliance and the system holding the electronic data source. Thus, the limited or cumbersome input/output capability of a mobile appliance presents a substantial barrier to its ability to access a data resource that requires extensive and complex interaction.

To address this criticality, Applicants disclose a speech-based navigation method that is deployed in conjunction with mobile appliances. To illustrate, the user can request via a mobile appliance, e.g., a cellular telephone, all the names of a particular ethnic restaurant on a particular street. Clearly, this request is rather complex given the limited input capability (generally a numeric keypad) of a cellular phone. Without additional input devices, this complex request may require numerous interactions between the user and a remote data resource, e.g., long repeated sequences of presenting a menu, scrolling within the menu and selecting the desired information within the menu and so on for the next menu and beyond. Such tedium discourages a user from attempting to acquire complex information via mobile appliances.

In contrast, Applicants' invention allows the complex request to be received as a spoken request directly via the user's mobile information appliance, thereby substantially reducing the amount of interaction of the user with the remote data resource. The present method will interpret and construct a navigation query that is utilized to obtain the selected data. For example, if the navigation query produces three possible results, then the results can be simply transmitted to the user via a menu on the screen of the mobile appliance.

In contrast, Levin teaches that "[u]sing a personal computer (PC) 102, a user establishes a connection with packet network 108 via an access server 106". Levin then states that "[t]he user may also use a telephone 103 to connect to the packet network 108" and that "[t]ypically a modem connection (not shown) may be used to connect the PC 102 to the packet 108 in a conventional manner". (emphasis added) (See Levin, Column 3, lines 5-10). Additionally, Levin states that "[t]he PC 102 dials into an access server 106 that is connected to the Internet or other database service via a logical network interface (not shown)" and that "[t]he logical network interface may be a local area network (LAN), a Serial Line Internet Protocol (SLIP) connection over a modem, an ISDN port or via a connection to a special LAN such as an ATM LAN or a LAN that offers bandwidth reservation". (See Levin, Column 4, lines 23-29) It is respectfully submitted that none of Levin's statements provides any specific teaching as to mobile appliances or wireless communication. In fact, terms such as "modem connection" and "ISDN port" are typically associated with hard-wired appliances. Thus, Levin does not teach or disclose a method that receives spoken request for desired information from the user utilizing the mobile information appliance of the user and where, in turn, the selected electronic data source from the network server is transmitted to the mobile information appliance of the user. Namely, the scope of Applicants' claims is specifically directed to speech-based navigation via mobile information appliances. This novel concept is not disclosed by the Levin reference and Applicants' claims would not read on the Levin reference.

Therefore, the Applicants respectfully submit that independent claims 56, 65 and 74 are not anticipated by the Levin reference. As such, claims 56, 65 and 74 fully satisfy the requirements of 35 U.S.C. §102 and are patentable thereunder.

Claims 57-64, 66-73 and 75-82 depend, either directly or indirectly, from claims 56, 65 and 74 and recite additional features therefor. Since Levin fails to anticipate Applicants' invention as recited in Applicants' independent claims 56, 65 and 74, dependent claims 57-64, 66-73 and 75-82 are also not anticipated under 35 U.S.C. § 102 and are allowable for the same reason noted above.

Conclusion

Thus, the Applicants submit that all of these claims now fully satisfy the requirements of 35 U.S.C. §102. Consequently, the Applicants believe that all these claims are presently in condition for allowance. Accordingly, both reconsideration of this application and its swift passage to issue are earnestly solicited.

If, however, the Examiner believes that there are any unresolved issues requiring the issuance of a final action in any of the claims now pending in the application, it is requested that the Examiner telephone Mr. Kin-Wah Tong, Esq. at (732) 530-9404 so that appropriate arrangements can be made for resolving such issues as expeditiously as possible.

Respectfully submitted,

9/19/01

Kin-Wah Tong, Attorney
Reg. No. 39,400
(732) 530-9404

Moser, Patterson & Sheridan, LLP
595 Shrewsbury Avenue
First Floor,
Shrewsbury, New Jersey 07702

GAU 2155

$

PTO/SB/21 (08-00)

Please type a plus sign (+) inside this box ⟶ [+]

Approved for use through 10/31/2002. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	09/608,872
Filing Date	June 30, 2000
First Named Inventor	HALVERSON
Group Art Unit	2155
Examiner Name	F. BACKER
Attorney Docket Number	SRI 1 P 037B

Total Number of Pages in This Submission

ENCLOSURES *(check all that apply)*

☒ Fee Transmittal Form

☒ Fee Attached

☒ Amendment / Response

☐ After Final

☐ Affidavits/declaration(s)

☒ Extension of Time Request

☐ Express Abandonment Request

☐ Information Disclosure Statement

☐ Certified Copy of Priority Document(s)

☐ Response to Missing Parts/ Incomplete Application

☐ Response to Missing Parts under 37 CFR 1.52 or 1.53

☐ Assignment Papers *(for an Application)*

☐ Drawing(s)

☐ Licensing-related Papers

☐ Petition

☐ Petition to Convert to a Provisional Application

☒ Power of Attorney, Revocation Change of Correspondence Address

☐ Terminal Disclaimer

☐ Request for Refund

☐ CD, Number of CD(s)

☐ After Allowance Communication to Group

☐ Appeal Communication to Board of Appeals and Interferences

☐ Appeal Communication to Group *(Appeal Notice, Brief, Reply Brief)*

☐ Proprietary Information

☐ Status Letter

☐ Other Enclosure(s) *(please identify below)*:

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm or Individual name	KIN-WAH TONG
Signature	
Date	September 19, 2001

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231 on this date: September 19, 2001

Typed or printed name	Linda DeNardi		
Signature	Linda DeNardi	Date	September 19, 2001

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be send to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

U.S. Patent and Trademark Office; DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no ... is are required to respond to a collection of information unless it ... is a valid OMB control number.

FEE TRANSMITTAL for FY 2001

Patent fees are subject to annual revision.

DURING PENDENCY, PLEASE CHARGE DEPOSIT ACCOUNT 20-0782 FOR ANY 37 C.F.R. 1.16 AND/OR 37 C.F.R. 1.17 FEES DUE AND NOT OTHERWISE AUTHORIZED. PLEASE CREDIT DEPOSIT ACCOUNT 20-0782 FOR ANY OVERPAYMENTS

TOTAL AMOUNT OF PAYMENT ($) 195.00

Complete if Known	
Application Number	09/608,872
Filing Date	June 30, 2000
First Named Inventor	HALVERSON
Examiner Name	F. BACKER
Group / Art Unit	2155
Attorney Docket No.	SRI 1P037B

METHOD OF PAYMENT (check one)

1. ☒ The Commissioner is hereby authorized to charge indicated fees and credit any over payments to:

Deposit Account Number: 20-0782

Deposit Account Name:

☒ Charge Any Additional Fee Required Under 37 CFR 1.16 and 1.17

☒ Applicant claims small entity status. See 37 CFR 1.27

2. ☒ Payment Enclosed:

☒ Check ☐ Credit card ☐ Money Order ☐ Other

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
101	710	201	355	Utility filing fee	
106	320	206	160	Design filing fee	
107	490	207	245	Plant filing fee	
108	710	208	355	Reissue filing fee	
114	150	214	75	Provisional filing fee	

SUBTOTAL (1) ($) 0

2. EXTRA CLAIM FEES

			Extra Claims		Fee from below		Fee Paid
Total Claims		-20** =	0	X		=	0
Independent Claims		-3** =	0	X		=	0
Multiple Dependent				X		=	0

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
103	18	203	9	Claims in excess of 20
102	80	202	40	Independent claims in excess of 3
104	270	204	135	Multiple dependent claim, if not paid
109	80	209	40	** Reissue independent claims over original patent
110	18	210	9	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) ($) 0

**or number previously paid, if greater; For Reissues, see above

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
105	130	205	65	Surcharge - late filing fee or oath	
127	50	227	25	Surcharge - late provisional filing fee or cover sheet.	
139	130	139	130	Non-English specification	
147	2,520	147	2,520	For filing a request for reexamination	
112	920*	112	920*	Requesting publication of SIR prior to Examiner action	
113	1,840*	113	1,840*	Requesting publication of SIR after Examiner action	
115	110	215	55	Extension for reply within first month	
116	390	216	195	Extension for reply within second month	195.00
117	890	217	445	Extension for reply within third month	
118	1,390	218	695	Extension for reply within fourth month	
128	1,890	228	945	Extension for reply within fifth month	
119	310	219	155	Notice of Appeal	
120	310	220	155	Filing a brief in support of an appeal	
121	270	221	135	Request for oral hearing	
138	1,510	138	1,510	Petition to institute a public use proceeding	
140	110	240	55	Petition to revive – unavoidable	
141	1,240	241	620	Petition to revive – unintentional	
142	1,240	242	620	Utility issue fee (or reissue)	
143	440	243	220	Design issue fee	
144	600	244	300	Plant issue fee	
122	130	122	130	Petitions to the Commissioner	
123	130	123	130	Petitions related to provisional applications	
126	180	126	180	Submission of Information Disclosure Stmt	
581	40	581	40	Recording each patent assignment per property (times number of properties)	
146	710	246	355	Filing a submission after final rejection (37 CFR § 1.129(a))	
149	710	249	355	For each additional invention to be examined (37 CFR § 1.129(b))	
179	710	279	355	Request for Continued Examination (RCE)	
169	900	169	900	Request for expedited examination of a design application	

Other fee (specify)

*Reduced by Basic Filing Fee Paid

SUBTOTAL (3) ($) 195.00

SUBMITTED BY		Complete *(if applicable)*			
Name (Print/Type)	KIN-WAH TONG	*Registration No. Attorney/Agent)*	39,400	*Telephone*	(732) 530-9404
Signature				*Date*	SEPTEMBER 19, 2001

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

#11

UNITED STATES PATENT AND TRADEMARK OFFICE

COMMISSIONER FOR PATENTS
UNITED STATES PATENT AND TRADEMARK OFFICE
WASHINGTON, D.C. 20231
www.uspto.gov

APPLICATION NUMBER	FILING DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
09/608,872	06/30/2000	Christine Halversen	SRIlp037B

CONFIRMATION NO. 2382

OC000000006829442
OC000000006829442

C. DOUGLAS McDONALD, ESQ.
CALTON FIELDS, et al.
P. O. BOX 3239
TAMPA,, FL 33601-3239

Date Mailed: 10/02/2001

NOTICE REGARDING POWER OF ATTORNEY

This is in response to the Power of Attorney filed 09/21/2001.

- The Power of Attorney to you in this application has been revoked by the assignee who has intervened as provided by 37 CFR 3.71. Future correspondence will be mailed to the new address of record(37 CFR 1.33).

Lavinia Johnson

LAVINIA D JOHNSON
2100 7033085229

OFFICE COPY

#13

UNITED STATES PATENT AND TRADEMARK OFFICE

COMMISSIONER FOR PATENTS
UNITED STATES PATENT AND TRADEMARK OFFICE
WASHINGTON, D.C. 20231
www.uspto.gov

APPLICATION NUMBER	FILING DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
09/608,872	06/30/2000	Christine Halversen	SRIlp037B

CONFIRMATION NO. 2382

OC000000006829467
OC000000006829467

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY, NJ 07702

Date Mailed: 10/02/2001

NOTICE REGARDING POWER OF ATTORNEY

This is in response to the Power of Attorney filed 09/21/2001.

The Power of Attorney in this application is accepted. Correspondence in this application will be mailed to the above address as provided by 37 CFR 1.33.

Lavinia Johnson
LAVINIA D JOHNSON
2100 7033085229

OFFICE COPY

UNITED STA... DEPARTMENT OF COMMERCE
Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.
09/608,872	06/30/00	HALVERSEN	C SRILP037B

TM02/1010

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY NJ 07702

EXAMINER	
BACKER, F	
ART UNIT	PAPER NUMBER
2155	14

DATE MAILED:
10/10/01

Please find below and/or attached an Office communication concerning this application or proceeding.

Commissioner of Patents and Trademarks

PTO-90C (Rev. 2/95)
*U.S. GPO: 2000-473-000/44602

1- File Copy

Office Action Summary	**Application No.** 09/608,872	**Applicant(s)** HALVERSEN ET AL.	
	Examiner Firmin Backer	**Art Unit** 2155	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136 (a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *26 September 2001* .

2a)☒ This action is **FINAL.** 2b)□ This action is non-final.

3)□ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *56-82* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)□ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *56-82* is/are rejected.

7)□ Claim(s) _____ is/are objected to.

8)□ Claims _____ are subject to restriction and/or election requirement.

Application Papers

9)□ The specification is objected to by the Examiner.

10)□ The drawing(s) filed on _____ is/are objected to by the Examiner.

11)□ The proposed drawing correction filed on _____ is: a)□ approved b)□ disapproved.

12)□ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. § 119

13)□ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)□ All b)□ Some * c)□ None of:

1.□ Certified copies of the priority documents have been received.

2.□ Certified copies of the priority documents have been received in Application No. _____ .

3.□ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)□ Acknowledgement is made of a claim for domestic priority under 35 U.S.C. § 119(e).

Attachment(s)

15) □ Notice of References Cited (PTO-892)

16) □ Notice of Draftsperson's Patent Drawing Review (PTO-948)

17) □ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____ .

18) □ Interview Summary (PTO-413) Paper No(s). _____ .

19) □ Notice of Informal Patent Application (PTO-152)

20) □ Other:

Response to Request for Reconsideration

This is in response to a request for reconsideration file on September 26th, 2001. Claims 56-82 are being reconsidered in this action.

Double Patenting

1. A rejection based on double patenting of the "same invention" type finds its support in the language of 35 U.S.C. 101 which states that "whoever invents or discovers any new and useful process ... may obtain a patent therefor ..." (Emphasis added). Thus, the term "same invention," in this context, means an invention drawn to identical subject matter. See *Miller v. Eagle Mfg. Co.*, 151 U.S. 186 (1894); *In re Ockert*, 245 F.2d 467, 114 USPQ 330 (CCPA 1957); and *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970).

A statutory type (35 U.S.C. 101) double patenting rejection can be overcome by canceling or amending the conflicting claims so they are no longer coextensive in scope. The filing of a terminal disclaimer cannot overcome a double patenting rejection based upon 35 U.S.C. 101.

2. Claims 56-82 are provisionally rejected under 35 U.S.C. 101 as claiming the same invention as that of claims 56-126 of copending Application No. 09/524,095. Although the conflicting claims are not identical, they are not patentably distinct. It would have been obvious to one of ordinary skill in the art to observed that the omission of the limitations "**soliciting additional input from the user, including user interaction in a modality different that the original request and, refining the navigation query, based upon the additional input**", of applicant claims 56-82 are already in the Co-pending application 09/524,095, as such they are obvious variation of the inventive concept defined in claims 56-126 of the Co-pending application 09/524,095. See In re Karlson, 136USPQ 184 (CCPA 1963). This is a provisional double patenting rejection since the conflicting claims have not in fact been patented.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in a patent granted on an application for patent by another filed in the United States before the invention thereof by the applicant for patent, or on an international application by another who has fulfilled the requirements of paragraphs (1), (2), and (4) of section 371(c) of this title before the invention thereof by the applicant for patent.

4. Claims 56-82 are rejected under 35 U.S.C. 102(e) as being anticipated by Levin et al. (U.S. Patent No. 6,173,279).

5. As per claim 56, Levin et al teach a method for speech-based navigation (*information server, 110*) of an electronic data source located at one or more network servers located remotely from a user, wherein at least a portion of a data link between a mobile information appliance of the user and the one or more network servers utilizes wireless communication (see abstract, fig 1, column 3 lines 5-35), comprising receiving a spoken request (*receive a natural language query*) for desired information from the user (*user, 112*) utilizing the mobile information appliance (*PC, 102*) of the user; rendering an interpretation (*creating a semantic representation*) of the spoken request, constructing a navigation (*generating search)* query based upon the interpretation; utilizing the navigation query to select a portion of the electronic data source; and transmitting (*sending*) the selected portion of the electronic data source from the network server to the mobile information appliance of the user. (see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22)

6. As per claim 57, 58, 62-64, Levin et al teach a method of rendering the interpretation of the spoken request is performed at the one or more network servers by the mobile information appliance including a wireless telephone, a portable computer that is a personal digital assistance (see abstract, fig 1, column 3 lines 5-35).

7. As per claim 59, Levin et al teach a method of soliciting additional input from the user, including user interaction in a modality different than the original request; refining the navigation query, based upon the additional input; and using the refined navigation query to select a portion of the electronic data source (see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22).

8. As per claim 60, Levin et al teach a method wherein the data link includes a cellular telephone system (see fig 1, column 2 line 61-67).

9. As per claim 61, Levin et al teach a method wherein steps (a)-(d) are performed with respect to multiple users (see abstract, fig 1, column 3 lines 5-35).

10. As per claim 65, Levin et al teach a computer system for speech-based navigation (information server, 110) of an electronic data source located at one or more network servers located remotely from a user, wherein at least a portion of a data link between a mobile information appliance of the user and the one or more network servers utilizes wireless communication (see abstract, fig 1, column 3 lines 5-35), comprising a code segment receiving a

spoken request (*receive a natural language query*) for desired information from the user (user) utilizing the mobile information appliance (PC, 102) of the user; a code segment rendering an interpretation (*creating a semantic representation*) of the spoken request, a code segment constructing a navigation (*generating search)* query based upon the interpretation; a code segment utilizing the navigation query to select a portion of the electronic data source; and a code segment transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user. (see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22)

11. As per claim 66, 67, 71-73, Levin et al teach a system of rendering the interpretation of the spoken request is performed at the one or more network servers by the mobile information appliance including a wireless telephone, a portable computer that is a personal digital assistance (see abstract, fig 1, column 3 lines 5-35).

12. As per claim 68, Levin et al teach a system of soliciting additional input from the user, including user interaction in a modality different than the original request; refining the navigation query, based upon the additional input; and using the refined navigation query to select a portion of the electronic data source (see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22).

13. As per claim 69, Levin et al teach a system wherein the data link includes a cellular telephone system (see fig 1, column 2 line 61-67).

14. As per claim 70, Levin et al teach a system wherein steps (a)-(d) are performed with respect to multiple users (see abstract, fig 1, column 3 lines 5-35).

15. As per claim 74, Levin et al teach a system for speech-based navigation (information server, 110) of an electronic data source located at one or more network servers located remotely from a user, wherein at least a portion of a data link between a mobile information appliance of the user and the one or more network servers utilizes wireless communication (see abstract, fig 1, column 3 lines 5-35), comprising receiving a spoken request (*receive a natural language query*) for desired information from the user (user) utilizing the mobile information appliance (PC, 102) of the user; rendering an interpretation (*creating a semantic representation*) of the spoken request, constructing a navigation (*generating search)* query based upon the interpretation; utilizing the navigation query to select a portion of the electronic data source; and transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user. (see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22)

16. As per claim 75, 76, 80-81, Levin et al teach a method of rendering the interpretation of the spoken request is performed at the one or more network servers by the mobile information appliance including a wireless telephone, a portable computer that is a personal digital assistance (see abstract, fig 1, column 3 lines 5-35).

17. As per claim 77, Levin et al teach a system of soliciting additional input from the user, including user interaction in a modality different than the original request; refining the

navigation query, based upon the additional input; and using the refined navigation query to select a portion of the electronic data source (see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22).

18. As per claim 78, Levin et al teach a system wherein the data link includes a cellular telephone system (see fig 1, column 2 line 61-67).

19. As per claim 79, Levin et al teach a system wherein steps (a)-(d) are performed with respect to multiple users (see abstract, fig 1, column 3 lines 5-35).

Response to Arguments

1. Applicant's arguments filed on September 26th, 2001 have been fully considered but they are not persuasive. ***

a. Applicant argues that the statutory-type obviousness double patenting is not appropriate. Examiner respectfully disagrees with applicant characterization of the statutory-type obviousness double patenting concept. The inventive concepts in the applications are not patenbly different. Different variation of the same inventive concept is being claimed twice. According to MPEP in determining whether a statutory basis for a double patenting rejection exists, the question to be asked is: Is the same invention being claimed twice? 35 U.S.C. 101 prevents two patents from issuing on the same invention. "Same invention" means identical subject matter. Miller v. Eagle Mfg. Co., 151 U.S.

186 (1984); In re Vogel, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and In re Ockert, 245 F.2d 467, 114 USPQ 330 (CCPA 1957).

b. Applicant further argues that the prior art "fails to teach or suggest the novel concept of speech-based navigation where the method receives spoken request for desired information from the user utilizing the mobile information appliance of the user and where in turn the selected electronic data source from the network server is transmitted to the mobile information appliance of the user." Examiner respectfully disagrees with the applicant perspective and characterization of Levin inventive concept. Levin teach that use of a personal computer, a user establishes connection with a network. In the field of the network communication, a personal computer is not limited to desktop, but also handheld computer as well as laptop which are considered to be mobile appliances. In Levin inventive concept, an information server 110 receives natural language which is the same as spoken word. One the natural language query is process, the service host then transmit the result of the query to the pc. (see column 3 lines 5-35, 6 lines 25-59).

Conclusion

THIS ACTION IS MADE FINAL. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Firmin Backer whose telephone number is 703-305-0624. The examiner can normally be reached on Mon-Thu 8:30-6:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Sheikh Ayaz can be reached on 703-305-9648. The fax phone numbers for the organization where this application or proceeding is assigned are 703-305-3718 for regular communications and 703-305-5352 for After Final communications.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is 703-305-3900.

Firmin Backer
October 2, 2001

AYAZ SHEIKH
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

UNITED STATES F :NT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/608,872	06/30/2000	Christine Halversen	SRIIp037B	2382

7590 01/16/2002

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY, NJ 07702

EXAMINER
BACKER, FIRMIN

ART UNIT	PAPER NUMBER
2155	15

DATE MAILED: 01/16/2002

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 07-01)

Interview Summary	**Application No.** 09/608,872	**Applicant(s)** HALVERSEN ET AL.	
	Examiner Firmin Backer	**Art Unit** 2155	

All participants (applicant, applicant's representative, PTO personnel):

(1) *Firmin Backer*. (3) *Kin-Wah Tong*.

(2) *Ario Etienne*. (4)____.

Date of Interview: *08 January 2002* .

Type: a)☒ Telephonic b)□ Video Conference
c)□ Personal [copy given to: 1)□ applicant 2)□ applicant's representative]

Exhibit shown or demonstration conducted: d)□ Yes e)□ No.
If Yes, brief description: _____ .

Claim(s) discussed: *56* .

Identification of prior art discussed: *6,173,279* .

Agreement with respect to the claims f)□ was reached. g)□ was not reached. h)□ N/A.

Substance of Interview including description of the general nature of what was agreed to if an agreement was reached, or any other comments: *Applicant argues that the statutory double patenting rejection is improper and should be withdrawn. Applicant argues that the prior art fails to teach all the limitations of the inventive concept especially the use of wireless communication...* .

(A fuller description, if necessary, and a copy of the amendments which the examiner agreed would render the claims allowable, if available, must be attached. Also, where no copy of the amendments that would render the claims allowable is available, a summary thereof must be attached.)

i)□ It is not necessary for applicant to provide a separate record of the substance of the interview(if box is checked).

Unless the paragraph above has been checked, THE FORMAL WRITTEN REPLY TO THE LAST OFFICE ACTION MUST INCLUDE THE SUBSTANCE OF THE INTERVIEW. (See MPEP Section 713.04). If a reply to the last Office action has already been filed, APPLICANT IS GIVEN ONE MONTH FROM THIS INTERVIEW DATE TO FILE A STATEMENT OF THE SUBSTANCE OF THE INTERVIEW. See Summary of Record of Interview requirements on reverse side or on attached sheet.

Examiner Note: You must sign this form unless it is an Attachment to a signed Office action.

Examiner's signature, if required

#16/C
B
1/17/02
~~Auth~~
Enter
2-8-02

09/608,872

IN THE UNITED STATES
PATENT AND TRADEMARK OFFICE

PATENT APPLICATION

Applicant: **Halverson et al.**

Case: **SRI1P037B**

Serial No.: **09/608,872** Filed: **June 30, 2000**

Group Art Unit: **2155**

Examiner: **Firmin Backer**

Title: **MOBILE NAVIGATION OF NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN INPUT**

ASSISTANT COMMISSIONER FOR PATENTS
Box AF
Washington, D. C. 20231

S I R:

RESPONSE UNDER 37 C.F.R. § 1.116

This response addresses the Final Office Action dated October 10, 2001 (Paper No. 14).

IN THE CLAIMS

Please amend claims 56 and 65 as shown below. These claims are "clean version" of the amended claims, i.e., with changes incorporated into the claims, whereas the Appendix to this Amendment illustrates the amended claims using underlines and brackets to indicate addition and deletion, respectively.

56. (Amended) A method for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, wherein a data link is established between a mobile information appliance of the user and the one or more network servers, comprising the steps of:

(a) receiving a spoken request for desired information from the user utilizing the mobile information appliance of the user;

(b) rendering an interpretation of the spoken request;

(c) constructing a navigation query based upon the interpretation;

(d) utilizing the navigation query to select a portion of the electronic data source; and

(e) transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user, wherein at least a portion of said data link between said mobile information appliance of the user and the one or more network servers utilizes wireless communication.

65. (Amended) A computer program embodied on a computer readable medium for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, wherein a data link is established between a mobile information appliance of the user and the one or more network servers, comprising:

(a) a code segment that receives a spoken request for desired information from the user utilizing the mobile information appliance of the user;

(b) a code segment that renders an interpretation of the spoken request;

(c) a code segment that constructs a navigation query based upon the interpretation;

(d) a code segment that utilizes the navigation query to select a portion of the electronic data source; and

(e) a code segment that transmits the selected portion of the electronic data source from the network server to the mobile information appliance of the user, wherein at least a portion of said data link between said mobile information appliance of the user and the one or more network servers utilizes wireless communication.

REMARKS

Applicants' representative would like to thank Examiner Backer and Primary Examiner Etienne for kindly taking a substantial amount of time on January 8, 2002 to discuss the merits of the subject invention. Applicants' representative is aware of the time constraint that is placed on the Examiners and is appreciative of the Examiners' willingness to devote such large quantity of time to discuss the case on the merit.

In view of the following discussion, the Applicants submit that none of the claims now pending in the application are anticipated under the provisions of 35 U.S.C. § 102. Thus, the Applicants believe that all of these claims are now in allowable form.

I. REJECTION OF CLAIMS 56-82 UNDER DOUBLE PATENTING

The Examiner provisionally rejected claims 56-82 in Paragraphs 1-2 of the Final Office Action based on statutory type double patenting under 35 U.S.C. § 101 as claiming the same invention as that of claims 56-126 of copending Application No. 09/524,095. Applicants respectfully traverse the rejection.

First, the Examiner noted that "it would have been obvious to one of ordinary skill in the art to observe that the omission of the limitations '**soliciting additional input from the user, including user interaction in a modality different tha[n] the original request and, refining the navigation query, based upon the additional input**'. After noting the differences between the scope of the claims between the two applications, the Examiner then concluded that claims 56-82 "are obvious variation of the inventive concept defined in claims 56-126 of co-pending application 09/524,095".

Pursuant to the Examiner Interview, Applicants again directed Examiner's attention to the fact that there are two types of double patenting rejections: "statutory" and "non-statutory (obviousness-type)". MPEP 804 states that "[i]n determining whether a statutory basis for a double patenting rejection exists, the question to be asked is: Is the same invention being claimed twice?" "A reliable test for double patenting under 35 U.S.C. 101 is whether a claim in the application could be literally infringed without literally infringing a corresponding claim in the patent". Given the substantial differences between the claims of the two applications as noted by the Examiner, Applicants respectfully submit that applying the statutory double patenting test as promoted in the MPEP would not produce a statutory double patenting rejection in the present application.

Second, it should be noted that the present application is a continuation of the co-pending application 09/524,095. As such, if and when these two applications mature into issued patents, both patents will have the same term.

As such, Applicants submit that the present statutory double patenting rejection against claims 56-82 is inappropriate. The Examiners indicated that they will reconsider the present statutory type double patenting under 35 U.S.C. § 101.

II. REJECTION OF CLAIMS 56-82 UNDER 35 U.S.C. § 102

The Examiner has rejected claims 56-82 in Paragraphs 4-19 of the Final Office Action as being anticipated by the Levin et al. patent (US Patent 6,173,279 issued January 9, 2001, hereinafter referred to as Levin). The rejection is respectfully traversed.

Levin teaches "a method of using at least one natural language query to retrieve information from one or more data resources and further performing a requested action using the retrieved information is disclosed". (See Levin, Column 2, lines 15-18) Namely, Levin teaches a method for using natural language query to obtain information, where upon receipt of the requested information, a desired action is executed based upon the requested information. To illustrate, Levin provides the example, where a user employs natural language to request the telephone number of a restaurant. Upon receipt of the telephone number, the telephone number is actually dialed for the user. (See Levin, Column 3 line 62 to Column 4, line 1)

In contrast, Levin fails to teach or suggest the novel concept of speech-based navigation where the method receives spoken request for desired information from the user utilizing the mobile information appliance of the user and where, in turn, the selected electronic data source from the network server is transmitted to the mobile information appliance of the user, wherein at least a portion of said data link between said mobile information appliance of the user and the one or more network servers utilizes wireless communication. Specifically, Applicants' independent claims 56, 65 and 74 positively recite:

56. A method for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, wherein a data link is established between a mobile information appliance of the user and the one or more network servers, comprising the steps of:

(a) receiving a spoken request for desired information from the user utilizing the mobile information appliance of the user;

(b) rendering an interpretation of the spoken request;

(c) constructing a navigation query based upon the interpretation;

(d)utilizing the navigation query to select a portion of the electronic data source; and

(e) transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user, wherein at least a portion of said data link between said mobile information appliance of the user and the one or more network servers utilizes wireless communication. (emphasis added)

65. A computer program embodied on a computer readable medium for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, wherein a data link is established between a mobile information appliance of the user and the one or more network servers, comprising:

(a) a code segment that receives a spoken request for desired information from the user utilizing the mobile information appliance of the user;

(b) a code segment that renders an interpretation of the spoken request;

(c) a code segment that constructs a navigation query based upon the interpretation;

(d) a code segment that utilizes the navigation query to select a portion of the electronic data source; and

(e) a code segment that transmits the selected portion of the electronic data source from the network server to the mobile information appliance of the user, wherein at least a portion of said data link between said mobile information appliance of the user and the one or more network servers utilizes wireless communication. (emphasis added)

74. A system for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, comprising:

(a) a mobile information appliance operable to receive a spoken request for desired information from the user;

(b) spoken language processing logic, operable to render an interpretation of the spoken request;

(c) query construction logic, operable to construct a navigation query based upon the interpretation;

(d) navigation logic, operable to select a portion of the electronic data source using the navigation query, and

> (e) electronic communications infrastructure for transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user, wherein at least a portion of a data link of the electronic communications infrastructure between a mobile information appliance of the user and the one or more network servers utilizes wireless communication. (emphasis added)

Applicants' invention teaches a novel method and apparatus for speech-based navigation where the method receives spoken request for desired information from the user utilizing the **mobile information appliance** of the user and where, in turn, the selected electronic data source from the network server is **transmitted to the mobile information appliance** of the user, wherein at least a portion of said data link between said mobile information appliance of the user and the one or more network servers utilizes wireless communication. Specifically, Applicants address the criticality of providing speech-based navigation via a mobile, i.e., wireless communication, approach in addition to spoken natural language. It has been noted that with the proliferation of various mobile appliances, it would be advantageous to allow these mobile appliances to access the same vastness of electronic data sources that are available to hard-wired appliances like a desktop computer. However, the very essence of a mobile appliance is its portability, small size and ease of use. As such, unlike hard-wired appliances, mobile appliances are not equipped with large bulky input devices. In fact, even if the mobile appliance is equipped with extensive input devices, most users would still find these "shrunken" input devices to be cumbersome and difficult to use, e.g., an electronic representation of a keyboard on a PDA and the like.

To further exacerbate the problem, obtaining information from an electronic data source may require extensive and complex interaction between the user's mobile appliance and the system holding the electronic data source. Thus, the limited or cumbersome input/output capability of a mobile appliance presents a substantial barrier to its ability to access a data resource that requires extensive and complex interaction.

In contrast, Levin teaches that "[u]sing a personal computer (PC) 102, a user establishes a connection with packet network 108 via an access server 106". Levin then states that "[t]he user may also use a telephone 103 to connect to the packet network 108" and that "[t]ypically a modem connection (not shown) may be used to connect the PC 102 to the packet 108 in a conventional manner". (emphasis added) (See Levin, Column 3, lines 5-10). Additionally, Levin states that "[t]he PC 102 dials into an access server 106 that is connected to the Internet or other database service via a logical network interface (not shown)" and that "[t]he logical network interface may be a local area network (LAN), a Serial Line Internet Protocol (SLIP) connection over a modem, an ISDN port or via a connection to a special LAN such as an ATM LAN or a LAN that offers bandwidth reservation". (See Levin, Column 4, lines 23-29) It is respectfully submitted that none of Levin's statements provides any specific teaching as to mobile appliances or wireless communication. In fact, terms such as "modem connection" and "ISDN port" are typically associated with hard-wired appliances. Thus, Levin does not teach or disclose a method that receives spoken request for desired information from the user utilizing the mobile information appliance of the user and where, in turn, the selected electronic data source from the network server is transmitted to the mobile information appliance of the user via wireless communication over at least a portion of the data link. Namely, the scope of Applicants' claims is specifically directed to speech-based navigation via mobile information appliances. This novel concept is not disclosed by the Levin reference and Applicants' claims would not read on the Levin reference.

Pursuant to the Examiner Interview, Applicants have agreed to incorporate the term " wherein at least a portion of said data link between said mobile information appliance of the user and the one or more network servers utilizes wireless communication", into the body of the independent claims. This term previously existed in the preamble of the independent claims. Thus, since this term previously existed in the originally filed independent claims, the present amendment is not implemented in view of the cited prior art. In fact, Applicants take the position that the scope of the independent claims did not change as a result of this amendment and that this amendment served to clarify the claims to the Examiner's satisfaction.

Additionally, it should be noted that no amendment was applied to independent claim 74, since the above-identified term is already in the body of the independent claim

74.

Therefore, the Applicants respectfully submit that independent claims 56, 65 and 74 are not anticipated by the Levin reference. As such, claims 56, 65 and 74 fully satisfy the requirements of 35 U.S.C. §102 and are patentable thereunder.

Claims 57-64, 66-73 and 75-82 depend, either directly or indirectly, from claims 56, 65 and 74 and recite additional features therefor. Since Levin fails to anticipate Applicants' invention as recited in Applicants' independent claims 56, 65 and 74, dependent claims 57-64, 66-73 and 75-82 are also not anticipated under 35 U.S.C. § 102 and are allowable for the same reason noted above.

Conclusion

Thus, the Applicants submit that all of these claims now fully satisfy the requirements of 35 U.S.C. §102. Consequently, the Applicants believe that all these claims are presently in condition for allowance. Accordingly, both reconsideration of this application and its swift passage to issue are earnestly solicited.

If, however, the Examiner believes that there are any unresolved issues requiring the maintenance of the present final action in any of the claims now pending in the application, it is requested that the Examiner telephone Mr. Kin-Wah Tong, Esq. at (732) 530-9404 so that appropriate arrangements can be made for resolving such issues as expeditiously as possible.

Respectfully submitted,

1/10/02

Kin-Wah Tong, Attorney
Reg. No. 39,400
(732) 530-9404

Moser, Patterson & Sheridan, LLP
595 Shrewsbury Avenue
First Floor,
Shrewsbury, New Jersey 07702

Appendix
(Marked-up copy of amended claims)

56. (Amended) A method for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, wherein [at least a portion of] a data link is established between a mobile information appliance of the user and the one or more network servers [utilize wireless communication], comprising the steps of:

(a) receiving a spoken request for desired information from the user utilizing the mobile information appliance of the user;

(b) rendering an interpretation of the spoken request;

(c) constructing a navigation query based upon the interpretation;

(d)utilizing the navigation query to select a portion of the electronic data source; and

(e) transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user, wherein at least a portion of said data link between said mobile information appliance of the user and the one or more network servers utilizes wireless communication.

65. (Amended) A computer program embodied on a computer readable medium for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, wherein [at least a portion of] a data link is established between a mobile information appliance of the user and the one or more network servers [utilizes wireless communication], comprising:

(a) a code segment that receives a spoken request for desired information from the user utilizing the mobile information appliance of the user;

(b) a code segment that renders an interpretation of the spoken request.

(c) a code segment that constructs a navigation query based upon the interpretation;

(d) a code segment that utilizes the navigation query to select a portion of the electronic data source; and

(e) a code segment that transmits the selected portion of the electronic data source from the network server to the mobile information appliance of the user, wherein at least a portion of said data link between said mobile information appliance of the user and the one or more network servers utilizes wireless communication.

TELEFAX COVER SHEET

MOSER, PATTERSON & SHERIDAN
ATTORNEYS AT LAW
595 SHREWSBURY AVENUE
FIRST FLOOR
SHREWSBURY, NJ 07702
TELEPHONE (732) 530-9404
TELEFAX (732) 530-9808

****

THIS TELEFAX MESSAGE IS ADDRESSED TO THE PERSON OR COMPANY LISTED BELOW. IF IT WAS SENT OR RECEIVED INCORRECTLY, OR YOU ARE NOT THE INTENDED RECIPIENT, PLEASE TAKE NOTICE THAT THIS MESSAGE MAY CONTAIN PRIVILEGED OR CONFIDENTIAL MATERIAL, AND YOUR DUE REGARD FOR THIS INFORMATION IS NECESSARY. YOU MAY ARRANGE TO RETURN THIS MATERIAL BY CALLING THE FIRM LISTED ABOVE AT (732) 530-9404

****

THIS MESSAGE HAS 13 PAGES INCLUDING THIS SHEET

TO: Assistant Commissioner of Patents

FAX NO.: 703-746-7238

FROM: Kin-Wah Tong

DATE: January 10, 2002

MATTER: Serial No. 09/608,872 Filed: June 30, 2000

DOCKET NO.: SRI 1P037B

APPLICANT: HALVERSON, et al

The following has been received in the U.S. Patent and Trademark Office on the date of this facsimile:

___ Petition
___ Disclosure Statement & PTO-1449
___ Priority Document
___ Drawings (___ sheets) informal
_X_ Response Under 37 CFR 1.116
___ ____________
___ ____________

_X_ Transmittal Letter (2 copies)
___ Fee Transmittal (2 copies)
___ Deposit Account Transaction
_X_ Facsimile Transmission Certificate dated January 10, 2002
___ ____________
___ ____________

CERTIFICATE OF TRANSMISSION UNDER 37 C.F.R. §1.6

I hereby certify that this correspondence is being transmitted by facsimile to the Assistant Commissioner for Patents, Box AF, Washington, DC 20231 on January 10, 2002, Facsimile No. 703-746-7238.

Linda DeNardi
Name of person signing this certificate

Linda DeNardi January 10, 2002
Signature and date

PTO/SB/21 (08-00)
Approved for use through 10/31/2002. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Please type a plus sign (+) inside this box → [+]

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	09/608,872
Filing Date	June 30, 2000
First Named Inventor	HALVERSON
Group Art Unit	2155
Examiner Name	F. BACKER
Attorney Docket Number	SRI 1 P 037B

Total Number of Pages in This Submission: 13

ENCLOSURES *(check all that apply)*

- □ Fee Transmittal Form
  - □ Fee Attached
- ☒ Amendment / Response
  - □ After Final
  - □ Affidavits/declaration(s)
- □ Extension of Time Request
- □ Express Abandonment Request
- □ Information Disclosure Statement
- □ Certified Copy of Priority Document(s)
- □ Response to Missing Parts/ Incomplete Application
  - □ Response to Missing Parts under 37 CFR 1.52 or 1.53

- □ Assignment Papers *(for an Application)*
- □ Drawing(s)
- □ Licensing-related Papers
- □ Petition
- □ Petition to Convert to a Provisional Application
- □ Power of Attorney, Revocation Change of Correspondence Address
- □ Terminal Disclaimer
- □ Request for Refund
- □ CD, Number of CD(s)

- □ After Allowance Communication to Group
- □ Appeal Communication to Board of Appeals and Interferences
- □ Appeal Communication to Group *(Appeal Notice, Brief, Reply Brief)*
- □ Proprietary Information
- □ Status Letter
- □ Other Enclosure(s) *(please identify below)*:

Remarks: **It is believed no fee is due. However, in the event a fee is due, kindly charge that fee to deposit account number 20-0782. To facilitate that charge, a duplicate copy of this letter is enclosed**

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm *or* Individual name	PATRICIA A. VERLANGIERI, Reg. No. 42,201
Signature	Patricia A Verlangieri
Date	January 10, 2001

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be send to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

PTO/SB/21 (08-00)

Please type a plus sign (+) inside this box → [+]

Approved for use through 10/31/2002. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	09/608,872
Filing Date	June 30, 2000
First Named Inventor	HALVERSON
Group Art Unit	2155
Examiner Name	F. BACKER
Attorney Docket Number	SRI 1 P 037B

Total Number of Pages in This Submission: 13

ENCLOSURES *(check all that apply)*

- □ Fee Transmittal Form
  - □ Fee Attached
- ☒ Amendment / Response
  - □ After Final
  - □ Affidavits/declaration(s)
- □ Extension of Time Request
- □ Express Abandonment Request
- □ Information Disclosure Statement
- □ Certified Copy of Priority Document(s)
- □ Response to Missing Parts/ Incomplete Application
  - □ Response to Missing Parts under 37 CFR 1.52 or 1.53

- □ Assignment Papers *(for an Application)*
- □ Drawing(s)
- □ Licensing-related Papers
- □ Petition
- □ Petition to Convert to a Provisional Application
- □ Power of Attorney, Revocation Change of Correspondence Address
- □ Terminal Disclaimer
- □ Request for Refund
- □ CD, Number of CD(s)

- □ After Allowance Communication to Group
- □ Appeal Communication to Board of Appeals and Interferences
- □ Appeal Communication to Group *(Appeal Notice, Brief, Reply Brief)*
- □ Proprietary Information
- □ Status Letter
- □ Other Enclosure(s) *(please identify below)*:

Remarks: **It is believed no fee is due. However, in the event a fee is due, kindly charge that fee to deposit account number 20-0782. To facilitate that charge, a duplicate copy of this letter is enclosed**

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm or Individual name	PATRICIA A. VERLANGIERI, Reg. No. 42,201
Signature	Patricia A Verlangieri
Date	January 10, 2001

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be send to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

Received from < 732 530 9808 > at 1/10/02 4:06:40 PM [Eastern Standard Time]

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/608,872	06/30/2000	Christine Halversen	SRIlp037B	2382

7590 01/28/2002

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY, NJ 07702

EXAMINER	
BACKER, FIRMIN	
ART UNIT	PAPER NUMBER
2155	17

DATE MAILED: 01/28/2002

Please find below and/or attached an Office communication concerning this application or proceeding.

Advisory Action	**Application No.** *09/608,872*	**Applicant(s)** HALVERSEN ET AL.	
	Examiner Firmin Backer	**Art Unit** 2155	

--The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

THE REPLY FILED 17 January 2002 FAILS TO PLACE THIS APPLICATION IN CONDITION FOR ALLOWANCE. Therefore, further action by the applicant is required to avoid abandonment of this application. A proper reply to a final rejection under 37 CFR 1.113 may only be either: (1) a timely filed amendment which places the application in condition for allowance; (2) a timely filed Notice of Appeal (with appeal fee); or (3) a timely filed Request for Continued Examination (RCE) in compliance with 37 CFR 1.114.

PERIOD FOR REPLY [check either a) or b)]

a) ☒ The period for reply expires 3 months from the mailing date of the final rejection.

b) □ The period for reply expires on: (1) the mailing date of this Advisory Action, or (2) the date set forth in the final rejection, whichever is later. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of the final rejection. ONLY CHECK THIS BOX WHEN THE FIRST REPLY WAS FILED WITHIN TWO MONTHS OF THE FINAL REJECTION. See MPEP 706.07(f).

Extensions of time may be obtained under 37 CFR 1.136(a). The date on which the petition under 37 CFR 1.136(a) and the appropriate extension fee have been filed is the date for purposes of determining the period of extension and the corresponding amount of the fee. The appropriate extension fee under 37 CFR 1.17(a) is calculated from: (1) the expiration date of the shortened statutory period for reply originally set in the final Office action; or (2) as set forth in (b) above, if checked. *Any reply received by the Office later than three months after the mailing* date of the final rejection, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

1.□ A Notice of Appeal was filed on _____. Appellant's Brief must be filed within the period set forth in 37 CFR 1.192(a), or any extension thereof (37 CFR 1.191(d)), to avoid dismissal of the appeal.

2.☒ The proposed amendment(s) will not be entered because:

(a) ☒ they raise new issues that would require further consideration and/or search (see NOTE below);

(b) □ they raise the issue of new matter (see Note below);

(c) □ they are not deemed to place the application in better form for appeal by materially reducing or simplifying the issues for appeal; and/or

(d) □ they present additional claims without canceling a corresponding number of finally rejected claims.

NOTE: *See Continuation Sheet*.

3.□ Applicant's reply has overcome the following rejection(s): _____.

4.□ Newly proposed or amended claim(s) _____ would be allowable if submitted in a separate, timely filed amendment canceling the non-allowable claim(s).

5.□ The a)□ affidavit, b)□ exhibit, or c)□ request for reconsideration has been considered but does NOT place the application in condition for allowance because: _____.

6.□ The affidavit or exhibit will NOT be considered because it is not directed SOLELY to issues which were newly raised by the Examiner in the final rejection.

7.☒ For purposes of Appeal, the proposed amendment(s) a)□ will not be entered or b)□ will be entered and an explanation of how the new or amended claims would be rejected is provided below or appended.

The status of the claim(s) is (or will be) as follows:

Claim(s) allowed: _____.

Claim(s) objected to: _____.

Claim(s) rejected: *56-82*.

Claim(s) withdrawn from consideration: _____.

8.□ The proposed drawing correction filed on _____ is a)□ approved or b)□ disapproved by the Examiner.

9.□ Note the attached Information Disclosure Statement(s)(PTO-1449) Paper No(s). _____.

10.□ Other: _____

Continuation Sheet (PTO-303) **Application No. 09/608,872**

Continuation of 2. NOTE: The proposed amendments will not be entered because the raised new issue such as in claims 56 and 65 "wherein at least a portion of said data link between said mobile information appliance of the user and the one or more network utilizes wireless communication" that require further search and/or consideration .

AYAZ SHEIKH
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

Official
2/8/02

#18
2-8-02

PTO/SB/30 (08-00)
Approved for use through 10/31/2002. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

REQUEST FOR CONTINUED EXAMINATION (RCE) TRANSMITTAL

Subsection (b) of 35 U.S.C. § 132, effective on May 29, 2000, provides for continued examination of an utility or plant application filed on or after June 8, 1995. See The American Inventors Protection Act of 1999 (AIPA).

Application Number	09/608,872
Filing Date	June 30, 2000
Examiner Name	F. Backer
First Named Inventor	Halversen
Group Art Unit	2155
Attorney Docket Number	SRI 1P037B

This is a Request for Continued Examination (RCE) under 37 C.F.R. § 1.114 of the above-identified application.
NOTE: *37 C.F.R. § 1.114 is effective on May 29, 2000. If the above-identified application was filed prior to May 29, 2000, applicant may wish to consider filing a continued prosecution application (CPA) under 37 C.F.R. § 1.53 (d) (PTO/SB/29) instead of a RCE to be eligible* for the patent term adjustment provisions of the AIPA. See Changes to Application Examination and Provisional Application Practice, Interim Rule, 65 *Fed. Reg.* 14865 (Mar. 20, 2000), 1233 *Off. Gaz. Pat. Office* 47 (Apr. 11, 2000), which established RCE practice.

1. Submission required under 37 C.F.R. § 1.114
   - a. ☒ Previously submitted
     - i. ☒ Consider the amendment(s)/reply under 37 C.F.R. § 1.116 previously filed on 1/10/02
       (Any unentered amendment(s) referred to above will be entered).
     - ii. ☐ Consider the arguments in the Appeal Brief or Reply Brief previously filed on _____
     - iii. ☐ Other _____
   - b. Enclosed
     - i. ☐ Amendment/Reply
     - ii ☐ Affidavit(s)/Declaration(s)
     - iii. ☐ Information Disclosure Statement (IDS)
     - iv. ☐ Other _____
2. Miscellaneous
   - a. ☐ Suspension of action on the above-identified application is requested under 37 C.F.R. § 1.103(c) for a period of _____ months. (Period of suspension shall not exceed 3 months; Fee under 37 C.F.R. § 1.17(i) required)
   - b. ☒ Other Extension Request and Fee Transmittal Sheet
3. Fees The RCE fee under 37 C.F.R. § 1.17(e) is required by 37 C.F.R. § 1.114 when the RCE is filed.
   - a. ☒ The Director is hereby authorized to charge the following fees, or credit any overpayments, to Deposit Account No.20-0782
     - i. ☒ RCE fee required under 37 C.F.R. § 1.17(e)
     - ii. ☒ Extension of time fee (37 C.F.R. §§ 1.136 and 1.17)
     - iii. ☐ Other
   - b. ☐ Check in the amount of $_____ enclosed
   - c. ☐ Payment by credit card (Form PTO-2038 enclosed)

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT REQUIRED

Name (Print/Type)	KIN-WAH TONG	*Registration No. (Attorney/Agent)*	39,400
Signature		*Date*	February 8, 2002

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND Fees and Completed Forms to the following address: Commissioner for Patents, Box RCE, Washington, DC 20231.

U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a)	Docket Number (Optional) SRI 1P037B

In re Application of HALVERSEN	
Application Number 09/606,872	Filed June 30, 2000
For Mobile Navigation of Network-Based Electronic Information Using Spoken Input	
Group Art Unit 2155	Examiner F. Backer

This is a request under the provisions of 37 CFR 1.136(a) to extend the period for filing a response in the above identified application.

The requested extension and appropriate non-small-entity fee are as follows (check time period desired):

☒ One month (37 CFR 1.17(a)(1))	$110.00
□ Two months (37 CFR 1.17(a)(2))	$
□ Three months (37 CFR 1.17(a)(3))	$
□ Four months (37 CFR 1.17(a)(4))	$
□ Five months (37 CFR 1.17(a)(5))	$

☒ Applicant claims small entity status. See 37 CFR 1.27. Therefore, the fee amount shown above is reduced by one-half, and the resulting fee is: $ 55.00 .

□ A check in the amount of the fee is enclosed.

□ Payment by credit card. Form PTO-2038 is attached.

□ The Commissioner has already been authorized to charge fees in this application to a Deposit Account.

☒ The Commissioner is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account Number 20-0782 .

I have enclosed a duplicate copy of this sheet.

I am the □ applicant/inventor.

□ assignee of record of the entire interest. See 37 CFR 3.71

Statement under 37 CFR 3.73(b) is enclosed. (Form PTO/SB/96).

☒ attorney or agent of record.

□ attorney or agent under 37 CFR 1.34(a).

Registration number if acting under 37 CFR 1.34(a). ______ .

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

February 8, 2002
Date

Signature

KIN-WAH TONG
Typed or printed name

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below*.

□ *Total of ____ forms are submitted.

Burden Hour Statement: This form is estimated to take 0.1 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

[illegible]
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a)	Docket Number (Optional) SRI 1P037B

In re Application of HALVERSEN		
Application Number 09/608,872	Filed June 30, 2000	
For Mobile Navigation of Network-Based Electronic Information Using Spoken Input		
Group Art Unit 2155	Examiner F. Backer	

This is a request under the provisions of 37 CFR 1.136(a) to extend the period for filing a response in the above identified application.

The requested extension and appropriate non-small-entity fee are as follows (check time period desired):

☒ One month (37 CFR 1.17(a)(1))	$110.00
□ Two months (37 CFR 1.17(a)(2))	$
□ Three months (37 CFR 1.17(a)(3))	$
□ Four months (37 CFR 1.17(a)(4))	$
□ Five months (37 CFR 1.17(a)(5))	$

☒ Applicant claims small entity status. See 37 CFR 1.27. Therefore, the fee amount shown above is reduced by one-half, and the resulting fee is: $ 55.00 .

□ A check in the amount of the fee is enclosed.

□ Payment by credit card. Form PTO-2038 is attached.

□ The Commissioner has already been authorized to charge fees in this application to a Deposit Account.

☒ The Commissioner is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account Number 20-0782 .
I have enclosed a duplicate copy of this sheet.

I am the □ applicant/inventor.

□ assignee of record of the entire interest. See 37 CFR 3.71
Statement under 37 CFR 3.73(b) is enclosed. (Form PTO/SB/96).

☒ attorney or agent of record.

□ attorney or agent under 37 CFR 1.34(a).
Registration number if acting under 37 CFR 1.34(a). ______ .

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

February 8, 2002
Date

Signature

KIN-WAH TONG
Typed or printed name

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below*.

□ *Total of ____ forms are submitted.

Burden Hour Statement: This form is estimated to take 0.1 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

FEE TRANSMITTAL for FY 2002

Patent fees are subject to annual revision.

Complete If Known	
Application Number	09/608,672
Filing Date	June 30, 2000
First Named Inventor	Halversen
Examiner Name	F. Backer
Group / Art Unit	2165
Attorney Docket No.	SRI 1P037B

TOTAL AMOUNT OF PAYMENT ($) 425

METHOD OF PAYMENT (check one)

1. ☒ The Commissioner is hereby authorized to charge indicated fees and credit any over payments to:

Deposit Account Number	20-0782
Deposit Account Name	

☒ Charge Any Additional Fee Required Under 37 CFR 1.16 and 1.17

☒ Applicant claims small entity status. See 37 CFR 1.27

2. ☐ Payment Enclosed:

☐ Check ☐ Credit card ☐ Money Order ☐ Other

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
101	740	201	370	Utility filing fee	
106	330	206	165	Design filing fee	
107	510	207	255	Plant filing fee	
108	740	208	370	Reissue filing fee	
114	160	214	80	Provisional filing fee	

SUBTOTAL (1) ($) 0

2. EXTRA CLAIM FEES

			Extra Claims		Fee from below		Fee Paid
Total Claims		-20 ** =	0	X		=	0
Independent Claims		-3 ** =	0	X		=	0
Multiple Dependent				X		=	0

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
103	18	203	9	Claims in excess of 20
102	84	202	42	Independent claims in excess of 3
104	280	204	140	Multiple dependent claim, if not paid
109	84	209	42	** Reissue independent claims over original patent
110	18	210	9	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) ($) 0

**or number previously paid, if greater; For Reissues, see above

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
105	130	205	65	Surcharge - late filing fee or oath	
127	50	227	25	Surcharge - late provisional filing fee or cover sheet.	
139	130	139	130	Non-English specification	
147	2,520	147	2,520	For filing a request for reexamination	
112	920*	112	920*	Requesting publication of SIR prior to Examiner action	
113	1,840*	113	1,840*	Requesting publication of SIR after Examiner action	
115	110	215	55	Extension for reply within first month	55.00
116	400	216	200	Extension for reply within second month	
117	920	217	460	Extension for reply within third month	
118	1,440	218	720	Extension for reply within fourth month	
128	1,960	228	980	Extension for reply within fifth month	
119	320	219	160	Notice of Appeal	
120	320	220	160	Filing a brief in support of an appeal	
121	280	221	140	Request for oral hearing	
138	1,510	138	1,510	Petition to institute a public use proceeding	
140	110	240	55	Petition to revive – unavoidable	
141	1,280	241	640	Petition to revive – unintentional	
142	1,280	242	640	Utility issue fee (or reissue)	
143	460	243	230	Design issue fee	
144	620	244	310	Plant issue fee	
122	130	122	130	Petitions to the Commissioner	
123	50	123	50	Processing fee under 37 CFR 1.17 (q)	
126	180	126	180	Submission of Information Disclosure Stmt	
581	40	581	40	Recording each patent assignment per property (times number of properties)	
146	740	246	370	Filing a submission after final rejection (37 CFR § 1.129(a))	
149	740	249	370	For each additional invention to be examined (37 CFR § 1.129(b))	
179	740	279	370	Request for Continued Examination (RCE)	370.00
169	900	169	900	Request for expedited examination of a design application	

Other fee (specify)

*Reduced by Basic Filing Fee Paid SUBTOTAL (3) ($) 425

SUBMITTED BY				Complete *(if applicable)*	
Name (Print/Type)	KIN-WAH TONG	*Registration No. Attorney/Agent)*	38,400	*Telephone*	(732)530-9404
Signature				*Date*	FEBRUARY 8, 2002

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

TELEFAX COVER SHEET

MOSER, PATTERSON & SHERIDAN, LLP
ATTORNEYS AT LAW
595 SHREWSBURY AVENUE
FIRST FLOOR
SHREWSBURY, NJ 07702
TELEPHONE (732) 530-9404
TELEFAX (732) 530-9808

****

THIS TELEFAX MESSAGE IS ADDRESSED TO THE PERSON OR COMPANY LISTED BELOW. IF IT WAS SENT OR RECEIVED INCORRECTLY, OR YOU ARE NOT THE INTENDED RECIPIENT, PLEASE TAKE NOTICE THAT THIS MESSAGE MAY CONTAIN PRIVILEGED OR CONFIDENTIAL MATERIAL, AND YOUR DUE REGARD FOR THIS INFORMATION IS NECESSARY. YOU MAY ARRANGE TO RETURN THIS MATERIAL BY CALLING THE FIRM LISTED ABOVE AT (732) 530-9404

****

THIS MESSAGE HAS 6 PAGES INCLUDING THIS SHEET

TO: Assistant Commissioner of Patents

FAX NO.: 703-746-7238

FROM: Kin-Wah Tong

DATE: February 8, 2002

MATTER: Serial No. 09/608,872 Filed: June 30, 2000

DOCKET NO.: SRI 1P037B

APPLICANT: HALVERSON, et al

The following has been received in the U.S. Patent and Trademark Office on the date of this facsimile:

___ Petition
___ Disclosure Statement & PTO-1449
___ Priority Document
___ Drawings (____ sheets) informal
X Petition for Extension of Time (2 copies)

X RCE Transmittal Letter
X Fee Transmittal (2 copies)
X Deposit Account Transaction
X Facsimile Transmission Certificate dated February 8, 2002

CERTIFICATE OF TRANSMISSION UNDER 37 C.F.R. §1.6

I hereby certify that this correspondence is being transmitted by facsimile to the Assistant Commissioner for Patents, Box AF, Washington, DC 20231 on February 8, 2002, Facsimile No. 703-746-7238.

Linda DeNardi
Name of person signing this certificate

Linda DeNardi February 8, 2002
Signature and date

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
[illegible]
Washington, D.C. 20231
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO	CONFIRMATION NO
09/608,872	06/30/2000	Christine Halversen	SRI1p037B	2382

7590 02/19/2002

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY, NJ 07702

EXAMINER	
BACKER, FIRMIN	
ART UNIT	PAPER NUMBER
2155	19

DATE MAILED: 02/19/2002

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	**Application No.** 09/608,872	**Applicant(s)** HALVERSEN ET AL.
	Examiner Firmin Backer	**Art Unit** 2155

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136 (a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 08 February 2002 .

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 56-82 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 56-82 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claims _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are objected to by the Examiner.

11)☐ The proposed drawing correction filed on _____ is: a)☐ approved b)☐ disapproved.

12)☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. § 119

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____ .

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgement is made of a claim for domestic priority under 35 U.S.C. § 119(e).

Attachment(s)

15)☐ Notice of References Cited (PTO-892)

16)☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

17)☐ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____ .

18)☐ Interview Summary (PTO-413) Paper No(s). _____ .

19)☐ Notice of Informal Patent Application (PTO-152)

20)☐ Other:

Continued Examination Under 37 CFR 1.114

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on February 8th, 2002 has been entered.

Double Patenting

2. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and, *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

3. Claims 56-82 are provisionally rejected under the judicially created doctrine of double patenting over claims 56-126 of copending Application No. 09/524,095. This is a provisional double patenting rejection since the conflicting claims have not yet been patented.

The subject matter claimed in the instant application is fully disclosed in the referenced copending application and would be covered by any patent granted on that copending application since the referenced copending application and the instant application are claiming common subject matter, as follows. Although the conflicting claims are not identical, they are not patentably distinct from each other because it would have been obvious to one of ordinary skill in the art to observed that the omission of the limitations "**soliciting additional input from the user, including user interaction in a modality different that the original request and, refining the navigation query, based upon the additional input**", of applicant claims 56-82 are already in the Co-pending application 09/524,095, as such they are obvious variation of the inventive concept defined in claims 56-126 of the Co-pending application 09/524,095. See In re Karlson, 136USPQ 184 (CCPA 1963). This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

Claim Rejections - 35 USC § 102

4. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless --
>
> (e) the invention was described in a patent granted on an application for patent by another filed in the United States before the invention thereof by the applicant for patent, or on an international application by another who has fulfilled the requirements of paragraphs (1), (2), and (4) of section 371(c) of this title before the invention thereof by the applicant for patent._

5. Claims 56-82 are rejected under 35 U.S.C. 102(e) as being anticipated by Levin et al. (U.S. Patent No. 6,173,279).

6. As per claim 56, Levin et al teach a method for speech-based navigation (*information server, 110*) of an electronic data source located at one or more network servers located remotely from a user, wherein at least a portion of a data link between a mobile information appliance of the user and the one or more network servers utilizes wireless communication (*see abstract, fig*

1, column 3 lines 5-35), comprising receiving a spoken request (*receive a natural language query*) for desired information from the user (*user, 112*) utilizing the mobile information appliance (*PC, 102*) of the user; rendering an interpretation (*creating a semantic representation*) of the spoken request, constructing a navigation (*generating search)* query based upon the interpretation; utilizing the navigation query to select a portion of the electronic data source; and transmitting (*sending*) the selected portion of the electronic data source from the network server to the mobile information appliance of the user. (*see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22)*

7. As per claim 57, 58, 62-64, Levin et al teach a method of rendering the interpretation of the spoken request is performed at the one or more network servers by the mobile information appliance including a wireless telephone, a portable computer that is a personal digital assistance (*see abstract, fig 1, column 3 lines 5-35*).

8. As per claim 59, Levin et al teach a method of soliciting additional input from the user, including user interaction in a modality different than the original request; refining the navigation query, based upon the additional input; and using the refined navigation query to select a portion of the electronic data source (*see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22*).

9. As per claim 60, Levin et al teach a method wherein the data link includes a cellular telephone system (*see fig 1, column 2 line 61-67*).

10. As per claim 61, Levin et al teach a method wherein steps (a)-(d) are performed with respect to multiple users (*see abstract, fig 1, column 3 lines 5-35*).

11. As per claim 65, Levin et al teach a computer system for speech-based navigation (*information server, 110*) of an electronic data source located at one or more network servers located remotely from a user, wherein at least a portion of a data link between a mobile information appliance of the user and the one or more network servers utilizes wireless communication (*see abstract, fig 1, column 3 lines 5-35*), comprising a code segment receiving a spoken request (*receive a natural language query*) for desired information from the user (*user*) utilizing the mobile information appliance (*PC, 102*) of the user; a code segment rendering an interpretation (*creating a semantic representation*) of the spoken request, a code segment constructing a navigation (*generating search)* query based upon the interpretation; a code segment utilizing the navigation query to select a portion of the electronic data source; and a code segment transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user. (*see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22*).

12. As per claim 66, 67, 71-73, Levin et al teach a system of rendering the interpretation of the spoken request is performed at the one or more network servers by the mobile information appliance including a wireless telephone, a portable computer that is a personal digital assistance (*see abstract, fig 1, column 3 lines 5-35*).

13. As per claim 68, Levin et al teach a system of soliciting additional input from the user, including user interaction in a modality different than the original request; refining the navigation query, based upon the additional input; and using the refined navigation query to select a portion of the electronic data source (*see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22*).

14. As per claim 69, Levin et al teach a system wherein the data link includes a cellular telephone system (*see fig 1, column 2 line 61-67*).

15. As per claim 70, Levin et al teach a system wherein steps (a)-(d) are performed with respect to multiple users *(see abstract, fig 1, column 3 lines 5-35*).

16. As per claim 74, Levin et al teach a system for speech-based navigation (*information server, 110*) of an electronic data source located at one or more network servers located remotely from a user, wherein at least a portion of a data link between a mobile information appliance of the user and the one or more network servers utilizes wireless communication (*see abstract, fig 1, column 3 lines 5-35*), comprising receiving a spoken request (*receive a natural language query*) for desired information from the user (*user*) utilizing the mobile information appliance (*PC, 102*) of the user; rendering an interpretation (*creating a semantic representation*) of the spoken request, constructing a navigation (*generating search)* query based upon the interpretation; utilizing the navigation query to select a portion of the electronic data source; and transmitting the selected portion of the electronic data source from the network server to the

mobile information appliance of the user. (*see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22*)

17. As per claim 75, 76, 80-81, Levin et al teach a method of rendering the interpretation of the spoken request is performed at the one or more network servers by the mobile information appliance including a wireless telephone, a portable computer that is a personal digital assistance (*see abstract, fig 1, column 3 lines 5-35*).

18. As per claim 77, Levin et al teach a system of soliciting additional input from the user, including user interaction in a modality different than the original request; refining the navigation query, based upon the additional input; and using the refined navigation query to select a portion of the electronic data source (*see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22*).

19. As per claim 78, Levin et al teach a system wherein the data link includes a cellular telephone system (*see fig 1, column 2 line 61-67*).

20. As per claim 79, Levin et al teach a system wherein steps (a)-(d) are performed with respect to multiple users (*see abstract, fig 1, column 3 lines 5-35*).

Response to Arguments

21. Applicant's arguments filed on September 26th, 2001 have been fully considered but they are not persuasive.

a. Applicant argues that the prior art "fails to teach or suggest the novel concept of speech-based navigation where the method receives spoken request for desired information from the user utilizing the mobile information appliance of the user and where in turn the selected electronic data source from the network server is transmitted to the mobile information appliance of the user." Examiner respectfully disagrees with the applicant perspective and characterization of Levin inventive concept. Levin teach that the URL for a data resource is inputted into PC 102 either by typing the request using a keyboard 104 or ***by speaking the request into a microphone 105,*** which is considered to be a mobile appliance of the user. Furthermore, Levin et al indicate that the spoken requests either from a PC microphone 105 or from a telephone 103 can be handled by a speech recognition system residing at the information server (*see column 4 lines 7-22*). Applicant further argues that the prior art "fails to teach or suggest that the selected electronic data source from the network server is transmitted to the mobile information appliance of the user." Examiner respectfully disagrees with the applicant perspective and characterization of Levin inventive concept. Levin teach that once an information server is accessed, the user can send a text or a spoken query requesting a particular action or service (step 204), for example: "call the pizza place on Main Street in Westfield". The query is received by the access server 106 and the natural language query is sent to the information server 110 via packet network 108. It is to be understood that the packet network 108 may be connected to a plurality of information servers which each relate to one or more particular information services, or there may be a single centralized information server 110 which is accessed by all information services which are capable of receiving and processing natural language queries and contains at least some of the data resources (e.g., URLs and associated site/service-specific grammars) *capable of receiving and responding to a natural language query.* It is obvious inventive concept referring to *response* is in the field of sending or transmitting the requested information to the user. Moreover, it is understood in the art of information request, in order to complete the transaction, the host must transmit to the requester the requested information.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Firmin Backer whose telephone number is 703-305-0624. The examiner can normally be reached on Mon-Thu 8:30-6:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Sheikh Ayaz can be reached on 703-305-9648. The fax phone numbers for the organization where this application or proceeding is assigned are 703-746-7239 for regular communications and 703-746-7238 for After Final communications.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is 703-305-3900.

Firmin Backer
February 14, 2002

AYAZ SHEIKH
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

UNITED STAT DEPARTMENT OF COMMERCE
Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231

SERIAL NUMBER	FILING DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKETT NO.

EXAMINER

ART UNIT	PAPER NUMBER
	20

DATE MAILED:

EXAMINER INTERVIEW SUMMARY RECORD

All participants (applicant, applicant's representative, PTO personnel):

(1) David Wiley (3) ______

(2) Kin-wah Tong 39,400 (4) ______

Date of interview 5/23/2002

Type: □ Telephonic ☒ Personal (copy is given to □ applicant □ applicant's representative).

Exhibit shown or demonstration conducted: □ Yes ☒ No. If yes, brief description: ______

Agreement ☒ was reached with respect to some or all of the claims in question. □ was not reached.

Claims discussed: 56-82

Identification of prior art discussed: Levin et al.

Description of the general nature of what was agreed to if an agreement was reached, or any other comments: The applicant agreed to amend the claims to further identify the mobile device to overcome the Levin reference.

(A fuller description, if necessary, and a copy of the amendments, if available, which the examiner agreed would render the claims allowable must be attached. Also, where no copy of the amendments which would render the claims allowable is available, a summary thereof must be attached.)

☒ 1. It is not necessary for applicant to provide a separate record of the substance of the interview.

Unless the paragraph below has been checked to indicate to the contrary, A FORMAL WRITTEN RESPONSE TO THE LAST OFFICE ACTION IS NOT WAIVED AND MUST INCLUDE THE SUBSTANCE OF THE INTERVIEW (e.g., items 1-7 on the reverse side of this form). *If a response to the last Office action has already been filed, then applicant is given one month from this interview date to provide a statement of the substance of the interview.*

□ 2. Since the examiner's interview summary above (including any attachments) reflects a complete response to each of the objections, rejections and *requirements that may be present in the last Office action*, and since the claims are now allowable, this completed form is considered to fulfill the response requirements of the last Office action. Applicant is not relieved from providing a separate record of the substance of the interview unless box 1 above is also checked.

Examiner's Signature

PTOL-413 (REV. 2-93)

ORIGINAL FOR INSERTION IN RIGHT HAND FLAP OF FILE WRAPPER

U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a)	Docket Number (Optional) SRI 1P037B

In re Application of Halverson, et al	
Application Number 09/608,872	Filed June 30, 2000
For Mobile Navigation of Network-Based Electronic Information Using Spoken Output	
Group Art Unit 2155	Examiner F. Backer

#21
S. Collins
9/18/02

This is a request under the provisions of 37 CFR 1.136(a) to extend the period for filing a response in the above identified application.

The requested extension and appropriate non-small-entity fee are as follows (check time period desired):

□ One month (37 CFR 1.17(a)(1))	$
☒ Two months (37 CFR 1.17(a)(2))	$400.00
□ Three months (37 CFR 1.17(a)(3))	$
□ Four months (37 CFR 1.17(a)(4))	$
□ Five months (37 CFR 1.17(a)(5))	$

☒ Applicant claims small entity status. See 37 CFR 1.27. Therefore, the fee amount shown above is reduced by one-half, and the resulting fee is: $ 200.00 .

□ A check in the amount of the fee is enclosed.

□ Payment by credit card. Form PTO-2038 is attached.

□ The Commissioner has already been authorized to charge fees in this application to a Deposit Account.

☒ The Commissioner is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account Number 20-0782 .

I have enclosed a duplicate copy of this sheet.

I am the □ applicant/inventor.

□ assignee of record of the entire interest. See 37 CFR 3.71
Statement under 37 CFR 3.73(b) is enclosed. (Form PTO/SB/96).

☒ attorney or agent of record.

□ attorney or agent under 37 CFR 1.34(a).

Registration number if acting under 37 CFR 1.34(a). ______ .

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

July 17, 2002
Date

Signature

Kin-Wah Tong
Typed or printed name

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below*.

□ *Total of ____ forms are submitted.

Burden Hour Statement: This form is estimated to take 0.1 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

MISSING PAGE(S) FROM THE U.S. PATENT OFFICE OFFICIAL FILE WRAPPER

Facsimile Page 4

(Note: This page is not a part of the official USPTO record.)

09/608,872

IN THE UNITED STATES
PATENT AND TRADEMARK OFFICE

#22/D
S. Catton
7-18-02

PATENT APPLICATION

Applicant: **Halverson et al.**

Case: **SRI1P037B**

Serial No.: **09/608,872** Filed: **June 30, 2000**

Group Art Unit: **2155**

Examiner: **Firmin Backer**

Title: **MOBILE NAVIGATION OF NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN INPUT**

ASSISTANT COMMISSIONER FOR PATENTS
Box Non-Fee Amendment
Washington, D. C. 20231

S I R:

AMENDMENT AND RESPONSE UNDER 37 C.F.R. § 1.111

This amendment addresses the Office Action dated February 19, 2002 (Paper No. 19).

IN THE CLAIMS

Please amend claims 56, 65 and 74 as shown below. These claims are "clean version" of the amended claims, i.e., with changes incorporated into the claims, whereas the Appendix to this Amendment illustrates the amended claims using underlines and brackets to indicate addition and deletion, respectively.

1 ~~56.~~ (Twice Amended) A method for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, wherein a data link is established between a mobile information appliance of the user and the one or more network servers, comprising the steps of:

31

34

D

09/608,872

(a) receiving a spoken request for desired information from the user utilizing the mobile information appliance of the user, wherein said mobile information appliance comprises a portable remote control device or a set-top box for a television;

N1

(b) rendering an interpretation of the spoken request;

(c) constructing a navigation query based upon the interpretation;

(d) utilizing the navigation query to select a portion of the electronic data source; and

(e) transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user.

10 ~~65~~. (Twice Amended) A computer program embodied on a computer readable medium for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, wherein a data link is established between a mobile information appliance of the user and the one or more network servers, comprising:

(a) a code segment that receives a spoken request for desired information from the user utilizing the mobile information appliance of the user, wherein said mobile information appliance comprises a portable remote control device or a set-top box for a television;

N2

(b) a code segment that renders an interpretation of the spoken request;

(c) a code segment that constructs a navigation query based upon the interpretation;

(d) a code segment that utilizes the navigation query to select a portion of the electronic data source; and

(e) a code segment that transmits the selected portion of the electronic data source from the network server to the mobile information appliance of the user.

19 ~~74~~. (Amended) A system for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, comprising:

N3

(a) a mobile information appliance operable to receive a spoken request for desired information from the user, wherein said mobile information appliance comprises

35

D

a portable remote control device or a set-top box for a television;

(b) spoken language processing logic, operable to render an interpretation of the spoken request;

(c) query construction logic, operable to construct a navigation query based upon the interpretation;

A3

(d) navigation logic, operable to select a portion of the electronic data source using the navigation query, and

(e) electronic communications infrastructure for transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user.

REMARKS

Applicants' representative would like to thank Primary Examiner David Wiley for kindly taking a substantial amount of time on May 23, 2002 to discuss the merits of the subject invention in a face-to-face Examiner Interview. Applicants' representative is aware of the time constraint that is placed on the Examiner and is appreciative of the Examiner's willingness to devote such large quantity of time to discuss the case on the merit.

In view of the following discussion, the Applicants submit that none of the claims now pending in the application are anticipated under the provisions of 35 U.S.C. § 102. Thus, the Applicants believe that all of these claims are now in allowable form.

I. REJECTION OF CLAIMS 56-82 UNDER DOUBLE PATENTING

The Examiner provisionally rejected claims 56-82 in Paragraphs 2-3 of the Office Action based on the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 56-126 of copending Application No. 09/524,095.

Responsive to the Examiner, Applicants provisionally agree to file a terminal disclaimer to resolve the present judicially created doctrine of obviousness-type double patenting rejection if and when one of the applications is finally allowed. In accordance with MPEP 804 I.B, "if the 'provisional' double patenting rejection in one application is

36

D

the only rejection remaining in that application, the examiner should then withdraw that rejection and permit the application to issue as a patent, thereby converting the 'provisional' doubling patenting rejection in the other application(s) into a double patenting rejection at the time the one application issues as a patent". As such, Applicants will file a terminal disclaimer in the future, if necessary.

II. REJECTION OF CLAIMS 56-82 UNDER 35 U.S.C. § 102

The Examiner has again rejected claims 56-82 in Paragraphs 4-20 of the Office Action as being anticipated by the Levin et al. patent (US Patent 6,173,279 issued January 9, 2001, hereinafter referred to as Levin). The rejection is respectfully traversed.

Levin teaches "a method of using at least one natural language query to retrieve information from one or more data resources and further performing a requested action using the retrieved information is disclosed". (See Levin, Column 2, lines 15-18) Namely, Levin teaches a method for using natural language query to obtain information, where upon receipt of the requested information, a desired action is executed based upon the requested information. To illustrate, Levin provides the example, where a user employs natural language to request the telephone number of a restaurant. Upon receipt of the telephone number, the telephone number is actually dialed for the user. (See Levin, Column 3 line 62 to Column 4, line 1)

In contrast, Levin fails to teach or suggest the novel concept of speech-based navigation where the method receives spoken request for desired information from the user utilizing the mobile information appliance of the user, wherein said mobile information appliance comprises a portable remote control device or a set-top box for a television. Specifically, Applicants' independent claims 56, 65 and 74 positively recite:

> 56. A method for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, wherein a data link is established between a mobile information appliance of the user and the one or more network servers, comprising the steps of:
>
> (a) receiving a spoken request for desired information from the user utilizing the mobile information appliance of the user, wherein said mobile

information appliance comprises a portable remote control device or a set-top box for a television;

(b) rendering an interpretation of the spoken request;

(c) constructing a navigation query based upon the interpretation;

(d)utilizing the navigation query to select a portion of the electronic data source; and

(e) transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user. (emphasis added)

65. A computer program embodied on a computer readable medium for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, wherein a data link is established between a mobile information appliance of the user and the one or more network servers, comprising:

(a) a code segment that receives a spoken request for desired information from the user utilizing the mobile information appliance of the user, wherein said mobile information appliance comprises a portable remote control device or a set-top box for a television;

(b) a code segment that renders an interpretation of the spoken request;

(c) a code segment that constructs a navigation query based upon the interpretation;

(d) a code segment that utilizes the navigation query to select a portion of the electronic data source; and

(e) a code segment that transmits the selected portion of the electronic data source from the network server to the mobile information appliance of the user. (emphasis added)

74. A system for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, comprising:

(a) a mobile information appliance operable to receive a spoken request for desired information from the user, wherein said mobile information appliance comprises a portable remote control device or a set-top box for a television;

(b) spoken language processing logic, operable to render an interpretation of the spoken request;

(c) query construction logic, operable to construct a navigation query based upon the interpretation;

(d) navigation logic, operable to select a portion of the electronic data source using the navigation query, and

(e) electronic communications infrastructure for transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user. (emphasis added)

D

Applicants' invention teaches a novel method and apparatus for speech-based navigation where the method receives spoken request for desired information from the user utilizing the mobile information appliance of the user, wherein said mobile information appliance comprises a portable remote control device or a set-top box for a television. This teaching is completely absent in the Levin reference.

During the Examiner Interview, Primary Examiner David Wiley indicated that a specific identification of the mobile information appliance that comprises a portable remote control device or a set-top box for a television would likely overcome the Levin reference.

Therefore, the Applicants respectfully submit that independent claims 56, 65 and 74 are not anticipated by the Levin reference. As such, claims 56, 65 and 74 fully satisfy the requirements of 35 U.S.C. §102 and are patentable thereunder.

Claims 57-64, 66-73 and 75-82 depend, either directly or indirectly, from claims 56, 65 and 74 and recite additional features therefor. Since Levin fails to anticipate Applicants' invention as recited in Applicants' independent claims 56, 65 and 74, dependent claims 57-64, 66-73 and 75-82 are also not anticipated under 35 U.S.C. § 102 and are allowable for the same reason noted above.

Conclusion

Thus, the Applicants submit that all of these claims now fully satisfy the requirements of 35 U.S.C. §102. Consequently, the Applicants believe that all these claims are presently in condition for allowance. Accordingly, both reconsideration of this application and its swift passage to issue are earnestly solicited.

If, however, the Examiner believes that there are any unresolved issues requiring the issuance of a final action in any of the claims now pending in the application, it is requested that the Examiner telephone Mr. Kin-Wah Tong, Esq. at (732) 530-9404 so that appropriate arrangements can be made for resolving such issues as expeditiously as possible.

D

09/608,872

Respectfully submitted,

7/17/02

Kin-Wah Tong, Attorney
Reg. No. 39,400
(732) 530-9404

Moser, Patterson & Sheridan, LLP
595 Shrewsbury Avenue
First Floor,
Shrewsbury, New Jersey 07702

D

Appendix
(Marked-up copy of amended claims)

56. (Twice Amended) A method for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, wherein a data link is established between a mobile information appliance of the user and the one or more network servers, comprising the steps of:

(a) receiving a spoken request for desired Information from the user utilizing the mobile information appliance of the user, wherein said mobile information appliance comprises a portable remote control device or a set-top box for a television;

(b) rendering an interpretation of the spoken request;

(c) constructing a navigation query based upon the interpretation;

(d)utilizing the navigation query to select a portion of the electronic data source; and

(e) transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user[, wherein at least a portion of said data link between said mobile information appliance of the user and the one or more network servers utilizes wireless communication].

65. (Twice Amended) A computer program embodied on a computer readable medium for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, wherein a data link is established between a mobile information appliance of the user and the one or more network servers, comprising:

(a) a code segment that receives a spoken request for desired Information from the user utilizing the mobile information appliance of the user, wherein said mobile information appliance comprises a portable remote control device or a set-top box for a television;

(b) a code segment that renders an interpretation of the spoken request.

(c) a code segment that constructs a navigation query based upon the interpretation;

D

(d) a code segment that utilizes the navigation query to select a portion of the electronic data source; and

(e) a code segment that transmits the selected portion of the electronic data source from the network server to the mobile information appliance of the user[, wherein at least a portion of said data link between said mobile information appliance of the user and the one or more network servers utilizes wireless communication].

74. (Amended) A system for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, comprising:

(a) a mobile information appliance operable to receive a spoken request for desired information from the user, wherein said mobile information appliance comprises a portable remote control device or a set-top box for a television;

(b) spoken language processing logic, operable to render an interpretation of the spoken request;

(c) query construction logic, operable to construct a navigation query based upon the interpretation;

(d) navigation logic, operable to select a portion of the electronic data source using the navigation query, and

(e) electronic communications infrastructure for transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user[, wherein at least a portion of a data link of the electronic communications infrastructure between a mobile information appliance of the user and the one or more network servers utilizes wireless communication].

D

TELEFAX COVER SHEET

MOSER, PATTERSON & SHERIDAN, LLP

ATTORNEYS AT LAW
595 SHREWSBURY AVENUE
FIRST FLOOR
SHREWSBURY, NJ 07702
TELEPHONE (732) 530-9404
TELEFAX (732) 530-9808

****

THIS TELEFAX MESSAGE IS ADDRESSED TO THE PERSON OR COMPANY LISTED BELOW. IF IT WAS SENT OR RECEIVED INCORRECTLY, OR YOU ARE NOT THE INTENDED RECIPIENT, PLEASE TAKE NOTICE THAT THIS MESSAGE MAY CONTAIN PRIVILEGED OR CONFIDENTIAL MATERIAL, AND YOUR DUE REGARD FOR THIS INFORMATION IS NECESSARY. YOU MAY ARRANGE TO RETURN THIS MATERIAL BY CALLING THE FIRM LISTED ABOVE AT (732) 530-9404

****

THIS MESSAGE HAS 13 PAGES INCLUDING THIS SHEET

TO: Assistant Commissioner of Patents

FAX NO.: 703-746-7239

FROM: Kin-Wah Tong

DATE: July 17, 2002

MATTER: Serial No. 09/608,872 Filed: June 30, 2000

DOCKET NO.: SRI 1P037B

APPLICANT: HALVERSON, et al

The following has been received in the U.S. Patent and Trademark Office on the date of this facsimile:

___ Petition
___ Disclosure Statement & PTO-1449
___ Priority Document
___ Drawings (___ sheets) informal
X Petition for Extension of Time (2 copies)
X Amendment and Response
___ ______

X Transmittal Letter
___ Fee Transmittal (2 copies)
X Deposit Account Transaction
X Facsimile Transmission Certificate dated July 17, 2002
___ ______
___ ______

CERTIFICATE OF TRANSMISSION UNDER 37 C.F.R. §1.8

I hereby certify that this correspondence is being transmitted by facsimile to the Assistant Commissioner for Patents, Box Non-Fee Amendment, Washington, DC 20231 on July 17, 2002.
Facsimile No. 703-746-7239.

Linda DeNardi
Name of person signing this certificate

Linda DeNardi July 17, 2002
Signature and date

D

Official

PTO/SB/21 (08-00)
Approved for use through 10/31/2002. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

Please type a plus sign (+) inside this box → +

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	09/608,872
Filing Date	June 30, 2000
First Named Inventor	HALVERSON
Group Art Unit	2155
Examiner Name	F. BACKER
Attorney Docket Number	SRI 1 P 037B

Total Number of Pages in This Submission: 13

ENCLOSURES *(check all that apply)*

☐ Fee Transmittal Form
☐ Fee Attached
☒ Amendment / Response
☐ After Final
☐ Affidavits/declaration(s)
☒ Extension of Time Request
☐ Express Abandonment Request
☐ Information Disclosure Statement
☐ Certified Copy of Priority Document(s)
☐ Response to Missing Parts/ Incomplete Application
☐ Response to Missing Parts under 37 CFR 1.52 or 1.53

☐ Assignment Papers *(for an Application)*
☐ Drawing(s)
☐ Licensing-related Papers
☐ Petition
☐ Petition to Convert to a Provisional Application
☐ Power of Attorney, Revocation Change of Correspondence Address
☐ Terminal Disclaimer
☐ Request for Refund
☐ CD, Number of CD(s)

☐ After Allowance Communication to Group
☐ Appeal Communication to Board of Appeals and Interferences
☐ Appeal Communication to Group *(Appeal Notice, Brief, Reply Brief)*
☐ Proprietary Information
☐ Status Letter
☐ Other Enclosure(s) *(please identify below)*:

Certificate of Facsimile Transmission

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm or Individual name	KIN-WAH TONG, ESQ., Reg. No. 39,400
Signature	
Date	July 17, 2002

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be send to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

D

2155

TRANSMITTAL OF INFORMATION DISCLOSURE STATEMENT **Under 37 CFR 1.97(b), (c), or (d)**	Docket No. **SRI1P037B**

In re Application of: **Halverson, et al.**

Serial No.	Filing Date	Examiner	Group Art Unit
09/608,872	**June 30, 2000**	***Firmin Backer***	**2155**

Title: **MOBILE NAVIGATION OF NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN INPUT**

Address to:
Assistant Commissioner for Patents
Washington, D.C. 20231

37 CFR 1.97(b)

1. □ The Information Disclosure Statement submitted herewith is being filed within three months of the filing of a national application other than a continued prosecution application under 37 CFR 1.53(d); within three months of the date of entry of the national stage as set forth in 37 CFR 1.491 in an international application; before the mailing of a first Office Action on the merits; or before the mailing of a first Office Action after the filing of a request for continued examination under 37 CFR 1.114.

37 CFR 1.97(c)

2. X The Information Disclosure Statement submitted herewith is being filed after the period specified in 37 CFR 1.97(b), but prior to the mailing date of a Final Action under 37 CFR 1.113, a Notice of Allowance under 37 CFR 1.311, or an Action that otherwise closes prosecution in the application, and is accompanied by the statement or fee as indicated below.

37 CFR 1.97(d)

3. □ The Information Disclosure Statement submitted herewith is being filed after the period specified in 37 CFR 1.97(c), but on or before payment of the issue fee and is accompanied by the statement and fee as indicated below.

Required Statements and/or Fees Under 37 CFR 1.97(c) or (d)

X Each item of information contained in the accompanying Information Disclosure Statement was first cited in any communication from a foreign patent office in a counterpart foreign application not more than three months prior to the filing of the Information Disclosure Statement. (37 CFR 1.97(e)(1))

□ No item of information in the accompanying Information Disclosure Statement was cited in a communication from a foreign patent office in a counterpart foreign application, and, to the knowledge of the undersigned person, after making reasonable inquiry, no item of information contained in the accompanying Information Disclosure Statement was known to any individual designated in 37 CFR 1.56(c) more than three months prior to the filing of the Information Disclosure Statement. (37 CFR 1.97(e)(2))

□ The fee set forth in 37 CFR 1.17(p). Please credit any overpayment or charge any insufficiencies to deposit account number 20-0782.

37 CFR §1.704(d)

4. X Each item of information in the accompanying Information Disclosure Statement was cited in a communication from a foreign patent office in a counterpart application and this communication was not received by any individual designated in 37 CFR §1.56(c) more than thirty days prior to the filing of the Information Disclosure Statement.

Dated: July 23 2002

Kin-Wah Tong, Attorney
Reg. No. 39,400

Moser, Patterson & Sheridan, LLP
Attorneys at Law
595 Shrewsbury Avenue, Suite 100
Shrewsbury, New Jersey 07702
732-530-9404

Certificate of Mailing by First Class Mail

I certify that this document is being deposited on July 23, 2002 with the U.S. Postal Service as first class mail under 37 CFR §1.8 and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231

Barbara J. Jackson
Signature of Person Mailing Correspondence

Barbara J. Jackson
Typed or Printed Name of Person Mailing Correspondence

Sheet 1 of 1 sheet(s)

U.S. Department of Commerce, Patent and Trademark Office	Docket No.	Serial No.
(PTO Form 1449 modified)	SRI1P037B	09/608,872
INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Applicant Halversen, et al. #23	Confirmation No.: 2382
(Use several sheets if necessary)	Filing Date	Group
Examiner Firmin Backer	June 30, 2000	2155

U.S. Patent Documents

*Examiner Initial		Document Number	Issue Date	Applicant(s) Name	Class	Subclass	Filing Date If Appropriate
FB	A1	6,016,476	01/18/2000	Maes, et al.	705	1	
	A2						
	A3						
	A4						
	A5						
	A6						
	A7						
	A8						
	A9						
	A10						
	A11						
	A12						
	A13						

Foreign Patent Documents

*Examiner Initial		Document Number	Date	Country	Class	Subclass	Translation	
							YES	NO
FB	B1	0 867 861	09/30/1998	EPO	G10L	5/06	□	□
FB	B2	99/50826	10/07/1999	WIPO	G10L	3/00	□	□
FB	B3	00/05638	02/03/2000	WIPO	G06F	—	□	□
	B4						□	□
	B5						□	□

OTHER ART

*Examiner Initial		Including Author, Title, Date, Pertinent Pages, Etc.
FB	C1	International Search Report, Intl Appl No. PCT/US01/07987
	C2	
	C3	

Examiner	Date Considered 9/27/02

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609; Draw line through citation if not in conformance and not considered. Include copy of this form with your communication to applicant.

PATENT COOPERATION TREATY

From the INTERNATIONAL SEARCHING AUTHORITY

To:

CARLTON, FIELDS, WARD, EMMANUEL,
SMITH & CUTLER, P.A.
Attn. TONG,Kin-Wah
P.O. Box 3239
TAMPA, FL 33601-3239
UNITED STATES OF AMERICA

PCT

NOTIFICATION OF TRANSMITTAL OF THE INTERNATIONAL SEARCH REPORT OR THE DECLARATION

(PCT Rule 44.1)

	Date of mailing *(day/month/year)* 03/07/2002
Applicant's or agent's file reference SRI1P037B.P	**FOR FURTHER ACTION** See paragraphs 1 and 4 below
International application No. PCT/US 01/07987	International filing date *(day/month/year)* 12/03/2001
Applicant SRI INTERNATIONAL et al.	

1. ☒ The applicant is hereby notified that the International Search Report has been established and is transmitted herewith.

   Filing of amendments and statement under Article 19:
   The applicant is entitled, if he so wishes, to amend the claims of the International Application (see Rule 46):

   When? The time limit for filing such amendments is normally 2 months from the date of transmittal of the International Search Report, however, for more details, see the notes on the accompanying sheet.

   Where? Directly to the International Bureau of WIPO
   34, chemin des Colombettes
   1211 Geneva 20, Switzerland
   Fascimile No.: (41-22) 740.14.35

   For more detailed instructions, see the notes on the accompanying sheet.

2. □ The applicant is hereby notified that no International Search Report will be established and that the declaration under Article 17(2)(a) to that effect is transmitted herewith.

3. □ **With regard to the protest** against payment of (an) additional fee(s) under Rule 40.2, the applicant is notified that:

   □ the protest together with the decision thereon has been transmitted to the International Bureau together with the applicant's request to forward the texts of both the protest and the decision thereon to the designated Offices.

   □ no decision has been made yet on the protest; the applicant will be notified as soon as a decision is made.

4. **Further action(s):** The applicant is reminded of the following:

   Shortly after **18 months** from the priority date, the international application will be published by the International Bureau. If the applicant wishes to avoid or postpone publication, a notice of withdrawal of the international application, or of the priority claim, must reach the International Bureau as provided in Rules 90*bis*.1 and 90*bis*.3, respectively, before the completion of the technical preparations for international publication.

   Within **19 months** from the priority date, a demand for international preliminary examination must be filed if the applicant wishes to postpone the entry into the national phase until 30 months from the priority date (in some Offices even later).

   Within **20 months** from the priority date, the applicant must perform the prescribed acts for entry into the national phase before all designated Offices which have not been elected in the demand or in a later election within 19 months from the priority date or could not be elected because they are not bound by Chapter II.

Name and mailing address of the International Searching Authority	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL-2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Claude Berthon

NOTES TO FORM PCT/ISA/220

These Notes are intended to give the basic instructions concerning the filing of amendments under article 19. The Notes are based on the requirements of the Patent Cooperation Treaty, the Regulations and the Administrative Instructions under that Treaty. In case of discrepancy between these Notes and those requirements, the latter are applicable. For more detailed information, see also the PCT Applicant's Guide, a publication of WIPO.

In these Notes, "Article", "Rule", and "Section" refer to theprovisions of the PCT, the PCT Regulations and the PCT Administrative Instructions respectively.

INSTRUCTIONS CONCERNING AMENDMENTS UNDER ARTICLE 19

The applicant has, after having received the international search report, one opportunity to amend the claims of the international application. It should however be emphasized that, since all parts of the international application (claims, description and drawings) may be amended during the international preliminary examination procedure, there is usually no need to file amendments of the claims under Article 19 except where, e.g. the applicant wants the latter to be published for the purposes of provisional protection or has another reason for amending the claims before international pbulication. Furthermore, it should be emphasized that provisional protection is available in some States only.

What parts of the international application may be amended?

Under Article 19, only the claims may be amended.

During the international phase, the claims may also be amended (or further amended) under Article 34 before the International Preliminary Examining Authority. The description and drawings may only be amended under Article 34 before the International Examining Authority.

Upon entry into the national phase, all parts of the international application may be amended under Article 28 or, where applicable, Article 41.

When? Within 2 months from the date of transmittal of the international search report or 16 months from the priority date, whichever time limit expires later. It should be noted, however, that the amendments will be considered as having been received on time if they are received by the International Bureau after the expiration of the applicable time limit but before the completion of the technical preparations for international publication (Rule 46.1).

Where not to file the amendments?

The amendments may only be filed with the International Bureau and not with the receiving Office or the International Searching Authority (Rule 46.2).

Where a demand for international preliminary examination has been/is filed, see below.

How? Either by cancelling one or more entire claims, by adding one or more new claims or by amending the text of one or more of the claims as filed.

A replacement sheet must be submitted for each sheet of the claims which, on account of an amendment or amendments, differs from the sheet originally filed.

All the claims appearing on a replacement sheet must be numbered in Arabic numerals. Where a claim is cancelled, no renumbering of the other claims is required. In all cases where claims are renumbered, they must be renumbered consecutively (Administrative Instructions, Section 205(b)).

The amendments must be made in the language in which the international application is to be published.

What documents must/may accompany the amendments?

Letter (Section 205(b)):

The amendments must be submitted with a letter.

The letter will not be published with the international application and the amended claims. It should not be confused with the "Statement under Article 19(1)" (see below, under "Statement under Article 19(1)").

The letter must be in English or French, at the choice of the applicant. However, if the language of the international application is English, the letter must be in English; if the language of the international application is French, the letter must be in French.

NOTES TO FORM PCT/ISA/220 (continued)

The letter must indicate the differences between the claims as filed and the claims as amended. It must, in particular, indicate, *in connection with each claim appearing in* the international application (it being understood that identical indications concerning several claims may be grouped),whether

(i) the claim is unchanged;

(ii) the claim is cancelled;

(iii) the claim is new;

(iv) the claim replaces one or more claims as filed;

(v) the claim is the result of the division of a claim as filed.

The following examples illustrate the manner in which *amendments must be explained in the* accompanying letter:

1. [Where originally there were 48 claims and *after amendment of some claims there are 51*]:
"Claims 1 to 29, 31, 32, 34, 35, 37 to 48 replaced by amended claims bearing the same numbers; claims 30, 33 and 36 unchanged; new claims 49 to 51 added."

2. [Where originally there were 15 claims and after amendment of all claims there are 11]:
"Claims 1 to 15 replaced by amended claims 1 to 11."

3. [Where originally there were 14 claims and the amendments *consist in cancelling some claims and in* adding new claims]:
"Claims 1 to 6 and 14 unchanged; claims 7 to 13 cancelled; new claims 15, 16 and 17 added." or
"Claims 7 to 13 cancelled; new claims 15, 16 and 17 added; all other claims unchanged."

4. [Where various kinds of amendments are made]:
"Claims 1-10 unchanged; claims 11 to 13, 18 and 19 cancelled; claims 14, 15 and 16 replaced by amended claim 14; claim 17 subdivided into amended claims 15, 16 *and 17; new claims 20 and 21 added.*"

"Statement under article 19(1)" *(Rule 46.4)*

The amendments may be accompanied by a statement explaining the amendments and indicating any impact that such amendments might have on the description and the drawings (which cannot be amended under *Article 19(1)*).

The statement will be published with the international application and the amended claims.

It must be in the language in which the international appplication is to be published.

It must be brief, not exceeding 500 words if in English or if translated into English.

It should not be confused with and does not replace the letter indicating the differences between the claims as filed and as amended. It must be filed on a separate sheet and must be identified as such by a heading, preferably by using the words "Statement under Article 19(1)."

It may not contain any disparaging comments on the international search report or the relevance of citations contained in that report. Reference to citations, relevant to a given claim, contained in the international search report may be made only in connection with an amendment of that claim.

***Consequence if a demand for international* preliminary examination has already been filed**

If, at the time of filing any amendments under Article 19, a demand for international preliminary examination has already been submitted, the applicant must *preferably*, at the *same time of filing the amendments with the* International Bureau, also file a copy of such amendments with the International Preliminary Examining Authority (see Rule 62.2(a), first sentence).

Consequence with regard to translation of the international application for entry into the national phase

The applicant's attention is drawn to the fact that, where upon entry into the national phase, a translation of the claims as amended under Article 19 may have to be furnished to the designated/elected Offices, instead of, or in addition to, the translation of the claims as filed.

For further details on the requirements of each designated/elected Office, see Volume II of the PCT Applicant's Guide.

PATENT COOPERATION TREATY

PCT

INTERNATIONAL SEARCH REPORT

(PCT Article 18 and Rules 43 and 44)

Applicant's or agent's file reference SRI1P037B.P	**FOR FURTHER ACTION** see Notification of Transmittal of International Search Report (Form PCT/ISA/220) as well as, where applicable, item 5 below.	
International application No. PCT/US 01/ 07987	International filing date *(day/month/year)* 12/03/2001	(Earliest) Priority Date *(day/month/year)* 13/03/2000
Applicant SRI INTERNATIONAL et al.		

This International Search Report has been prepared by this International Searching Authority and is transmitted to the applicant according to Article 18. A copy is being transmitted to the International Bureau.

This International Search Report consists of a total of 3 sheets.

☒ It is also accompanied by a copy of each prior art document cited in this report.

1. **Basis of the report**

a. With regard to the **language**, the international search was carried out on the basis of the international application in the language in which it was filed, unless otherwise indicated under this item.

□ the international search was carried out on the basis of a translation of the international application furnished to this Authority (Rule 23.1(b)).

b. With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, the international search was carried out on the basis of the sequence listing :

□ contained in the international application in written form.

□ filed together with the international application in computer readable form.

□ furnished subsequently to this Authority in written form.

□ furnished subsequently to this Authority in computer readble form.

□ the statement that the subsequently furnished written sequence listing does not go beyond the disclosure in the international application as filed has been furnished.

□ the statement that the information recorded in computer readable form is identical to the written sequence listing has been furnished

2. □ **Certain claims were found unsearchable** (See Box I).

3. □ **Unity of invention is lacking** (see Box II).

4. With regard to the **title**,

☒ the text is approved as submitted by the applicant.

□ the text has been established by this Authority to read as follows:

5. With regard to the **abstract**,

☒ the text is approved as submitted by the applicant.

□ the text has been established, according to Rule 38.2(b), by this Authority as it appears in Box III. The applicant may, within one month from the date of mailing of this international search report, submit comments to this Authority.

6. The figure of the **drawings** to be published with the abstract is Figure No. 1A

☒ as suggested by the applicant.

□ because the applicant failed to suggest a figure.

□ because this figure better characterizes the invention.

□ None of the figures.

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 01/07987

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04M3/493 G10L15/22 G06F17/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04M G10L G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, PAJ

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 00 05638 A (MOTOROLA INC) 3 February 2000 (2000-02-03) page 4, line 30 -page 5, line 11 page 6, line 13 - line 32 page 22, line 28 -page 23, line 15 figures 3,5A ----	1-27
A	EP 0 867 861 A (OCTEL COMMUNICATIONS CORP) 30 September 1998 (1998-09-30) column 2, line 33 -column 3, line 48 ---	1-27
A	WO 99 50826 A (ANDREA ELECTRONICS CORP ;ANDREA DOUGLAS (US); MARIANO JOSEPH (US)) 7 October 1999 (1999-10-07) page 3, line 13 - line 17 figure 1A --- -/--	1-27

☒ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search 26 June 2002	Date of mailing of the international search report 03/07/2002
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Schweitz, M

1

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 01/07987

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	US 6 016 476 A (SEDIVY JAN ET AL) 18 January 2000 (2000-01-18) column 3, line 17 - line 37 -----	1-27

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US 01/07987

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 0005638 A	03-02-2000	US 2002006126 A1	17-01-2002
		AU 5006799 A	14-02-2000
		AU 5126999 A	14-02-2000
		AU 5127099 A	14-02-2000
		AU 5227899 A	14-02-2000
		CN 1354851 T	19-06-2002
		EP 1099152 A1	16-05-2001
		EP 1101343 A1	23-05-2001
		EP 1099146 A2	16-05-2001
		EP 1099213 A1	16-05-2001
		WO 0005861 A1	03-02-2000
		WO 0005708 A1	03-02-2000
		WO 0005643 A1	03-02-2000
		WO 0005638 A2	03-02-2000
		US 6269336 B1	31-07-2001
EP 0867861 A	30-09-1998	US 6094476 A	25-07-2000
		CA 2233019 A1	24-09-1998
		EP 0867861 A2	30-09-1998
		JP 11088502 A	30-03-1999
		US 6385304 B1	07-05-2002
		US 6377662 B1	23-04-2002
WO 9950826 A	07-10-1999	AU 3212899 A	18-10-1999
		CA 2323874 A1	07-10-1999
		EP 1066624 A1	10-01-2001
		JP 2002510074 T	02-04-2002
		WO 9950826 A1	07-10-1999
US 6016476 A	18-01-2000	EP 1004099 A1	31-05-2000
		WO 9908238 A1	18-02-1999
		HU 0004470 A2	28-05-2001
		JP 2001512876 T	28-08-2001
		PL 338353 A1	23-10-2000
		TW 385400 B	21-03-2000

5l

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/608,872	06/30/2000	Christine Halversen	SRIlp037B	2382

7590 10/04/2002

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY, NJ 07702

EXAMINER	
JEAN, FRANTZ B	
ART UNIT	PAPER NUMBER
2155	24

DATE MAILED: 10/04/2002

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 07-01)

Office Action Summary	**Application No.** 09/608,872	**Applicant(s)** HALVERSEN ET AL.
	Examiner Frantz B. Jean	**Art Unit** 2155

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 7/29/2002.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 56-82 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 56-82 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)☐ The proposed drawing correction filed on _____ is: a)☐ approved b)☐ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____ .

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) ☐ The translation of the foreign language provisional application has been received.

15)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application (PTO-152)
3) ☒ Information Disclosure Statement(s) (PTO-1449) Paper No(s) 7,23 .	6) ☐ Other:

DETAILED ACTION

1. This office action is in response to an amendment received on 7/18/02. Claims 56, 65 and 74 were amended. Claims 56-82 are still pending in this application.

Information Disclosure Statement

2. The IDS received on 7/29/02 have been considered.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 56-82 are rejected under 35 U.S.C. 103(a) as being unpatentable over Levin et al. (U.S. Patent No. 6,173,279) in view of Bailey, III US patent No. 6,353,66.

5. As per claim 56, Levin et al teach a method for speech-based navigation (information server, 110) of an electronic data source located at one or more network servers located remotely from a user, wherein at least a portion of a data link between a mobile information appliance of the user and the one or more network servers utilizes wireless communication (see abstract, fig 1, column 3 lines 5-35), comprising receiving a request (receive a natural language query) for desired information from the user (user, 112) utilizing the mobile appliance (PC, 102) of the user

wherein said mobile information comprises a portable remote control device or top-box for a television; rendering an interpretation (creating a semantic representation) of the request, constructing a navigation (generating search) query based upon the interpretation; utilizing the navigation query to select a portion of the electronic data source; and transmitting (sending) the selected portion of the electronic data source from the network server to the mobile information appliance of the user. (see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22). Although Levin teaches natural language, Levin does not explicitly elaborate on a spoken request for desired information from a user. Bailey III is directed to a network and communication access system which includes a spoken (audible) request for desired information from a user (col. 9 lines 47 et seq; col. 3 lines 21 et seq). It would have been obvious to one of ordinary skill in the art at the time of the invention to have combined Bailey's, III features to Levin's because they would have speeded up the communication process while providing a secure system (see Bailey, III col. 4 lines 41 et seq).

6. As per claims 57, 58, 62-64, Levin et al teach a method of rendering the interpretation of the request is performed at the one or more network servers by the mobile information appliance including a Wireless telephone, a portable computer that is a personal digital assistance (See abstract, fig 1, column 3 lines 5-35).

7. As per claim 59, Levin et al teach a method of soliciting additional input from the user, including user interaction in a modality different than the original request; refining the navigation query, based upon the additional input; and using the refined navigation query to select a portion of the electronic data source (see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22).

8. As per claim 60, Levin et al teach a method wherein the data link includes a cellular telephone system (see fig 1, column 2 line 61-67).

9. As per claim 61, Levin et al teach a method wherein steps (a)-(d) are performed with respect to multiple users (see abstract, fig 1, column 3 lines 5-35).

10. As per claim 65, Levin et al teach a computer system for speech-based navigation (information server, 110) of an electronic data source located at one or more network servers located remotely from a user, wherein at least a portion of a data link between a mobile information appliance of the user and the one or more network servers utilizes wireless communication (see abstract, fig 1, column 3 lines 5-35), comprising a code segment receiving a request (receive a natural language query) for desired information from the user (user) utilizing the mobile information appliance (PC, 102) of the user- a code segment rendering an interpretation (creating a semantic representation) of the request, a code segment constructing a navigation (generating search) query based upon the interpretation; a code segment utilizing the

navigation query to select a portion of the electronic data source; and a code segment transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user. (see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22). Although Levin teaches natural language, Levin does not explicitly elaborate on a spoken request for desired information from a user. Bailey III is directed to a network and communication access system which includes a spoken (audible) request for desired information from a user (col. 9 lines 47 et seq; col. 3 lines 21 et seq). It would have been obvious to one of ordinary skill in the art at the time of the invention to have combined Bailey's, III features to Levin's because they would have speeded up the communication process while providing a secure system (see Bailey, III col. 4 lines 41 et seq).

11. As per claims 66, 67, 71-73, Levin et al teach a system of rendering the interpretation of the request is performed at the one or more network servers by the mobile information appliance including a wireless telephone, a portable computer that is a personal digital assistance (see abstract, fig 1, column 3 lines 5-35).

12. As per claim 68, Levin et at teach a system of soliciting additional input from the user, including user interaction in a modality different than the original request; refining the navigation query, based upon the additional input; and using the refined navigation query to select a portion

of the electronic data source (see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22).

13. As per claim 69, Levin et al teach a system wherein the data link includes a cellular telephone system (see fig 1, column 2 line 61-67).

14. As per claim 70, Levin et a] teach a system wherein steps (a)-(d) are performed with respect to multiple users (see abstract, fig 1, column 3 lines 5-35).

15. As per claim 74, Levin et at teach a system for speech-based navigation (information server, 110) of an electronic data source located at one or more network servers located remotely from a user, wherein at least a portion of a data link between a mobile information appliance of the user and the one or more network servers utilizes wireless communication (see abstract, fig 1, column 3 lines 5-35), comprising receiving a request (receive a natural language query) for desired information from the user (user) utilizing the mobile information appliance (PC, 102) of the user; rendering an interpretation (creating a semantic representation) of the request, constructing a navigation (generating search) query based upon the interpretation; utilizing the navigation query to select a portion of the electronic data source; and transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user. (see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22). Although

Levin teaches natural language, Levin does not explicitly elaborate on a spoken request for desired information from a user. Bailey III is directed to a network and communication access system which includes a spoken (audible) request for desired information from a user (col. 9 lines 47 et seq; col. 3 lines 21 et seq). It would have been obvious to one of ordinary skill in the art at the time of the invention to have combined Bailey's, III features to Levin's because they would have speeded up the communication process while providing a secure system (see Bailey, III col. 4 lines 41 et seq).

16. As per claims 75, 76, 80-8 1, Levin et al teach a method of rendering the interpretation of a request that is performed at the one or more network servers by the mobile information appliance including a wireless telephone, a portable computer that is a personal digital assistance (see abstract, fig 1, column 3 lines 5-35).

17. As per claim 77, Levin et al teach a system of soliciting additional input from the user, including user interaction in a modality different than the original request; refining the navigation query, based upon the additional input; and using the refined navigation query to select a portion of the electronic data source (see abstract, fig. 1-3, column 3 line 36-9 line 5, see also claim 1, 10, 22).

18. As per claim 78, Levin et al teach a system wherein the data link includes a cellular telephone system (see fig 1, column 2 line 61-67).

19. As per claim 79, Levin et al teach a system wherein steps (a)-(d) are performed with respect to multiple users (see abstract, fig 1, column 3 lines 5-35).

Response to Arguments

20. Applicant's arguments filed on....7/18/02......have been fully considered but they are not persuasive. a. Applicant argues that the prior art "falls to teach or suggest the novel concept of speech-based navigation where the method receives spoken request for desired information from the user utilizing the mobile information appliance of the user and where in turn the selected electronic data source from the network server is transmitted to the mobile information appliance of the user." Examiner respectfully disagrees with the applicant perspective and characterization of Levin inventive concept. Levin teach that the URL for a data resource is inputted into PC 102 either by typing the request using a keyboard 104 or by speaking the request into a microphone 105, which is considered to be a mobile appliance of the user. Furthermore, Levin et al indicate that the spoken requests either from a PC microphone 105 or from a telephone 103 can be handled by a speech recognition system residing at the information server (see column 4 lines 7-22). Applicant further argues that the prior art "falls to teach or suggest that the selected electronic data source from the network server is transmitted to the mobile information appliance of the user." Examiner respectfully disagrees with the applicant perspective and characterization of Levin inventive concept. Levin teach that once an information server is accessed, the user can

send a text or a spoken query requesting a particular action or service (step 204), for example: "call the pizza place on Main Street in Westfield". The query is received by the access server 106 and the natural language query is sent to the information server I 10 via packet network 108. It is to be understood that the packet network 108 may be connected to a plurality of information servers which each relate to one or more particular information services, or there may be a single centralized information server 110 which is accessed by all information services which are capable of receiving and processing natural language queries and contains at least some of the data resources (e.g., URLs and associated site/service-specific grammars) capable of receiving and responding to a natural language query. It is obvious inventive concept referring to response is in the field of sending or transmitting the requested information to the user. Moreover, it is understood in the art of information request, in order to complete the transaction, the host must transmit to the requester the requested information.

21. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

22. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Frantz B. Jean whose telephone number is (703) 305-3970. The examiner can normally be reached on Monday thru Friday from 8:30 to 6:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Ayaz R. Sheikh, can be reached on (703) 305-9648. The fax phone numbers for this Group are

(703) 746-7238 for After-Final, (703) 746-7239 for Official, and (703) 746-7240 for Non-Official/Draft.

Communications via Internet e-mail regarding this application, other than those under 35 U.S.C. 132 or which otherwise require a signature, may be used by the applicant and should be addressed to [**Ayaz.Sheikh@uspto.gov**].

All Internet e-mail communications will be made of record in the application file. PTO employees do not engage in Internet communications where there exists a possibility that sensitive information could be identified or exchanged unless the record includes a properly signed express waiver of the confidentiality requirements of 35 U.S.C. 122. This is more clearly set forth in the Interim Internet Usage Policy published in the Official Gazette of the Patent and Trademark on February 25, 1997 at 1195 OG 89.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the Group receptionist whose telephone number is (703) 305-3900.

Frantz B. Jean
September 29, 2002
FBJ/

Notice of References Cited	Application/Control No. 09/608,872	Applicant(s)/Patent Under Reexamination HALVERSEN ET AL.
	Examiner Frantz B. Jean	Art Unit 2155 / Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-6,317,684 B1	11-2001	Roeseler et al.	340/990
*	B	US-6,349,257 B1	02-2002	Liu et al.	340/5.6
*	C	US-6,314,365 B1	11-2001	Smith, Nicholas E.	340/988
	D	US-6,353,661 B1	03-2002	Bailey, III, John Edson	379/88.17
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

#25

09/608,872

IN THE UNITED STATES
PATENT AND TRADEMARK OFFICE

PATENT APPLICATION

Applicant: **Halverson et al.**

Case: **SRI1P037B**

Serial No.: **09/608,872** Filed: **June 30, 2000**

Group Art Unit: **2155**

Examiner: **Frantz Jean**

Title: **MOBILE NAVIGATION OF NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN INPUT**

ASSISTANT COMMISSIONER FOR PATENTS
Box Non-Fee Amendment
Washington, D. C. 20231

S I R:

RESPONSE UNDER 37 C.F.R. § 1.111

This response addresses the Office Action dated October 4, 2002 (Paper No. 24).

REMARKS

Applicants' representative would like to thank Primary Examiner Frantz Jean for kindly taking a substantial amount of time on December 23, 2002 to discuss the merits of the subject invention in a face-to-face Examiner Interview. Applicants' representative is aware of the time constraint that is placed on the Examiner and is appreciative of the Examiner's willingness to devote such large quantity of time to discuss the case on the merit.

In view of the following discussion, the Applicants submit that none of the claims now pending in the application are made obvious under the provisions of 35 U.S.C. § 103. Thus, the Applicants believe that all of these claims are now in allowable form.

I. REJECTION OF CLAIMS 56-82 UNDER 35 U.S.C. § 103

The Examiner rejected claims 56-82 in Paragraphs 4-19 of the Office Action as being unpatentable over Levin et al. patent (US Patent 6,173,279 issued January 9, 2001, hereinafter referred to as Levin) in view of Bailey III (US Patent 6,353,661 issued March 5, 2002, hereinafter referred to as Bailey). The rejection is respectfully traversed.

Levin teaches "a method of using at least one natural language query to retrieve information from one or more data resources and further performing a requested action using the retrieved information is disclosed". (See Levin, Column 2, lines 15-18) Namely, Levin teaches a method for using natural language query to obtain information, where upon receipt of the requested information, a desired action is executed based upon the requested information. To illustrate, Levin provides the example, where a user employs natural language to request the telephone number of a restaurant. Upon receipt of the telephone number, the telephone number is actually dialed for the user. (See Levin, Column 3 line 62 to Column 4, line 1)

Bailey teaches a system for using a telephone to interact with a remote system. Specifically, Bailey teaches the use of a conventional phone to allow users to browse, search, store, and create information stored on the Internet. (See Bailey, Abstract; Column 3, lines 8-39)

In contrast, the alleged combination of Levin and Bailey (either singly or in any permissible combination) fails to teach or suggest the novel concept of speech-based navigation where the method receives spoken request for desired information from the user utilizing the mobile information appliance of the user, wherein said mobile information appliance comprises a portable remote control device or a set-top box for a television. Specifically, Applicants' independent claims 56, 65 and 74 positively recite:

56. A method for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, wherein a data link is established between a mobile information appliance of the user and the one or more network servers, comprising the steps of:

(a) receiving a spoken request for desired information from the user utilizing the mobile information appliance of the user, wherein said mobile information appliance comprises a portable remote control device or a set-top box for a television;

(b) rendering an interpretation of the spoken request;

(c) constructing a navigation query based upon the interpretation;

(d)utilizing the navigation query to select a portion of the electronic data source; and

(e) transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user. (emphasis added)

65. A computer program embodied on a computer readable medium for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, wherein a data link is established between a mobile information appliance of the user and the one or more network servers, comprising:

(a) a code segment that receives a spoken request for desired information from the user utilizing the mobile information appliance of the user, wherein said mobile information appliance comprises a portable remote control device or a set-top box for a television;

(b) a code segment that renders an interpretation of the spoken request;

(c) a code segment that constructs a navigation query based upon the interpretation;

(d) a code segment that utilizes the navigation query to select a portion of the electronic data source; and

(e) a code segment that transmits the selected portion of the electronic data source from the network server to the mobile information appliance of the user. (emphasis added)

74. A system for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, comprising:

(a) a mobile information appliance operable to receive a spoken request for desired information from the user, wherein said mobile information appliance comprises a portable remote control device or a set-top box for a television;

(b) spoken language processing logic, operable to render an interpretation of the spoken request;

(c) query construction logic, operable to construct a navigation query based upon the interpretation;

(d) navigation logic, operable to select a portion of the electronic data source using the navigation query, and

(e) electronic communications infrastructure for transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user. (emphasis added)

Applicants' invention teaches a novel method and apparatus for speech-based navigation where the method receives spoken request for desired information from the user utilizing the mobile information appliance of the user, wherein said mobile information appliance comprises a portable remote control device or a set-top box for a television. This teaching is completely absent in the Levin and Bailey references.

During the Examiner Interview, Applicants' representative indicated to the Examiner that the present claims specifically recite said mobile information appliance comprises a portable remote control device or a set-top box for a television. Applicants' specification (e.g., on page 2) describes a need for a user interface that does not require the user to learn a highly specialized command language or format. In describing Applicants' invention in the context of a home entertainment setting, Applicants disclose the present invention within the context of a portable remote control device or a set-top box for a television. (e.g., See Applicants' specification, page 6, lines 4-20; and page 18, line 4 to page 19, line 9). In sum, Applicants' novel speech-based navigation method is claimed specifically within the context of a portable remote control device or a set-top box for a television.

During the Examiner Interview, Applicants' representative presented to the Examiner that the combination of Levin and Bailey will fall short of making Applicants' invention obvious. Namely, both references do not disclose Applicants' novel speech-based navigation method within the context of a portable remote control device or a set-top box for a television. For example, Bailey states that "the present invention generally relates to a method and system for combining the power, flexibility, and access to information and communications of the Internet with the simplicity, reliability and wide availability of the existing plain old telephone system (POTS)." (See Bailey, Column 1, lines 5-9) Specifically, the entire purpose of Bailey is to salvage the use of a plain old telephone system to access the Internet. Thus, Bailey does not disclose or suggest Applicants' novel speech-based navigation method within the context of a portable remote control device or a set-top box for a television.

Second, the alleged combination (as taught by Bailey) states that "once the information is obtained the system presents the information to the user by transforming the downloaded text into speech in a manner emulating the behavior of a web browser." (Emphasis added) (See Bailey, Column 3, lines 21-25) Bailey then discloses a complicated method of notifying content, e.g., hyperlinks, of a web page to a user via audible signals. (See Bailey, Column 7, line 5 to Column 8, line 10). In sum, Bailey converts a telephone into a user interface that serves as a web browser as positively asserted by Bailey. This teaching is directly contrary to Applicants' invention which recites "receiving a spoken request for desired information from the user utilizing the mobile information appliance of the user, wherein said mobile information appliance comprises a portable remote control device or a set-top box for a television" and interpreting the spoken request. Applicants' invention is intended to address the criticality of not having to navigate the electronic data source, whereas Bailey simply converts the web page content so that the user is required to manually navigate the data source by listening to different audible signals. Thus, Bailey **teaches away** from Applicants' novel speech-based navigation method.

During the Examiner Interview, the Examiner indicated that he will re-evaluate the cited references and reconsider the present rejections. Therefore, the Applicants respectfully submit that independent claims 56, 65 and 74 are not made obvious by the Levin and Bailey references. As such, claims 56, 65 and 74 fully satisfy the requirements of 35 U.S.C. §103 and are patentable thereunder.

Claims 57-64, 66-73 and 75-82 depend, either directly or indirectly, from claims 56, 65 and 74 and recite additional features therefor. Since Levin and Bailey fail to make Applicants' invention obvious as recited in Applicants' independent claims 56, 65 and 74, dependent claims 57-64, 66-73 and 75-82 are also not made obvious under 35 U.S.C. § 103 and are allowable for the same reason noted above.

Conclusion

Thus, the Applicants submit that all of these claims now fully satisfy the requirements of 35 U.S.C. §103. Consequently, the Applicants believe that all these claims are presently in condition for allowance. Accordingly, both reconsideration of this application and its swift passage to issue are earnestly solicited.

If, however, the Examiner believes that there are any unresolved issues requiring the issuance of a final action in any of the claims now pending in the application, it is requested that the Examiner telephone Mr. Kin-Wah Tong, Esq. at (732) 530-9404 so that appropriate arrangements can be made for resolving such issues as expeditiously as possible.

Respectfully submitted,

1/6/03

Kin-Wah Tong, Attorney
Reg. No. 39,400
(732) 530-9404

Moser, Patterson & Sheridan, LLP
595 Shrewsbury Avenue
First Floor,
Shrewsbury, New Jersey 07702

TELEFAX COVER SHEET

MOSER, PATTERSON & SHERIDAN, LLP
ATTORNEYS AT LAW
595 SHREWSBURY AVENUE
FIRST FLOOR
SHREWSBURY, NJ 07702
TELEPHONE (732) 530-9404
TELEFAX (732) 530-9808

****

THIS TELEFAX MESSAGE IS ADDRESSED TO THE PERSON OR COMPANY LISTED BELOW. IF IT WAS SENT OR RECEIVED INCORRECTLY, OR YOU ARE NOT THE INTENDED RECIPIENT, PLEASE TAKE NOTICE THAT THIS MESSAGE MAY CONTAIN PRIVILEGED OR CONFIDENTIAL MATERIAL, AND YOUR DUE REGARD FOR THIS INFORMATION IS NECESSARY. YOU MAY ARRANGE TO RETURN THIS MATERIAL BY CALLING THE FIRM LISTED ABOVE AT (732) 530-9404

****

THIS MESSAGE HAS 9 PAGES INCLUDING THIS SHEET

TO: Commissioner of Patents

FAX NO.: 703-746-7239

FROM: Kin-Wah Tong

DATE: January 6, 2003

MATTER: Serial No. 09/608,872 Filed: June 30, 2000

DOCKET NO.: SRI 1P037B

APPLICANT: HALVERSON, et al

The following has been received in the U.S. Patent and Trademark Office on the date of this facsimile:

__ Petition
__ Disclosure Statement & PTO-1449
__ Priority Document
__ Drawings (____ sheets) informal
__ Petition for Extension of Time (2 copies)
X Response

X Transmittal Letter (2 copies)
__ Fee Transmittal (2 copies)
__ Deposit Account Transaction
X Facsimile Transmission Certificate dated January 6, 2003

CERTIFICATE OF TRANSMISSION UNDER 37 C.F.R. §1.8

I hereby certify that this correspondence is being transmitted by facsimile to the Commissioner for Patents, Box Non-Fee Amendment, Washington, DC 20231 on **January 6, 2003**. Facsimile No. 703-746-7239.

Kin-Wah Tong
Name of person signing this certificate

January 6, 2003
Signature and date

PTO/SB/21 (08-00)

Please type a plus sign (+) inside this box ⟶ [+]

Approved for use through 10/31/2002. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	09/608,872
Filing Date	June 30, 2000
First Named Inventor	HALVERSON
Group Art Unit	2155
Examiner Name	FRANTZ JEAN
Total Number of Pages in This Submission	
Attorney Docket Number	SRI 1 P 037B

ENCLOSURES *(check all that apply)*

- □ Fee Transmittal Form
  - □ Fee Attached
- ☒ Amendment / Response
  - □ After Final
  - □ Affidavits/declaration(s)
- □ Extension of Time Request
- □ Express Abandonment Request
- □ Information Disclosure Statement
- □ Certified Copy of Priority Document(s)
- □ Response to Missing Parts/ Incomplete Application
  - □ Response to Missing Parts under 37 CFR 1.52 or 1.53

- □ Assignment Papers *(for an Application)*
- □ Drawing(s)
- □ Licensing-related Papers
- □ Petition
- □ Petition to Convert to a Provisional Application
- □ Power of Attorney, Revocation Change of Correspondence Address
- □ Terminal Disclaimer
- □ Request for Refund
- □ CD, Number of CD(s) _____

- □ After Allowance Communication to Group
- □ Appeal Communication to Board of Appeals and Interferences
- □ Appeal Communication to Group *(Appeal Notice, Brief, Reply Brief)*
- □ Proprietary Information
- □ Status Letter
- ☒ Other Enclosure(s) *(please identify below)*:

  Certificate of Facsimile Transmission

Remarks

It is believed no fee is due. However, in the event a fee is due, kindly charge that fee to deposit account number 20-0782. To facilitate that charge, a duplicate copy of this letter is enclosed

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm *or* individual name	KIN-WAH TONG, ESQ., Reg. No. 39,400
Signature	K. W. Tong
Date	January 6, 2003

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be send to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

PTO/SB/21 (08-00)

Please type a plus sign (+) inside this box ⟶ ⊞

Approved for use through 10/31/2002. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	09/608,872
Filing Date	June 30, 2000
First Named Inventor	HALVERSON
Group Art Unit	2155
Examiner Name	FRANTZ JEAN
Attorney Docket Number	SRI 1 P 037B

Total Number of Pages in This Submission ______

ENCLOSURES *(check all that apply)*

- □ Fee Transmittal Form
  - □ Fee Attached
- ☒ Amendment / Response
  - □ After Final
  - □ Affidavits/declaration(s)
- □ Extension of Time Request
- □ Express Abandonment Request
- □ Information Disclosure Statement
- □ Certified Copy of Priority Document(s)
- □ Response to Missing Parts/ Incomplete Application
  - □ Response to Missing Parts under 37 CFR 1.52 or 1.53

- □ Assignment Papers *(for an Application)*
- □ Drawing(s)
- □ Licensing-related Papers
- □ Petition
- □ Petition to Convert to a Provisional Application
- □ Power of Attorney, Revocation Change of Correspondence Address
- □ Terminal Disclaimer
- □ Request for Refund
- □ CD, Number of CD(s) ____

- □ After Allowance Communication to Group
- □ Appeal Communication to Board of Appeals and Interferences
- □ Appeal Communication to Group *(Appeal Notice, Brief, Reply Brief)*
- □ Proprietary Information
- □ Status Letter
- ☒ Other Enclosure(s) *(please identify below)*:

  Certificate of Facsimile Transmission

Remarks: **It is believed no fee is due. However, in the event a fee is due, kindly charge that fee to deposit account number 20-0782. To facilitate that charge, a duplicate copy of this letter is enclosed**

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm *or* Individual name	KIN-WAH TONG, ESQ., Reg. No. 39,400
Signature	
Date	January 6, 2003

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be send to the Chief information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/608,872	06/30/2000	Christine Halversen	SRIIp037B	2382

7590 01/09/2003

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY, NJ 07702

EXAMINER	
JEAN, FRANTZ B	
ART UNIT	PAPER NUMBER 26
2155	

DATE MAILED: 01/09/2003

Please find below and/or attached an Office communication concerning this application or proceeding.

Interview Summary	**Application No.** 09/608,872	**Applicant(s)** HALVERSEN ET AL.	
	Examiner Frantz B. Jean	**Art Unit** 2155	

All participants (applicant, applicant's representative, PTO personnel):

(1) *Frantz B. Jean*. (3)____.

(2) *Kin-Wah Tong*. (4)____.

Date of Interview: *23 December 2002* .

Type: a)□ Telephonic b)□ Video Conference
c)☒ Personal [copy given to: 1)□ applicant 2)☒ applicant's representative]

Exhibit shown or demonstration conducted: d)□ Yes e)☒ No.
If Yes, brief description: ____ .

Claim(s) discussed: Independent claims .

Identification of prior art discussed: LeVine & Bailey .

Agreement with respect to the claims f)□ was reached. g)☒ was not reached. h)□ N/A.

Substance of Interview including description of the general nature of what was agreed to if an agreement was reached, or any other comments: See below .

(A fuller description, if necessary, and a copy of the amendments which the examiner agreed would render the claims allowable, if available, must be attached. Also, where no copy of the amendments that would render the claims allowable is available, a summary thereof must be attached.)

i)☑ It is not necessary for applicant to provide a separate record of the substance of the interview(if box is checked).

Unless the paragraph above has been checked, THE FORMAL WRITTEN REPLY TO THE LAST OFFICE ACTION MUST INCLUDE THE SUBSTANCE OF THE INTERVIEW. (See MPEP Section 713.04). If a reply to the last Office action has already been filed, APPLICANT IS GIVEN ONE MONTH FROM THIS INTERVIEW DATE TO FILE A STATEMENT OF THE SUBSTANCE OF THE INTERVIEW. See Summary of Record of Interview requirements on reverse side or on attached sheet.

Applicants' representative believes that the invention as claimed does define over the prior art of record. LeVine & Bailey. Examiner disagrees. Examiner has decided to review & go through LeVine & Bailey prior art for further consideration.

Examiner Note: You must sign this form unless it is an Attachment to a signed Office action.

Examiner's signature, if required

ımmary of Record of Interview Requireme.

Manual of Patent Examining Procedure (MPEP), Section 713.04, Substance of Interview Must be Made of Record
A complete written statement as to the substance of any face-to-face, video conference, or telephone interview with regard to an application must be made of record in the application whether or not an agreement with the examiner was reached at the interview.

Title 37 Code of Federal Regulations (CFR) § 1.133 Interviews
Paragraph (b)
In every instance where reconsideration is requested in view of an interview with an examiner, a complete written statement of the reasons presented at the interview as warranting favorable action must be filed by the applicant. An interview does not remove the necessity for reply to Office action as specified in §§ 1.111, 1.135. (35 U.S.C. 132)

37 CFR §1.2 Business to be transacted in writing.
All business with the Patent or Trademark Office should be transacted in writing. The personal attendance of applicants or their attorneys or agents at the Patent and Trademark Office is unnecessary. The action of the Patent and Trademark Office will be based exclusively on the written record in the Office. No attention will be paid to any alleged oral promise, stipulation, or understanding in relation to which there is disagreement or doubt.

The action of the Patent and Trademark Office cannot be based exclusively on the written record in the Office if that record is itself incomplete through the failure to record the substance of interviews.

It is the responsibility of the applicant or the attorney or agent to make the substance of an interview of record in the application file, unless the examiner indicates he or she will do so. It is the examiner's responsibility to see that such a record is made and to correct material inaccuracies which bear directly on the question of patentability.

Examiners must complete an Interview Summary Form for each interview held where a matter of substance has been discussed during the interview by checking the appropriate boxes and filling in the blanks. Discussions regarding only procedural matters, directed solely to restriction requirements for which interview recordation is otherwise provided for in Section 812.01 of the Manual of Patent Examining Procedure, or pointing out typographical errors or unreadable script in Office actions or the like, are excluded from the interview recordation procedures below. Where the substance of an interview is completely recorded in an Examiners Amendment, no separate Interview Summary Record is required.

The Interview Summary Form shall be given an appropriate Paper No., placed in the right hand portion of the file, and listed on the "Contents" section of the file wrapper. In a personal interview, a duplicate of the Form is given to the applicant (or attorney or agent) at the conclusion of the interview. In the case of a telephone or video-conference interview, the copy is mailed to the applicant's correspondence address either with or prior to the next official communication. If additional correspondence from the examiner is not likely before an allowance or if other circumstances dictate, the Form should be mailed promptly after the interview rather than with the next official communication.

The Form provides for recordation of the following information:
- Application Number (Series Code and Serial Number)
- Name of applicant
- Name of examiner
- Date of interview
- Type of interview (telephonic, video-conference, or personal)
- Name of participant(s) (applicant, attorney or agent, examiner, other PTO personnel, etc.)
- An indication whether or not an exhibit was shown or a demonstration conducted
- An identification of the specific prior art discussed
- An indication whether an agreement was reached and if so, a description of the general nature of the agreement (may be by attachment of a copy of amendments or claims agreed as being allowable). Note: Agreement as to allowability is tentative and does not restrict further action by the examiner to the contrary.
- The signature of the examiner who conducted the interview (if Form is not an attachment to a signed Office action)

It is desirable that the examiner orally remind the applicant of his or her obligation to record the substance of the interview of each case *unless both applicant and examiner agree that the examiner will record same.* Where the examiner agrees to record the substance of the interview, or when it is adequately recorded on the Form or in an attachment to the Form, the examiner should check the appropriate box at the bottom of the Form which informs the applicant that the submission of a separate record of the substance of the interview as a supplement to the Form is not required.

It should be noted, however, that the Interview Summary Form will not normally be considered a complete and proper recordation of the interview unless it includes, or is supplemented by the applicant or the examiner to include, all of the applicable items required below concerning the substance of the interview.

A complete and proper recordation of the substance of any interview should include at least the following applicable items:
1) A brief description of the nature of any exhibit shown or any demonstration conducted,
2) an identification of the claims discussed,
3) an identification of the specific prior art discussed,
4) an identification of the principal proposed amendments of a substantive nature discussed, *unless these are already described on the* Interview Summary Form completed by the Examiner,
5) a brief identification of the general thrust of the principal arguments presented to the examiner,
(The identification of arguments need not be lengthy or elaborate. A verbatim or highly detailed description of the arguments is not required. The identification of the arguments is sufficient if the general nature or thrust of the principal arguments made to the examiner can be understood in the context of the application file. Of course, the applicant may desire to emphasize and fully describe those arguments which he or she feels were or might be persuasive to the examiner.)
6) a general indication of any other pertinent matters discussed, and
7) if appropriate, the general results or outcome of the interview unless already described in the Interview Summary Form completed by the examiner.

Examiners are expected to carefully review the applicant's record of the substance of an interview. If the record is not complete and accurate, the examiner will give the applicant an extendable one month time period to correct the record.

Examiner to Check for Accuracy

If the claims are allowable for other reasons of record, the examiner should send a letter setting forth the examiner's version of the statement attributed to him or her. If the record is complete and accurate, the examiner should place the indication, "Interview Record OK" on the paper recording the substance of the interview along with the date and the examiner's initials.

Notice of Allowability	**Application No.** 09/608,872	**Applicant(s)** HALVERSEN ET AL.
	Examiner Frantz B. Jean	**Art Unit** 2155

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to the response filed on 1/06/2003.
2. ☒ The allowed claim(s) is/are 56-82.
3. ☐ The drawings filed on _____ are accepted by the Examiner.
4. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

   a) ☐ All b) ☐ Some* c) ☐ None of the:

   1. ☐ Certified copies of the priority documents have been received.
   2. ☐ Certified copies of the priority documents have been received in Application No. _____.
   3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.
5. ☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

   (a) ☐ The translation of the foreign language provisional application has been received.
6. ☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application. ***THIS THREE-MONTH PERIOD IS NOT EXTENDABLE***

7. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.

8. ☒ CORRECTED DRAWINGS must be submitted.

   (a) ☒ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached

   1) ☒ hereto or 2) ☐ to Paper No. ____.

   (b) ☐ including changes required by the proposed drawing correction filed _____, which has been approved by the Examiner.

   (c) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No. _____.

Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the top margin (not the back) of each sheet. The drawings should be filed as a separate paper with a transmittal letter addressed to the Official Draftsperson.

9. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1☐ Notice of References Cited (PTO-892)

2☐ Notice of Informal Patent Application (PTO-152)

3☒ Notice of Draftperson's Patent Drawing Review (PTO-948)

4☐ Interview Summary (PTO-413), Paper No.____.

5☐ Information Disclosure Statements (PTO-1449), Paper No. ____.

6☐ Examiner's Amendment/Comment

7☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material

8☒ Examiner's Statement of Reasons for Allowance

9☐ Other

1. Claims 56-82 are allowable over the prior art made of record and in light of Applicants' arguments..

2. The response filed on 01/08/2003 has been entered.

Reasons for Allowance

3. The examiner respectfully submits that the specific techniques of providing a speech-based navigation where a spoken request for desired information is received from a user utilizing a mobile information appliance of the user, wherein the mobile information appliance comprises a portable remote control device or a set-top box for a television; in conjunction with the other limitations of the dependent and independent claims 56-82 were not shown by, would not have been obvious over, nor would have been fairly suggested by the prior art made of record.

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

4. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Frantz B. Jean whose telephone number is (703) 305-3970. The examiner can normally be reached on Monday thru Friday from 8:30 to 6:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Ayaz R. Sheikh, can be reached on (703) 305-9648. The fax phone numbers for this Group are (703) 746-7238 for After-Final, (703) 746-7239 for Official, and (703) 746-7240 for Non-Official/Draft.

Communications via Internet e-mail regarding this application, other than those under 35 U.S.C. 132 or which otherwise require a signature, may be used by the applicant and should be addressed to [Ayaz.Sheikh@uspto.gov].

All Internet e-mail communications will be made of record in the application file. PTO employees do not engage in Internet communications where there exists a possibility that sensitive information could be identified or exchanged unless the record includes a properly signed express waiver of the confidentiality requirements of 35 U.S.C. 122. This is more clearly set forth in the Interim Internet Usage Policy published in the Official Gazette of the Patent and Trademark on February 25, 1997 at 1195 OG 89.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the Group receptionist whose telephone number is (703) 305-3900.

Frantz B. Jean
March 07, 2003
FBJ/

Form PTO 948 (Rev. 03/01) U.S. DEPARTMENT OF COMMERCE - Patent and Trademark Office Application No. 09/608872

NOTICE OF DRAFTSPERSON'S PATENT DRAWING REVIEW

The drawing(s) filed (insert date) 6-30-00 are:

A. □ approved by the Draftsperson under 37 CFR 1.84 or 1.152.

B. ☒ objected to by the Draftsperson under 37 CFR 1.84 or 1.152 for the reasons indicated below. The Examiner will require submission of new, corrected drawings when necessary. Corrected drawing must be sumitted according to the instructions on the back of this notice.

1. DRAWINGS. 37 CFR 1.84(a): Acceptable categories of drawings: Black ink. Color.
   ____ Color drawings are not acceptable until petiton is granted. Fig(s) ____
   ____ Pencil and non black ink not permitted. Fig(s) ____
2. PHOTOGRAPHS. 37 CFR 1.84(b)
   ____ 1 full-tone set is required. Fig(s) ____
   ____ Photographs may not be mounted. 37 CFR 1.84(e)
   ____ Poor quality (half-tone). Fig(s) ____
3. TYPE OF PAPER. 37 CFR 1.84(e)
   ____ Paper not flexible, strong, white, and durable. Fig(s) ____
   ____ Erasures, alterations, overwritings, interlineations, folds, copy machine marks not accepted. Fig(s) ____
   ____ Mylar, velum paper is not acceptable (too thin). Fig(s) ____
4. SIZE OF PAPER. 37 CFR 1.84(f): Acceptable sizes:
   ____ 21.0 cm by 29.7 cm (DIN size A4)
   ____ 21.6 cm by 27.9 cm (8 1/2 x 11 inches)
   ____ All drawing sheets not the same size. Sheet(s) ____
   ____ Drawings sheets not an acceptable size. Fig(s) ____
5. MARGINS. 37 CFR 1.84(g): Acceptable margins:

   Top 2.5 cm Left 2.5cm Right 1.5 cm Bottom 1.0 cm
   SIZE: A4 Size
   Top 2.5 cm Left 2.5 cm Right 1.5 cm Bottom 1.0 cm
   SIZE: 8 1/2 x 11
   Margins not acceptable. Fig(s) ____
   ____ Top (T) ____ Left (L)
   ____ Right (R) ____ Bottom (B)
6. VIEWS. 37 CFR 1.84(h)
   REMINDER: Specification may require revision to correspond to drawing changes.
   Partial views. 37 CFR 1.84(h)(2)
   ____ Brackets needed to show figure as one entity. Fig(s) ____
   ____ Views not labeled separately or properly. Fig(s) ____
   ____ Enlarged view not labeled separetely or properly. Fig(s) ____
7. SECTIONAL VIEWS. 37 CFR 1.84 (h)(3)
   ____ Hatching not indicated for sectional portions of an object. Fig(s) ____
   ____ Sectional designation should be noted with Arabic or Roman numbers. Fig(s) ____
8. ARRANGEMENT OF VIEWS. 37 CFR 1.84(i)
   ____ Words do not appear on a horizontal, left-to-right fashion when page is either upright or turned so that the top becomes the right side, except for graphs. Fig(s) ____
9. SCALE. 37 CFR 1.84(k)
   ____ Scale not large enough to show mechanism without crowding when drawing is reduced in size to two-thirds in reproduction. Fig(s) ____
10. CHARACTER OF LINES, NUMBERS, & LETTERS. 37 CFR 1.84(l)
   X Lines, numbers & letters not uniformly thick and well defined, clean, durable, and black (poor line quality). Fig(s) ALL
11. SHADING. 37 CFR 1.84(m)
   ____ Solid black areas pale. Fig(s) ____
   X Solid black shading not permitted. Fig(s) 6
   ____ Shade lines, pale, rough and blurred. Fig(s) ____
12. NUMBERS, LETTERS, & REFERENCE CHARACTERS. 37 CFR 1.84(p)
   X Numbers and reference characters not plain and legible. Fig(s) ALL
   X Figure legends are poor. Fig(s) ALL
   ____ Numbers and reference characters not oriented in the same direction as the view. 37 CFR 1.84(p)(1) Fig(s) ____
   ____ English alphabet not used. 37 CFR 1.84(p)(2) Figs ____
   ____ Numbers, letters and reference characters must be at least .32 cm (1/8 inch) in height. 37 CFR 1.84(p)(3) Fig(s) ____
13. LEAD LINES. 37 CFR 1.84(q)
   ____ Lead lines cross each other. Fig(s) ____
   ____ Lead lines missing. Fig(s) ____
14. NUMBERING OF SHEETS OF DRAWINGS. 37 CFR 1.84(t)
   ____ Sheets not numbered consecutively, and in Arabic numerals beginning with number 1. Sheet(s) ____
15. NUMBERING OF VIEWS. 37 CFR 1.84(u)
   ____ Views not numbered consecutively, and in Arabic numerals, beginning with number 1. Fig(s) ____
16. CORRECTIONS. 37 CFR 1.84(w)
   ____ Corrections not made from prior PTO-948 dated ____
17. DESIGN DRAWINGS. 37 CFR 1.152
   ____ Surface shading shown not appropriate. Fig(s) ____
   ____ Solid black shading not used for color contrast. Fig(s) ____

COMMENTS

REVIEWER ____ DATE 3-10-03 TELEPHONE NO. ____

ATTACHMENT TO PAPER NO. 27

Attachment for PTO-948 (Rev. 03/01, or earlier)
6/18/01

The below text replaces the pre-printed text under the heading, "Information on How to Effect Drawing Changes," on the back of the PTO-948 (Rev. 03/01, or earlier) form.

INFORMATION ON HOW TO EFFECT DRAWING CHANGES

1. Correction of Informalities -- 37 CFR 1.85

New corrected drawings must be filed with the changes incorporated therein Identifying indicia, if provided, should include the title of the invention, inventor's name, and application number, or docket number (if any) if an application number has not been assigned to the application If this information is provided, it must be placed on the front of each sheet and centered within the top margin If corrected drawings are required in a Notice of Allowability (PTOL-37), the new drawings **MUST** be filed within the **THREE MONTH** shortened statutory period set for reply in the Notice of Allowability Extensions of time may NOT be obtained under the provisions of 37 CFR 1.136(a) or (b) for filing the corrected drawings after the mailing of a Notice of Allowability The drawings should be filed as a separate paper with a transmittal letter addressed to the Official Draftsperson

2. Corrections other than Informalities Noted by Draftsperson on form PTO-948.

All changes to the drawings, other than informalities noted by the Draftsperson, **MUST** be made in the same manner as above except that, normally, a highlighted (preferably red ink) sketch of the changes to be incorporated into the new drawings MUST be approved by the examiner before the application will be allowed No changes will be permitted to be made, other than correction of informalities, unless the examiner has approved the proposed changes

Timing of Corrections

Applicant is required to submit the drawing corrections within the time period set in the attached Office communication See 37 CFR 1.85(a).

Failure to take corrective action within the set period will result in ABANDONMENT of the application

06/01/01

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

#27

NOTICE OF ALLOWANCE AND FEE(S) DUE

7590 03/11/2003

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY, NJ 07702

EXAMINER	
JEAN, FRANTZ B	
ART UNIT	CLASS-SUBCLASS
2155	709-218000

DATE MAILED: 03/11/2003

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/608,872	06/30/2000	Christine Halversen	SRILP037B	2382

TITLE OF INVENTION: MOBILE NAVIGATION OF NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN INPUT

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	YES	$650	$0	$650	06/11/2003

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED. THIS NOTICE OF ALLOWANCE IS NOT A GRANT OF PATENT RIGHTS. THIS APPLICATION IS SUBJECT TO WITHDRAWAL FROM ISSUE AT THE INITIATIVE OF THE OFFICE OR UPON PETITION BY THE APPLICANT. SEE 37 CFR 1.313 AND MPEP 1308.

THE ISSUE FEE AND PUBLICATION FEE (IF REQUIRED) MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED. SEE 35 U.S.C. 151. THE ISSUE FEE DUE INDICATED ABOVE REFLECTS A CREDIT FOR ANY PREVIOUSLY PAID ISSUE FEE APPLIED IN THIS APPLICATION. THE PTOL-85B (OR AN EQUIVALENT) MUST BE RETURNED WITHIN THIS PERIOD EVEN IF NO FEE IS DUE OR THE APPLICATION WILL BE REGARDED AS ABANDONED.

HOW TO REPLY TO THIS NOTICE:

I. Review the SMALL ENTITY status shown above.

If the SMALL ENTITY is shown as YES, verify your current SMALL ENTITY status:

A. If the status is the same, pay the TOTAL FEE(S) DUE shown above.

B. If the status is changed, pay the PUBLICATION FEE (if required) and twice the amount of the ISSUE FEE shown above and notify the United States Patent and Trademark Office of the change in status, or

If the SMALL ENTITY is shown as NO:

A. Pay TOTAL FEE(S) DUE shown above, or

B. If applicant claimed SMALL ENTITY status before, or is now claiming SMALL ENTITY status, check the box below and enclose the PUBLICATION FEE and 1/2 the ISSUE FEE shown above.

□ Applicant claims SMALL ENTITY status.
See 37 CFR 1.27.

II. PART B - FEE(S) TRANSMITTAL should be completed and returned to the United States Patent and Trademark Office (USPTO) with your ISSUE FEE and PUBLICATION FEE (if required). Even if the fee(s) have already been paid, Part B - Fee(s) Transmittal should be completed and returned. If you are charging the fee(s) to your deposit account, section "4b" of Part B - Fee(s) Transmittal should be completed and an extra copy of the form should be submitted.

III. All communications regarding this application must give the application number. Please direct all communications prior to issuance to Box ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

PTOL-85 (REV. 04-02) Approved for use through 01/31/2004.

PART B - FEE(S) TRANSMITTAL

---,lete and send this form, together with applicable fee(s), to: Mail **Box ISSUE FEE**
Commissioner for Patents
Washington, D.C. 20231
Fax (703)746-4000

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 4 should be completed where appropriate. *All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as* indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Legibly mark-up with any corrections or use Block 1)

7590 03/11/2003

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY, NJ 07702

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Box Issue Fee address above, or being facsimile transmitted to the USPTO, on the date indicated below.

(Depositor's name)
(Signature)
(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/608,872	06/30/2000	Christine Halversen	SRILP037B	2382

TITLE OF INVENTION: MOBILE NAVIGATION OF NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN INPUT

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	YES	$650	$0	$650	06/11/2003

EXAMINER	ART UNIT	CLASS-SUBCLASS
JEAN, FRANTZ B	2155	709-218000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).

□ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.

□ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list (1) the names of up to 3 registered patent attorneys or agents OR, alternatively, (2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

1 ____________

2 ____________

3 ____________

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. Inclusion of assignee data is only appropriate when an assignment has been previously submitted to the USPTO or is being submitted under separate cover. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE

(B) RESIDENCE: (CITY and STATE OR COUNTRY)

Please check the appropriate assignee category or categories (will not be printed on the patent) □ individual □ corporation or other private group entity □ government

4a. The following fee(s) are enclosed:

□ Issue Fee

□ Publication Fee

□ Advance Order - # of Copies ____________

4b. Payment of Fee(s):

□ A check in the amount of the fee(s) is enclosed.

□ Payment by credit card. Form PTO-2038 is attached.

□ The Commissioner is hereby authorized by charge the required fee(s), or credit any overpayment, to Deposit Account Number ____________ (enclose an extra copy of this form).

Commissioner for Patents is requested to apply the Issue Fee and Publication Fee (if any) or to re-apply any previously paid issue fee to the application identified above.

(Authorized Signature) (Date)

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. *Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection* is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, Washington, D.C. 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMIT THIS FORM WITH FEE(S)

PTOL-85 (REV. 04-02) Approved for use through 01/31/2004. OMB 0651-0033 U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/608,872	06/30/2000	Christine Halversen	SRILP037B	2382

7590 03/11/2003

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY, NJ 07702

EXAMINER
JEAN, FRANTZ B

ART UNIT	PAPER NUMBER
2155	

DATE MAILED: 03/11/2003

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(application filed on or after May 29, 2000)

The patent term adjustment to date is 0 days. If the issue fee is paid on the date that is three months after the mailing date of this notice and the patent issues on the Tuesday before the date that is 28 weeks (six and a half months) after the mailing date of this notice, the term adjustment will be 0 days.

If a continued prosecution application (CPA) was filed in the above-identified application, the filing date that determines patent term adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) system. (http://pair.uspto.gov)

Any questions regarding the patent term extension or adjustment determination should be directed to the Office of Patent Legal Administration at (703)305-1383.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/608,872	06/30/2000	Christine Halversen	SRILP037B	2382

7590 03/11/2003

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY, NJ 07702
UNITED STATES

EXAMINER	
JEAN, FRANTZ B	
ART UNIT	PAPER NUMBER
2155	

DATE MAILED: 03/11/2003

Notice of Fee Increase on January 1, 2003

If a reply to a "Notice of Allowance and Fee(s) Due" is filed in the Office on or after January 1, 2003, then the amount due will be higher than that set forth in the "Notice of Allowance and Fee(s) Due" since there will be an increase in fees effective on January 1, 2003. See Revision of Patent and Trademark Fees for Fiscal Year 2003; Final Rule, 67 Fed. Reg. 70847, 70849 (November 27, 2002).

The current fee schedule is accessible from: http://www.uspto.gov/main/howtofees.htm.

If the issue fee paid is the amount shown on the "Notice of Allowance and Fee(s) Due," but not the correct amount in view of the fee increase, a "Notice to Pay Balance of Issue Fee" will be mailed to applicant. In order to avoid processing delays associated with mailing of a "Notice to Pay Balance of Issue Fee," if the response to the Notice of Allowance and Fee(s) due form is to be filed on or after January 1, 2003 (or mailed with a certificate of mailing on or after January 1, 2003), the issue fee paid should be the fee that is required at the time the fee is paid. If the issue fee was previously paid, and the response to the "Notice of Allowance and Fee(s) Due" includes a request to apply a previously-paid issue fee to the issue fee now due, then the difference between the issue fee amount at the time the response is filed and the previously paid issue fee should be paid. See Manual of Patent Examining Procedure, Section 1308.01 (Eighth Edition, August 2001).

Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at (703) 305-8283.

#28
JL

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Halverson, et al.

Serial No.: 09/608,872 Art Unit: 2155

Filing Date: June 30, 2000 Examiner: Jean, Frantz B

For: MOBILE NAVIGATION OF NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN INPUT

Docket No. SRI 4116-6

Assistant Commissioner for Patents
Washington, D.C. 20231
S I R:

SUBMISSION OF FORMAL DRAWINGS

The Applicants submit herewith 7 sheets of formal drawings (FIGS. 1 through 6), properly labeled, in connection with the above-captioned application. The Examiner is requested to substitute these formal drawings for the informal drawings previously submitted.

Respectfully submitted,

Dated: 4/29/03

KIN-WAH TONG
Reg. No. 39,400
(732) 530-9404

Moser, Patterson & Sheridan, LLP
595 Shrewsbury Avenue
Suite 100
Shrewsbury, NJ 07702

CERTIFICATE OF MAILING under 37 C.F.R. 1.8(a)

I hereby certify that this correspondence is being deposited on April 30 2003, with the United States Postal Service as first class mail, with sufficient postage, in an envelope addressed to the Commissioner for Patents, Box Issue Fee, Washington, D.C. 20231.

Signature

April 30, 2003
Date of signature

6757718

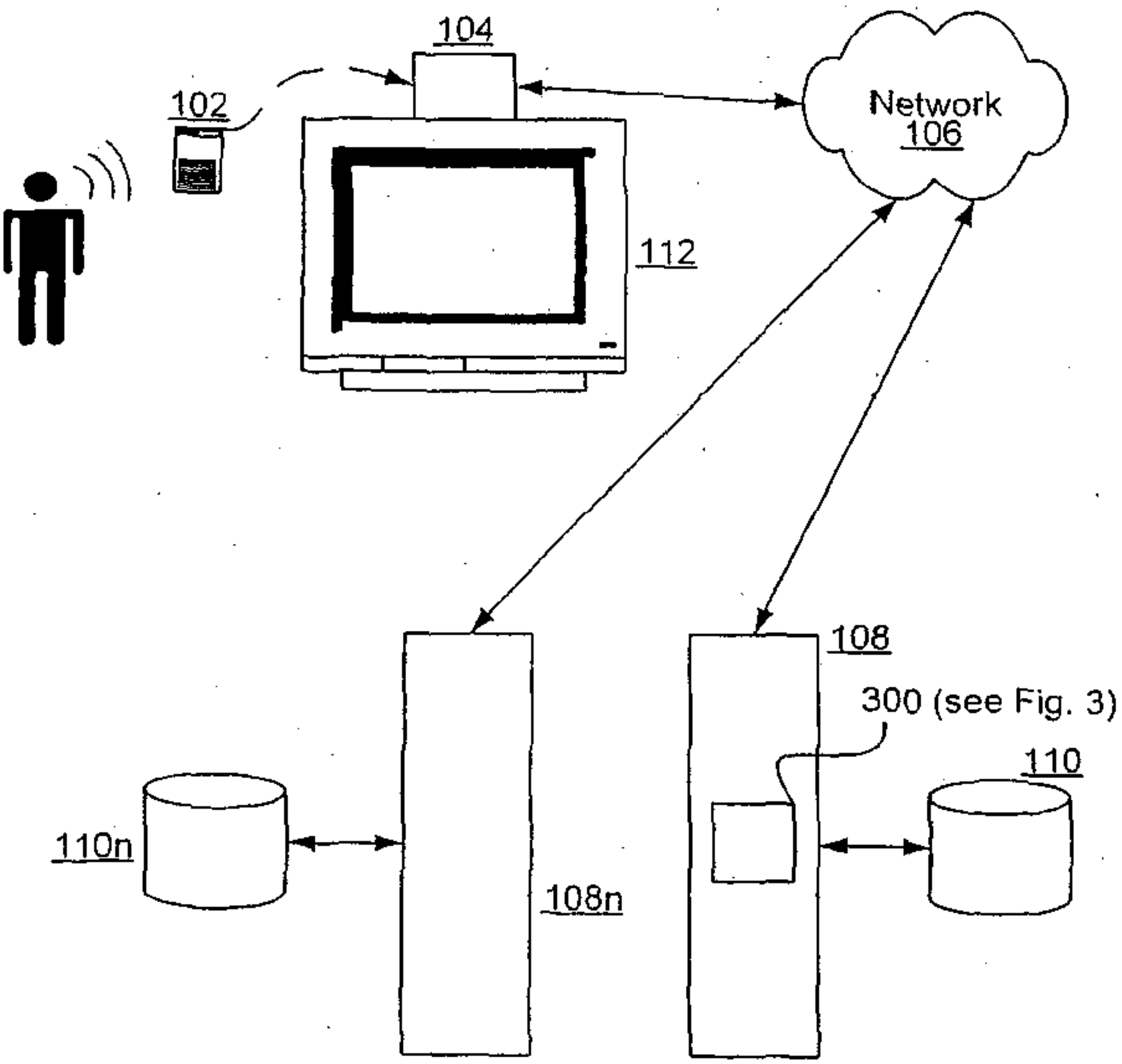


Fig. 1a

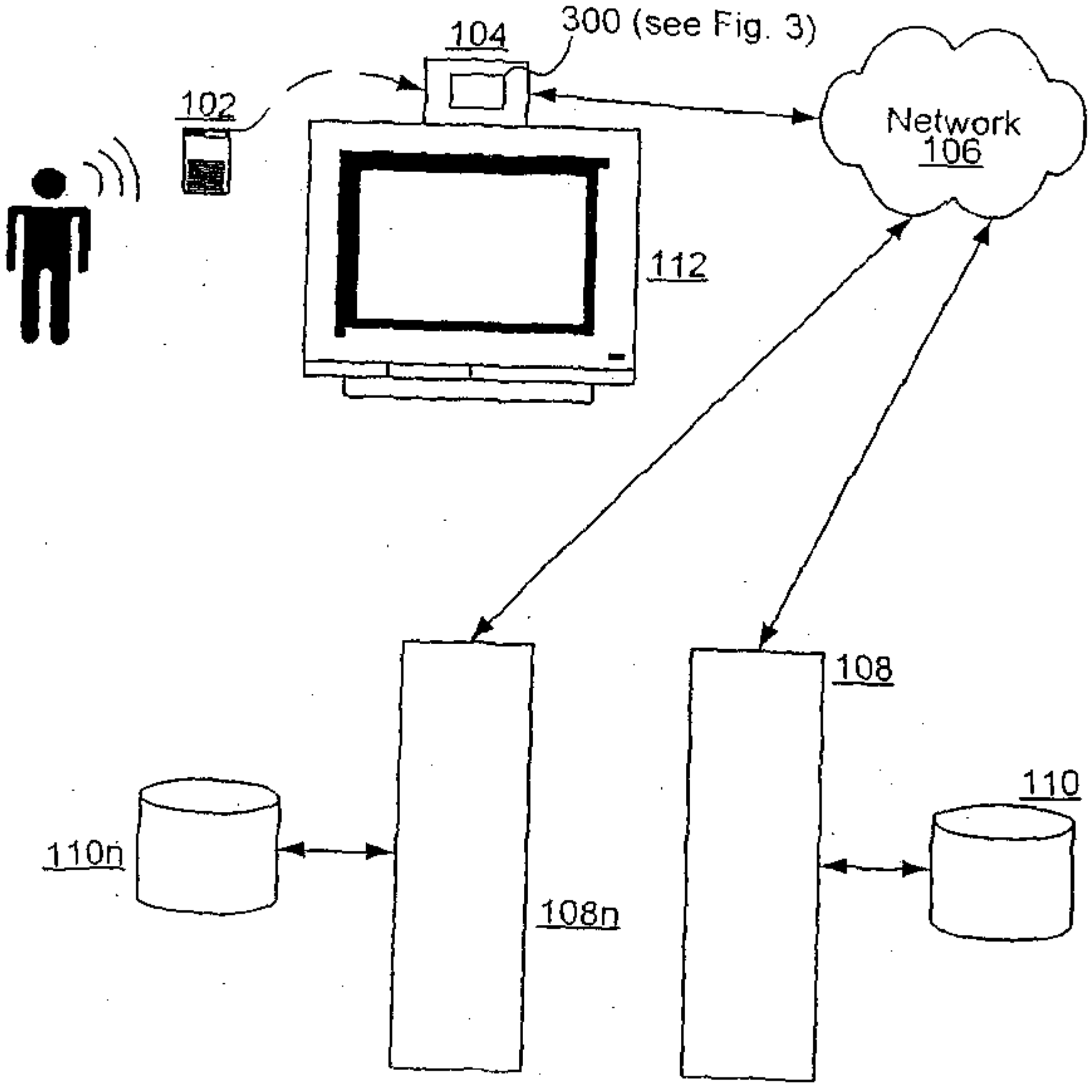


Fig. 1b

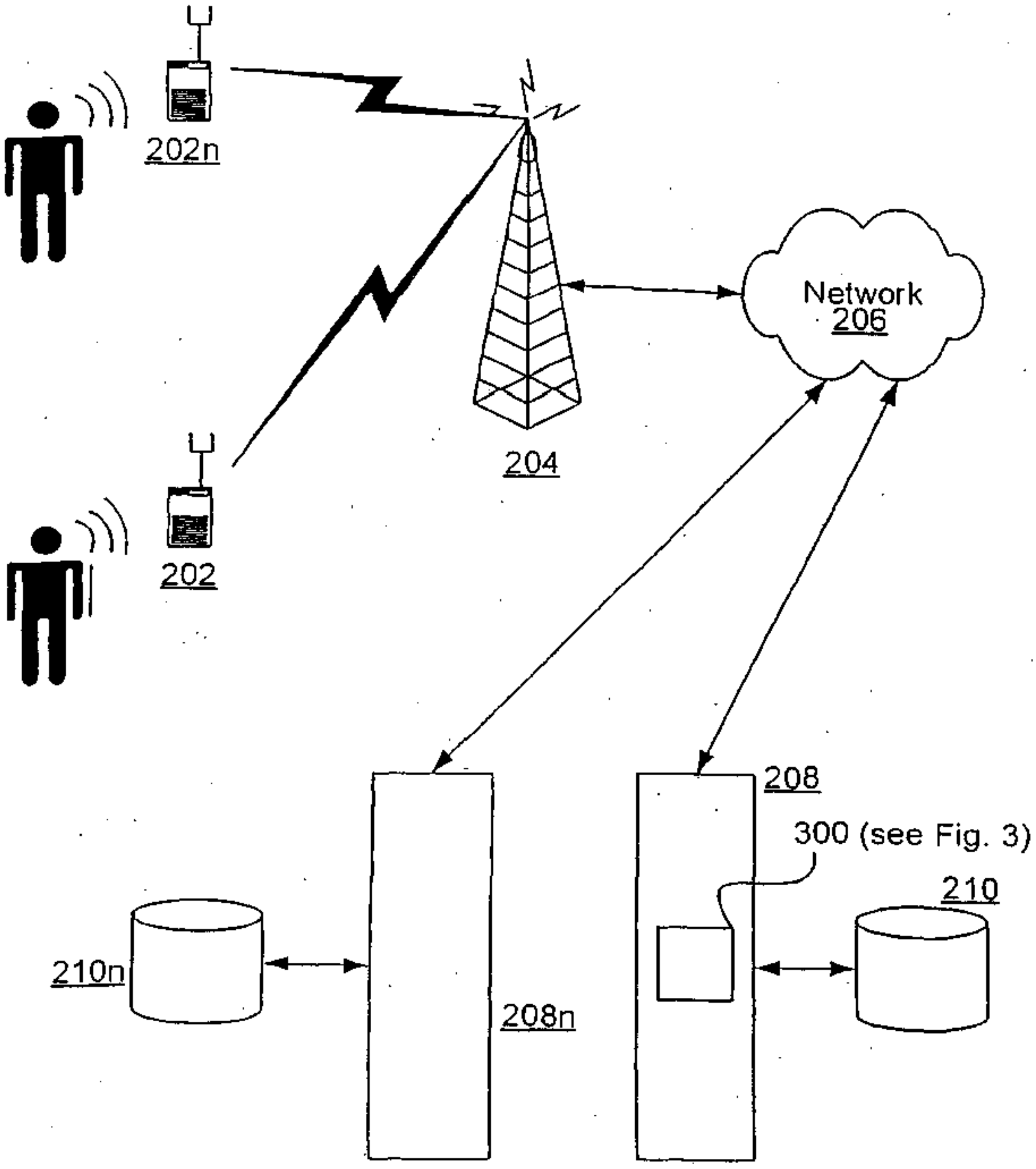


Fig. 2

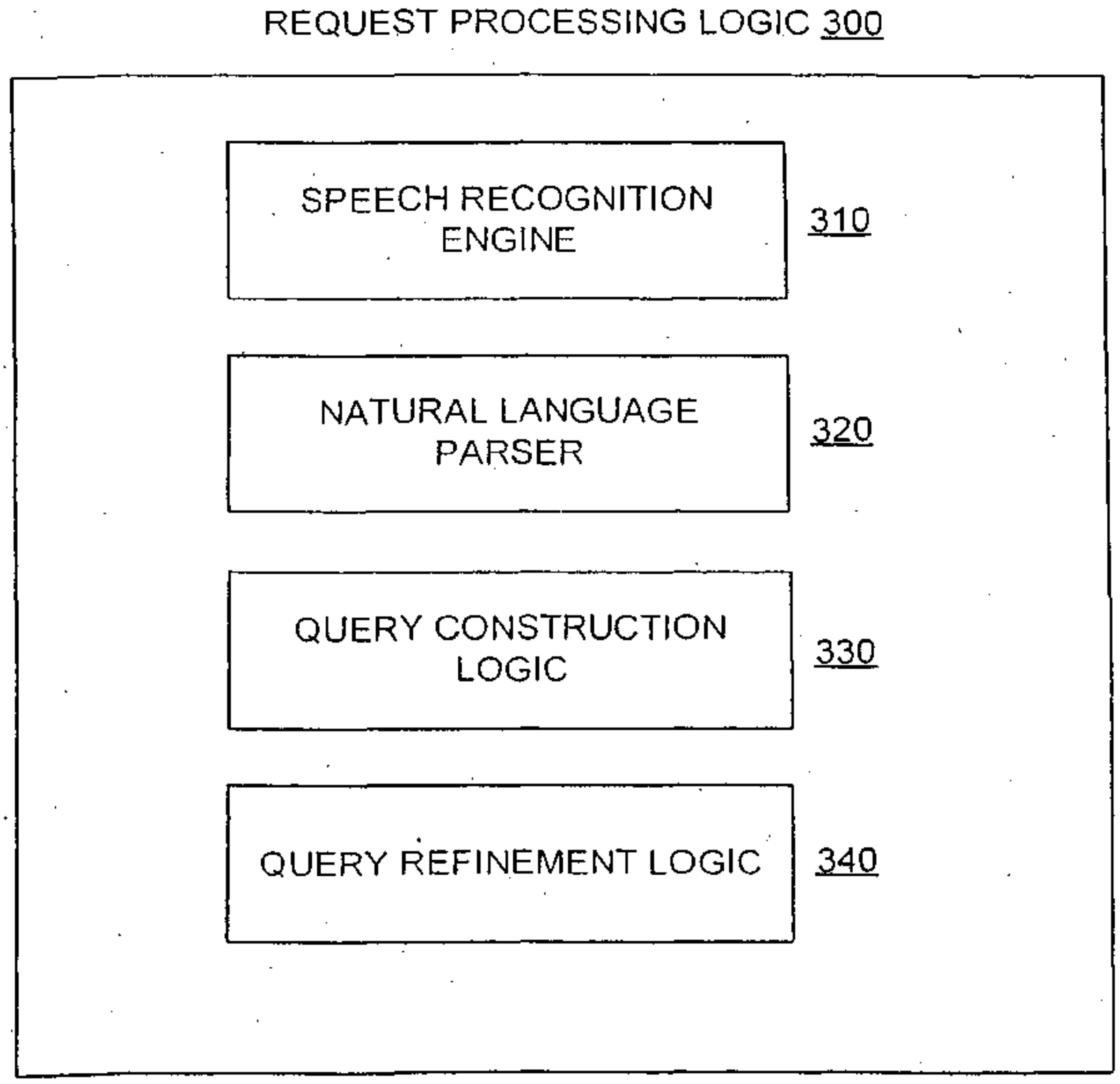

REQUEST PROCESSING LOGIC 300
SPEECH RECOGNITION ENGINE
310
NATURAL LANGUAGE PARSER
320
QUERY CONSTRUCTION LOGIC
330
QUERY REFINEMENT LOGIC
340

Fig. 3

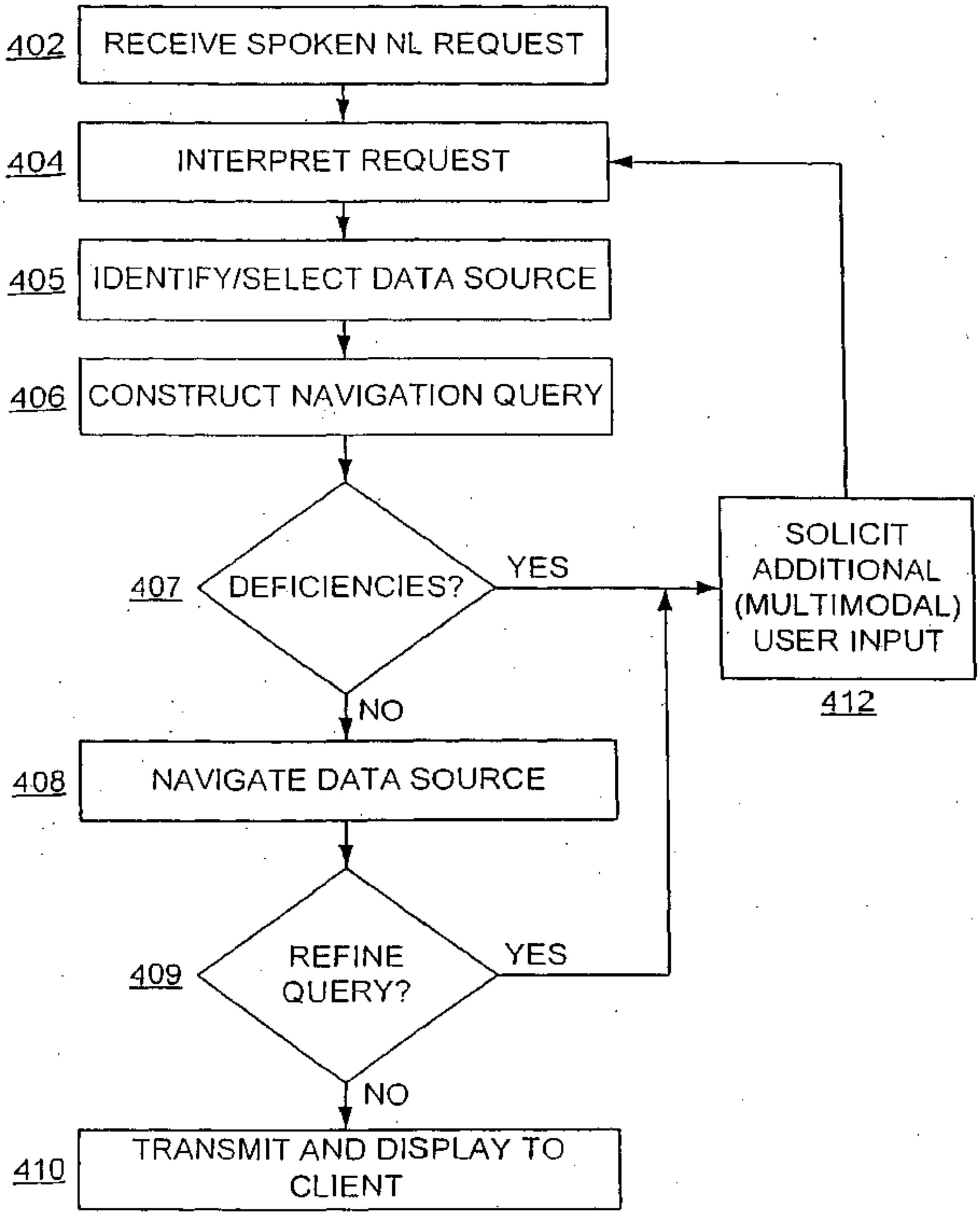

402
RECEIVE SPOKEN NL REQUEST
404
INTERPRET REQUEST
405
IDENTIFY/SELECT DATA SOURCE
406
CONSTRUCT NAVIGATION QUERY
407
DEFICIENCIES?
YES
NO
SOLICIT ADDITIONAL (MULTIMODAL) USER INPUT
412
408
NAVIGATE DATA SOURCE
409
REFINE QUERY?
YES
NO
410
TRANSMIT AND DISPLAY TO CLIENT

Fig. 4

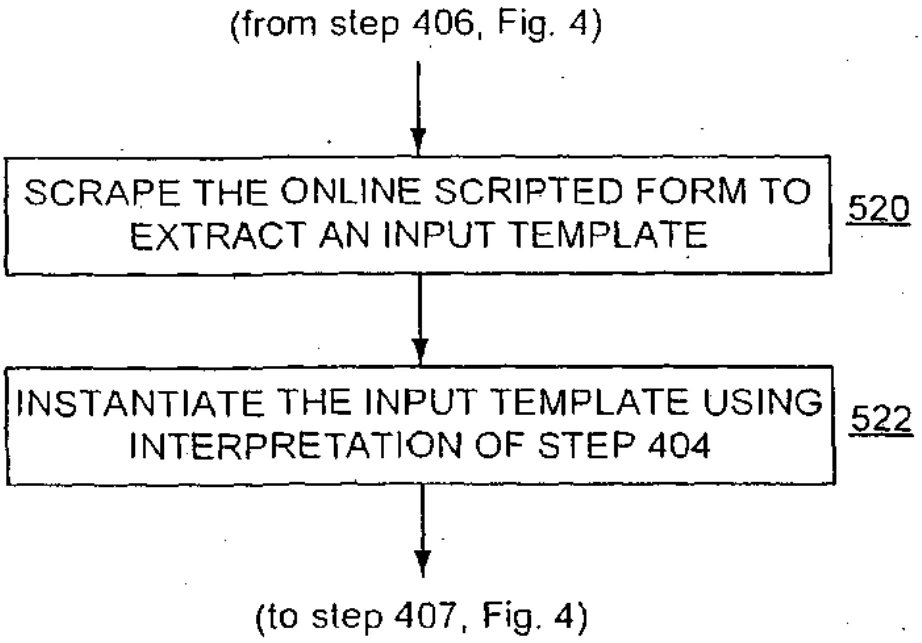

(from step 406, Fig. 4)
SCRAPE THE ONLINE SCRIPTED FORM TO EXTRACT AN INPUT TEMPLATE
520
INSTANTIATE THE INPUT TEMPLATE USING INTERPRETATION OF STEP 404
522
(to step 407, Fig. 4)

Fig. 5

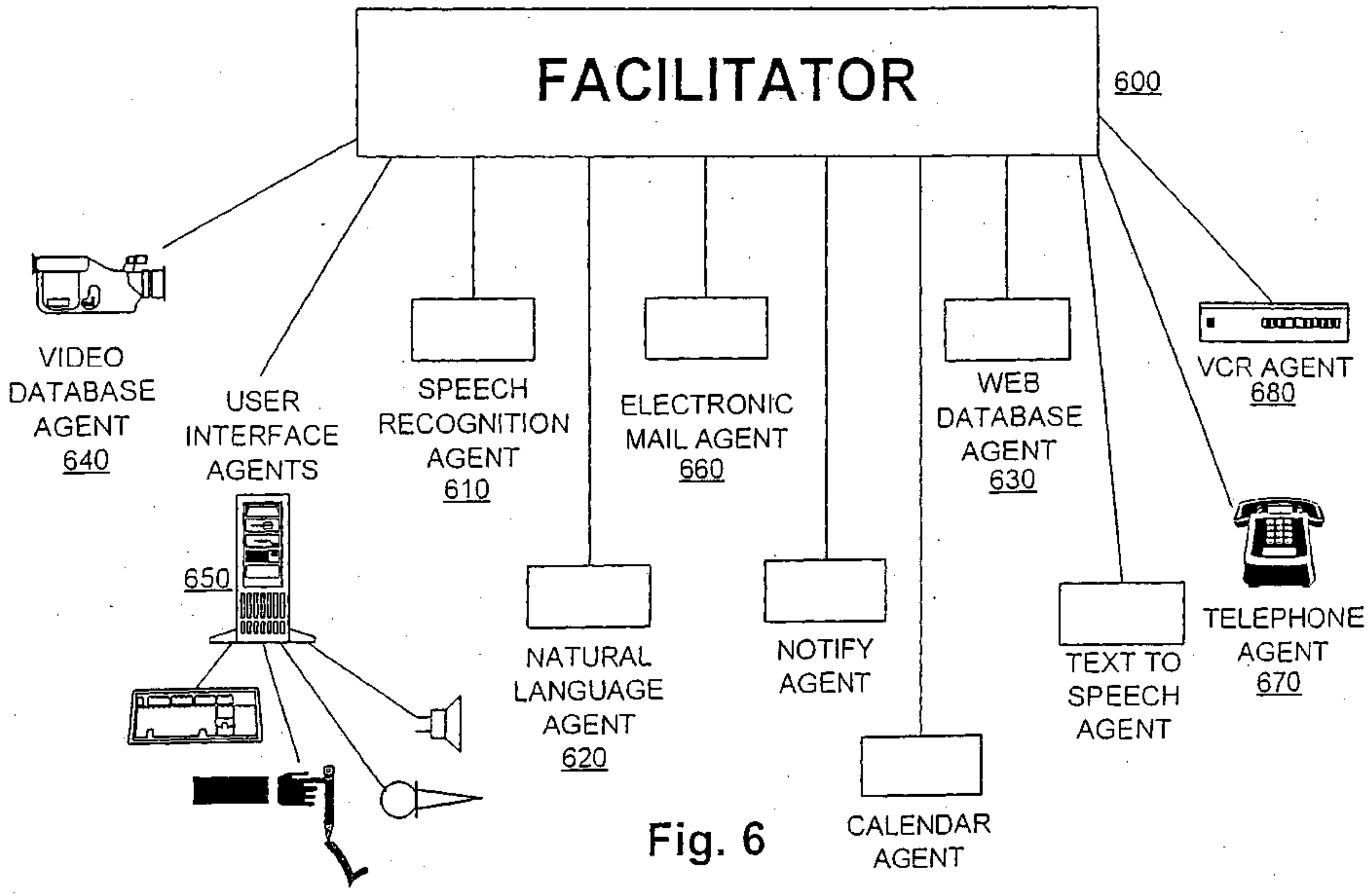

FACILITATOR
600
VIDEO DATABASE AGENT 640
USER INTERFACE AGENTS
650
SPEECH RECOGNITION AGENT 610
NATURAL LANGUAGE AGENT 620
ELECTRONIC MAIL AGENT 660
NOTIFY AGENT
CALENDAR AGENT
WEB DATABASE AGENT 630
TEXT TO SPEECH AGENT
VCR AGENT 680
TELEPHONE AGENT 670

Fig. 6

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: **Mail** **Box ISSUE FEE**
Commissioner for Patents
Washington, D.C. 20231
Fax (703)746-4000

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 4 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as [illegible] corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Legibly mark-up with any corrections or use Block 1)

7590 03/11/2003

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY, NJ 07702

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or *formal drawing*, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the *United States Postal Service with sufficient postage for first class mail* in an envelope addressed to the Box Issue Fee address above, or being facsimile transmitted to the USPTO, on the date indicated below.

Barbara J. Jackson	(Depositor's name)
Barbara J. Jackson	(Signature)
April 30, 2003	(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/608,872	06/30/2000	Christine Halverson	SRILP037B	2382

TITLE OF INVENTION: MOBILE NAVIGATION OF NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN INPUT

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	YES	$650	$0	$650	06/11/2003

EXAMINER	ART UNIT	CLASS-SUBCLASS
JEAN, FRANTZ B	2155	709-218000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).

□ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.

□ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. Use of a Customer Number is required.

2. For printing on the patent front page, list (1) *the names of up to 3* registered patent attorneys or agents OR, alternatively, (2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

1 Moser, Patterson & Sheridan, LLP
2 Kin-Wah Tong
3 ____________

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED *ON THE PATENT (print or type)*

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. Inclusion of assignee data is only appropriate when an assignment has been previously submitted to the USPTO or is being submitted under separate cover. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE: SRI International

(B) RESIDENCE: *(CITY and STATE OR COUNTRY)*: Menlo Park, CA

Please check the appropriate assignee category or categories (will not be printed on the patent) □ individual ☒ corporation or other private group entity □ government

4a. The following fee(s) are enclosed:

☒ Issue Fee

□ Publication Fee

☒ Advance Order - # of Copies 1

4b. Payment of Fee(s):

☒ *A check in the amount of the fee(s) is enclosed.*

□ Payment by credit card. Form PTO-2038 is attached.

☒ The Commissioner is hereby authorized by charge the required fee(s), or credit any overpayment, to *Deposit Account Number* 20-0782 (enclose an extra copy of this form).

Commissioner for Patents is requested to apply the Issue Fee and Publication Fee (if any) or to re-apply any previously paid issue fee to the application identified above.

(Authorized Signature) (Date) 4/30/03

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, Washington, D.C. 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: *Commissioner for Patents, Washington, DC 20231.*

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

```
05/07/2003 SDENBOB2 00000052 09608872

01 FC:2501                          650.00 OP
02 FC:8001                            3.00 OP
```

TRANSMIT THIS FORM WITH FEE(S)

COD HEET FOR CONTINUING DA A

Line	Code	Serial No.	Filing Date	Status	Document No.	Issue Date
104	71	09/524,095	3/13/00			
105	82	09/225,198	1/5/99			
106	68	60/124,720	3/17/99			
107	68	60/124,719	3/17/99			
108	68	60/124,718	3/17/99			
109						
110						
111						
112						
113						
114						
115						
116						
117						

Condition and Status Codes for Continuing Data

CONDITION CODE:

71	Continuation of application No.
81	which is a continuation of application No.
91	and a continuation of application No.
72	Continuation-in-part of application No.
82	which is a continuation-in-part of application No.
75	and a continuation-in-part of application No.
74	Division of application No.
84	which is a division of application No.
76	and a division of application No.
86	, said application No.
89	Application No.
90	and application No.
92	each
65	filed as application No.
66	substitute for application No.
68	Provisional application No.

STATUS CODE

01	Patent No.
03	abandoned
04	SIR No.

NOTE I: When the code 86 and 92 are used, they must be followed by 81, 82 or 84 – condition beginning with "which is"
NOTE II: Codes 71, 72 and 74 may be used only on the first line; one of them must be used on the first line in regular continuing data. 66 or 68 may be used on the first line in Substitute or Provisional cases. Remember, however, that if there is a Provisional and other continuing data, the Provisional is always listed last.

Application or Docket Number

PATENT APPLICATION FEE DETERMINATION RECORD
Effective December 29, 1999

CLAIMS AS FILED - PART I

	(Column 1)		(Column 2)	SMALL ENTITY TYPE □		OR	OTHER THAN SMALL ENTITY	
FOR	NUMBER FILED		NUMBER EXTRA	RATE	FEE		RATE	FEE
BASIC FEE					345.00	OR		690.00
TOTAL CLAIMS	27	minus 20=	* 7	X$ 9=	63	OR	X$18=	
INDEPENDENT CLAIMS	3	minus 3 =	* /	X39=	/	OR	X78=	
MULTIPLE DEPENDENT CLAIM PRESENT				+130=	/	OR	+260=	
				TOTAL	408	OR	TOTAL	

* If the difference in column 1 is less than zero, enter "0" in column 2

CLAIMS AS AMENDED - PART II

		(Column 1)		(Column 2)	(Column 3)	SMALL ENTITY		OR	OTHER THAN SMALL ENTITY	
AMENDMENT A	C	CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE	ADDI-TIONAL FEE		RATE	ADDI-TIONAL FEE
	Total	* 27	Minus	** 27	=	X$ 9=		OR	X$18=	
	Independent	* 3	Minus	*** 3	=	X39=		OR	X78=	
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					+130=		OR	+260=	
						TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

		(Column 1)		(Column 2)	(Column 3)					
AMENDMENT B		CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE	ADDI-TIONAL FEE		RATE	ADDI-TIONAL FEE
	Total	* 27	Minus	** 27	= /	X$ 9=		OR	X$18=	
	Independent	* 3	Minus	*** 3	= /	X39=		OR	X78=	
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					+130=		OR	+260=	
						TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

		(Column 1)		(Column 2)	(Column 3)					
AMENDMENT C		CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE	ADDI-TIONAL FEE		RATE	ADDI-TIONAL FEE
	Total	*	Minus	**	=	X$ 9=		OR	X$18=	
	Independent	*	Minus	***	=	X39=		OR	X78=	
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					+130=		OR	+260=	
						TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20."
***If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3."
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

FORM PTO-875
(Rev. 12/99)

Patent and Trademark Office, U.S. DEPARTMENT OF COMMERCE

*U.S. GPO: 2000-463-433/29044

US006757718B1

(12) United States Patent
Halverson et al.

(10) Patent No.: **US 6,757,718 B1**

(45) Date of Patent: **Jun. 29, 2004**

(54) **MOBILE NAVIGATION OF NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN INPUT**

(75) Inventors: **Christine Halverson**, San Jose, CA (US); **Luc Julia**, Menlo Park, CA (US); **Dimitris Voutsas**, Thessaloniki (GR); **Adam Cheyer**, Palo Alto, CA (US)

(73) Assignee: **SRI International**, Menlo Park, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 56 days.

(21) Appl. No.: **09/608,872**

(22) Filed: **Jun. 30, 2000**

Related U.S. Application Data

(63) Continuation of application No. 09/524,095, filed on Mar. 13, 2000, which is a continuation-in-part of application No. 09/225,198, filed on Jan. 5, 1999.

(60) Provisional application No. 60/124,720, filed on Mar. 17, 1999, provisional application No. 60/124,719, filed on Mar. 17, 1999, and provisional application No. 60/124,718, filed on Mar. 17, 1999.

(51) **Int. Cl.**^{7} **G06F 15/16**

(52) **U.S. Cl.** **709/218**; 709/202; 709/217; 709/219; 709/227; 704/257

(58) **Field of Search** 709/202, 218, 709/217, 219, 227; 707/5, 3, 4; 704/257, 270.1, 275, 246

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,197,005	A	3/1993	Shwartz et al.	364/419
5,386,556	A	1/1995	Hedin et al.	395/600
5,434,777	A	7/1995	Luciw	364/419.13
5,519,608	A	5/1996	Kupiec	364/419.08
5,608,624	A	3/1997	Luciw	395/794
5,721,938	A	2/1998	Stuckey	395/754
5,729,659	A	3/1998	Potter	395/2.79
5,748,974	A	5/1998	Johnson	395/759
5,774,859	A	6/1998	Houser et al.	704/275
5,794,050	A	8/1998	Dahlgren et al.	395/708
5,802,526	A	9/1998	Fawcett et al.	707/104
5,805,775	A	9/1998	Eberman et al.	395/12
5,855,002	A	12/1998	Armstrong	704/270
5,890,123	A	3/1999	Brown et al.	704/275
5,963,940	A	10/1999	Liddy et al.	707/5

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

EP	0 867 861	9/1998	G10L/5/06
WO	99/50826	10/1999	G10L/3/00
WO	00/05638	2/2000	

OTHER PUBLICATIONS

International Search Report, Intl Appl No. PCT/US01/07987.

Stent, Amanda et al., "The CommandTalk Spoken Dialogue System", SRI International.

(List continued on next page.)

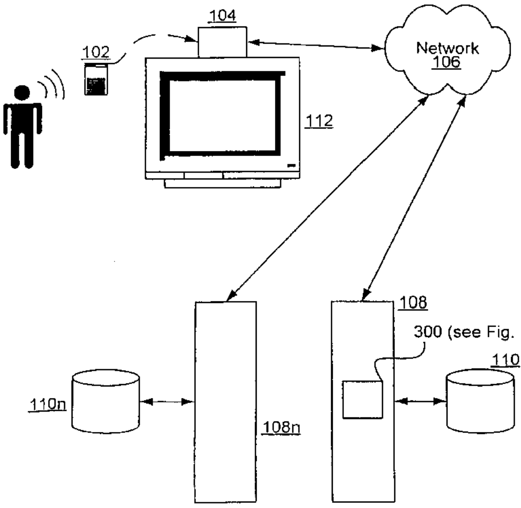
Primary Examiner—Frantz B Jean

(74) *Attorney, Agent, or Firm*—Moser, Patterson & Sheridan, LLP; Kin-Wah Tong

(57) **ABSTRACT**

A system, method, and article of manufacture are provided for navigating an electronic data source by means of spoken language where a portion of the data link between a mobile information appliance of the user and the data source utilizes wireless communication. When a spoken input request is received from a user who is using the mobile information appliance, it is interpreted. The resulting interpretation of the request is thereupon used to automatically construct an operational navigation query to retrieve the desired information from one or more electronic network data sources, which is transmitted to the mobile information appliance.

27 Claims, 7 Drawing Sheets



U.S. PATENT DOCUMENTS

6,003,072	A		12/1999	Gerritsen et al.	709/218
6,016,476	A		1/2000	Maes et al.	705/1
6,026,388	A		2/2000	Liddy et al.	707/1
6,102,030	A		8/2000	Brown et al.	704/275
6,173,279	B1	*	1/2001	Levin et al.	707/5
6,192,338	B1	*	2/2001	Haszto et al.	704/257
6,314,365	B1	*	11/2001	Smith	340/988
6,317,684	B1	*	11/2001	Roeseler et al.	340/990
6,349,257	B1	*	2/2002	Liu et al.	340/5.6
6,353,661	B1	*	3/2002	Bailey, III	379/88.17

OTHER PUBLICATIONS

Moore, Robert et al., "CommandTalk: A Spoken–Language Interface for Battlefield Simulations", Oct. 23, 1997, SRI International.

Dowding, John et al., "Interpreting Language in Context in CommandTalk", Feb. 5, 1999, SRI International.

http://www.ai.sri.com/~oaa/infowiz.html, InfoWiz: An Animated Voice Interactive Information System, May 8, 2000.

Dowding, John, "Interleaving Syntax and Semantics in an Efficient Bottom–up Parser", SRI International.

Moore, Robert et al., "Combining Linguistic and Statistical Knowledge Sources in Natural–Language Processing for ATIS", SRI International.

Dowding, John et al., "Gemini: A Natural Language System For Spoken–Language Understanding", SRI International.

* cited by examiner

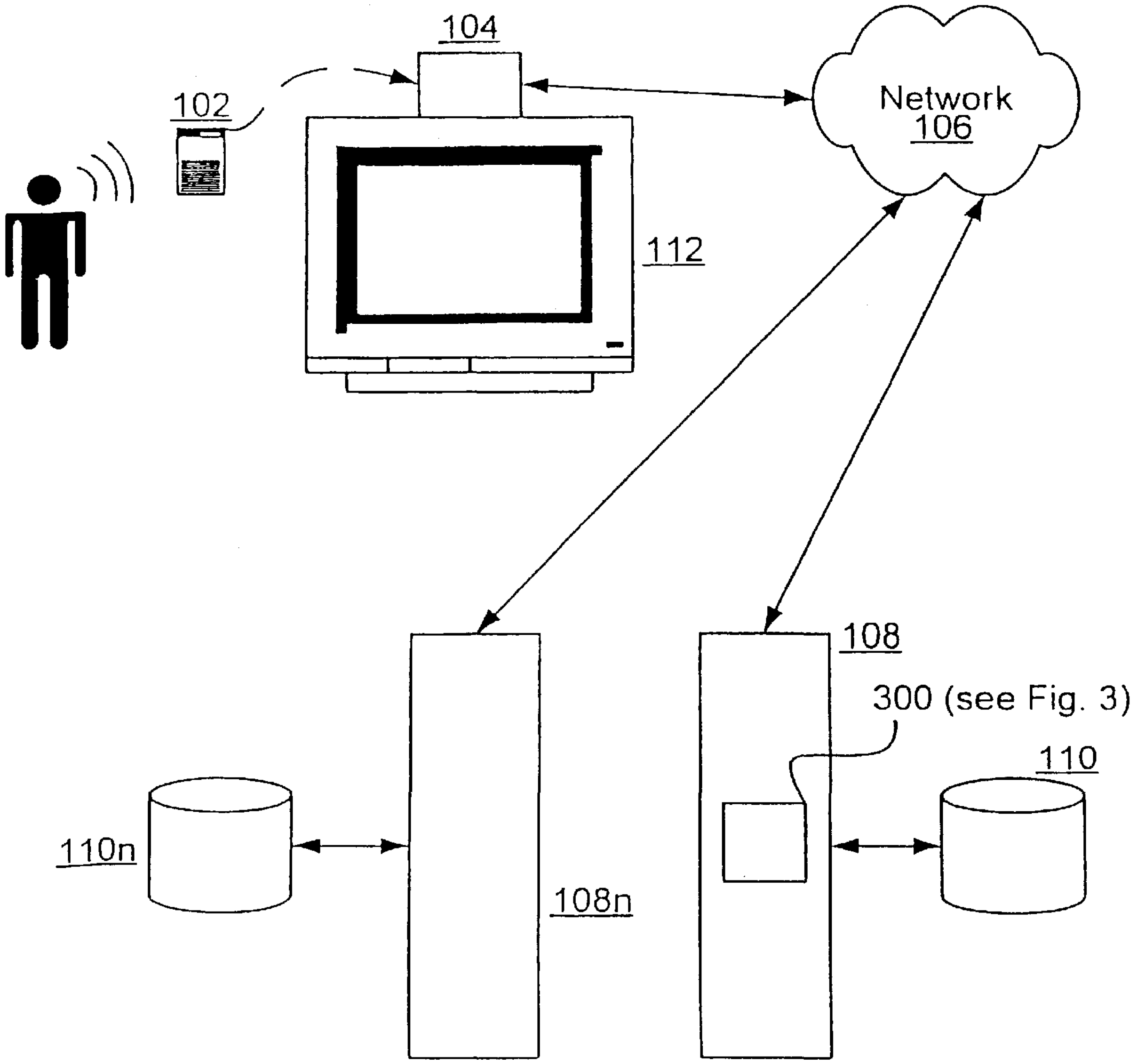


Fig. 1a

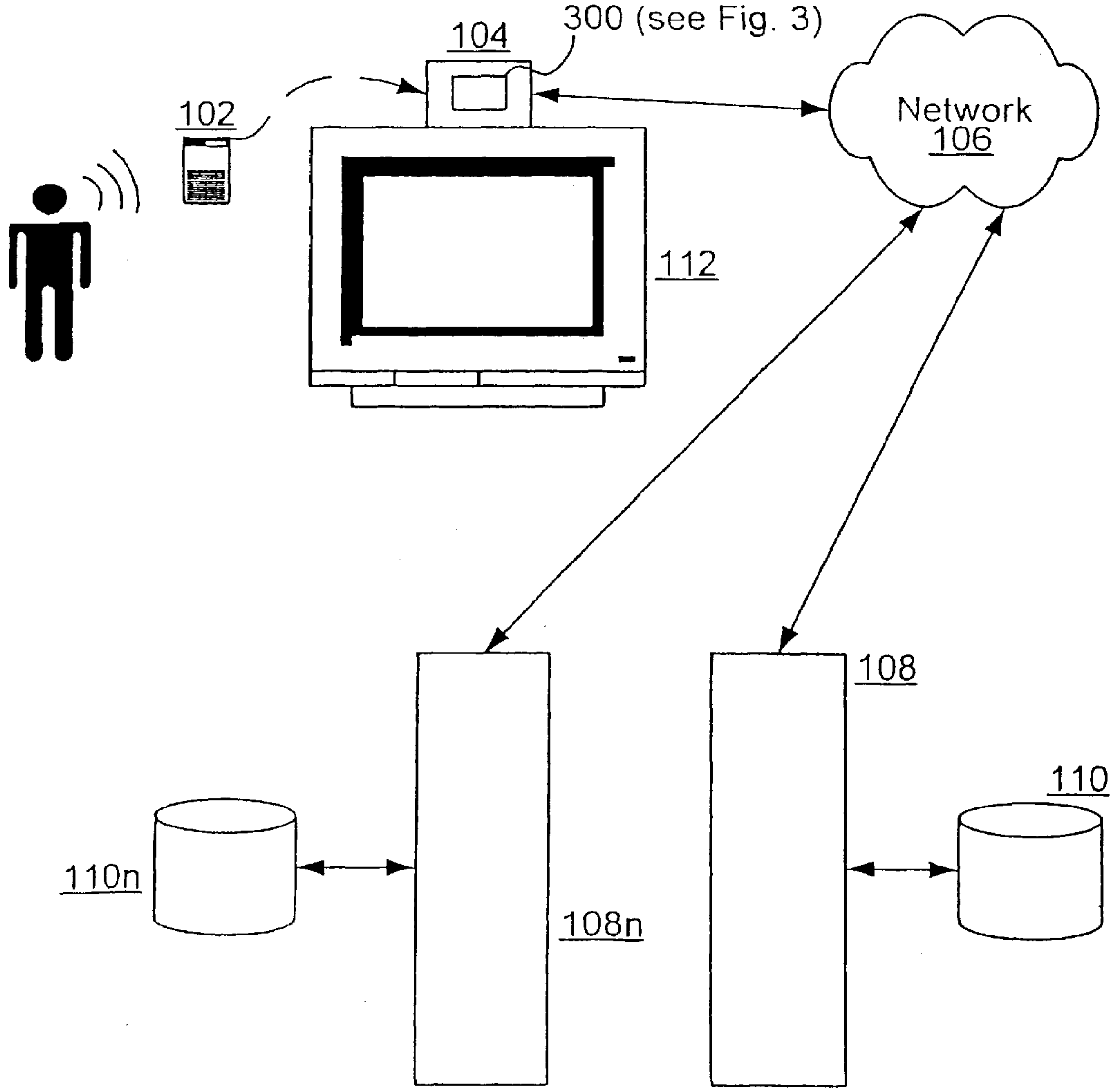


Fig. 1b

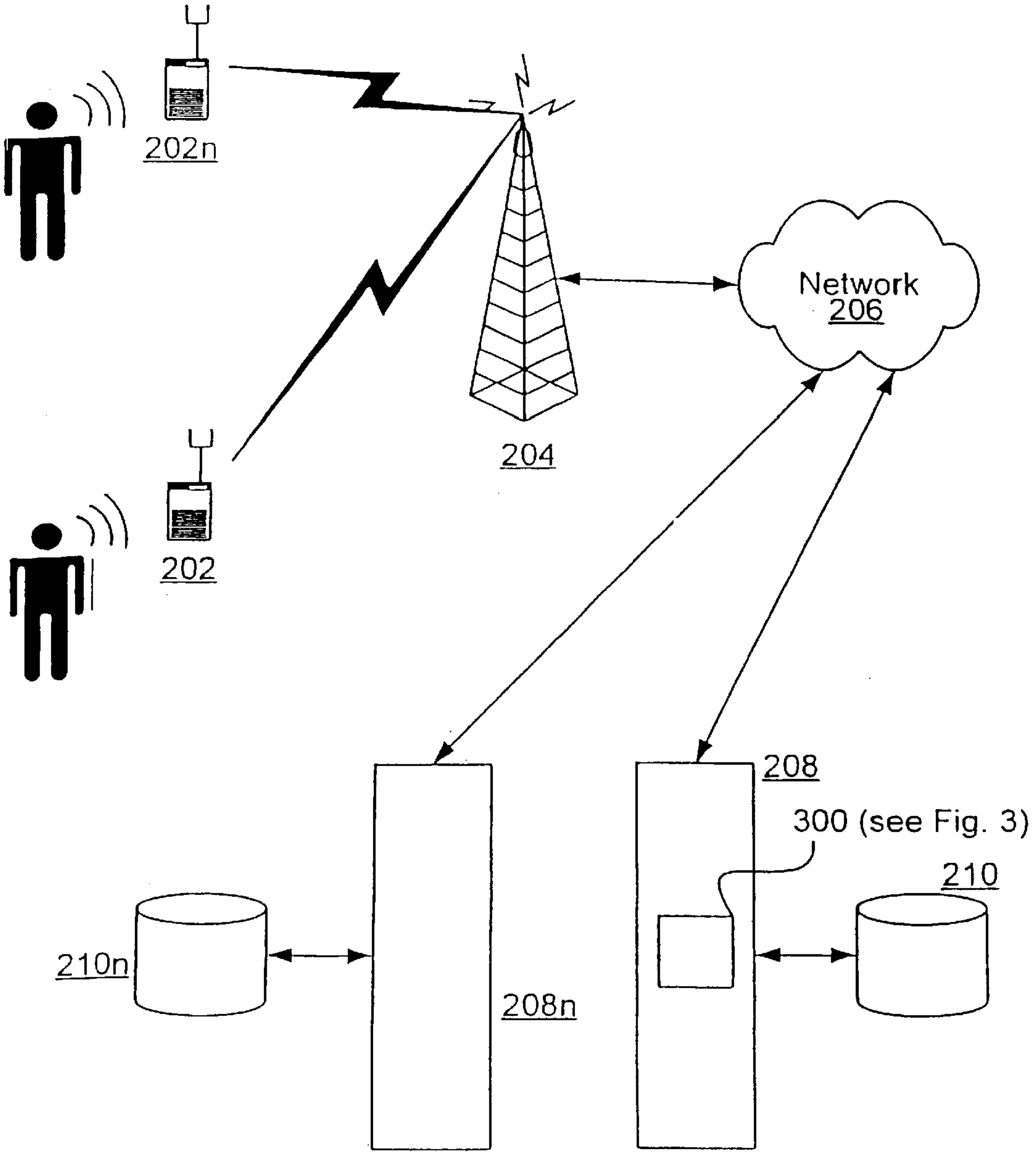


Fig. 2

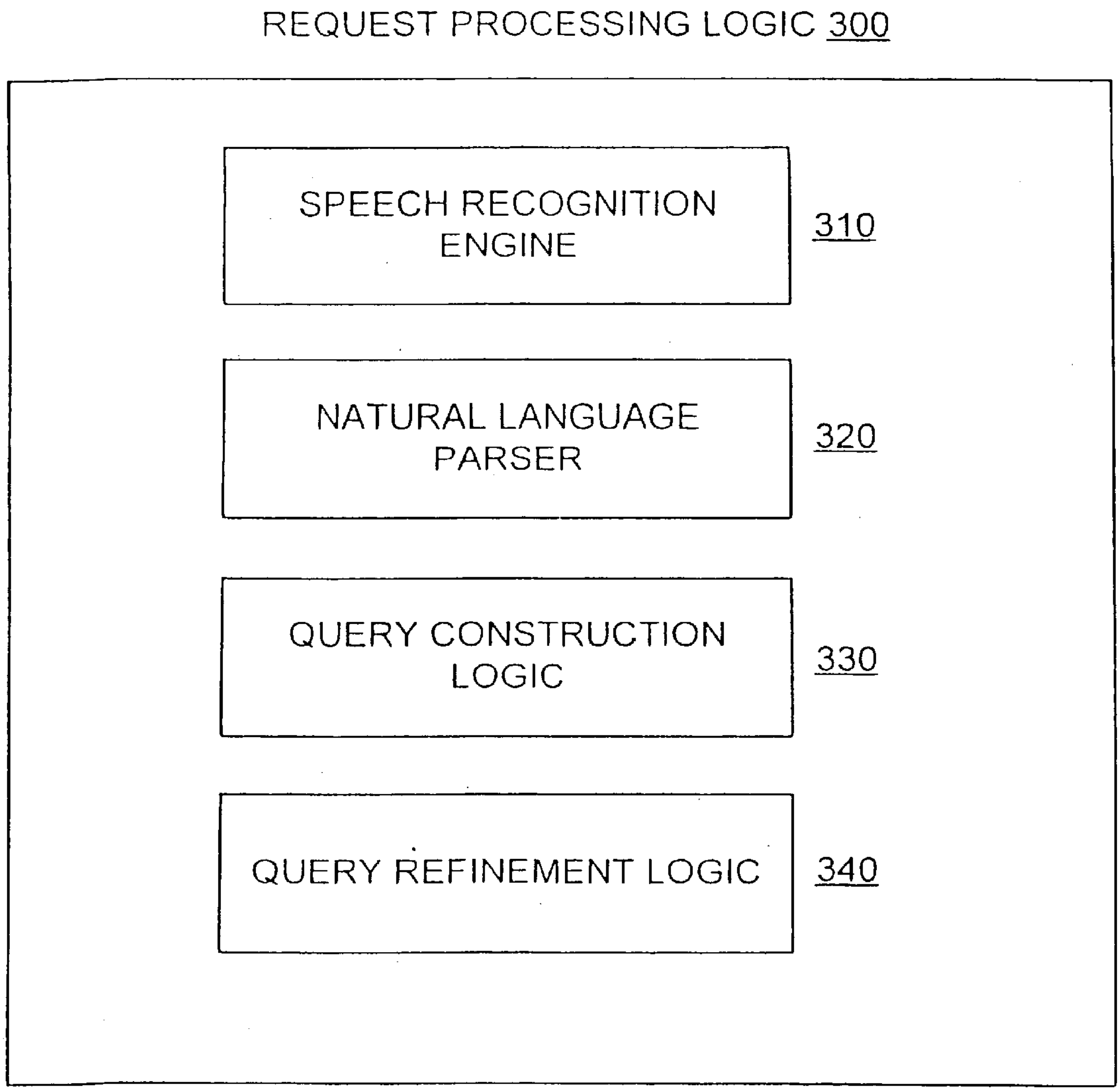

REQUEST PROCESSING LOGIC 300
SPEECH RECOGNITION ENGINE
310
NATURAL LANGUAGE PARSER
320
QUERY CONSTRUCTION LOGIC
330
QUERY REFINEMENT LOGIC
340

Fig. 3

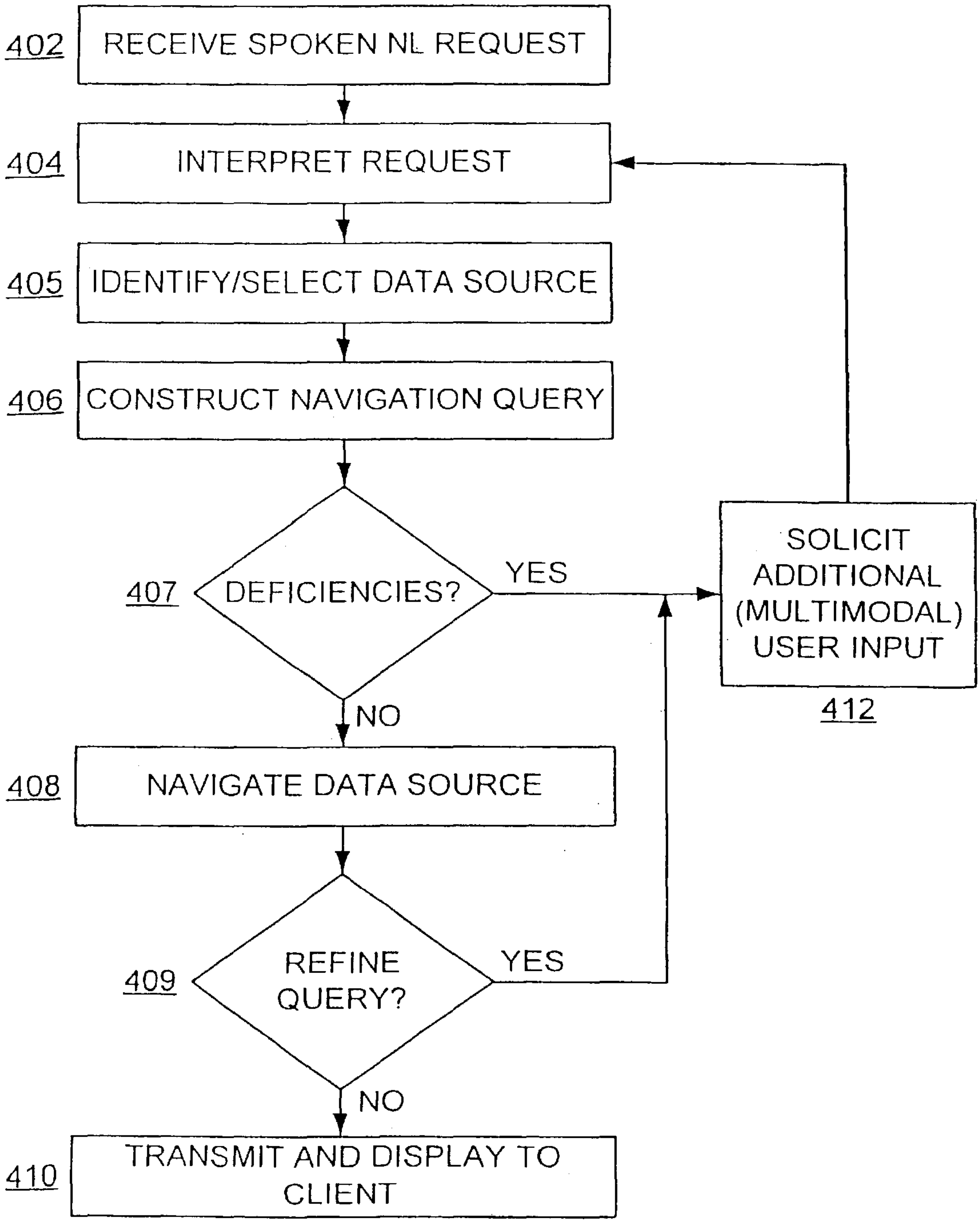

402
RECEIVE SPOKEN NL REQUEST
404
INTERPRET REQUEST
405
IDENTIFY/SELECT DATA SOURCE
406
CONSTRUCT NAVIGATION QUERY
407
DEFICIENCIES?
YES
NO
SOLICIT ADDITIONAL (MULTIMODAL) USER INPUT
412
408
NAVIGATE DATA SOURCE
409
REFINE QUERY?
YES
NO
410
TRANSMIT AND DISPLAY TO CLIENT

Fig. 4

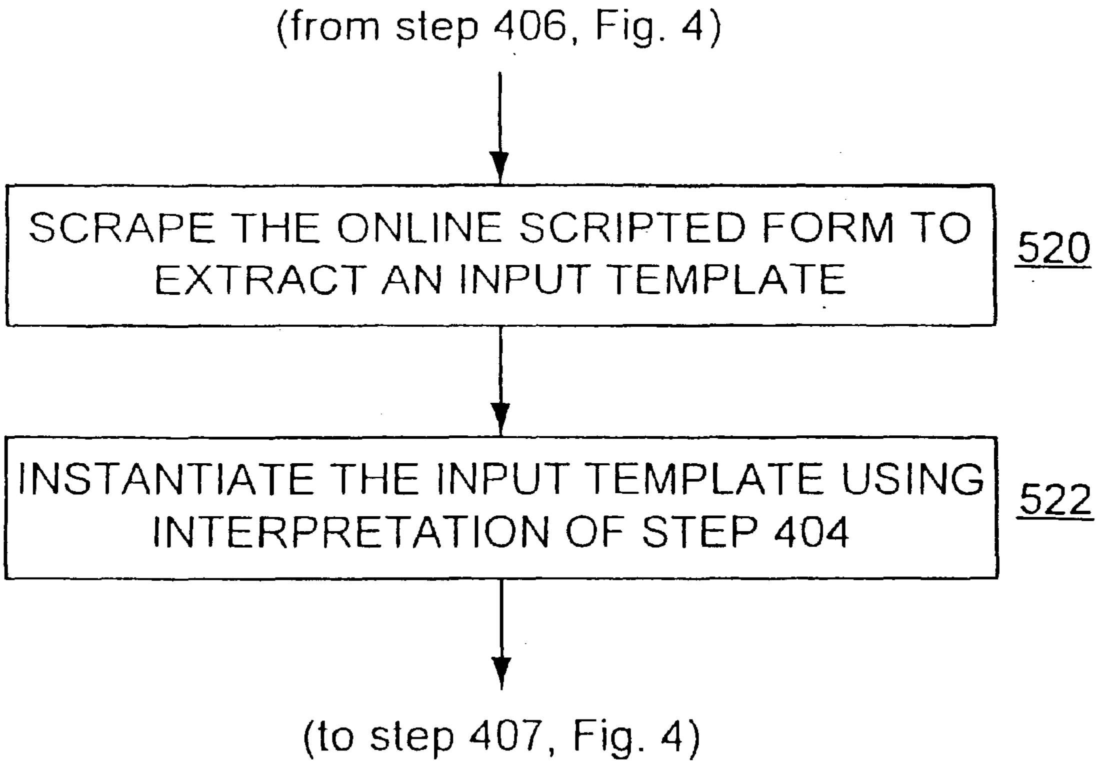

(from step 406, Fig. 4)
SCRAPE THE ONLINE SCRIPTED FORM TO EXTRACT AN INPUT TEMPLATE
520
INSTANTIATE THE INPUT TEMPLATE USING INTERPRETATION OF STEP 404
522
(to step 407, Fig. 4)

Fig. 5

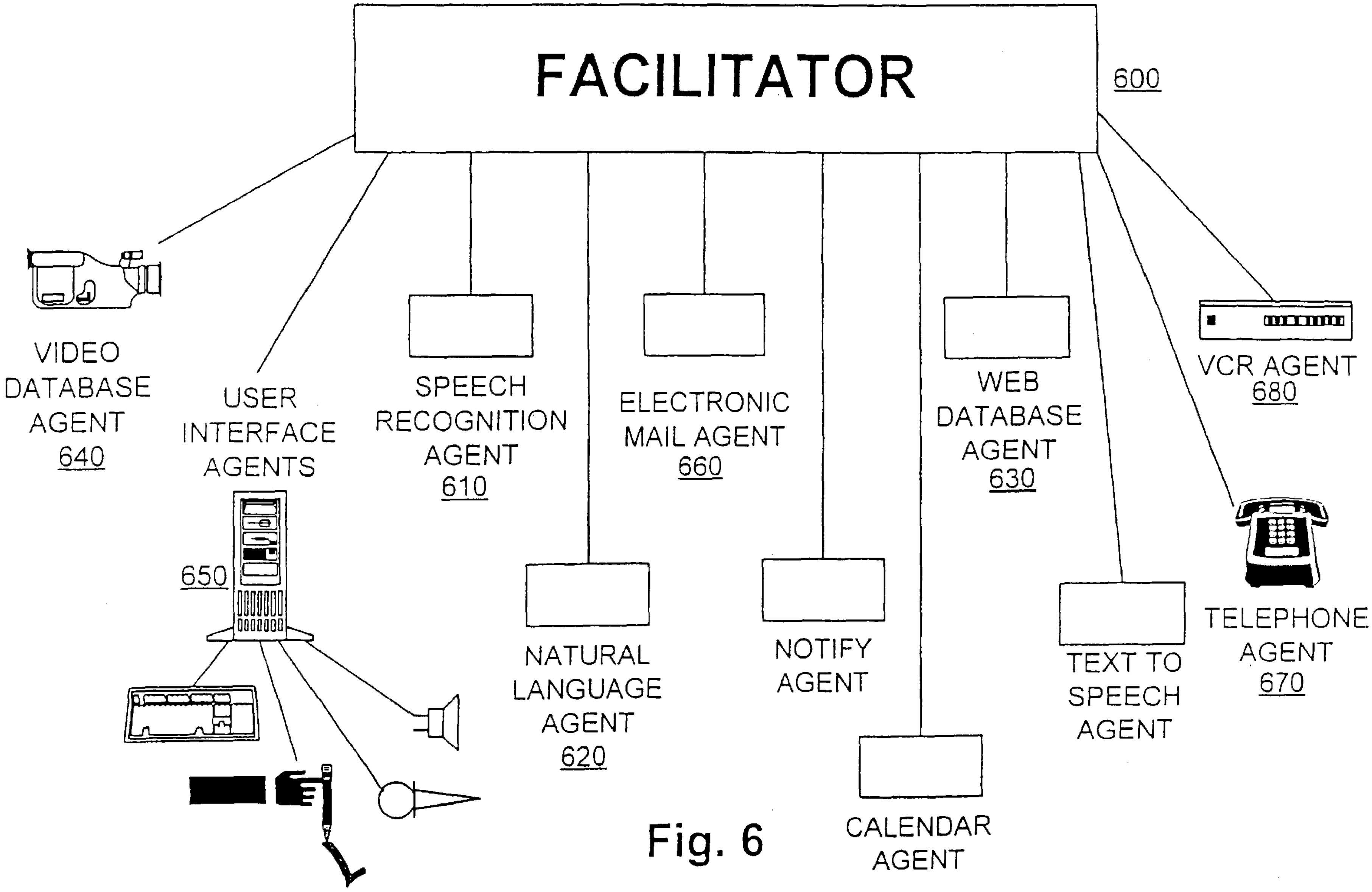

FACILITATOR
600
VIDEO DATABASE AGENT
640
USER INTERFACE AGENTS
650
SPEECH RECOGNITION AGENT
610
NATURAL LANGUAGE AGENT
620
ELECTRONIC MAIL AGENT
660
NOTIFY AGENT
CALENDAR AGENT
WEB DATABASE AGENT
630
TEXT TO SPEECH AGENT
TELEPHONE AGENT
670
VCR AGENT
680

Fig. 6

MOBILE NAVIGATION OF NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN INPUT

This application is a continuation of an application entitled NAVIGATING NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN NATURAL LANGUAGE INPUT WITH MULTIMODAL ERROR FEEDBACK which was filed on Mar. 13, 2000 under Ser. No. 09/524,095 and which is a Continuation In Part of co-pending U.S. patent application Ser. No. 09/225,198, filed Jan. 5, 1999, Provisional U.S. patent application Ser. No. 60/124,718, filed Mar. 17, 1999, Provisional U.S. patent application Ser. No. 60/124,720, filed Mar. 17, 1999, and Provisional U.S. patent application Ser. No. 60/124,719, filed Mar. 17, 1999, from which applications priority is claimed and these application are incorporated herein by reference.

BACKGROUND OF THE INVENTION

The present invention relates generally to the navigation of electronic data by means of spoken natural language requests, and to feedback mechanisms and methods for resolving the errors and ambiguities that may be associated with such requests.

As global electronic connectivity continues to grow, and the universe of electronic data potentially available to users continues to expand, there is a growing need for information navigation technology that allows relatively naive users to navigate and access desired data by means of natural language input. In many of the most important markets—including the home entertainment arena, as well as mobile computing—spoken natural language input is highly desirable, if not ideal. As just one example, the proliferation of high-bandwidth communications infrastructure for the home entertainment market (cable, satellite, broadband) enables delivery of movies-on-demand and other interactive multimedia content to the consumer's home television set. For users to take full advantage of this content stream ultimately requires interactive navigation of content databases in a manner that is too complex for user-friendly selection by means of a traditional remote-control clicker. Allowing spoken natural language requests as the input modality for rapidly searching and accessing desired content is an important objective for a successful consumer entertainment product in a context offering a dizzying range of database content choices. As further examples, this same need to drive navigation of (and transaction with) relatively complex data warehouses using spoken natural language requests applies equally to surfing the Internet/Web or other networks for general information, multimedia content, or e-commerce transactions.

In general, the existing navigational systems for browsing electronic databases and data warehouses (search engines, menus, etc.), have been designed without navigation via spoken natural language as a specific goal. So today's world is full of existing electronic data navigation systems that do not assume browsing via natural spoken commands, but rather assume text and mouse-click inputs (or in the case of TV remote controls, even less). Simply recognizing voice commands within an extremely limited vocabulary and grammar—the spoken equivalent of button/click input (e.g., speaking "channel 5" selects TV channel 5)—is really not sufficient by itself to satisfy the objectives described above. In order to deliver a true "win" for users, the voice-driven front-end must accept spoken natural language input in a manner that is intuitive to users. For example, the front-end should not require learning a highly specialized command language or format. More fundamentally, the front-end must allow users to speak directly in terms of what the user ultimately wants —e.g., "I'd like to see a Western film directed by Clint Eastwood" —as opposed to speaking in terms of arbitrary navigation structures (e.g., hierarchical layers of menus, commands, etc.) that are essentially artifacts reflecting constraints of the pre-existing text/click navigation system. At the same time, the front-end must recognize and accommodate the reality that a stream of naive spoken natural language input will, over time, typically present a variety of errors and/or ambiguities: e.g., garbled/unrecognized words (did the user say "Eastwood" or "Easter"?) and under-constrained requests ("Show me the Clint Eastwood movie"). An approach is needed for handling and resolving such errors and ambiguities in a rapid, user-friendly, non-frustrating manner.

What is needed is a methodology and apparatus for rapidly constructing a voice-driven front-end atop an existing, non-voice data navigation system, whereby users can interact by means of intuitive natural language input not strictly conforming to the step-by-step browsing architecture of the existing navigation system, and wherein any errors or ambiguities in user input are rapidly and conveniently resolved. The solution to this need should be compatible with the constraints of a multi-user, distributed environment such as the Internet/Web or a proprietary high-bandwidth content delivery network; a solution contemplating one-at-a-time user interactions at a single location is insufficient, for example.

SUMMARY OF THE INVENTION

The present invention addresses the above needs by providing a system, method, and article of manufacture for mobile navigation of network-based electronic data sources in response to spoken input requests. When a spoken input request is received from a user using a mobile information appliance that communicates with a network server via an at least partially wireless communications system, it is interpreted, such as by using a speech recognition engine to extract speech data from acoustic voice signals, and using a language parser to linguistically parse the speech data. The interpretation of the spoken request can be performed on a computing device locally with the user, such as the mobile information appliance, or remotely from the user. The resulting interpretation of the request is thereupon used to automatically construct an operational navigation query to retrieve the desired information from one or more electronic network data sources, which is then transmitted to a client device of the user. If the network data source is a database, the navigation query is constructed in the format of a database query language.

Typically, errors or ambiguities emerge in the interpretation of the spoken request, such that the system cannot instantiate a complete, valid navigational template. This is to be expected occasionally, and one preferred aspect of the invention is the ability to handle such errors and ambiguities in relatively graceful and user-friendly manner. Instead of simply rejecting such input and defaulting to traditional input modes or simply asking the user to try again, a preferred embodiment of the present invention seeks to converge rapidly toward instantiation of a valid navigational template by soliciting additional clarification from the user as necessary, either before or after a navigation of the data source, via multimodal input, i.e., by means of menu selection or other input modalities including and in addition to spoken input. This clarifying, multi-modal dialogue takes advantage of whatever partial navigational information has been gleaned from the initial interpretation of the user's spoken request. This clarification process continues until the system converges toward an adequately instantiated navigational template, which is in turn used to navigate the network-based data and retrieve the user's desired information. The retrieved information is transmitted across the network and presented to the user on a suitable client display device.

In a further aspect of the present invention, the construction of the navigation query includes extracting an input template for an online scripted interface to the data source and using the input template to construct the navigation query. The extraction of the input template can include dynamically scraping the online scripted interface.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention, together with further advantages thereof, may best be understood by reference to the following description taken in conjunction with the accompanying drawings in which:

FIG. 1*a* illustrates a system providing a spoken natural language interface for network-based information navigation, in accordance with an embodiment of the present invention with server-side processing of requests;

FIG. 1*b* illustrates another system providing a spoken natural language interface for network-based information navigation, in accordance with an embodiment of the present invention with client-side processing of requests;

FIG. 2 illustrates a system providing a spoken natural language interface for network-based information navigation, in accordance with an embodiment of the present invention for a mobile computing scenario;

FIG. 3 illustrates the functional logic components of a request processing module in accordance with an embodiment of the present invention;

FIG. 4 illustrates a process utilizing spoken natural language for navigating an electronic database in accordance with one embodiment of the present invention;

FIG. 5 illustrates a process for constructing a navigational query for accessing an online data source via an interactive, scripted (e.g., CGI) form; and

FIG. 6 illustrates an embodiment of the present invention utilizing a community of distributed, collaborating electronic agents.

DETAILED DESCRIPTION OF THE INVENTION

1. System Architecture

a. Server-End Processing of Spoken Input

FIG. 1*a* is an illustration of a data navigation system driven by spoken natural language input, in accordance with one embodiment of the present invention. As shown, a user's voice input data is captured by a voice input device **102**, such as a microphone. Preferably voice input device **102** includes a button or the like that can be pressed or held-down to activate a listening mode, so that the system need not continually pay attention to, or be confused by, irrelevant background noise. In one preferred embodiment well-suited for the home entertainment setting, voice input device **102** is a portable remote control device with an integrated microphone, and the voice data is transmitted from device **102** preferably via infrared (or other wireless) link to communications box **104** (e.g., a set-top box or a similar communications device that is capable of retransmitting the raw voice data and/or processing the voice data) local to the user's environment and coupled to communications network **106**. The voice data is then transmitted across network **106** to a remote server or servers **108**. The voice data may preferably be transmitted in compressed digitized form, or alternatively—particularly where bandwidth constraints are significant—in analog format (e.g., via frequency modulated transmission), in the latter case being digitized upon arrival at remote server **108**.

At remote server **108**, the voice data is processed by request processing logic **300** in order to understand the user's request and construct an appropriate query or request for navigation of remote data source **110**, in accordance with the interpretation process exemplified in FIG. **4** and FIG. **5** and discussed in greater detail below. For purposes of executing this process, request processing logic **300** comprises functional modules including speech recognition engine **310**, natural language (NL) parser **320**, query construction logic **330**, and query refinement logic **340**, as shown in FIG. **3**. Data source **110** may comprise database(s), Internet/web site(s), or other electronic information repositories, and preferably resides on a central server or servers—which may or may not be the same as server **108**, depending on the storage and bandwidth needs of the application and the resources available to the practitioner. Data source **110** may include multimedia content, such as movies or other digital video and audio content, other various forms of entertainment data, or other electronic information. The contents of data source **110** are navigated—i.e., the contents are accessed and searched, for retrieval of the particular information desired by the user—using the processes of FIGS. **4** and **5** as described in greater detail below.

Once the desired information has been retrieved from data source **110**, it is electronically transmitted via network **106** to the user for viewing on client display device **112**. In a preferred embodiment well-suited for the home entertainment setting, display device **112** is a television monitor or similar audiovisual entertainment device, typically in stationary position for comfortable viewing by users. In addition, in such preferred embodiment, display device **112** is coupled to or integrated with a communications box (which is preferably the same as communications box **104**, but may also be a separate unit) for receiving and decoding/formatting the desired electronic information that is received across communications network **106**.

Network **106** is a two-way electronic communications network and may be embodied in electronic communication infrastructure including coaxial (cable television) lines, DSL, fiber-optic cable, traditional copper wire (twisted pair), or any other type of hardwired connection. Network **106** may also include a wireless connection such as a satellite-based connection, cellular connection, or other type of wireless connection. Network **106** may be part of the Internet and may support TCP/IP communications, or may be embodied in a proprietary network, or in any other electronic communications network infrastructure, whether packet-switched or connection-oriented. A design consideration is that network **106** preferably provide suitable bandwidth depending upon the nature of the content anticipated for the desired application.

b. Client-End Processing of Spoken Input

FIG. 1*b* is an illustration of a data navigation system driven by spoken natural language input, in accordance with a second embodiment of the present invention. Again, a user's voice input data is captured by a voice input device

102, such as a microphone. In the embodiment shown in FIG. **1***b*, the voice data is transmitted from device **202** to requests processing logic **300**, hosted on a local speech processor, for processing and interpretation. In the preferred embodiment illustrated in FIG. **1***b*, the local speech processor is conveniently integrated as part of communications box **104**, although implementation in a physically separate (but communicatively coupled) unit is also possible as will be readily apparent to those of skill in the art. The voice data is processed by the components of request processing logic **300** in order to understand the user's request and construct an appropriate query or request for navigation of remote data source **110**, in accordance with the interpretation process exemplified in FIGS. **4** and **5** as discussed in greater detail below.

The resulting navigational query is then transmitted electronically across network **106** to data source **110**, which preferably resides on a central server or servers **108**. As in FIG. **1***a*, data source **110** may comprise database(s), Internet/web site(s), or other electronic information repositories, and preferably may include multimedia content, such as movies or other digital video and audio content, other various forms of entertainment data, or other electronic information. The contents of data source **110** are then navigated—i.e., the contents are accessed and searched, for retrieval of the particular information desired by the user—preferably using the process of FIGS. **4** and **5** as described in greater detail below. Once the desired information has been retrieved from data source **110**, it is electronically transmitted via network **106** to the user for viewing on client display device **112**.

In one embodiment in accordance with FIG. **1***b* and well-suited for the home entertainment setting, voice input device **102** is a portable remote control device with an integrated microphone, and the voice data is transmitted from device **102** preferably via infrared (or other wireless) link to the local speech processor. The local speech processor is coupled to communications network **106**, and also preferably to client display device **112** (especially for purposes of query refinement transmissions, as discussed below in connection with FIG. **4**, step **412**), and preferably may be integrated within or coupled to communications box **104**. In addition, especially for purposes of a home entertainment application, display device **112** is preferably a television monitor or similar audiovisual entertainment device, typically in stationary position for comfortable viewing by users. In addition, in such preferred embodiment, display device **112** is coupled to a communications box (which is preferably the same as communications box **104**, but may also be a physically separate unit) for receiving and decoding/formatting the desired electronic information that is received across communications network **106**.

Design considerations favoring server-side processing and interpretation of spoken input requests, as exemplified in FIG. **1***a*, include minimizing the need to distribute costly computational hardware and software to all client users in order to perform speech and language processing. Design considerations favoring client-side processing, as exemplified in FIG. **1***b*, include minimizing the quantity of data sent upstream across the network from each client, as the speech recognition is performed before transmission across the network and only the query data and/or request needs to be sent, thus reducing the upstream bandwidth requirements.

c. Mobile Client Embodiment

A mobile computing embodiment of the present invention may be implemented by practitioners as a variation on the embodiments of either FIG. **1***a* or FIG. **1***b*. For example, as depicted in FIG. **2**, a mobile variation in accordance with the server-side processing architecture illustrated in FIG. **1***a* may be implemented by replacing voice input device **102**, communications box **104**, and client display device **112**, with an integrated, mobile, information appliance **202** such as a cellular telephone or wireless personal digital assistant (wireless PDA). Mobile information appliance **202** essentially performs the functions of the replaced components. Thus, mobile information appliance **202** receives spoken natural language input requests from the user in the form of voice data, and transmits that data (preferably via wireless data receiving station **204**) across communications network **206** for server-side interpretation of the request, in similar fashion as described above in connection with FIG. **1**. Navigation of data source **210** and retrieval of desired information likewise proceeds in an analogous manner as described above. Display information transmitted electronically back to the user across network **206** is displayed for the user on the display of information appliance **202**, and audio information is output through the appliance's speakers.

Practitioners will further appreciate, in light of the above teachings, that if mobile information appliance **202** is equipped with sufficient computational processing power, then a mobile variation of the client-side architecture exemplified in FIG. **2** may similarly be implemented. In that case, the modules corresponding to request processing logic **300** would be embodied locally in the computational resources of mobile information appliance **202**, and the logical flow of data would otherwise follow in a manner analogous to that previously described in connection with FIG. **1***b*.

As illustrated in FIG. **2**, multiple users, each having their own client input device, may issue requests, simultaneously or otherwise, for navigation of data source **210**. This is equally true (though not explicitly drawn) for the embodiments depicted in FIGS. **1***a* and **1***b*. Data source **210** (or **100**), being a network accessible information resource, has typically already been constructed to support access requests from simultaneous multiple network users, as known by practitioners of ordinary skill in the art. In the case of server-side speech processing, as exemplified in FIGS. **1***a* and **2**, the interpretation logic and error correction logic modules are also preferably designed and implemented to support queuing and multi-tasking of requests from multiple simultaneous network users, as will be appreciated by those of skill in the art.

It will be apparent to those skilled in the art that additional implementations, permutations and combinations of the embodiments set forth in FIGS. **1***a*, **1***b*, and **2** may be created without straying from the scope and spirit of the present invention. For example, practitioners will understand, in light of the above teachings and design considerations, that it is possible to divide and allocate the functional components of request processing logic **300** between client and server. For example, speech recognition—in entirety, or perhaps just early stages such as feature extraction—might be performed locally on the client end, perhaps to reduce bandwidth requirements, while natural language parsing and other necessary processing might be performed upstream on the server end, so that more extensive computational power need not be distributed locally to each client. In that case, corresponding portions of request processing logic **300**, such as speech recognition engine **310** or portions thereof, would reside locally at the client as in FIG. **1***b*, while other component modules would be hosted at the server end as in FIGS. **1***a* and **2**.

Further, practitioners may choose to implement the each of the various embodiments described above on any number of different hardware and software computing platforms and

environments and various combinations thereof, including, by way of just a few examples: a general-purpose hardware microprocessor such as the Intel Pentium series; operating system software such as Microsoft Windows/CE, Palm OS, or Apple Mac OS (particularly for client devices and client-side processing), or Unix, Linux, or Windows/NT (the latter three particularly for network data servers and server-side processing), and/or proprietary information access platforms such as Microsoft's WebTV or the Diva Systems video-on-demand system.

2. Processing Methodology

The present invention provides a spoken natural language interface for interrogation of remote electronic databases and retrieval of desired information. A preferred embodiment of the present invention utilizes the basic methodology outlined in the flow diagram of FIG. **4** in order to provide this interface. This methodology will now be discussed.

a. Interpreting Spoken Natural Language Requests

At step **402**, the user's spoken request for information is initially received in the form of raw (acoustic) voice data by a suitable input device, as previously discussed in connection with FIGS. **1**–**2**. At step **404** the voice data received from the user is interpreted in order to understand the user's request for information. Preferably this step includes performing speech recognition in order to extract words from the voice data, and further includes natural language parsing of those words in order to generate a structured linguistic representation of the user's request.

Speech recognition in step **404** is performed using speech recognition engine **310**. A variety of commercial quality, speech recognition engines are readily available on the market, as practitioners will know. For example, Nuance Communications offers a suite of speech recognition engines, including Nuance 6, its current flagship product, and Nuance Express, a lower cost package for entry-level applications. As one other example, IBM offers the ViaVoice speech recognition engine, including a low-cost shrink-wrapped version available through popular consumer distribution channels. Basically, a speech recognition engine processes acoustic voice data and attempts to generate a text stream of recognized words.

Typically, the speech recognition engine is provided with a vocabulary lexicon of likely words or phrases that the recognition engine can match against its analysis of acoustical signals, for purposes of a given application. Preferably, the lexicon is dynamically adjusted to reflect the current user context, as established by the preceding user inputs. For example, if a user is engaged in a dialogue with the system about movie selection, the recognition engine's vocabulary may preferably be adjusted to favor relevant words and phrases, such as a stored list of proper names for popular movie actors and directors, etc. Whereas if the current dialogue involves selection and viewing of a sports event, the engine's vocabulary might preferably be adjusted to favor a stored list of proper names for professional sports teams, etc. In addition, a speech recognition engine is provided with language models that help the engine predict the most likely interpretation of a given segment of acoustical voice data, in the current context of phonemes or words in which the segment appears. In addition, speech recognition engines often echo to the user, in more or less real-time, a transcription of the engine's best guess at what the user has said, giving the user an opportunity to confirm or reject.

In a further aspect of step **404**, natural language interpreter (or parser) **320** linguistically parses and interprets the textual output of the speech recognition engine. In a preferred embodiment of the present invention, the natural-language interpreter attempts to determine both the meaning of spoken words (semantic processing) as well as the grammar of the statement (syntactic processing), such as the Gemini Natural Language Understanding System developed by SRI International. The Gemini system is described in detail in publications entitled "Gemini: A Natural Language System for Spoken-Language Understanding" and "Interleaving Syntax and Semantics in an Efficient Bottom-Up Parser," both of which are currently available online at http://www.ai.sri.com/natural-language/projects/arpa-sls/nat-lang.html. (Copies of those publications are also included in an information disclosure statement submitted herewith, and are incorporated herein by this reference). Briefly, Gemini applies a set of syntactic and semantic grammar rules to a word string using a bottom-up parser to generate a logical form, which is a structured representation of the context-independent meaning of the string. Gemini can be used with a variety of grammars, including general English grammar as well as application-specific grammars. The Gemini parser is based on "unification grammar," meaning that grammatical categories incorporate features that can be assigned values; so that when grammatical category expressions are matched in the course of parsing or semantic interpretation, the information contained in the features is combined, and if the feature values are incompatible the match fails.

It is possible for some applications to achieve a significant reduction in speech recognition error by using the natural-language processing system to re-score recognition hypotheses. For example, the grammars defined for a language parser like Gemini may be compiled into context-free grammar that, in turn, can be used directly as language models for speech recognition engines like the Nuance recognizer. Further details on this methodology are provided in the publication "Combining Linguistic and Statistical Knowledge Sources in Natural-Language Processing for ATIS" which is currently available online through http://www.ai.sri.com/natural-language/projects/arpa-sls/spnl-int.html. A copy of this publication is included in an information disclosure submitted herewith, and is incorporated herein by this reference.

In an embodiment of the present invention that may be preferable for some applications, the natural language interpreter "learns" from the past usage patterns of a particular user or of groups of users. In such an embodiment, the successfully interpreted requests of users are stored, and can then be used to enhance accuracy by comparing a current request to the stored requests, thereby allowing selection of a most probable result.

b. Constructing Navigation Queries

In step **405** request processing logic **300** identifies and selects an appropriate online data source where the desired information (in this case, current weather reports for a given city) can be found. Such selection may involve look-up in a locally stored table, or possibly dynamic searching through an online search engine, or other online search techniques. For some applications, an embodiment of the present invention may be implemented in which only access to a particular data source (such as a particular vendor's proprietary content database) is supported; in that case, step **405** may be trivial or may be eliminated entirely.

Step **406** attempts to construct a navigation query, reflecting the interpretation of step **404**. This operation is preferably performed by query construction logic **330**.

A "navigation query" means an electronic query, form, series of menu selections, or the like; being structured appropriately so as to navigate a particular data source of

interest in search of desired information. In other words, a navigation query is constructed such that it includes whatever content and structure is required in order to access desired information electronically from a particular database or data source of interest.

For example, for many existing electronic databases, a navigation query can be embodied using a formal database query language such as Standard Query Language (SQL). For many databases, a navigation query can be constructed through a more user-friendly interactive front-end, such as a series of menus and/or interactive forms to be selected or filled in. SQL is a standard interactive and programming language for getting information from and updating a database. SQL is both an ANSI and an ISO standard. As is well known to practitioners, a Relational Database Management System (RDBMS), such as Microsoft's Access, Oracle's Oracle**7**, and Computer Associates' CA-OpenIngres, allow programmers to create, update, and administer a relational database. Practitioners of ordinary skill in the art will be thoroughly familiar with the notion of database navigation through structured query, and will be readily able to appreciate and utilize the existing data structures and navigational mechanisms for a given database, or to create such structures and mechanisms where desired.

In accordance with the present invention, the query constructed in step **406** must reflect the user's request as interpreted by the speech recognition engine and the NL parser in step **404**. In embodiments of the present invention wherein data source **110** (or **210** in the corresponding embodiment of FIG. **2**) is a structured relational database or the like, step **406** of the present invention may entail constructing an appropriate Structured Query Language (SQL) query or the like, or automatically filling out a front-end query form, series of menus or the like, as described above.

In many existing Internet (and Intranet) applications, an online electronic data source is accessible to users only through the medium of interaction with a so-called Common Gateway Interface (CGI) script. Typically the user who visits a web site of this nature must fill in the fields of an online interactive form. The online form is in turn linked to a CGI script, which transparently handles actual navigation of the associated data source and produces output for viewing by the user's web browser. In other words, direct user access to the data source is not supported, only mediated access through the form and CGI script is offered.

For applications of this nature, an advantageous embodiment of the present invention "scrapes" the scripted online site where information desired by a user may be found in order to facilitate construction of an effective navigation query. For example, suppose that a user's spoken natural language request is: "What's the weather in Miami?" After this request is received at step **402** and interpreted at step **404**, assume that step **405** determines that the desired weather information is available online through the medium of a CGI-scripted interactive form. Step **406** is then preferably carried out using the expanded process diagrammed in FIG. **5**. In particular, at sub-step **520**, query construction logic **330** electronically "scrapes" the online interactive form, meaning that query construction logic **330** automatically extracts the format and structure of input fields accepted by the online form. At sub-step **522**, a navigation query is then constructed by instantiating (filling in) the extracted input format—essentially an electronic template—in a manner reflecting the user's request for information as interpreted in step **404**. The flow of control then returns to step **407** of FIG. **4**. Ultimately, when the query thus constructed by scraping is used to navigate the online data source in step **408**, the query effectively initiates the same scripted response as if a human user had visited the online site and had typed appropriate entries into the input fields of the online form.

In the embodiment just described, scraping step **520** is preferably carried out with the assistance of an online extraction utility such as WebL. WebL is a scripting language for automating tasks on the World Wide Web. It is an imperative, interpreted language that has built-in support for common web protocols like HTTP and FTP, and popular data types like HTML and XML. WebL's implementation language is Java, and the complete source code is available from Compaq. In addition, step **520** is preferably performed dynamically when necessary—in other words, on-the-fly in response to a particular user query—but in some applications it may be possible to scrape relatively stable (unchanging) web sites of likely interest in advance and to cache the resulting template information.

It will be apparent, in light of the above teachings, that preferred embodiments of the present invention can provide a spoken natural language interface atop an existing, non-voice data navigation system, whereby users can interact by means of intuitive natural language input not strictly conforming to the linear browsing architecture or other artifacts of an existing menu/text/click navigation system. For example, users of an appropriate embodiment of the present invention for a video-on-demand application can directly speak the natural request: "Show me the movie 'Unforgiven'"—instead of walking step-by-step through a typically linear sequence of genre/title/actor/director menus, scrolling and selecting from potentially long lists on each menu, or instead of being forced to use an alphanumeric keyboard that cannot be as comfortable to hold or use as a lightweight remote control. Similarly, users of an appropriate embodiment of the present invention for a web-surfing application in accordance with the process shown in FIG. **5** can directly speak the natural request: "Show me a one-month price chart for Microsoft stock" —instead of potentially having to navigate to an appropriate web site, search for the right ticker symbol, enter/select the symbol, and specify display of the desired one-month price chart, each of those steps potentially involving manual navigation and data entry to one or more different interaction screens. (Note that these examples are offered to illustrate some of the potential benefits offered by appropriate embodiments of the present invention, and not to limit the scope of the invention in any respect.)

c. Error Correction

Several problems can arise when attempting to perform searches based on spoken natural language input. As indicated at decision step **407** in the process of FIG. **4**, certain deficiencies may be identified during the process of query construction, before search of the data source is even attempted. For example, the user's request may fail to specify enough information in order to construct a navigation query that is specific enough to obtain a satisfactory search result. For example, a user might orally request "what's the weather?" whereas the national online data source identified in step **405** and scraped in step **520** might require specifying a particular city.

Additionally, certain deficiencies and problems may arise following the navigational search of the data source at step **408**, as indicated at decision step **409** in FIG. **4**. For example, with reference to a video-on-demand application, a user may wish to see the movie "Unforgiven", but perhaps the user can't recall name of the film, but knows it was

directed by and starred actor Clint Eastwood. A typical video-on-demand database might indeed be expected to allow queries specifying the name of a leading actor and/or director, but in the case of this query—as in many cases—that will not be enough to narrow the search to a single film, and additional user input in some form is required.

In the event that one or more deficiencies in the user's spoken request, as processed, result in the problems described, either at step **407** or **409**, some form of error handling is in order. A straightforward, crude technique might be for the system to respond simply "input not understood/insufficient, please try again." However, that approach will likely result in frustrated users, and is not optimal or even acceptable for most applications. Instead, a preferred technique in accordance with the present invention handles such errors and deficiencies in user input at step **412**, whether detected at step **407** or step **409**, by soliciting additional input from the user in a manner taking advantage of the partial construction already performed and via user interface modalities in addition to spoken natural language ("multi-modality"). This supplemental interaction is preferably conducted through client display device **112** (**202**, in the embodiment of FIG. **2**), and may include textual, graphical, audio and/or video media. Further details and examples are provided below. Query refinement logic **340** preferably carries out step **412**. The additional input received from the user is fed into and augments interpreting step **404**, and query construction step **406** is likewise repeated with the benefit of the augmented interpretation. These operations, and subsequent navigation step **408**, are preferably repeated until no remaining problems or deficiencies are identified at decision points **407** or **409**. Further details and examples for this query refinement process are provided immediately below.

Consider again the example in which the user of a video-on-demand application wishes to see "Unforgiven" but can only recall that it was directed by and starred Clint Eastwood. First, it bears noting that using a prior art navigational interface, such as a conventional menu interface, will likely be relatively tedious in this case. The user can proceed through a sequence of menus, such as Genre (select "western"), Title (skip), Actor ("Clint Eastwood"), and Director ("Clint Eastwood"). In each case—especially for the last two items—the user would typically scroll and select from fairly long lists in order to enter his or her desired name, or perhaps use a relatively couch-unfriendly keypad to manually type the actor's name twice.

Using a preferred embodiment of the present invention, the user instead speaks aloud, holding remote control microphone **102**, "I want to see that movie starring and directed by Clint Eastwood. Can't remember the title." At step **402** the voice data is received. At step **404** the voice data is interpreted. At step **405** an appropriate online data source is selected (or perhaps the system is directly connected to a proprietary video-on-demand provider). At step **406** a query is automatically constructed by the query construction logic **330** specifying "Clint Eastwood" in both the actor and director fields. Step **407** detects no obvious problems, and so the query is electronically submitted and the data source is navigated at step **408**, yielding a list of several records satisfying the query (e.g., "Unforgiven", "True Crime", "Absolute Power", etc.). Step **409** detects that additional user input is needed to further refine the query in order to select a particular film for viewing.

At that point, in step **412** query refinement logic **340** might preferably generate a display for client display device **112** showing the (relatively short) list of film titles that satisfy the user's stated constraints. The user can then preferably use a relatively convenient input modality, such as buttons on the remote control, to select the desired title from the menu. In a further preferred embodiment, the first title on the list is highlighted by default, so that the user can simply press an "OK" button to choose that selection. In a further preferred feature, the user can mix input modalities by speaking a response like "I want number one on the list." Alternatively, the user can preferably say, "Let's see Unforgiven," having now been reminded of the title by the menu display.

Utilizing the user's supplemental input, request processing logic **300** iterates again through steps **404** and **406**, this time constructing a fully-specified query that specifically requests the Eastwood film "Unforgiven." Step **408** navigates the data source using that query and retrieves the desired film, which is then electronically transmitted in step **410** from network server **108** to client display device **112** via communications network **106**.

Now consider again the example in which the user of a web surfing application wants to know his or her local weather, and simply asks, "what's the weather?" At step **402** the voice data is received. At step **404** the voice data is interpreted. At step **405** an online web site providing current weather information for major cities around the world is selected. At step **406** and sub-step **520**, the online site is scraped using a WebL-style tool to extract an input template for interacting with the site. At sub-step **522**, query construction logic **330** attempts to construct a navigation query by instantiating the input template, but determines (quite rightly) that a required field—name of city—cannot be determined from the user's spoken request as interpreted in step **404**. Step **407** detects this deficiency, and in step **412** query refinement logic **340** preferably generates output for client display device **112** soliciting the necessary supplemental input. In a preferred embodiment, the output might display the name of the city where the user is located highlighted by default. The user can then simply press an "OK" button—or perhaps mix modalities by saying "yes, exactly" —to choose that selection. A preferred embodiment would further display an alphabetical scrollable menu listing other major cities, and/or invite the user to speak or select the name of the desired city.

Here again, utilizing the user's supplemental input, request processing logic **300** iterates through steps **404** and **406**. This time, in performing sub-step **520**, a cached version of the input template already scraped in the previous iteration might preferably be retrieved. In sub-step **522**, query construction logic **330** succeeds this time in instantiating the input template and constructing an effective query, since the desired city has now been clarified. Step **408** navigates the data source using that query and retrieves the desired weather information, which is then electronically transmitted in step **410** from network server **108** to client display device **112** via communications network **106**.

It is worth noting that in some instances, there may be details that are not explicitly provided by the user, but that query construction logic **330** or query refinement logic **340** may preferably deduce on their own through reasonable assumptions, rather than requiring the use to provide explicit clarification. For example, in the example previously described regarding a request for a weather report, in some applications it might be preferable for the system to simply assume that the user means a weather report for his or her home area and to retrieve that information, if the cost of doing so is not significantly greater than the cost of asking the user to clarify the query. Making such an assumption

might be even more strongly justified in a preferred embodiment, as described earlier, where user histories are tracked, and where such history indicates that a particular user or group of users typically expect local information when asking for a weather forecast. At any rate, in the event such an assumption is made, if the user actually intended to request the weather for a different city, the user would then need to ask his or her question again. It will be apparent to practitioners, in light of the above teachings, that the choice of whether to program query construction logic **330** and query refinement logic **340** to make make particular assumptions will typically involve trade-offs involving user conveience that can be assessed in the context of specific applications.

3. Open Agent Architecture (OAA®)

Open Agent Architecture™ (OAA®) is a software platform, developed by the assignee of the present invention, that enables effective, dynamic collaboration among communities of distributed electronic agents. OAA is described in greater detail in co-pending U.S. patent application Ser. No. 09/225,198, which has been incorporated herein by reference. Very briefly, the functionality of each client agent is made available to the agent community through registration of the client agent's capabilities with a facilitator. A software "wrapper" essentially surrounds the underlying application program performing the services offered by each client. The common infrastructure for constructing agents is preferably supplied by an agent library. The agent library is preferably accessible in the runtime environment of several different programming languages. The agent library preferably minimizes the effort required to construct a new system and maximizes the ease with which legacy systems can be "wrapped" and made compatible with the agent-based architecture of the present invention. When invoked, a client agent makes a connection to a facilitator, which is known as its parent facilitator. Upon connection, an agent registers with its parent facilitator a specification of the capabilities and services it can provide, using a high-level, declarative Interagent Communication Language ("ICL") to express those capabilities. Tasks are presented to the facilitator in the form of ICL goal expressions. When a facilitator determines that the registered capabilities of one of its client agents will help satisfy a current goal or sub-goal thereof, the facilitator delegates that sub-goal to the client agent in the form of an ICL request. The client agent processes the request and returns answers or information to the facilitator. In processing a request, the client agent can use ICL to request services of other agents, or utilize other infrastructure services for collaborative work. The facilitator coordinates and integrates the results received from different client agents on various sub-goals, in order to satisfy the overall goal.

OAA provides a useful software platform for building systems that integrate spoken natural language as well as other user input modalities. For example, see the above-referenced co-pending patent application, especially FIG. **13** and the corresponding discussion of a "multi-modal maps" application, and FIG. **12** and the corresponding discussion of a "unified messaging" application. Another example is the InfoWiz interactive information kiosk developed by the assignee and described in the document entitled "InfoWiz: An Animated Voice Interactive Information System" available online at http://www.ai.sri.com/~oaa/applications.html. A copy of the InfoWhiz document is provided in an Information Disclosure Statement submitted herewith and incorporated herein by this reference. A further example is the "CommandTalk" application developed by the assignee for the U.S. military, as described online at http://www.ai.sri.com/~lesaf/commandtalk.html and in the following publications, copies of which are provided in an Information Disclosure Statement submitted herewith and incorporated herein by this reference:

"CommandTalk: A Spoken-Language Interface for Battlefield Simulations", 1997, by Robert Moore, John Dowding, Harry Bratt, J. Mark Gawron, Yonael Gorfu and Adam Cheyer, in "Proceedings of the Fifth Conference on Applied Natural Language Processing", Washington, D.C., pp. 1–7, Association for Computational Linguistics

"The CommandTalk Spoken Dialogue System", 1999, by Amanda Stent, John Dowding, Jean Mark Gawron, Elizabeth Owen Bratt and Robert Moore, in "Proceedings of the Thirty-Seventh Annual Meeting of the ACL", pp. 183–190, University of Maryland, College Park, Md., Association for Computational Linguistics

"Interpreting Language in Context in CommandTalk", 1999, by John Dowding and Elizabeth Owen Bratt and Sharon Goldwater, in "Communicative Agents: The Use of Natural Language in Embodied Systems", pp. 63–67, Association for Computing Machinery (ACM) Special Interest Group on Artificial Intelligence (SIGART), Seattle, Wash.

For some applications and systems, OAA can provide an advantageous platform for constructing embodiments of the present invention. For example, a representative application is now briefly presented, with reference to FIG. **6**. If the statement "show me movies starring John Wayne" is spoken into the voice input device, the voice data for this request will be sent by UI agent **650** to facilitator **600**, which in turn will ask natural language (NL) agent **620** and speech recognition agent **610** to interpret the query and return the interpretation in ICL format. The resulting ICL goal expression is then routed by the facilitator to appropriate agents—in this case, video-on-demand database agent **640**—to execute the request. Video database agent **640** preferably includes or is coupled to an appropriate embodiment of query construction logic **330** and query refinement logic **340**, and may also issue ICL requests to facilitator **600** for additional assistance—e.g., display of menus and capture of additional user input in the event that query refinement is needed—and facilitator **600** will delegate such requests to appropriate client agents in the community. When the desired video content is ultimately retrieved by video database agent **640**, UI agent **650** is invoked by facilitator **600** to display the movie.

Other spoken user requests, such as a request for the current weather in New York City or for a stock quote, would eventually lead facilitator to invoke web database agent **630** to access the desired information from an appropriate Internet site. Here again, web database agent **630** preferably includes or is coupled to an appropriate embodiment of query construction logic **330** and query refinement logic **340**, including a scraping utility such as WebL. Other spoken requests, such as a request to view recent emails or access voice mail, would lead the facilitator to invoke the appropriate email agent **660** and/or telephone agent **680**. A request to record a televised program of interest might lead facilitator **600** to invoke web database agent **630** to return televised program schedule information, and then invoke VCR controller agent **680** to program the associated VCR unit to record the desired television program at the scheduled time.

Control and connectivity embracing additional electronic home appliances (e.g., microwave oven, home surveillance system, etc.) can be integrated in comparable fashion. Indeed, an advantage of OAA-based embodiments of the present invention, that will be apparent to practitioners in light of the above teachings and in light of the teachings disclosed in the cited co-pending patent applications, is the relative ease and flexibility with which additional service agents can be plugged into the existing platform, immediately enabling the facilitator to respond dynamically to spoken natural language requests for the corresponding services.

4. Further Embodiments and Equivalents

While the present invention has been described in terms of several preferred embodiments, there are many alterations, permutations, and equivalents that may fall within the scope of this invention. It should also be noted that there are many alternative ways of implementing the methods and apparatuses of the present invention. It is therefore intended that the following appended claims be interpreted as including all such alterations, permutations, and equivalents as fall within the true spirit and scope of the present invention.

What is claimed is:

1. A method for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, wherein a data link is established between a mobile information appliance of the user and the one or more network servers, comprising the steps of:

(a) receiving a spoken request for desired information from the user utilizing the mobile information appliance of the user, wherein said mobile information appliance comprises a portable remote control device or a set-top box for a television;

(b) rendering an interpretation of the spoken request;

(c) constructing a navigation query based upon the interpretation;

(d) utilizing the navigation query to select a portion of the electronic data source; and

(e) transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user.

2. The method of claim **1**, wherein the step of rendering the interpretation of the spoken request is performed by the mobile information appliance.

3. The method of claim **1**, wherein the step of rendering the interpretation of the spoken request is performed by the mobile information appliance.

4. The method of claim **1**, further comprising the steps of soliciting additional input from the user, including user interaction in a modality different than the original request; refining the navigation query, based upon the additional input; and using the refined navigation query to select a portion of the electronic data source.

5. The method of claim **1**, wherein the data link includes a cellular telephone system.

6. The method of claim **1**, wherein steps (a)–(d) are performed with respect to multiple users.

7. The method of claim **1**, wherein the mobile information appliance is a wireless telephone.

8. The method of claim **1**, wherein the mobile information appliance is a portable computing device.

9. The method of claim **8**, wherein the portable computing device is a personal digital assistant.

10. A computer program embodied on a computer readable medium for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, wherein a data link is established between a mobile information appliance of the user and the one or more network servers, comprising:

(a) a code segment that receives a spoken request for desired information from the user utilizing the mobile information appliance of the user, wherein said mobile information appliance comprises a portable remote control device or a set-top box for a television;

(b) a code segment that renders an interpretation of the spoken request;

(c) a code segment that constructs a navigation query based upon the interpretation;

(d) a code segment that utilizes the navigation query to select a portion of the electronic data source; and

(e) a code segment that transmits the selected portion of the electronic data source from the network server to the mobile information appliance of the user.

11. The computer program of claim **10**, wherein the rendering of the interpretation of the spoken request is performed at the one or more network servers.

12. The computer program of claim **10**, wherein the rendering of the interpretation of the spoken request is performed by the mobile information appliance.

13. The computer program of claim **10**, further comprising a code segment that solicits additional input from the user, including user interaction in a modality different than the original request; a code segment that refines the navigation query, based upon the additional input; and a code segment that uses the refined navigation query to select a portion of the electronic data source.

14. The computer program of claim **10**, wherein the data link includes a wireless telephone system.

15. The computer program of claim **10**, wherein code segments (a)–(d) are executed with respect to multiple users.

16. The computer program of claim **10**, wherein the mobile information appliance is a wireless telephone.

17. The computer program of claim **10**, wherein the mobile information appliance is a portable computing device.

18. The computer program of claim **17**, wherein the portable computing device is a personal digital assistant.

19. A system for speech-based navigation of an electronic data source located at one or more network servers located remotely from a user, comprising:

(a) a mobile information appliance operable to receive a spoken request for desired information from the user, wherein said mobile information appliance comprises a portable remote control device or a set-top box for a television;

(b) spoken language processing logic, operable to render an interpretation of the spoken request;

(c) query construction logic, operable to construct a navigation query based upon the interpretation;

(d) navigation logic, operable to select a portion of the electronic data source using the navigation query, and

(e) electronic communications infrastructure for transmitting the selected portion of the electronic data source from the network server to the mobile information appliance of the user.

20. The system of claim **19**, wherein the spoken language processing logic renders the interpretation of the spoken request at the one or more network servers.

21. The system of claim **19**, wherein the spoken language processing logic renders the interpretation of the spoken request at the mobile information appliance.

22. The system of claim **19**, further comprising user interaction logic operable to solicit additional input from the user, including user interaction in a modality different than the original request; and query refining logic operable to refine the navigation query based upon the additional input; wherein the navigation logic users the refined navigation query to select a portion of the electronic data source.

23. The system of claim **19**, wherein the data link includes a cellular telephone system.

24. The system of claim **19**, wherein the system operates with respect to multiple users.

25. The system of claim **19**, wherein the mobile information appliance is a wireless telephone.

26. The system of claim **19**, wherein the mobile information appliance is a portable computing device.

27. The system of claim **26**, wherein the portable computing device is a personal digital assistant.

* * * * *

US006523061B1

(12) **United States Patent**
Halverson et al.

(10) **Patent No.: US 6,523,061 B1**
(45) **Date of Patent: Feb. 18, 2003**

(54) **SYSTEM, METHOD, AND ARTICLE OF MANUFACTURE FOR AGENT-BASED NAVIGATION IN A SPEECH-BASED DATA NAVIGATION SYSTEM**

(75) Inventors: **Christine Halverson**, San Jose, CA (US); **Luc Julia**, Menlo Park, CA (US); **Dimitris Voutsas**, Thessaloniki (GR); **Adam Cheyer**, Palo Alto, CA (US)

(73) Assignee: **SRI International, Inc.**, Menlo Park, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/607,672**

(22) Filed: **Jun. 30, 2000**

Related U.S. Application Data

(63) Continuation of application No. 09/524,095, filed on Mar. 13, 2000, which is a continuation-in-part of application No. 09/225,198, filed on Jan. 5, 1999.

(60) Provisional application No. 60/124,720, filed on Mar. 17, 1999, provisional application No. 60/124,719, filed on Mar. 17, 1999, and provisional application No. 60/124,718, filed on Mar. 17, 1999.

(51) **Int. Cl.**^{7} **G06F 15/16**

(52) **U.S. Cl.** **709/202**; 709/202; 709/217; 709/219; 379/88.01; 379/88.02; 379/88.22; 370/331; 704/270; 704/275

(58) **Field of Search** 395/700.12; 370/331, 370/202; 709/217–219, 227; 379/88.01, 88.17, 88.22, 88.02, 90.01, 900; 704/270, 275; 382/313; 707/203

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,197,005 A	3/1993	Shwartz et al.	....	364/419
5,386,556 A	1/1995	Hedin et al.	....	395/600
5,434,777 A	7/1995	Luciw	....	364/419.13
5,519,608 A	5/1996	Kupiec	....	364/419.08

(List continued on next page.)

OTHER PUBLICATIONS

http://www–3.ibm.com/software.speech/desktop/w9–pro.html. IBM Via Voice for windows, Pro USB edition release 9 by IBM corp.*

Stent, Amanda et al., "The CommandTalk Spoken Dialogue System", SRI International.

Moore, Robert et al., "CommandTalk: A Spoken–Language Interface for Battlefield Simulations", Oct. 23, 1997, SRI International.

Dowding, John et al., "Interpreting Language in Context in CommandTalk", Feb. 5, 1999, SRI International.

http://www.ai.sri.com/~oaa/infowiz.html, InfoWiz: An Animated Voice Interactive Information System, May 8, 2000.

Dowding, John, "Interleaving Syntax and Semantics in an Efficient Bottom–up Parser", SRI International.

Moore, Robert et al., "Combining Linguistic and Statistical Knowledge Sources in Natural–Language Processing for ATIS", SRI International.

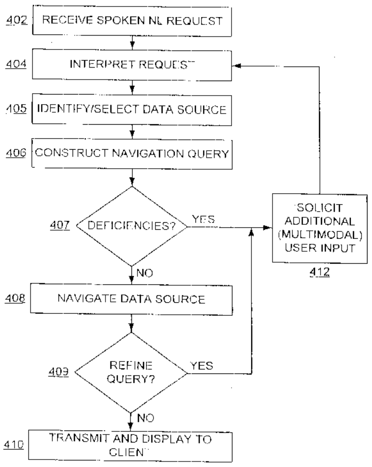
Dowding, John et al., "Gemini: A Natural Language System For Spoken–Language Understanding", SRI International.

Primary Examiner—Ayaz Sheikh
Assistant Examiner—Thu Ha Nguyen
(74) *Attorney, Agent, or Firm*—Moser, Patterson & Sheridan, LLP.; Kin-Wah Tong, Esq.

(57) **ABSTRACT**

A system, method, and article of manufacture are provided for navigating an electronic data source by means of spoken language where a portion of the data link between a mobile information appliance of the user and the data source utilizes wireless communication. When a spoken input request is received from a user, it is interpreted. The resulting interpretation of the request is thereupon used to automatically construct an operational navigation query. The navigation query is routed to one or more agents, which use the navigation query to retrieve the desired information from one or more electronic network data sources.

18 Claims, 7 Drawing Sheets


402 RECEIVE SPOKEN NL REQUEST
404 INTERPRET REQUEST
405 IDENTIFY/SELECT DATA SOURCE
406 CONSTRUCT NAVIGATION QUERY
407 DEFICIENCIES?
YES
NO
SOLICIT ADDITIONAL (MULTIMODAL) USER INPUT
412
408 NAVIGATE DATA SOURCE
409 REFINE QUERY?
YES
NO
410 TRANSMIT AND DISPLAY TO CLIENT

U.S. PATENT DOCUMENTS

5,608,624	A		3/1997	Luciw	395/794
5,717,860	A	*	2/1998	Graber et al.	395/200.12
5,721,938	A		2/1998	Stuckey	395/754
5,729,659	A		3/1998	Potter	395/2.79
5,748,974	A		5/1998	Johnson	395/759
5,774,859	A		6/1998	Houser et al.	704/275
5,794,050	A		8/1998	Dahlgren et al.	395/708
5,802,526	A		9/1998	Fawcett et al.	707/104
5,805,775	A		9/1998	Eberman et al.	395/12
5,855,002	A		12/1998	Armstrong	704/270
5,884,262	A	*	3/1999	Wise et al.	704/270
5,890,123	A		3/1999	Brown et al.	704/275
5,902,353	A	*	5/1999	Reber et al.	709/219
5,949,772	A	*	9/1999	Sugikawa et al.	370/331
5,963,940	A		10/1999	Liddy et al.	707/5
5,978,848	A	*	11/1999	Maddalozzo, Jr. et al.	709/227
6,003,072	A		12/1999	Gerritsen et al.	709/218
6,012,030	A		1/2000	French-St. George et al.	704/275
6,026,388	A		2/2000	Liddy et al.	707/1
6,026,437	A	*	2/2000	Muschett et al.	709/219
6,052,716	A	*	4/2000	Gibson	709/217
6,101,473	A	*	8/2000	Scott et al.	704/275
6,157,705	A	*	12/2000	Perrone	379/88.01
6,282,270	B1	*	8/2001	Porter	379/88.17
6,282,511	B1	*	8/2001	Mayer	704/270
6,289,140	B1	*	9/2001	Oliver	382/313

* cited by examiner

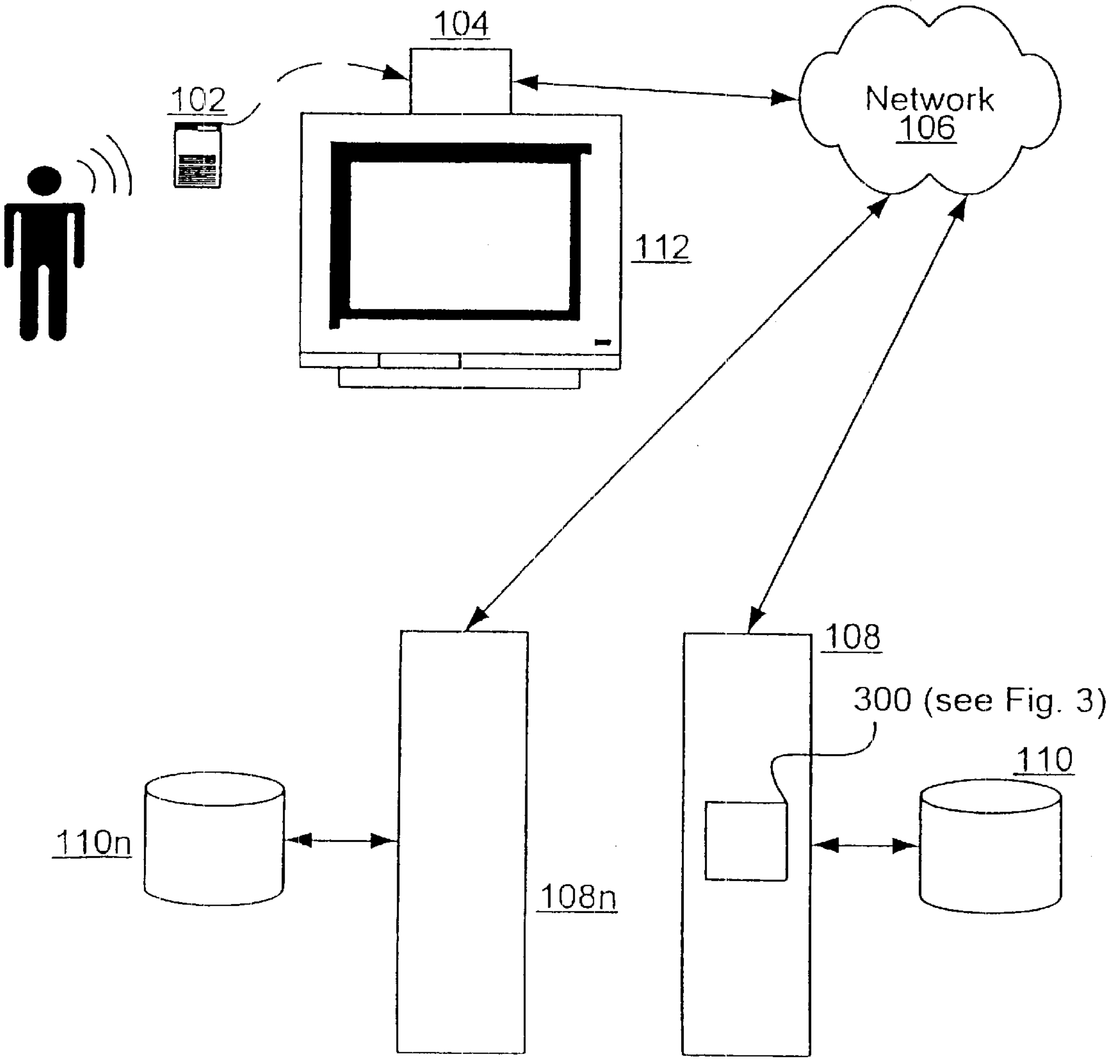


Fig. 1a

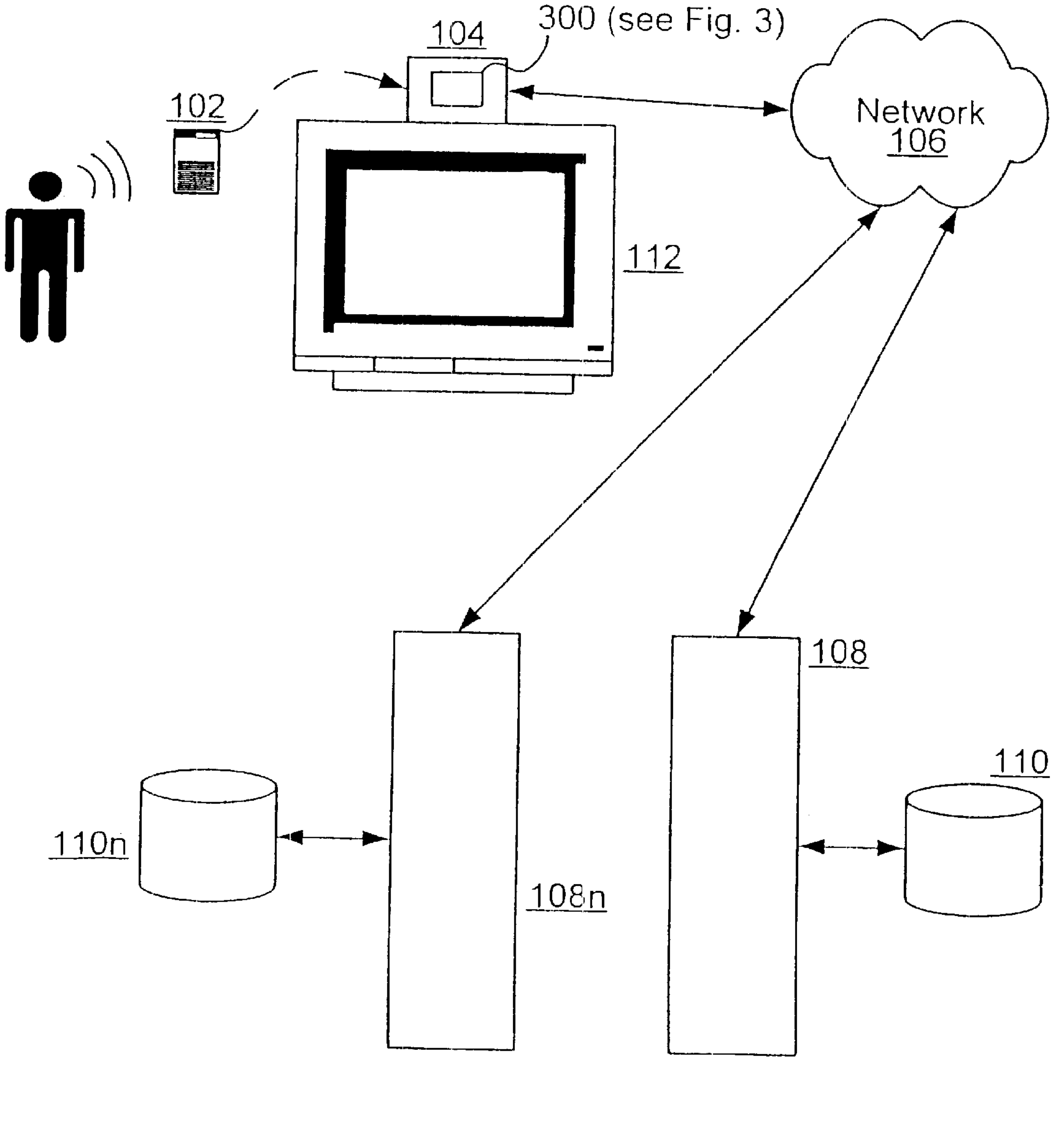


Fig. 1b

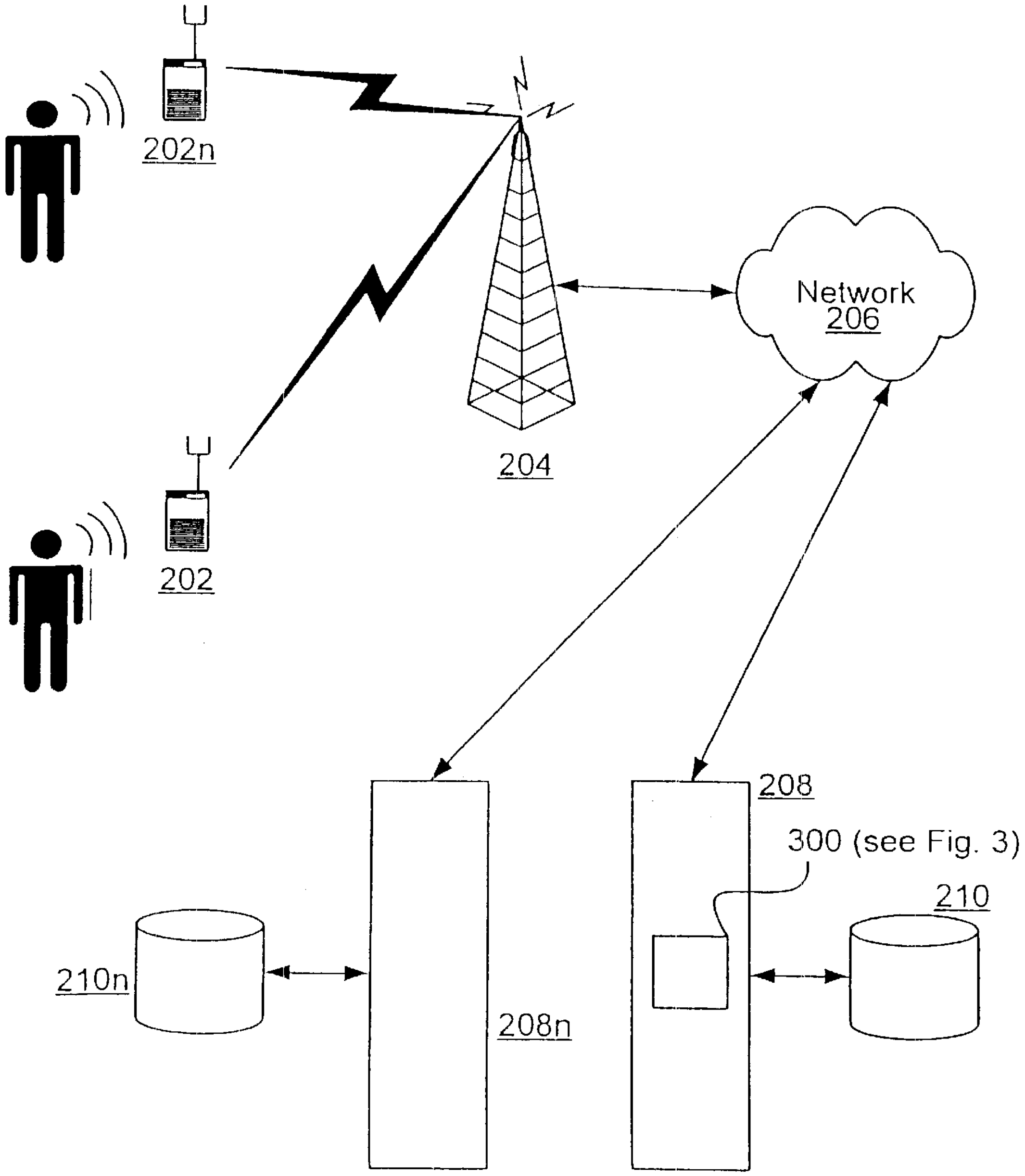


Fig. 2

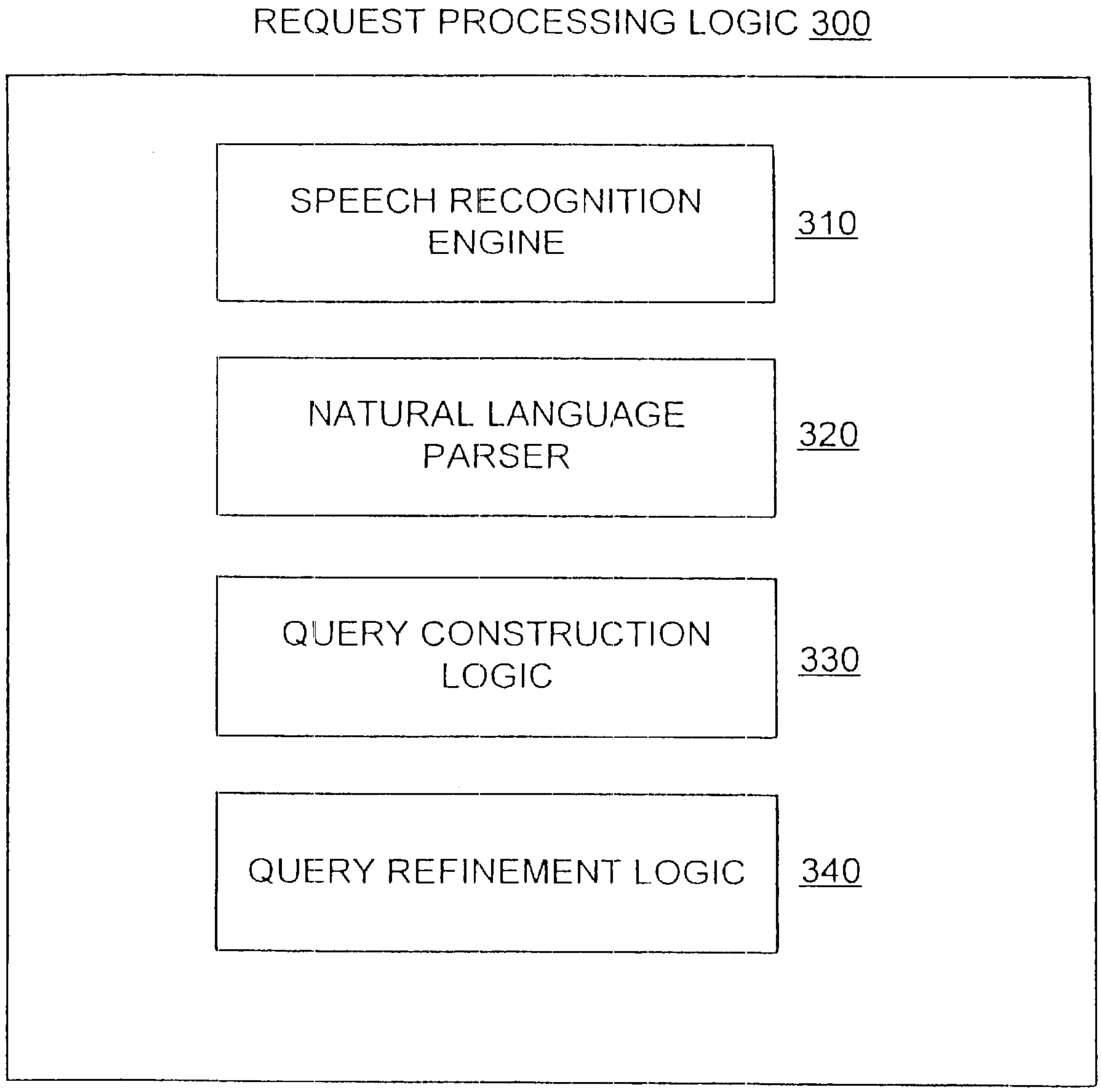

REQUEST PROCESSING LOGIC 300
SPEECH RECOGNITION ENGINE
310
NATURAL LANGUAGE PARSER
320
QUERY CONSTRUCTION LOGIC
330
QUERY REFINEMENT LOGIC
340

Fig. 3

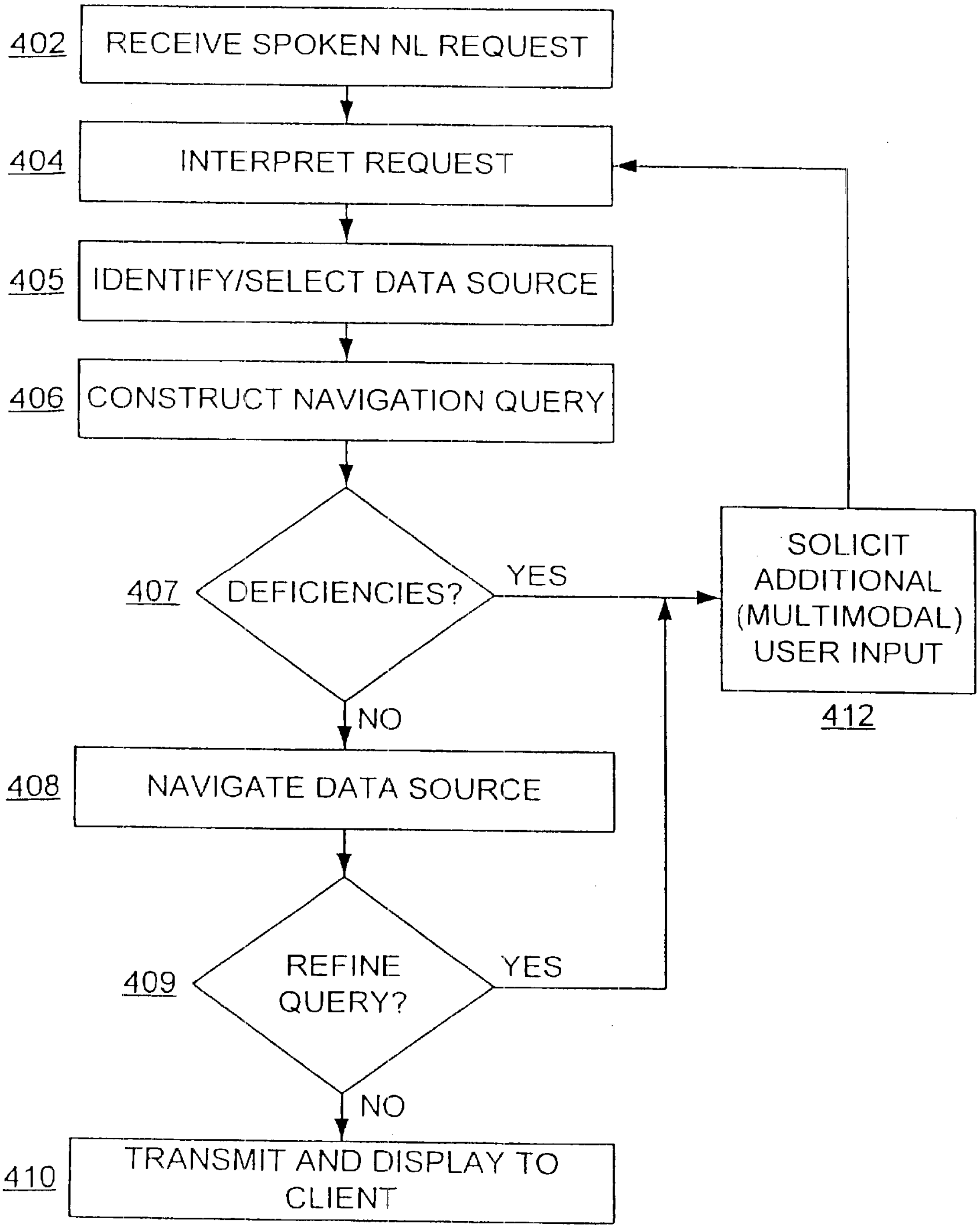

402
RECEIVE SPOKEN NL REQUEST
404
INTERPRET REQUEST
405
IDENTIFY/SELECT DATA SOURCE
406
CONSTRUCT NAVIGATION QUERY
407
DEFICIENCIES?
YES
SOLICIT ADDITIONAL (MULTIMODAL) USER INPUT
412
NO
408
NAVIGATE DATA SOURCE
409
REFINE QUERY?
YES
NO
410
TRANSMIT AND DISPLAY TO CLIENT

Fig. 4

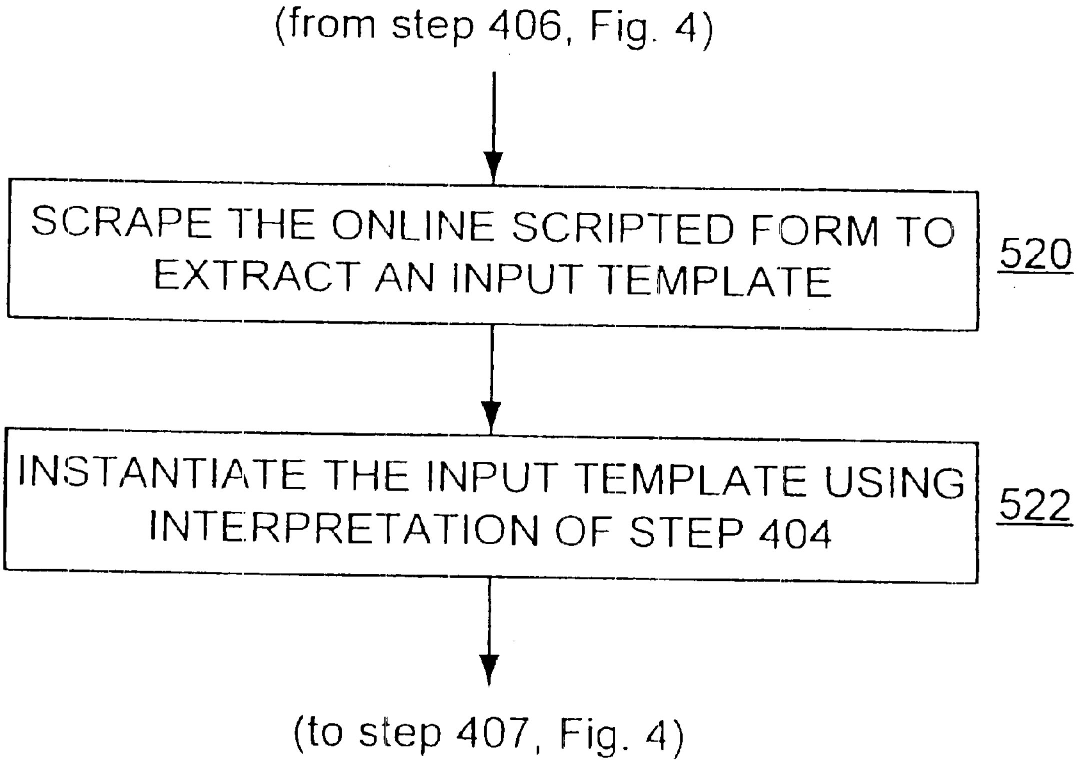

(from step 406, Fig. 4)
SCRAPE THE ONLINE SCRIPTED FORM TO EXTRACT AN INPUT TEMPLATE
520
INSTANTIATE THE INPUT TEMPLATE USING INTERPRETATION OF STEP 404
522
(to step 407, Fig. 4)

Fig. 5

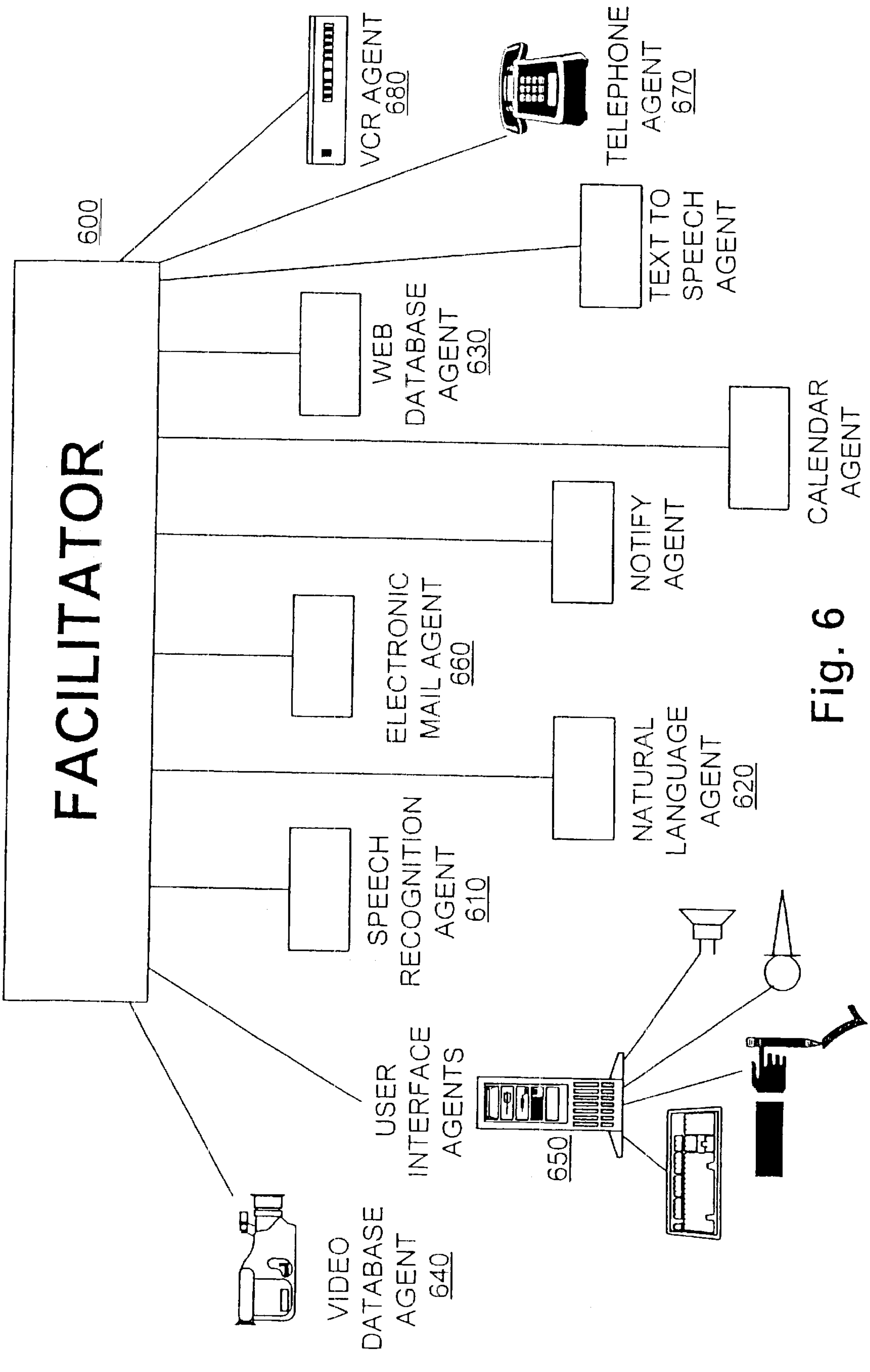

FACILITATOR
600
VCR AGENT
680
TELEPHONE AGENT
670
TEXT TO SPEECH AGENT
WEB DATABASE AGENT
630
CALENDAR AGENT
NOTIFY AGENT
ELECTRONIC MAIL AGENT
660
NATURAL LANGUAGE AGENT
620
SPEECH RECOGNITION AGENT
610
USER INTERFACE AGENTS
650
VIDEO DATABASE AGENT
640

Fig. 6

SYSTEM, METHOD, AND ARTICLE OF MANUFACTURE FOR AGENT-BASED NAVIGATION IN A SPEECH-BASED DATA NAVIGATION SYSTEM

This application is a continuation of an application entitled NAVIGATING NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN NATURAL LANGUAGE INPUT WITH MULTIMODAL ERROR FEEDBACK which was filed on Mar. 13, 2000 under Ser. No. 09/524,095 and which is a Continuation In Part of co-pending U.S. patent application Ser. No. 09/225,198, filed Jan. 5, 1999, Provisional U.S. patent application Ser. No. 60/124,718, filed Mar. 17, 1999, Provisional U.S. patent application Ser. No. 60/124,720, filed Mar. 17, 1999, and Provisional U.S. patent application Ser. No. 60/124,719, filed Mar. 17, 1999, from which applications priority is claimed and these application are incorporated herein by reference.

BACKGROUND OF THE INVENTION

The present invention relates generally to the navigation of electronic data by means of spoken natural language requests, and to feedback mechanisms and methods for resolving the errors and ambiguities that may be associated with such requests.

As global electronic connectivity continues to grow, and the universe of electronic data potentially available to users continues to expand, there is a growing need for information navigation technology that allows relatively naive users to navigate and access desired data by means of natural language input. In many of the most important markets—including the home entertainment arena, as well as mobile computing—spoken natural language input is highly desirable, if not ideal. As just one example, the proliferation of high-bandwidth communications infrastructure for the home entertainment market (cable, satellite, broadband) enables delivery of movies-on-demand and other interactive multimedia content to the consumer's home television set. For users to take full advantage of this content stream ultimately requires interactive navigation of content databases in a manner that is too complex for user-friendly selection by means of a traditional remote-control clicker. Allowing spoken natural language requests as the input modality for rapidly searching and accessing desired content is an important objective for a successful consumer entertainment product in a context offering a dizzying range of database content choices. As further examples, this same need to drive navigation of (and transaction with) relatively complex data warehouses using spoken natural language requests applies equally to surfing the Internet/Web or other networks for general information, multimedia content, or e-commerce transactions.

In general, the existing navigational systems for browsing electronic databases and data warehouses (search engines, menus, etc.), have been designed without navigation via spoken natural language as a specific goal. So today's world is full of existing electronic data navigation systems that do not assume browsing via natural spoken commands, but rather assume text and mouse-click inputs (or in the case of TV remote controls, even less). Simply recognizing voice commands within an extremely limited vocabulary and grammar—the spoken equivalent of button/click input (e.g., speaking "channel **5**" selects TV channel **5**)—is really not sufficient by itself to satisfy the objectives described above. In order to deliver a true "win" for users, the voice-driven front-end must accept spoken natural language input in a manner that is intuitive to users. For example, the front-end should not require learning a highly specialized command language or format. More fundamentally, the front-end must allow users to speak directly in terms of what the user ultimately wants—e.g., "I'd like to see a Western film directed by Clint Eastwood"—as opposed to speaking in terms of arbitrary navigation structures (e.g., hierarchical layers of menus, commands, etc.) that are essentially artifacts reflecting constraints of the pre-existing text/click navigation system. At the same time, the front-end must recognize and accommodate the reality that a stream of naive spoken natural language input will, over time, typically present a variety of errors and/or ambiguities: e.g., garbled/unrecognized words (did the user say "Eastwood" or "Easter"?) and under-constrained requests ("Show me the Clint Eastwood movie"). An approach is needed for handling and resolving such errors and ambiguities in a rapid, user-friendly, non-frustrating manner.

What is needed is a methodology and apparatus for rapidly constructing a voice-driven front-end atop an existing, non-voice data navigation system, whereby users can interact by means of intuitive natural language input not strictly conforming to the step-by-step browsing architecture of the existing navigation system, and wherein any errors or ambiguities in user input are rapidly and conveniently resolved. The solution to this need should be compatible with the constraints of a multi-user, distributed environment such as the Internet/Web or a proprietary high-bandwidth content delivery network; a solution contemplating one-at-a-time user interactions at a single location is insufficient, for example.

SUMMARY OF THE INVENTION

The present invention addresses the above needs by providing a system, method, and article of manufacture for using agents for navigation of network-based electronic data sources in response to spoken input requests. When a spoken input request is received from a user, it is interpreted, such as by using a speech recognition agent to extract speech data from acoustic voice signals, and using a language parsing agent to linguistically parse the speech data. The interpretation of the spoken request can be performed on a computing device locally with the user, such as the mobile information appliance, or remotely from the user. The resulting interpretation of the request is thereupon used to automatically construct an operational navigation query. The navigation query is routed to one or more agents that use the navigation query to retrieve the desired information from one or more electronic network data sources, which is then transmitted to a client device of the user. If the network data source is a database, the navigation query is constructed in the format of a database query language.

Typically, errors or ambiguities emerge in the interpretation of the spoken NL request, such that the system cannot instantiate a complete, valid navigational template. This is to be expected occasionally, and one preferred aspect of the invention is the ability to handle such errors and ambiguities in relatively graceful and user-friendly manner. Instead of simply rejecting such input and defaulting to traditional input modes or simply asking the user to try again, a preferred embodiment of the present invention seeks to converge rapidly toward instantiation of a valid navigational template by soliciting additional clarification from the user as necessary, either before or after a navigation of the data source, via multimodal input, i.e., by means of menu selection or other input modalities including and in addition to spoken natural language. This clarifying, multi-modal dialogue takes advantage of whatever partial navigational information has been gleaned from the initial interpretation of the user's spoken NL request. This clarification process continues until the system converges toward an adequately instantiated navigational template, which is in turn used to navigate the network-based data and retrieve the user's desired information. The retrieved information is transmitted across the network and presented to the user on a suitable client display device.

In a further aspect of the present invention, the construction of the navigation query includes extracting an input template for an online scripted interface to the data source and using the input template to construct the navigation query. The extraction of the input template can include dynamically scraping the online scripted interface.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention, together with further advantages thereof, may best be understood by reference to the following description taken in conjunction with the accompanying drawings in which:

FIG. **1***a* illustrates a system providing a spoken natural language interface for network-based information navigation, in accordance with an embodiment of the present invention with server-side processing of requests;

FIG. **1***b* illustrates another system providing a spoken natural language interface for network-based information navigation, in accordance with an embodiment of the present invention with client-side processing of requests;

FIG. **2** illustrates a system providing a spoken natural language interface for network-based information navigation, in accordance with an embodiment of the present invention for a mobile computing scenario;

FIG. **3** illustrates the functional logic components of a request processing module in accordance with an embodiment of the present invention;

FIG. **4** illustrates a process utilizing spoken natural language for navigating an electronic database in accordance with one embodiment of the present invention;

FIG. **5** illustrates a process for constructing a navigational query for accessing an online data source via an interactive, scripted (e.g., CGI) form; and

FIG. **6** illustrates an embodiment of the present invention utilizing a community of distributed, collaborating electronic agents.

DETAILED DESCRIPTION OF THE INVENTION

1. System Architecture

a. Server-End Processing of Spoken Input

FIG. **1***a* is an illustration of a data navigation system driven by spoken natural language input, in accordance with one embodiment of the present invention. As shown, a user's voice input data is captured by a voice input device **102**, such as a microphone. Preferably voice input device **102** includes a button or the like that can be pressed or held-down to activate a listening mode, so that the system need not continually pay attention to, or be confused by, irrelevant background noise. In one preferred embodiment well-suited for the home entertainment setting, voice input device **102** is a portable remote control device with an integrated microphone, and the voice data is transmitted from device **102** preferably via infrared (or other wireless) link to communications box **104** (e.g., a set-top box or a similar communications device that is capable of retransmitting the raw voice data and/or processing the voice data) local to the user's environment and coupled to communications network **106**. The voice data is then transmitted across network **106** to a remote server or servers **108**. The voice data may preferably be transmitted in compressed digitized form, or alternatively—particularly where bandwidth constraints are significant—in analog format (e.g., via frequency modulated transmission), in the latter case being digitized upon arrival at remote server **108**.

At remote server **108**, the voice data is processed by request processing logic **300** in order to understand the user's request and construct an appropriate query or request for navigation of remote data source **110**, in accordance with the interpretation process exemplified in FIG. **4** and FIG. **5** and discussed in greater detail below. For purposes of executing this process, request processing logic **300** comprises functional modules including speech recognition engine **310**, natural language (NL) parser **320**, query construction logic **330**, and query refinement logic **340**, as shown in FIG. **3**. Data source **110** may comprise database(s), Internet/web site(s), or other electronic information repositories, and preferably resides on a central server or servers—which may or may not be the same as server **108**, depending on the storage and bandwidth needs of the application and the resources available to the practitioner. Data source **110** may include multimedia content, such as movies or other digital video and audio content, other various forms of entertainment data, or other electronic information. The contents of data source **110** are navigated—i.e., the contents are accessed and searched, for retrieval of the particular information desired by the user—using the processes of FIGS. **4** and **5** as described in greater detail below.

Once the desired information has been retrieved from data source **110**, it is electronically transmitted via network **106** to the user for viewing on client display device **112**. In a preferred embodiment well-suited for the home entertainment setting, display device **112** is a television monitor or similar audiovisual entertainment device, typically in stationary position for comfortable viewing by users. In addition, in such preferred embodiment, display device **112** is coupled to or integrated with a communications box (which is preferably the same as communications box **104**, but may also be a separate unit) for receiving and decoding/formatting the desired electronic information that is received across communications network **106**.

Network **106** is a two-way electronic communications network and may be embodied in electronic communication infrastructure including coaxial (cable television) lines, DSL, fiber-optic cable, traditional copper wire (twisted pair), or any other type of hardwired connection. Network **106** may also include a wireless connection such as a satellite-based connection, cellular connection, or other type of wireless connection. Network **106** may be part of the Internet and may support TCP/IP communications, or may be embodied in a proprietary network, or in any other electronic communications network infrastructure, whether packet-switched or connection-oriented. A design consideration is that network **106** preferably provide suitable bandwidth depending upon the nature of the content anticipated for the desired application.

b. Client-End Processing of Spoken Input

FIG. **1***b* is an illustration of a data navigation system driven by spoken natural language input, in accordance with a second embodiment of the present invention. Again, a user's voice input data is captured by a voice input device

102, such as a microphone. In the embodiment shown in FIG. 1*b*, the voice data is transmitted from device **202** to requests processing logic **300**, hosted on a local speech processor, for processing and interpretation. In the preferred embodiment illustrated in FIG. 1*b*, the local speech processor is conveniently integrated as part of communications box **104**, although implementation in a physically separate (but communicatively coupled) unit is also possible as will be readily apparent to those of skill in the art. The voice data is processed by the components of request processing logic **300** in order to understand the user's request and construct an appropriate query or request for navigation of remote data source **110**, in accordance with the interpretation process exemplified in FIGS. **4** and **5** as discussed in greater detail below.

The resulting navigational query is then transmitted electronically across network **106** to data source **110**, which preferably resides on a central server or servers **108**. As in FIG. 1*a*, data source **110** may comprise database(s), Internet/web site(s), or other electronic information repositories, and preferably may include multimedia content, such as movies or other digital video and audio content, other various forms of entertainment data, or other electronic information. The contents of data source **110** are then navigated—i.e., the contents are accessed and searched, for retrieval of the particular information desired by the user—preferably using the process of FIGS. **4** and **5** as described in greater detail below. Once the desired information has been retrieved from data source **110**, it is electronically transmitted via network **106** to the user for viewing on client display device **112**.

In one embodiment in accordance with FIG. 1*b* and well-suited for the home entertainment setting, voice input device **102** is a portable remote control device with an integrated microphone, and the voice data is transmitted from device **102** preferably via infrared (or other wireless) link to the local speech processor. The local speech processor is coupled to communications network **106**, and also preferably to client display device **112** (especially for purposes of query refinement transmissions, as discussed below in connection with FIG. **4**, step **412**), and preferably may be integrated within or coupled to communications box **104**. In addition, especially for purposes of a home entertainment application, display device **112** is preferably a television monitor or similar audiovisual entertainment device, typically in stationary position for comfortable viewing by users. In addition, in such preferred embodiment, display device **112** is coupled to a communications box (which is preferably the same as communications box **104**, but may also be a physically separate unit) for receiving and decoding/formatting the desired electronic information that is received across communications network **106**.

Design considerations favoring server-side processing and interpretation of spoken input requests, as exemplified in FIG. 1*a*, include minimizing the need to distribute costly computational hardware and software to all client users in order to perform speech and language processing. Design considerations favoring client-side processing, as exemplified in FIG. 1*b*, include minimizing the quantity of data sent upstream across the network from each client, as the speech recognition is performed before transmission across the network and only the query data and/or request needs to be sent, thus reducing the upstream bandwidth requirements.

c. Mobile Client Embodiment

A mobile computing embodiment of the present invention may be implemented by practitioners as a variation on the embodiments of either FIG. 1*a* or FIG. 1*b*. For example, as depicted in FIG. **2**, a mobile variation in accordance with the server-side processing architecture illustrated in FIG. 1*a* may be implemented by replacing voice input device **102**, communications box **104**, and client display device **112**, with an integrated, mobile, information appliance **202** such as a cellular telephone or wireless personal digital assistant (wireless PDA). Mobile information appliance **202** essentially performs the functions of the replaced components. Thus, mobile information appliance **202** receives spoken natural language input requests from the user in the form of voice data, and transmits that data (preferably via wireless data receiving station **204**) across communications network **206** for server-side interpretation of the request, in similar fashion as described above in connection with FIG. **1**. Navigation of data source **210** and retrieval of desired information likewise proceeds in an analogous manner as described above. Display information transmitted electronically back to the user across network **206** is displayed for the user on the display of information appliance **202**, and audio information is output through the appliance's speakers.

Practitioners will further appreciate, in light of the above teachings, that if mobile information appliance **202** is equipped with sufficient computational processing power, then a mobile variation of the client-side architecture exemplified in FIG. **2** may similarly be implemented. In that case, the modules corresponding to request processing logic **300** would be embodied locally in the computational resources of mobile information appliance **202**, and the logical flow of data would otherwise follow in a manner analogous to that previously described in connection with FIG. 1*b*.

As illustrated in FIG. **2**, multiple users, each having their own client input device, may issue requests, simultaneously or otherwise, for navigation of data source **210**. This is equally true (though not explicitly drawn) for the embodiments depicted in FIGS. 1*a* and 1*b*. Data source **210** (or **100**), being a network accessible information resource, has typically already been constructed to support access requests from simultaneous multiple network users, as known by practitioners of ordinary skill in the art. In the case of server-side speech processing, as exemplified in FIGS. 1*a* and **2**, the interpretation logic and error correction logic modules are also preferably designed and implemented to support queuing and multi-tasking of requests from multiple simultaneous network users, as will be appreciated by those of skill in the art.

It will be apparent to those skilled in the art that additional implementations, permutations and combinations of the embodiments set forth in FIGS. 1*a*, 1*b*, and **2** may be created without straying from the scope and spirit of the present invention. For example, practitioners will understand, in light of the above teachings and design considerations, that it is possible to divide and allocate the functional components of request processing logic **300** between client and server. For example, speech recognition—in entirety, or perhaps just early stages such as feature extraction—might be performed locally on the client end, perhaps to reduce bandwidth requirements, while natural language parsing and other necessary processing might be performed upstream on the server end, so that more extensive computational power need not be distributed locally to each client. In that case, corresponding portions of request processing logic **300**, such as speech recognition engine **310** or portions thereof, would reside locally at the client as in FIG. 1*b*, while other component modules would be hosted at the server end as in FIGS. 1*a* and **2**.

Further, practitioners may choose to implement the each of the various embodiments described above on any number of different hardware and software computing platforms and

environments and various combinations thereof, including, by way of just a few examples: a general-purpose hardware microprocessor such as the Intel Pentium series; operating system software such as Microsoft Windows/CE, Palm OS, or Apple Mac OS (particularly for client devices and client-side processing), or Unix, Linux, or Windows/NT (the latter three particularly for network data servers and server-side processing), and/or proprietary information access platforms such as Microsoft's WebTV or the Diva Systems video-on-demand system.

2. Processing Methodology

The present invention provides a spoken natural language interface for interrogation of remote electronic databases and retrieval of desired information. A preferred embodiment of the present invention utilizes the basic methodology outlined in the flow diagram of FIG. **4** in order to provide this interface. This methodology will now be discussed.

a. Interpreting Spoken Natural Language Requests

At step **402**, the user's spoken request for information is initially received in the form of raw (acoustic) voice data by a suitable input device, as previously discussed in connection with FIGS. **1**–**2**. At step **404** the voice data received from the user is interpreted in order to understand the user's request for information. Preferably this step includes performing speech recognition in order to extract words from the voice data, and further includes natural language parsing of those words in order to generate a structured linguistic representation of the user's request.

Speech recognition in step **404** is performed using speech recognition engine **310**. A variety of commercial quality, speech recognition engines are readily available on the market, as practitioners will know. For example, Nuance Communications offers a suite of speech recognition engines, including Nuance **6**, its current flagship product, and Nuance Express, a lower cost package for entry-level applications. As one other example, IBM offers the ViaVoice speech recognition engine, including a low-cost shrink-wrapped version available through popular consumer distribution channels. Basically, a speech recognition engine processes acoustic voice data and attempts to generate a text stream of recognized words.

Typically, the speech recognition engine is provided with a vocabulary lexicon of likely words or phrases that the recognition engine can match against its analysis of acoustical signals, for purposes of a given application. Preferably, the lexicon is dynamically adjusted to reflect the current user context, as established by the preceding user inputs. For example, if a user is engaged in a dialogue with the system about movie selection, the recognition engine's vocabulary may preferably be adjusted to favor relevant words and phrases, such as a stored list of proper names for popular movie actors and directors, etc. Whereas if the current dialogue involves selection and viewing of a sports event, the engine's vocabulary might preferably be adjusted to favor a stored list of proper names for professional sports teams, etc. In addition, a speech recognition engine is provided with language models that help the engine predict the most likely interpretation of a given segment of acoustical voice data, in the current context of phonemes or words in which the segment appears. In addition, speech recognition engines often echo to the user, in more or less real-time, a transcription of the engine's best guess at what the user has said, giving the user an opportunity to confirm or reject.

In a further aspect of step **404**, natural language interpreter (or parser) **320** linguistically parses and interprets the textual output of the speech recognition engine. In a preferred embodiment of the present invention, the natural-language interpreter attempts to determine both the meaning of spoken words (semantic processing) as well as the grammar of the statement (syntactic processing), such as the Gemini Natural Language Understanding System developed by SRI International. The Gemini system is described in detail in publications entitled "Gemini: A Natural Language System for Spoken-Language Understanding" and "Interleaving Syntax and Semantics in an Efficient Bottom-Up Parser," both of which are currently available online at http://www.ai.sri.com/natural-language/projects/arpa-sls/nat-lang.html. (Copies of those publications are also included in an information disclosure statement submitted herewith, and are incorporated herein by this reference). Briefly, Gemini applies a set of syntactic and semantic grammar rules to a word string using a bottom-up parser to generate a logical form, which is a structured representation of the context-independent meaning of the string. Gemini can be used with a variety of grammars, including general English grammar as well as application-specific grammars. The Gemini parser is based on "unification grammar," meaning that grammatical categories incorporate features that can be assigned values; so that when grammatical category expressions are matched in the course of parsing or semantic interpretation, the information contained in the features is combined, and if the feature values are incompatible the match fails.

It is possible for some applications to achieve a significant reduction in speech recognition error by using the natural-language processing system to re-score recognition hypotheses. For example, the grammars defined for a language parser like Gemini may be compiled into context-free grammar that, in turn, can be used directly as language models for speech recognition engines like the Nuance recognizer. Further details on this methodology are provided in the publication "Combining Linguistic and Statistical Knowledge Sources in Natural-Language Processing for ATIS" which is currently available online through http://www.ai.sri.com/natural-language/projects/arpa-sls/spnl-int.html. A copy of this publication is included in an information disclosure submitted herewith, and is incorporated herein by this reference.

In an embodiment of the present invention that may be preferable for some applications, the natural language interpreter "learns" from the past usage patterns of a particular user or of groups of users. In such an embodiment, the successfully interpreted requests of users are stored, and can then be used to enhance accuracy by comparing a current request to the stored requests, thereby allowing selection of a most probable result.

b. Constructing Navigation Queries

In step **405** request processing logic **300** identifies and selects an appropriate online data source where the desired information (in this case, current weather reports for a given city) can be found. Such selection may involve look-up in a locally stored table, or possibly dynamic searching through an online search engine, or other online search techniques. For some applications, an embodiment of the present invention may be implemented in which only access to a particular data source (such as a particular vendor's proprietary content database) is supported; in that case, step **405** may be trivial or may be eliminated entirely.

Step **406** attempts to construct a navigation query, reflecting the interpretation of step **404**. This operation is preferably performed by query construction logic **330**.

A "navigation query" means an electronic query, form, series of menu selections, or the like; being structured appropriately so as to navigate a particular data source of

interest in search of desired information. In other words, a navigation query is constructed such that it includes whatever content and structure is required in order to access desired information electronically from a particular database or data source of interest.

For example, for many existing electronic databases, a navigation query can be embodied using a formal database query language such as Standard Query Language (SQL). For many databases, a navigation query can be constructed through a more user-friendly interactive front-end, such as a series of menus and/or interactive forms to be selected or filled in. SQL is a standard interactive and programming language for getting information from and updating a database. SQL is both an ANSI and an ISO standard. As is well known to practitioners, a Relational Database Management System (RDBMS), such as Microsoft's Access, Oracle's Oracle7, and Computer Associates' CA-OpenIngres, allow programmers to create, update, and administer a relational database. Practitioners of ordinary skill in the art will be thoroughly familiar with the notion of database navigation through structured query, and will be readily able to appreciate and utilize the existing data structures and navigational mechanisms for a given database, or to create such structures and mechanisms where desired.

In accordance with the present invention, the query constructed in step **406** must reflect the user's request as interpreted by the speech recognition engine and the NL parser in step **404**. In embodiments of the present invention wherein data source **110** (or **210** in the corresponding embodiment of FIG. **2**) is a structured relational database or the like, step **406** of the present invention may entail constructing an appropriate Structured Query Language (SQL) query or the like, or automatically filling out a front-end query form, series of menus or the like, as described above.

In many existing Internet (and Intranet) applications, an online electronic data source is accessible to users only through the medium of interaction with a so-called Common Gateway Interface (CGI) script. Typically the user who visits a web site of this nature must fill in the fields of an online interactive form. The online form is in turn linked to a CGI script, which transparently handles actual navigation of the associated data source and produces output for viewing by the user's web browser. In other words, direct user access to the data source is not supported, only mediated access through the form and CGI script is offered.

For applications of this nature, an advantageous embodiment of the present invention "scrapes" the scripted online site where information desired by a user may be found in order to facilitate construction of an effective navigation query. For example, suppose that a user's spoken natural language request is: "What's the weather in Miami?" After this request is received at step **402** and interpreted at step **404**, assume that step **405** determines that the desired weather information is available online through the medium of a CGI-scripted interactive form. Step **406** is then preferably carried out using the expanded process diagrammed in FIG. **5**. In particular, at sub-step **520**, query construction logic **330** electronically "scrapes" the online interactive form, meaning that query construction logic **330** automatically extracts the format and structure of input fields accepted by the online form. At sub-step **522**, a navigation query is then constructed by instantiating (filling in) the extracted input format—essentially an electronic template—in a manner reflecting the user's request for information as interpreted in step **404**. The flow of control then returns to step **407** of FIG. **4**. Ultimately, when the query thus constructed by scraping is used to navigate the online data source in step **408**, the query effectively initiates the same scripted response as if a human user had visited the online site and had typed appropriate entries into the input fields of the online form.

In the embodiment just described, scraping step **520** is preferably carried out with the assistance of an online extraction utility such as WebL. WebL is a scripting language for automating tasks on the World Wide Web. It is an imperative, interpreted language that has built-in support for common web protocols like HTTP and FTP, and popular data types like HTML and XML. WebL's implementation language is Java, and the complete source code is available from Compaq. In addition, step **520** is preferably performed dynamically when necessary—in other words, on-the-fly in response to a particular user query—but in some applications it may be possible to scrape relatively stable (unchanging) web sites of likely interest in advance and to cache the resulting template information.

It will be apparent, in light of the above teachings, that preferred embodiments of the present invention can provide a spoken natural language interface atop an existing, non-voice data navigation system, whereby users can interact by means of intuitive natural language input not strictly conforming to the linear browsing architecture or other artifacts of an existing menu/text/click navigation system. For example, users of an appropriate embodiment of the present invention for a video-on-demand application can directly speak the natural request: "Show me the movie 'Unforgiven'"—instead of walking step-by-step through a typically linear sequence of genre/title/actor/director menus, scrolling and selecting from potentially long lists on each menu, or instead of being forced to use an alphanumeric keyboard that cannot be as comfortable to hold or use as a lightweight remote control. Similarly, users of an appropriate embodiment of the present invention for a web-surfing application in accordance with the process shown in FIG. **5** can directly speak the natural request: "Show me a one-month price chart for Microsoft stock"—instead of potentially having to navigate to an appropriate web site, search for the right ticker symbol, enter/select the symbol, and specify display of the desired one-month price chart, each of those steps potentially involving manual navigation and data entry to one or more different interaction screens. (Note that these examples are offered to illustrate some of the potential benefits offered by appropriate embodiments of the present invention, and not to limit the scope of the invention in any respect.)

c. Error Correction

Several problems can arise when attempting to perform searches based on spoken natural language input. As indicated at decision step **407** in the process of FIG. **4**, certain deficiencies may be identified during the process of query construction, before search of the data source is even attempted. For example, the user's request may fail to specify enough information in order to construct a navigation query that is specific enough to obtain a satisfactory search result. For example, a user might orally request "what's the weather? " whereas the national online data source identified in step **405** and scraped in step **520** might require specifying a particular city.

Additionally, certain deficiencies and problems may arise following the navigational search of the data source at step **408**, as indicated at decision step **409** in FIG. **4**. For example, with reference to a video-on-demand application, a user may wish to see the movie "Unforgiven",but perhaps the user can't recall name of the film, but knows it was

directed by and starred actor Clint Eastwood. A typical video-on-demand database might indeed be expected to allow queries specifying the name of a leading actor and/or director, but in the case of this query—as in many cases—that will not be enough to narrow the search to a single film, and additional user input in some form is required.

In the event that one or more deficiencies in the user's spoken request, as processed, result in the problems described, either at step **407** or **409**, some form of error handling is in order. A straightforward, crude technique might be for the system to respond simply "input not understood/insufficient, please try again." However, that approach will likely result in frustrated users, and is not optimal or even acceptable for most applications. Instead, a preferred technique in accordance with the present invention handles such errors and deficiencies in user input at step **412**, whether detected at step **407** or step **409**, by soliciting additional input from the user in a manner taking advantage of the partial construction already performed and via user interface modalities in addition to spoken natural language ("multi-modality"). This supplemental interaction is preferably conducted through client display device **112** (**202**, in the embodiment of FIG. **2**), and may include textual, graphical, audio and/or video media. Further details and examples are provided below. Query refinement logic **340** preferably carries out step **412**. The additional input received from the user is fed into and augments interpreting step **404**, and query construction step **406** is likewise repeated with the benefit of the augmented interpretation. These operations, and subsequent navigation step **408**, are preferably repeated until no remaining problems or deficiencies are identified at decision points **407** or **409**. Further details and examples for this query refinement process are provided immediately below.

Consider again the example in which the user of a video-on-demand application wishes to see "Unforgiven" but can only recall that it was directed by and starred Clint Eastwood. First, it bears noting that using a prior art navigational interface, such as a conventional menu interface, will likely be relatively tedious in this case. The user can proceed through a sequence of menus, such as Genre (select "western"), Title (skip), Actor ("Clint Eastwood"), and Director ("Clint Eastwood"). In each case—especially for the last two items—the user would typically scroll and select from fairly long lists in order to enter his or her desired name, or perhaps use a relatively couch-unfriendly keypad to manually type the actor's name twice.

Using a preferred embodiment of the present invention, the user instead speaks aloud, holding remote control microphone **102**, "I want to see that movie starring and directed by Clint Eastwood. Can't remember the title." At step **402** the voice data is received. At step **404** the voice data is interpreted. At step **405** an appropriate online data source is selected (or perhaps the system is directly connected to a proprietary video-on-demand provider). At step **406** a query is automatically constructed by the query construction logic **330** specifying "Clint Eastwood" in both the actor and director fields. Step **407** detects no obvious problems, and so the query is electronically submitted and the data source is navigated at step **408**, yielding a list of several records satisfying the query (e.g., "Unforgiven","True Crime", "Absolute Power",etc.). Step **409** detects that additional user input is needed to further refine the query in order to select a particular film for viewing.

At that point, in step **412** query refinement logic **340** might preferably generate a display for client display device **112** showing the (relatively short) list of film titles that satisfy the user's stated constraints. The user can then preferably use a relatively convenient input modality, such as buttons on the remote control, to select the desired title from the menu. In a further preferred embodiment, the first title on the list is highlighted by default, so that the user can simply press an "OK" button to choose that selection. In a further preferred feature, the user can mix input modalities by speaking a response like "I want number one on the list." Alternatively, the user can preferably say, "Let's see Unforgiven," having now been reminded of the title by the menu display.

Utilizing the user's supplemental input, request processing logic **300** iterates again through steps **404** and **406**, this time constructing a fully-specified query that specifically requests the Eastwood film "Unforgiven." Step **408** navigates the data source using that query and retrieves the desired film, which is then electronically transmitted in step **410** from network server **108** to client display device **112** via communications network **106**.

Now consider again the example in which the user of a web surfing application wants to know his or her local weather, and simply asks, "what's the weather?" At step **402** the voice data is received. At step **404** the voice data is interpreted. At step **405** an online web site providing current weather information for major cities around the world is selected. At step **406** and sub-step **520**, the online site is scraped using a WebL-style tool to extract an input template for interacting with the site. At sub-step **522**, query construction logic **330** attempts to construct a navigation query by instantiating the input template, but determines (quite rightly) that a required field—name of city—cannot be determined from the user's spoken request as interpreted in step **404**. Step **407** detects this deficiency, and in step **412** query refinement logic **340** preferably generates output for client display device **112** soliciting the necessary supplemental input. In a preferred embodiment, the output might display the name of the city where the user is located highlighted by default. The user can then simply press an "OK" button—or perhaps mix modalities by saying "yes, exactly"—to choose that selection. A preferred embodiment would further display an alphabetical scrollable menu listing other major cities, and/or invite the user to speak or select the name of the desired city.

Here again, utilizing the user's supplemental input, request processing logic **300** iterates through steps **404** and **406**. This time, in performing sub-step **520**, a cached version of the input template already scraped in the previous iteration might preferably be retrieved. In sub-step **522**, query construction logic **330** succeeds this time in instantiating the input template and constructing an effective query, since the desired city has now been clarified. Step **408** navigates the data source using that query and retrieves the desired weather information, which is then electronically transmitted in step **410** from network server **108** to client display device **112** via communications network **106**.

It is worth noting that in some instances, there may be details that are not explicitly provided by the user, but that query construction logic **330** or query refinement logic **340** may preferably deduce on their own through reasonable assumptions, rather than requiring the use to provide explicit clarification. For example, in the example previously described regarding a request for a weather report, in some applications it might be preferable for the system to simply assume that the user means a weather report for his or her home area and to retrieve that information, if the cost of doing so is not significantly greater than the cost of asking the user to clarify the query. Making such an assumption

might be even more strongly justified in a preferred embodiment, as described earlier, where user histories are tracked, and where such history indicates that a particular user or group of users typically expect local information when asking for a weather forecast. At any rate, in the event such an assumption is made, if the user actually intended to request the weather for a different city, the user would then need to ask his or her question again. It will be apparent to practitioners, in light of the above teachings, that the choice of whether to program query construction logic **330** and query refinement logic **340** to make make particular assumptions will typically involve trade-offs involving user conveience that can be assessed in the context of specific applications.

3. Open Agent Architecture™ (OAA®)

Open Agent Architecture™(OAA®) is a software platform, developed by the assignee of the present invention, that enables effective, dynamic collaboration among communities of distributed electronic agents. OAA is described in greater detail in co-pending U.S. patent application Ser. No. 09/225,198, which has been incorporated herein by reference. Very briefly, the functionality of each client agent is made available to the agent community through registration of the client agent's capabilities with a facilitator. A software "wrapper" essentially surrounds the underlying application program performing the services offered by each client. The common infrastructure for constructing agents is preferably supplied by an agent library. The agent library is preferably accessible in the runtime environment of several different programming languages. The agent library preferably minimizes the effort required to construct a new system and maximizes the ease with which legacy systems can be "wrapped" and made compatible with the agent-based architecture of the present invention. When invoked, a client agent makes a connection to a facilitator, which is known as its parent facilitator. Upon connection, an agent registers with its parent facilitator a specification of the capabilities and services it can provide, using a highlevel, declarative Interagent Communication Language ("ICL") to express those capabilities. Tasks are presented to the facilitator in the form of ICL goal expressions. When a facilitator determines that the registered capabilities of one of its client agents will help satisfy a current goal or sub-goal thereof, the facilitator delegates that sub-goal to the client agent in the form of an ICL request. The client agent processes the request and returns answers or information to the facilitator. In processing a request, the client agent can use ICL to request services of other agents, or utilize other infrastructure services for collaborative work. The facilitator coordinates and integrates the results received from different client agents on various sub-goals, in order to satisfy the overall goal.

OAA provides a useful software platform for building systems that integrate spoken natural language as well as other user input modalities. For example, see the above-referenced co-pending patent application, especially FIG. **13** and the corresponding discussion of a "multi-modal maps" application, and FIG. **12** and the corresponding discussion of a "unified messaging" application. Another example is the InfoWiz interactive information kiosk developed by the assignee and described in the document entitled "InfoWiz: An Animated Voice Interactive Information System" available online at http://www.ai.sri.com/~oaa/applications.html. A copy of the InfoWhiz document is provided in an Information Disclosure Statement submitted herewith and incorporated herein by this reference. A further example is the "CommandTalk" application developed by the assignee for the U.S. military, as described online at http://www.ai.sri.com/~lesaf/commandtalk.html and in the following publications, copies of which are provided in an Information Disclosure Statement submitted herewith and incorporated herein by this reference:

"CommandTalk: A Spoken-Language Interface for Battlefield Simulations", 1997, by Robert Moore, John Dowding, Harry Bratt, J. Mark Gawron, Yonael Gorfu and Adam Cheyer, in "Proceedings of the Fifth Conference on Applied Natural Language Processing", Washington, D.C., pp. 1–7, Association for Computational Linguistics

"The CommandTalk Spoken Dialogue System", 1999, by Amanda Stent, John Dowding, Jean Mark Gawron, Elizabeth Owen Bratt and Robert Moore, in "Proceedings of the Thirty-Seventh Annual Meeting of the ACL", pp. 183–190, University of Maryland, College Park, Md., Association for Computational Linguistics

"Interpreting Language in Context in CommandTalk", 1999, by John Dowding and Elizabeth Owen Bratt and Sharon Goldwater, in "Communicative Agents: The Use of Natural Language in Embodied Systems", pp. 63–67, Association for Computing Machinery (ACM) Special Interest Group on Artificial Intelligence (SIGART), Seattle, Wash.

For some applications and systems, OAA can provide an advantageous platform for constructing embodiments of the present invention. For example, a representative application is now briefly presented, with reference to FIG. **6**. If the statement "show me movies starring John Wayne" is spoken into the voice input device, the voice data for this request will be sent by UI agent **650** to facilitator **600**, which in turn will ask natural language (NL) agent **620** and speech recognition agent **610** to interpret the query and return the interpretation in ICL format. The resulting ICL goal expression is then routed by the facilitator to appropriate agents—in this case, video-on-demand database agent **640**—to execute the request. Video database agent **640** preferably includes or is coupled to an appropriate embodiment of query construction logic **330** and query refinement logic **340**, and may also issue ICL requests to facilitator **600** for additional assistance—e.g., display of menus and capture of additional user input in the event that query refinement is needed—and facilitator **600** will delegate such requests to appropriate client agents in the community. When the desired video content is ultimately retrieved by video database agent **640**, UI agent **650** is invoked by facilitator **600** to display the movie.

Other spoken user requests, such as a request for the current weather in New York City or for a stock quote, would eventually lead facilitator to invoke web database agent **630** to access the desired information from an appropriate Internet site. Here again, web database agent **630** preferably includes or is coupled to an appropriate embodiment of query construction logic **330** and query refinement logic **340**, including a scraping utility such as WebL. Other spoken requests, such as a request to view recent emails or access voice mail, would lead the facilitator to invoke the appropriate email agent **660** and/or telephone agent **680**. A request to record a televised program of interest might lead facilitator **600** to invoke web database agent **630** to return televised program schedule information, and then invoke VCR controller agent **680** to program the associated VCR unit to record the desired television program at the scheduled time.

Control and connectivity embracing additional electronic home appliances (e.g., microwave oven, home surveillance system, etc.) can be integrated in comparable fashion.

Indeed, an advantage of OAA-based embodiments of the present invention, that will be apparent to practitioners in light of the above teachings and in light of the teachings disclosed in the cited co-pending patent applications, is the relative ease and flexibility with which additional service agents can be plugged into the existing platform, immediately enabling the facilitator to respond dynamically to spoken natural language requests for the corresponding services.

4. Further Embodiments and Equivalents

While the present invention has been described in terms of several preferred embodiments, there are many alterations, permutations, and equivalents that may fall within the scope of this invention. It should also be noted that there are many alternative ways of implementing the methods and apparatuses of the present invention. It is therefore intended that the following appended claims be interpreted as including all such alterations, permutations, and equivalents as fall within the true spirit and scope of the present invention.

What is claimed is:

1. A method for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:

(a) receiving a spoken request for desired information from a user;

(b) rendering an interpretation of the spoken request;

(c) constructing a navigation query based upon the interpretation;

(d) routing the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source; and

(e) invoking a user interface agent for outputting the selected portion of the electronic data source to the user, wherein a facilitator manages data flow among multiple agents and maintains a registration of each of said agents' capabilities.

2. The method of claim **1**, wherein an agent renders the interpretation of the spoken request.

3. The method of claim **1**, wherein the step of rendering the interpretation of the spoken request is performed by a speech recognition agent and a parsing agent.

4. The method of claim **1**, further comprising the steps of soliciting additional input from the user, including user interaction in a modality different than the original request; and refining the navigation query, based upon the additional input; wherein the at least one agent uses the refined navigation query to select a portion of the electronic data source.

5. The method of claim **4**, wherein agents are utilized for performing the steps of soliciting additional input from the user and refining the navigation query.

6. The method of claim **1**, wherein the electronic data source is a web page, wherein the at least one agent scrapes the web page for selecting a portion of the web page.

7. A computer program embodied on a computer readable medium for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:

(a) a code segment that receives a spoken request for desired information from a user;

(b) a code segment that renders an interpretation of the spoken request;

(c) a code segment that constructs a navigation query based upon the interpretation;

(d) a code segment that routes the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source; and

(e) a code segment that invokes a user interface agent for outputting the selected portion of the electronic data source to the user, wherein a facilitator manages data flow among multiple agents and maintains a registration of each of said agents' capabilities.

8. The computer program of claim **7**, wherein the code segment that renders the interpretation of the spoken request is executed by an agent.

9. The computer program of claim **7**, wherein a speech recognition agent and a parsing agent execute the code segment that renders the interpretation of the spoken request.

10. The computer program of claim **7**, further comprising a code segment that solicits additional input from the user, including user interaction in a modality different than the original request; and a code segment that refines the navigation query, based upon the additional input; wherein the at least one agent uses the refined navigation query to select a portion of the electronic data source.

11. The computer program of claim **10**, wherein a solicitor agent executes the code segment that solicit the additional input from the user and a refining agent executes the code segment that refines the navigation query.

12. The computer program of claim **7**, wherein the electronic data source is a web page, wherein the at least one agent scrapes the web page for selecting a portion of the web page.

13. A system for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:

(a) a client device, operable to receive a spoken request for desired information from a user; (b) spoken language processing logic, operable to render an interpretation of the spoken request;

(c) query construction logic, operable to construct a navigation query based upon the interpretation;

(d) routing logic, operable to route the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source; and

(e) invoking logic, operable to invoke a user interface agent for outputting the selected portion of the electronic data source to the user, Wherein a facilitator manages data flow among multiple agents and maintains a registration of each of said agents' capabilities.

14. The system of claim **13**, wherein the query construction logic that renders the interpretation of the spoken request is executed by an agent.

15. The system of claim **13**, wherein a speech recognition agent and a parsing agent execute the spoken language processing logic that renders the interpretation of the spoken request.

16. The system of claim **13**, further comprising user interaction logic operable to solicit additional input from the user, including user interaction in a modality different than the original request; and query refining logic operable to refine the navigation query, based upon the additional input; wherein the at least one agent uses the refined navigation query to select a portion of the electronic data source.

17. The system of claim **16**, wherein a solicitor agent executes the user interaction logic and a refining agent executes the query refinement logic.

18. The system of in claim **13**, wherein the electronic data source is a web page, wherein the at least one agent scrapes the web page for selecting a portion of the web page.

* * * * *

US Patent & Trademark Office

US 6,523,061
USPTO Transaction Information*

SEQ.δ	DATE	DESCRIPTION
1	18 Jan 2017	File Marked Found
2	29 Mar 2016	File Marked Found
3	28 Mar 2016	File Marked Found
4	16 Aug 2006	ENTITY STATUS SET TO UNDISCOUNTED (INITIAL DEFAULT SETTING OR STATUS CHANGE)
5	18 Feb 2003	Recordation of Patent Grant Mailed
6	30 Jan 2003	Issue Notification Mailed
7	18 Feb 2003	Patent Issue Date Used in PTA Calculation
8	21 Jan 2003	Receipt into Pubs
9	13 Jan 2003	Application Is Considered Ready for Issue
10	26 Dec 2002	Issue Fee Payment Verified
11	26 Dec 2002	Workflow - Drawings Finished
12	26 Dec 2002	Workflow - Drawings Matched with File at Contractor
13	08 Jan 2003	Receipt into Pubs
14	07 Jan 2003	Mail Miscellaneous Communication to Applicant
15	06 Jan 2003	Miscellaneous Communication to Applicant - No Action Count
16	26 Dec 2002	Workflow - Drawings Received at Contractor
17	31 Dec 2002	Workflow - Customer Service Request - Finish
18	26 Dec 2002	Workflow - Drawings Sent to Contractor
19	31 Dec 2002	Workflow - Customer Service Request - Begin
20	26 Dec 2002	Issue Fee Payment Received
21	27 Dec 2002	Receipt into Pubs
22	23 Dec 2002	Workflow - Customer Service Request - Finish
23	23 Dec 2002	Workflow - Customer Service Request - Begin
24	29 Oct 2002	Receipt into Pubs
25	23 Sep 2002	Workflow - File Sent to Contractor
26	23 Sep 2002	Receipt into Pubs
27	20 Sep 2002	Dispatch to Publications
28	23 Sep 2002	Mail Notice of Allowance
29	20 Sep 2002	Notice of Allowance Data Verification Completed
30	17 Jul 2002	Date Forwarded to Examiner
31	16 Jul 2002	Response after Non-Final Action
32	16 Jul 2002	Request for Extension of Time - Granted
33	23 May 2002	Examiner Interview Summary Record (PTOL - 413)
34	13 Feb 2002	Mail Non-Final Rejection
35	08 Feb 2002	Non-Final Rejection
36	08 Feb 2002	Case Docketed to Examiner in GAU
37	29 Nov 2001	Date Forwarded to Examiner
38	27 Nov 2001	Response after Non-Final Action
39	12 Nov 2001	Case Docketed to Examiner in GAU
40	12 Oct 2001	Change in Power of Attorney (May Include Associate POA)
41	12 Oct 2001	Correspondence Address Change
42	12 Oct 2001	Change in Power of Attorney (May Include Associate POA)
43	27 Aug 2001	Mail Non-Final Rejection
44	24 Aug 2001	Non-Final Rejection
45	22 May 2001	Correspondence Address Change
46	13 Feb 2001	Case Docketed to Examiner in GAU

* Document generated on 01/24/2017 by PATENTEC from official USPTO records, external to this file. Information deemed accurate, but not Certified.

δ Transaction Sequence Number (SEQ.) is unrelated to Paper Number in File Table of contents.

US Patent & Trademark Office

US 6,523,061

USPTO Transaction Information*

SEQ.[δ]	DATE	DESCRIPTION
47	30 Jun 2000	Preliminary Amendment
48	02 Dec 2000	Application Dispatched from OIPE
49	02 Dec 2000	Application Is Now Complete
50	22 Aug 2000	Notice Mailed--Application Incomplete--Filing Date Assigned
51	21 Aug 2000	Correspondence Address Change
52	21 Aug 2000	Correspondence Address Change
53	17 Jul 2000	IFW Scan &amp; PACR Auto Security Review
54	30 Jun 2000	Initial Exam Team nn

* Document generated on 01/24/2017 by PATENTEC from official USPTO records, external to this file. Information deemed accurate, but not Certified.

δ Transaction Sequence Number (SEQ.) is unrelated to Paper Number in File Table of contents.

US Patent & Trademark Office

US 6,523,061
Assignment History*

Assignment: 1 / 1

Reel / Frame:	039857/0097	**Recorded:**	09/26/2016	**Pages in document:**	5
Conveyance:	ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS).				
Assignor:	Sri International			**Exec. Dt:**	05/20/2016
Assignee:	IPA TECHNOLOGIES INC. 600 ANTON BLVD. SUITE 1350 COSTA MESA, CALIFORNIA 92626				
Correspondent:	IPA TECHNOLOGIES INC. 600 ANTON BLVD. SUITE 1350 COSTA MESA, CA 92626				

* Document generated on 01/24/2017 by PATENTEC from official USPTO records, external to this file. Information deemed accurate, but not Certified.

United States Patent and Trademark Office
Office of the Commissioner for Patents

SYSTEM, METHOD, AND ARTICLE OF MANUFACTURE FOR AGENT-BASED NAVIGATION IN A SPEECH-BASED DATA NAVIGATION SYSTEM

PATENT #	APPLICATION #	FILING DATE	ISSUE DATE
6523061	09607672	06/30/2000	02/18/2003

Payment Window Status

No maintenance fees are due.

WINDOW	STATUS	FEES
11.5 Year	Closed	Paid

Window	First Day to Pay	Surcharge Starts	Last Day to Pay	Status	Fees
3.5 Year	02/18/2006	08/19/2006	02/20/2007	Closed	Paid
7.5 Year	02/18/2010	08/19/2010	02/18/2011	Closed	Paid
11.5 Year	02/18/2014	08/19/2014	02/18/2015	Closed	Paid

Patent Holder Information

Customer #	
Entity Status	UNDISCOUNTED
Phone Number	4085055100

Address

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY, NJ 07702
UNITED STATES

Generated: 01/24/2017 09:41:53

ISSUE SLIP ST

JC836 U.S. PTO
09/607672
06/30/00

ISSUE CLASSIFICATION	
Class	Subclass
709	202

PATENT NUMBER
6523061
6523061

U.S. UTILITY Patent Application

TC-2
TC-459 C

O.I.P.E.	PATENT DATE
① SCANNED AC NH Q.A. A.G.	FEB 18 2003

APPLICATION NO.	CONT/PRIOR	CLASS	SUBCLASS	ART UNIT	EXAMINER
09/607672	D	709	202	~~2154~~ 2155	~~T. Lee~~ Nguyen, T

APPLICANTS
Christine Halversen
Luc Julia
Dimitris Voutsas
Adam Cheyer

TITLE
System, method, and article of manufacture for agent-based navigation in a speech-based data navigation system

PTO-2040
12/99

ISSUING CLASSIFICATION

ORIGINAL		CROSS REFERENCE(S)						
CLASS	SUBCLASS	CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)					
709	202	709	202	217	219			
INTERNATIONAL CLASSIFICATION		379	88.01	88.02	88.22			
G06F	15/16	370	331					
		704	270	275				
				□ Continued on Issue Slip Inside File Jacket				

□ TERMINAL DISCLAIMER	DRAWINGS			CLAIMS ALLOWED	
	Sheets Drwg.	Figs. Drwg.	Print Fig.	Total Claims	Print Claim for O.G.
	7	18	4	18	1

□ The term of this patent subsequent to ____________ (date) has been disclaimed.

THU HA NGUYEN 09/18/02
(Assistant Examiner) (Date)

NOTICE OF ALLOWANCE MAILED
9-23-02

□ The term of this patent shall not extend beyond the expiration date of U.S Patent. No. ____________

AYAZ SHEIKH
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100
(Primary Examiner) 9/18/02 (Date)

ISSUE FEE

Amount Due	Date Paid
$ 640.00	12/26/02

□ The terminal ____months of this patent have been disclaimed.

L. Johnson
(Legal Instruments Examiner) (Date)

ISSUE BATCH NUMBER

WARNING:
The information disclosed herein may be restricted. Unauthorized disclosure may be prohibited by the United States Code Title 35, Sections 122, 181 and 368. Possession outside the U.S. Patent & Trademark Office is restricted to authorized employees and contractors only.

Form PTO-436A
(Rev. 6/99)

FILED WITH: □ DISK (CRF) □ FICHE □ CD-ROM
(Attached in pocket on right inside flap)

ISSUE FEE
IN FILE

(FACE)

PATENT APPLICATION

09607672

INITIALS

CONTENTS

		Date Received (Incl. C. of M.) or Date Mailed			Date Received (Incl. C. of M.) or Date Mailed
1.	Application ______ papers.		42.		
2.	LTR-RE Dec Missing	8/22/00	43.		
3.	Dec & Surcharge	11-2-0	44.		
4.	I-OT	11-2-0	45.		
5.	2-13-01 Pre Amdt A	6-30-00	46.		
6.	I.D.S. w/References	4-30-01	47.		
7.	Associate P/A	5-14-01	48.		
8.	Change of address	5-14-01	49.		
9.	8-27-01 Rejection 3months	8/27/01	50.		
10.	Revocation + Power of Attorney	10-9-01	51.		
11.	notice of acceptance	10-15-01	52.		
12.	Reconsideration	11/27/01	53.		
13.	2-13-02 Rejection 3months	2-13-02	54.		
14.	interview summary	5-23-02	55.		
15.	Ext. of Time (2 Mos)	7-16-02	56.		
16.	Amdt B	7-16-02	57.		
17.	notice of allow.	9-23-02	58.		
18.	Suppl. notice of allow.	1-7-03	59.		
19.	1-13-03 Formal Drawings (7 sheets)	12-26-02	60.		
20.			61.		
21.			62.		
22.			63.		
23.			64.		
24.			65.		
25.			66.		
26.			67.		
27.			68.		
28.			69.		
29.			70.		
30.			71.		
31.			72.		
32.			73.		
33.			74.		
34.			75.		
35.			76.		
36.			77.		
37.			78.		
38.			79.		
39.			80.		
40.			81.		
41.			82.		

(LEFT OUTSIDE)

ny
8/23

ISSUE SLIP STAPLE AREA (for additional cross references)

POSITION	INITIALS	ID NO.	DATE
FEE DETERMINATION	vr G		7/9/00
O.I.P.E. CLASSIFIER			
FORMALITY REVIEW	AL		8/21/00
RESPONSE FORMALITY REVIEW		64149	12-20

INDEX OF CLAIMS

✓	Rejected	N	Non-elected
=	Allowed	I	Interference
— (Through numeral)...	Canceled	A	Appeal
÷	Restricted	O	Objected

Claim		Date									
Final	Original										
	1										
	2										
	3										
	4										
	5										
	6										
	7										
	8										
	9										
	10										
	11										
	12										
	13										
	14										
	15										
	16										
	17										
	18										
	19										
	20										
	21										
	22										
	23										
	24										
	25										
	26										
	27										
	28										
	29										
	30										
	31										
	32										
	33										
	34										
	35										
	36										
	37										
	38										
	39										
	40										
	41										
	42										
	43										
	44										
	45										
	46										
	47										
	48										
	49										
	50										

Claim		Date									
Final	Original		02/06/02	09/18/02							
	51										
	52										
	53										
	54										
	55										
1	56	✓	✓	=	I						
2	57	✓		=	I						
	~~58~~	✓			I						
3	59	✓		=							
4	60	✓		=							
5	61	✓		=							
6	62	✓		=							
7	63	✓		=							
8	64	✓		=							
	~~65~~	✓									
9	66	✓		=							
10	67	✓		=							
11	68	✓		=							
12	69	✓		=							
13	70	✓		=							
14	71	✓		=							
	~~72~~	✓									
15	73	✓		=							
16	74	✓		=							
17	75	✓		=							
18	76	✓	✓	=							
	77										
	78										
	79										
	80										
	81										
	82										
	83										
	84										
	85										
	86										
	87										
	88										
	89										
	90										
	91										
	92										
	93										
	94										
	95										
	96										
	97										
	98										
	99										
	100										

Claim		Date									
Final	Original										
	101										
	102										
	103										
	104										
	105										
	106										
	107										
	108										
	109										
	110										
	111										
	112										
	113										
	114										
	115										
	116										
	117										
	118										
	119										
	120										
	121										
	122										
	123										
	124										
	125										
	126										
	127										
	128										
	129										
	130										
	131										
	132										
	133										
	134										
	135										
	136										
	137										
	138										
	139										
	140										
	141										
	142										
	143										
	144										
	145										
	146										
	147										
	148										
	149										
	150										

If more than 150 claims or 10 actions
staple additional sheet here

(LEFT INSIDE)

SEARCHED

Class	Sub.	Date	Exmr.
395 370 709	200.12 331 202 217 219 227	7/30/01	Tang
379 704 382	88.01 88.17 270 275 313	02/06/02	Thn
Update search		09/18/02	Thn
379 704 707	88.22 88.02 90.01 900 270 203		

SEARCH NOTES
(INCLUDING SEARCH STRATEGY)

	Date	Exmr.
East West Search	7/30/01	Tang
East search results attached Consult with SPE Ayaz Sheikh	02/06/02	Thn
East search Results attached Consults with David wiley	05/23/02	Thn
Consults with David Wiley	09/18/02	Thn

INTERFERENCE SEARCHED

Class	Sub.	Date	Exmr.
709 379	202 217 219 88.01 88.02	09/18/02	Thn

(RIGHT OUTSIDE)

UNITED STATES PATENT AND TRADEMARK OFFICE

COMMISSIONER FOR PATENTS
UNITED STATES PATENT AND TRADEMARK OFFICE
WASHINGTON, D.C. 20231
www.uspto.gov

Bib Data Sheet

SERIAL NUMBER	FILING DATE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
09/607,672	06/30/2000 RULE –	709	~~2758~~ 2155	SRI1P037C

APPLICANTS

Christine Halversen, San Jose, CA ;
Luc Julia, Menlo Park, CA ;
Dimitris Voutsas, Thessaloniki, GREECE;
Adam Cheyer, Palo Alto, CA ;

** **CONTINUING DATA** *************************

THIS APPLICATION IS A CON OF 09/524,095 03/13/2000
WHICH IS A CIP OF 09/225,198 01/05/1999
WHICH CLAIMS BENEFIT OF 60/124,718 03/17/1999
WHICH CLAIMS BENEFIT OF 60/124,720 03/17/1999
WHICH CLAIMS BENEFIT OF 60/124,719 03/17/1999

** **FOREIGN APPLICATIONS** ********************

IF REQUIRED, FOREIGN FILING LICENSE GRANTED ** 08/21/2000

** SMALL ENTITY **

	STATE OR COUNTRY	SHEETS DRAWING	TOTAL CLAIMS	INDEPENDENT CLAIMS
Foreign Priority claimed □ yes ☒ no 35 USC 119 (a-d) conditions met □ yes ☒ no □ Met after Allowance Verified and Acknowledged ______ Examiner's Signature TL Initials	CA	7	21	3

ADDRESS

24277

TITLE

System, method, and article of manufacture for agent-based navigation in a speech-based data navigation system

FILING FEE RECEIVED		
FILING FEE RECEIVED 419	FEES: Authority has been given in Paper No. ________ to charge/credit DEPOSIT ACCOUNT No. ________ for following:	□ All Fees □ 1.16 Fees (Filing) □ 1.17 Fees (Processing Ext. of time) □ 1.18 Fees (Issue) □ Other ________ □ Credit

UNITED STATES PATENT AND TRADEMARK OFFICE

COMMISSIONER FOR PATENTS
UNITED STATES PATENT AND TRADEMARK OFFICE
WASHINGTON, D.C. 20231
www.uspto.gov

Bib Data Sheet

CONFIRMATION NO. 1291

SERIAL NUMBER	FILING DATE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
09/607,672	06/30/2000 RULE	709	2155	SRI1P037C

APPLICANTS

Christine Halversen, San Jose, CA;
Luc Julia, Menlo Park, CA;
Dimitris Voutsas, Thessaloniki, GREECE;
Adam Cheyer, Palo Alto, CA;

** **CONTINUING DATA** *************************

THIS APPLICATION IS A CON OF 09/524,095 03/13/2000
WHICH IS A CIP OF 09/225,198 01/05/1999
AND CLAIMS BENEFIT OF 60/124,718 03/17/1999
AND CLAIMS BENEFIT OF 60/124,720 03/17/1999
AND CLAIMS BENEFIT OF 60/124,719 03/17/1999

** **FOREIGN APPLICATIONS** ********************

IF REQUIRED, FOREIGN FILING LICENSE GRANTED ** SMALL ENTITY **
** 08/21/2000

		STATE OR COUNTRY	SHEETS DRAWING	TOTAL CLAIMS	INDEPENDENT CLAIMS
Foreign Priority claimed 35 USC 119 (a-d) conditions met Verified and Acknowledged	□ yes □ no □ yes □ no □ Met after Allowance Examiner's Signature Initials	CA	7	21	3

ADDRESS

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY , NJ 07702

TITLE

System, method, and article of manufacture for agent-based navigation in a speech-based data navigation system

FILING FEE RECEIVED 419	FEES: Authority has been given in Paper No. _________ to charge/credit DEPOSIT ACCOUNT No. _________ for following:	□ All Fees □ 1.16 Fees (Filing) □ 1.17 Fees (Processing Ext. of time) □ 1.18 Fees (Issue) □ Other _______________ □ Credit

Sub A1

~~NAVIGATING NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN NATURAL LANGUAGE INPUT WITH MULTIMODAL ERROR FEEDBACK~~

A1

BACKGROUND OF THE INVENTION

Sub A3

~~This is~~ A a Continuation In Part of co-pending U.S. Patent Application No. 09/225,198, filed January 5, 1999, Provisional U.S. Patent Application No. 60/124,718, filed March 17, 1999, Provisional U.S. Patent Application No. 60/124,720, filed March 17, 1999, and Provisional U.S. Patent Application No. 60/124,719, filed March 17, 1999, from which applications priority is claimed and these application are incorporated herein by reference.

The present invention relates generally to the navigation of electronic data by means of spoken natural language requests, and to feedback mechanisms and methods for resolving the errors and ambiguities that may be associated with such requests.

As global electronic connectivity continues to grow, and the universe of electronic data potentially available to users continues to expand, there is a growing need for information navigation technology that allows relatively naïve users to navigate and access desired data by means of natural language input. In many of the most important markets -- including the home entertainment arena, as well as mobile computing -- spoken natural language input is highly desirable, if not ideal. As just one example, the proliferation of high-bandwidth communications infrastructure for the home entertainment market (cable, satellite, broadband) enables delivery of movies-on-demand and other interactive multimedia content to the consumer's home television set. For users to take full advantage of this content stream ultimately requires interactive navigation of content databases in a manner that is too complex for user-friendly selection by means of a traditional remote-control clicker. Allowing spoken natural language requests as the input modality for rapidly searching and accessing desired content is an important objective for a successful consumer entertainment product in a context offering a dizzying range of database content choices. As further examples, this same need to drive navigation of (and transaction with) relatively complex data warehouses using spoken natural language requests applies equally to surfing the Internet/Web or other networks for general information, multimedia content, or e-commerce transactions.

In general, the existing navigational systems for browsing electronic databases and data warehouses (search engines, menus, etc.), have been designed without navigation via spoken natural language as a specific goal. So today's world is full of existing electronic data navigation systems that do not assume browsing via natural spoken commands, but rather assume text and mouse-click inputs (or in the case of TV remote controls, even less). Simply recognizing voice commands within an extremely limited vocabulary and grammar -- the spoken equivalent of button/click input (e.g., speaking "channel 5" selects TV channel 5) -- is really not sufficient by itself to satisfy the objectives described above. In order to deliver a true "win" for users, the voice-driven front-end must accept spoken natural language input in a manner that is intuitive to users. For example, the front-end should not require learning a highly specialized command language or format. More fundamentally, the front-end must allow users to speak directly in terms of what the user ultimately wants -- e.g., "I'd like to see a Western film directed by Clint Eastwood" -- as opposed to speaking in terms of arbitrary navigation structures (e.g., hierarchical layers of menus, commands, etc.) that are essentially artifacts reflecting constraints of the pre-existing text/click navigation system. At the same time, the front-end must recognize and accommodate the reality that a stream of naïve spoken natural language input will, over time, typically present a variety of errors and/or ambiguities: e.g., garbled/unrecognized words (did the user say "Eastwood" or "Easter"?) and under-constrained requests ("Show me the Clint Eastwood movie"). An approach is needed for handling and resolving such errors and ambiguities in a rapid, user-friendly, non-frustrating manner.

What is needed is a methodology and apparatus for rapidly constructing a voice-driven front-end atop an existing, non-voice data navigation system, whereby users can interact by means of intuitive natural language input not strictly conforming to the step-by-step browsing architecture of the existing navigation system, and wherein any errors or ambiguities in user input are rapidly and conveniently resolved. The solution to this need should be compatible with the constraints of a multi-user, distributed environment such as the Internet/Web or a proprietary high-bandwidth content delivery network; a solution contemplating one-at-a-time user interactions at a single location is insufficient, for example.

SUMMARY OF THE INVENTION

Sub A4

A4

The present invention addresses the above needs by providing a system, method, and article of manufacture for navigating network-based electronic data sources in response to spoken NL input requests. When a spoken natural language input request is received from a user, it is interpreted, such as by using a speech recognition engine to extract speech data from acoustic voice signals, and using a natural language parser to linguistically parse the speech data. The interpretation of the spoken natural language request can be performed on a computing device locally with the user or remotely from the user. The resulting interpretation of the request is thereupon used to automatically construct an operational navigation query to retrieve the desired information from one or more electronic network data sources, which is then transmitted to a client device of the user. If the network data source is a database, the navigation query is constructed in the format of a database query ~~language.~~

Typically, errors or ambiguities emerge in the interpretation of the spoken NL request, such that the system cannot instantiate a complete, valid navigational template. This is to be expected occasionally, and one preferred aspect of the invention is the ability to handle such errors and ambiguities in relatively graceful and user-friendly manner. Instead of simply rejecting such input and defaulting to traditional input modes or simply asking the user to try again, a preferred embodiment of the present invention seeks to converge rapidly toward instantiation of a valid navigational template by soliciting additional clarification from the user as necessary, either before or after a navigation of the data source, via multimodal input, i.e., by means of menu selection or other input modalities including and in addition to spoken natural language. This clarifying, multi-modal dialogue takes advantage of whatever partial navigational information has been gleaned from the initial interpretation of the user's spoken NL request. This clarification process continues until the system converges toward an adequately instantiated navigational template, which is in turn used to navigate the network-based data and retrieve the user's desired information. The retrieved information is transmitted across the network and presented to the user on a suitable client display device.

In a further aspect of the present invention, the construction of the navigation query includes extracting an input template for an online scripted interface to the data source and using the input template to construct the navigation query. The extraction of the input template can include dynamically scraping the online scripted interface.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention, together with further advantages thereof, may best be understood by reference to the following description taken in conjunction with the accompanying drawings in which:

Figure 1a illustrates a system providing a spoken natural language interface for network-based information navigation, in accordance with an embodiment of the present invention with server-side processing of requests;

Figure 1b illustrates another system providing a spoken natural language interface for network-based information navigation, in accordance with an embodiment of the present invention with client-side processing of requests;

Figure 2 illustrates a system providing a spoken natural language interface for network-based information navigation, in accordance with an embodiment of the present invention for a mobile computing scenario;

Figure 3 illustrates the functional logic components of a request processing module in accordance with an embodiment of the present invention;

Figure 4 illustrates a process utilizing spoken natural language for navigating an electronic database in accordance with one embodiment of the present invention;

Figure 5 illustrates a process for constructing a navigational query for accessing an online data source via an interactive, scripted (e.g., CGI) form; and

Figure 6 illustrates an embodiment of the present invention utilizing a community of distributed, collaborating electronic agents.

DETAILED DESCRIPTION OF THE INVENTION

1. System Architecture

a. Server-End Processing of Spoken Input

Figure 1a is an illustration of a data navigation system driven by spoken natural language input, in accordance with one embodiment of the present invention. As shown, a user's voice input data is captured by a voice input device 102, such as a microphone. Preferably voice input device 102 includes a button or the like that can be pressed or held-down to activate a listening mode, so that the system need not continually pay attention to, or be confused by, irrelevant background noise. In one preferred embodiment well-suited for the home entertainment setting, voice input device 102 is a portable remote control device with an integrated microphone, and the voice data is transmitted from device 102 preferably via infrared (or other wireless) link to communications box 104 (e.g., a set-top box or a similar communications device that is capable of retransmitting the raw voice data and/or processing the voice data) local to the user's environment and coupled to communications network 106. The voice data is then transmitted across network 106 to a remote server or servers 108. The voice data may preferably be transmitted in compressed digitized form, or alternatively --particularly where bandwidth constraints are significant-- in analog format (e.g., via frequency modulated transmission), in the latter case being digitized upon arrival at remote server 108.

At remote server 108, the voice data is processed by request processing logic 300 in order to understand the user's request and construct an appropriate query or request for navigation of remote data source 110, in accordance with the interpretation process exemplified in Figure 4 and Figure 5 and discussed in greater detail below. For purposes of executing this process, request processing logic 300 comprises functional modules including speech recognition engine 310, natural language (NL) parser 320, query construction logic 330, and query refinement logic 340, as shown in Figure 3. Data source 110 may comprise database(s), Internet/web site(s), or other electronic information repositories, and preferably resides on a central server or servers -- which may or may not be the same as server 108, depending on the storage

7

and bandwidth needs of the application and the resources available to the practitioner. Data source 110 may include multimedia content, such as movies or other digital video and audio content, other various forms of entertainment data, or other electronic information. The contents of data source 110 are navigated -- i.e., the contents are accessed and searched, for retrieval of the particular information desired by the user -- using the processes of Figures 4 and 5 as described in greater detail below.

Once the desired information has been retrieved from data source 110, it is electronically transmitted via network 106 to the user for viewing on client display device 112. In a preferred embodiment well-suited for the home entertainment setting, display device 112 is a television monitor or similar audiovisual entertainment device, typically in stationary position for comfortable viewing by users. In addition, in such preferred embodiment, display device 112 is coupled to or integrated with a communications box (which is preferably the same as communications box 104, but may also be a separate unit) for receiving and decoding/formatting the desired electronic information that is received across communications network 106.

Network 106 is a two-way electronic communications network and may be embodied in electronic communication infrastructure including coaxial (cable television) lines, DSL, fiber-optic cable, traditional copper wire (twisted pair), or any other type of hardwired connection. Network 106 may also include a wireless connection such as a satellite-based connection, cellular connection, or other type of wireless connection. Network 106 may be part of the Internet and may support TCP/IP communications, or may be embodied in a proprietary network, or in any other electronic communications network infrastructure, whether packet-switched or connection-oriented. A design consideration is that network 106 preferably provide suitable bandwidth depending upon the nature of the content anticipated for the desired application.

b. Client-End Processing of Spoken Input

Figure 1b is an illustration of a data navigation system driven by spoken natural language input, in accordance with a second embodiment of the present invention. Again, a user's voice input data is captured by a voice input device 102, such as a microphone. In the embodiment shown in Figure 1b, the voice data is transmitted from device 202 to requests processing logic 300, hosted on a local speech processor, for processing and interpretation. In the preferred embodiment illustrated in Figure 1b, the local speech processor is conveniently integrated as part of communications box 104, although implementation in a physically separate (but communicatively coupled) unit is also possible as will be readily apparent to those of skill in the art. The voice data is processed by the components of request processing logic 300 in order to understand the user's request and construct an appropriate query or request for navigation of remote data source 110, in accordance with the interpretation process exemplified in Figures 4 and 5 as discussed in greater detail below.

The resulting navigational query is then transmitted electronically across network 106 to data source 110, which preferably resides on a central server or servers 108. As in Figure 1a, data source 110 may comprise database(s), Internet/web site(s), or other electronic information repositories, and preferably may include multimedia content, such as movies or other digital video and audio content, other various forms of entertainment data, or other electronic information. The contents of data source 110 are then navigated -- i.e., the contents are accessed and searched, for retrieval of the particular information desired by the user -- preferably using the process of Figures 4 and 5 as described in greater detail below. Once the desired information has been retrieved from data source 110, it is electronically transmitted via network 106 to the user for viewing on client display device 112.

In one embodiment in accordance with Figure 1b and well-suited for the home entertainment setting, voice input device 102 is a portable remote control device with an integrated microphone, and the voice data is transmitted from device 102 preferably via infrared (or other wireless) link to the local speech processor. The local speech processor is coupled to communications network 106, and also preferably to client display device 112 (especially for purposes of query refinement transmissions, as discussed below in connection with Figure 4, step 412), and preferably may be integrated within or coupled to communications box 104. In addition, especially for purposes of a home entertainment application, display device 112 is preferably a television monitor or similar audiovisual entertainment device, typically in stationary position for comfortable viewing by users. In addition, in such preferred embodiment, display device 112 is coupled to a communications box (which is preferably the same as communications box 104, but may also be a physically separate unit) for receiving and decoding/formatting the desired electronic information that is received across communications network 106.

Design considerations favoring server-side processing and interpretation of spoken input requests, as exemplified in Figure 1a, include minimizing the need to distribute costly computational hardware and software to all client users in order to perform speech and language processing. Design considerations favoring client-side processing, as exemplified in Figure 1b, include minimizing the quantity of data sent upstream across the network from each client, as the speech recognition is performed before transmission across the network and only the query data and/or request needs to be sent, thus reducing the upstream bandwidth requirements.

c. Mobile Client Embodiment

A mobile computing embodiment of the present invention may be implemented by practitioners as a variation on the embodiments of either Figure 1a or Figure 1b. For example, as depicted in Figure 2, a mobile variation in accordance with the server-side processing architecture illustrated in Figure 1a may be implemented by replacing voice input device 102, communications box 104, and client display device 112, with an integrated, mobile, information appliance 202 such as a cellular telephone or wireless personal digital assistant (wireless PDA). Mobile information appliance 202 essentially performs the functions of the replaced components. Thus, mobile information appliance 202 receives spoken natural language input requests from the user in the form of voice data, and transmits that data (preferably via wireless data receiving station 204) across communications network 206 for server-side interpretation of the request, in similar fashion as described above in connection with Figure 1. Navigation of data source 210 and retrieval of desired information likewise proceeds in an analogous manner as described above. Display information transmitted electronically back to the user across network 206 is displayed for the user on the display of information appliance 202, and audio information is output through the appliance's speakers.

Practitioners will further appreciate, in light of the above teachings, that if mobile information appliance 202 is equipped with sufficient computational processing power, then a mobile variation of the client-side architecture exemplified in Figure 2 may similarly be implemented. In that case, the modules corresponding to request processing logic 300 would be embodied locally in the computational resources of mobile information appliance 202, and the logical flow of data would otherwise follow in a manner analogous to that previously described in connection with Figure 1b.

As illustrated in Figure 2, multiple users, each having their own client input device, may issue requests, simultaneously or otherwise, for navigation of data source 210. This is equally true (though not explicitly drawn) for the embodiments depicted in Figures 1a and 1b. Data source 210 (or 100), being a network accessible information resource, has typically already been constructed to support access requests from simultaneous multiple network users, as known by practitioners of ordinary skill in the art. In the case of server-side speech processing, as exemplified in Figures 1a and 2, the interpretation logic and error correction logic modules are also preferably designed and implemented to support queuing and multi-tasking of requests from multiple simultaneous network users, as will be appreciated by those of skill in the art.

It will be apparent to those skilled in the art that additional implementations, permutations and combinations of the embodiments set forth in Figures 1a, 1b, and 2 may be created without straying from the scope and spirit of the present invention. For example, practitioners will understand, in light of the above teachings and design considerations, that it is possible to divide and allocate the functional components of request processing logic 300 between client and server. For example, speech recognition -- in entirety, or perhaps just early stages such as feature extraction -- might be performed locally on the client end, perhaps to reduce bandwidth requirements, while natural language parsing and other necessary processing might be performed upstream on the server end, so that more extensive computational power need not be distributed locally to each client. In that case, corresponding portions of request processing logic 300, such as speech recognition engine 310 or portions thereof, would reside locally at the client as in Figure 1b, while other component modules would be hosted at the server end as in Figures 1a and 2.

Further, practitioners may choose to implement the each of the various embodiments described above on any number of different hardware and software computing platforms and environments and various combinations thereof, including, by way of just a few examples: a general-purpose hardware microprocessor such as the Intel Pentium series; operating system software such as Microsoft Windows/CE, Palm OS, or Apple Mac OS (particularly for client devices and client-side processing), or Unix, Linux, or Windows/NT (the latter three particularly for network data servers and server-side processing), and/or proprietary information access platforms such as Microsoft's WebTV or the Diva Systems video-on-demand system.

2. Processing Methodology

The present invention provides a spoken natural language interface for interrogation of remote electronic databases and retrieval of desired information. A preferred embodiment of the present invention utilizes the basic methodology outlined in the flow diagram of Figure 4 in order to provide this interface. This methodology will now be discussed.

a. Interpreting Spoken Natural Language Requests

At step 402, the user's spoken request for information is initially received in the form of raw (acoustic) voice data by a suitable input device, as previously discussed in connection with Figures 1-2. At step 404 the voice data received from the user is interpreted in order to understand the user's request for information. Preferably this step includes performing speech recognition in order to extract words from the voice data, and further includes natural language parsing of those words in order to generate a structured linguistic representation of the user's request.

Speech recognition in step 404 is performed using speech recognition engine 310. A variety of commercial quality, speech recognition engines are readily available on the market, as practitioners will know. For example, Nuance Communications offers a suite of speech recognition engines, including Nuance 6, its current flagship product, and Nuance Express, a lower cost package for entry-level applications. As one other example, IBM offers the ViaVoice speech recognition engine, including a low-cost shrink-wrapped version available through popular consumer distribution channels. Basically, a speech recognition engine processes acoustic voice data and attempts to generate a text stream of recognized words.

Typically, the speech recognition engine is provided with a vocabulary lexicon of likely words or phrases that the recognition engine can match against its analysis of acoustical signals, for purposes of a given application. Preferably, the lexicon is dynamically adjusted to reflect the current user context, as established by the preceding user inputs. For example, if a user is engaged in a dialogue with the system about movie selection, the recognition engine's vocabulary may preferably be adjusted to favor relevant words and phrases, such as a stored list of proper names for popular movie actors and directors, etc. Whereas if the current dialogue involves selection and viewing of a sports event, the engine's vocabulary might preferably be adjusted to favor a stored list of proper names for professional sports teams, etc. In addition, a speech recognition engine is provided with language models that help the engine predict the most likely interpretation of a given segment of acoustical voice data, in the current context of phonemes or words in which the segment appears. In addition, speech recognition engines often echo to the user, in more or less real-time, a transcription of the engine's best guess at what the user has said, giving the user an opportunity to confirm or reject.

In a further aspect of step 404, natural language interpreter (or parser) 320 linguistically parses and interprets the textual output of the speech recognition engine. In a preferred embodiment of the present invention, the natural-language interpreter attempts to determine both the meaning of spoken words (semantic processing) as well as the grammar of the statement (syntactic processing), such as the Gemini Natural Language Understanding System developed by SRI International. The Gemini system is described in detail in publications entitled "Gemini: A Natural Language System for Spoken-Language Understanding" and "Interleaving Syntax and Semantics in an Efficient Bottom-Up Parser," both of which are currently available online at http://www.ai.sri.com/natural-language/projects/arpa-sls/nat-lang.html. (Copies of those publications are also included in an information disclosure statement submitted herewith, and are incorporated herein by this reference). Briefly, Gemini applies a set of syntactic and semantic grammar rules to a word string using a bottom-up parser to generate a logical form, which is a structured representation of the context-independent meaning of the string. Gemini can be used with a variety of grammars, including general English grammar as well as application-specific grammars. The Gemini parser is based on "unification grammar," meaning that grammatical categories incorporate features that can be assigned values; so that when grammatical category expressions are matched in the course of parsing or semantic interpretation, the information contained in the features is combined, and if the feature values are incompatible the match fails.

It is possible for some applications to achieve a significant reduction in speech recognition error by using the natural-language processing system to re-score recognition hypotheses. For example, the grammars defined for a language parser like Gemini may be compiled into context-free grammar that, in turn, can be used directly as language models for speech recognition engines like the Nuance recognizer. Further details on this methodology are provided in the publication "Combining Linguistic and Statistical Knowledge Sources in Natural-Language Processing for ATIS" which is currently available online through http://www.ai.sri.com/natural-language/projects/arpa-sls/spnl-int.html. A copy of this publication is included in an information disclosure submitted herewith, and is incorporated herein by this reference.

In an embodiment of the present invention that may be preferable for some applications, the natural language interpreter "learns" from the past usage patterns of a particular user or of groups of users. In such an embodiment, the successfully interpreted requests of users are stored, and can then be used to enhance accuracy by comparing a current request to the stored requests, thereby allowing selection of a most probable result.

b. Constructing Navigation Queries

In step 405 request processing logic 300 identifies and selects an appropriate online data source where the desired information (in this case, current weather reports for a given city) can be found. Such selection may involve look-up in a locally stored table, or possibly dynamic searching through an online search engine, or other online search techniques. For some applications, an embodiment of the present invention may be implemented in which only access to a particular data source (such as a particular vendor's proprietary content database) is supported; in that case, step 405 may be trivial or may be eliminated entirely.

Step 406 attempts to construct a navigation query, reflecting the interpretation of step 404. This operation is preferably performed by query construction logic 330.

A "navigation query" means an electronic query, form, series of menu selections, or the like; being structured appropriately so as to navigate a particular data source of interest in search of desired information. In other words, a navigation query is constructed such that it includes whatever content and structure is required in order to access desired information electronically from a particular database or data source of interest.

For example, for many existing electronic databases, a navigation query can be embodied using a formal database query language such as Standard Query Language (SQL). For many databases, a navigation query can be constructed through a more user-friendly interactive front-end, such as a series of menus and/or interactive forms to be selected or filled in. SQL is a standard interactive and programming language for getting information from and updating a database. SQL is both an ANSI and an ISO standard. As is well known to practitioners, a Relational Database Management System (RDBMS), such as Microsoft's Access, Oracle's Oracle7, and Computer Associates' CA-OpenIngres, allow programmers to create, update, and administer a relational database. Practitioners of ordinary skill in the art will be thoroughly familiar with the notion of database navigation through structured query, and will be readily able to appreciate and utilize the existing data structures and navigational mechanisms for a given database, or to create such structures and mechanisms where desired.

In accordance with the present invention, the query constructed in step 406 must reflect the user's request as interpreted by the speech recognition engine and the NL parser in step 404. In embodiments of the present invention wherein data source 110 (or 210 in the corresponding embodiment of Figure 2) is a structured relational database or the like, step 406 of the present invention may entail constructing an appropriate Structured Query Language (SQL) query or the like, or automatically filling out a front-end query form, series of menus or the like, as described above.

In many existing Internet (and Intranet) applications, an online electronic data source is accessible to users only through the medium of interaction with a so-called Common Gateway Interface (CGI) script. Typically the user who visits a web site of this nature must fill in the fields of an online interactive form. The online form is in turn linked to a CGI script, which transparently handles actual navigation of the associated data source and produces output for viewing by the user's web browser. In other words, direct user access to the data source is not supported, only mediated access through the form and CGI script is offered.

For applications of this nature, an advantageous embodiment of the present invention "scrapes" the scripted online site where information desired by a user may be found in order to facilitate construction of an effective navigation query. For example, suppose that a user's spoken natural language request is: "What's the weather in Miami?" After this request is received at step 402 and interpreted at step 404, assume that step 405 determines that the desired weather information is available online through the medium of a CGI-scripted interactive form. Step 406 is then preferably carried out using the expanded process diagrammed in Figure 5. In particular, at sub-step 520, query construction logic 330 electronically "scrapes" the online interactive form, meaning that query construction logic 330 automatically extracts the format and structure of input fields accepted by the online form. At sub-step 522, a navigation query is then constructed by instantiating (filling in) the extracted input format -- essentially an electronic template -- in a manner reflecting the user's request for information as interpreted in step 404. The flow of control then returns to step 407 of Figure 4. Ultimately, when the query thus constructed by scraping is used to navigate the online data source in step 408, the query effectively initiates the same scripted response as if a human user had visited the online site and had typed appropriate entries into the input fields of the online form.

In the embodiment just described, scraping step 520 is preferably carried out with the assistance of an online extraction utility such as WebL. WebL is a scripting language for automating tasks on the World Wide Web. It is an imperative, interpreted language that has built-in support for common web protocols like HTTP and FTP, and popular data types like HTML and XML. WebL's implementation language is Java, and the complete source code is available from Compaq. In addition, step 520 is preferably performed dynamically when necessary -- in other words, on-the-fly in response to a particular user query -- but in some applications it may be possible to scrape relatively stable (unchanging) web sites of likely interest in advance and to cache the resulting template information.

It will be apparent, in light of the above teachings, that preferred embodiments of the present invention can provide a spoken natural language interface atop an existing, non-voice data navigation system, whereby users can interact by means of intuitive natural language input not strictly conforming to the linear browsing architecture or other artifacts of an existing menu/text/click navigation system. For example, users of an appropriate embodiment of the present invention for a video-on-demand application can directly speak the natural request: "Show me the movie 'Unforgiven'" -- instead of walking step-by-step through a typically linear sequence of genre/title/actor/director menus, scrolling and selecting from potentially long lists on each menu, or instead of being forced to use an alphanumeric keyboard that cannot be as comfortable to hold or use as a lightweight remote control. Similarly, users of an appropriate embodiment of the present invention for a web-surfing application in accordance with the process shown in Figure 5 can directly speak the natural request: "Show me a one-month price chart for Microsoft stock" -- instead of potentially having to navigate to an appropriate web site, search for the right ticker symbol, enter/select the symbol, and specify display of the desired one-month price chart, each of those steps potentially involving manual navigation and data entry to one or more different interaction screens. (Note that these examples are offered to illustrate some of the potential benefits offered by appropriate embodiments of the present invention, and not to limit the scope of the invention in any respect.)

c. Error Correction

Several problems can arise when attempting to perform searches based on spoken natural language input. As indicated at decision step 407 in the process of Figure 4, certain deficiencies may be identified during the process of query construction, before search of the data source is even attempted. For example, the user's request may fail to specify enough information in order to construct a navigation query that is specific enough to obtain a satisfactory search result. For example, a user might orally request "what's the weather?" whereas the national online data source identified in step 405 and scraped in step 520 might require specifying a particular city.

Additionally, certain deficiencies and problems may arise following the navigational search of the data source at step 408, as indicated at decision step 409 in Figure 4. For example, with reference to a video-on-demand application, a user may wish to see the movie "Unforgiven", but perhaps the user can't recall name of the film, but knows it was directed by and starred actor Clint Eastwood. A typical video-on-demand database might indeed be expected to allow queries specifying the name of a leading actor and/or director, but in the case of this query -- as in many cases -- that will not be enough to narrow the search to a single film, and additional user input in some form is required.

In the event that one or more deficiencies in the user's spoken request, as processed, result in the problems described, either at step 407 or 409, some form of error handling is in order. A straightforward, crude technique might be for the system to respond simply *"input not understood / insufficient; please try again."* However, that approach will likely result in frustrated users, and is not optimal or even acceptable for most applications. Instead, a preferred technique in accordance with the present invention handles such errors and deficiencies in user input at step 412, whether detected at step 407 or step 409, by soliciting additional input from the user in a manner taking advantage of the partial construction already performed and via user interface modalities in addition to spoken natural language ("multi-modality"). This supplemental interaction is preferably conducted through client display device 112 (202, in the embodiment of Figure 2), and may include textual, graphical, audio and/or video media. Further details and examples are provided below. Query refinement logic 340 preferably carries out step 412. The additional input received from the user is fed into and augments interpreting step 404, and query construction step 406 is likewise repeated with the benefit of the augmented interpretation. These operations, and subsequent navigation step 408, are preferably repeated until no

18

remaining problems or deficiencies are identified at decision points 407 or 409. Further details and examples for this query refinement process are provided immediately below.

Consider again the example in which the user of a video-on-demand application wishes to see "Unforgiven" but can only recall that it was directed by and starred Clint Eastwood. First, it bears noting that using a prior art navigational interface, such as a conventional menu interface, will likely be relatively tedious in this case. The user can proceed through a sequence of menus, such as Genre (select "western"), Title (skip), Actor ("Clint Eastwood"), and Director ("Clint Eastwood"). In each case --especially for the last two items -- the user would typically scroll and select from fairly long lists in order to enter his or her desired name, or perhaps use a relatively couch-unfriendly keypad to manually type the actor's name twice.

Using a preferred embodiment of the present invention, the user instead speaks aloud, holding remote control microphone 102, "I want to see that movie starring and directed by Clint Eastwood. Can't remember the title." At step 402 the voice data is received. At step 404 the voice data is interpreted. At step 405 an appropriate online data source is selected (or perhaps the system is directly connected to a proprietary video-on-demand provider). At step 406 a query is automatically constructed by the query construction logic 330 specifying "Clint Eastwood" in both the actor and director fields. Step 407 detects no obvious problems, and so the query is electronically submitted and the data source is navigated at step 408, yielding a list of several records satisfying the query (e.g., "Unforgiven", "True Crime", "Absolute Power", etc.). Step 409 detects that additional user input is needed to further refine the query in order to select a particular film for viewing.

At that point, in step 412 query refinement logic 340 might preferably generate a display for client display device 112 showing the (relatively short) list of film titles that satisfy the user's stated constraints. The user can then preferably use a relatively convenient input modality, such as buttons on the remote control, to select the desired title from the menu. In a further preferred embodiment, the first title on the list is highlighted by default, so that the user can simply press an "OK" button to choose that selection. In a further preferred feature, the user can mix input modalities by speaking a response like "I want number one on the list." Alternatively, the user can preferably say, "Let's see Unforgiven," having now been reminded of the title by the menu display.

Utilizing the user's supplemental input, request processing logic 300 iterates again through steps 404 and 406, this time constructing a fully-specified query that specifically requests the Eastwood film "Unforgiven." Step 408 navigates the data source using that query and retrieves the desired film, which is then electronically transmitted in step 410 from network server 108 to client display device 112 via communications network 106.

Now consider again the example in which the user of a web surfing application wants to know his or her local weather, and simply asks, "what's the weather?" At step 402 the voice data is received. At step 404 the voice data is interpreted. At step 405 an online web site providing current weather information for major cities around the world is selected. At step 406 and sub-step 520, the online site is scraped using a WebL-style tool to extract an input template for interacting with the site. At sub-step 522, query construction logic 330 attempts to construct a navigation query by instantiating the input template, but determines (quite rightly) that a required field -- name of city -- cannot be determined from the user's spoken request as interpreted in step 404. Step 407 detects this deficiency, and in step 412 query refinement logic 340 preferably generates output for client display device 112 soliciting the necessary supplemental input. In a preferred embodiment, the output might display the name of the city where the user is located highlighted by default. The user can then simply press an "OK" button -- or perhaps mix modalities by saying "yes, exactly" -- to choose that selection. A preferred embodiment would further display an alphabetical scrollable menu listing other major cities, and/or invite the user to speak or select the name of the desired city.

Here again, utilizing the user's supplemental input, request processing logic 300 iterates through steps 404 and 406. This time, in performing sub-step 520, a cached version of the input template already scraped in the previous iteration might preferably be retrieved. In sub-step 522, query construction logic 330 succeeds this time in instantiating the input template and constructing an effective query, since the desired city has now been clarified. Step 408 navigates the data source using that query and retrieves the desired weather information, which is then electronically transmitted in step 410 from network server 108 to client display device 112 via communications network 106.

It is worth noting that in some instances, there may be details that are not explicitly provided by the user, but that query construction logic 330 or query refinement logic 340 may preferably deduce on their own through reasonable assumptions, rather than requiring the use to provide explicit clarification. For example, in the example previously described regarding a request for a weather report, in some applications it might be preferable for the system to simply assume that the user means a weather report for his or her home area and to retrieve that information, if the cost of doing so is not significantly greater than the cost of asking the user to clarify the query. Making such an assumption might be even more strongly justified in a preferred embodiment, as described earlier, where user histories are tracked, and where such history indicates that a particular user or group of users typically expect local information when asking for a weather forecast. At any rate, in the event such an assumption is made, if the user actually intended to request the weather for a different city, the user would then need to ask his or her question again. It will be apparent to practitioners, in light of the above teachings, that the choice of whether to program query construction logic 330 and query refinement logic 340 to make make particular assumptions will typically involve trade-offs involving user conveience that can be assessed in the context of specific applications.

21

3. Open Agent Architecture (OAA®)

Open Agent Architecture™ (OAA®) is a software platform, developed by the assignee of the present invention, that enables effective, dynamic collaboration among communities of distributed electronic agents. OAA is described in greater detail in co-pending U.S. Patent Application No. 09/225,198, which has been incorporated herein by reference. Very briefly, the functionality of each client agent is made available to the agent community through registration of the client agent's capabilities with a facilitator. A software "wrapper" essentially surrounds the underlying application program performing the services offered by each client. The common infrastructure for constructing agents is preferably supplied by an *agent library*. The agent library is preferably accessible in the runtime environment of several different programming languages. The agent library preferably minimizes the effort required to construct a new system and maximizes the ease with which legacy systems can be "wrapped" and made compatible with the agent-based architecture of the present invention. When invoked, a client agent makes a connection to a facilitator, which is known as its *parent facilitator*. Upon connection, an agent registers with its parent facilitator a specification of the capabilities and services it can provide, using a high-level, declarative Interagent Communication Language (*"ICL"*) to express those capabilities. Tasks are presented to the facilitator in the form of ICL goal expressions. When a facilitator determines that the registered capabilities of one of its client agents will help satisfy a current goal or sub-goal thereof, the facilitator delegates that sub-goal to the client agent in the form of an ICL request. The client agent processes the request and returns answers or information to the facilitator. In processing a request, the client agent can use *ICL* to request services of other agents, or utilize other infrastructure services for collaborative work. The facilitator coordinates and integrates the results received from different client agents on various sub-goals, in order to satisfy the overall goal.

OAA provides a useful software platform for building systems that integrate spoken natural language as well as other user input modalities. For example, see the above-referenced co-pending patent application, especially Figure 13 and the corresponding discussion of a "multi-modal maps" application, and Figure 12 and the corresponding discussion of a "unified messaging" application. Another example is the InfoWiz interactive information kiosk developed by the assignee and described in the document entitled "InfoWiz: An Animated Voice Interactive Information System" available online at http://www.ai.sri.com/~oaa/applications.html. A copy of the InfoWhiz document is provided in an Information Disclosure Statement submitted herewith and incorporated herein by this reference. A further example is the "CommandTalk" application developed by the assignee for the U.S. military, as described online at http://www.ai.sri.com/~lesaf/commandtalk.html and in the following publications, copies of which are provided in an Information Disclosure Statement submitted herewith and incorporated herein by this reference:

- "CommandTalk: A Spoken-Language Interface for Battlefield Simulations", 1997, by Robert Moore, John Dowding, Harry Bratt, J. Mark Gawron, Yonael Gorfu and Adam Cheyer, in "Proceedings of the Fifth Conference on Applied Natural Language Processing", Washington, DC, pp. 1-7, Association for Computational Linguistics

- "The CommandTalk Spoken Dialogue System", 1999, by Amanda Stent, John Dowding, Jean Mark Gawron, Elizabeth Owen Bratt and Robert Moore, in "Proceedings of the Thirty-Seventh Annual Meeting of the ACL", pp. 183-190, University of Maryland, College Park, MD, Association for Computational Linguistics

- "Interpreting Language in Context in CommandTalk", 1999, by John Dowding and Elizabeth Owen Bratt and Sharon Goldwater, in "Communicative Agents: The Use of Natural Language in Embodied Systems", pp. 63-67, Association for Computing Machinery (ACM) Special Interest Group on Artificial Intelligence (SIGART), Seattle, WA

For some applications and systems, OAA can provide an advantageous platform for constructing embodiments of the present invention. For example, a representative application is now briefly presented, with reference to Figure 6. If the statement "show me movies starring John Wayne" is spoken into the voice input device, the voice data for this request will be sent by UI agent 650 to facilitator 600, which in turn will ask natural language (NL) agent 620 and speech recognition agent 610 to interpret the query and return the interpretation in *ICL* format. The resulting *ICL* goal expression is then routed by the facilitator to appropriate agents -- in this case, video-on-demand database agent 640 -- to execute the request. Video database agent 640 preferably includes or is coupled to an appropriate embodiment of query construction logic 330 and query refinement logic 340, and may also issue ICL requests to facilitator 600 for additional assistance -- e.g., display of menus and capture of additional user input in the event that query refinement is needed -- and facilitator 600 will delegate such requests to appropriate client agents in the community. When the desired video content is ultimately retrieved by video database agent 640, UI agent 650 is invoked by facilitator 600 to display the movie.

Other spoken user requests, such as a request for the current weather in New York City or for a stock quote, would eventually lead facilitator to invoke web database agent 630 to access the desired information from an appropriate Internet site. Here again, web database agent 630 preferably includes or is coupled to an appropriate embodiment of query construction logic 330 and query refinement logic 340, including a scraping utility such as WebL. Other spoken requests, such as a request to view recent emails or access voice mail, would lead the facilitator to invoke the appropriate email agent 660 and/or telephone agent 680. A request to record a televised program of interest might lead facilitator 600 to invoke web database agent 630 to return televised program schedule information, and then invoke VCR controller agent 680 to program the associated VCR unit to record the desired television program at the scheduled time.

Control and connectivity embracing additional electronic home appliances (e.g., microwave oven, home surveillance system, etc.) can be integrated in comparable fashion. Indeed, an advantage of OAA-based embodiments of the present invention, that will be apparent to practitioners in light of the above teachings and in light of the teachings disclosed in the cited co-pending patent applications, is the relative ease and flexibility with which additional service agents can be plugged into the existing platform, immediately enabling the facilitator to respond dynamically to spoken natural language requests for the corresponding services.

4. Further Embodiments and Equivalents

While the present invention has been described in terms of several preferred embodiments, there are many alterations, permutations, and equivalents that may fall within the scope of this invention. It should also be noted that there are many alternative ways of implementing the methods and apparatuses of the present invention. It is therefore intended that the following appended claims be interpreted as including all such alterations, permutations, and equivalents as fall within the true spirit and scope of the present invention.

CLAIMS

What is claimed is:

1. A method for utilizing spoken natural language for navigating an electronic data source, the electronic data source being located at one or more network servers located remotely from a user, comprising the steps of:

(a) receiving a spoken natural language ("NL") request for desired information from the user;

(b) rendering an interpretation of the spoken natural language request;

(c) constructing at least part of a navigation query based upon the interpretation;

(d) soliciting additional input from the user, including user interaction in a modality different than the original request;

(e) refining the navigation query, based upon the additional input;

(f) using the refined navigation query to select a portion of the electronic data source; and

A

(g) transmitting the selected portion of the electronic data source from the network server to a client device of the user.

2. The method of claim 1, wherein the step of rendering an interpretation further includes deriving linguistic information by using a speech recognition engine and an NL parser.

3. The method of claim 1, wherein the step of constructing a navigation query further includes the steps of extracting an input template for an online scripted interface to the data source, and using the input template to construct the navigation query.

~~4. The method of claim 3, wherein the step of extracting an input template includes dynamically scraping the online scripted interface.~~

~~5. The method of claim 1, wherein the navigation query is constructed in the format of a database query language.~~

~~6. The method of claim 1, wherein the step of rendering an interpretation and the step of constructing a navigation query are performed, at least in part, on a computing device located locally with the user.~~

~~7. The method of claim 1, wherein the step of rendering an interpretation and the step of constructing a navigation query are performed, at least in part, on a network computing device located remotely from the user.~~

~~8. The method of claim 1, wherein the step of soliciting additional input is performed in response to one or more deficiencies encountered during the step of constructing a navigation query.~~

~~9. The method of claim 8, wherein the deficiencies include unresolved words of the spoken NL request.~~

A

~~10. The method of claim 8, wherein the deficiencies include one or more required elements of the navigational query not determinable from the interpretation of the spoken NL request.~~

~~11. The method of claim 1, wherein the step of soliciting additional input is performed in response to one or more deficiencies encountered after a first navigation of the data source using the navigation query constructed in step (c).~~

~~12. The method of claim 11, wherein the deficiencies include existence of more than one data record within the data source responsive to the navigation query.~~

~~13. The method of claim 11, wherein the deficiencies include failure to identify a single data record within the data source responsive to the navigation query.~~

~~14. The method of claim 1, wherein the input modality of step (d) includes selecting from a displayed option menu.~~

~~15. The method of claim 14, wherein the act of selecting from the displayed option menu is performed by speaking.~~

~~16. The method of claim 1, wherein the method is performed with respect to a plurality of simultaneous users and corresponding client devices.~~

~~17. The method of claim 1, further including the step of selecting the data source from among a plurality of candidate electronic data sources, in response to the interpretation of the spoken NL request.~~

~~18. The method of claim 1, wherein the electronic data source stores multimedia content including at least one of video content and audio content.~~

~~19. A system for utilizing spoken natural language to navigate an electronic data source, the electronic data source being located at one or more network servers located remotely from a user, the system comprising:~~

~~(a) a portable microphone operable to receive a spoken natural language ("NL") request for desired information from the user;~~

~~(b) spoken language processing logic, operable to render an interpretation of the spoken natural language request;~~

~~(c) query construction logic, operable to construct a navigation query in response to the interpretation of the spoken natural language request;~~

~~(d) user interaction logic, operable to solicit additional input from the user, including user interaction in a modality different than the original request;~~

~~(e) query refining logic, operable to refine the navigation query, based upon the additional input;~~

~~(f) navigation logic, operable to select a portion of the electronic data source using the navigation query; and~~

(g) electronic communications infrastructure for transmitting the selected portion of the electronic data source from the network server to a primarily stationary, display device located locally with the user.

20. The system of claim 19, wherein the spoken language processing logic includes speech recognition logic and an NL parsing logic for deriving linguistic information.

21. The system of claim 19, wherein the spoken language processing logic extracts an input template for an online scripted interface to the data source, and uses the input template to construct the navigation query.

22. The system of claim 21, wherein the spoken language processing logic dynamically scrapes the online scripted interface.

23. The system of claim 19, wherein the query construction logic constructs the query in the format of a database query language.

24. The system of claim 19, wherein at least a portion of the spoken language processing logic is hosted on a computing device located locally with the user, and wherein the portable microphone is electronically coupled to the local computing device.

25. The system of claim 19, wherein at least a portion of the spoken language processing logic is hosted on a network computing device located remotely from the user, and wherein the portable microphone sends data to the remote network computing device via the communications infrastructure.

26. The system of claim 19, wherein the user interaction logic solicits additional input in response to one or more deficiencies encountered during construction of the navigation query.

27. The system of claim 26, wherein the deficiencies include unresolved words of the spoken NL request.

28. The system of claim 26, wherein the deficiencies include one or more required elements of the navigational query not determinable from the interpretation of the spoken NL request.

29. The system of claim 19, wherein the user interaction logic solicits additional input in response to one or more deficiencies encountered after a first navigation of the data source performed by the navigation logic.

30. The system of claim 29, wherein the deficiencies include existence of more than one data record within the data source responsive to the navigation query.

31. The system of claim 29, wherein the deficiencies include failure to identify a single data record within the data source responsive to the navigation query.

32. The system of claim 19, wherein the user interaction logic displays an option menu.

33. The system of claim 32, wherein the act of selecting from the displayed option menu is performed by speaking.

34. The system of claim 19, wherein the navigation logic selects the data source from among a plurality of candidate electronic data sources, in response to the interpretation of the spoken NL request.

35. The system of claim 19, wherein the electronic data source stores multimedia content including at least one of video content and audio content.

36. The system of claim 19, wherein the display device receives data from the electronicdata source on the network servers via a communications box.

37. The system of claim 19, wherein the electronic communication infrastructure is a two-way infrastructure and is selected from among one or more of the following group: {coaxial cable, DSL, satellite, wireless/cellular, fiber-optic}.

38. An computer program embodied on a computer readable medium for utilizing spoken natural language for navigating an electronic data source, the

electronic data source being located at one or more network servers located remotely from a user, comprising:

(a) a code segment that receives a spoken natural language ("NL") request for desired information from the user;

(b) a code segment that renders an interpretation of the spoken natural language request;

(c) a code segment that constructs at least part of a navigation query based upon the interpretation;

(d) a code segment that solicits additional input from the user, including user interaction in a modality different than the original request;

(e) a code segment that refines the navigation query, based upon the additional input;

(f) a code segment that uses the refined navigation query to select a portion of the electronic data source; and

(g) a code segment that transmits the selected portion of the electronic data source from the network server to a primarily stationary, display device located locally with the user.

39. The computer program of claim 38, further comprising a code segment that derives linguistic information by using a speech recognition engine and an NL parser.

40. The computer program of claim 38, further comprising a code segment that extract an input template for an online scripted interface to the data source, and a code segment that uses the input template to construct the navigation query.

41. The computer program of claim 40, further comprising a code segment that dynamically scrapes the online scripted interface.

42. The computer program of claim 38, wherein the navigation query is constructed in the format of a database query language.

43. The computer program of claim 38, wherein rendering of the interpretation and the construction of the navigation query are performed, at least in part, on a computing device located locally with the user.

44. The computer program of claim 38, wherein the rendering of the interpretation and the construction of a navigation query are performed, at least in part, on a network computing device located remotely from the user.

45. The computer program of claim 38, wherein code segment that solicits additional input solicits the additional input in response to one or more deficiencies encountered during the constructing of the navigation query.

46. The computer program of claim 45, wherein the deficiencies include unresolved words of the spoken NL request.

47. The computer program of claim 45, wherein the deficiencies include one or more required elements of the navigational query not determinable from the interpretation of the spoken NL request.

48. The computer program of claim 38, wherein the code segment that solicits the additional input solicits the additional input in response to one or more deficiencies encountered after a first navigation of the data source.

49. The computer program of claim 48, wherein the deficiencies include existence of more than one data record within the data source responsive to the navigation query.

A

50. The computer program of claim 48, wherein the deficiencies include failure to identify a single data record within the data source responsive to the navigation query.

51. The computer program of claim 38, wherein code segment that solicits additional input displays an option menu.

52. The computer program of claim 51, wherein the act of selecting from the displayed option menu is performed by speaking.

~~53. The computer program of claim 38, wherein the code segments of the computer program operate with respect to a plurality of simultaneous users and corresponding client devices.~~

~~54. The computer program of claim 38, further comprising a code segment that selects the data source from among a plurality of candidate electronic data sources, in response to the interpretation of the spoken NL request.~~

~~55. The computer program of claim 38, wherein the electronic data source stores multimedia content including at least one of video content and audio content.~~

Add A5

NAVIGATING NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN NATURAL LANGUAGE INPUT WITH MULTIMODAL ERROR FEEDBACK

ABSTRACT OF THE INVENTION

Sub A7

~~A system, method, and article of manufacture are provided for navigating an electronic data source by means of spoken natural language. When a spoken natural language input request is received from a user, it is interpreted. Additional input is solicited from the user in a modality different than the original request and used to refine the navigation query. The resulting interpretation of the request is thereupon used to automatically construct an operational navigation query to retrieve the desired information from one or more electronic network data sources.~~

1

Post-it® ~~Fax Note~~ 7671

Date 6 29 00	# of pages ▶ 7
To Domenic Kotab	From K. Elarsri
Co./Dept.	Co. SRI Intl.
Phone # 408-971-4660	Phone # 650-859-6631
Fax # 408-971-4660	Fax # 650.859-6420

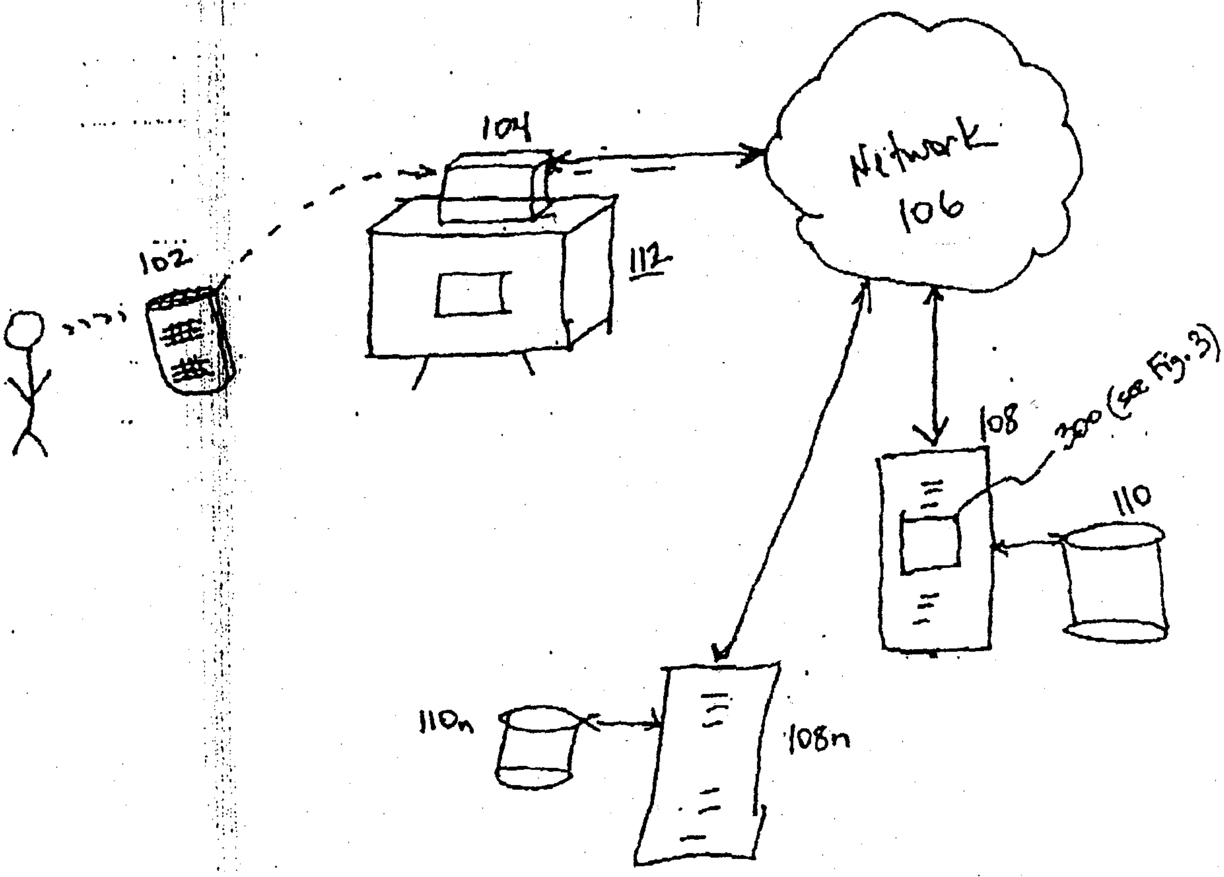


Fig. 1a

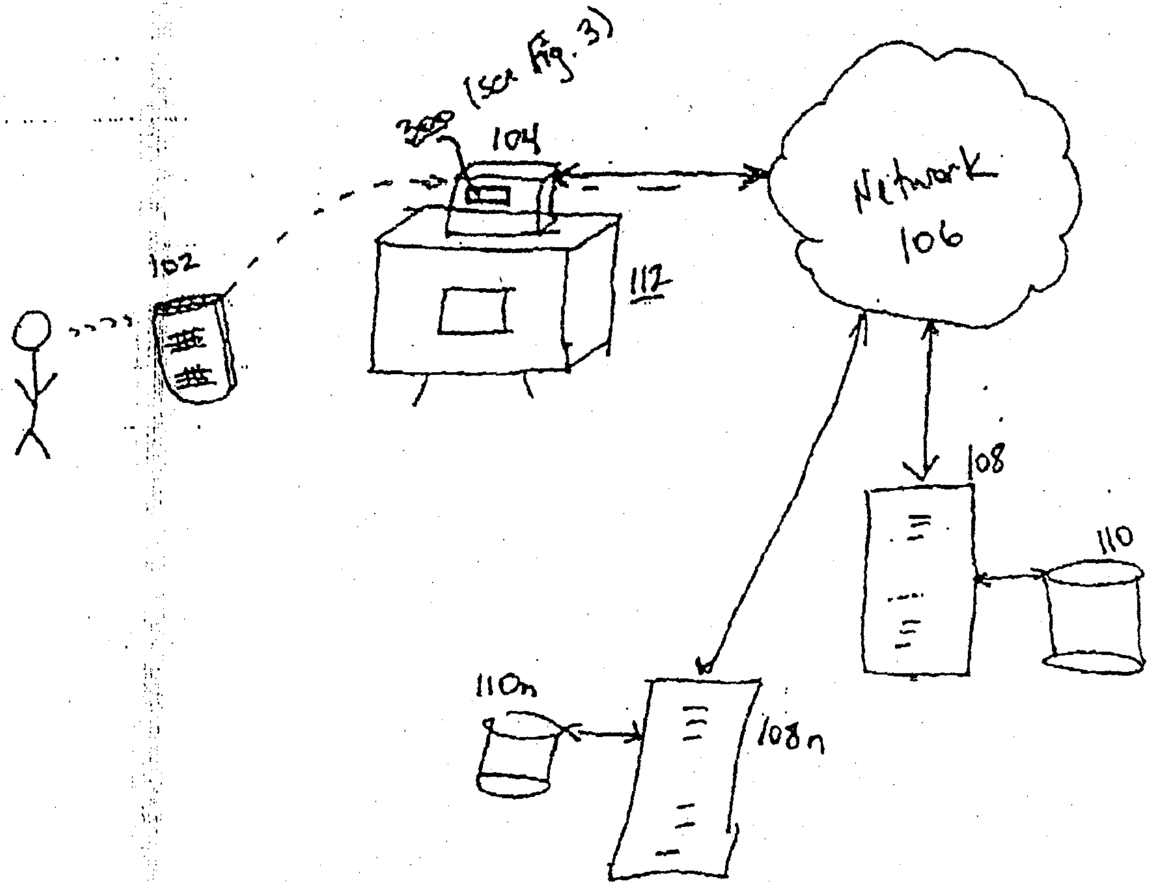


Fig. 1b

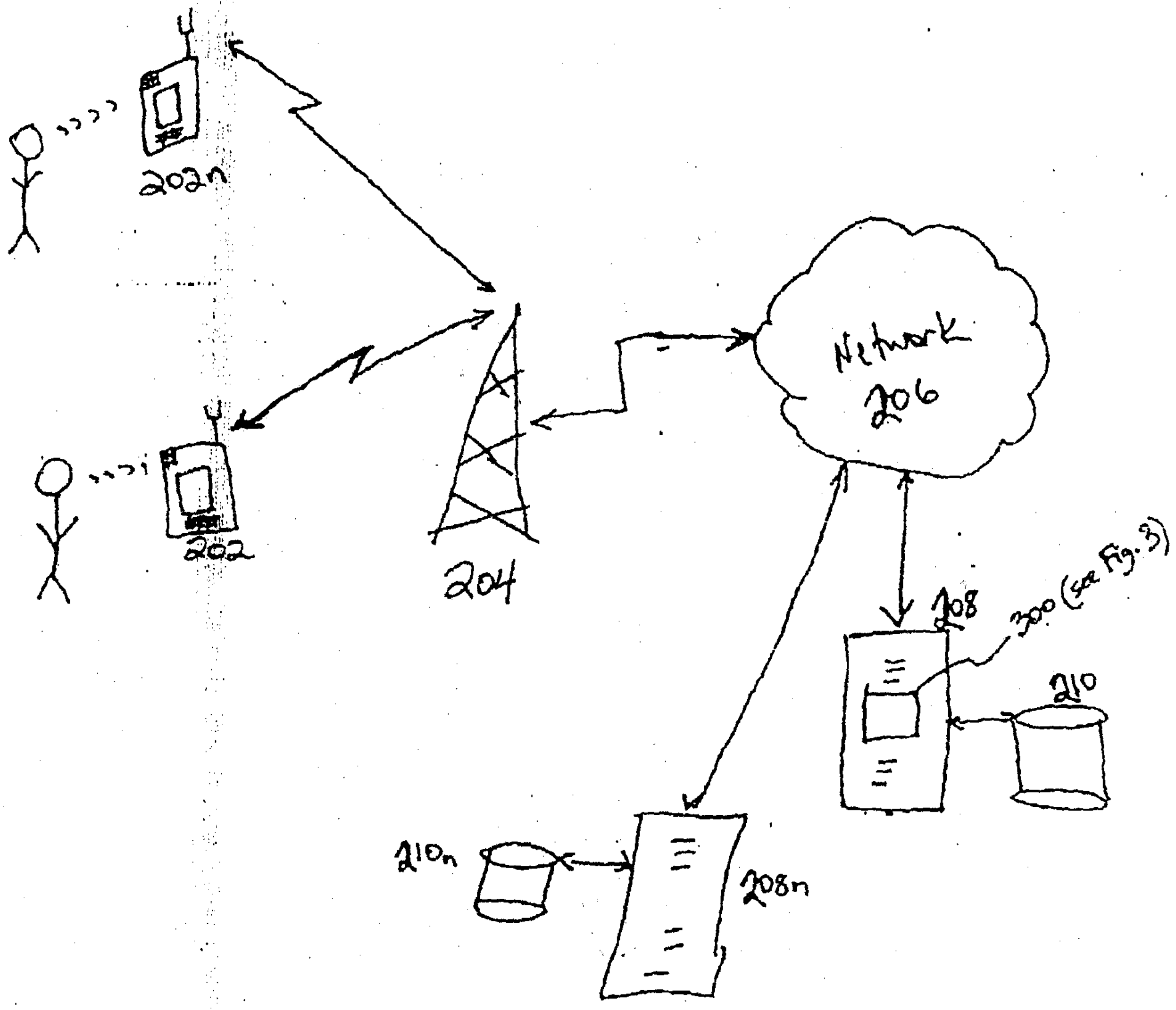


Fig. 2

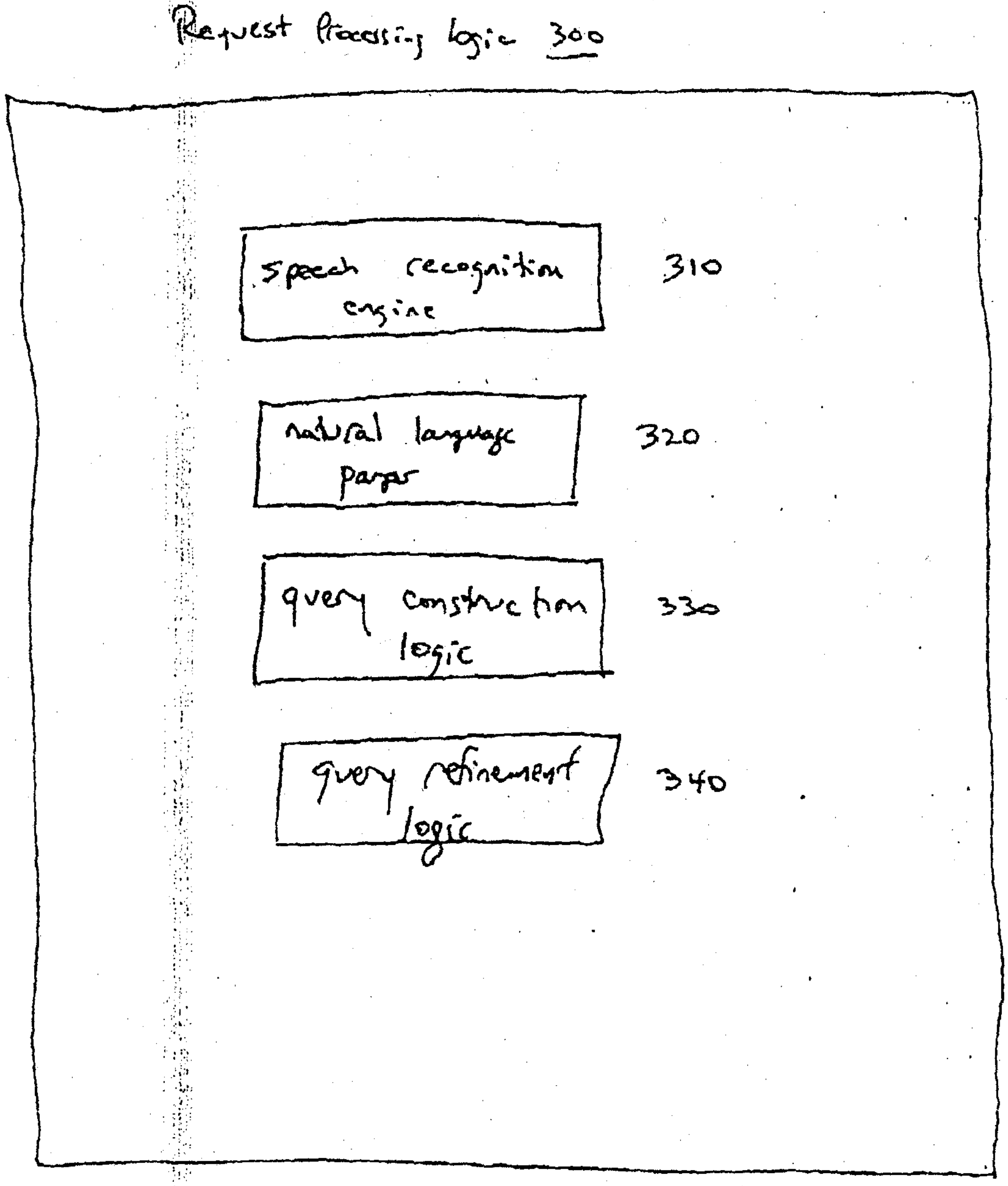

Request Processing logic 300
speech recognition engine
310
natural language parser
320
query construction logic
330
query refinement logic
340

Fig. 3

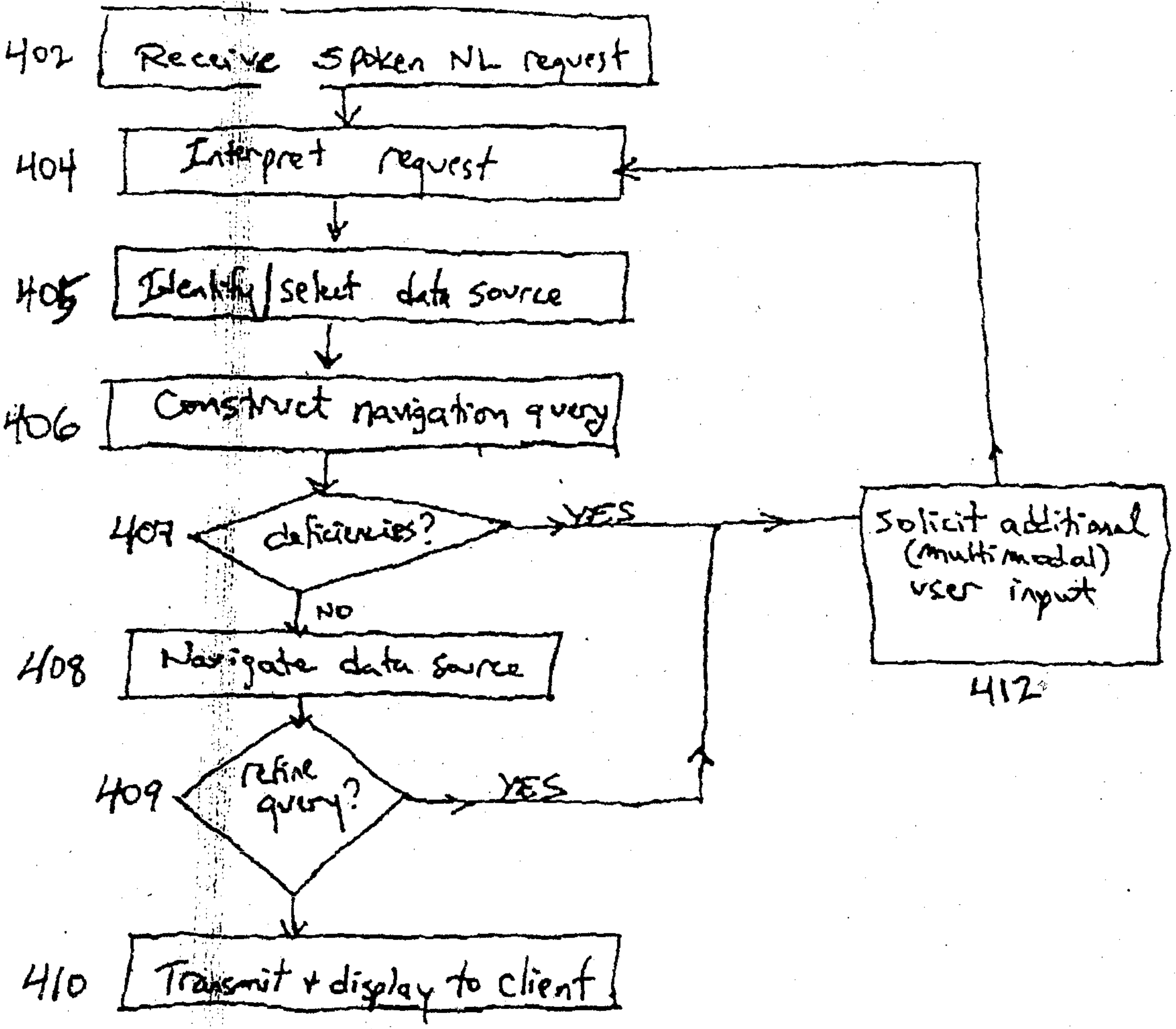

402
Receive spoken NL request
404
Interpret request
405
Identify/select data source
406
Construct navigation query
407
deficiencies?
YES
NO
408
Navigate data source
409
refine query?
YES
410
Transmit + display to client
Solicit additional (multimodal) user input
412

Fig. 4

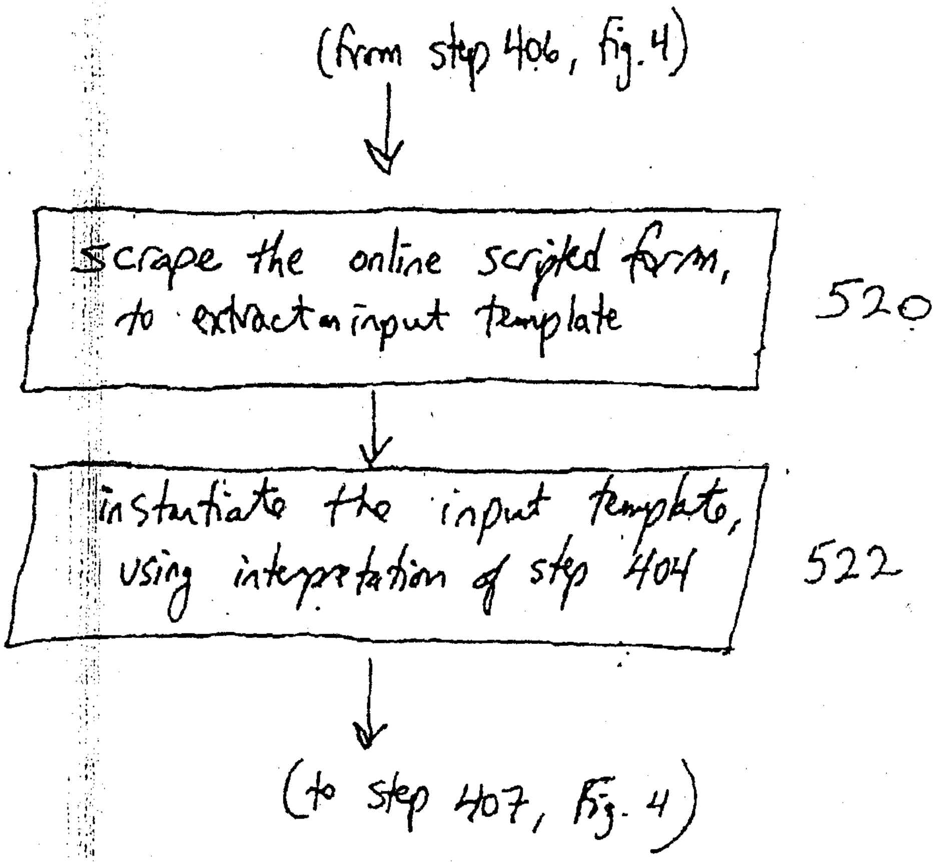

(from step 406, fig. 4)
scrape the online scripted form, to extract an input template
520
instantiate the input template, using interpretation of step 404
522
(to step 407, Fig. 4)

Fig. 5

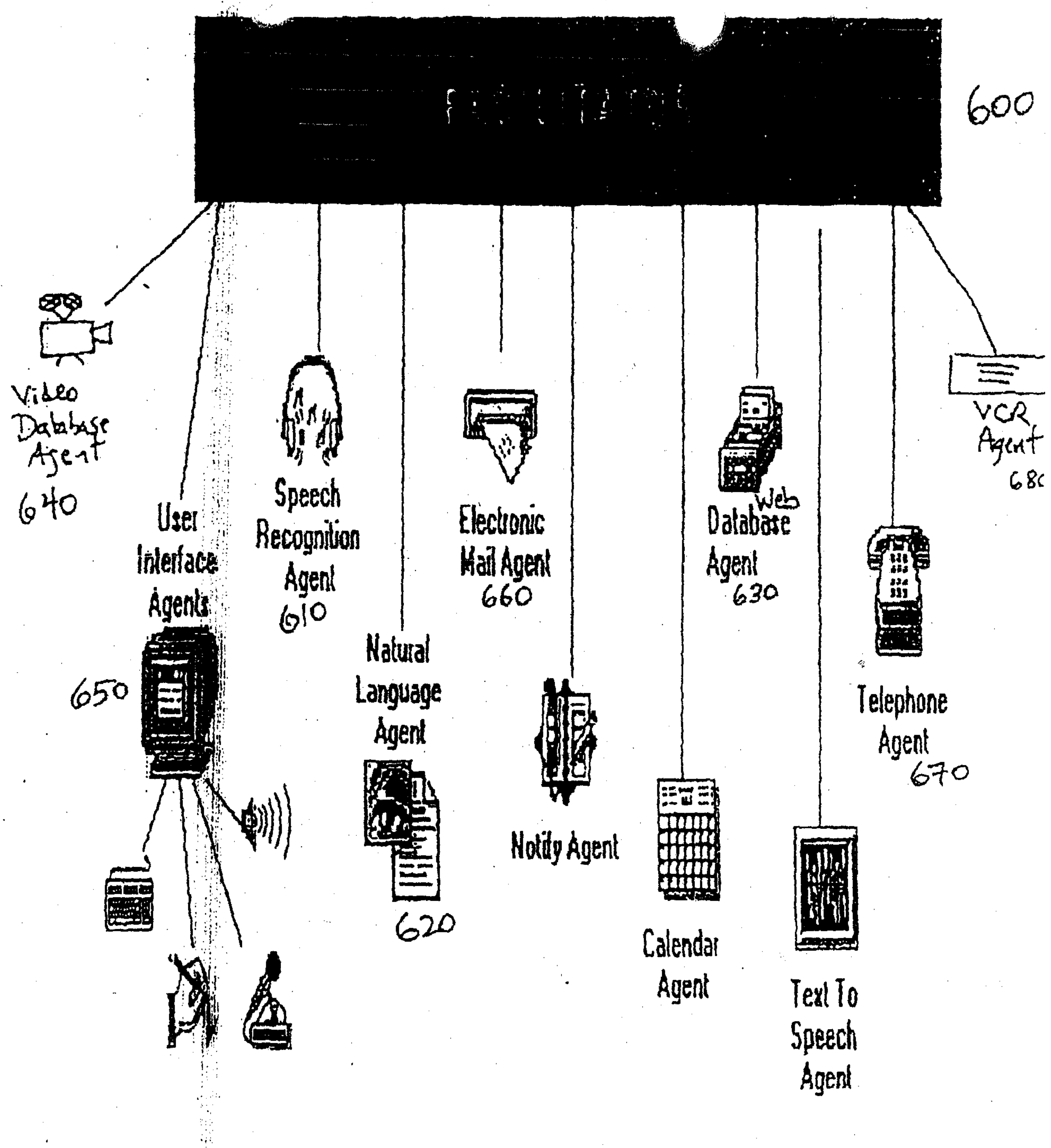

600
Video Database Agent
640
User Interface Agents
650
Speech Recognition Agent
610
Natural Language Agent
620
Electronic Mail Agent
660
Notify Agent
Calendar Agent
Web Database Agent
630
Text To Speech Agent
Telephone Agent
670
VCR Agent
680

Fig. 6

PRINT OF DRAWINGS
AS ORIGINALLY FI

Post-it® Fax Note 7671	Date 6/29/00	# of pages ▶ 7
To Domenic Rotab	From K. Elorsmi	
Co./Dept.	Co. SRI Intl.	
Phone # 408-971-4660	Phone # 650-859-6631	
Fax # 408-971-4660	Fax # 650.859-6420	

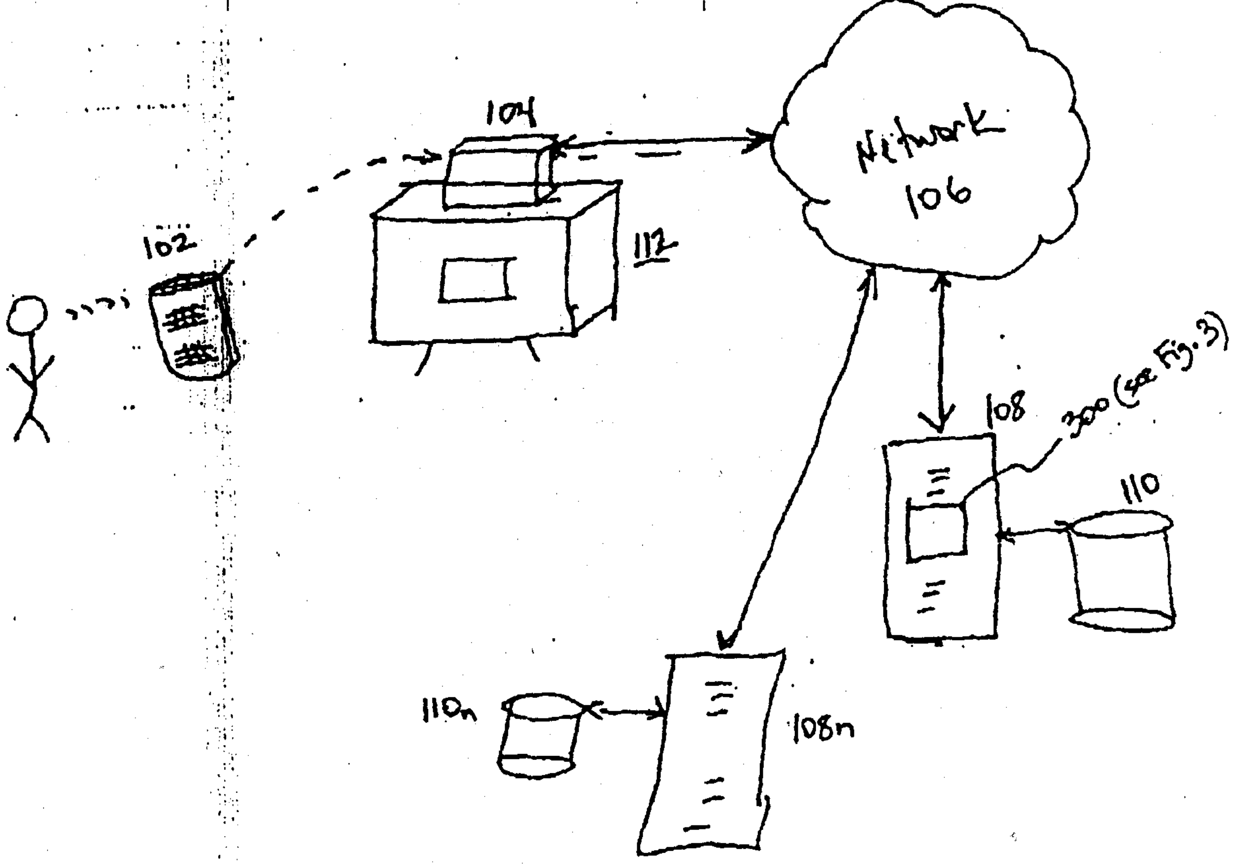


Fig. 1a

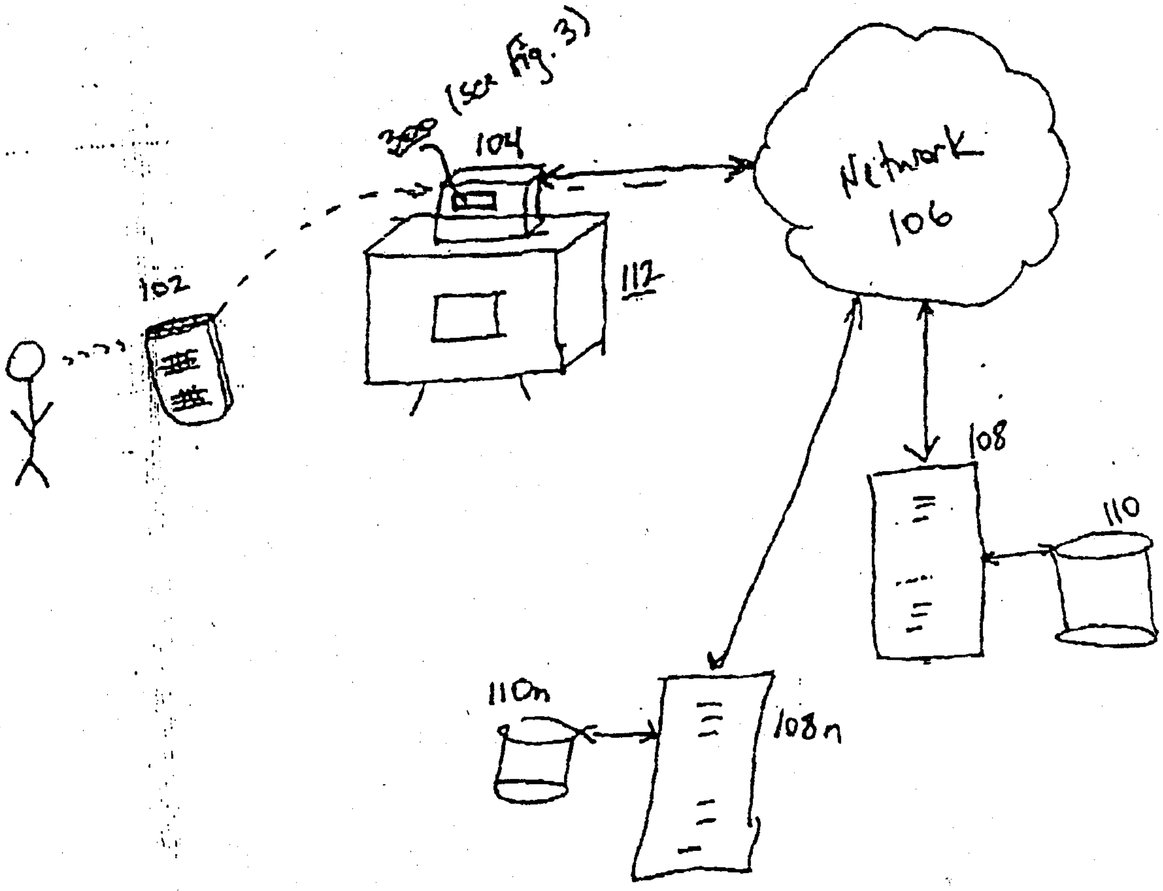


Fig. 1b.

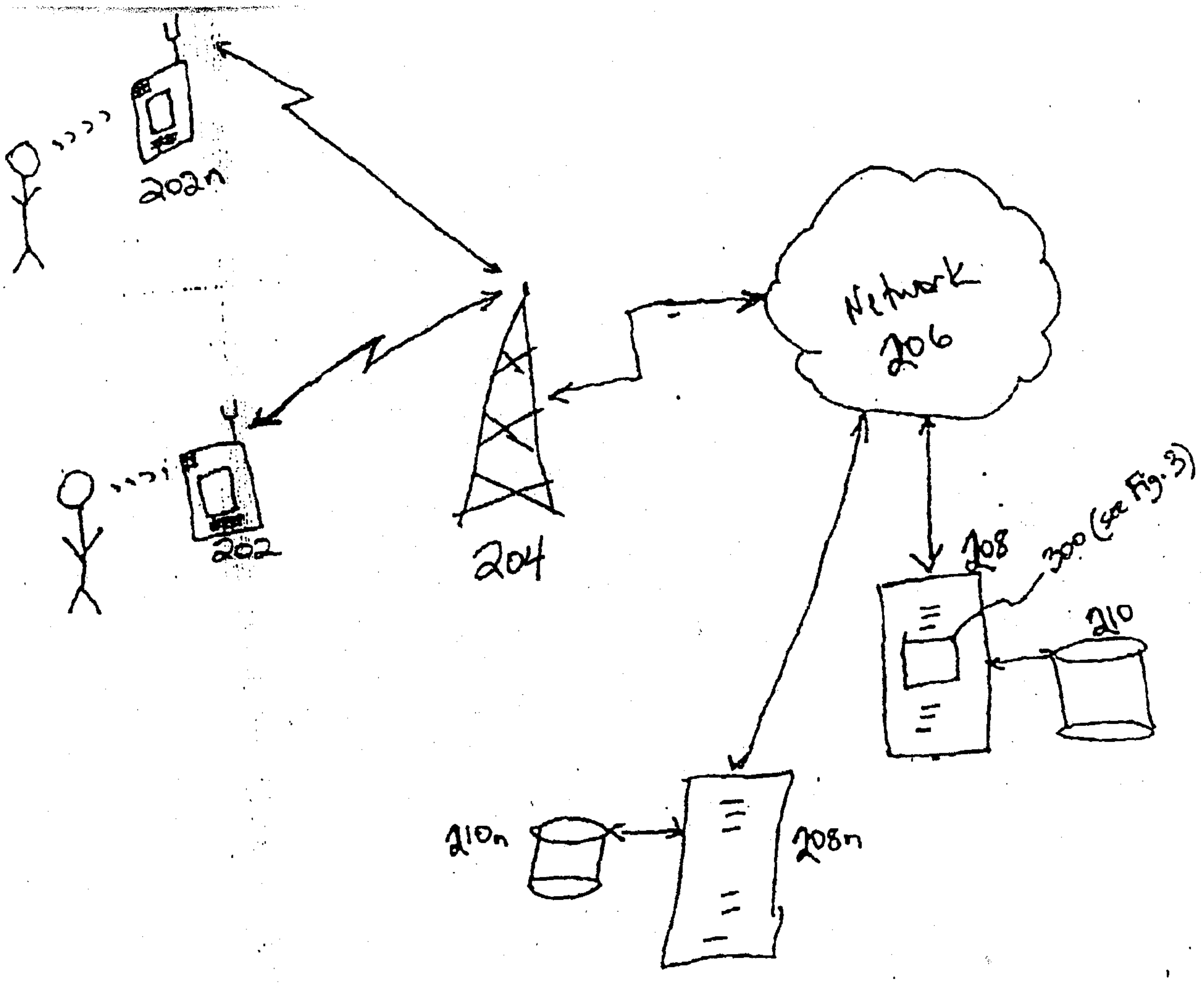


Fig. 2

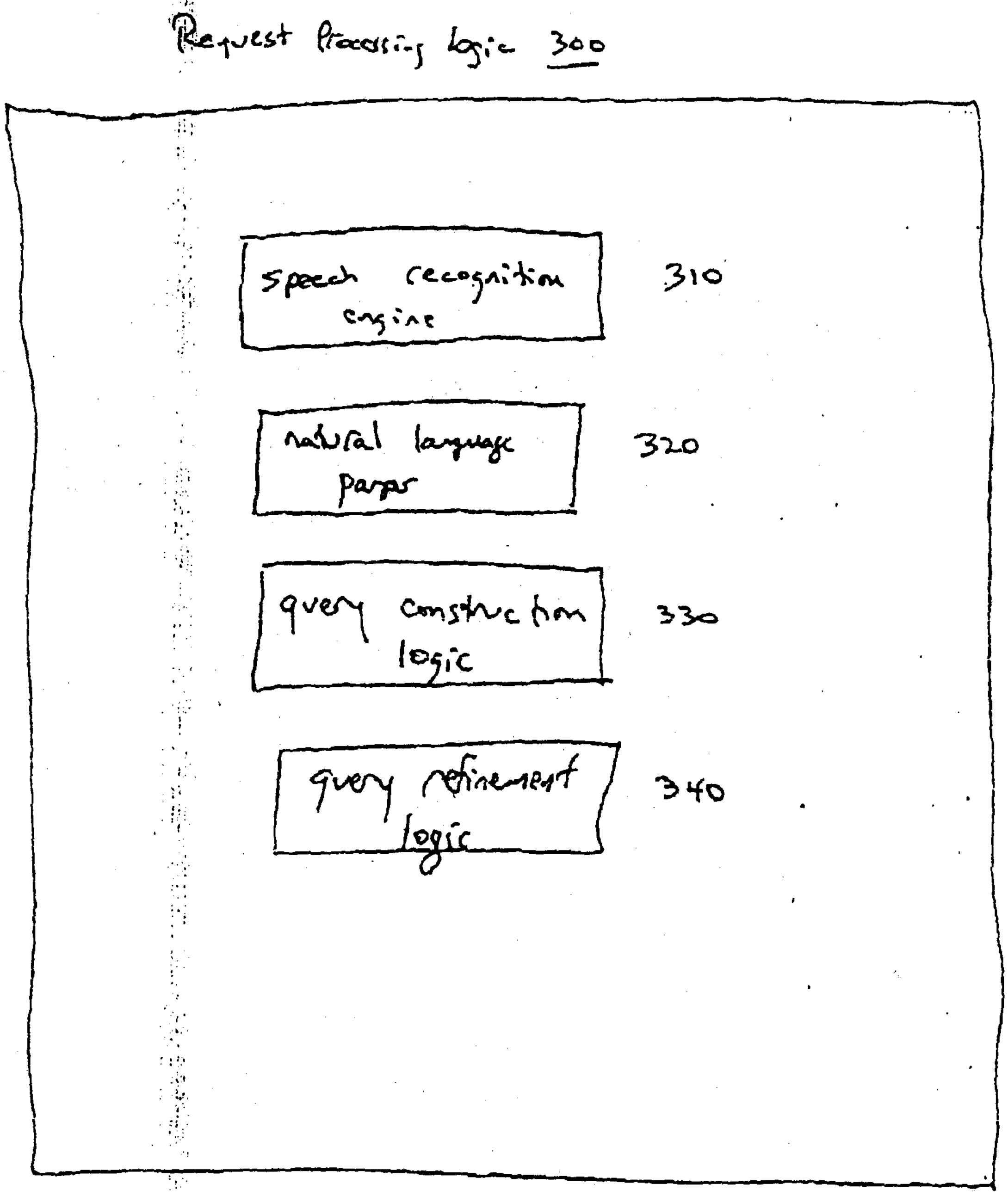

Request Processing logic 300
speech recognition engine
310
natural language parser
320
query construction logic
330
query refinement logic
340

Fig. 3

PRINT OF DRAWINGS
AS ORIGINALLY FI

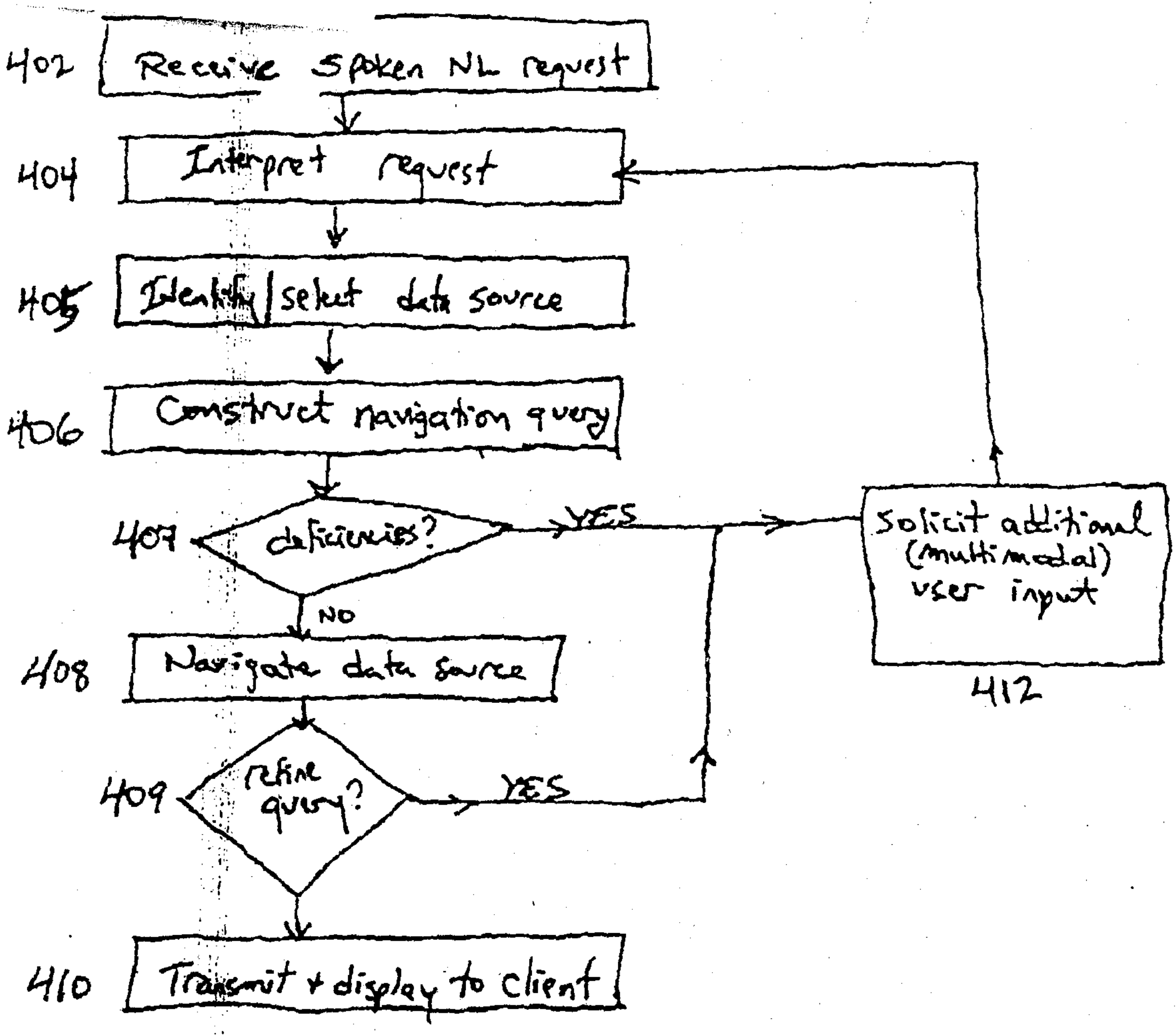

402
Receive spoken NL request
404
Interpret request
405
Identify/select data source
406
Construct navigation query
407
deficiencies?
YES
NO
Solicit additional (multimodal) user input
412
408
Navigate data source
409
refine query?
YES
410
Transmit + display to client

Fig. 4

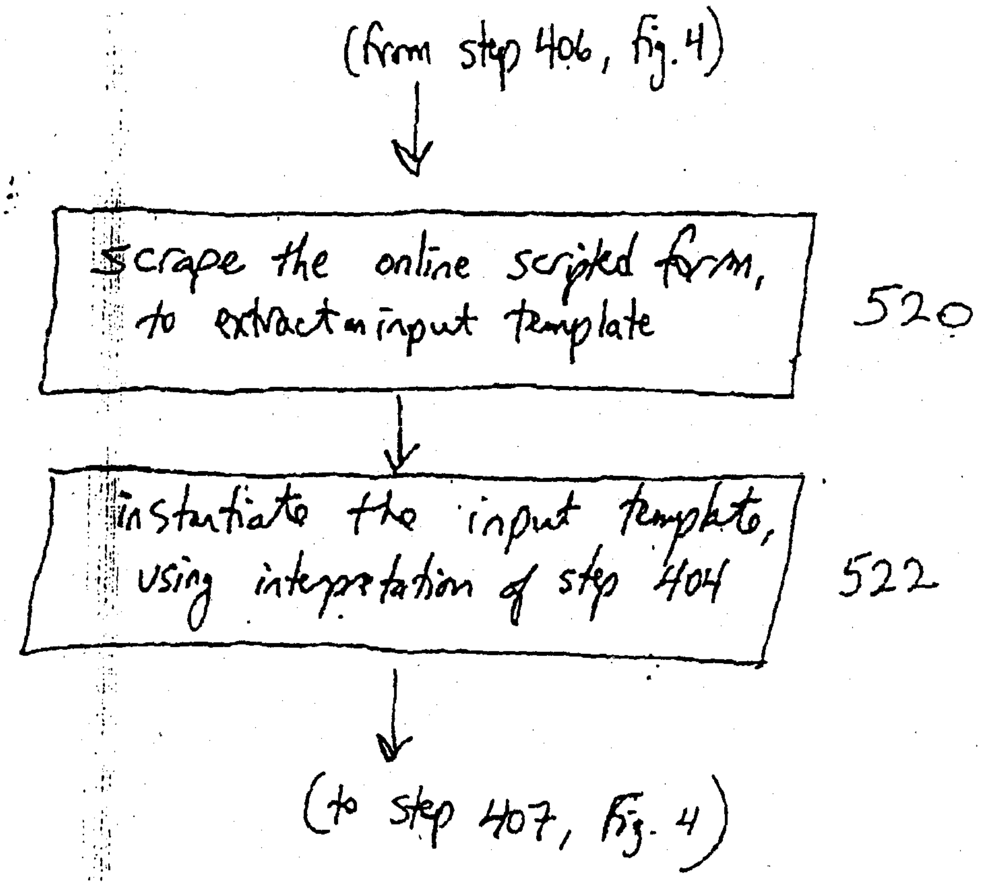

(from step 406, fig. 4)
scrape the online scripted form, to extract an input template
520
instantiate the input template, using interpretation of step 404
522
(to step 407, fig. 4)

Fig. 5

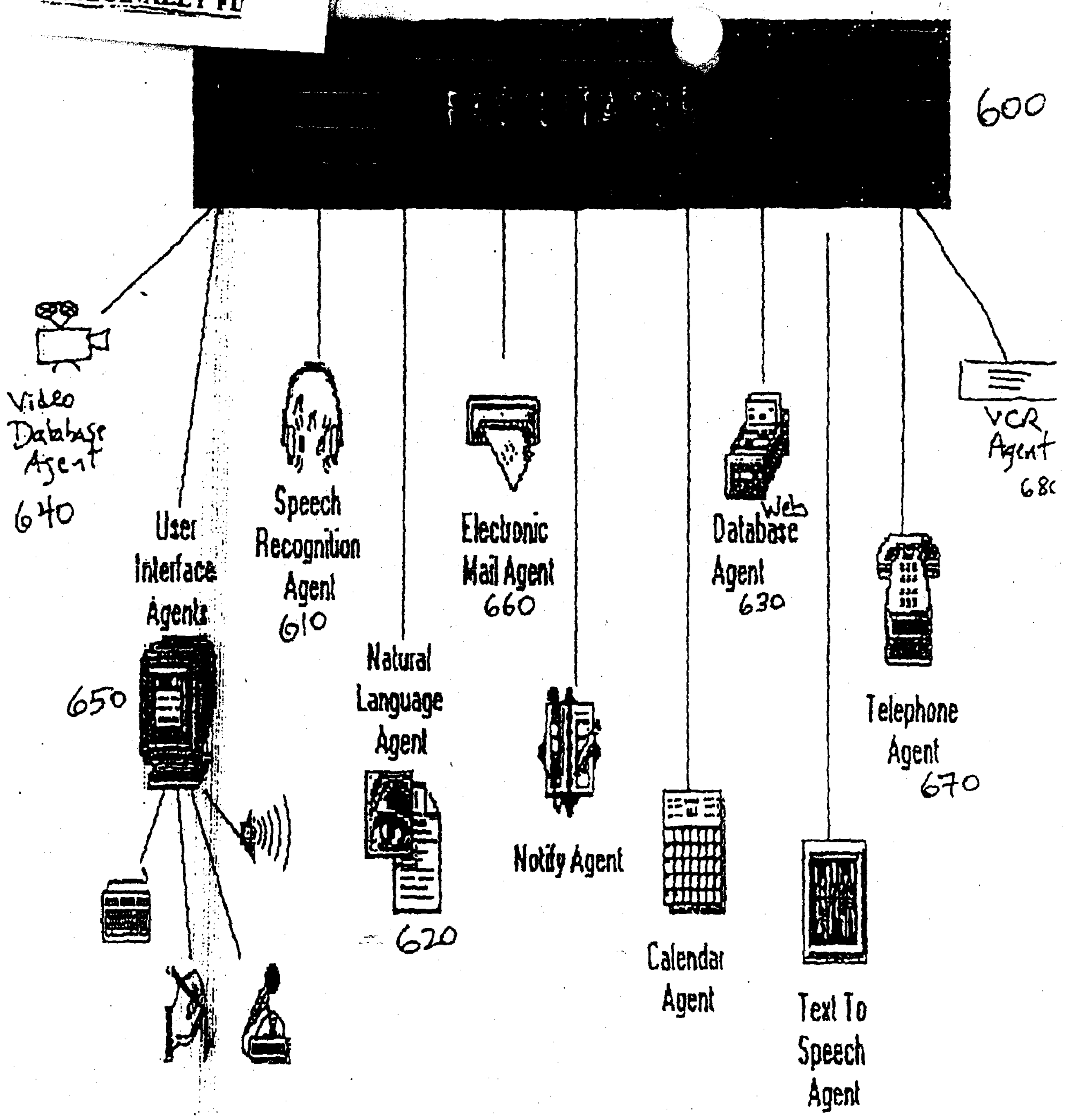

600
Video Database Agent
640
User Interface Agents
650
Speech Recognition Agent
610
Natural Language Agent
620
Electronic Mail Agent
660
Notify Agent
Calendar Agent
Web Database Agent
630
Text To Speech Agent
Telephone Agent
670
VCR Agent
680

Fig. 6

07-03-00

IN THE UNI ED STATES PATENT AND TRADEMARK OFFICE

A

CERTIFICATE OF EXPRESS MAILING

I hereby certify that this paper and the documents and/or fees referred to as attached therein are being deposited with the United States Postal Service on June 30, 2000 in an envelope as "Express Mail Post Office to Addressee" service under 37 CFR §1.10, Mailing Label Number EK858788226US, addressed to the Assistant Commissioner for Patents, Washington, DC 20231.

Kevin J. Zilka

Attorney Docket No.: SRI1P037C

First Named Inventor:
HALVERSEN, Christine

UTILITY PATENT APPLICATION TRANSMITTAL (37 CFR. § 1.53(b))

(Continuation, Divisional or Continuation-in-part application)

Assistant Commissioner for Patents
Box Patent Application
Washington, DC 20231

□ Duplicate for fee processing

Sir: This is a request for filing a patent application under 37 CFR. § 1.53(b) in the name of inventors: Christine Halversen, Luc Julia, Dimitris Voutsas, Adam Cheyer

For: SYSTEM, METHOD, AND ARTICLE OF MANUFACTURE FOR AGENT-BASED NAVIGATION IN A SPEECH-BASED DATA NAVIGATION SYSTEM

This application is a ☒ Continuation □ Divisional □ Continuation-in-part

of prior Application No.: **09/524,095**, from which priority under 35 U.S.C. §120 is claimed.

Application Elements:

☒ 33 Pages of Specification, Claims and Abstract
☒ 07 Sheets of Drawings
□ Declaration
□ Newly executed (original or copy)
□ Copy from a prior application (37 CFR 1.63(d) for a continuation or divisional). The entire disclosure of the prior application from which a copy of the declaration is herein supplied is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.
□ Deletion of inventors Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b).

Accompanying Application Parts:

□ Assignment and Assignment Recordation Cover Sheet (recording fee of $40.00 enclosed)
□ Power of Attorney
□ 37 CFR 3.73(b) Statement by Assignee

A

□ Information Dis re Statement with Form PTO-1449 □ Copies of IDS Citations

☒ Preliminary Amendment

☒ Return Receipt Postcard

□ Small Entity Statement(s) ☒ Statement filed in prior application. Status still proper and desired.

□ Other:

Claim For Foreign Priority

□ Priority of ________ Application No. __________ filed on __________ is claimed under 35 U.S.C. § 119.

□ The certified copy has been filed in prior application U.S. Application No. _______

□ The certified copy will follow.

Extension of Time for Prior Pending Application

□ A Petition for Extension of Time is being concurrently filed in the prior pending application. A copy of the Petition for Extension of Time is attached.

09507452.053000

Amendments

□ Amend the specification by inserting before the first line the sentence: "This is a

□ Continuation □ Continuation-in-part □ Divisional

application of copending prior

□ Application No.__________ filed on _____________,

□ International Application _________ filed on _________ which designated the United States,

the disclosure of which is incorporated herein by reference."

☒ Cancel in this application original claims __2-55__ of the prior application before calculating the filing fee. (*At least one original independent claim must be retained.*)

Fee Calculation (37 CFR § 1.16)

	(Col. 1) NO. FILED	(Col. 2) NO. EXTRA	SMALL ENTITY RATE	 FEE	OR	LARGE ENTITY RATE	 FEE
BASIC FEE			$345	$ 345	OR	$690	$
TOTAL CLAIMS	21 -20 =	1	x09 =	$ 09	OR	x18 =	$
INDEP CLAIMS	03 -03 =	0	x39 =	$	OR	x78 =	$
[ ] Multiple Dependent Claim Presented			$130 =	$ 354	OR	$260 =	$
* If the difference in Col. 1 is less than zero, enter "0" in Col. 2.			Total	$	OR	Total	$

□ Check No. 138 in the amount of $ 345.00 is enclosed.

A

☒ The Commissioner is aut .ed to charge any fees beyond the amou ıclosed which may be required, or to credit any overpayment, to Deposit Account No. 50-1351 (Order No. SRI1P037C).

General Authorization for Petition for Extension of Time (37 CFR §1.136)

☒ Applicants hereby make and generally authorize any Petitions for Extensions of Time as may be needed for any subsequent filings. The Commissioner is also authorized to charge any extension fees under 37 CFR §1.17 as may be needed to Deposit Account No. 50-1351 (Order No. SRI1P037C).

☒ Please send correspondence to the following address:

Kevin J. Zilka
P.O. BOX 721030
San Jose, California 95172-1030

Direct Telephone Calls To: **Kevin J. Zilka at telephone number (408) 505-5100**

Date: June 30, 2000

Kevin J. Zilka
Registration No. 41,429

A

PATENT APPLICATION SERIAL NO. ____________________

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
FEE RECORD SHEET

07/10/2000 MGORDON 00000044 09607672

01 FC:201	345.00 OP
02 FC:203	9.00 OP

PTO-1556
(5/87)

*U.S. GPO: 1999-459-082/19144

PATENT APPLICATION FEE DETERMINATION RECORD
Effective December 29, 1999

Application or Docket Number

CLAIMS AS FILED - PART I

SMALL ENTITY TYPE □ OR OTHER THAN SMALL ENTITY

	(Column 1)	(Column 2)					
FOR	NUMBER FILED	NUMBER EXTRA	RATE	FEE		RATE	FEE
BASIC FEE	▨	▨	▨	345.00	OR	▨	690.00
TOTAL CLAIMS	21 minus 20=	* 1	X$ 9=	9	OR	X$18=	
INDEPENDENT CLAIMS	3 minus 3 =	*	X39=		OR	X78=	
MULTIPLE DEPENDENT CLAIM PRESENT			+130=		OR	+260=	
			TOTAL	354	OR	TOTAL	

* If the difference in column 1 is less than zero, enter "0" in column 2

CLAIMS AS AMENDED - PART II

SMALL ENTITY OR OTHER THAN SMALL ENTITY

AMENDMENT A	(Column 1)		(Column 2)	(Column 3)					
	CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE	ADDITIONAL FEE		RATE	ADDITIONAL FEE
Total	*	Minus	**	=	X$ 9=		OR	X$18=	
Independent	*	Minus	***	=	X39=		OR	X78=	
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					+130=		OR	+260=	
					TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

AMENDMENT B	(Column 1)		(Column 2)	(Column 3)					
	CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE	ADDITIONAL FEE		RATE	ADDITIONAL FEE
Total	*	Minus	**	=	X$ 9=		OR	X$18=	
Independent	*	Minus	***	=	X39=		OR	X78=	
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					+130=		OR	+260=	
					TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

AMENDMENT C	(Column 1)		(Column 2)	(Column 3)					
	CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE	ADDITIONAL FEE		RATE	ADDITIONAL FEE
Total	*	Minus	**	=	X$ 9=		OR	X$18=	
Independent	*	Minus	***	=	X39=		OR	X78=	
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					+130=		OR	+260=	
					TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20."
***If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3."
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

FORM PTO-875
(Rev. 12/99)

Patent and Trademark Office, U.S. DEPARTMENT OF COMMERCE

*U.S. GPO: 2000-463-433/29044

FORMALITIES LETTER

OC000000005341790

UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office
Address: COMMISSIONER OF PATENT AND TRADEMARKS
Washington, D.C. 20231

APPLICATION NUMBER	FILING/RECEIPT DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NUMBER
09/607,672	06/30/2000	Christine Halversen	SRI1P037C

Kevin J Zilka
P O Box 721030
San Jose, CA 95172-1030

Date Mailed: 08/22/2000

NOTICE TO FILE MISSING PARTS OF NONPROVISIONAL APPLICATION

FILED UNDER 37 CFR 1.53(b)

Filing Date Granted

An application number and filing date have been accorded to this application. The item(s) indicated below, however, are missing. Applicant is given TWO MONTHS from the date of this Notice within which to file all required items and pay any fees required below to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

- The oath or declaration is missing.
  A properly signed oath or declaration in compliance with 37 CFR 1.63, identifying the application by the above Application Number and Filing Date, is required.
- To avoid abandonment, a late filing fee or oath or declaration surcharge as set forth in 37 CFR 1.16(e) of $65 for a small entity in compliance with 37 CFR 1.27, must be submitted with the missing items identified in this letter.
- **The balance due by applicant is $ 65.**

A copy of this notice <u>MUST</u> be returned with the reply.

A. Legesse
Customer Service Center
Initial Patent Examination Division (703) 308-1202

PART 3 - OFFICE COPY

#3

DECLARATION AND POWER OF ATTORNEY FOR ORIGINAL U.S. PATENT APPLICATION

Attorney's Docket No. SRI1P037

As a below-named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe that I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled: NAVIGATING NETWORK-BASED ELECTRONIC INFORMATIN USING SPOKEN NATURAL LANGUAGE INPUT WITH MULTIMODAL ERROR FEEDBACK, the specification of which,

(check one)

1. □ is attached hereto.

2. ☒ was filed on March 13, 2000 as
U.S. Application Serial No. 09/524,095
and was amended on __________.

3. □ was filed on __________ as
International PCT Application Serial No. __________
and was amended on __________.

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, CFR § 1.56.

I hereby claim foreign priority benefits under Title 35, United States code, § 119(a)-(d) or § 365(b) of any foreign application(s) for patent or inventor's certificate, or § 365(a) of any PCT International application which designated at least one country other than the United States, listed below and have identified below, by checking the box, any foreign application for patent or inventor's certificate, or PCT International application having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)			Priority Benefits Claimed?
(Appl. No.)	(Country)	(Filing Date)	□Yes □No
(Appl. No.)	(Country)	(Filing Date)	□Yes □No
(Appl. No.)	(Country)	(Filing Date)	□Yes □No

I hereby claim the benefit under 35 U.S.C. §119(e) of any United States provisional application(s) listed below:

__________ (Application Serial No.) __________ (Filing Date)

__________ (Application Serial No.) __________ (Filing Date)

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s), or § 365(c) of any PCT International application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

Prior U.S. Application(s)

(Application Serial No.)	(Filing Date)	(Status - patented, pending, abandoned)
(Application Serial No.)	(Filing Date)	(Status - patented, pending, abandoned)

And I hereby appoint the law firm of Hickman Stephens Coleman & Hughes, including Paul L. Hickman (Reg. No. 28,516); L. Keith Stephens (Reg. No. 32,632); Brian R. Coleman (Reg. No. 39,145); Michael J. Hughes (Reg. No. 29,077); Michael E. Melton (Reg. No. 32,276); Raymond E. Roberts (Reg. No. 38,597); Vidya R. Bhakar (Reg. No. 42,323); Larry B. Guernsey (Reg. No. 40,008); Douglas E. Mackenzie (Reg. No. 38,955); Michael D. Plimier (Reg. No. 43,004); Ronald R. Feece (Reg. No. P46,327); Stefanie M. Howell (Reg. No. P45,929); and Robert D. Hayden (Reg. No. 42,645) as my principal attorneys to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith:

Send Correspondence To:

HICKMAN STEPHENS COLEMAN & HUGHES, LLP
P.O. BOX 52037
Palo Alto, California 94303-0746

Direct Telephone Calls To: Raymond E. Roberts at telephone number (408) 558-9950

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

Typewritten Full Name of Sole or First Inventor: Christine Halverson   Citizenship: USA

Inventor's signature: Christine Halverson   Date of Signature: 6-16-00

Residence: (City) San Jose   (State/Country) California/USA

Post Office Address: 1623 Fairorchard Avenue, San Jose, California 95125

Full Name of Second Joint Inventor (if any): Luc Julia   Citizenship: ~~USA~~ FRANCE

Inventor's signature: Date of Signature: 6.21.00

Residence: (City) Menlo Park   (State/Country) California/USA

Post Office Address: 607 Menlo Avenue, Menlo Park, California 94025

Full Name of Third Joint Inventor (if any): Dimitris Voutsas   Citizenship: Greece

Inventor's signature: Date of Signature: 6/16/00

Residence: (City) Thessaloniki   (State/Country) Greece

Post Office Address: 14 M. Pyrza Street, Neoi Epivates, Thessaloniki 57019, Greece

Full Name of Fourth Joint Inventor (if any): Adam Cheyer

Citizenship: USA

Inventor's signature:

Date of Signature: 6/22/00

Residence: (City) Palo Alto

(State/Country) California /USA

Post Office Address: 757 Cereza Drive, Palo Alto, California 94306

SECTOR $
✓

FORMALITIES LETTER

OC000000005341790

UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office
Address: COMMISSIONER OF PATENT AND TRADEMARKS
Washington, D.C. 20231

APPLICATION NUMBER	FILING/RECEIPT DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NUMBER
09/607,672	06/30/2000	Christine Halversen	SRI1P037C

Kevin J Zilka
P O Box 721030
San Jose, CA 95172-1030

Date Mailed: 08/22/2000

NOTICE TO FILE MISSING PARTS OF NONPROVISIONAL APPLICATION

FILED UNDER 37 CFR 1.53(b)

Filing Date Granted

An application number and filing date have been accorded to this application. The item(s) indicated below, however, are missing. Applicant is given TWO MONTHS from the date of this Notice within which to file all required items and pay any fees required below to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

- The oath or declaration is missing.
  A properly signed oath or declaration in compliance with 37 CFR 1.63, identifying the application by the above Application Number and Filing Date, is required.
- To avoid abandonment, a late filing fee or oath or declaration surcharge as set forth in 37 CFR 1.16(e) of $65 for a small entity in compliance with 37 CFR 1.27, must be submitted with the missing items identified in this letter.
- **The balance due by applicant is $ 65.**

A copy of this notice MUST be returned with the reply.

A. Legesse
Customer Service Center
Initial Patent Examination Division (703) 308-1202

PART 2 - COPY TO BE RETURNED WITH RESPONSE

11/03/2000 HNOOR1 00000068 09607672
01 FC:205 65.00 OP

#4

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the application of.)
) Examiner: Not Assigned
Halverson et al.)
) Art Unit: 2758
Application No. 09/607,672)
) Atty. Docket No. SRI1P037C
Filed: June 30, 2000)
) Date: October 30, 2000
For: SYSTEM, METHOD AND ARTICLE OF-)
MANUFACTURE FOR AGENT-BASED)
NAVIGATION IN A SPEECH-BASED)
DATA NAVIGATION SYSTEM)

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231 on October 30, 2000.

Signed: Julie Curts
Julie A. Curts

RESPONSE TO NOTICE TO FILE MISSING PARTS

Assistant Commissioner for Patents
Box: Missing Parts
Washington, D.C. 20231

Sir:

In response to the Notice to File Missing Parts of Application--Filing Date Granted dated August 22, 2000, Applicants hereby attach an original executed Declaration and Power of Attorney, and the copy of the Notice to be returned with this response.

Applicants are also attaching Check No. 236 for $120.00 in payment of the surcharge fee and one month extension of time fee. The Commissioner is authorized to charge any other fees that may be due to our Deposit Account No. 50-1351 (Order No. SRI1P037C). A copy of this sheet is enclosed for this purpose.

Respectfully submitted,
SILICON VALLEY IP LAW GROUP

Kevin J. Zilka
Reg. No. 41,429

P.O. Box 721030
San Jose, CA 95172-1030
(408) 505-5100

Attorney Docket No. SRI1P037C

#5/A
2-13-01
B.Hilliard

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the application of)
) Docket:
Christine HALVERSEN et al.) SRI1P037C
)
Application No. 09/524,095)
)
Filed: March 13, 2000)
) Date: June 30, 2000
For: NAVIGATING NETWORK BASED)
ELECTRONIC INFORMATION USING SPOKEN)
NATURAL LANGUAGE INPUT WITH MULTIMODAL)
ERROR FEEDBACK)

Preliminary Amendment

Assistant Commissioner for Patents
and Trademarks
Washington, DC 20231

Dear Sir:

In regard to the above-named patent application, please enter the following amendments.

IN THE TITLE:

Please delete "NAVIGATING NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN NATURAL LANGUAGE INPUT WITH MULTIMODAL ERROR FEEDBACK", and insert therefore, --SYSTEM, METHOD, AND ARTICLE OF MANUFACTURE FOR AGENT-BASED NAVIGATION IN A SPEECH-BASED DATA NAVIGATION SYSTEM--

A1

IN THE ABSTRACT:

Please delete the Abstract and insert therefore: A system, method, and article of manufacture are provided for navigating an electronic data source by means of spoken language

A2

27

A

A2

where a portion of the data link between a mobile information appliance of the user and the data source utilizes wireless communication. When a spoken input request is received from a user, it is interpreted. The resulting interpretation of the request is thereupon used to automatically construct an operational navigation query. The navigation query is routed to one or more agents, which use the navigation query to retrieve the desired information from one or more electronic network data sources.--

IN THE SPECIFICATION:

On page 1, line 5, please delete "This is" and insert therefore, --This application is a continuation of an application entitled NAVIGATING NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN NATURAL LANGUAGE INPUT WITH MULTIMODAL ERROR FEEDBACK which was filed on March 13, 2000 under serial number 09/524,095 and which is--

A3

Please delete page 3, lines 3 to 15, and insert therefore, --The present invention addresses the above needs by providing a system, method, and article of manufacture for using agents for navigation of network-based electronic data sources in response to spoken input requests. When a spoken input request is received from a user, it is interpreted, such as by using a speech recognition agent to extract speech data from acoustic voice signals, and using a language parsing agent to linguistically parse the speech data. The interpretation of the spoken request can be performed on a computing device locally with the user, such as the mobile information appliance, or remotely from the user. The resulting interpretation of the request is thereupon used to automatically construct an operational navigation query. The navigation query is routed to one or more agents that use the navigation query to retrieve the desired information from one or more electronic network data sources, which is then transmitted to a client device of the user. If the network data source is a database, the navigation query is constructed in the format of a database query language.--

A4

IN THE CLAIMS:

Please delete claims 1-55, and insert therefore the following claims 1-21:

R.126(b) B1

56 ~~1.~~ (New) A method for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:

A5

28

A

Sub B1

(a) receiving a spoken request for desired information from a user;

(b) rendering an interpretation of the spoken request;

(c) constructing a navigation query based upon the interpretation;

(d) routing the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source; and

(e) invoking a user interface agent for outputting the selected portion of the electronic data source to the user.

2 ~~57~~ ~~2~~. (New) The method of claim 1 ~~56~~ ~~1~~, wherein an agent renders the interpretation of the spoken request.

~~58~~ ~~3~~. ~~(New) The method of claim 56 1, wherein a facilitator manages data flow among multiple agents.~~

3 ~~59~~ ~~4~~. (New) The method of claim 1 ~~56~~ ~~1~~, wherein the step of rendering the interpretation of the spoken request is performed by a speech recognition agent and a parsing agent.

4 ~~60~~ ~~5~~. (New) The method of claim 1 ~~56~~ ~~1~~, further comprising the steps of soliciting additional input from the user, including user interaction in a modality different than the original request; and refining the navigation query, based upon the additional input; wherein the at least one agent uses the refined navigation query to select a portion of the electronic data source.

A5

5 ~~61~~ ~~6~~. (New) The method of claim 4 ~~60~~ ~~5~~, wherein agents are utilized for performing the steps of soliciting additional input from the user and refining the navigation query.

6 ~~62~~ ~~7~~. (New) The method of claim 1 ~~56~~ ~~1~~, wherein the electronic data source is a web page, wherein the at least one agent scrapes the web page for selecting a portion of the web page.

Sub B2

63 ~~8~~. ~~(New) A computer~~ program embodied on a computer readable medium for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:

29

4

Sub B2

~~(a) a code segment that receives a spoken request for desired information from a user;~~

~~(b) a code segment that renders an interpretation of the spoken request;~~

~~(c) a code segment that constructs a navigation query based upon the interpretation;~~

~~(d) a code segment that routes the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source; and~~

~~(e) a code segment that invokes a user interface agent for outputting the selected portion of the electronic data source to the user.~~

A5

8 ~~64~~ ~~9~~. (New) The computer program of claim 7 ~~63~~ ~~8~~, wherein the code segment that renders the interpretation of the spoken request is executed by an agent.

65 ~~10~~. (New) The computer program of claim 63 ~~8~~, wherein a facilitator manages data flow among multiple agents.

9 ~~66~~ ~~11~~. (New) The computer program of claim 7 ~~63~~ ~~8~~, wherein a speech recognition agent and a parsing agent execute the code segment that renders the interpretation of the spoken request.

10 ~~67~~ ~~12~~. (New) The computer program of claim 7 ~~63~~ ~~8~~, further comprising a code segment that solicits additional input from the user, including user interaction in a modality different than the original request; and a code segment that refines the navigation query, based upon the additional input; wherein the at least one agent uses the refined navigation query to select a portion of the electronic data source.

11 ~~68~~ ~~13~~. (New) The computer program of claim 10 ~~67~~ ~~12~~, wherein a solicitor agent executes the code segment that solicit the additional input from the user and a refining agent executes the code segment that refines the navigation query.

30

A

12 ~~09~~ ~~14~~. (New) The computer program of claim 7 ~~03~~ ~~8~~, wherein the electronic data source is a web page, wherein the at least one agent scrapes the web page for selecting a portion of the web page.

A5

70 ~~15~~. (New) A system for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:

(a) a client device, operable to receive a spoken request for desired information from a user;

(b) spoken language processing logic, operable to render an interpretation of the spoken request;

(c) query construction logic, operable to construct a navigation query based upon the interpretation;

(d) routing logic, operable to route the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source; and

(e) invoking logic, operable to invoke a user interface agent for outputting the selected portion of the electronic data source to the user.

14 ~~21~~ ~~16~~. (New) The system of claim 13 ~~20~~ ~~15~~, wherein the query construction logic that renders the interpretation of the spoken request is executed by an agent.

~~22~~ ~~17~~. (New) The system of claim ~~20~~ ~~15~~, wherein a facilitator manages data flow among multiple agents.

15 ~~23~~ ~~18~~. (New) The system of claim 13 ~~20~~ ~~15~~, wherein a speech recognition agent and a parsing agent execute the spoken language processing logic that renders the interpretation of the spoken request.

4

16 ~~74~~ ~~19~~. (New The system of claim 13 ~~70~~ ~~15~~, further comprising user interaction logic operable to solicit additional input from the user, including user interaction in a modality different than the original request; and query refining logic operable to refine the navigation query, based upon the additional input; wherein the at least one agent uses the refined navigation query to select a portion of the electronic data source.

A5

17 ~~75~~ ~~20~~. (New) The system of claim 16 ~~74~~ ~~19~~, wherein a solicitor agent executes the user interaction logic and a refining agent executes the query refinement logic.

18 ~~76~~ ~~21~~. (New) The system of in claim 13 ~~70~~ ~~15~~, wherein the electronic data source is a web page, wherein the at least one agent scrapes the web page for selecting a portion of the web page.

In the event a telephone conversation would expedite the prosecution of this application, the Examiner may reach the undersigned at (408) 505-5100. If any fees are due in connection with the filing of this paper, then the Commissioner is authorized to charge such fees to Deposit Account No. 50-1351 (Order No. SRI1P037C). A duplicate copy of the transmittal is enclosed for this purpose.

Respectfully submitted,

Kevin J. Zilka
Registration No. 41,429

P.O. Box 721030
San Jose, CA 95172
Telephone: (408) 505-5100

32

A

GAU ~~2758~~ #6
2154

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the application of:

Halverson et al.

Application No. 09/607,672

Filed: 06/30/2000

For: SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR AGENT-BASED NAVIGATION IN A SPEECH-BASED DATA NAVIGATION SYSTEM

Group Art Unit: 2758

Examiner: Unassigned

Atty. Docket No. SRI1P037C/
44454/03451

Date: April 27, 2001

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, DC 20231 on April 27, 2001

Signed: Erica L. Mann
Erica L. Mann

INFORMATION DISCLOSURE STATEMENT
UNDER 37 CFR §§ 1.56 AND 1.97(c)

Assistant Commissioner for Patents
Washington, DC 20231

Dear Sir:

The references listed in the attached PTO Form 1449, copies of which are attached, may be material to examination of the above-identified patent application. Applicants submit these references in compliance with their duty of disclosure pursuant to 37 CFR §§ 1.56 and 1.97. The Examiner is requested to make these references of official record in this application.

This Information Disclosure Statement is not to be construed as a representation that a search has been made, that additional information material to the examination of this application does not exist, or that these references indeed constitute prior art.

This Information Disclosure Statement is believed to be filed before the mailing date of a first Office Action on the merits. Accordingly, it is believed that no fees are due in connection with the filing of this Information Disclosure Statement. However, if it is determined that any fees are due, the Commissioner is hereby authorized to charge such fees to Deposit Account 03-0683 (Order No. 44454/03451/SRI1P037C).

Respectfully submitted,
CARLTON FIELDS

Dominic M. Kotab
Reg. No. 42,762

P.O. Box 721030
San Jose, CA 95172-1030
Telephone: (408) 271-2300

Form 1449 (Modified) **Information Disclosure Statement By Applicant** (Use Several Sheets if Necessary)	Atty. Docket No. SRI1P037C Applicant: Halverson et al. Filing Date: 06/30/2000
	Application No.: 09/607,672 Group Art Unit: ~~2758~~ 2155

U.S. Patent Documents

Examiner Initial	No.	Patent No.	Date	Patentee	Class	Sub-class	Filing Date
	A	6,026,388	02/15/00	Liddy et al.	707	1	08/14/96
	B	~~6,102,030~~ 6012030	01/04/00	French- St. George et al.	704	275	04/21/98
	C	6,003,072	12/14/99	Gerritsen et al.	709	218	06/30/94
	D	5,890,123	03/30/99	Brown et al.	704	275	06/05/95
	E	5,855,002	12/29/98	Armstrong	704	270	06/11/96
	F	5,963,940	10/05/99	Liddy et al.	707	5	08/14/96
	G	5,805,775	09/08/98	Eberman et al.	395	12	02/02/96
	H	5,802,526	09/01/98	Fawcett et al.	707	104	04/18/96
	I	5,794,050	08/11/98	Dahlgren et al.	395	708	10/02/97
	J	5,774,859	06/30/98	Houser et al.	704	275	01/03/95
	K	5,748,974	05/05/98	Johnson	395	759	12/13/94

6,012,030

Foreign Patent or Published Foreign Patent Application

Examiner Initial	No.	Document No.	Publication Date	Country or Patent Office	Class	Sub-class	Translation Yes	Translation No
	L							
	M							
	N							
	O							
	P							

Other Documents

Examiner Initial	No.	Author, Title, Date, Place (e.g. Journal) of Publication
	R	Stent, Amanda et al., "The CommandTalk Spoken Dialogue System", SRI International
	S	Moore, Robert et al., "CommandTalk: A Spoken-Language Interface for Battlefield Simulations", October 23, 1997, SRI International
	T	Dowding, John et al., "Interpreting Language in Context in CommandTalk", February 5, 1999, SRI International

Examiner	Date Considered 7/30/2001

Examiner: Initial citation considered. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Form 1449 (Modified) **Information Disclosure Statement By Applicant** (Use Several Sheets if Necessary)	Atty. Docket No. SRI1P037C Applicant: Halverson et al. Filing Date: 06/30/2000	Application No.: 09/607,672 Group Art Unit: ~~2758~~ 2155

U.S. Patent Documents

Examiner Initial	No.	Patent No.	Date	Patentee	Class	Sub-class	Filing Date
	A	5,729,659	03/17/98	Potter	395	2.79	06/06/95
	B	5,721,938	02/24/98	Stuckey	395	754	06/07/95
	C	5,608,624	03/04/97	Luciw	395	794	05/15/95
	D	5,519,608	05/21/96	Kupiec	364	419.08	06/24/93
	E	5,434,777	07/18/95	Luciw	364	419.13	03/18/94
	F	5,386,556	01/31/95	Hedin et al.	395	600	12/23/92
	G	5,197,005	03/23/93	Shwartz et al.	364	419	05/01/89
	H						
	I						
	J						
	K						

Foreign Patent or Published Foreign Patent Application

Examiner Initial	No.	Document No.	Publication Date	Country or Patent Office	Class	Sub-class	Translation Yes	Translation No
	L							
	M							
	N							
	O							
	P							

Other Documents

Examiner Initial	No.	Author, Title, Date, Place (e.g. Journal) of Publication
	R	http://www.ai.sri.com/~oaa/infowiz.html, "InfoWiz: An Animated Voice Interactive Information System, May 8, 2000
	S	Dowding, John, "Interleaving Syntax and Semantics in an Efficient Bottom-up Parser", SRI International
	T	Moore, Robert et al., "Combining Linguistic and Statistical Knowledge Sources in Natural-Language Processing for ATIS", SRI International

Examiner	Date Considered 7/30/2001

Examiner: Initial citation considered. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Form 1449 (Modified) **Information Disclosure Statement By Applicant** (Use Several Sheets if Necessary)	Atty. Docket No. SRI1P037C Applicant: Halverson et al. Filing Date: 06/30/2000	Application No.: 09/607,672 Group Art Unit: ~~2758~~ 2155

U.S. Patent Documents

Examiner Initial	No.	Patent No.	Date	Patentee	Class	Sub-class	Filing Date
	A						
	B						
	C						
	D						
	E						
	F						
	G						
	H						
	I						
	J						
	K						

Foreign Patent or Published Foreign Patent Application

Examiner Initial	No.	Document No.	Publication Date	Country or Patent Office	Class	Sub-class	Translation Yes	Translation No
	L							
	M							
	N							
	O							
	P							

Other Documents

Examiner Initial	No.	Author, Title, Date, Place (e.g. Journal) of Publication
M	R	Dowding, John et al., "Gemini: A Natural Language System For Spoken-Language Understanding", SRI International
	S	
	T	

Examiner	Date Considered 7/30/2001

Examiner: Initial citation considered. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

GAU - 2157

Attorney Docket No.: SRI1P037C

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF:	HALVERSON, CHRISTINE
SERIAL NO.:	09/607,672
FILED:	6/30/00
TITLE:	SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR AGENT-BASED NAVIGATION IN A SPEECH-BASED DATA NAVIGATION SYSTEM

ASSOCIATE POWER OF ATTORNEY

Assistant Commissioner for Patents
Washington, DC 20231

Dear Sir:

I hereby appoint: C. Douglas McDonald (Reg. No. 26,659)

whose post office address is

Carlton Fields, P.A.
P. O. Box 3239
Tampa, Florida 33601-3239

as my associate attorney in the above-entitled application, to prosecute this application, to make alterations and amendments therein, and to transact all business in the Patent and Trademark Office connected therewith.

Please continue to address all future communications to:

Carlton Fields, LLP
P. O. Box 721030
San Jose, CA 95172-1030

Respectfully submitted

Date: MAY 7, 2001

Kevin J. Zilka (Reg. No. 41,429)
Dominic Kotab (Reg. No. 42,762)
Carlton Fields LLP
P.O. Box 721030
San Jose, CA 95172-1030
Telephone: (408) 271-2300
Fax: (408) 275-9579

TPA#1680358.01

#8

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION NO.: 09/607,672
INVENTOR: Halverson, Christine
TITLE: SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR AGENT-BASED NAVIGATION IN A SPEECH-BASED DATA NAVIGATION SYSTEM

FILING DATE: 6/30/00
ATTORNEY DOCKET NO. SRI1P037C

NOTICE OF CHANGE OF CORRESPONDENCE ADDRESS

Assistant Commissioner for Patents
Washington, DC 20231

Sir:

Please change the correspondence address relating to the above-identified application as follows:

C. Douglas McDonald , Esq.
Carlton Fields, et al.
P.O. Box 3239
Tampa, FL 33601-3239

Respectfully submitted,

C. Douglas McDonald

Date: May 10, 2001

C. Douglas McDonald
Reg. No. 26,659
CARLTON FIELDS, P.A.
P.O. Box 3239
Tampa, FL 33601-3239
(813) 223-7000
Attorney of Record

TPA#1524975.01

UNITED STATES ·ARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR		ATTORNEY DOCKET NO.
09/607,672	06/30/00	HALVERSEN	C	SRI1P037C

TM02/0827

C. DOUGLAS MCDONALD, ESQ.
CARLTON FIELDS, ET AL.
P.O. BOX 3239
TAMPA, FL 33601-3239

EXAMINER	
LEE,T	
ART UNIT	PAPER NUMBER
2155	9

DATE MAILED:
08/27/01

Please find below and/or attached an Office communication concerning this application or proceeding.

Commissioner of Patents and Trademarks

PTO-90C (Rev.11/00)

1- File Copy

Office Action Summary	**Application No.** 09/607,672	**Applicant(s)** HALVERSEN ET AL.	
	Examiner Tammy T. Lee	**Art Unit** 2155	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 30 June 2000.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 56-76 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 56-76 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)☐ The proposed drawing correction filed on _____ is: a)☐ approved b)☐ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____ .

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) ☐ The translation of the foreign language provisional application has been received.

15)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449) Paper No(s) 6.
4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other:

Detailed Action

1. Claims 56-76 are presented for examination.

Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later

invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102((e), f) or (g) prior art under 35 U.S.C. 103(a).

4. Claims 56-76 are rejected under 35 U.S.C. § 103 (a) as being unpatentable over Sugikawa, U.S. patent no. 5,949,772 in view of Gibson, U.S. patent no. 6,052,716.

5. As per claim 56, Sugikawa teaches a method comprising steps of: (a) receiving a spoken request for desired information from a user (a service request data detection, col. 6, line 61); (b) rendering an interpretation of the spoken request (a judging means and human voices, col. 6, line 65 and col. 32, line 43); (c) constructing a query based upon the interpretation (a selection data communicating means, col. 7, line 3); (d) routing the query to at least one agent, wherein the at least one agent utilizes the query to select a portion of the electronic data source (a selection data communicating means, col. 7, line 3); and (e) invoking a user interface agent for outputting the selected portion of the electronic data source to the user (a deciding means, col. 7, line 13) (col. 6, line 56-col. 7, line 15).

Sugikawa teaches the invention substantially as claimed; however, Sugikawa does not disclose that a navigation query is being used in the system. Gibson discloses that a hierarchy of network addresses accessed by the network navigator during a network navigating session is compiled (col. 3, lines 33-36). It would have been obvious to one of ordinary skill in the art at the time of the invention to employ the teachings of Gibson within the system of Sugikawa because it will allow the user to search desired information efficiently.

6. As per claim 57, Sugikawa teaches an agent renders the interpretation of the spoken request (the layout is calculated based on active acoustic emissions such as the sound of a buzzer or music from the speaker means of the system in lieu of human voices, col. 32, lines 40-58).

7. As per claim 58, Sugikawa teaches a facilitator manages data flow among multiple agents (the communication controller unit, 102, fig 21 and col. 26, lines 1-48).

8. As per claim 59, Sugikawa teaches the step of rendering the interpretation of the spoken request is performed by a speech recognition agent and a parsing agent (col. 27, lines 4-53 and col. 32, lines 40-55).

9. As per claim 60, Sugikawa teaches the invention substantially as claimed; however, Sugikawa does not disclose the invention comprising the steps of soliciting additional input from the user, including user interaction in a modality different than the original request; and refining the navigation query, based upon the additional input; wherein the at least one agent uses the refined navigation query to select a portion of the electronic data source. Gibson discloses soliciting additional input from the user, including user interaction in a modality different than the original request; and refining the navigation query (an internet web navigation session is performed, col. 8, line 12), based upon the additional input; the at least one agent uses the refined navigation query to select a portion of the electronic data source (col. 7, line 65-col. 8, line 54). It would have been obvious to one of ordinary skill in the art at the time of the invention to employ the teachings of Gibson within the system of Sugikawa for refining the navigation query based

upon additional input because it will allow the user to retrieve and present associated electronic data in an effective way.

10. As per claim 61, Sugikawa teaches the invention substantially as claimed; however, Sugikawa does not disclose that agents are utilized for performing the steps of soliciting additional input from the user and refining the navigation query. Gibson discloses that agents are soliciting additional input from the user and refining the navigation query (col. 7, line 65-col. 8, line 54). It would have been obvious to one of ordinary skill in the art at the time of the invention to employ the teachings of Gibson within the system of Sugikawa for refining the navigation query based upon additional input because it will allow the user to retrieve and present associated electronic data in an effective way.

11. As per claim 62, Sugikawa teaches the invention substantially as claimed; however, Sugikawa does not disclose that the electronic data source is a web page, wherein the at least one agent scrapes the web page for selecting a portion of the web page. Gibson teaches that the electronic data source is a web page and the at least one agent scrapes the web page for selecting a portion of the web page (col. 8, line 55-col. 9, line 18). It would have been obvious to one of ordinary skill in the art at the time of the invention to employ the teachings of Gibson within the system of Sugikawa for scraping the web pages because it will allow the user to search desired information efficiently.

12. As per claim 63, Sugikawa teaches a computer program comprising the steps of:

(a) a code segment that receives a spoken request for desired information from a user (a service request data detection, col. 6, line 61); (b) a code segment that renders an interpretation of the spoken request (a judging means and human voices, col. 6, line 65 and col. 32, line 43); (c) a code segment that constructs a query based upon the interpretation (a selection data communicating means, col. 7, line 3); (d) a code segment that routes the query to at least one agent, wherein the at least one agent utilizes the query to select a portion of the electronic data source (a selection data communicating means, col. 7, line 3); and (e) a code segment that invokes a user interface agent for outputting the selected portions of the electronic data source to the user (a deciding means, col. 7, line 13) (col. 6, line 56-col. 7, line 15).

Sugikawa teaches the invention substantially as claimed; however, Sugikawa does not disclose that a navigation query is being used in the system. Gibson discloses that a hierarchy of network addresses accessed by the network navigator during a network navigating session is compiled (col. 3, lines 33-36). It would have been obvious to one of ordinary skill in the art at the time of the invention to employ the teachings of Gibson within the system of Sugikawa because it will allow the user to search desired information efficiently.

13. As per claim 64, Sugikawa teaches the code segment that renders the interpretation of the spoken request is executed by an agent (the layout is calculated based on active acoustic emissions such as the sound of a buzzer or music from the speaker means of the system in lieu of human voices, col. 32, lines 40-58).

14. As per claim 65, Sugikawa teaches a facilitator manages data flow among multiple agents (the communication controller unit, 102, fig 21 and col. 26, lines 1-48).

15. As per claim 66, Sugikawa teaches a speech recognition agent and a parsing agent execute the code segment that renders the interpretation of the spoken request (col. 27, lines 4-53 and col. 32, lines 40-55).

16. As per claim 67, Sugikawa teaches the invention substantially as claimed; however, Sugikawa does not disclose the invention comprising a code segment that solicits additional input from the user, including user interaction in a modality different than the original request; and a code segment that refines the navigation query, based upon the additional input; wherein the at least one agent uses the refined navigation query to select a portion of the electronic data source. Gibson discloses soliciting additional input from the user, including user interaction in a modality different than the original request; and refining the navigation query (an internet web navigation session is performed, col. 8, line 12), based upon the additional input; the at least one agent uses the refined navigation query to select a portion of the electronic data source (col. 7, line 65-col. 8, line 54). It would have been obvious to one of ordinary skill in the art at the time of the invention to employ the teachings of Gibson within the system of Sugikawa for refining the navigation query based upon additional input because it will allow the user to retrieve and present associated electronic data in an effective way.

17. As per claim 68, Sugikawa teaches the invention substantially as claimed; however, Sugikawa does not disclose that a solicitor agent executes the code segment that solicit the additional input from the user and a refining agent executes the code segment that refines the navigation query. Gibson discloses that agents are soliciting additional input from the user and refining the navigation query (col. 7, line 65-col. 8, line 54). It would have been obvious to one of ordinary skill in the art at the time of the invention to employ the teachings of Gibson within the system of Sugikawa for refining the navigation query based upon additional input because it will allow the user to retrieve and present associated electronic data in an effective way.

18. As per claim 69, Sugikawa teaches the invention substantially as claimed; however, Sugikawa does not disclose that the electronic data source is a web page, wherein the at least one agent scrapes the web page for selecting a portion of the web page. Gibson teaches that the electronic data source is a web page and the at least one agent scrapes the web page for selecting a portion of the web page (col. 8, line 55-col. 9, line 18). It would have been obvious to one of ordinary skill in the art at the time of the invention to employ the teachings of Gibson within the system of Sugikawa for scraping the web pages because it will allow the user to search desired information efficiently.

19. As per claim 70, Sugikawa teaches a system comprising the steps of: (a) a client device, operable to receive a spoken request for desired information from a user (a service request data detection, col. 6, line 61); (b) spoken language processing logic, operable to render an interpretation of the spoken request (a judging means and human voices, col. 6, line 65 and col.

32, line 43); (c) query construction logic, operable to construct a query based upon the interpretation (a selection data communicating means, col. 7, line 3); (d) routing logic, operable to route the query to at least one agent, wherein the at least one agent utilizes the query to select a portion of the electronic data source (a selection data communicating means, col. 7, line 3); and (e) invoking logic, operable to invoke a user interface agent for outputting the selected portions of the electronic data source to the user (a deciding means, col. 7, line 13) (col. 6, line 56-col. 7, line 15).

Sugikawa teaches the invention substantially as claimed; however, Sugikawa does not disclose that a navigation query is being used in the system. Gibson discloses that a hierarchy of network addresses accessed by the network navigator during a network navigating session is compiled (col. 3, lines 33-36). It would have been obvious to one of ordinary skill in the art at the time of the invention to employ the teachings of Gibson within the system of Sugikawa because it will allow the user to search desired information efficiently.

20. As per claim 71, Sugikawa teaches that query construction logic that renders the interpretation of the spoken request is executed by an agent (the layout is calculated based on active acoustic emissions such as the sound of a buzzer or music from the speaker means of the system in lieu of human voices, col. 32, lines 40-58).

21. As per claim 72, Sugikawa teaches a facilitator manages data flow among multiple agents (the communication controller unit, 102, fig 21 and col. 26, lines 1-48).

22. As per claim 73, Sugikawa teaches a speech recognition agent and a parsing agent execute the spoken request is execute the spoken language processing logic that renders the interpretation of the spoken request (col. 27, lines 4-53 and col. 32, lines 40-55).

23. As per claim 74, Sugikawa teaches the invention substantially as claimed; however, Sugikawa does not disclose that comprising user interaction logic operable to solicit additional input from the user, including user interaction in a modality different than the original request; and query refining logic operable to refine the navigation query, based upon the additional input; wherein the at least one agent uses the refined navigation query to select a portion of the electronic data source. Gibson discloses soliciting additional input from the user, including user interaction in a modality different than the original request; and refining the navigation query (an internet web navigation session is performed, col. 8, line 12), based upon the additional input; the at least one agent uses the refined navigation query to select a portion of the electronic data source (col. 7, line 65-col. 8, line 54). It would have been obvious to one of ordinary skill in the art at the time of the invention to employ the teachings of Gibson within the system of Sugikawa for refining the navigation query based upon additional input because it will allow the user to retrieve and present associated electronic data in an effective way.

24. As per claim 75, Sugikawa teaches the invention substantially as claimed; however, Sugikawa does not disclose that a solicitor agent executes the user interaction logic and a refining agent executes the query refining logic. Gibson discloses that a solicitor agent executes the user interaction logic and a refining agent executes the query refining logic (col. 7, line 65-

col. 8, line 54). It would have been obvious to one of ordinary skill in the art at the time of the invention to employ the teachings of Gibson within the system of Sugikawa for refining the navigation query based upon additional input because it will allow the user to retrieve and present associated electronic data in an effective way.

25. As per claim 76, Sugikawa teaches the invention substantially as claimed; however, Sugikawa does not disclose that the electronic data source is a web page, wherein the at least one agent scrapes the web page for selecting a portion of the web page. Gibson teaches that the electronic data source is a web page and the at least one agent scrapes the web page for selecting a portion of the web page (col. 8, line 55-col. 9, line 18). It would have been obvious to one of ordinary skill in the art at the time of the invention to employ the teachings of Gibson within the system of Sugikawa for scraping the web pages because it will allow the user to search desired information efficiently.

Conclusion

26. A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. Failure to respond within the period for response will result in Abandonment of the application (see 35 USC 133, MPEP 710.02, 710.02(b)).

27. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tammy T. Lee whose telephone number is 703-308-9119. The

examiner can normally be reached on Mon-Fri (9:30am-6:00pm). If attempts to reach the examiner by telephone is unsuccessful, the examiner's supervisor, Ayaz Sheikh can be reached on 703-305-9648. The fax phone numbers for the organization where this application or proceeding is 703-305-7201.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is 703-305-3900.

Tammy T. Lee
Patent Examiner
July 31, 2001

DAVID WILEY
PRIMARY EXAMINER

Notice of References Cited	Application/Control No. 09/607,672	Applicant(s)/Patent Under Reexamination HALVERSEN ET AL.	
	Examiner Tammy T. Lee	Art Unit 2155	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification	
	A	US-6052716 4-2000	05-1997	Gibson	709	217
	B	US-6026437 2-2000	04-1998	Muschett et al	709	219
	C	US-5902353 5-1999	07-1997	Reber et al	709	219
	D	US-5978848 11-1999	03-1999	Maddalozzo, Jr. et al	709	227
	E	US-5717860 2-1998	09-1995	Graber et al	395	200.12
	F	US-5949772 9-1999	11-1997	Sugikawa et al	370	331
	G	US-				
	H	US-				
	I	US-				
	J	US-				
	K	US-				
	L	US-				
	M	US-				

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification	
	N						
	O						
	P						
	Q						
	R						
	S						
	T						

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

SRI/4116-7

#10
LDJ
10-12-01

IN THE UNITED STATES
PATENT AND TRADEMARK OFFICE

PATENT APPLICATION

Applicant(s):	**HALVERSON, et al.**	Atty. Docket No.	**SRI 1P037C**
Serial No.:	**09/607,672**	Group Art Unit:	**2155**
Filed:	**June 30,.2000**	Examiner:	**T. Lee**

Title: **SYSTEM, METHOD, AND ARTICLE OF MANUFACTURE FOR AGENT-BASED NAVIGATION IN A SPEECH-BASED DATA NAVIGATION SYSTEM**

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

REVOCATION OF PREVIOUS POWER OF ATTORNEY AND NEW APPOINTMENT

The undersigned assignee of the above-identified application hereby revokes all previous Powers of Attorney and appoints the following attorneys with full power to prosecute the application, to make alterations and amendments therein, and to transact all business in the United States Patent and Trademark Office connected therewith and with full power of substitution and revocation:

Raymond R. Moser, Jr.; Reg. No. 34,682; Kin-Wah Tong, Reg. No. 39,400; Robert Brush, Reg. No. 45,710; Steven Weiner, Reg. No. 38,360; and Edward E. Davis, Reg. No. 35,112.

CHANGE OF CORRESPONDENCE ADDRESS

Please change the correspondence address for the above-identified application to:

Thomason, Moser & Patterson, LLP
595 Shrewsbury Avenue – Suite 100
Shrewsbury, New Jersey 07702

Please direct all telephone calls to: Kin-Wah Tong, telephone # (732) 530-9404

CERTIFICATE UNDER 37 C.F.R. § 3.73(B)

SRI International, a corporation of the State of California, certifies that it is the assignee of the entire right, title and interest in the patent application identified above by virtue of:

An Assignment from the inventor(s) of the parent patent application that is claimed as priority in the above-identified patent application. The Assignment was recorded in the United States Patent and Trademark Office, for which a copy thereof is attached.

The undersigned (whose title is supplied below) is empowered to act on behalf of the assignee.

Respectfully submitted,

Date: 9/11/01

~~EDWARD E. DAVIS, Assistant Secretary~~
STEVEN WEINER, VICE PRESIDENT

SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025
Telephone No.: 650-859-3115

ASSIGNMENT OF PATENT APPLICATION

(Not Accompanying Application)

Whereas I/we the undersigned inventor(s) have invented certain new and useful improvements as set forth in the patent application entitled:

NAVIGATING NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN NATURAL LANGUAGE INPUT WITH MULTIMODAL ERROR FEEDBACK

for which I/we have executed an application for a United States Letters Patent which was filed in the U.S. Patent and Trademark Office on March 13, 2000, and which bears the Application No. 09/524,095.

For good and valuable consideration, the receipt and sufficiency of which is hereby acknowledged, I/we the undersigned inventor(s) hereby:

1) Sell(s), assign(s) and transfer(s) to **SRI International**, a California non-profit corporation having a place of business at 333 Ravenswood Avenue, Menlo Park, California 94025, (hereinafter referred to as "ASSIGNEE"), the entire right title and interest in any and all improvements and inventions disclosed in, application(s) based upon, and Patent(s) (including foreign patents) granted upon the information which is disclosed in the above referenced application.

2) Authorize and request the Commissioner of Patents to issue any and all Letters Patents resulting from said application or any division(s), continuation(s), substitutes(s) or reissue(s) thereof to the ASSIGNEE.

3) Agree to execute all papers and documents and, entirely at the ASSIGNEE's expense, perform any acts which are reasonably necessary in connection with the prosecution of said application, as well as any derivative and applications thereof, foreign applications based thereon, and/or the enforcement of patents resulting from such applications.

4) Agree that the terms, covenants and conditions of this assignment shall inure to the benefit of the Assignee, its successors, assigns and other legal representative, and shall be binding upon the inventor(s), as well as the inventor's heirs, legal representatives and assigns.

5) Warrant and represent that I/we have not entered, and will not enter into any assignment, contract, or understanding that conflicts with this assignment.

Signed on the date(s) indicated beside my (our) signature(s).

1) Signature: Date: 6-16-00.
Typed Name: Christine Halverson

2) Signature: Date:
Typed Name: Luc Julia

3) Signature: Date: 6/16/00
Typed Name: Dimitris Voutsas

4) Signature: Date: 6/22/00
Typed Name: Adam Cheyer

Attny Docket No. SRI1P037

ASSIGNMENT OF PATENT APPLICATION

(Not Accompanying Application)

Whereas I/we the undersigned inventor(s) have invented certain new and useful improvements as set forth in the patent application entitled:

NAVIGATING NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN NATURAL LANGUAGE INPUT WITH MULTIMODAL ERROR FEEDBACK

for which I/we have executed an application for a United States Letters Patent which was filed in the U.S. Patent and Trademark Office on March 13, 2000, and which bears the Application No. 09/524,095.

For good and valuable consideration, the receipt and sufficiency of which is hereby acknowledged, I/we the undersigned inventor(s) hereby:

1) Sell(s), assign(s) and transfer(s) to **SRI International**, a California non-profit corporation having a place of business at 333 Ravenswood Avenue, Menlo Park, California 94025, (hereinafter referred to as "ASSIGNEE"), the entire right title and interest in any and all improvements and inventions disclosed in, application(s) based upon, and Patent(s) (including foreign patents) granted upon the information which is disclosed in the above referenced application.

2) Authorize and request the Commissioner of Patents to issue any and all Letters Patents resulting from said application or any division(s), continuation(s), substitutes(s) or reissue(s) thereof to the ASSIGNEE.

3) Agree to execute all papers and documents and, entirely at the ASSIGNEE's expense, perform any acts which are reasonably necessary in connection with the prosecution of said application, as well as any derivative and applications thereof, foreign applications based thereon, and/or the enforcement of patents resulting from such applications.

4) Agree that the terms, covenants and conditions of this assignment shall inure to the benefit of the Assignee, its successors, assigns and other legal representative, and shall be binding upon the inventor(s), as well as the inventor's heirs, legal representatives and assigns.

5) Warrant and represent that I/we have not entered, and will not enter into any assignment, contract, or understanding that conflicts with this assignment.

Signed on the date(s) indicated beside my (our) signature(s).

1) Signature: ______________ Date: 6-16-00.
Typed Name: Christine Halverson

2) Signature: ______________ Date: 6·20.00
Typed Name: Luc Julia

3) Signature: ______________ Date: 6/16/00
Typed Name: Dimitris Voutsas

4) Signature: ______________ Date: __________
Typed Name: Adam Cheyer

Attny Docket No. SRI1P037

ASSIGNMENT OF PATENT APPLICATION

(Not Accompanying Application)

Whereas I/we the undersigned inventor(s) have invented certain new and useful improvements as set forth in the patent application entitled:

NAVIGATING NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN NATURAL LANGUAGE INPUT WITH MULTIMODAL ERROR FEEDBACK

for which I/we have executed an application for a United States Letters Patent which was filed in the U.S. Patent and Trademark Office on March 13, 2000, and which bears the Application No. 09/524,095.

For good and valuable consideration, the receipt and sufficiency of which is hereby acknowledged, I/we the undersigned inventor(s) hereby:

1) Sell(s), assign(s) and transfer(s) to **SRI International,** a California non-profit corporation having a place of business at 333 Ravenswood Avenue, Menlo Park, California 94025, (hereinafter referred to as "ASSIGNEE"), the entire right title and interest in any and all improvements and inventions disclosed in, application(s) based upon, and Patent(s) (including foreign patents) granted upon the information which is disclosed in the above referenced application.

2) Authorize and request the Commissioner of Patents to issue any and all Letters Patents resulting from said application or any division(s), continuation(s), substitutes(s) or reissue(s) thereof to the ASSIGNEE.

3) Agree to execute all papers and documents and, entirely at the ASSIGNEE's expense, perform any acts which are reasonably necessary in connection with the prosecution of said application, as well as any derivative and applications thereof, foreign applications based thereon, and/or the enforcement of patents resulting from such applications.

4) Agree that the terms, covenants and conditions of this assignment shall inure to the benefit of the Assignee, its successors, assigns and other legal representative, and shall be binding upon the inventor(s), as well as the inventor's heirs, legal representatives and assigns.

5) Warrant and represent that I/we have not entered, and will not enter into any assignment, contract, or understanding that conflicts with this assignment.

Signed on the date(s) indicated beside my (our) signature(s).

1) Signature: Christine Halverson Date: 6-16-00.
Typed Name: Christine Halverson

2) Signature: ____________ Date: ________
Typed Name: Luc Julia

3) Signature: ____________ Date: 6/16/00
Typed Name: Dimitris Voutsas

4) Signature: ____________ Date: ________
Typed Name: Adam Cheyer

2155

PTO/SB/21 (08-00)

Please type a plus sign (+) inside this box ⟶ [+]

Approved for use through 10/31/2002. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	09/607,672
Filing Date	June 30, 2000
First Named Inventor	HALVERSON
Group Art Unit	2155
Examiner Name	T. Lee
Attorney Docket Number	SRI 1P037C
Total Number of Pages in This Submission	6

ENCLOSURES *(check all that apply)*

- □ Fee Transmittal Form
  - □ Fee Attached
- □ Amendment / Response
  - □ After Final
  - □ Affidavits/declaration(s)
- □ Extension of Time Request
- □ Express Abandonment Request
- □ Information Disclosure Statement
- □ Certified Copy of Priority Document(s)
- □ Response to Missing Parts/ Incomplete Application
  - □ Response to Missing Parts under 37 CFR 1.52 or 1.53

- □ Assignment Papers *(for an Application)*
- □ Drawing(s)
- □ Licensing-related Papers
- □ Petition
- □ Petition to Convert to a Provisional Application
- ☒ Power of Attorney, Revocation Change of Correspondence Address
- □ Terminal Disclaimer
- □ Request for Refund
- □ CD, Number of CD(s)

- □ After Allowance Communication to Group
- □ Appeal Communication to Board of Appeals and Interferences
- □ Appeal Communication to Group *(Appeal Notice, Brief, Reply Brief)*
- □ Proprietary Information
- □ Status Letter
- □ Other Enclosure(s) *(please identify below)*:
  POSTCARD

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm *or* Individual name	KIN-WAH TONG, Reg. No. 39,400
Signature	
Date	October 3, 2001

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231 on this date: October 3, 2001

Typed or printed name	Linda DeNardi		
Signature	Linda DeNardi	Date	October 3, 2001

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be send to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

#10

UNITED STATES PATENT AND TRADEMARK OFFICE

COMMISSIONER FOR PATENTS
UNITED STATES PATENT AND TRADEMARK OFFICE
WASHINGTON, D.C. 20231
www.uspto.gov

APPLICATION NUMBER	FILING DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
09/607,672	06/30/2000	Christine Halversen	SRI1P037C

CONFIRMATION NO. 1291

OC000000006901543

C. DOUGLAS McDONALD, ESQ.
CARLTON FIELDS, et al.
P.O. BOX 3239
TAMPA,, FL 33601-3239

Date Mailed: 10/15/2001

NOTICE REGARDING POWER OF ATTORNEY

This is in response to the Power of Attorney filed 10/09/2001.

- The Power of Attorney to you in this application has been revoked by the assignee who has intervened as provided by 37 CFR 3.71. Future correspondence will be mailed to the new address of record(37 CFR 1.33).

Lavinia Johnson

LAVINIA D JOHNSON
2100 7033085229

OFFICE COPY

11

UNITED STATES PATENT AND TRADEMARK OFFICE

COMMISSIONER FOR PATENTS
UNITED STATES PATENT AND TRADEMARK OFFICE
WASHINGTON, D.C. 20231
www.uspto.gov

APPLICATION NUMBER	FILING DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
09/607,672	06/30/2000	Christine Halversen	SRI1P037C

CONFIRMATION NO. 1291

OC000000006901587

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY, NJ 07702

Date Mailed: 10/15/2001

NOTICE REGARDING POWER OF ATTORNEY

This is in response to the Power of Attorney filed 10/09/2001.

The Power of Attorney in this application is accepted. Correspondence in this application will be mailed to the above address as provided by 37 CFR 1.33.

Lavinia Johnson
LAVINIA D JOHNSON
2100 7033085229

OFFICE COPY

#12/Reconsideration
T. McBeth-Brown
11/29/01

09/607,672

IN THE UNITED STATES
PATENT AND TRADEMARK OFFICE

PATENT APPLICATION

Applicant: **Halversen et al.**

Case: **SRI1P037C**

Serial No.: **09/607,672** Filed: **June 30, 2000**

Group Art Unit: **2155**

Examiner: **Tammy Lee**

Title: **System, Method, And Article Of Manufacture For Agent-Based Navigation In A Speech-Based Data Navigation System**

ASSISTANT COMMISSIONER FOR PATENTS
Box Non-Fee Amendment
Washington, D. C. 20231

S I R:

RESPONSE UNDER 37 C.F.R. § 1.111

This response addresses the Office Action dated August 27, 2001 (Paper No. 9).

REMARKS

In view of the following discussion, the Applicants submit that none of the claims now pending in the application are made obvious under the provisions of 35 U.S.C. § 103. Thus, the Applicants believe that all of these claims are now in allowable form.

I. REJECTION OF CLAIMS 56-76 UNDER 35 U.S.C. § 103

The Examiner has rejected claims 56-76 in Paragraphs 4-25 of the Office Action as being unpatentable over Sugikawa, (US Patent 5,949,772 issued September 7, 1999) in view of Gibson (US Patent 6,052,716, issued April 18, 2000). The rejection is respectfully traversed.

Sugikawa teaches a network of communication devices where services provided by the communication devices are available without prior registration of device information. Namely, it discloses "a communication device such that the service to be provided by any device on a network need not be registered in the respective devices or a service providing device...". (See Sugikawa, Column 5, lines 19-48) Thus, Sugikawa simply discloses a particular type of communication protocol between various hardware devices. However, the reference is completely devoid of any disclosure pertaining to the use of speech recognition in conjunction with the generation of navigation query to provide data to a user.

Gibson teaches a browser method that allows a user to rapidly return to an index page. Specifically, a Web based search engine typically provides an index page after a search request. As the user peruses through a link on the index page, the user may desire to return to the original index page to explore another link. Typically, a user will need to back up in a hierarchical manner to reach the original index page. Gibson provides a method to quickly return to the index page without having to traverse back up the entire hierarchical tree of links. (See Gibson, Column 3, lines 22-51)

In contrast, Sugikawa and Gibson (either singly or in any permissible combination) fail to teach or suggest the novel concept of speech-based navigation where a navigation query is constructed based upon the interpretation of a spoken request from a user. Specifically, Applicants' independent claims 56, 63 and 70 positively recite:

> 56. A method for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:
> (a) receiving a spoken request for desired information from a user;
> (b) rendering an interpretation of the spoken request;
> (c) constructing a navigation query based upon the interpretation;
> (d) routing the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source; and
> (e) invoking a user interface agent for outputting the selected portion of the electronic data source to the user.
>
> (emphasis added)

63. A computer program embodied on a computer readable medium for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:

(a) a code segment that receives a spoken request for desired information from a user;

(b) a code segment that renders an interpretation of the spoken request;

(c) a code segment that constructs a navigation query based upon the interpretation;

(d) a code segment that routes the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source; and

(e) a code segment that invokes a user interface agent for outputting the selected portion of the electronic data source to the user. (emphasis added)

70. A system for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:

(a) a client device, operable to receive a spoken request for desired information from a user;

(b) spoken language processing logic, operable to render an interpretation of the spoken request;

(c) query construction logic, operable to construct a navigation query based upon the interpretation;

(d) routing logic, operable to route the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source; and

(e) invoking logic, operable to invoke a user interface agent for outputting the selected portion of the electronic data source to the user. (emphasis added)

Applicants' invention teaches a novel method and apparatus for speech-based navigation where the method interprets the spoken request and then constructs a navigation query based upon the interpretation. Specifically, Applicants address the criticality of using spoken language to access a data source that may require a structured query in order to allow access to the desired information. For example, some databases may require the user to provide certain information in a particular structured format before access is allowed. To avoid this laborious and unwieldy requirement, Applicants' invention interprets the spoken request and automatically generates the necessary navigation query to access the desired information. This novel approach significantly minimizes the amount of manual navigation and data entry in accessing the desired information. (See Applicants' specification, page 14, line 5 to page 16, line 27)

In contrast, the alleged combination of Sugikawa and Gibson simply would not make Applicants' invention obvious. First, the Examiner alleged that Sugikawa teaches "(b) rendering an interpretation of the spoken request (a judging means and human voices, col.6, line 65 and col. 32, line 43)". Applicants respectfully disagree.

The section cited by the Examiner simply states that "a judging means which does, upon detection of said first service request data by said service request data detection means, judge whether that the conditions are met for providing the first service represented by said first service request data". This statement by Sugikawa cannot be interpreted to mean "rendering an interpretation of the spoken request" as recited by the Applicants. Specifically, Sugikawa states that:

> "The communication control unit 410 compares the data in the data section of the packet with said predetermined service request data.
>
> Where an agreement is found, the service request is notified to the controller unit 402 (cf. step 32 w2 in FIG. 2). The controller unit 402 examines the address.
>
> When all the devices are destinations, the controller unit 402 judges whether the device of its own can provide the service (cf. Step w3 in FIG. 2)". (See Sugikawa, Column 12, lines 21-29)

Thus, the section in Sugikawa cited by the Examiner is completely devoid of any teaching or suggestion of "rendering an interpretation of the spoken request" as claimed by the Applicants. Sugikawa is simply performing data comparison in the data section of a data packet.

Additionally, in rejecting Applicants' dependent claim 59 which recites the limitation of "wherein the step of rendering an interpretation of the spoken request is performed by a speech recognition agent and a parsing agent", the Examiner cited "(Column 27, lines 4-53 and col. 32 and lines 40-55)". However, Sugikawa states:

> "The comparator 16 compares the input signal from the communicable terminal identifier 14 with the communication terminal identification data stored in the communicable terminal memory 15 and, according to the result of comparison, outputs a control signal to the communicable terminal memory 15. When an agreement is found by the comparison, it is not necessary to rewrite the contents of the communicable terminal memory 15, so that there is no control signal

> output from the comparator 16 to the communicable terminal memory 15. On the other hand, when the comparison shows a disagreement, the comparator 16 outputs a control signal to update the contents of the communicable terminal memory 15." (See Sugikawa, Column 27, lines 30-43)

Once again, data is being compared and there is no disclosure of rendering an interpretation of the spoken request. In fact, speech data as disclosed in Sugikawa is simply the actual data. In other words, since Sugikawa discloses a communication network, some of the transported data is actual speech data, but there is absolutely no disclosure within Sugikawa as to the interpretation of the speech data in the context of speech recognition.

Second, the Examiner conceded in paragraph 5 of the Office Action that Sugikawa fails to disclose the novel concept of generating a "navigation query". However, the Examiner alleged that this deficiency in Sugikawa is bridged by the teaching of Gibson. Applicants respectfully disagree.

As noted above, Gibson only discloses a browser method that allows a user to return quickly back to an index page. Gibson achieves this function by compiling a list of network addresses accessed by the network navigator during a session. Finally, the network navigator jumps to the network address corresponding to the search engine network address. (See Gibson, Column 3, lines 28-51). Recording network addresses is clearly not equivalent to the construction of a navigation query. In other words, recording the Web addresses traversed by use of manual navigation is not equivalent to the construction of a navigation query from an interpretation of a spoken request that minimizes the need to perform manual navigation.

Therefore, the Applicants respectfully submit that independent claims 56, 63 and 70 are not made obvious by the alleged combination of Sugikawa and Gibson. As such, claims 56, 63 and 70 fully satisfy the requirements of 35 U.S.C. §103 and are patentable thereunder.

Claims 57-62, 64-69 and 71-76 depend, either directly or indirectly, from claims 56, 63 and 70 and recite additional features therefor. Since Sugikawa and Gibson fail to make obvious Applicants' invention as recited in Applicants' independent claims 56, 63 and 70, dependent claims 57-62, 64-69 and 71-76 are also not made obvious under 35 U.S.C. § 103 and are allowable for the same reason noted above.

Conclusion

Thus, the Applicants submit that all of these claims now fully satisfy the requirements of 35 U.S.C. §103. Consequently, the Applicants believe that all these claims are presently in condition for allowance. Accordingly, both reconsideration of this application and its swift passage to issue are earnestly solicited.

If, however, the Examiner believes that there are any unresolved issues requiring the issuance of a final action in any of the claims now pending in the application, it is requested that the Examiner telephone Mr. Kin-Wah Tong, Esq. at (732) 530-9404 so that appropriate arrangements can be made for resolving such issues as expeditiously as possible.

Respectfully submitted,

11/27/01

Kin-Wah Tong, Attorney
Reg. No. 39,400
(732) 530-9404

Moser, Patterson & Sheridan, LLP
595 Shrewsbury Avenue
First Floor,
Shrewsbury, New Jersey 07702

TELEFAX COVER SHEET

MOSER, PATTERSON & SHERIDAN, LLP
ATTORNEYS AT LAW
595 SHREWSBURY AVENUE
FIRST FLOOR
SHREWSBURY, NJ 07702
TELEPHONE (732) 530-9404
TELEFAX (732) 530-9808

****

THIS TELEFAX MESSAGE IS ADDRESSED TO THE PERSON OR COMPANY LISTED BELOW. IF IT WAS SENT OR RECEIVED INCORRECTLY, OR YOU ARE NOT THE INTENDED RECIPIENT, PLEASE TAKE NOTICE THAT THIS MESSAGE MAY CONTAIN PRIVILEGED OR CONFIDENTIAL MATERIAL, AND YOUR DUE REGARD FOR THIS INFORMATION IS NECESSARY. YOU MAY ARRANGE TO RETURN THIS MATERIAL BY CALLING THE FIRM LISTED ABOVE AT (732) 530-9404

****

THIS MESSAGE HAS ____ PAGES INCLUDING THIS SHEET

TO: Assistant Commissioner for Patents
FAX NO.: 703-746-7239
FROM: Kin-Wah Tong, Reg. #39,400
DATE: 11/27/01
MATTER: Serial No. 09/607,672 Filed: 6/30/00
DOCKET NO.: SRI1P037C
APPLICANT: HALVERSEN ET AL.

The following has been received in the U.S. Patent and Trademark Office on the date of this facsimile:

___ Specification (___ pages)
___ ___ Claims (___ pages)
___ Abstract (1 page)
___ Oath or Declaration
___ Petition
___ Power of Attorney
___ Claim of Priority
___ Disclosure Statement & PTO-1449
X Facsimile Transmission Certificate, dated November 27, 2001

___ Priority Document
___ Transmittal Letter (2 copies)
___ Deposit Acct. Transaction
___ Assignment & Cover Sheet
___ Drawings (___ sheets) FORMAL
___ Drawings (___ sheets) informal
___ Check No. ________ for $________
X RESPONSE UNDER 37 CFR 1.111
X FEE TRANSMITTAL
X TRANSMITTAL
___ ________________

I certify that is document is being transmitted by facsimile under 37 C.F.R. 1.6 on 11/27/01 and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231

Laura E Crater
Signature
LAURA E CRATER
Name

PTO/SB/21 (08-00)

Please type a plus sign (+) inside this box → [+]

Approved for use through 10/31/2002. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	09/607,872
Filing Date	6/30/00
First Named Inventor	HALVERSEN ET AL..
Group Art Unit	2155
Examiner Name	TAMMY LEE
Attorney Docket Number	SRI1P037C
Total Number of Pages in This Submission	

ENCLOSURES *(check all that apply)*

☒ Fee Transmittal Form

□ Fee Attached

☒ Amendment / Response

□ After Final

□ Affidavits/declaration(s)

□ Extension of Time Request

□ Express Abandonment Request

□ Information Disclosure Statement

□ Certified Copy of Priority Document(s)

□ Response to Missing Parts/ Incomplete Application

□ Response to Missing Parts under 37 CFR 1.52 or 1.53

□ Assignment Papers *(for an Application)*

□ Drawing(s)

□ Licensing-related Papers

□ Petition Routing Slip (PTO/SB/69) and Accompanying Petition

□ Petition to Convert to a Provisional Application

□ Power of Attorney, Revocation Change of Correspondence Address

□ Terminal Disclaimer

□ Request for Refund

□ CD, Number of CD(s)

□ After Allowance Communication to Group

□ Appeal Communication to Board of Appeals and Interferences

□ Appeal Communication to Group *(Appeal Notice, Brief, Reply Brief)*

□ Proprietary Information

□ Status Letter

☒ Other Enclosure(s) *(please identify below)*:

Remarks	**The Commissioner is authorized to charge any underpayment or credit any overpayment of fees (including but not limited to any extension fees pursuant to 1.136(a)), to Deposit Account 20-0782. A duplicate copy of this transmittal is attached.**

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm or Individual name	KIN-WAH TONG, Reg. No. 39,400
Signature	
Date	11/27/01

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be send to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

FEE TRANSMITTAL for FY 2002

Patent fees are subject to annual revision.

TOTAL AMOUNT OF PAYMENT	($) 0

Complete if Known

Application Number	09/607,672
Filing Date	6/30/00
First Named Inventor	HALVERSEN ET AL.
Examiner Name	TAMMY LEE
Group / Art Unit	2155
Attorney Docket No.	SRI1P037C

METHOD OF PAYMENT (check one)

1. ☒ The Commissioner is hereby authorized to charge indicated fees and credit any over payments to:

Deposit Account Number: 20-0782

Deposit Account Name: MOSER, PATTERSON & SHERIDAN, LLP

☒ Charge Any Additional Fee Required Under 37 CFR 1.16 and 1.17

☐ Applicant claims small entity status. See 37 CFR 1.27

2. ☐ Payment Enclosed:

☐ Check ☐ Credit card ☐ Money Order ☐ Other

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
101	740	201	370	Utility filing fee	
106	330	206	165	Design filing fee	
107	510	207	255	Plant filing fee	
108	740	208	370	Reissue filing fee	
114	160	214	80	Provisional filing fee	

SUBTOTAL (1) ($) 0

2. EXTRA CLAIM FEES

				Extra Claims		Fee from below		Fee Paid
Total Claims	21	-21 **	=	0	X		=	0
Independent Claims	3	-3 **	=	0	X		=	0
Multiple Dependent					X		=	0

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
103	18	203	9	Claims in excess of 20
102	84	202	42	Independent claims in excess of 3
104	280	204	140	Multiple dependent claim, if not paid
109	84	209	42	** Reissue independent claims over original patent
110	18	210	9	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) ($) 0

**or number previously paid, if greater; For Reissues, see above

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
105	130	205	65	Surcharge - late filing fee or oath	
127	50	227	25	Surcharge - late provisional filing fee or cover sheet.	
139	130	139	130	Non-English specification	
147	2,520	147	2,520	For filing a request for reexamination	
112	920*	112	920*	Requesting publication of SIR prior to Examiner action	
113	1,840*	113	1,840*	Requesting publication of SIR after Examiner action	
115	110	215	55	Extension for reply within first month	
116	400	216	200	Extension for reply within second month	
117	920	217	460	Extension for reply within third month	
118	1,440	218	720	Extension for reply within fourth month	
128	1.960	228	980	Extension for reply within fifth month	
119	320	219	160	Notice of Appeal	
120	320	220	160	Filing a brief in support of an appeal	
121	280	221	140	Request for oral hearing	
138	1,510	138	1,510	Petition to institute a public use proceeding	
140	110	240	55	Petition to revive – unavoidable	
141	1,280	241	640	Petition to revive – unintentional	
142	1.280	242	640	Utility issue fee (or reissue)	
143	460	243	230	Design issue fee	
144	620	244	310	Plant issue fee	
122	130	122	130	Petitions to the Commissioner	
123	50	123	50	Processing fee under 37 CFR 1.17 (q)	
126	180	126	180	Submission of Information Disclosure Stmt	
581	40	581	40	Recording each patent assignment per property (times number of properties)	
146	740	246	370	Filing a submission after final rejection (37 CFR § 1.129(a))	
149	740	249	370	For each additional invention to be examined (37 CFR § 1.129(b))	
179	740	279	370	Request for Continued Examination (RCE)	
169	900	169	900	Request for expedited examination of a design application	

Other fee (specify)

*Reduced by Basic Filing Fee Paid SUBTOTAL (3) ($) 0

SUBMITTED BY Complete *(if applicable)*

Name (Print/Type)	KIN-WAH TONG	*Registration No. Attorney/Agent)*	39,400	*Telephone*	732-530-9404
Signature				*Date*	11/27/01

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

SR

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/607,672	06/30/2000	Christine Halversen	SRI1P037C	1291

7590 02/13/2002

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY, NJ 07702

EXAMINER	
NGUYEN, THU HA T	
ART UNIT	**PAPER NUMBER**
2155	

DATE MAILED: 02/13/2002

#13

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 07-01)

HG

A G

Office Action Summary	**Application No.**	**Applicant(s)**
	09/607,672	HALVERSEN ET AL.
	Examiner	**Art Unit**
	Thu Ha T. Nguyen	2155

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). *In no event, however, may a reply be timely filed* after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 30 June 2000 .

2a)□ This action is **FINAL.** 2b)☒ This action is non-final.

3)□ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 56-76 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)□ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 56-76 is/are rejected.

7)□ Claim(s) _____ is/are objected to.

8)□ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)□ The specification is objected to by the Examiner.

10)□ The drawing(s) filed on _____ is/are: a)□ accepted or b)□ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)□ The proposed drawing correction filed on _____ is: a)□ approved b)□ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)□ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)□ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)□ All b)□ Some * c)□ None of:

1.□ Certified copies of the priority documents have been received.

2.□ Certified copies of the priority documents have been received in Application No. _____ .

3.□ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)□ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) □ The translation of the foreign language provisional application has been received.

15)□ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) □ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) □ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____ .

4) □ Interview Summary (PTO-413) Paper No(s). _____ .

5) □ Notice of Informal Patent Application (PTO-152)

6) □ Other: .

DETAILED ACTION

1. Claims **56-76** are presented for examination.

Claim Rejections - 35 USC § 102

2. The following is a quotation of the appropriate paragraphs of 35 U.S.C. § 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless --
> (e) the invention was described in a patent granted on an application for patent by another filed in the United States before the invention thereof by the applicant for patent, or on an international application by another who has fulfilled the requirements of paragraphs (1), (2), and (4) of section 37 1(c) of this title before the invention thereof by the applicant for patent.

3. Claims 56-76 are rejected under 35 U.S.C. § 102(e) as being anticipated by **Perrone** U.S. Patent No. **6,157,705**.

4. As to claim 56, **Perrone** teaches the invention as claimed, including a method for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:

receiving a spoken request for desired information from a user (abstract, figures 1, 2, 4, col. 8 lines 4-20),

rendering an interpretation of the spoken request (abstract, figures 1, 2, 4, col. 5 lines 16-col. 6 lines 33, col. 8 lines 21-55, col. 18 lines 29-63), constructing a navigation query based upon the interpretation (abstract, figure 4, col. 11 lines 19-45, col. 18 lines 29-63),

routing the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source (abstract, figure 4, col. 3 lines 41-col. 4 lines 35, col. 18 lines 29-col. 19 lines 12), and

invoking a user interface agent for outputting the selected portion of the electronic data source to the user (abstract, figures 3, 5, col. 5 lines 17-col. 6 lines 33, col. 8 lines 4-55, col. 18 lines 29-col. 19 lines 52).

5. As to claim 57, **Perrone** teaches the invention as claimed, wherein an agent renders the interpretation of the spoken request (abstract, figures 1, 2, 4, col. 5 lines 16-col. 6 lines 33, col. 8 lines 21-55, col. 18 lines 29-63).

6. As to claim 58, **Perrone** teaches the invention as claimed, wherein a facilitator manages data flow among multiple agents (figure 5, col. 1 lines 9-col. 2 lines 6, col. 17 lines 10-col. 19 lines 12).

7. As to claim 59, **Perrone** teaches the invention as claimed, wherein the step of rendering the interpretation of the spoken request is performed by a speech recognition agent and a parsing agent (abstract, figures 1, 2, 4, col. 5 lines 16-col. 6 lines 33, col. 8 lines 21-55, col. 18 lines 29-63).

8. As to claim 60, **Perrone** teaches the invention as claimed, further comprising the steps of soliciting additional input from the user, including user interaction in a modality different than the original request; and refining the navigation query, based upon the additional input; wherein the at least one agent uses the refined navigation query to select a portion of the electronic data source (figure 5, col. 1 lines 9-col. 2 lines 6, col. 17 lines 10-col. 19 lines 12).

9. As to claim 61, **Perrone** teaches the invention as claimed, wherein agents are utilized for performing the steps of soliciting additional input from the user and refining the navigation query (abstract, figures 4, 5, col. 1 lines 9-col. 2 lines 6, col. 3 lines 41-col. 4 lines 35, col. 17 lines 10-col. 19 lines 12).

10. As to claim 62, **Perrone** teaches the invention as claimed, wherein the electronic data source is a web page, wherein the at least one agent scrapes the web page for selecting a portion of the web page (col. 9 lines 9-col. 11 lines 10, col. 12 lines 36-col. 13 lines 35).

11. As to claim 63, **Perrone** teaches the invention as claimed, including a computer program embodied on a computer readable medium for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:

a code segment that receives a spoken request for desired information from a user (abstract, figures 1, 2, 4, col. 8 lines 4-20), a code segment that renders an interpretation of the spoken request (abstract, figures 1, 2, 4, col. 5 lines 16-col. 6 lines 33, col. 8 lines 21-55, col. 18 lines 29-63),

a code segment that constructs a navigation query based upon the interpretation (abstract, figure 4, col. 11 lines 19-45, col. 18 lines 29-63),

a code segment that routes the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source (abstract, figure 4, col. 3 lines 41-col. 4 lines 35, col. 18 lines 29-col. 19 lines 12), and

a code segment that invokes a user interface agent for outputting the selected portion of the electronic data source to the user (abstract, figures 3, 5, col. 5 lines 17-col. 6 lines 33, col. 8 lines 4-55, col. 18 lines 29-col. 19 lines 52).

12. As to claim 64, **Perrone** teaches the invention as claimed, wherein the code segment that renders the interpretation of the spoken request is executed by an agent (abstract, figures 1, 2, 4, col. 5 lines 16-col. 6 lines 33, col. 8 lines 21-55, col. 18 lines 29-63).

13. As to claim 65, **Perrone** teaches the invention as claimed, wherein a facilitator manages data flow among multiple agents (figure 5, col. 1 lines 9-col. 2 lines 6, col. 17 lines 10-col. 19 lines 12).

14. As to claim 66, **Perrone** teaches the invention as claimed, wherein a speech recognition agent and a parsing agent execute the code segment that renders the interpretation of the spoken request (abstract, figures 1, 2, 4, col. 5 lines 16-col. 6 lines 33, col. 8 lines 21-55, col. 18 lines 29-63).

15. As to claim 67, **Perrone** teaches the invention as claimed, further comprising a code segment that solicits additional input from the user, including user interaction in a modality different than the original request; and a code segment that refines the navigation query, based upon the additional input; wherein the at least one agent uses the refined navigation query to select a portion of the electronic data source (figure 5, col. 1 lines 9-col. 2 lines 6, col. 17 lines 10-col. 19 lines 12).

16. As to claim 68, **Perrone** teaches the invention as claimed, wherein a solicitor agent executes the code segment that solicit the additional input from the user and a refining agent executes the code segment that refines the navigation query (abstract, figures 4, 5, col. 1 lines 9-col. 2 lines 6, col. 3 lines 41-col. 4 lines 35, col. 17 lines 10-col. 19 lines 12).

17. As to claim 69, **Perrone** teaches the invention as claimed, wherein the electronic data source is a web page, wherein the at least one agent scrapes the web page for selecting a portion of the web page (col. 9 lines 9-col. 11 lines 10, col. 12 lines 36-col. 13 lines 35).

18. As to claim 70, **Perrone** teaches the invention as claimed, including a system for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:

a client device, operable to receive a spoken request for desired information from a user (abstract, figures 1, 2, 4, col. 8 lines 4-20),

spoken language processing logic, operable to render an interpretation of the spoken request (abstract, figures 1, 2, 4, col. 5 lines 16-col. 6 lines 33, col. 8 lines 21-55, col. 18 lines 29-63),

query construction logic, operable to construct a navigation query based upon the interpretation (abstract, figure 4, col. 11 lines 19-45, col. 18 lines 29-63),

routing logic, operable to route the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source (abstract, figure 4, col. 3 lines 41-col. 4 lines 35, col. 18 lines 29-col. 19 lines 12), and

invoking logic, operable to invoke a user interface agent for outputting the selected portion of the electronic data source to the user (abstract, figures 3, 5, col. 5 lines 17-col. 6 lines 33, col. 8 lines 4-55, col. 18 lines 29-col. 19 lines 52).

19. As to claim 71, **Perrone** teaches the invention as claimed, wherein the query construction logic that renders the interpretation of the spoken request is

executed by an agent (abstract, figures 1, 2, 4, col. 5 lines 16-col. 6 lines 33, col. 8 lines 21-55, col. 18 lines 29-63).

20. As to claim 72, **Perrone** teaches the invention as claimed, wherein a facilitator manages data flow among multiple agents (figure 5, col. 1 lines 9-col. 2 lines 6, col. 17 lines 10-col. 19 lines 12).

21. As to claim 73, **Perrone** teaches the invention as claimed, wherein a speech recognition agent and a parsing agent execute the spoken language processing logic that renders the interpretation of the spoken request (abstract, figures 1, 2, 4, col. 5 lines 16-col. 6 lines 33, col. 8 lines 21-55, col. 18 lines 29-63).

22. As to claim 74, **Perrone** teaches the invention as claimed, further comprising user interaction logic operable to solicit additional input from the user, including user interaction in a modality different than the original request; and query refining logic operable to refine the navigation query, based upon the additional input; wherein the at least one agent uses the refined navigation query to select a portion of the electronic data source (figure 5, col. 1 lines 9-col. 2 lines 6, col. 17 lines 10-col. 19 lines 12).

23. As to claim 75, **Perrone** teaches the invention as claimed, wherein a solicitor agent executes the user interaction logic and a refining agent executes the

query refinement logic (abstract, figures 4, 5, col. 1 lines 9-col. 2 lines 6, col. 3 lines 41-col. 4 lines 35, col. 17 lines 10-col. 19 lines 12).

24. As to claim 76, **Perrone** teaches the invention as claimed, wherein the electronic data source is a web page, wherein the at least one agent scrapes the web page for selecting a portion of the web page (col. 9 lines 9-col. 11 lines 10, col. 12 lines 36-col. 13 lines 35).

Response to Arguments

25. Applicant's arguments with respect to claims 56-76 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

26. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

27. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Thu Ha Nguyen, whose telephone number is (703) 305-7447. The examiner can normally be reached Monday through Friday from 7:30 AM to 4:30 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, SPE Ayaz R. Sheikh, can be reached at (703) 305-9648.

Any inquiry of a general nature of relating to the status of this application should be directed to the Group receptionist whose telephone number is (703) 305-9600.

The fax number for art unit 2155 is (703) 305-7201.

Thu Ha Nguyen

February 6, 2002

AYAZ SHEIKH
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

PAGE 1 OF 1

FORM PTO-892 U.S. DEPARTMENT OF COMMERCE PATENT AND TRADEMARK OFFICE	SERIAL NO. 09/607,672	GROUP ART UNIT 2155	ATTACHMENT TO PAPER NO.	13
NOTICE OF REFERENCES CITED	APPLICANT(S) HALVERSEN ET AL.			

U.S. PATENT DOCUMENTS

*		DOCUMENT NO.	DATE	NAME	CLASS	SUB-CLASS	FILING DATE
	A	6,157,705	12/2000	Perrone	379	88.01	12/1997
	B	6,101,473	8/2000	Scott et al.	704	275	08/1997
	C	6,289,140	9/2001	Oliver	382	313	12/1998
	D	6,282,270	8/2001	Porter	379	88.17	08/1995
	E	6,282,511	8/2001	Mayer	704	270	12/1996
	F	5,884,262	3/1999	Wise et al	704	270	03/1996
	G						
	H						
	I						
	J						
	K						

FOREIGN PATENT DOCUMENTS

*		DOCUMENT NO.	DATE	COUNTRY	NAME	CLASS	SUB-CLASS
	L						
	M						
	N						
	O						
	P						
	Q						

OTHER REFERENCES (Including Author, Title, Date, Pertinent Pages, Etc.)

	R	http://www-3.ibm.com/software/speech/desktop/w9-pro.html. IBM Via Voice for windows, Pro USB edition release 9 by IBM corp.
	S	
	T	
	U	

EXAMINER Thu Ha Nguyen	DATE February 6, 2002	Form892ccs2106b

* A copy of this reference is not being furnished with this office action.
(See Manual of Patent Examining Procedure, section 707.05(a).)

UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231

SERIAL NUMBER	FILING DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKETT NO.
09/607,672			

EXAMINER

ART UNIT	PAPER NUMBER
	14

DATE MAILED:

EXAMINER INTERVIEW SUMMARY RECORD

All participants (applicant, applicant's representative, PTO personnel):

(1) David Wiley (3) Kinwah Tong 39,400

(2) Thu- Nguyen (4)

Date of interview 5/23/2002

Type: □ Telephonic ☒ Personal (copy is given to □ applicant □ applicant's representative).

Exhibit shown or demonstration conducted: □ Yes ☒ No. If yes, brief description:

Agreement ☒ was reached with respect to some or all of the claims in question. □ was not reached.

Claims discussed: 56 - 76

Identification of prior art discussed: Perrone

Description of the general nature of what was agreed to if an agreement was reached, or any other comments: The applicant agreed to expand the aspect of agent & agent registration to overcome the rejection with Perrone.

(A fuller description, if necessary, and a copy of the amendments, if available, which the examiner agreed would render the claims allowable must be attached. Also, where no copy of the amendments which would render the claims allowable is available, a summary thereof must be attached.)

☒ 1. It is not necessary for applicant to provide a separate record of the substance of the interview.

Unless the paragraph below has been checked to indicate to the contrary, A FORMAL WRITTEN RESPONSE TO THE LAST OFFICE ACTION IS NOT WAIVED AND MUST INCLUDE THE SUBSTANCE OF THE INTERVIEW (e.g., items 1-7 on the reverse side of this form). If a response to the last Office action has already been filed, then applicant is given one month from this interview date to provide a statement of the substance of the interview.

□ 2. Since the examiner's interview summary above (including any attachments) reflects a complete response to each of the objections, rejections and requirements that may be present in the last Office action, and since the claims are now allowable, this completed form is considered to fulfill the response requirements of the last Office action. Applicant is not relieved from providing a separate record of the substance of the interview unless box 1 above is also checked.

Examiner's Signature

ORIGINAL FOR INSERTION IN RIGHT HAND FLAP OF FILE WRAPPER

U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a)	Docket Number (Optional) SRI 1P037C
In re Application of Halversen, et al	
Application Number 09/607,672	Filed June 30, 2000
For System, Method, and Article of Manufacture for Agent-Based Navigation in a Speech-Based Data Navigation System	
Group Art Unit 2155	Examiner T. Nguyen

#15
S. Carter
7-17-02

This is a request under the provisions of 37 CFR 1.136(a) to extend the period for filing a response in the above identified application.

The requested extension and appropriate non-small-entity fee are as follows (check time period desired):

☐ One month (37 CFR 1.17(a)(1))	$
☒ Two months (37 CFR 1.17(a)(2))	$400.00
☐ Three months (37 CFR 1.17(a)(3))	$
☐ Four months (37 CFR 1.17(a)(4))	$
☐ Five months (37 CFR 1.17(a)(5))	$

☒ Applicant claims small entity status. See 37 CFR 1.27. Therefore, the fee amount shown above is reduced by one-half, and the resulting fee is: $ 200.00 .

☐ A check in the amount of the fee is enclosed.

☐ Payment by credit card. Form PTO-2038 is attached.

☐ The Commissioner has already been authorized to charge fees in this application to a Deposit Account.

☒ The Commissioner is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account Number 20-0782 .

I have enclosed a duplicate copy of this sheet.

I am the ☐ applicant/inventor.

☐ assignee of record of the entire interest. See 37 CFR 3.71

Statement under 37 CFR 3.73(b) is enclosed. (Form PTO/SB/96).

☒ attorney or agent of record.

☐ attorney or agent under 37 CFR 1.34(a).

Registration number if acting under 37 CFR 1.34(a). ______ .

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

July 15, 2002
Date

Signature
Kin-Wah Tong
Typed or printed name

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below*.

☐ *Total of _____ forms are submitted.

Burden Hour Statement: This form is estimated to take 0.1 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

MISSING PAGE(S) FROM THE U.S. PATENT OFFICE OFFICIAL FILE WRAPPER

Facsimile Page 4

(Note: This page is not a part of the official USPTO record.)

09/607,672

#16/B
O. Cotton
7-17-02

IN THE UNITED STATES
PATENT AND TRADEMARK OFFICE

PATENT APPLICATION

Applicant: **Halversen et al.**

Case: **SRI1P037C**

Serial No.: **09/607,672** Filed: **June 30, 2000**

Group Art Unit: **2155**

Examiner: **Thu Ha Nguyen**

Title: **System, Method, And Article Of Manufacture For Agent-Based Navigation In A Speech-Based Data Navigation System**

ASSISTANT COMMISSIONER FOR PATENTS
Box Fee Amendment
Washington, D. C. 20231

S I R:

AMENDMENT AND RESPONSE UNDER 37 C.F.R. § 1.111

This response addresses the Office Action dated February 13, 2002 (Paper No. 13).

IN THE CLAIMS

Please cancel claims 58, 65, and 72 without prejudice.

Please amend claims 56, 63 and 70 as shown below. The claims are "clean version" of the amended claims, i.e., with changes incorporated into the claims, whereas the Appendix to this Amendment illustrates the amended claims using underlines and brackets to indicate addition and deletion, respectively.

31 1 ~~56~~. (Amended) A method for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:

33

B

(a) receiving a spoken request for desired information from a user;

(b) rendering an interpretation of the spoken request;

(c) constructing a navigation query based upon the interpretation;

B1

(d) routing the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source; and

(e) invoking a user interface agent for outputting the selected portion of the electronic data source to the user, wherein a facilitator manages data flow among multiple agents and maintains a registration of each of said agents' capabilities.

1 ~~63~~. (Amended) A computer program embodied on a computer readable medium for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:

(a) a code segment that receives a spoken request for desired information from a user;

(b) a code segment that renders an interpretation of the spoken request;

B2

(c) a code segment that constructs a navigation query based upon the interpretation;

(d) a code segment that routes the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source; and

(e) a code segment that invokes a user interface agent for outputting the selected portion of the electronic data source to the user, wherein a facilitator manages data flow among multiple agents and maintains a registration of each of said agents' capabilities.

13 ~~70~~. (Amended) A system for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:

B3

(a) a client device, operable to receive a spoken request for desired information from a user;

(b) spoken language processing logic, operable to render an interpretation of the

34

B

spoken request;

(c) query construction logic, operable to construct a navigation query based upon the interpretation;

(d) routing logic, operable to route the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source; and

B3

(e) invoking logic, operable to invoke a user interface agent for outputting the selected portion of the electronic data source to the user, wherein a facilitator manages data flow among multiple agents and maintains a registration of each of said agents' capabilities.

REMARKS

Applicants' representative would like to thank Examiner Nguyen and Primary Examiner David Wiley for kindly taking a substantial amount of time on May 23, 2002 to discuss the merits of the subject invention in a face-to-face Examiner Interview. Applicants' representative is aware of the time constraint that is placed on the Examiners and is appreciative of the Examiners' willingness to devote such large quantity of time to discuss the case on the merit.

In view of the following discussion, the Applicants submit that none of the claims now pending in the application are anticipated under the provisions of 35 U.S.C. § 102. Thus, the Applicants believe that all of these claims are now in allowable form.

I. REJECTION OF CLAIMS 56-76 UNDER 35 U.S.C. § 102

The Examiner has rejected claims 56-76 in Paragraphs 3-24 of the Office Action as being unpatentable over Perrone, (US Patent 6,157,705, issued December 5, 2000). The rejection is respectfully traversed.

Perrone teaches a method for controlling a server using voice. Specifically, Perrone discloses the establishment of a "voice communication channel" and a separate "data communication channel" between a local client and a remote server. In operation, voice command over the voice communication channel is received and

35

B

processed and then the desired data is returned on the "data communication channel" to the local client. (See Perrone, Column 3, lines 41-55) However, the Perrone reference is completely devoid of any disclosure pertaining to the use of agents for speech-based navigation.

Specifically, Applicants' amended independent claims 56, 63 and 70 positively recite:

> 56. A method for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:
> (a) receiving a spoken request for desired information from a user;
> (b) rendering an interpretation of the spoken request;
> (c) constructing a navigation query based upon the interpretation;
> (d) <u>routing the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source</u>; and
> (e) <u>invoking a user interface agent for outputting the selected portion of the electronic data source to the user, wherein a facilitator manages data flow among multiple agents and maintains a registration of each of said agents' capabilities</u>.
> (emphasis added)
>
> 63. A computer program embodied on a computer readable medium for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:
> (a) a code segment that receives a spoken request for desired information from a user;
> (b) a code segment that renders an interpretation of the spoken request;
> (c) a code segment that constructs a navigation query based upon the interpretation;
> (d) <u>a code segment that routes the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source</u>; and
> (e) <u>a code segment that invokes a user interface agent for outputting the selected portion of the electronic data source to the user, wherein a facilitator manages data flow among multiple agents and maintains a registration of each of said agents' capabilities</u>. (emphasis added)
>
> 70. A system for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:
> (a) a client device, operable to receive a spoken request for desired information from a user;
> (b) spoken language processing logic, operable to render an interpretation

B

of the spoken request;

(c) query construction logic, operable to construct a navigation query based upon the interpretation;

(d) routing logic, operable to route the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source; and

(e) invoking logic, operable to invoke a user interface agent for outputting the selected portion of the electronic data source to the user, wherein a facilitator manages data flow among multiple agents and maintains a registration of each of said agents' capabilities. (emphasis added)

Applicants' invention teaches a novel method and apparatus for speech-based navigation where the method interprets the spoken request and then constructs a navigation query based upon the interpretation. More specifically, Applicants' invention exploits the dynamic collaboration of a set of distributed agents in executing the above claimed speech-based navigation. In one embodiment, the functionality of each agent is made available to the agent community through registration of the agent's capabilities with a facilitator. In turn, the facilitator coordinates and integrates the results received from different agents on various sub-goals in order to satisfy the overall goal of the spoken request. (See Applicants' specification, page 21, lines 1-27) Applicants' distributed approach is a powerful and flexible platform that will allow the system to evolve and expand as the complexity of the spoken request increases.

As noted by Primary Examiner Wiley during the Examiner Interview, Perrone is completely devoid of this novel teaching pertaining to the use of agents. The Examiner indicated that if Applicants clarify the registration aspect of the agent in the independent claims, then the Examiner would withdraw the present rejection.

Therefore, the Applicants respectfully submit that independent claims 56, 63 and 70 are not anticipated by Perrone. As such, claims 56, 63 and 70 fully satisfy the requirements of 35 U.S.C. §102 and are patentable thereunder.

Claims 57, 59-62, 64, 66-69 and 71, 73-76 depend, either directly or indirectly, from claims 56, 63 and 70 and recite additional features therefor. Since Perrone fails to anticipate Applicants' invention as recited in Applicants' independent claims 56, 63 and 70, dependent claims 57, 59-62, 64, 66-69 and 71, 73-76 are also not anticipated under 35 U.S.C. § 102 and are allowable for the same reason noted above.

B

09/607,672

Conclusion

Thus, the Applicants submit that all of these claims now fully satisfy the requirements of 35 U.S.C. §102. Consequently, the Applicants believe that all these claims are presently in condition for allowance. Accordingly, both reconsideration of this application and its swift passage to issue are earnestly solicited.

If, however, the Examiner believes that there are any unresolved issues requiring the issuance of a final action in any of the claims now pending in the application, it is requested that the Examiner telephone Mr. Kin-Wah Tong, Esq. at (732) 530-9404 so that appropriate arrangements can be made for resolving such issues as expeditiously as possible.

Respectfully submitted,

7/15/02

Kin-Wah Tong, Attorney
Reg. No. 39,400
(732) 530-9404

Moser, Patterson & Sheridan, LLP
595 Shrewsbury Avenue
First Floor,
Shrewsbury, New Jersey 07702

09/607,672

APPENDIX

(Marked-up version of amended claims)

56. (Amended) A method for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:

(a) receiving a spoken request for desired information from a user;

(b) rendering an interpretation of the spoken request;

(c) constructing a navigation query based upon the interpretation;

(d) routing the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source; and

(e) invoking a user interface agent for outputting the selected portion of the electronic data source to the user, wherein a facilitator manages data flow among multiple agents and maintains a registration of each of said agents' capabilities.

63. (Amended) A computer program embodied on a computer readable medium for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:

(a) a code segment that receives a spoken request for desired information from a user;

(b) a code segment that renders an interpretation of the spoken request;

(c) a code segment that constructs a navigation query based upon the interpretation;

(d) a code segment that routes the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source; and

(e) a code segment that invokes a user interface agent for outputting the selected portion of the electronic data source to the user, wherein a facilitator manages data flow among multiple agents and maintains a registration of each of said agents' capabilities.

B

70. (Amended) A system for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:

(a) a client device, operable to receive a spoken request for desired information from a user;

(b) spoken language processing logic, operable to render an interpretation of the spoken request;

(c) query construction logic, operable to construct a navigation query based upon the interpretation;

(d) routing logic, operable to route the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source; and

(e) invoking logic, operable to invoke a user interface agent for outputting the selected portion of the electronic data source to the user, wherein a facilitator manages data flow among multiple agents and maintains a registration of each of said agents' capabilities.

B

TELEFAX COVER SHEET

MOSER, PATTERSON & SHERIDAN, LLP
ATTORNEYS AT LAW
595 SHREWSBURY AVENUE
FIRST FLOOR
SHREWSBURY, NJ 07702
TELEPHONE (732) 530-9404
TELEFAX (732) 530-9808

****

THIS TELEFAX MESSAGE IS ADDRESSED TO THE PERSON OR COMPANY LISTED BELOW. IF IT WAS SENT OR RECEIVED INCORRECTLY, OR YOU ARE NOT THE INTENDED RECIPIENT, PLEASE TAKE NOTICE THAT THIS MESSAGE MAY CONTAIN PRIVILEGED OR CONFIDENTIAL MATERIAL, AND YOUR DUE REGARD FOR THIS INFORMATION IS NECESSARY. YOU MAY ARRANGE TO RETURN THIS MATERIAL BY CALLING THE FIRM LISTED ABOVE AT (732) 530-9404

****

THIS MESSAGE HAS 12 PAGES INCLUDING THIS SHEET

TO: Assistant Commissioner of Patents

FAX NO.: 703-746-7239

FROM: Kin-Wah Tong

DATE: July 15, 2002

MATTER: Serial No. 09/607,672 Filed: June 30, 2000

DOCKET NO.: SRI 1P037C

APPLICANT: HALVERSON, et al

The following has been received in the U.S. Patent and Trademark Office on the date of this facsimile:

___ Petition
___ Disclosure Statement & PTO-1449
___ Priority Document
___ Drawings (____ sheets) informal
X Petition for Extension of Time (2 copies)
X Amendment and Response
___ ____________

X Transmittal Letter
___ Fee Transmittal (2 copies)
X Deposit Account Transaction
X Facsimile Transmission Certificate dated July 15, 2002
___ ____________
___ ____________

CERTIFICATE OF TRANSMISSION UNDER 37 C.F.R. §1.8

I hereby certify that this correspondence is being transmitted by facsimile to the Assistant Commissioner for Patents, Box Non-Fee Amendment, Washington, DC 20231 on July 15, 2002, Facsimile No. 703-746-7239.

Linda DeNardi
Name of person signing this certificate

Linda DeNardi July 15, 2002
Signature and date

B

PTO/SB/21 (08-00)

Please type a plus sign (+) inside this box ⟶ ⊞

Approved for use through 10/31/2002. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	09/607,672
Filing Date	6/30/00
First Named Inventor	HALVERSEN ET AL
Group Art Unit	2155
Examiner Name	T. Nguyen
Attorney Docket Number	SRI1P037C
Total Number of Pages in This Submission	

ENCLOSURES *(check all that apply)*

- ☐ Fee Transmittal Form
- ☐ Fee Attached
- ☒ Amendment / Response
- ☐ After Final
  - ☐ Affidavits/declaration(s)
- ☒ Extension of Time Request
- ☐ Express Abandonment Request
- ☐ Information Disclosure Statement
- ☐ Certified Copy of Priority Document(s)
- ☐ Response to Missing Parts/ Incomplete Application
  - ☐ Response to Missing Parts under 37 CFR 1.52 or 1.53
- ☐ Assignment Papers *(for an Application)*
- ☐ Drawing(s)
- ☐ Licensing-related Papers
- ☐ Petition Routing Slip (PTO/SB/69) and Accompanying Petition
- ☐ Petition to Convert to a Provisional Application
- ☐ Power of Attorney, Revocation Change of Correspondence Address
- ☐ Terminal Disclaimer
- ☐ Request for Refund
- ☐ CD, Number of CD(s)
- ☐ After Allowance Communication to Group
- ☐ Appeal Communication to Board of Appeals and Interferences
- ☐ Appeal Communication to Group *(Appeal Notice, Brief, Reply Brief)*
- ☐ Proprietary Information
- ☐ Status Letter
- ☒ Other Enclosure(s) *(please identify below)*:
  Certificate of Facsimile Transmission

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm or Individual name	KIN-WAH TONG, Reg. No. 39,400
Signature	
Date	July 15, 2002

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be send to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

Notice of Allowability	**Application No.** 09/607,672	**Applicant(s)** HALVERSEN ET AL.	
	Examiner Thu Ha T. Nguyen	**Art Unit** 2155	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *amendment B filed on 16 July 2002*.
2. ☒ The allowed claim(s) is/are *56-76*.
3. □ The drawings filed on _____ are accepted by the Examiner.
4. □ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a) □ All b) □ Some* c) □ None of the:
   1. □ Certified copies of the priority documents have been received.
   2. □ Certified copies of the priority documents have been received in Application No. _____ .
   3. □ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.
5. □ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).
   (a) □ The translation of the foreign language provisional application has been received.
6. □ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application. **THIS THREE-MONTH PERIOD IS NOT EXTENDABLE**

7. □ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.

8. □ CORRECTED DRAWINGS must be submitted.
   (a) □ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
   1) □ hereto or 2) □ to Paper No. _____.
   (b) □ including changes required by the proposed drawing correction filed _____, which has been approved by the Examiner.
   (c) □ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No. _____.

Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the top margin (not the back) of each sheet. The drawings should be filed as a separate paper with a transmittal letter addressed to the Official Draftsperson.

9. □ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1□ Notice of References Cited (PTO-892)
3□ Notice of Draftperson's Patent Drawing Review (PTO-948)
5□ Information Disclosure Statements (PTO-1449), Paper No. _____.
7□ Examiner's Comment Regarding Requirement for Deposit of Biological Material

2□ Notice of Informal Patent Application (PTO-152)
4□ Interview Summary (PTO-413), Paper No. _____.
6□ Examiner's Amendment/Comment
8□ Examiner's Statement of Reasons for Allowance
9□ Other

AYAZ SHEIKH
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

NOTICE OF ALLOWANCE AND FEE(S) DUE

#17

7590 09/23/2002

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY, NJ 07702

EXAMINER	
NGUYEN, THU HA T	
ART UNIT	CLASS-SUBCLASS
2155	709-202000

DATE MAILED: 09/23/2002

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/607,672	06/30/2000	Christine Halversen	SRI1P037C	1291

TITLE OF INVENTION: SYSTEM, METHOD, AND ARTICLE OF MANUFACTURE FOR AGENT-BASED NAVIGATION IN A SPEECH-BASED DATA NAVIGATION SYSTEM

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	YES	$640	$0	$640	12/23/2002

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED. THIS NOTICE OF ALLOWANCE IS NOT A GRANT OF PATENT RIGHTS. THIS APPLICATION IS SUBJECT TO WITHDRAWAL FROM ISSUE AT THE INITIATIVE OF THE OFFICE OR UPON PETITION BY THE APPLICANT. SEE 37 CFR 1.313 AND MPEP 1308.

THE ISSUE FEE AND PUBLICATION FEE (IF REQUIRED) MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED. SEE 35 U.S.C. 151. THE ISSUE FEE DUE INDICATED ABOVE REFLECTS A CREDIT FOR ANY PREVIOUSLY PAID ISSUE FEE APPLIED IN THIS APPLICATION. THE PTOL-85B (OR AN EQUIVALENT) MUST BE RETURNED WITHIN THIS PERIOD EVEN IF NO FEE IS DUE OR THE APPLICATION WILL BE REGARDED AS ABANDONED.

HOW TO REPLY TO THIS NOTICE:

I. Review the SMALL ENTITY status shown above.

If the SMALL ENTITY is shown as YES, verify your current SMALL ENTITY status:

A. If the status is the same, pay the TOTAL FEE(S) DUE shown above.

B. If the status is changed, pay the PUBLICATION FEE (if required) and twice the amount of the ISSUE FEE shown above and notify the United States Patent and Trademark Office of the change in status, or

If the SMALL ENTITY is shown as NO:

A. Pay TOTAL FEE(S) DUE shown above, or

B. If applicant claimed SMALL ENTITY status before, or is now claiming SMALL ENTITY status, check the box below and enclose the PUBLICATION FEE and 1/2 the ISSUE FEE shown above.

□ Applicant claims SMALL ENTITY status.
See 37 CFR 1.27.

II. PART B - FEE(S) TRANSMITTAL should be completed and returned to the United States Patent and Trademark Office (USPTO) with your ISSUE FEE and PUBLICATION FEE (if required). Even if the fee(s) have already been paid, Part B - Fee(s) Transmittal should be completed and returned. If you are charging the fee(s) to your deposit account, section "4b" of Part B - Fee(s) Transmittal should be completed and an extra copy of the form should be submitted.

III. All communications regarding this application must give the application number. Please direct all communications prior to issuance to Box ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Box ISSUE FEE**
Commissioner for Patents
Washington, D.C. 20231
Fax (703)746-4000

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 4 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Legibly mark-up with any corrections or use Block 1)

7590 09/23/2002

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY, NJ 07702

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Box Issue Fee address above, or being facsimile transmitted to the USPTO, on the date indicated below.

(Depositor's name)

(Signature)

(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/607,672	06/30/2000	Christine Halversen	SRI1P037C	1291

TITLE OF INVENTION: SYSTEM, METHOD, AND ARTICLE OF MANUFACTURE FOR AGENT-BASED NAVIGATION IN A SPEECH-BASED DATA NAVIGATION SYSTEM

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	YES	$640	$0	$640	12/23/2002

EXAMINER	ART UNIT	CLASS-SUBCLASS
NGUYEN, THU HA T	2155	709-202000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).

□ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.

□ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list (1) the names of up to 3 registered patent attorneys or agents OR, alternatively, (2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

1________________

2________________

3________________

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. Inclusion of assignee data is only appropriate when an assignment has been previously submitted to the USPTO or is being submitted under separate cover. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE (B) RESIDENCE: (CITY and STATE OR COUNTRY)

Please check the appropriate assignee category or categories (will not be printed on the patent) □ individual □ corporation or other private group entity □ government

4a. The following fee(s) are enclosed:

□ Issue Fee

□ Publication Fee

□ Advance Order - # of Copies ____________

4b. Payment of Fee(s):

□ A check in the amount of the fee(s) is enclosed.

□ Payment by credit card. Form PTO-2038 is attached.

□ The Commissioner is hereby authorized by charge the required fee(s), or credit any overpayment, to Deposit Account Number ____________(enclose an extra copy of this form).

Commissioner for Patents is requested to apply the Issue Fee and Publication Fee (if any) or to re-apply any previously paid issue fee to the application identified above.

(Authorized Signature) (Date)

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, Washington, D.C. 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMIT THIS FORM WITH FEE(S)

PTOL-85 (REV. 04-02) Approved for use through 01/31/2004. OMB 0651-0033 U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/607,672	06/30/2000	Christine Halversen	SRI1P037C	1291

7590 09/23/2002

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY, NJ 07702

EXAMINER	
NGUYEN, THU HA T	
ART UNIT	PAPER NUMBER
2155	

DATE MAILED: 09/23/2002

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)

(application filed on or after May 29, 2000)

The patent term adjustment to date is 0 days. If the issue fee is paid on the date that is three months after the mailing date of this notice and the patent issues on the Tuesday before the date that is 28 weeks (six and a half months) after the mailing date of this notice, the term adjustment will be 0 days.

If a continued prosecution application (CPA) was filed in the above-identified application, the filing date that determines patent term adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) system. (http://pair.uspto.gov)

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/607,672	06/30/2000	Christine Halversen	SRI1P037C	1291

7590 09/23/2002

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY, NJ 07702
UNITED STATES

EXAMINER	
NGUYEN, THU HA T	
ART UNIT	PAPER NUMBER
2155	

DATE MAILED: 09/23/2002

Notice of Possible Fee Increase on October 1, 2002

If a reply to a "Notice of Allowance and Fee(s) Due" is filed in the Office on or after October 1, 2002, then the amount due may be higher than that set forth in the "Notice of Allowance and Fee(s) Due" since there may be an increase in fees effective on October 1, 2002. See Revision of Patent and Trademark Fees for Fiscal Year 2003; Notice of Proposed Rulemaking, 67 Fed. Reg. 30634, 30636 (May 7, 2002). Although a change to the amount of the publication fee is not currently proposed for October 2002, if the issue fee or publication fee is to be paid on or after October 1, 2002, applicant should check the USPTO web site for the current fees before submitting the payment. The USPTO Internet address for the fee schedule is: http://www.uspto.gov/main/howtofees.htm.

If the issue fee paid is the amount shown on the "Notice of Allowance and Fee(s) Due," but not the correct amount in view of any fee increase, a "Notice to Pay Balance of Issue Fee" will be mailed to applicant. In order to avoid processing delays associated with mailing of a "Notice to Pay Balance of Issue Fee," if the response to the Notice of Allowance and Fee(s) due form is to be filed on or after October 1, 2002 (or mailed with a certificate of mailing on or after October 1, 2002), the issue fee paid should be the fee that is required at the time the fee is paid. If the issue fee was previously paid, and the response to the "Notice of Allowance and Fee(s) Due" includes a request to apply a previously-paid issue fee to the issue fee now due, then the difference between the issue fee amount at the time the response is filed and the previously paid issue fee should be paid. See Manual of Patent Examining Procedure, Section 1308.01 (Eighth Edition, August 2001).

Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at (703) 305-8283.

PTOL-85 (REV. 04-02) Approved for use through 01/31/2004.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/607,672	06/30/2000	Christine Halversen	SRI1P037C	1291

7590 01/07/2003

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY, NJ 07702

EXAMINER	
NGUYEN, THU HA T	
ART UNIT	PAPER NUMBER
2155	18

DATE MAILED: 01/07/2003

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 07-01)

Supplemental ***Notice of Allowability***	**Application No.** 09/607,672	**Applicant(s)** HALVERSEN ET AL.
	Examiner Thu Ha T. Nguyen	**Art Unit** 2155

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *amendment B filed on 16 July 2002*.
2. ☒ The allowed claim(s) is/are *56-57, 59-64, 66-71, and 73-76*.
3. ☐ The drawings filed on _____ are accepted by the Examiner.
4. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a) ☐ All b) ☐ Some* c) ☐ None of the:
   1. ☐ Certified copies of the priority documents have been received.
   2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
   3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.
5. ☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).
   (a) ☐ The translation of the foreign language provisional application has been received.
6. ☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application. **THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.**

7. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.

8. ☐ CORRECTED DRAWINGS must be submitted.
   (a) ☐ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
   1) ☐ hereto or 2) ☐ to Paper No. _____.
   (b) ☐ including changes required by the proposed drawing correction filed _____, which has been approved by the Examiner.
   (c) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No. _____.

Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the top margin (not the back) of each sheet. The drawings should be filed as a separate paper with a transmittal letter addressed to the Official Draftsperson.

9. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1☐ Notice of References Cited (PTO-892)
2☐ Notice of Informal Patent Application (PTO-152)
3☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
4☐ Interview Summary (PTO-413), Paper No._____ .
5☐ Information Disclosure Statements (PTO-1449), Paper No. _____.
6☐ Examiner's Amendment/Comment
7☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
8☐ Examiner's Statement of Reasons for Allowance
9☐ Other

AYAZ SHEIKH
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

#19

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: C. Halversen

Serial No.:	09/607,672	Art Unit: 2155
Filing Date:	June 30, 2000	Examiner: T. Nguyen
For:	SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR AGENT-BASED NAVIGATION IN A SPEECH-BASED DATA NAVIGATION SYSTEM	
Docket No.	SRI/4116-7	

Assistant Commissioner for Patents
Washington, D.C. 20231

SIR:

SUBMISSION OF FORMAL DRAWINGS

The Applicants submit herewith 7 sheets of formal drawings (FIGS. 1 through 6), properly labeled, in connection with the above-captioned application. The Examiner is requested to substitute these formal drawings for the informal drawings previously submitted.

Respectfully submitted,

12/20/02

KIN-WAH TONG, ESQ.
Reg. No. 39,400
(732) 530-9404

Moser, Patterson & Sheridan, LLP
595 Shrewsbury Avenue
Suite 100
Shrewsbury, NJ 07702

07-02

CERTIFICATE OF MAILING under 37 C.F.R. 1.8(a)

I hereby certify that this correspondence is being deposited on 12-20-2002, with the United States Postal Service as first class mail, with sufficient postage, in an envelope addressed to the Assistant Commissioner for Patents, Box Issue Fee, Washington, D.C. 20231.

Signature

December 20, 2002
Date of signature

Serial No. 09/607,672
Inventor: HALVERSEN
Title: System, Method, and Article of Manufacture for Agent-Based Navigation in a Speech-Based Data Navigation System

1/7

6523061

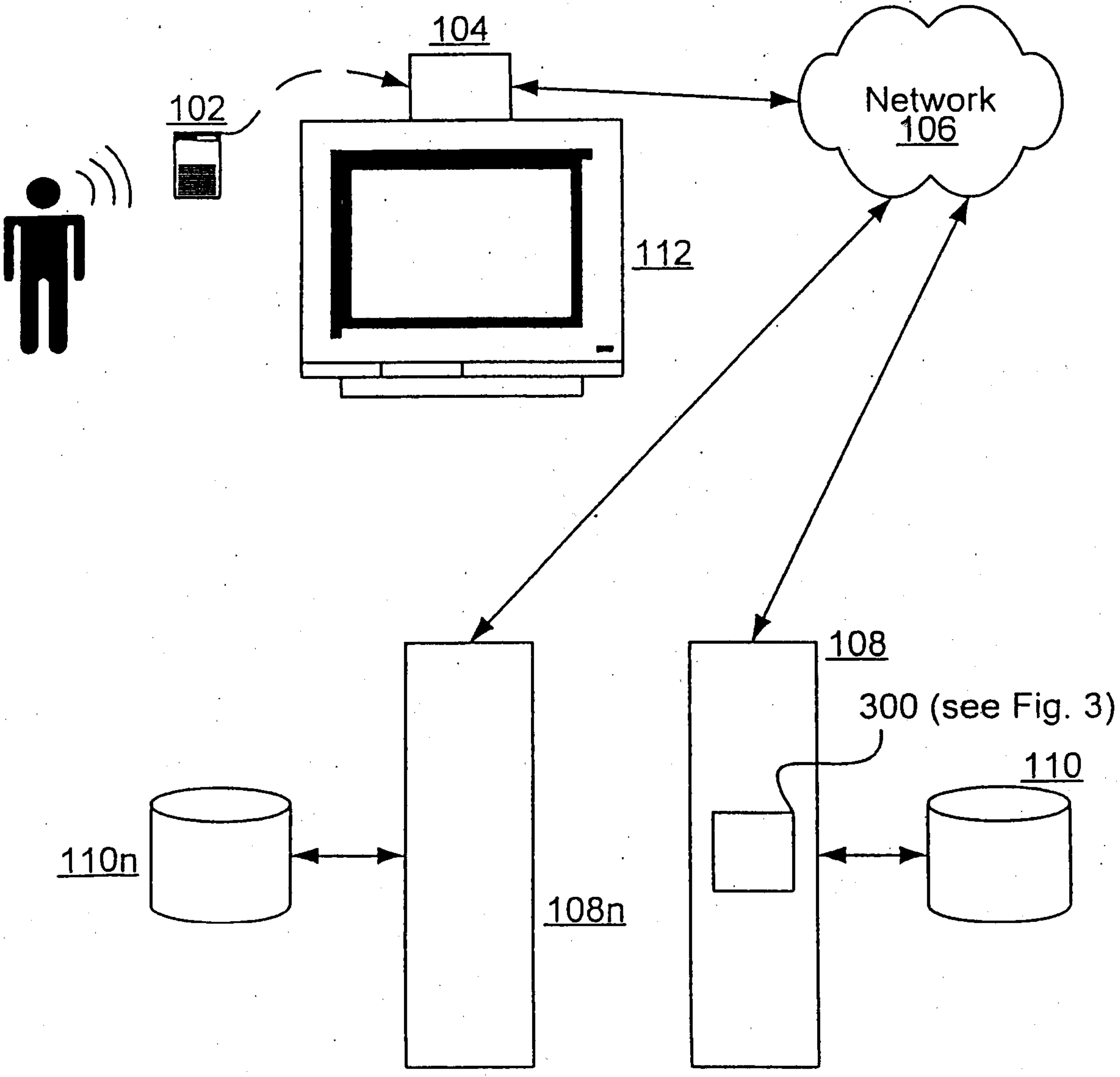


Fig. 1a

Serial No. 09/607,672
Inventor: HALVERSEN
Title: System, Method, and Article of Manufacture for Agent-Based Navigation in a Speech-Based Data Navigation System

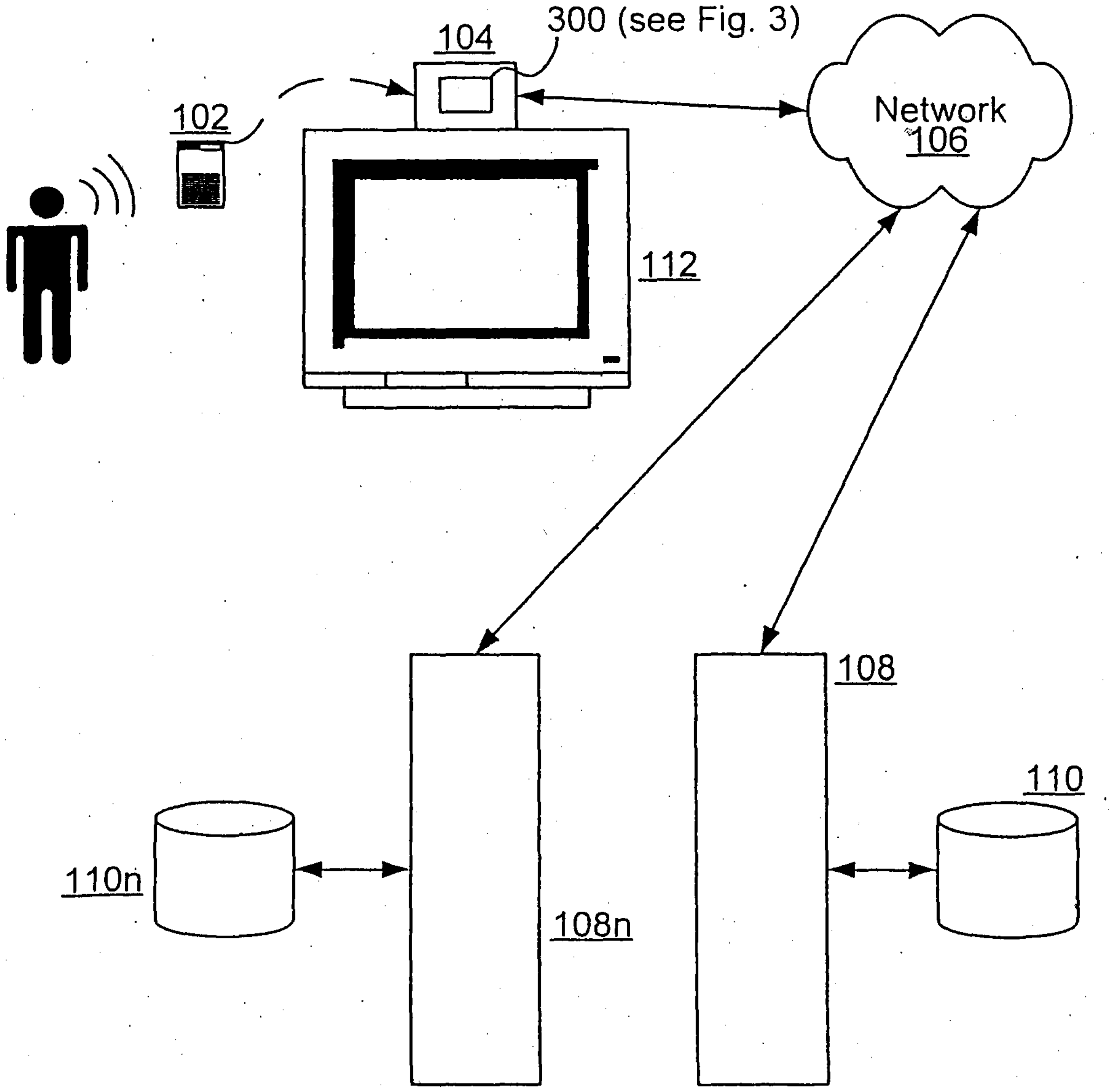


Fig. 1b

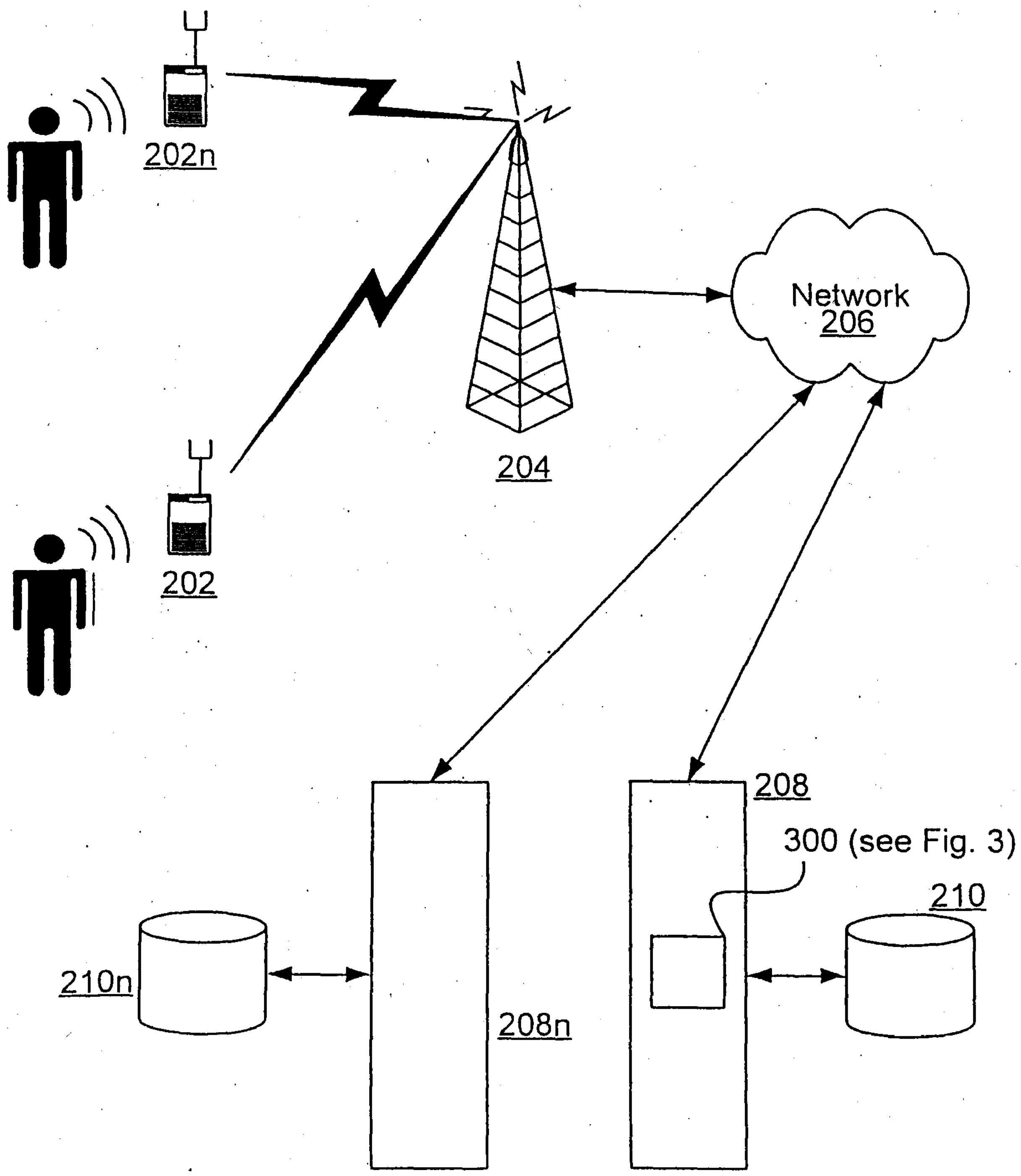


Fig. 2

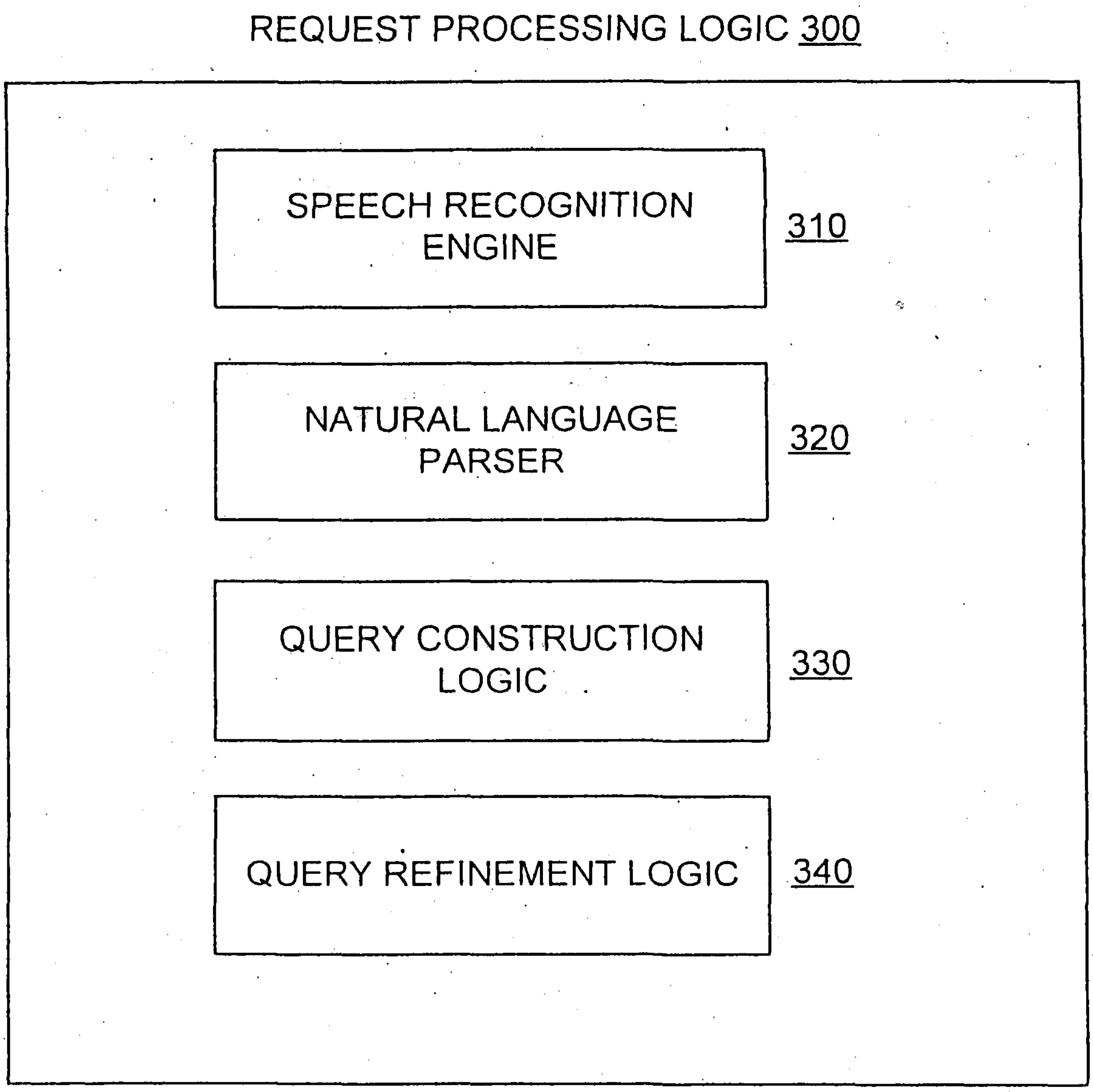

REQUEST PROCESSING LOGIC 300
SPEECH RECOGNITION ENGINE
310
NATURAL LANGUAGE PARSER
320
QUERY CONSTRUCTION LOGIC
330
QUERY REFINEMENT LOGIC
340

Fig. 3

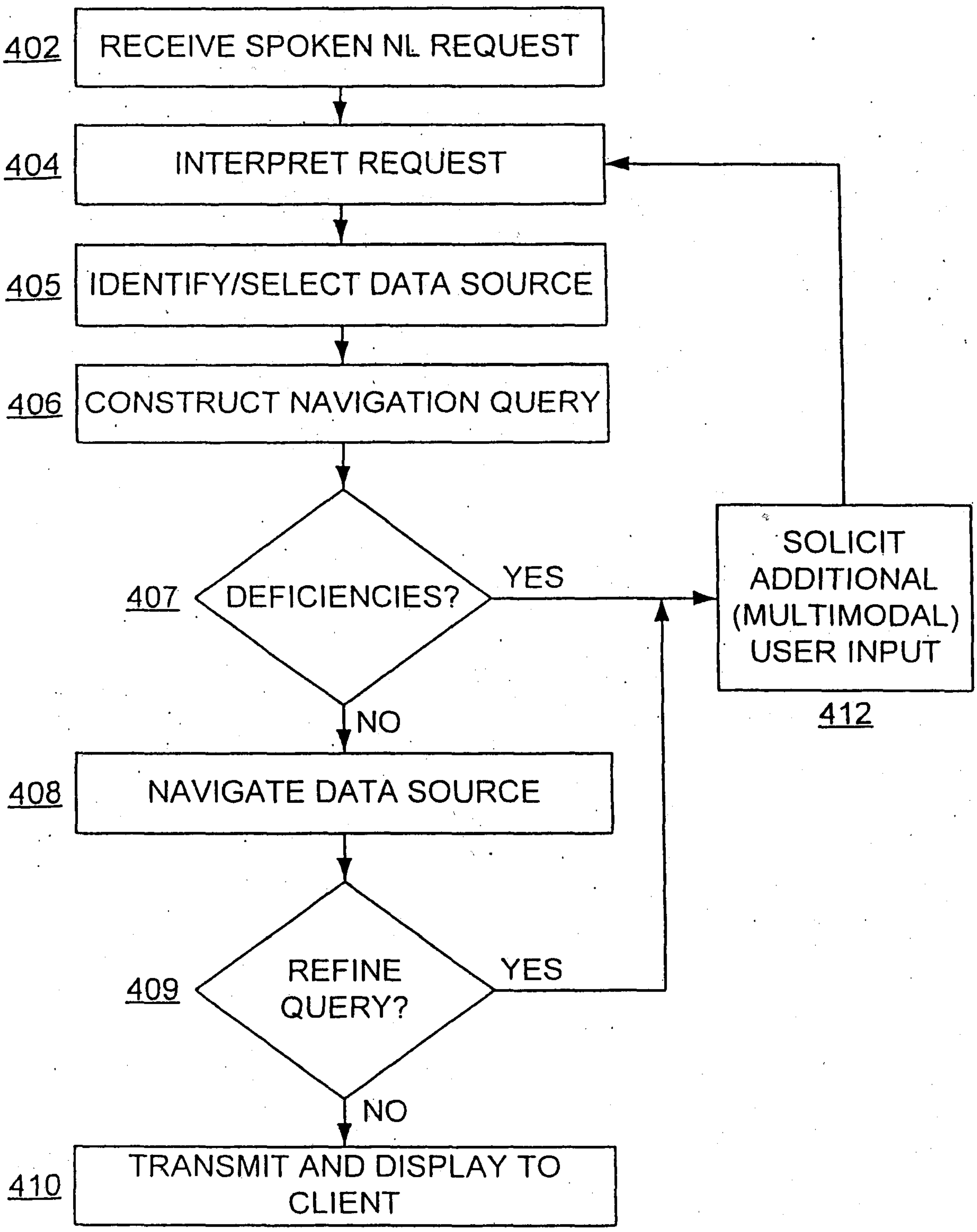

402
RECEIVE SPOKEN NL REQUEST
404
INTERPRET REQUEST
405
IDENTIFY/SELECT DATA SOURCE
406
CONSTRUCT NAVIGATION QUERY
407
DEFICIENCIES?
YES
NO
SOLICIT ADDITIONAL (MULTIMODAL) USER INPUT
412
408
NAVIGATE DATA SOURCE
409
REFINE QUERY?
YES
NO
410
TRANSMIT AND DISPLAY TO CLIENT

Fig. 4

Serial No. 09/607,672
Inventor: HALVERSEN
Title: System, Method, and Article of Manufacture for Agent-Based Navigation in a Speech-Based Data Navigation System

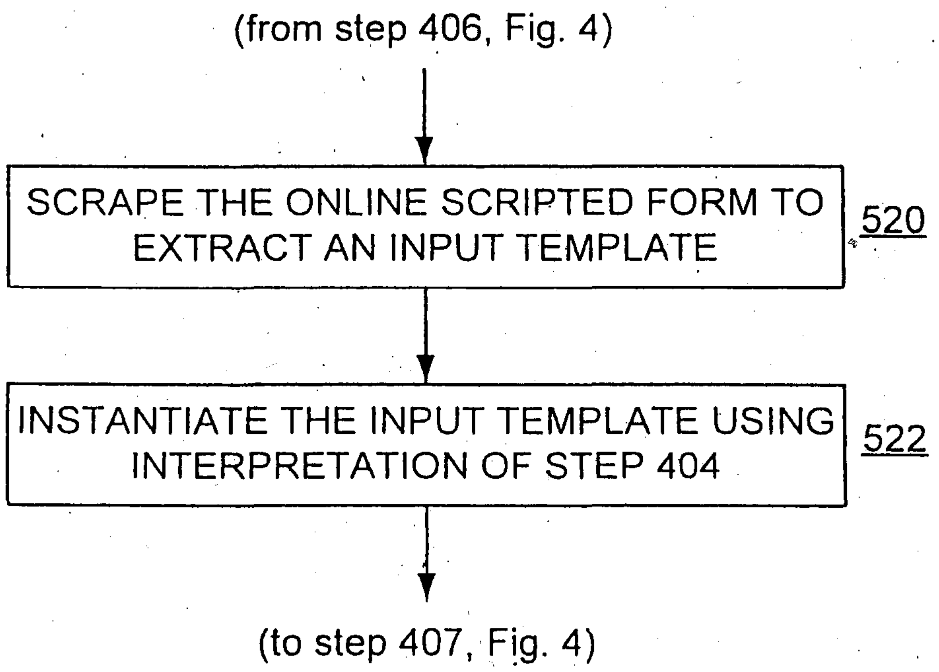

(from step 406, Fig. 4)
SCRAPE THE ONLINE SCRIPTED FORM TO EXTRACT AN INPUT TEMPLATE
520
INSTANTIATE THE INPUT TEMPLATE USING INTERPRETATION OF STEP 404
522
(to step 407, Fig. 4)

Fig. 5

Serial No. 09/607,672
Inventor: HALVERSEN
Title: System, Method, and Article of Manufacture for Agent-Based Navigation in a Speech-Based Data Navigation System

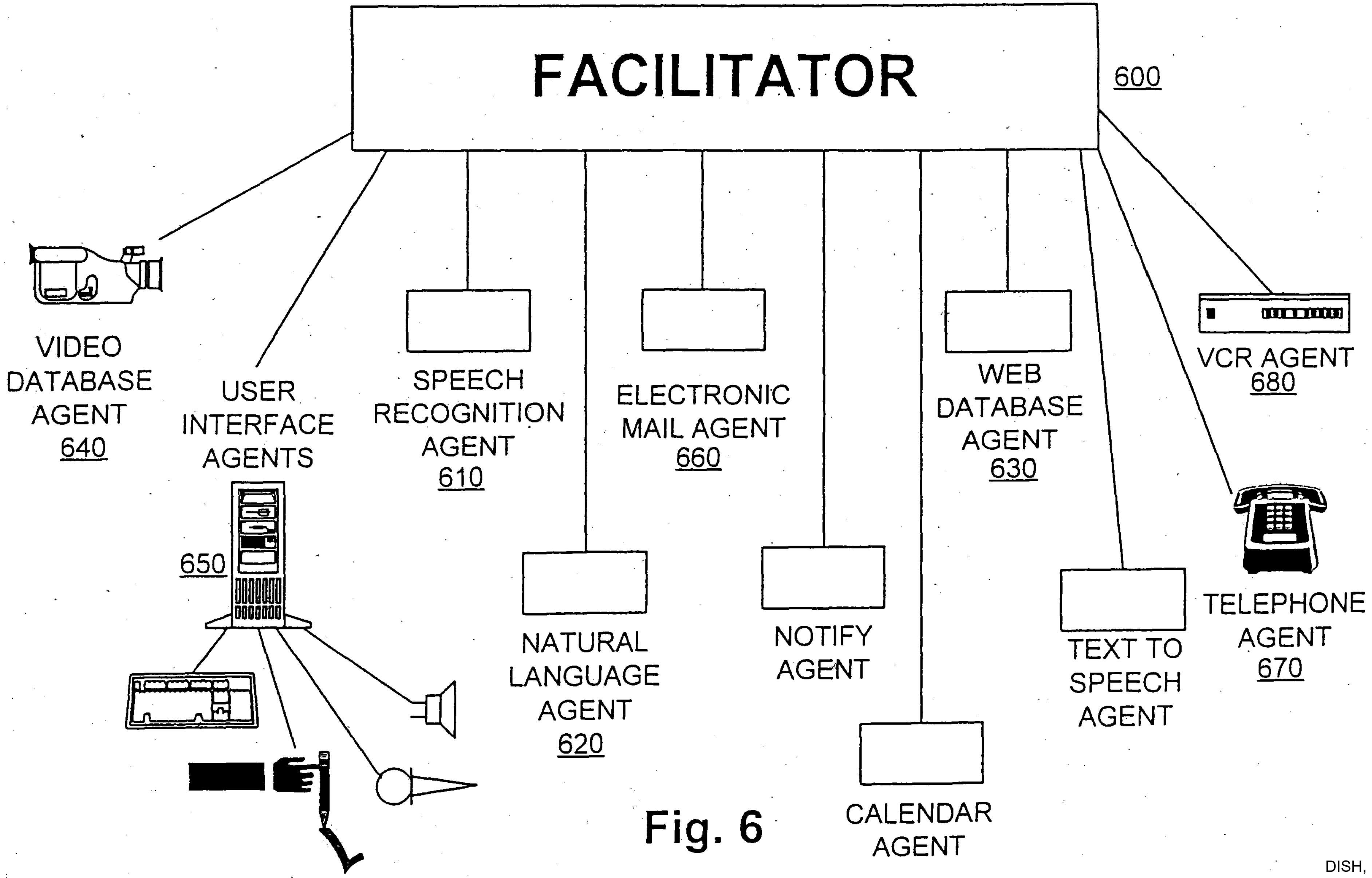


Fig. 6

B $ CW

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Box ISSUE FEE**
Commissioner for Patents
Washington, D.C. 20231
Fax (703)746-4000

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 4 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Legibly mark-up with any corrections or use Block 1)

7590 09/23/2002

THOMASON, MOSER & PATTERSON, LLP
595 SHREWSBURY AVENUE
SUITE 100
SHREWSBURY, NJ 07702

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Box Issue Fee address above, or being facsimile transmitted to the USPTO, on the date indicated below.

Barbara J. Jackson (Depositor's name)
(Signature)
December 20, 2002 (Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/607,672	06/30/2000	Christine Halversen	SRI1P037C	1291

TITLE OF INVENTION: SYSTEM, METHOD, AND ARTICLE OF MANUFACTURE FOR AGENT-BASED NAVIGATION IN A SPEECH-BASED DATA NAVIGATION SYSTEM

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	YES	$640	$0	$640	12/23/2002

EXAMINER	ART UNIT	CLASS-SUBCLASS
NGUYEN, THU HA T	2155	709-202000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).

□ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.

□ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list (1) the names of up to 3 registered patent attorneys or agents OR, alternatively, (2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

1 Moser, Patterson &
2 Sheridan, LLP.
3 Kin-Wah Tong, Esq.

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. Inclusion of assignee data is only appropriate when an assignment has been previously submitted to the USPTO or is being submitted under separate cover. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE: SRI International, Inc.

(B) RESIDENCE: (CITY and STATE OR COUNTRY): Menlo Park, CA

Please check the appropriate assignee category or categories (will not be printed on the patent) □ individual ☒ corporation or other private group entity □ government

4a. The following fee(s) are enclosed:

☒ Issue Fee
□ Publication Fee
☒ Advance Order - # of Copies 10

4b. Payment of Fee(s):

□ A check in the amount of the fee(s) is enclosed.
□ Payment by credit card. Form PTO-2038 is attached.
☒ The Commissioner is hereby authorized by charge the required fee(s), or credit any overpayment, to Deposit Account Number 20-0782 (enclose an extra copy of this form).

Commissioner for Patents is requested to apply the Issue Fee and Publication Fee (if any) or to re-apply any previously paid issue fee to the application identified above.

(Authorized Signature) (Date) 12/30/02
Kin-Wah Tong, Reg. No. 39,400

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, Washington, D.C. 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

```
12/30/2002 MBIZUNE2 00000223 200782    09607672

01 FC:2501          640.00 CH
02 FC:8001           30.00 CH
```

TRANSMIT THIS FORM WITH FEE(S)

US006523061B1

(12) **United States Patent**
Halverson et al.

(10) **Patent No.: US 6,523,061 B1**
(45) **Date of Patent: Feb. 18, 2003**

(54) **SYSTEM, METHOD, AND ARTICLE OF MANUFACTURE FOR AGENT-BASED NAVIGATION IN A SPEECH-BASED DATA NAVIGATION SYSTEM**

(75) Inventors: **Christine Halverson**, San Jose, CA (US); **Luc Julia**, Menlo Park, CA (US); **Dimitris Voutsas**, Thessaloniki (GR); **Adam Cheyer**, Palo Alto, CA (US)

(73) Assignee: **SRI International, Inc.**, Menlo Park, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/607,672**

(22) Filed: **Jun. 30, 2000**

Related U.S. Application Data

(63) Continuation of application No. 09/524,095, filed on Mar. 13, 2000, which is a continuation-in-part of application No. 09/225,198, filed on Jan. 5, 1999.

(60) Provisional application No. 60/124,720, filed on Mar. 17, 1999, provisional application No. 60/124,719, filed on Mar. 17, 1999, and provisional application No. 60/124,718, filed on Mar. 17, 1999.

(51) **Int. Cl.**^{7} **G06F 15/16**

(52) **U.S. Cl.** **709/202**; 709/202; 709/217; 709/219; 379/88.01; 379/88.02; 379/88.22; 370/331; 704/270; 704/275

(58) **Field of Search** 395/700.12; 370/331, 370/202; 709/217–219, 227; 379/88.01, 88.17, 88.22, 88.02, 90.01, 900; 704/270, 275; 382/313; 707/203

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,197,005 A	3/1993	Shwartz et al.	364/419
5,386,556 A	1/1995	Hedin et al.	395/600
5,434,777 A	7/1995	Luciw	364/419.13
5,519,608 A	5/1996	Kupiec	364/419.08

(List continued on next page.)

OTHER PUBLICATIONS

http://www–3.ibm.com/software.speech/desktop/w9–pro.html. IBM Via Voice for windows, Pro USB edition release 9 by IBM corp.*

Stent, Amanda et al., "The CommandTalk Spoken Dialogue System", SRI International.

Moore, Robert et al., "CommandTalk: A Spoken–Language Interface for Battlefield Simulations", Oct. 23, 1997, SRI International.

Dowding, John et al., "Interpreting Language in Context in CommandTalk", Feb. 5, 1999, SRI International.

http://www.ai.sri.com/~oaa/infowiz.html, InfoWiz: An Animated Voice Interactive Information System, May 8, 2000.

Dowding, John, "Interleaving Syntax and Semantics in an Efficient Bottom–up Parser", SRI International.

Moore, Robert et al., "Combining Linguistic and Statistical Knowledge Sources in Natural–Language Processing for ATIS", SRI International.

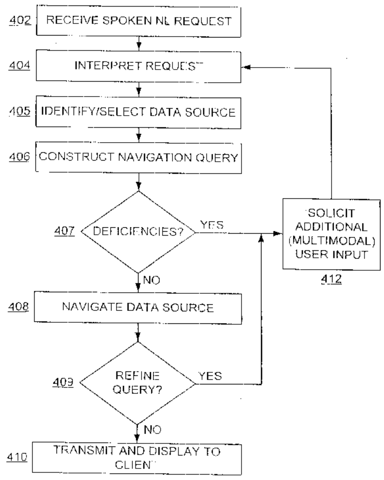
Dowding, John et al., "Gemini: A Natural Language System For Spoken–Language Understanding", SRI International.

Primary Examiner—Ayaz Sheikh
Assistant Examiner—Thu Ha Nguyen
(74) *Attorney, Agent, or Firm*—Moser, Patterson & Sheridan, LLP.; Kin-Wah Tong, Esq.

(57) **ABSTRACT**

A system, method, and article of manufacture are provided for navigating an electronic data source by means of spoken language where a portion of the data link between a mobile information appliance of the user and the data source utilizes wireless communication. When a spoken input request is received from a user, it is interpreted. The resulting interpretation of the request is thereupon used to automatically construct an operational navigation query. The navigation query is routed to one or more agents, which use the navigation query to retrieve the desired information from one or more electronic network data sources.

18 Claims, 7 Drawing Sheets


402
RECEIVE SPOKEN NL REQUEST
404
INTERPRET REQUEST
405
IDENTIFY/SELECT DATA SOURCE
406
CONSTRUCT NAVIGATION QUERY
407
DEFICIENCIES?
YES
NO
SOLICIT ADDITIONAL (MULTIMODAL) USER INPUT
412
408
NAVIGATE DATA SOURCE
409
REFINE QUERY?
YES
NO
410
TRANSMIT AND DISPLAY TO CLIENT

U.S. PATENT DOCUMENTS

5,608,624	A		3/1997	Luciw	395/794
5,717,860	A	*	2/1998	Graber et al.	395/200.12
5,721,938	A		2/1998	Stuckey	395/754
5,729,659	A		3/1998	Potter	395/2.79
5,748,974	A		5/1998	Johnson	395/759
5,774,859	A		6/1998	Houser et al.	704/275
5,794,050	A		8/1998	Dahlgren et al.	395/708
5,802,526	A		9/1998	Fawcett et al.	707/104
5,805,775	A		9/1998	Eberman et al.	395/12
5,855,002	A		12/1998	Armstrong	704/270
5,884,262	A	*	3/1999	Wise et al.	704/270
5,890,123	A		3/1999	Brown et al.	704/275
5,902,353	A	*	5/1999	Reber et al.	709/219
5,949,772	A	*	9/1999	Sugikawa et al.	370/331
5,963,940	A		10/1999	Liddy et al.	707/5
5,978,848	A	*	11/1999	Maddalozzo, Jr. et al.	709/227
6,003,072	A		12/1999	Gerritsen et al.	709/218
6,012,030	A		1/2000	French-St. George et al.	704/275
6,026,388	A		2/2000	Liddy et al.	707/1
6,026,437	A	*	2/2000	Muschett et al.	709/219
6,052,716	A	*	4/2000	Gibson	709/217
6,101,473	A	*	8/2000	Scott et al.	704/275
6,157,705	A	*	12/2000	Perrone	379/88.01
6,282,270	B1	*	8/2001	Porter	379/88.17
6,282,511	B1	*	8/2001	Mayer	704/270
6,289,140	B1	*	9/2001	Oliver	382/313

* cited by examiner

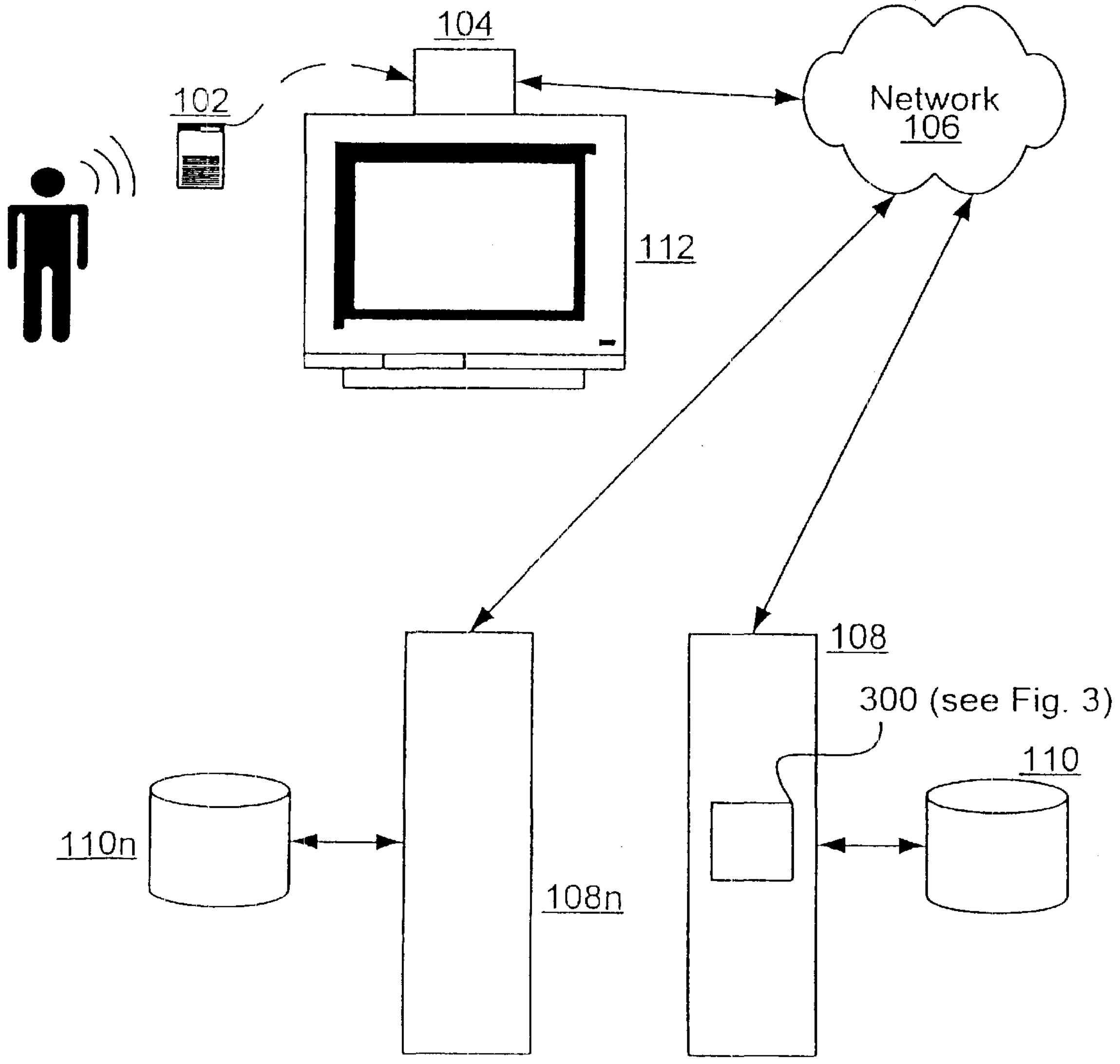


Fig. 1a

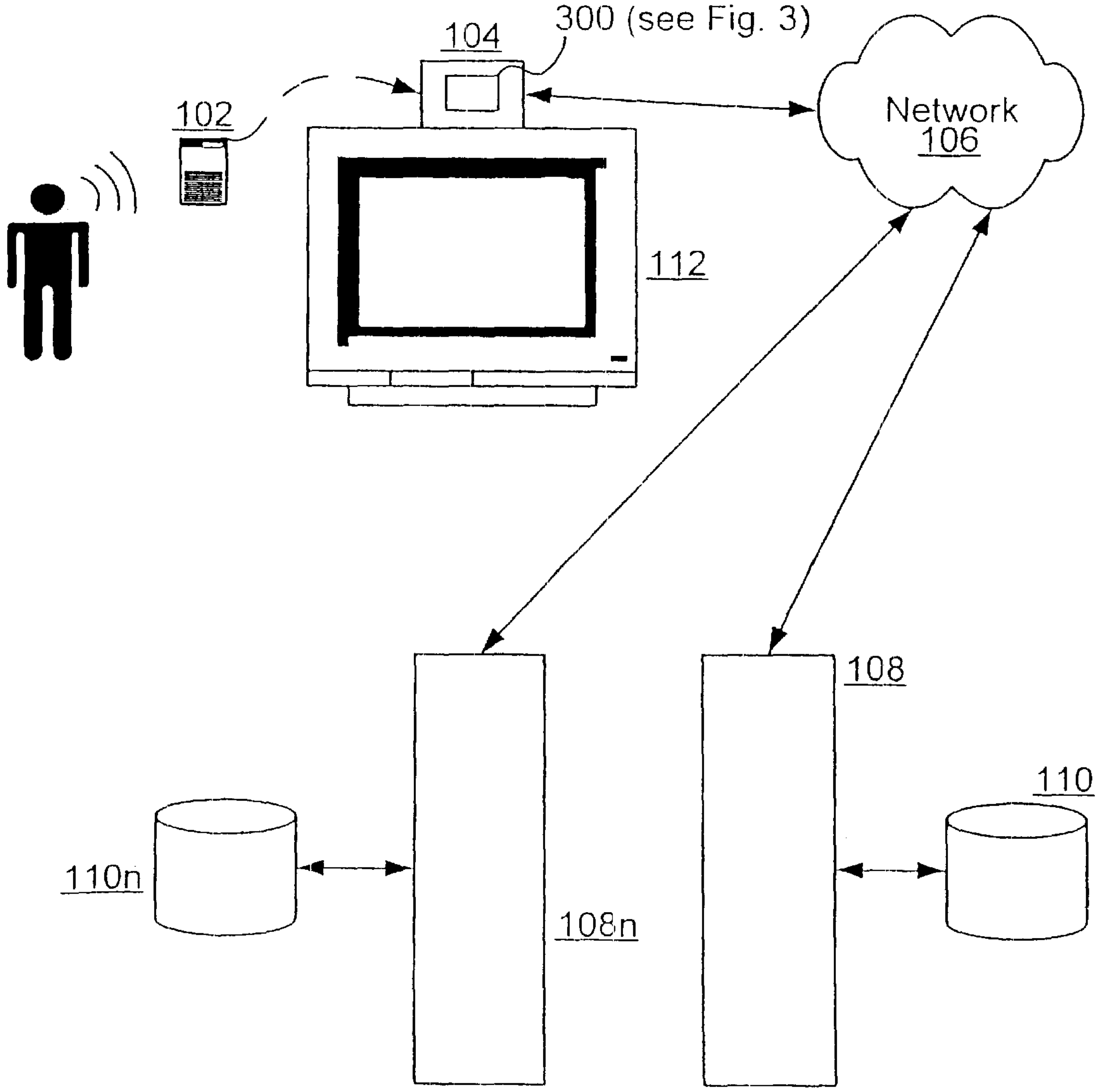


Fig. 1b

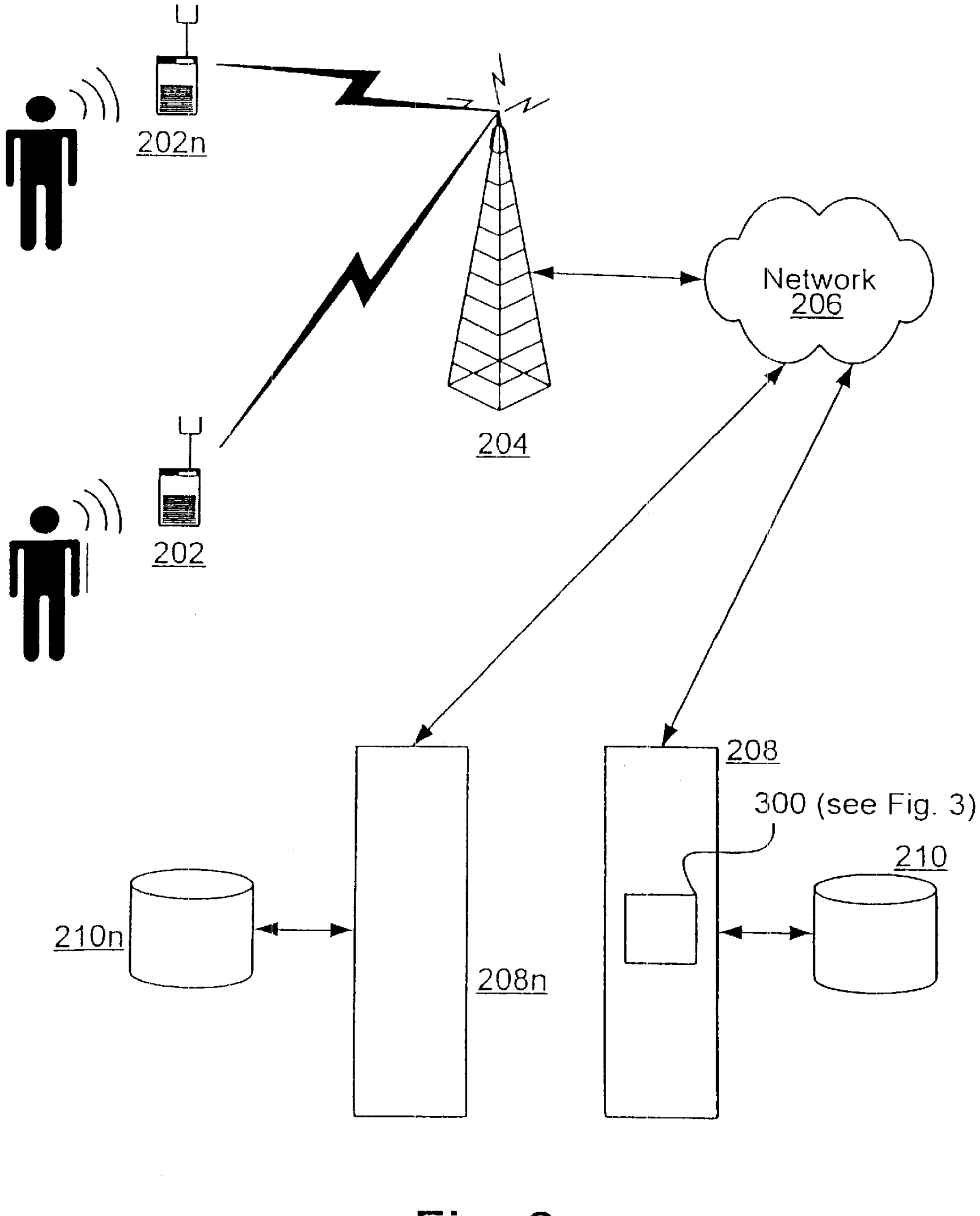


Fig. 2

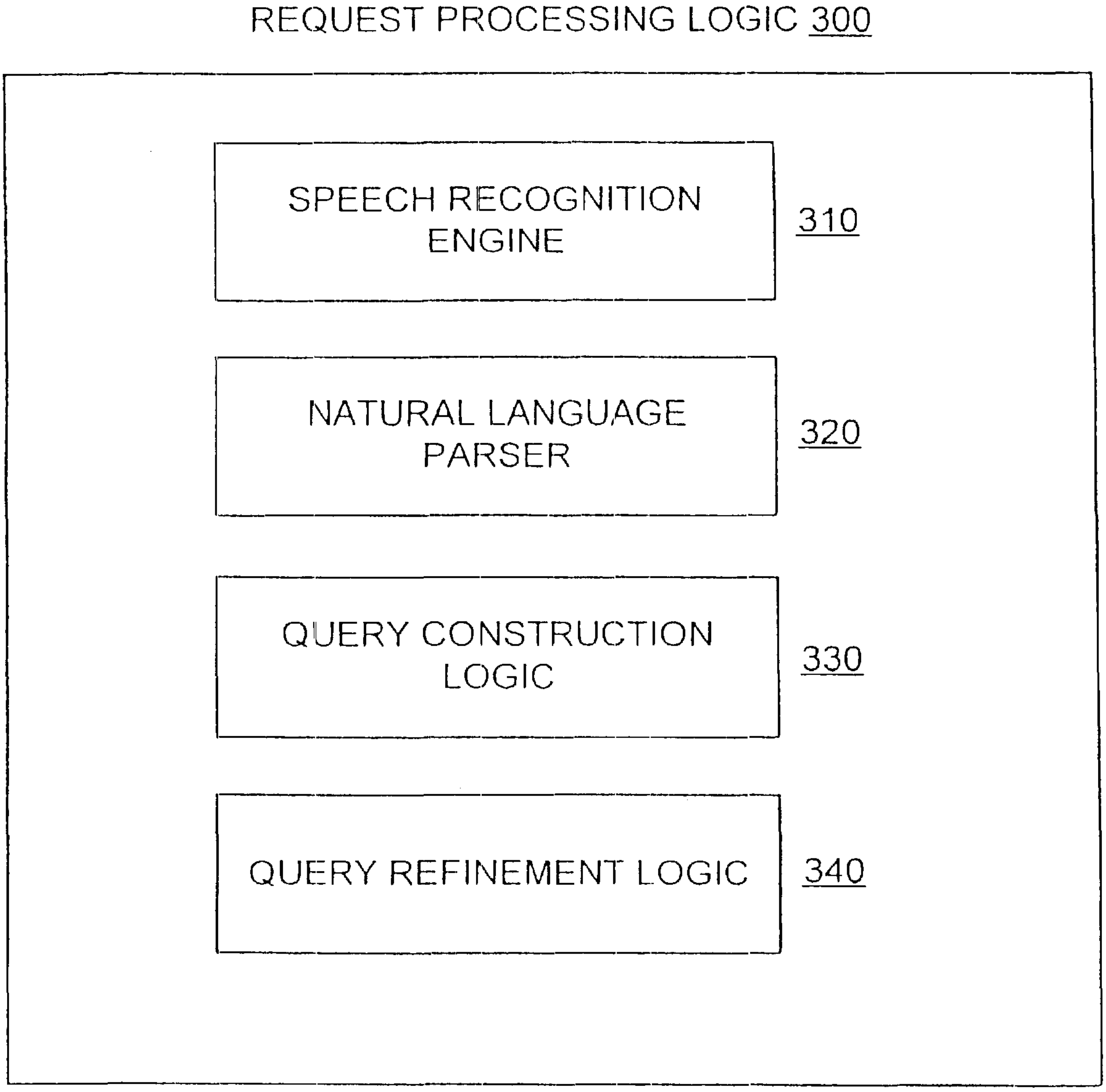

REQUEST PROCESSING LOGIC 300
SPEECH RECOGNITION ENGINE
310
NATURAL LANGUAGE PARSER
320
QUERY CONSTRUCTION LOGIC
330
QUERY REFINEMENT LOGIC
340

Fig. 3

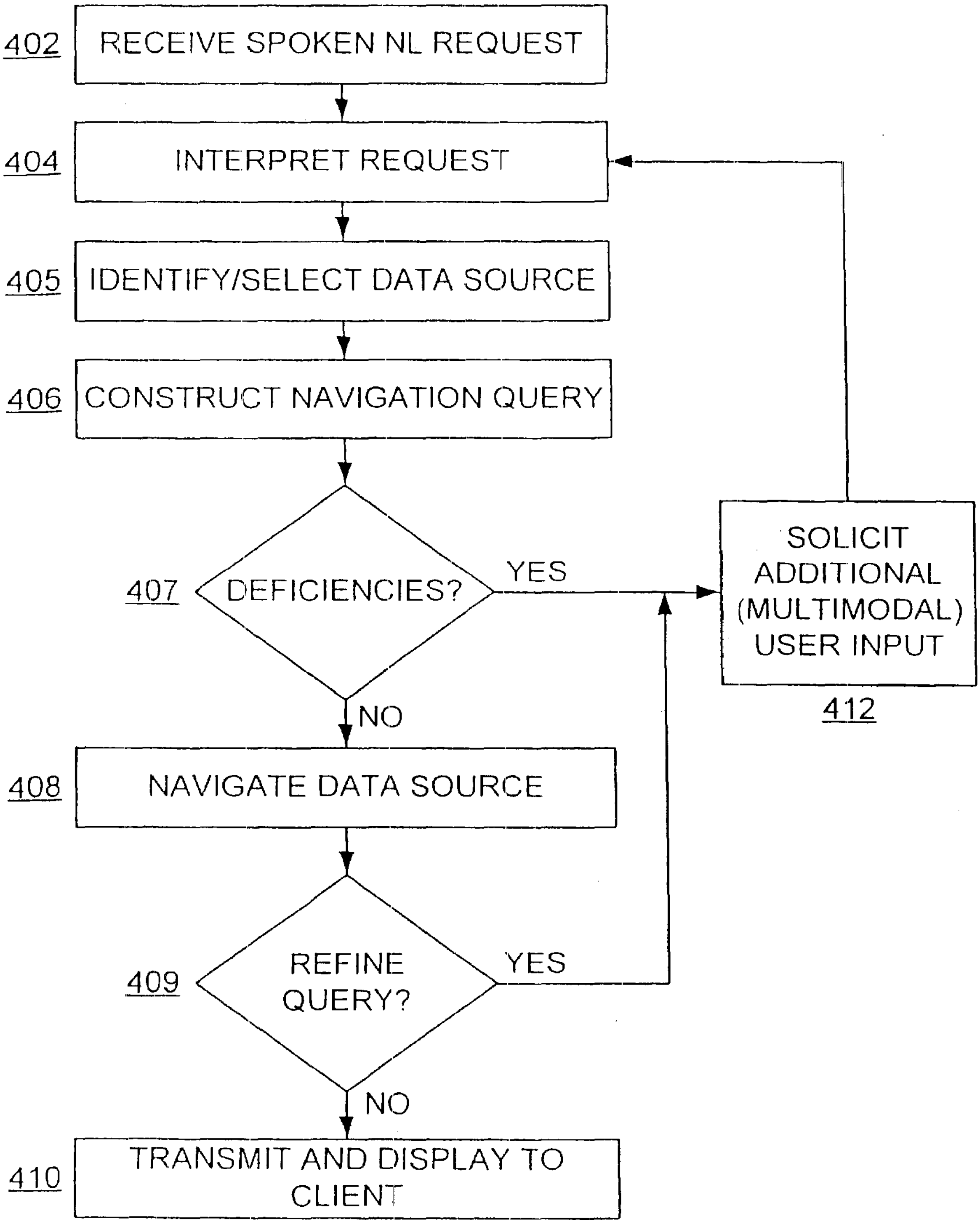

402
RECEIVE SPOKEN NL REQUEST
404
INTERPRET REQUEST
405
IDENTIFY/SELECT DATA SOURCE
406
CONSTRUCT NAVIGATION QUERY
407
DEFICIENCIES?
YES
NO
SOLICIT ADDITIONAL (MULTIMODAL) USER INPUT
412
408
NAVIGATE DATA SOURCE
409
REFINE QUERY?
YES
NO
410
TRANSMIT AND DISPLAY TO CLIENT

Fig. 4

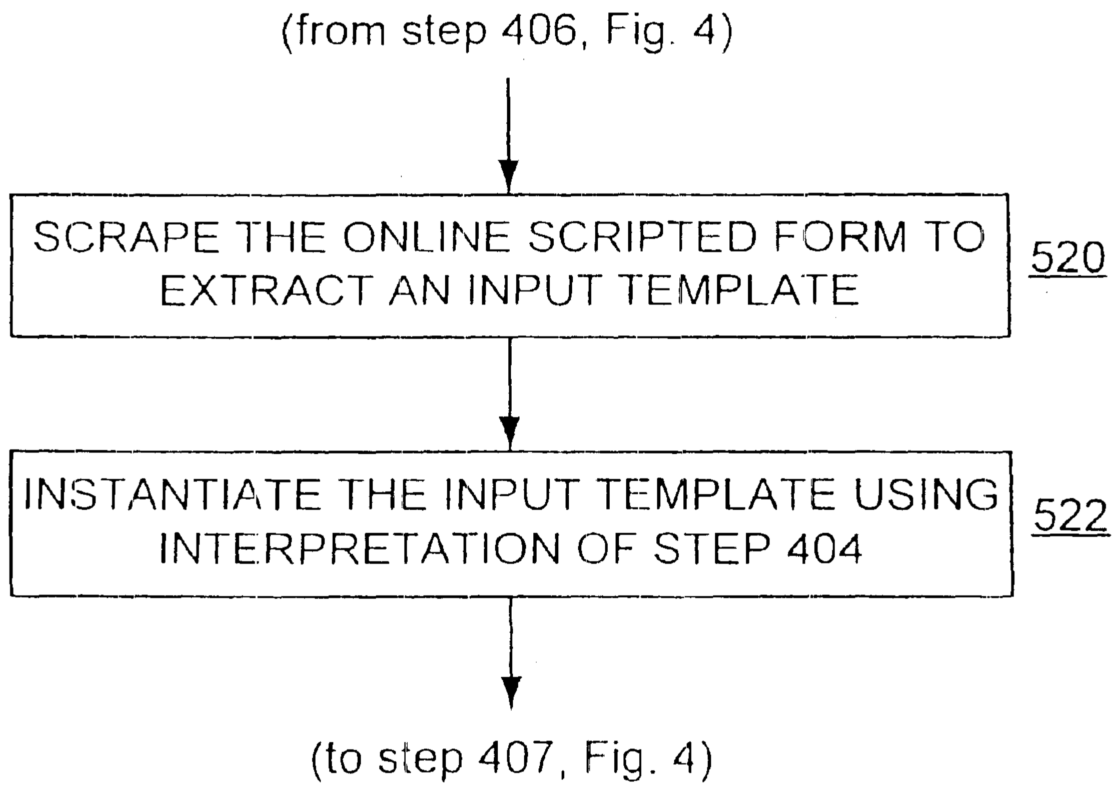

(from step 406, Fig. 4)
SCRAPE THE ONLINE SCRIPTED FORM TO EXTRACT AN INPUT TEMPLATE
520
INSTANTIATE THE INPUT TEMPLATE USING INTERPRETATION OF STEP 404
522
(to step 407, Fig. 4)

Fig. 5

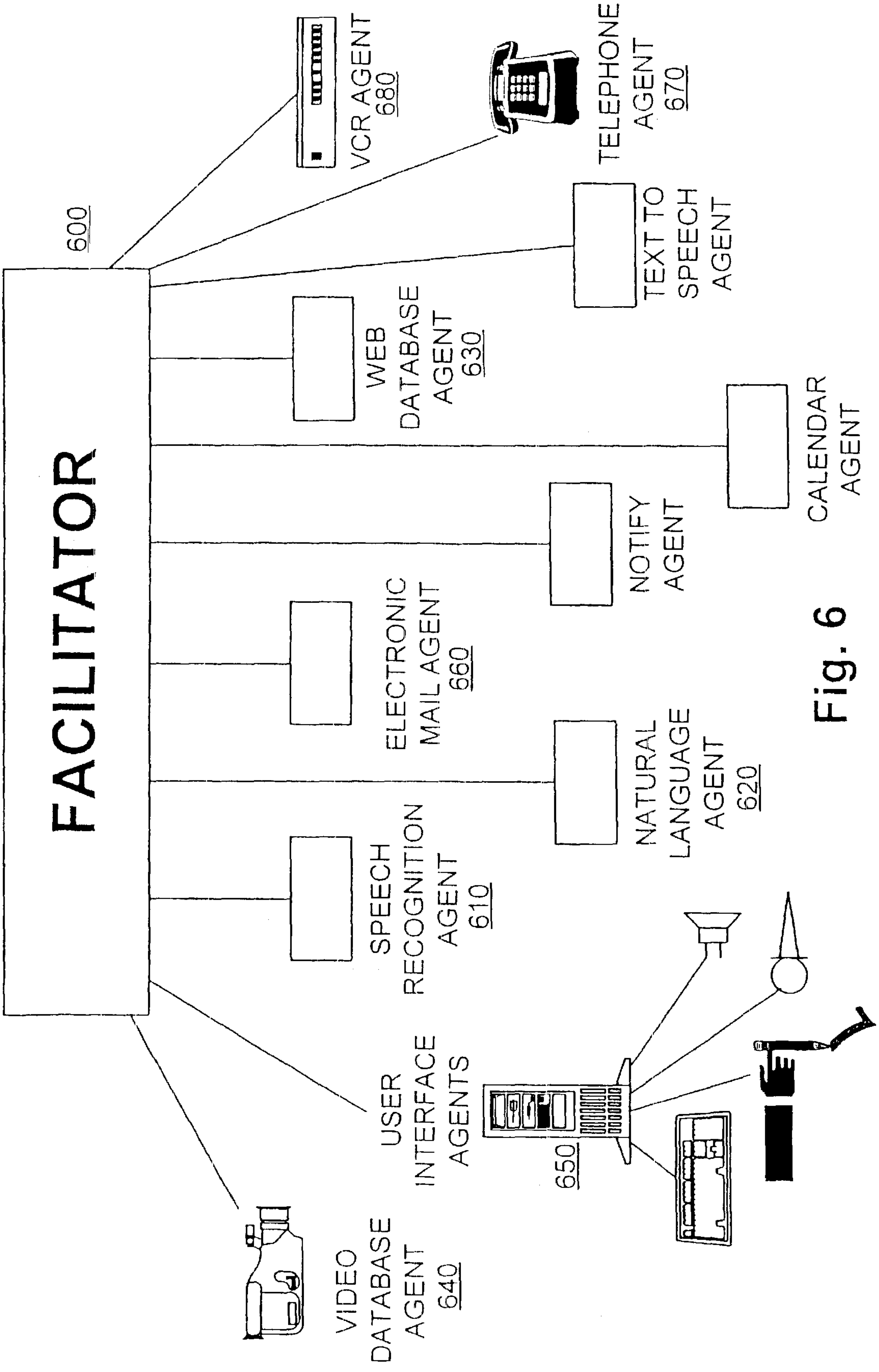

FACILITATOR
600
VCR AGENT
680
TELEPHONE
AGENT
670
TEXT TO
SPEECH
AGENT
WEB
DATABASE
AGENT
630
CALENDAR
AGENT
NOTIFY
AGENT
ELECTRONIC
MAIL AGENT
660
NATURAL
LANGUAGE
AGENT
620
SPEECH
RECOGNITION
AGENT
610
USER
INTERFACE
AGENTS
650
VIDEO
DATABASE
AGENT
640

Fig. 6

SYSTEM, METHOD, AND ARTICLE OF MANUFACTURE FOR AGENT-BASED NAVIGATION IN A SPEECH-BASED DATA NAVIGATION SYSTEM

This application is a continuation of an application entitled NAVIGATING NETWORK-BASED ELECTRONIC INFORMATION USING SPOKEN NATURAL LANGUAGE INPUT WITH MULTIMODAL ERROR FEEDBACK which was filed on Mar. 13, 2000 under Ser. No. 09/524,095 and which is a Continuation In Part of co-pending U.S. patent application Ser. No. 09/225,198, filed Jan. 5, 1999, Provisional U.S. patent application Ser. No. 60/124,718, filed Mar. 17, 1999, Provisional U.S. patent application Ser. No. 60/124,720, filed Mar. 17, 1999, and Provisional U.S. patent application Ser. No. 60/124,719, filed Mar. 17, 1999, from which applications priority is claimed and these application are incorporated herein by reference.

BACKGROUND OF THE INVENTION

The present invention relates generally to the navigation of electronic data by means of spoken natural language requests, and to feedback mechanisms and methods for resolving the errors and ambiguities that may be associated with such requests.

As global electronic connectivity continues to grow, and the universe of electronic data potentially available to users continues to expand, there is a growing need for information navigation technology that allows relatively naive users to navigate and access desired data by means of natural language input. In many of the most important markets—including the home entertainment arena, as well as mobile computing—spoken natural language input is highly desirable, if not ideal. As just one example, the proliferation of high-bandwidth communications infrastructure for the home entertainment market (cable, satellite, broadband) enables delivery of movies-on-demand and other interactive multimedia content to the consumer's home television set. For users to take full advantage of this content stream ultimately requires interactive navigation of content databases in a manner that is too complex for user-friendly selection by means of a traditional remote-control clicker. Allowing spoken natural language requests as the input modality for rapidly searching and accessing desired content is an important objective for a successful consumer entertainment product in a context offering a dizzying range of database content choices. As further examples, this same need to drive navigation of (and transaction with) relatively complex data warehouses using spoken natural language requests applies equally to surfing the Internet/Web or other networks for general information, multimedia content, or e-commerce transactions.

In general, the existing navigational systems for browsing electronic databases and data warehouses (search engines, menus, etc.), have been designed without navigation via spoken natural language as a specific goal. So today's world is full of existing electronic data navigation systems that do not assume browsing via natural spoken commands, but rather assume text and mouse-click inputs (or in the case of TV remote controls, even less). Simply recognizing voice commands within an extremely limited vocabulary and grammar—the spoken equivalent of button/click input (e.g., speaking "channel **5**" selects TV channel **5**)—is really not sufficient by itself to satisfy the objectives described above. In order to deliver a true "win" for users, the voice-driven front-end must accept spoken natural language input in a manner that is intuitive to users. For example, the front-end should not require learning a highly specialized command language or format. More fundamentally, the front-end must allow users to speak directly in terms of what the user ultimately wants—e.g., "I'd like to see a Western film directed by Clint Eastwood"—as opposed to speaking in terms of arbitrary navigation structures (e.g., hierarchical layers of menus, commands, etc.) that are essentially artifacts reflecting constraints of the pre-existing text/click navigation system. At the same time, the front-end must recognize and accommodate the reality that a stream of naive spoken natural language input will, over time, typically present a variety of errors and/or ambiguities: e.g., garbled/unrecognized words (did the user say "Eastwood" or "Easter"?) and under-constrained requests ("Show me the Clint Eastwood movie"). An approach is needed for handling and resolving such errors and ambiguities in a rapid, user-friendly, non-frustrating manner.

What is needed is a methodology and apparatus for rapidly constructing a voice-driven front-end atop an existing, non-voice data navigation system, whereby users can interact by means of intuitive natural language input not strictly conforming to the step-by-step browsing architecture of the existing navigation system, and wherein any errors or ambiguities in user input are rapidly and conveniently resolved. The solution to this need should be compatible with the constraints of a multi-user, distributed environment such as the Internet/Web or a proprietary high-bandwidth content delivery network; a solution contemplating one-at-a-time user interactions at a single location is insufficient, for example.

SUMMARY OF THE INVENTION

The present invention addresses the above needs by providing a system, method, and article of manufacture for using agents for navigation of network-based electronic data sources in response to spoken input requests. When a spoken input request is received from a user, it is interpreted, such as by using a speech recognition agent to extract speech data from acoustic voice signals, and using a language parsing agent to linguistically parse the speech data. The interpretation of the spoken request can be performed on a computing device locally with the user, such as the mobile information appliance, or remotely from the user. The resulting interpretation of the request is thereupon used to automatically construct an operational navigation query. The navigation query is routed to one or more agents that use the navigation query to retrieve the desired information from one or more electronic network data sources, which is then transmitted to a client device of the user. If the network data source is a database, the navigation query is constructed in the format of a database query language.

Typically, errors or ambiguities emerge in the interpretation of the spoken NL request, such that the system cannot instantiate a complete, valid navigational template. This is to be expected occasionally, and one preferred aspect of the invention is the ability to handle such errors and ambiguities in relatively graceful and user-friendly manner. Instead of simply rejecting such input and defaulting to traditional input modes or simply asking the user to try again, a preferred embodiment of the present invention seeks to converge rapidly toward instantiation of a valid navigational template by soliciting additional clarification from the user as necessary, either before or after a navigation of the data source, via multimodal input, i.e., by means of menu selection or other input modalities including and in addition to spoken natural language. This clarifying, multi-modal dialogue takes advantage of whatever partial navigational information has been gleaned from the initial interpretation of the user's spoken NL request. This clarification process continues until the system converges toward an adequately instantiated navigational template, which is in turn used to navigate the network-based data and retrieve the user's desired information. The retrieved information is transmitted across the network and presented to the user on a suitable client display device.

In a further aspect of the present invention, the construction of the navigation query includes extracting an input template for an online scripted interface to the data source and using the input template to construct the navigation query. The extraction of the input template can include dynamically scraping the online scripted interface.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention, together with further advantages thereof, may best be understood by reference to the following description taken in conjunction with the accompanying drawings in which:

FIG. **1***a* illustrates a system providing a spoken natural language interface for network-based information navigation, in accordance with an embodiment of the present invention with server-side processing of requests;

FIG. **1***b* illustrates another system providing a spoken natural language interface for network-based information navigation, in accordance with an embodiment of the present invention with client-side processing of requests;

FIG. **2** illustrates a system providing a spoken natural language interface for network-based information navigation, in accordance with an embodiment of the present invention for a mobile computing scenario;

FIG. **3** illustrates the functional logic components of a request processing module in accordance with an embodiment of the present invention;

FIG. **4** illustrates a process utilizing spoken natural language for navigating an electronic database in accordance with one embodiment of the present invention;

FIG. **5** illustrates a process for constructing a navigational query for accessing an online data source via an interactive, scripted (e.g., CGI) form; and

FIG. **6** illustrates an embodiment of the present invention utilizing a community of distributed, collaborating electronic agents.

DETAILED DESCRIPTION OF THE INVENTION

1. System Architecture

a. Server-End Processing of Spoken Input

FIG. **1***a* is an illustration of a data navigation system driven by spoken natural language input, in accordance with one embodiment of the present invention. As shown, a user's voice input data is captured by a voice input device **102**, such as a microphone. Preferably voice input device **102** includes a button or the like that can be pressed or held-down to activate a listening mode, so that the system need not continually pay attention to, or be confused by, irrelevant background noise. In one preferred embodiment well-suited for the home entertainment setting, voice input device **102** is a portable remote control device with an integrated microphone, and the voice data is transmitted from device **102** preferably via infrared (or other wireless) link to communications box **104** (e.g., a set-top box or a similar communications device that is capable of retransmitting the raw voice data and/or processing the voice data) local to the user's environment and coupled to communications network **106**. The voice data is then transmitted across network **106** to a remote server or servers **108**. The voice data may preferably be transmitted in compressed digitized form, or alternatively—particularly where bandwidth constraints are significant—in analog format (e.g., via frequency modulated transmission), in the latter case being digitized upon arrival at remote server **108**.

At remote server **108**, the voice data is processed by request processing logic **300** in order to understand the user's request and construct an appropriate query or request for navigation of remote data source **110**, in accordance with the interpretation process exemplified in FIG. **4** and FIG. **5** and discussed in greater detail below. For purposes of executing this process, request processing logic **300** comprises functional modules including speech recognition engine **310**, natural language (NL) parser **320**, query construction logic **330**, and query refinement logic **340**, as shown in FIG. **3**. Data source **110** may comprise database(s), Internet/web site(s), or other electronic information repositories, and preferably resides on a central server or servers—which may or may not be the same as server **108**, depending on the storage and bandwidth needs of the application and the resources available to the practitioner. Data source **110** may include multimedia content, such as movies or other digital video and audio content, other various forms of entertainment data, or other electronic information. The contents of data source **110** are navigated—i.e., the contents are accessed and searched, for retrieval of the particular information desired by the user—using the processes of FIGS. **4** and **5** as described in greater detail below.

Once the desired information has been retrieved from data source **110**, it is electronically transmitted via network **106** to the user for viewing on client display device **112**. In a preferred embodiment well-suited for the home entertainment setting, display device **112** is a television monitor or similar audiovisual entertainment device, typically in stationary position for comfortable viewing by users. In addition, in such preferred embodiment, display device **112** is coupled to or integrated with a communications box (which is preferably the same as communications box **104**, but may also be a separate unit) for receiving and decoding/formatting the desired electronic information that is received across communications network **106**.

Network **106** is a two-way electronic communications network and may be embodied in electronic communication infrastructure including coaxial (cable television) lines, DSL, fiber-optic cable, traditional copper wire (twisted pair), or any other type of hardwired connection. Network **106** may also include a wireless connection such as a satellite-based connection, cellular connection, or other type of wireless connection. Network **106** may be part of the Internet and may support TCP/IP communications, or may be embodied in a proprietary network, or in any other electronic communications network infrastructure, whether packet-switched or connection-oriented. A design consideration is that network **106** preferably provide suitable bandwidth depending upon the nature of the content anticipated for the desired application.

b. Client-End Processing of Spoken Input

FIG. **1***b* is an illustration of a data navigation system driven by spoken natural language input, in accordance with a second embodiment of the present invention. Again, a user's voice input data is captured by a voice input device

102, such as a microphone. In the embodiment shown in FIG. 1*b*, the voice data is transmitted from device **202** to requests processing logic **300**, hosted on a local speech processor, for processing and interpretation. In the preferred embodiment illustrated in FIG. 1*b*, the local speech processor is conveniently integrated as part of communications box **104**, although implementation in a physically separate (but communicatively coupled) unit is also possible as will be readily apparent to those of skill in the art. The voice data is processed by the components of request processing logic **300** in order to understand the user's request and construct an appropriate query or request for navigation of remote data source **110**, in accordance with the interpretation process exemplified in FIGS. **4** and **5** as discussed in greater detail below.

The resulting navigational query is then transmitted electronically across network **106** to data source **110**, which preferably resides on a central server or servers **108**. As in FIG. 1*a*, data source **110** may comprise database(s), Internet/web site(s), or other electronic information repositories, and preferably may include multimedia content, such as movies or other digital video and audio content, other various forms of entertainment data, or other electronic information. The contents of data source **110** are then navigated—i.e., the contents are accessed and searched, for retrieval of the particular information desired by the user—preferably using the process of FIGS. **4** and **5** as described in greater detail below. Once the desired information has been retrieved from data source **110**, it is electronically transmitted via network **106** to the user for viewing on client display device **112**.

In one embodiment in accordance with FIG. 1*b* and well-suited for the home entertainment setting, voice input device **102** is a portable remote control device with an integrated microphone, and the voice data is transmitted from device **102** preferably via infrared (or other wireless) link to the local speech processor. The local speech processor is coupled to communications network **106**, and also preferably to client display device **112** (especially for purposes of query refinement transmissions, as discussed below in connection with FIG. **4**, step **412**), and preferably may be integrated within or coupled to communications box **104**. In addition, especially for purposes of a home entertainment application, display device **112** is preferably a television monitor or similar audiovisual entertainment device, typically in stationary position for comfortable viewing by users. In addition, in such preferred embodiment, display device **112** is coupled to a communications box (which is preferably the same as communications box **104**, but may also be a physically separate unit) for receiving and decoding/formatting the desired electronic information that is received across communications network **106**.

Design considerations favoring server-side processing and interpretation of spoken input requests, as exemplified in FIG. 1*a*, include minimizing the need to distribute costly computational hardware and software to all client users in order to perform speech and language processing. Design considerations favoring client-side processing, as exemplified in FIG. 1*b*, include minimizing the quantity of data sent upstream across the network from each client, as the speech recognition is performed before transmission across the network and only the query data and/or request needs to be sent, thus reducing the upstream bandwidth requirements.

c. Mobile Client Embodiment

A mobile computing embodiment of the present invention may be implemented by practitioners as a variation on the embodiments of either FIG. 1*a* or FIG. 1*b*. For example, as depicted in FIG. **2**, a mobile variation in accordance with the server-side processing architecture illustrated in FIG. 1*a* may be implemented by replacing voice input device **102**, communications box **104**, and client display device **112**, with an integrated, mobile, information appliance **202** such as a cellular telephone or wireless personal digital assistant (wireless PDA). Mobile information appliance **202** essentially performs the functions of the replaced components. Thus, mobile information appliance **202** receives spoken natural language input requests from the user in the form of voice data, and transmits that data (preferably via wireless data receiving station **204**) across communications network **206** for server-side interpretation of the request, in similar fashion as described above in connection with FIG. **1**. Navigation of data source **210** and retrieval of desired information likewise proceeds in an analogous manner as described above. Display information transmitted electronically back to the user across network **206** is displayed for the user on the display of information appliance **202**, and audio information is output through the appliance's speakers.

Practitioners will further appreciate, in light of the above teachings, that if mobile information appliance **202** is equipped with sufficient computational processing power, then a mobile variation of the client-side architecture exemplified in FIG. **2** may similarly be implemented. In that case, the modules corresponding to request processing logic **300** would be embodied locally in the computational resources of mobile information appliance **202**, and the logical flow of data would otherwise follow in a manner analogous to that previously described in connection with FIG. 1*b*.

As illustrated in FIG. **2**, multiple users, each having their own client input device, may issue requests, simultaneously or otherwise, for navigation of data source **210**. This is equally true (though not explicitly drawn) for the embodiments depicted in FIGS. 1*a* and 1*b*. Data source **210** (or **100**), being a network accessible information resource, has typically already been constructed to support access requests from simultaneous multiple network users, as known by practitioners of ordinary skill in the art. In the case of server-side speech processing, as exemplified in FIGS. 1*a* and **2**, the interpretation logic and error correction logic modules are also preferably designed and implemented to support queuing and multi-tasking of requests from multiple simultaneous network users, as will be appreciated by those of skill in the art.

It will be apparent to those skilled in the art that additional implementations, permutations and combinations of the embodiments set forth in FIGS. 1*a*, 1*b*, and **2** may be created without straying from the scope and spirit of the present invention. For example, practitioners will understand, in light of the above teachings and design considerations, that it is possible to divide and allocate the functional components of request processing logic **300** between client and server. For example, speech recognition—in entirety, or perhaps just early stages such as feature extraction—might be performed locally on the client end, perhaps to reduce bandwidth requirements, while natural language parsing and other necessary processing might be performed upstream on the server end, so that more extensive computational power need not be distributed locally to each client. In that case, corresponding portions of request processing logic **300**, such as speech recognition engine **310** or portions thereof, would reside locally at the client as in FIG. 1*b*, while other component modules would be hosted at the server end as in FIGS. 1*a* and **2**.

Further, practitioners may choose to implement the each of the various embodiments described above on any number of different hardware and software computing platforms and

environments and various combinations thereof, including, by way of just a few examples: a general-purpose hardware microprocessor such as the Intel Pentium series; operating system software such as Microsoft Windows/CE, Palm OS, or Apple Mac OS (particularly for client devices and client-side processing), or Unix, Linux, or Windows/NT (the latter three particularly for network data servers and server-side processing), and/or proprietary information access platforms such as Microsoft's WebTV or the Diva Systems video-on-demand system.

2. Processing Methodology

The present invention provides a spoken natural language interface for interrogation of remote electronic databases and retrieval of desired information. A preferred embodiment of the present invention utilizes the basic methodology outlined in the flow diagram of FIG. **4** in order to provide this interface. This methodology will now be discussed.

a. Interpreting Spoken Natural Language Requests

At step **402**, the user's spoken request for information is initially received in the form of raw (acoustic) voice data by a suitable input device, as previously discussed in connection with FIGS. **1–2**. At step **404** the voice data received from the user is interpreted in order to understand the user's request for information. Preferably this step includes performing speech recognition in order to extract words from the voice data, and further includes natural language parsing of those words in order to generate a structured linguistic representation of the user's request.

Speech recognition in step **404** is performed using speech recognition engine **310**. A variety of commercial quality, speech recognition engines are readily available on the market, as practitioners will know. For example, Nuance Communications offers a suite of speech recognition engines, including Nuance **6**, its current flagship product, and Nuance Express, a lower cost package for entry-level applications. As one other example, IBM offers the ViaVoice speech recognition engine, including a low-cost shrink-wrapped version available through popular consumer distribution channels. Basically, a speech recognition engine processes acoustic voice data and attempts to generate a text stream of recognized words.

Typically, the speech recognition engine is provided with a vocabulary lexicon of likely words or phrases that the recognition engine can match against its analysis of acoustical signals, for purposes of a given application. Preferably, the lexicon is dynamically adjusted to reflect the current user context, as established by the preceding user inputs. For example, if a user is engaged in a dialogue with the system about movie selection, the recognition engine's vocabulary may preferably be adjusted to favor relevant words and phrases, such as a stored list of proper names for popular movie actors and directors, etc. Whereas if the current dialogue involves selection and viewing of a sports event, the engine's vocabulary might preferably be adjusted to favor a stored list of proper names for professional sports teams, etc. In addition, a speech recognition engine is provided with language models that help the engine predict the most likely interpretation of a given segment of acoustical voice data, in the current context of phonemes or words in which the segment appears. In addition, speech recognition engines often echo to the user, in more or less real-time, a transcription of the engine's best guess at what the user has said, giving the user an opportunity to confirm or reject.

In a further aspect of step **404**, natural language interpreter (or parser) **320** linguistically parses and interprets the textual output of the speech recognition engine. In a preferred embodiment of the present invention, the natural-language interpreter attempts to determine both the meaning of spoken words (semantic processing) as well as the grammar of the statement (syntactic processing), such as the Gemini Natural Language Understanding System developed by SRI International. The Gemini system is described in detail in publications entitled "Gemini: A Natural Language System for Spoken-Language Understanding" and "Interleaving Syntax and Semantics in an Efficient Bottom-Up Parser," both of which are currently available online at http://www.ai.sri.com/natural-language/projects/arpa-sls/nat-lang.html. (Copies of those publications are also included in an information disclosure statement submitted herewith, and are incorporated herein by this reference). Briefly, Gemini applies a set of syntactic and semantic grammar rules to a word string using a bottom-up parser to generate a logical form, which is a structured representation of the context-independent meaning of the string. Gemini can be used with a variety of grammars, including general English grammar as well as application-specific grammars. The Gemini parser is based on "unification grammar," meaning that grammatical categories incorporate features that can be assigned values; so that when grammatical category expressions are matched in the course of parsing or semantic interpretation, the information contained in the features is combined, and if the feature values are incompatible the match fails.

It is possible for some applications to achieve a significant reduction in speech recognition error by using the natural-language processing system to re-score recognition hypotheses. For example, the grammars defined for a language parser like Gemini may be compiled into context-free grammar that, in turn, can be used directly as language models for speech recognition engines like the Nuance recognizer. Further details on this methodology are provided in the publication "Combining Linguistic and Statistical Knowledge Sources in Natural-Language Processing for ATIS" which is currently available online through http://www.ai.sri.com/natural-language/projects/arpa-sls/spnl-int.html. A copy of this publication is included in an information disclosure submitted herewith, and is incorporated herein by this reference.

In an embodiment of the present invention that may be preferable for some applications, the natural language interpreter "learns" from the past usage patterns of a particular user or of groups of users. In such an embodiment, the successfully interpreted requests of users are stored, and can then be used to enhance accuracy by comparing a current request to the stored requests, thereby allowing selection of a most probable result.

b. Constructing Navigation Queries

In step **405** request processing logic **300** identifies and selects an appropriate online data source where the desired information (in this case, current weather reports for a given city) can be found. Such selection may involve look-up in a locally stored table, or possibly dynamic searching through an online search engine, or other online search techniques. For some applications, an embodiment of the present invention may be implemented in which only access to a particular data source (such as a particular vendor's proprietary content database) is supported; in that case, step **405** may be trivial or may be eliminated entirely.

Step **406** attempts to construct a navigation query, reflecting the interpretation of step **404**. This operation is preferably performed by query construction logic **330**.

A "navigation query" means an electronic query, form, series of menu selections, or the like; being structured appropriately so as to navigate a particular data source of

interest in search of desired information. In other words, a navigation query is constructed such that it includes whatever content and structure is required in order to access desired information electronically from a particular database or data source of interest.

For example, for many existing electronic databases, a navigation query can be embodied using a formal database query language such as Standard Query Language (SQL). For many databases, a navigation query can be constructed through a more user-friendly interactive front-end, such as a series of menus and/or interactive forms to be selected or filled in. SQL is a standard interactive and programming language for getting information from and updating a database. SQL is both an ANSI and an ISO standard. As is well known to practitioners, a Relational Database Management System (RDBMS), such as Microsoft's Access, Oracle's Oracle7, and Computer Associates' CA-OpenIngres, allow programmers to create, update, and administer a relational database. Practitioners of ordinary skill in the art will be thoroughly familiar with the notion of database navigation through structured query, and will be readily able to appreciate and utilize the existing data structures and navigational mechanisms for a given database, or to create such structures and mechanisms where desired.

In accordance with the present invention, the query constructed in step **406** must reflect the user's request as interpreted by the speech recognition engine and the NL parser in step **404**. In embodiments of the present invention wherein data source **110** (or **210** in the corresponding embodiment of FIG. **2**) is a structured relational database or the like, step **406** of the present invention may entail constructing an appropriate Structured Query Language (SQL) query or the like, or automatically filling out a front-end query form, series of menus or the like, as described above.

In many existing Internet (and Intranet) applications, an online electronic data source is accessible to users only through the medium of interaction with a so-called Common Gateway Interface (CGI) script. Typically the user who visits a web site of this nature must fill in the fields of an online interactive form. The online form is in turn linked to a CGI script, which transparently handles actual navigation of the associated data source and produces output for viewing by the user's web browser. In other words, direct user access to the data source is not supported, only mediated access through the form and CGI script is offered.

For applications of this nature, an advantageous embodiment of the present invention "scrapes" the scripted online site where information desired by a user may be found in order to facilitate construction of an effective navigation query. For example, suppose that a user's spoken natural language request is: "What's the weather in Miami?" After this request is received at step **402** and interpreted at step **404**, assume that step **405** determines that the desired weather information is available online through the medium of a CGI-scripted interactive form. Step **406** is then preferably carried out using the expanded process diagrammed in FIG. **5**. In particular, at sub-step **520**, query construction logic **330** electronically "scrapes" the online interactive form, meaning that query construction logic **330** automatically extracts the format and structure of input fields accepted by the online form. At sub-step **522**, a navigation query is then constructed by instantiating (filling in) the extracted input format—essentially an electronic template—in a manner reflecting the user's request for information as interpreted in step **404**. The flow of control then returns to step **407** of FIG. **4**. Ultimately, when the query thus constructed by scraping is used to navigate the online data source in step **408**, the query effectively initiates the same scripted response as if a human user had visited the online site and had typed appropriate entries into the input fields of the online form.

In the embodiment just described, scraping step **520** is preferably carried out with the assistance of an online extraction utility such as WebL. WebL is a scripting language for automating tasks on the World Wide Web. It is an imperative, interpreted language that has built-in support for common web protocols like HTTP and FTP, and popular data types like HTML and XML. WebL's implementation language is Java, and the complete source code is available from Compaq. In addition, step **520** is preferably performed dynamically when necessary—in other words, on-the-fly in response to a particular user query—but in some applications it may be possible to scrape relatively stable (unchanging) web sites of likely interest in advance and to cache the resulting template information.

It will be apparent, in light of the above teachings, that preferred embodiments of the present invention can provide a spoken natural language interface atop an existing, non-voice data navigation system, whereby users can interact by means of intuitive natural language input not strictly conforming to the linear browsing architecture or other artifacts of an existing menu/text/click navigation system. For example, users of an appropriate embodiment of the present invention for a video-on-demand application can directly speak the natural request: "Show me the movie 'Unforgiven'"—instead of walking step-by-step through a typically linear sequence of genre/title/actor/director menus, scrolling and selecting from potentially long lists on each menu, or instead of being forced to use an alphanumeric keyboard that cannot be as comfortable to hold or use as a lightweight remote control. Similarly, users of an appropriate embodiment of the present invention for a web-surfing application in accordance with the process shown in FIG. **5** can directly speak the natural request: "Show me a one-month price chart for Microsoft stock"—instead of potentially having to navigate to an appropriate web site, search for the right ticker symbol, enter/select the symbol, and specify display of the desired one-month price chart, each of those steps potentially involving manual navigation and data entry to one or more different interaction screens. (Note that these examples are offered to illustrate some of the potential benefits offered by appropriate embodiments of the present invention, and not to limit the scope of the invention in any respect.)

c. Error Correction

Several problems can arise when attempting to perform searches based on spoken natural language input. As indicated at decision step **407** in the process of FIG. **4**, certain deficiencies may be identified during the process of query construction, before search of the data source is even attempted. For example, the user's request may fail to specify enough information in order to construct a navigation query that is specific enough to obtain a satisfactory search result. For example, a user might orally request "what's the weather? " whereas the national online data source identified in step **405** and scraped in step **520** might require specifying a particular city.

Additionally, certain deficiencies and problems may arise following the navigational search of the data source at step **408**, as indicated at decision step **409** in FIG. **4**. For example, with reference to a video-on-demand application, a user may wish to see the movie "Unforgiven",but perhaps the user can't recall name of the film, but knows it was

directed by and starred actor Clint Eastwood. A typical video-on-demand database might indeed be expected to allow queries specifying the name of a leading actor and/or director, but in the case of this query—as in many cases—that will not be enough to narrow the search to a single film, and additional user input in some form is required.

In the event that one or more deficiencies in the user's spoken request, as processed, result in the problems described, either at step **407** or **409**, some form of error handling is in order. A straightforward, crude technique might be for the system to respond simply "input not understood/insufficient, please try again." However, that approach will likely result in frustrated users, and is not optimal or even acceptable for most applications. Instead, a preferred technique in accordance with the present invention handles such errors and deficiencies in user input at step **412**, whether detected at step **407** or step **409**, by soliciting additional input from the user in a manner taking advantage of the partial construction already performed and via user interface modalities in addition to spoken natural language ("multi-modality"). This supplemental interaction is preferably conducted through client display device **112** (**202**, in the embodiment of FIG. **2**), and may include textual, graphical, audio and/or video media. Further details and examples are provided below. Query refinement logic **340** preferably carries out step **412**. The additional input received from the user is fed into and augments interpreting step **404**, and query construction step **406** is likewise repeated with the benefit of the augmented interpretation. These operations, and subsequent navigation step **408**, are preferably repeated until no remaining problems or deficiencies are identified at decision points **407** or **409**. Further details and examples for this query refinement process are provided immediately below.

Consider again the example in which the user of a video-on-demand application wishes to see "Unforgiven" but can only recall that it was directed by and starred Clint Eastwood. First, it bears noting that using a prior art navigational interface, such as a conventional menu interface, will likely be relatively tedious in this case. The user can proceed through a sequence of menus, such as Genre (select "western"), Title (skip), Actor ("Clint Eastwood"), and Director ("Clint Eastwood"). In each case—especially for the last two items—the user would typically scroll and select from fairly long lists in order to enter his or her desired name, or perhaps use a relatively couch-unfriendly keypad to manually type the actor's name twice.

Using a preferred embodiment of the present invention, the user instead speaks aloud, holding remote control microphone **102**, "I want to see that movie starring and directed by Clint Eastwood. Can't remember the title." At step **402** the voice data is received. At step **404** the voice data is interpreted. At step **405** an appropriate online data source is selected (or perhaps the system is directly connected to a proprietary video-on-demand provider). At step **406** a query is automatically constructed by the query construction logic **330** specifying "Clint Eastwood" in both the actor and director fields. Step **407** detects no obvious problems, and so the query is electronically submitted and the data source is navigated at step **408**, yielding a list of several records satisfying the query (e.g., "Unforgiven","True Crime", "Absolute Power",etc.). Step **409** detects that additional user input is needed to further refine the query in order to select a particular film for viewing.

At that point, in step **412** query refinement logic **340** might preferably generate a display for client display device **112** showing the (relatively short) list of film titles that satisfy the user's stated constraints. The user can then preferably use a relatively convenient input modality, such as buttons on the remote control, to select the desired title from the menu. In a further preferred embodiment, the first title on the list is highlighted by default, so that the user can simply press an "OK" button to choose that selection. In a further preferred feature, the user can mix input modalities by speaking a response like "I want number one on the list." Alternatively, the user can preferably say, "Let's see Unforgiven," having now been reminded of the title by the menu display.

Utilizing the user's supplemental input, request processing logic **300** iterates again through steps **404** and **406**, this time constructing a fully-specified query that specifically requests the Eastwood film "Unforgiven." Step **408** navigates the data source using that query and retrieves the desired film, which is then electronically transmitted in step **410** from network server **108** to client display device **112** via communications network **106**.

Now consider again the example in which the user of a web surfing application wants to know his or her local weather, and simply asks, "what's the weather?" At step **402** the voice data is received. At step **404** the voice data is interpreted. At step **405** an online web site providing current weather information for major cities around the world is selected. At step **406** and sub-step **520**, the online site is scraped using a WebL-style tool to extract an input template for interacting with the site. At sub-step **522**, query construction logic **330** attempts to construct a navigation query by instantiating the input template, but determines (quite rightly) that a required field—name of city—cannot be determined from the user's spoken request as interpreted in step **404**. Step **407** detects this deficiency, and in step **412** query refinement logic **340** preferably generates output for client display device **112** soliciting the necessary supplemental input. In a preferred embodiment, the output might display the name of the city where the user is located highlighted by default. The user can then simply press an "OK" button—or perhaps mix modalities by saying "yes, exactly"—to choose that selection. A preferred embodiment would further display an alphabetical scrollable menu listing other major cities, and/or invite the user to speak or select the name of the desired city.

Here again, utilizing the user's supplemental input, request processing logic **300** iterates through steps **404** and **406**. This time, in performing sub-step **520**, a cached version of the input template already scraped in the previous iteration might preferably be retrieved. In sub-step **522**, query construction logic **330** succeeds this time in instantiating the input template and constructing an effective query, since the desired city has now been clarified. Step **408** navigates the data source using that query and retrieves the desired weather information, which is then electronically transmitted in step **410** from network server **108** to client display device **112** via communications network **106**.

It is worth noting that in some instances, there may be details that are not explicitly provided by the user, but that query construction logic **330** or query refinement logic **340** may preferably deduce on their own through reasonable assumptions, rather than requiring the use to provide explicit clarification. For example, in the example previously described regarding a request for a weather report, in some applications it might be preferable for the system to simply assume that the user means a weather report for his or her home area and to retrieve that information, if the cost of doing so is not significantly greater than the cost of asking the user to clarify the query. Making such an assumption

might be even more strongly justified in a preferred embodiment, as described earlier, where user histories are tracked, and where such history indicates that a particular user or group of users typically expect local information when asking for a weather forecast. At any rate, in the event such an assumption is made, if the user actually intended to request the weather for a different city, the user would then need to ask his or her question again. It will be apparent to practitioners, in light of the above teachings, that the choice of whether to program query construction logic **330** and query refinement logic **340** to make make particular assumptions will typically involve trade-offs involving user conveience that can be assessed in the context of specific applications.

3. Open Agent Architecture™ (OAA®)

Open Agent Architecture™(OAA®) is a software platform, developed by the assignee of the present invention, that enables effective, dynamic collaboration among communities of distributed electronic agents. OAA is described in greater detail in co-pending U.S. patent application Ser. No. 09/225,198, which has been incorporated herein by reference. Very briefly, the functionality of each client agent is made available to the agent community through registration of the client agent's capabilities with a facilitator. A software "wrapper" essentially surrounds the underlying application program performing the services offered by each client. The common infrastructure for constructing agents is preferably supplied by an agent library. The agent library is preferably accessible in the runtime environment of several different programming languages. The agent library preferably minimizes the effort required to construct a new system and maximizes the ease with which legacy systems can be "wrapped" and made compatible with the agent-based architecture of the present invention. When invoked, a client agent makes a connection to a facilitator, which is known as its parent facilitator. Upon connection, an agent registers with its parent facilitator a specification of the capabilities and services it can provide, using a highlevel, declarative Interagent Communication Language ("ICL") to express those capabilities. Tasks are presented to the facilitator in the form of ICL goal expressions. When a facilitator determines that the registered capabilities of one of its client agents will help satisfy a current goal or sub-goal thereof, the facilitator delegates that sub-goal to the client agent in the form of an ICL request. The client agent processes the request and returns answers or information to the facilitator. In processing a request, the client agent can use ICL to request services of other agents, or utilize other infrastructure services for collaborative work. The facilitator coordinates and integrates the results received from different client agents on various sub-goals, in order to satisfy the overall goal.

OAA provides a useful software platform for building systems that integrate spoken natural language as well as other user input modalities. For example, see the above-referenced co-pending patent application, especially FIG. **13** and the corresponding discussion of a "multi-modal maps" application, and FIG. **12** and the corresponding discussion of a "unified messaging" application. Another example is the InfoWiz interactive information kiosk developed by the assignee and described in the document entitled "InfoWiz: An Animated Voice Interactive Information System" available online at http://www.ai.sri.com/~oaa/applications.html. A copy of the InfoWhiz document is provided in an Information Disclosure Statement submitted herewith and incorporated herein by this reference. A further example is the "CommandTalk" application developed by the assignee for the U.S. military, as described online at http://www.ai.sri.com/~lesaf/commandtalk.html and in the following publications, copies of which are provided in an Information Disclosure Statement submitted herewith and incorporated herein by this reference:

"CommandTalk: A Spoken-Language Interface for Battlefield Simulations", 1997, by Robert Moore, John Dowding, Harry Bratt, J. Mark Gawron, Yonael Gorfu and Adam Cheyer, in "Proceedings of the Fifth Conference on Applied Natural Language Processing", Washington, D.C., pp. 1–7, Association for Computational Linguistics

"The CommandTalk Spoken Dialogue System", 1999, by Amanda Stent, John Dowding, Jean Mark Gawron, Elizabeth Owen Bratt and Robert Moore, in "Proceedings of the Thirty-Seventh Annual Meeting of the ACL", pp. 183–190, University of Maryland, College Park, Md., Association for Computational Linguistics

"Interpreting Language in Context in CommandTalk", 1999, by John Dowding and Elizabeth Owen Bratt and Sharon Goldwater, in "Communicative Agents: The Use of Natural Language in Embodied Systems", pp. 63–67, Association for Computing Machinery (ACM) Special Interest Group on Artificial Intelligence (SIGART), Seattle, Wash.

For some applications and systems, OAA can provide an advantageous platform for constructing embodiments of the present invention. For example, a representative application is now briefly presented, with reference to FIG. **6**. If the statement "show me movies starring John Wayne" is spoken into the voice input device, the voice data for this request will be sent by UI agent **650** to facilitator **600**, which in turn will ask natural language (NL) agent **620** and speech recognition agent **610** to interpret the query and return the interpretation in ICL format. The resulting ICL goal expression is then routed by the facilitator to appropriate agents—in this case, video-on-demand database agent **640**—to execute the request. Video database agent **640** preferably includes or is coupled to an appropriate embodiment of query construction logic **330** and query refinement logic **340**, and may also issue ICL requests to facilitator **600** for additional assistance—e.g., display of menus and capture of additional user input in the event that query refinement is needed—and facilitator **600** will delegate such requests to appropriate client agents in the community. When the desired video content is ultimately retrieved by video database agent **640**, UI agent **650** is invoked by facilitator **600** to display the movie.

Other spoken user requests, such as a request for the current weather in New York City or for a stock quote, would eventually lead facilitator to invoke web database agent **630** to access the desired information from an appropriate Internet site. Here again, web database agent **630** preferably includes or is coupled to an appropriate embodiment of query construction logic **330** and query refinement logic **340**, including a scraping utility such as WebL. Other spoken requests, such as a request to view recent emails or access voice mail, would lead the facilitator to invoke the appropriate email agent **660** and/or telephone agent **680**. A request to record a televised program of interest might lead facilitator **600** to invoke web database agent **630** to return televised program schedule information, and then invoke VCR controller agent **680** to program the associated VCR unit to record the desired television program at the scheduled time.

Control and connectivity embracing additional electronic home appliances (e.g., microwave oven, home surveillance system, etc.) can be integrated in comparable fashion.

Indeed, an advantage of OAA-based embodiments of the present invention, that will be apparent to practitioners in light of the above teachings and in light of the teachings disclosed in the cited co-pending patent applications, is the relative ease and flexibility with which additional service agents can be plugged into the existing platform, immediately enabling the facilitator to respond dynamically to spoken natural language requests for the corresponding services.

4. Further Embodiments and Equivalents

While the present invention has been described in terms of several preferred embodiments, there are many alterations, permutations, and equivalents that may fall within the scope of this invention. It should also be noted that there are many alternative ways of implementing the methods and apparatuses of the present invention. It is therefore intended that the following appended claims be interpreted as including all such alterations, permutations, and equivalents as fall within the true spirit and scope of the present invention.

What is claimed is:

1. A method for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:

(a) receiving a spoken request for desired information from a user;

(b) rendering an interpretation of the spoken request;

(c) constructing a navigation query based upon the interpretation;

(d) routing the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source; and

(e) invoking a user interface agent for outputting the selected portion of the electronic data source to the user, wherein a facilitator manages data flow among multiple agents and maintains a registration of each of said agents' capabilities.

2. The method of claim **1**, wherein an agent renders the interpretation of the spoken request.

3. The method of claim **1**, wherein the step of rendering the interpretation of the spoken request is performed by a speech recognition agent and a parsing agent.

4. The method of claim **1**, further comprising the steps of soliciting additional input from the user, including user interaction in a modality different than the original request; and refining the navigation query, based upon the additional input; wherein the at least one agent uses the refined navigation query to select a portion of the electronic data source.

5. The method of claim **4**, wherein agents are utilized for performing the steps of soliciting additional input from the user and refining the navigation query.

6. The method of claim **1**, wherein the electronic data source is a web page, wherein the at least one agent scrapes the web page for selecting a portion of the web page.

7. A computer program embodied on a computer readable medium for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:

(a) a code segment that receives a spoken request for desired information from a user;

(b) a code segment that renders an interpretation of the spoken request;

(c) a code segment that constructs a navigation query based upon the interpretation;

(d) a code segment that routes the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source; and

(e) a code segment that invokes a user interface agent for outputting the selected portion of the electronic data source to the user, wherein a facilitator manages data flow among multiple agents and maintains a registration of each of said agents' capabilities.

8. The computer program of claim **7**, wherein the code segment that renders the interpretation of the spoken request is executed by an agent.

9. The computer program of claim **7**, wherein a speech recognition agent and a parsing agent execute the code segment that renders the interpretation of the spoken request.

10. The computer program of claim **7**, further comprising a code segment that solicits additional input from the user, including user interaction in a modality different than the original request; and a code segment that refines the navigation query, based upon the additional input; wherein the at least one agent uses the refined navigation query to select a portion of the electronic data source.

11. The computer program of claim **10**, wherein a solicitor agent executes the code segment that solicit the additional input from the user and a refining agent executes the code segment that refines the navigation query.

12. The computer program of claim **7**, wherein the electronic data source is a web page, wherein the at least one agent scrapes the web page for selecting a portion of the web page.

13. A system for utilizing agents for speech-based navigation of an electronic data source, comprising the steps of:

(a) a client device, operable to receive a spoken request for desired information from a user; (b) spoken language processing logic, operable to render an interpretation of the spoken request;

(c) query construction logic, operable to construct a navigation query based upon the interpretation;

(d) routing logic, operable to route the navigation query to at least one agent, wherein the at least one agent utilizes the navigation query to select a portion of the electronic data source; and

(e) invoking logic, operable to invoke a user interface agent for outputting the selected portion of the electronic data source to the user, Wherein a facilitator manages data flow among multiple agents and maintains a registration of each of said agents' capabilities.

14. The system of claim **13**, wherein the query construction logic that renders the interpretation of the spoken request is executed by an agent.

15. The system of claim **13**, wherein a speech recognition agent and a parsing agent execute the spoken language processing logic that renders the interpretation of the spoken request.

16. The system of claim **13**, further comprising user interaction logic operable to solicit additional input from the user, including user interaction in a modality different than the original request; and query refining logic operable to refine the navigation query, based upon the additional input; wherein the at least one agent uses the refined navigation query to select a portion of the electronic data source.

17. The system of claim **16**, wherein a solicitor agent executes the user interaction logic and a refining agent executes the query refinement logic.

18. The system of in claim **13**, wherein the electronic data source is a web page, wherein the at least one agent scrapes the web page for selecting a portion of the web page.

* * * * *

US006851115B1

(12) **United States Patent**
Cheyer et al.

(10) **Patent No.: US 6,851,115 B1**
(45) **Date of Patent: Feb. 1, 2005**

(54) **SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS**

(75) Inventors: **Adam J. Cheyer**, Palo Alto, CA (US); **David L. Martin**, Santa Clara, CA (US)

(73) Assignee: **SRI International**, Menlo Park, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/225,198**

(22) Filed: **Jan. 5, 1999**

(51) **Int. Cl.**^{7} **G06F 9/54**
(52) **U.S. Cl.** **719/317**; 709/202; 717/114
(58) **Field of Search** 719/317; 709/202, 709/207, 317; 717/114

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,638,494	A	*	6/1997	Pinard et al.	709/202
5,802,396	A	*	9/1998	Gray	710/20
5,960,404	A	*	9/1999	Chaar et al.	705/11
6,047,053	A		4/2000	Miner et al.	
6,212,649	B1	*	4/2001	Yalowitz et al.	714/31
6,216,173	B1	*	4/2001	Jones et al.	135/77
6,256,771	B1		7/2001	O'Neil et al.	
6,338,081	B1	*	1/2002	Furusawa et al.	709/202
6,411,684	B1		6/2002	Cohn et al.	
6,484,155	B1	*	11/2002	Kiss et al.	706/46
2001/0039562	A1	*	11/2001	Sato	709/202
2003/0167247	A1	*	9/2003	Masuoka	706/46

OTHER PUBLICATIONS

Cheyer, Adam. "Mechanisms of Cooperation." Oct. 19, 1998.*

DeVoe, Deborah. "SRI distributed agents promise flexibility." InfoWorld. Dec. 30, 1996.*

Sycara, Katia et al. "Distributed Intelligent Agents." IEEE. Dec. 1996.*

Nwana, Hyacinth et al. "Software Agent Technologies". BT Technology Journal. 1996.*

Busetta, Paolo et al. "The BDIM Agent Toolkit Design." 1997.*

Mayfield, James et al. "Desiderata for Agent Communication Languages." Mar. 27–29, 1995.*

Khedro, Taha et al. Concurrent Endineering through Interoperable Software Agents. Aug. 1994.*

Moran et al. "Multimodal User Interfaces in the Open Agent Architecture." Proceedings of the International Conference on Intelligent User Interfaces. 6–9/1997.*

Martin, David et al. "The Open Agent Architecture: A Framework for Building Distributed Software Systems." Oct. 19, 1998.*

Wilkins, David et al. "Multiagent Planning Architecture." SRI International. Dec. 8, 1997.*

Moran, Douglas B. and Cheyer, Adam J., "Intelligent Agent–based User Interfaces", Article Intelligence center, SRI International.

Martin, David L., Cheyer, Adam J. and Moran, Douglas B., "Building Distributed Software Systems with the Open Agent Architecture".

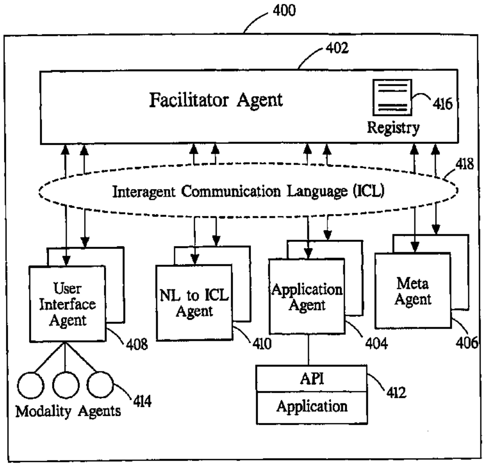
(List continued on next page.)

Primary Examiner—Lewis A. Bullock, Jr.
(74) *Attorney, Agent, or Firm*—Perkins Coie LLP

(57) **ABSTRACT**

A highly flexible, software-based architecture is disclosed for constructing distributed systems. The architecture supports cooperative task completion by flexible and autonomous electronic agents. One or more facilitators are used to broker communication and cooperation among the agents. The architecture provides for the construction of arbitrarily complex goals by users and service-requesting agents. Additional features include agent-based provision of multi modal interfaces, including natural language.

89 Claims, 16 Drawing Sheets



OTHER PUBLICATIONS

Cohen, Philip R. and Cheyer, Adam, SRI International, Wang, Michelle, Stanford University, BAEG, Soon Cheol, ETRI, "An Open Agent Architecture".

Julia, Luc E. and Cheyer, Adam J., SRI International "Cooperative Agents and Recognition Systems (CARS) for Drivers and Passengers".

Moran, Douglas B., Cheyer, Adam J., Julia, Luc E., Martin, David L., SRI International, and Park, Sangkyu, Electronics and Telecommunications Research Institute, "Multimodal User Interfaces in the Open Agent Architecture".

Cheyer, Adam and Lulia, Luc, SRI International "Multimodal Maps: An Agent–based Approach".

Cutkosky, Mark R., Engelmore, Robert S., Fikes, Richard E., Genesereth, Michael R., Gruber, Thomas R., Stanford University, Mark, William, Lockheed Palo Alto Research Labs, Tenenbaum, Jay M., Weber, Jay C., Enterprise Integration Technologies, "An Experiment in Integrating Concurrent Engineering Systems".

Martin, David L., Cheyer, Adam, SRI International, LEE, Gowang–Lo, ETRI, "Development Tools for the Open Agent Architecture", The Practical Application of Intelligent Agents and Multi–Agent Technology (PAAM96), London, Apr. 1996.

Cheyer, Adam, Martin, David and Moran, Douglas, "The Open Agent architecture™", SRI International, AI Center.

Dejima, Inc., http://www.dejima.com/.

Cohen, Philip R, Cheyer, Adam, Wang, Michelle, Stanford University, BAEG, Soon Cheol ETRI: "An Open Agent Architecture," AAAI Spring Symposium, pp. 1–8, Mar. 1994.

Martin, David; Oohama, Hiroki; Moran, Douglas; Cheyer, Adam; "Information Brokering in an Agent Architecture," Proceeding of the 2nd International Conference on Practical Application of Intelligent Agents & Multi–Agent Technology, London, Apr. 1997.

* cited by examiner

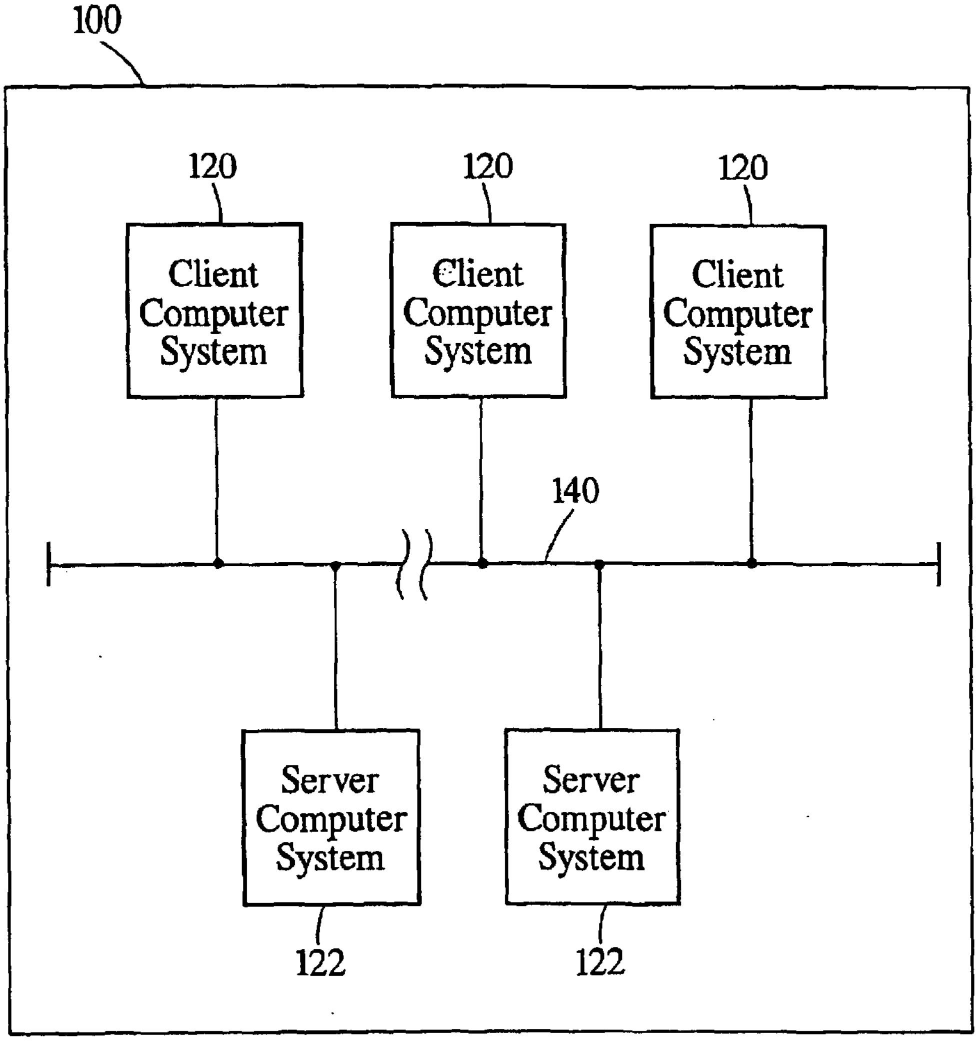


Fig. 1
(Prior Art)

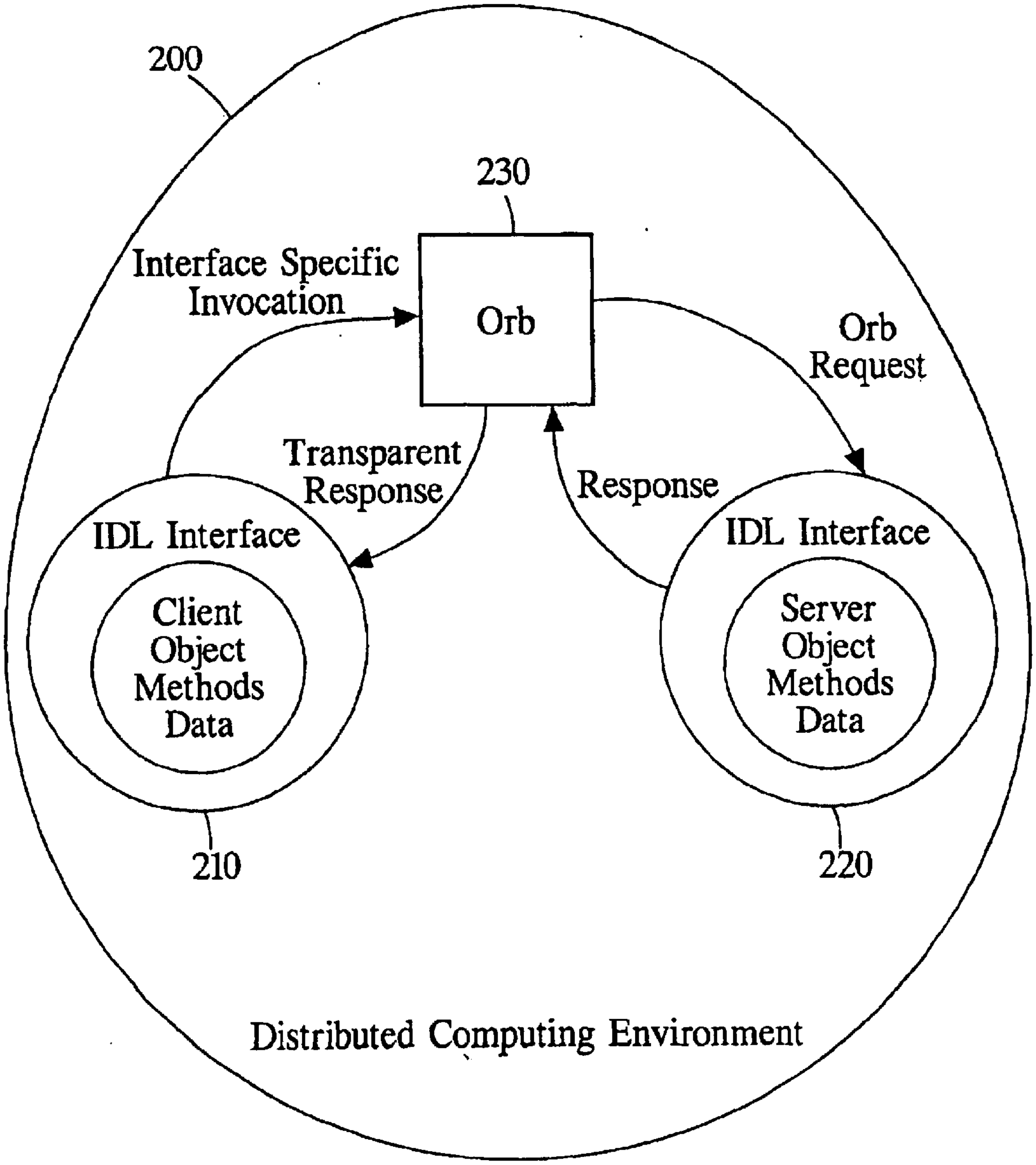

200
230
Interface Specific Invocation
Orb
Orb Request
Transparent Response
Response
IDL Interface
Client Object Methods Data
IDL Interface
Server Object Methods Data
210
220
Distributed Computing Environment

Fig. 2
(Prior Art)

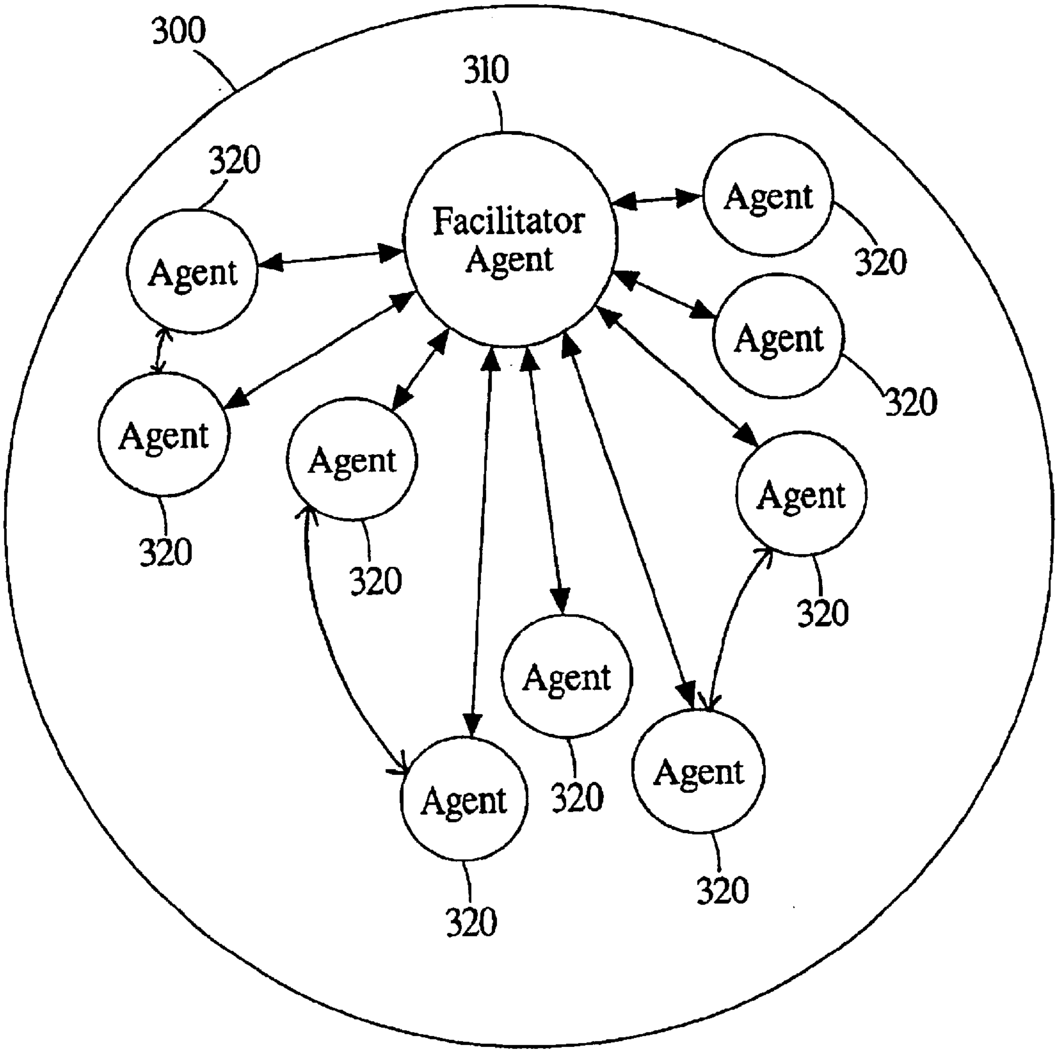

300
310
Facilitator Agent
Agent
320
Agent
320
Agent
320
Agent
320
Agent
320
Agent
320
Agent
320
Agent
320
Agent
320
Agent
320

Fig. 3

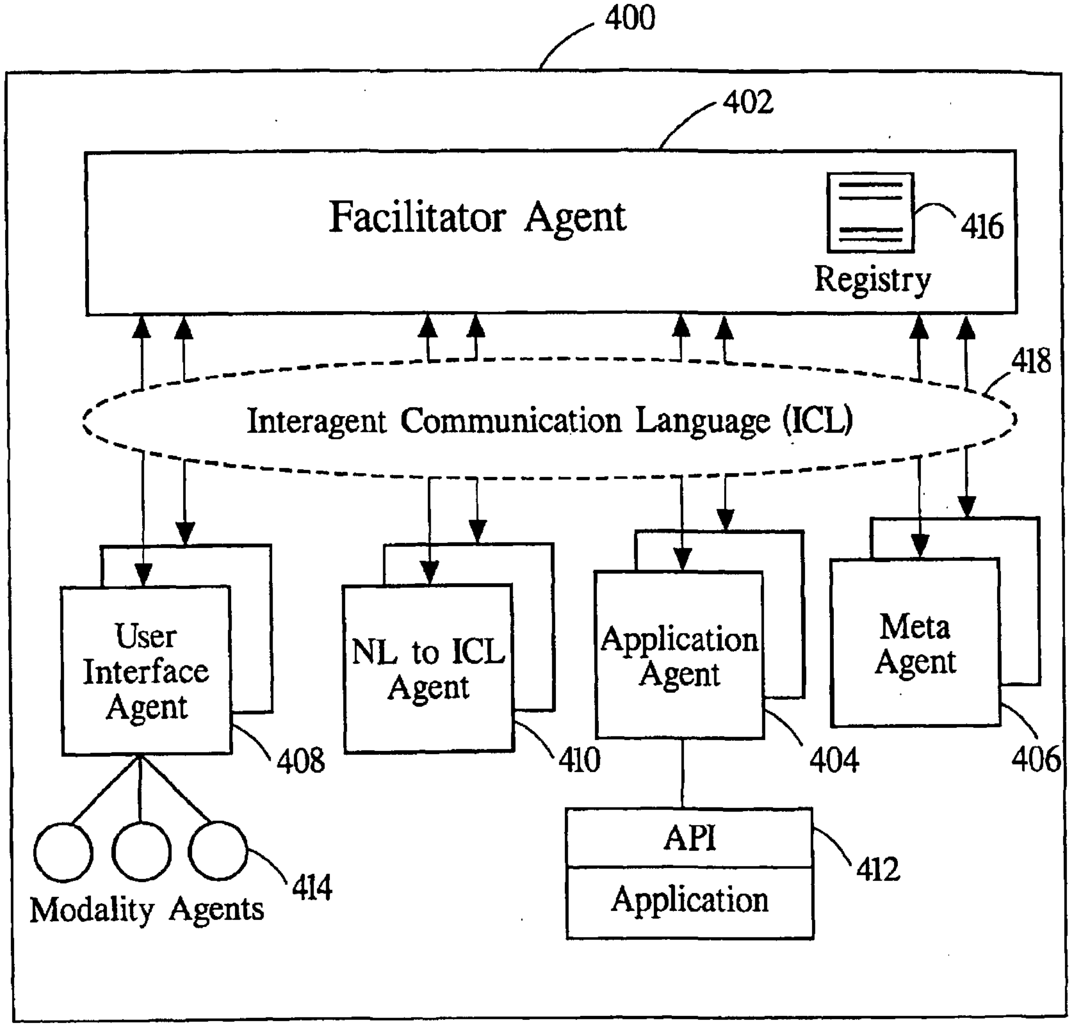
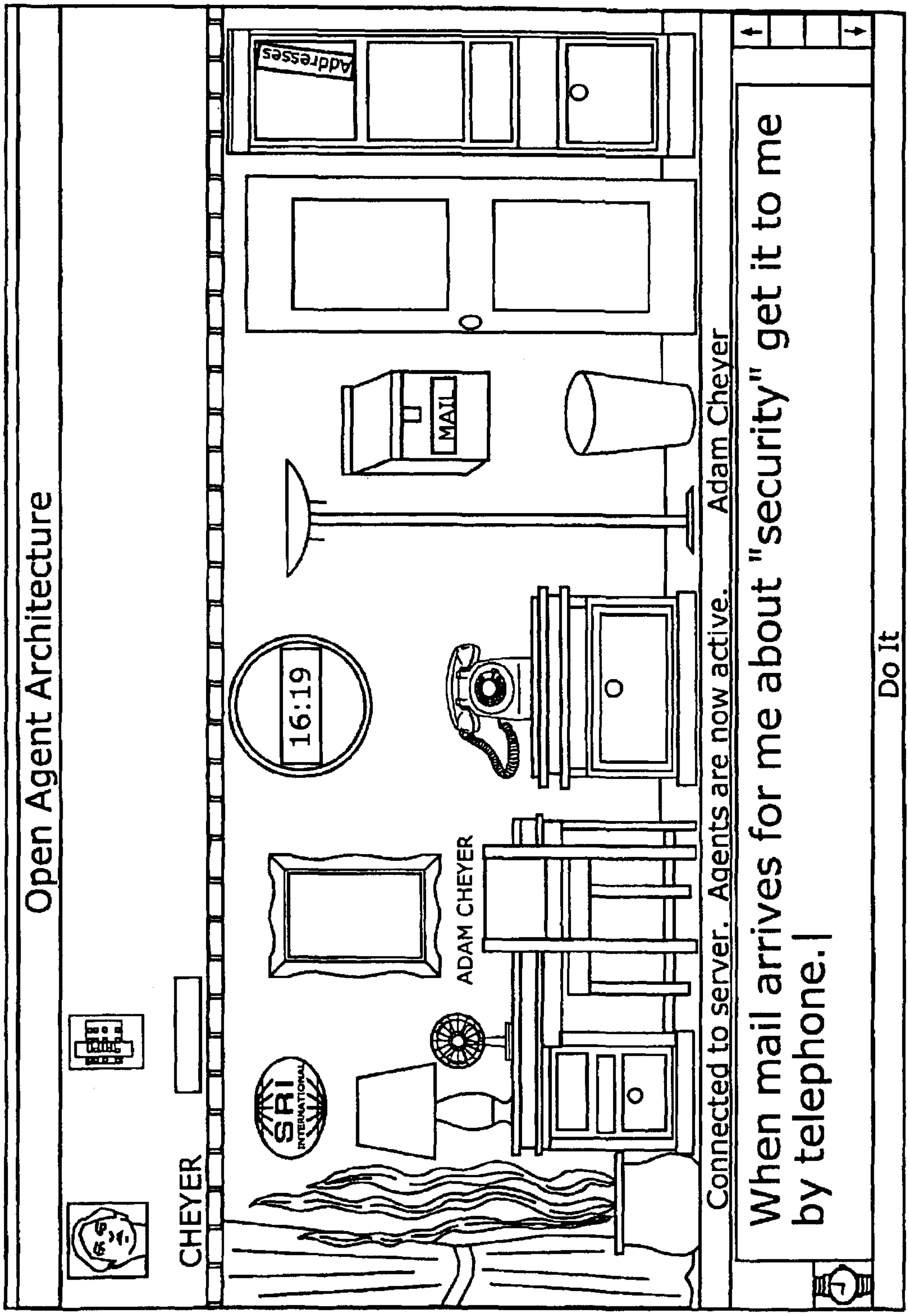

400
402
Facilitator Agent
416
Registry
418
Interagent Communication Language (ICL)
User Interface Agent
408
NL to ICL Agent
410
Application Agent
404
Meta Agent
406
API
Application
412
Modality Agents
414

Fig. 4



Connected to server. Agents are now active.

When mail arrives for me about "security" get it to me by telephone.|

Fig. 5

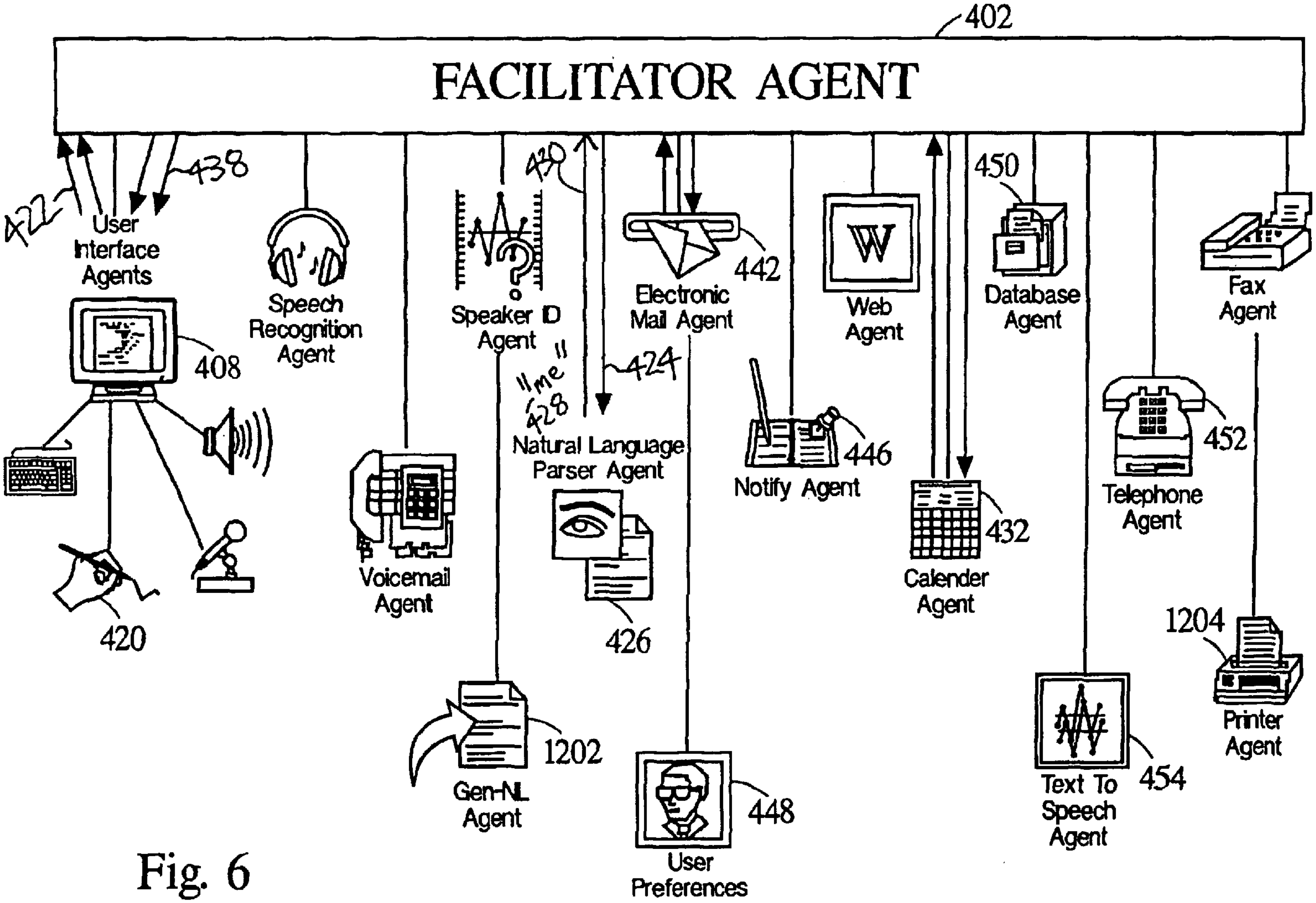

402
FACILITATOR AGENT
422
438
User Interface Agents
408
420
Speech Recognition Agent
Voicemail Agent
Speaker ID Agent
430
"me"
428
424
Natural Language Parser Agent
426
1202
Gen-NL Agent
Electronic Mail Agent
442
User Preferences
448
Notify Agent
446
Web Agent
W
Calender Agent
432
450
Database Agent
Text To Speech Agent
454
Telephone Agent
452
Fax Agent
1204
Printer Agent

Fig. 6

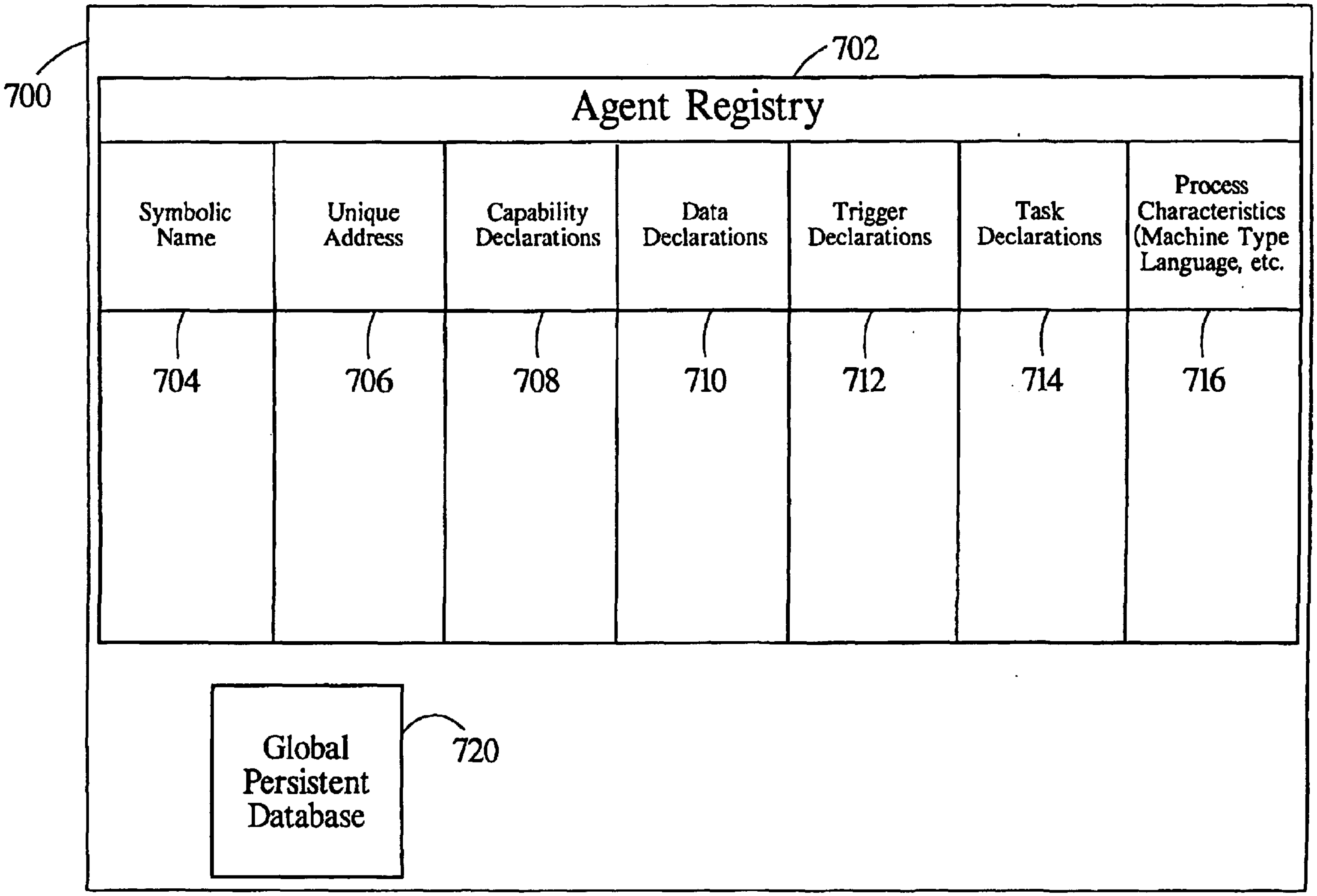

700
702
Agent Registry
Symbolic Name
Unique Address
Capability Declarations
Data Declarations
Trigger Declarations
Task Declarations
Process Characteristics (Machine Type Language, etc.
704
706
708
710
712
714
716
Global Persistent Database
720

Fig. 7

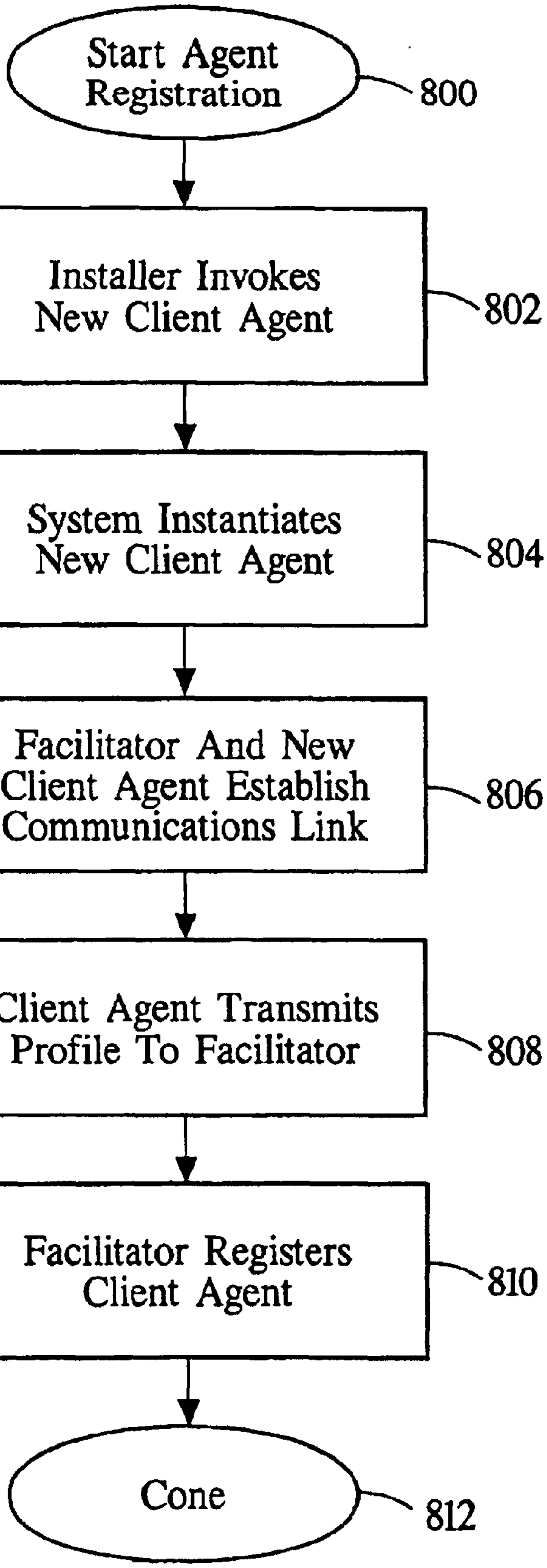

Start Agent Registration
800
Installer Invokes New Client Agent
802
System Instantiates New Client Agent
804
Facilitator And New Client Agent Establish Communications Link
806
Client Agent Transmits Profile To Facilitator
808
Facilitator Registers Client Agent
810
Cone
812

Fig. 8

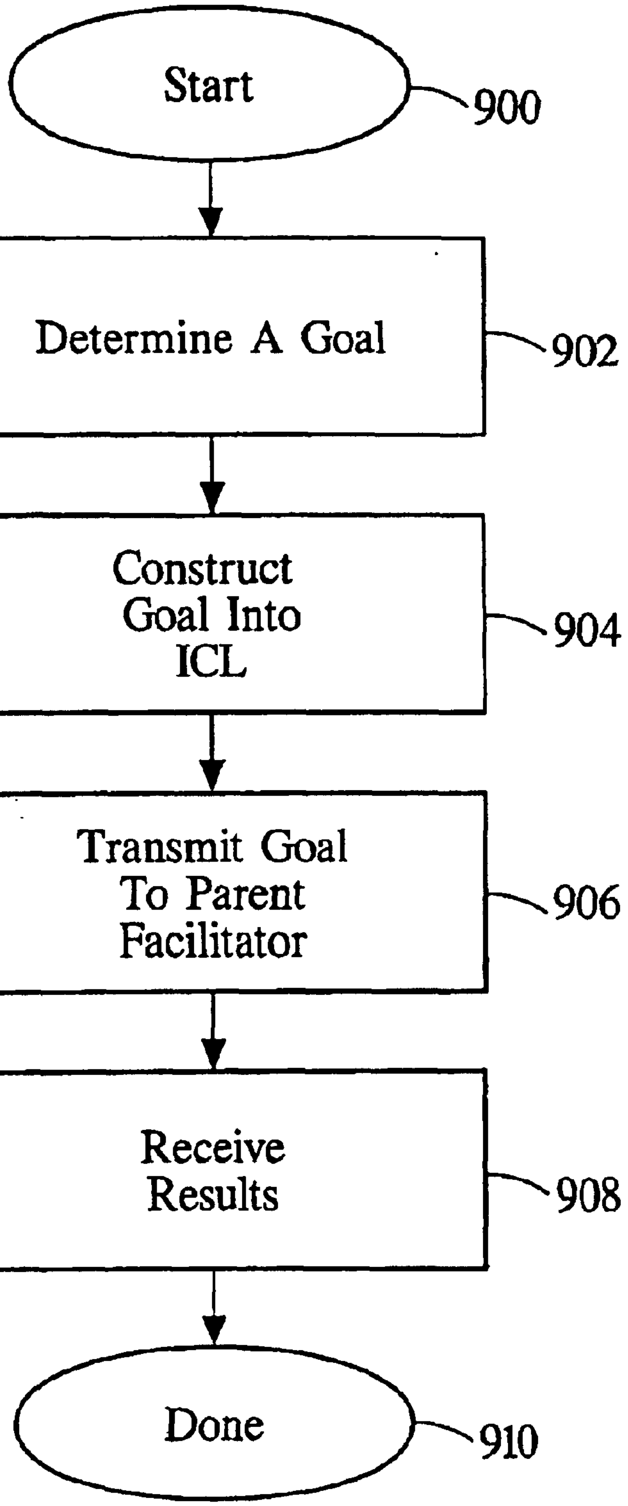

Start
900
Determine A Goal
902
Construct Goal Into ICL
904
Transmit Goal To Parent Facilitator
906
Receive Results
908
Done
910

Fig. 9

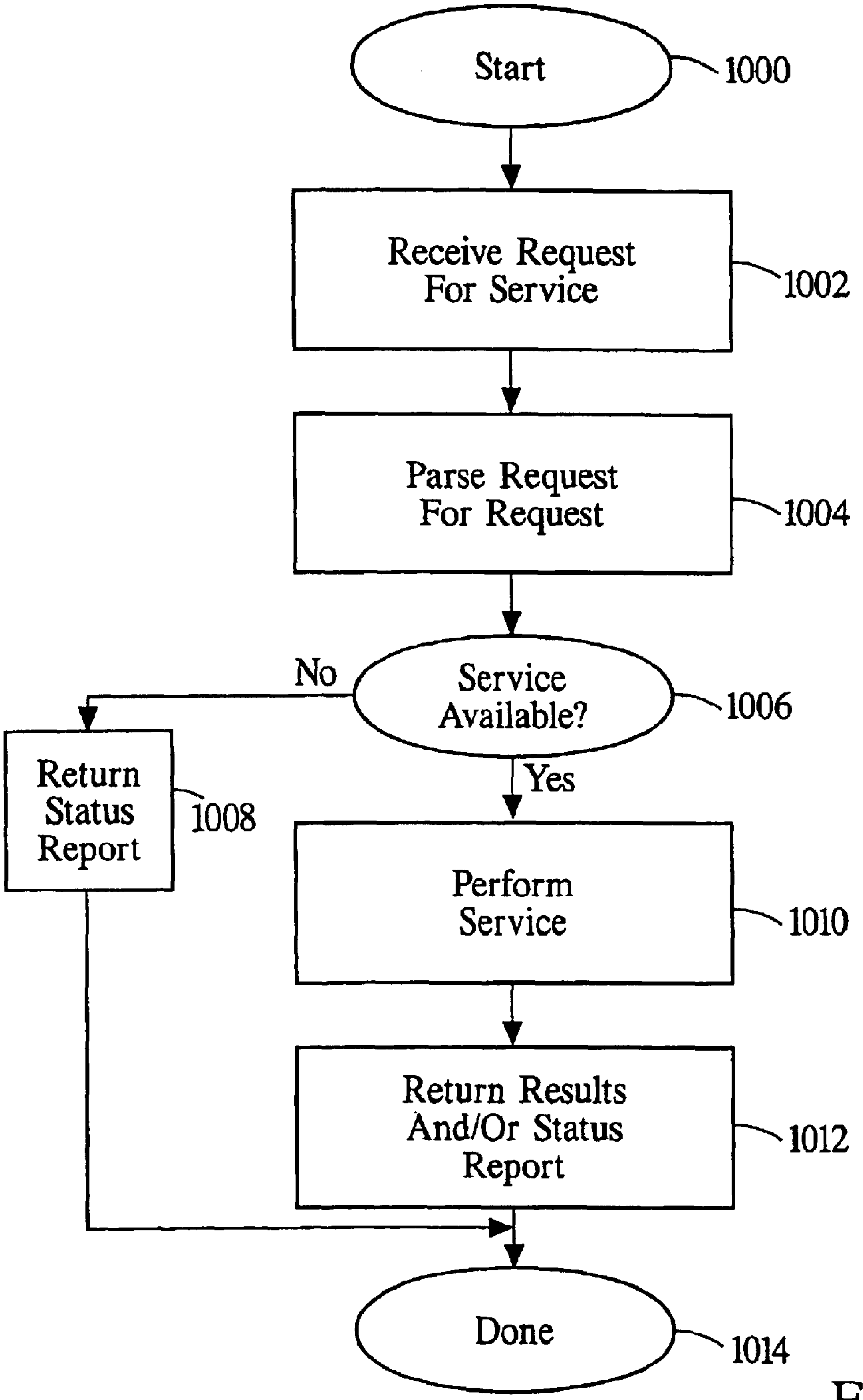

Start
1000
Receive Request For Service
1002
Parse Request For Request
1004
No
Service Available?
1006
Yes
Return Status Report
1008
Perform Service
1010
Return Results And/Or Status Report
1012
Done
1014

Fig. 10

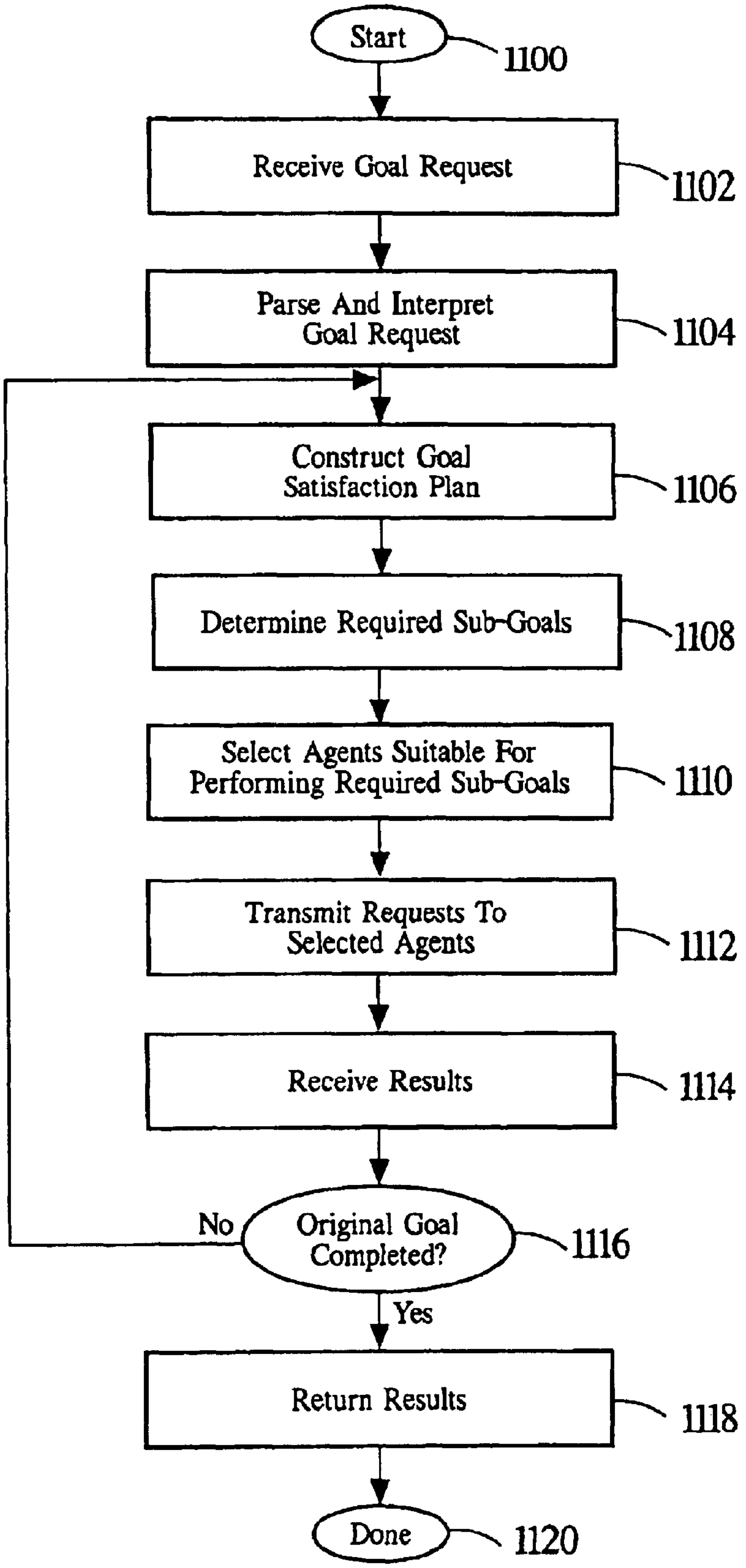

Start
1100
Receive Goal Request
1102
Parse And Interpret Goal Request
1104
Construct Goal Satisfaction Plan
1106
Determine Required Sub-Goals
1108
Select Agents Suitable For Performing Required Sub-Goals
1110
Transmit Requests To Selected Agents
1112
Receive Results
1114
No
Original Goal Completed?
1116
Yes
Return Results
1118
Done
1120

Fig. 11

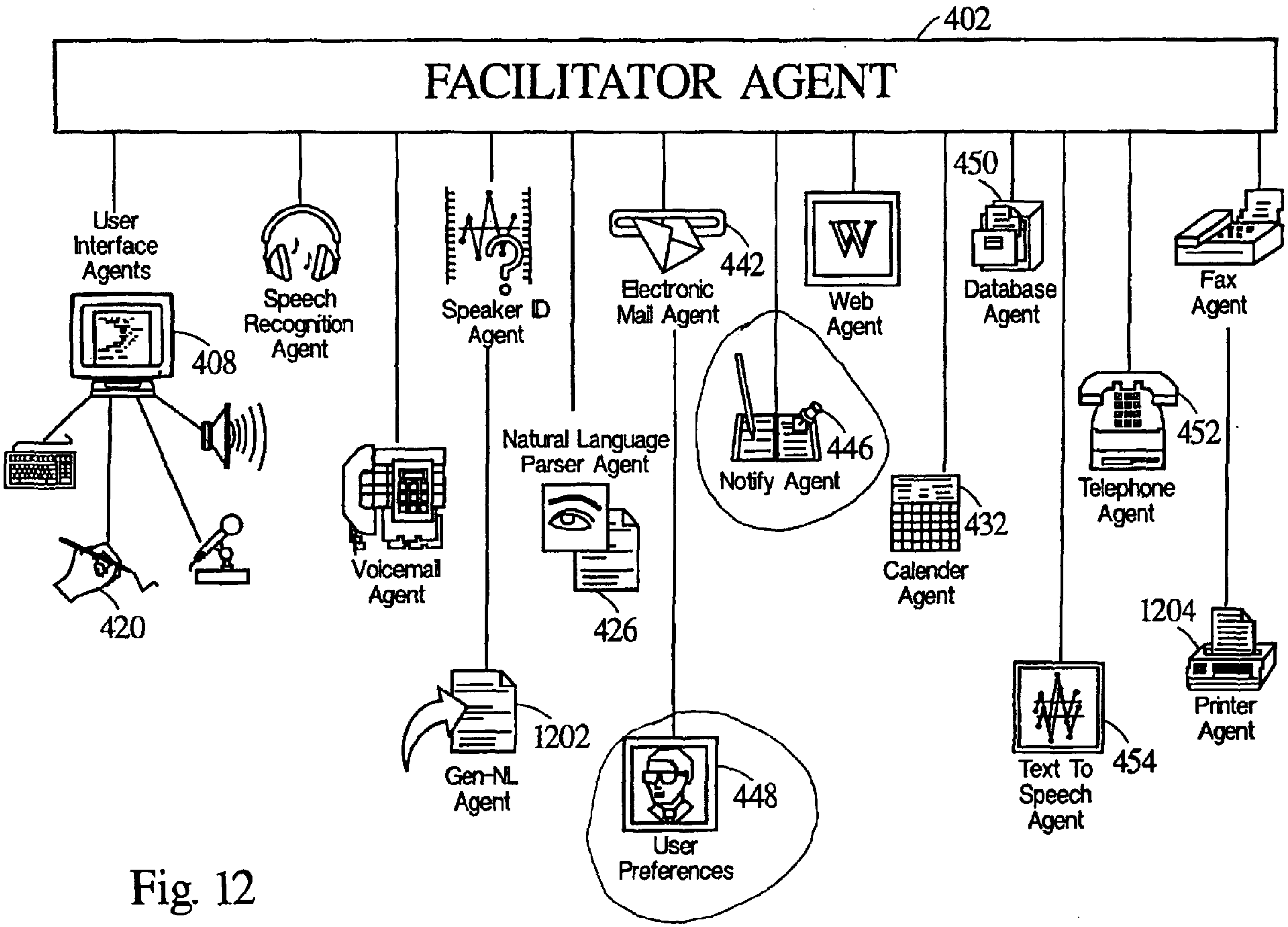

402
FACILITATOR AGENT
User Interface Agents
408
420
Speech Recognition Agent
Voicemail Agent
Speaker ID Agent
1202
Gen-NL Agent
Natural Language Parser Agent
426
Electronic Mail Agent
442
User Preferences
448
Notify Agent
446
Web Agent
450
Database Agent
Calender Agent
432
Text To Speech Agent
454
Telephone Agent
452
Fax Agent
1204
Printer Agent

Fig. 12

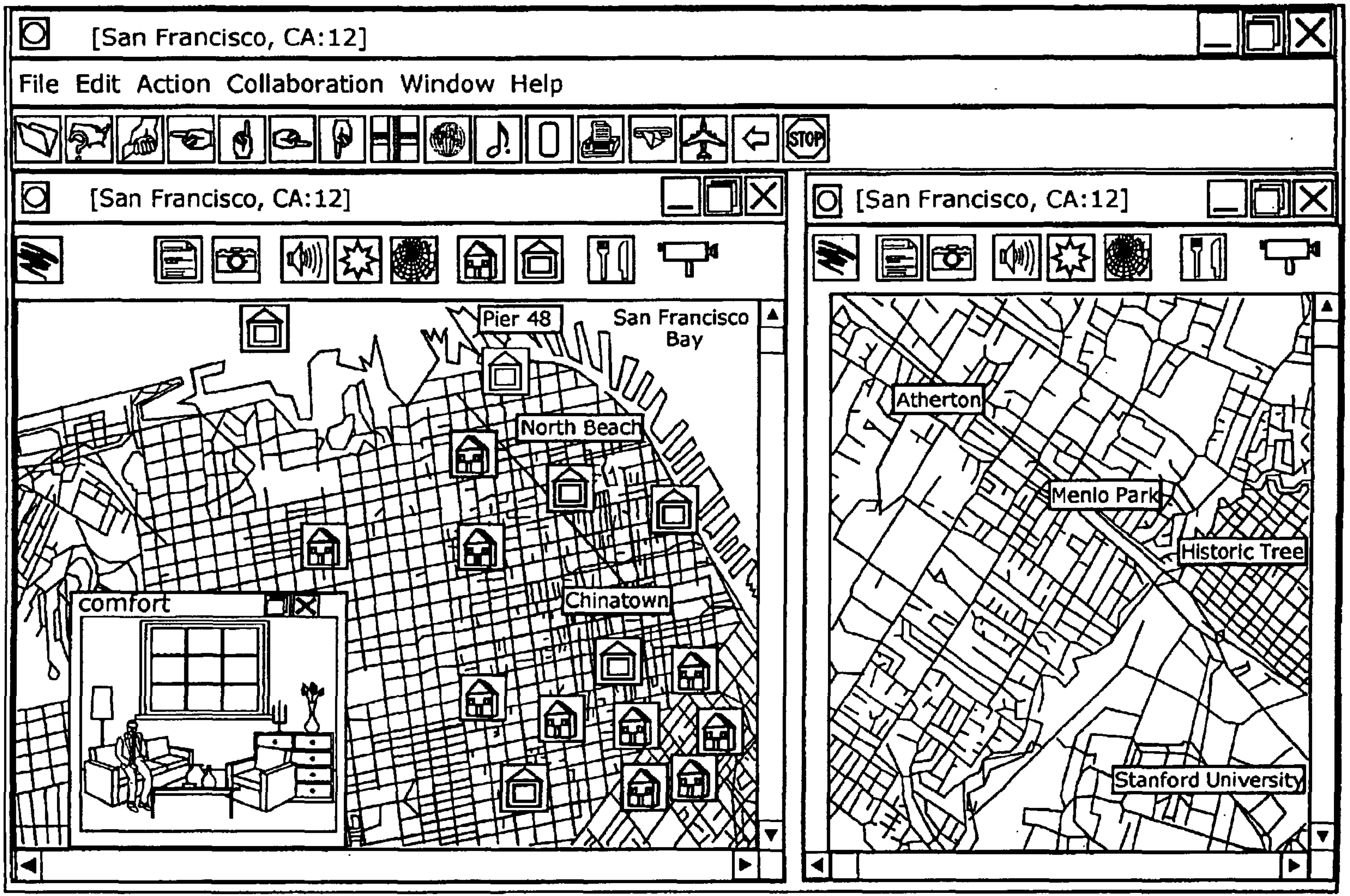

[San Francisco, CA:12]
File Edit Action Collaboration Window Help
STOP
[San Francisco, CA:12]
Pier 48
San Francisco Bay
North Beach
Chinatown
comfort
[San Francisco, CA:12]
Atherton
Menlo Park
Historic Tree
Stanford University

Fig. 13

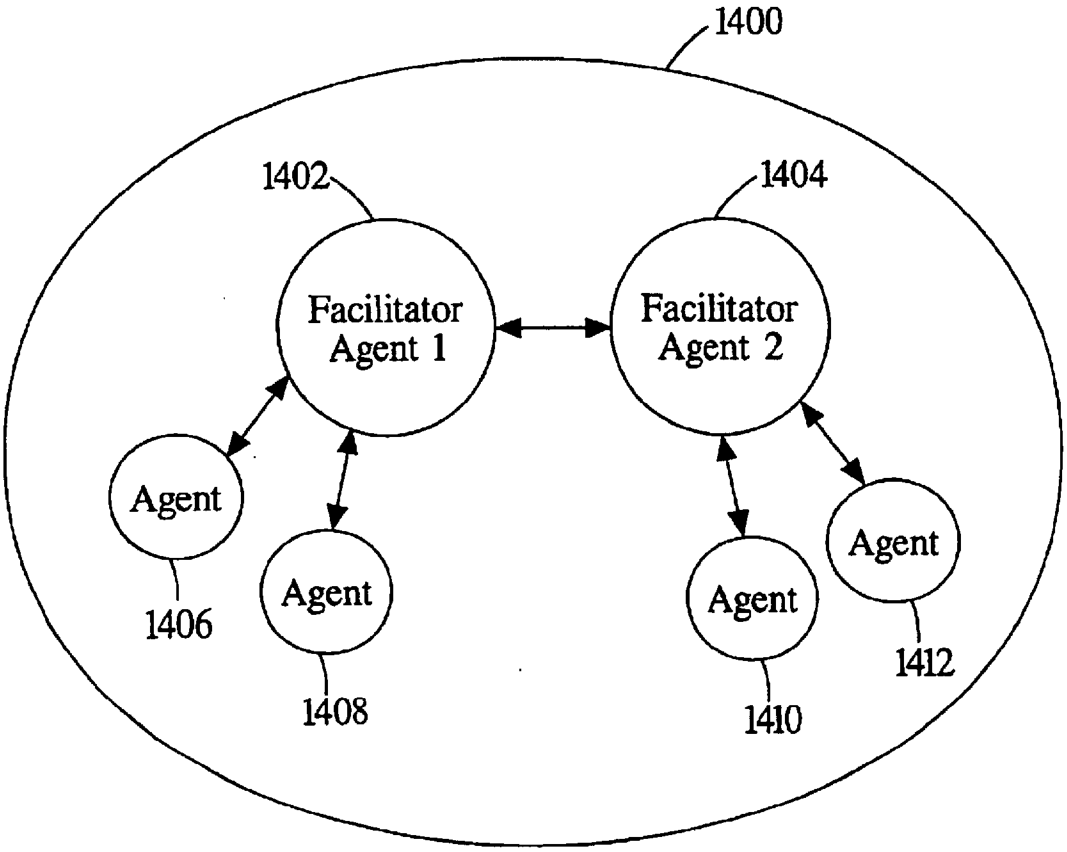

1400
1402
1404
Facilitator Agent 1
Facilitator Agent 2
Agent
1406
Agent
1408
Agent
1410
Agent
1412

Fig. 14

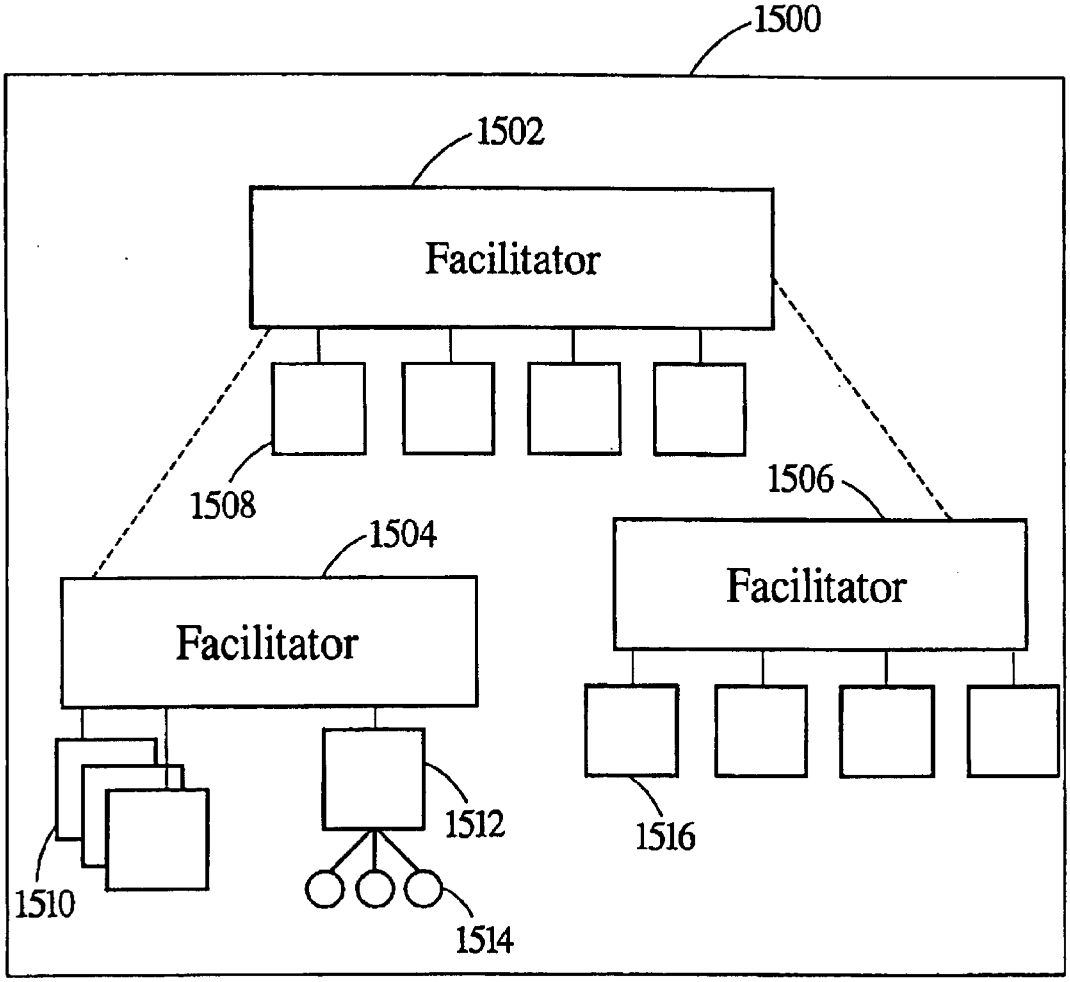

1500
1502
Facilitator
1508
1506
1504
Facilitator
Facilitator
1512
1516
1510
1514

Fig. 15

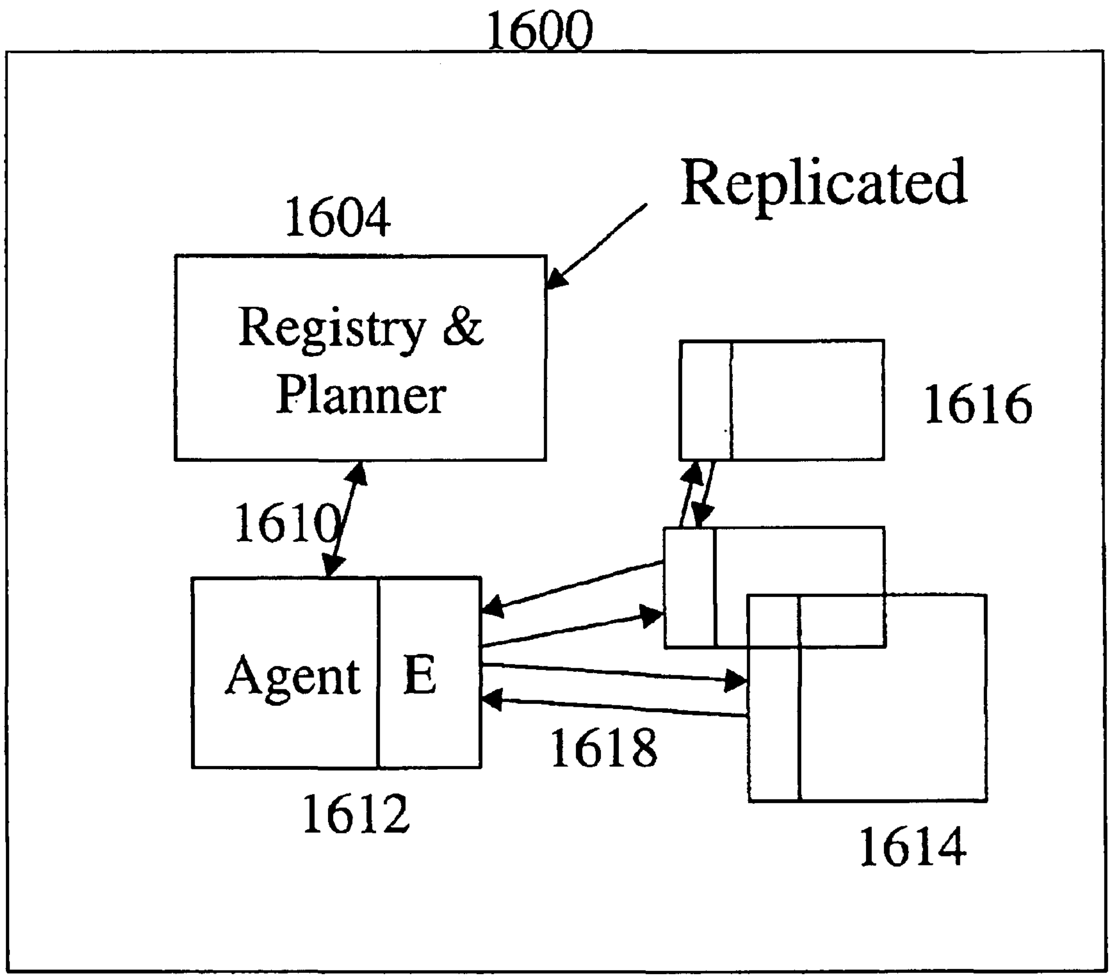

1600
Replicated
1604
Registry &
Planner
1616
1610
Agent
E
1618
1612
1614

Figure 16

SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS

A compact disk containing a computer program listing has been provided in duplicate (copy 1 and copy 2 of the compact disk are identical). The computer program listing in the compact disk is incorporated by reference herein. The compact disk contains files with their names, size and date of creation as follow:

File Name	Size	Creation Date	Last Date
oaa.pl	159,613 bytes	1996/10/08	1998/12/23
fac.pl	52,733 bytes	1997/04/24	1998/05/06
compound.pl	42,937 bytes	1996/12/11	1998/04/10
com_tcp.pl	18,010 bytes	1998/02/10	1998/05/06
translations.pl	19,583 bytes	1998/01/29	1998/12/23

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention is related to distributed computing environments and the completion of tasks within such environments. In particular, the present invention teaches a variety of software-based architectures for communication and cooperation among distributed electronic agents. Certain embodiments teach interagent communication languages enabling client agents to make requests in the form of arbitrarily complex goal expressions that are solved through facilitation by a facilitator agent.

Context and Motivation for Distributed Software Systems

The evolution of models for the design and construction of distributed software systems is being driven forward by several closely interrelated trends: the adoption of a networked computing model, rapidly rising expectations for smarter, longer-lived, more autonomous software applications and an ever increasing demand for more accessible and intuitive user interfaces.

Prior Art FIG. **1** illustrates a networked computing model **100** having a plurality of client and server computer systems **120** and **122** coupled together over a physical transport mechanism **140**. The adoption of the networked computing model **100** has lead to a greatly increased reliance on distributed sites for both data and processing resources. Systems such as the networked computing model **100** are based upon at least one physical transport mechanism **140** coupling the multiple computer systems **120** and **122** to support the transfer of information between these computers.

Some of these computers basically support using the network and are known as client computers (clients). Some of these computers provide resource to other computers and are known as server computers (servers). The servers **122** can vary greatly in the resources they possess, access they provide and services made available to other computers across a network. Servers may service other servers as well as clients.

The Internet is a computing system based upon this network computing model. The Internet is continually growing, stimulating a paradigm shift for computing away from requiring all relevant data and programs to reside on the user's desktop machine. The data now routinely accessed from computers spread around the world has become increasingly rich in format, comprising multimedia documents, and audio and video streams. With the popularization of programming languages such as JAVA, data transported between local and remote machines may also include programs that can be downloaded and executed on the local machine. There is an ever increasing reliance on networked computing, necessitating software design approaches that allow for flexible composition of distributed processing elements in a dynamically changing and relatively unstable environment.

In an increasing variety of domains, application designers and users are coming to expect the deployment of smarter, longer-lived, more autonomous, software applications. Push technology, persistent monitoring of information sources, and the maintenance of user models, allowing for personalized responses and sharing of preferences, are examples of the simplest manifestations of this trend. Commercial enterprises are introducing significantly more advanced approaches, in many cases employing recent research results from artificial intelligence, data mining, machine learning, and other fields.

More than ever before, the increasing complexity of systems, the development of new technologies, and the availability of multimedia material and environments are creating a demand for more accessible and intuitive user interfaces. Autonomous, distributed, multi-component systems providing sophisticated services will no longer lend themselves to the familiar "direct manipulation" model of interaction, in which an individual user masters a fixed selection of commands provided by a single application. Ubiquitous computing, in networked environments, has brought about a situation in which the typical user of many software services is likely to be a non-expert, who may access a given service infrequently or only a few times. Accommodating such usage patterns calls for new approaches, fortunately, input modalities now becoming widely available, such as speech recognition and pen-based handwriting/gesture recognition, and the ability to manage the presentation of systems' responses by using multiple media provide an opportunity to fashion a style of human-computer interaction that draws much more heavily on our experience with human-human interactions.

2. Prior Related Art

Existing approaches and technologies for distributed computing include to distributed objects, mobile objects, blackboard-style architectures, and agent-based software engineering.

The Distributed Object Approach

Object-oriented languages, such as C++ or JAVA, provide significant advances over standard procedural languages with respect to the reusability and modularity of code: encapsulation, inheritance and polymorhpism. Encapsulation encourages the creation of library interfaces that minimize dependencies on underlying algorithms or data structures. Changes to programming internals can be made at a later date with requiring modifications to the code that uses the library. Inheritance permits the extension and modification of a library of routines and data without requiring source code to the original library. Polymorphism allows one body of code to work on an arbitrary number of data types. For the sake of simplicity traditional objects may be seen to contain both methods and data. Methods provide the mechanisms by which the internal state of an object may be modified or by which communication may occur with another object or by which the instantiation or removal of objects may be directed.

With reference to FIG. **2**, a distributed object technology based around an Object Request Broker will now be described. Whereas "standard" object-oriented programming (OOP) languages can be used to build monolithic programs out of many object building blocks, distributed object technologies (DOOP) allow the creation of programs whose components may be spread across multiple machines. As shown in FIG. **2**, an object system **200** includes client objects **210** and server objects **220**. To implement a client-server relationship between objects, the distributed object system **200** uses a registry mechanism (CORBA's registry is called an object Request Broker, or ORB) **230** to store the interface descriptions of available objects. Through the services of the ORB **230**, a client can transparently invoke a method on a remote server object. The ORB **230** is then responsible for finding the object **220** that can implement the request, passing it the parameters, invoking its method, and returning the results. In the most sophisticated systems, the client **210** does not have to be aware of where the object is located, its programming language, its operating system, or any other system aspects that are not part of the server object's interface.

Although distributed objects offer a powerful paradigm for creating networked applications, certain aspects of the approach are not perfectly tailored to the constantly changing environment of the Internet. A major restriction of the DOOP approach is that the interactions among objects are fixed through explicitly coded instructions by the application developer. It is often difficult to reuse an object in a new application without bringing along all its inherent dependencies on other objects (embedded interface definitions and explicit method calls). Another restriction of the DOOP approach is the result of its reliance on a remote procedure call (RPC) style of communication. Although easy to debug, this single thread of execution model does not facilitate programming to exploit the potential for parallel computation that one would expect in a distributed environment. In addition, RPC uses a blocking (synchronous) scheme that does not scale well for high-volume transactions.

Mobile Objects

Mobile objects, sometimes called mobile agents, are bits of code that can move to another execution site (presumably on a different machine) under their own programmatic control, where they can then interact with the local environment. For certain types of problems, the mobile object paradigm offers advantages over more traditional distributed object approaches. These advantages include network bandwidth and parallelism. Network bandwidth advantages exist for some database queries or electronic commerce applications, where it is more efficient to perform tests on data by bringing the tests to the data than by bringing large amounts of data to the testing program. Parallelism advantages include situations in which mobile agents can be spawned in parallel to accomplish many tasks at once.

Some of the disadvantages and inconveniences of the mobile agent approach include the programmatic specificity of the agent interactions, lack of coordination support between participant agents and execution environment irregularities regarding specific programming languages supported by host processors upon which agents reside. In a fashion similar to that of DOOP programming, an agent developer must programmatically specify where to go and how to interact with the target environment. There is generally little coordination support to encourage interactions among multiple (mobile) participants. Agents must be written in the programming language supported by the execution environment, whereas many other distributed technologies support heterogeneous communities of components, written in diverse programming languages.

Blackboard Architectures

Blackboard architectures typically allow multiple processes to communicate by reading and writing tuples from a global data store. Each process can watch for items of interest, perform computations based on the state of the blackboard, and then add partial results or queries that other processes can consider. Blackboard architectures provide a flexible framework for problem solving by a dynamic community of distributed processes. A blackboard architecture provides one solution to eliminating the tightly bound interaction links that some of the other distributed technologies require during interprocess communication. This advantage can also be a disadvantage: although a programmer does not need to refer to a specific process during computation, the framework does not provide programmatic control for doing so in cases where this would be practical.

Agent-based Software Engineering

Several research communities have approached distributed computing by casting it as a problem of modeling communication and cooperation among autonomous entities, or agents. Effective communication among independent agents requires four components: (1) a transport mechanism carrying messages in an asynchronous fashion, (2) an interaction protocol defining various types of communication interchange and their social implications (for instance, a response is expected of a question), (3) a content language permitting the expression and interpretation of utterances, and (4) an agreed-upon set of shared vocabulary and meaning for concepts often called an ontology). Such mechanisms permit a much richer style of interaction among participants than can be expressed using a distributed object's RPC model or a blackboard architecture's centralized exchange approach.

Agent-based systems have shown much promise for flexible, fault-tolerant, distributed problem solving. Several agent-based projects have helped to evolve the notion of facilitation. However, existing agent-based technologies and architectures are typically very limited in the extent to which agents can specify complex goals or influence the strategies used by the facilitator. Further, such prior systems are not sufficiently attuned to the importance of integrating human agents (i.e., users) through natural language and other human-oriented user interface technologies.

The initial version of SRI International's Open Agent Architecture™ ("OAA®") technology provided only a very limited mechanism for dealing with compound goals. Fixed formats were available for specifying a flat list of either conjoined (AND) sub-goals or disjoined (OR) sub-goals; in both cases, parallel goal solving was hard-wired in, and only a single set of parameters for the entire list could be specified. More complex goal expressions involving (for example) combinations of different boolean connectors, nested expressions, or conditionally interdependent ("IF . . . THEN") goals were not supported. Further, system scalability was not adequately addressed in this prior work.

SUMMARY OF INVENTION

A first embodiment of the present invention discloses a highly flexible, software-based architecture for constructing distributed systems. The architecture supports cooperative task completion by flexible, dynamic configurations of autonomous electronic agents. Communication and cooperation between agents are brokered by one or more facilitators, which are responsible for matching requests, from users and agents, with descriptions of the capabilities of other agents. It is not generally required that a user or agent know the identities, locations, or number of other agents involved in satisfying a request, and relatively minimal effort is involved in incorporating new agents and "wrapping" legacy applications. Extreme flexibility is achieved through an architecture organized around the declaration of capabilities by service-providing agents, the construction of arbitrarily complex goals by users and service-requesting agents, and the role of facilitators in delegating and coordinating the satisfaction of these goals, subject to advice and constraints that may accompany them. Additional mechanisms and features include facilities for creating and maintaining shared repositories of data; the use of triggers to instantiate commitments within and between agents; agent-based provision of multi-modal user interfaces, including natural language; and built-in support for including the user as a privileged member of the agent community. Specific embodiments providing enhanced scalability are also described.

BRIEF DESCRIPTION OF THE DRAWINGS

Prior Art

Prior Art FIG. **1** depicts a networked computing model;

Prior Art FIG. **2** depicts a distributed object technology based around an Object Resource Broker;

Examples of the Invention

FIG. **3** depicts a distributed agent system based around a facilitator agent;

FIG. **4** presents a structure typical of one small system of the present invention;

FIG. **5** depicts an Automated Office system implemented in accordance with an example embodiment of the present invention supporting a mobile user with a laptop computer and a telephone;

FIG. **6** schematically depicts an Automated Office system implemented as a network of agents in accordance with a preferred embodiment of the present invention;

FIG. **7** schematically shows data structures internal to a facilitator in accordance with a preferred embodiment of the present invention;

FIG. **8** depicts operations involved in instantiating a client agent with its parent facilitator in accordance with a preferred embodiment of the present invention;

FIG. **9** depicts operations involved in a client agent initiating a service request and receiving the response to that service request in accordance with a certain preferred embodiment of the present invention;

FIG. **10** depicts operations involved in a client agent responding to a service request in accordance with another preferable embodiment of the present invention;

FIG. **11** depicts operations involved in a facilitator agent response to a service request in accordance with a preferred embodiment of the present invention;

FIG. **12** depicts an Open Agent Architecture™ based system of agents implementing a unified messaging application in accordance with a preferred embodiment of the present invention;

FIG. **13** depicts a map oriented graphical user interface display as might be displayed by a multi-modal map application in accordance with a preferred embodiment of the present invention;

FIG. **14** depicts a peer to peer multiple facilitator based agent system supporting distributed agents in accordance with a preferred embodiment of the present invention;

FIG. **15** depicts a multiple facilitator agent system supporting at least a limited form of a hierarchy of facilitators in accordance with a preferred embodiment of the present invention; and

FIG. **16** depicts a replicated facilitator architecture in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

FIG. **3** illustrates a distributed agent system **300** in accordance with one embodiment of the present invention. The agent system **300** includes a facilitator agent **310** and a plurality of agents **320**. The illustration of FIG. **3** provides a high level view of one simple system structure contemplated by the present invention. The facilitator agent **310** is in essence the "parent" facilitator for its "children" agents **320**. The agents **320** forward service requests to the facilitator agent **310**. The facilitator agent **310** interprets these requests, organizing a set of goals which are then delegated to appropriate agents for task completion.

The system **300** of FIG. **3** can be expanded upon and modified in a variety of ways consistent with the present invention. For example, the agent system **300** can be distributed across a computer network such as that illustrated in FIG. **1**. The facilitator agent **310** may itself have its functionality distributed across several different computing platforms. The agents **320** may engage in interagent communication (also called peer to peer communications). Several different systems **300** may be coupled together for enhanced performance. These and a variety of other structural configurations are described below in greater detail.

FIG. **4** presents the structure typical of a small system **400** in one embodiment of the present invention, showing user interface agents **408**, several application agents **404** and meta-agents **406**, the system **400** organized as a community of peers by their common relationship to a facilitator agent **402**. As will be appreciated, FIG. **4** places more structure upon the system **400** than shown in FIG. **3**, but both are valid representations of structures of the present invention. The facilitator **402** is a specialized server agent that is responsible for coordinating agent communications and cooperative problem-solving. The facilitator **402** may also provide a global data store for its client agents, allowing them to adopt a blackboard style of interaction. Note that certain advantages are found in utilizing two or more facilitator agents within the system **400**. For example, larger systems can be assembled from multiple facilitator/client groups, each having the sort of structure shown in FIG. **4**. All agents that are not facilitators are referred to herein generically as client agents—so called because each acts (in some respects) as a client of some facilitator, which provides communication and other essential services for the client.

The variety of possible client agents is essentially unlimited. Some typical categories of client agents would include application agents **404**, meta-agents **406**, and user interface agents **408**, as depicted in FIG. **4**. Application agents **404** denote specialists that provide a collection of services of a particular sort. These services could be domain-independent technologies (such as speech recognition, natural language processing **410**, email, and some forms of data retrieval and data mining) or user-specific or domain-specific (such as a travel planning and reservations agent). Application agents may be based on legacy applications or libraries, in which case the agent may be little more than a wrapper that calls a pre-existing API **412**, for example. Meta-agents **406** are agents whose role is to assist the facilitator agent **402** in coordinating the activities of other agents. While the facilitator **402** possesses domain-independent coordination strategies, meta-agents **406** can augment these by using domain- and application-specific knowledge or reasoning (including but not limited to rules, learning algorithms and planning).

With further reference to FIG. **4**, user interface agents **408** can play an extremely important and interesting role in certain embodiments of the present invention. By way of explanation, in some systems, a user interface agent can be implemented as a collection of "micro-agents", each monitoring a different input modality (point-and-click, handwriting, pen gestures, speech), and collaborating to produce the best interpretation of the current inputs. These micro-agents are depicted in FIG. **4**, for example, as Modality Agents **414**. While describing such subcategories of client agents is useful for purposes of illustration and understanding, they need not be formally distinguished within the system in preferred implementations of the present invention.

The operation of one preferred embodiment of the present invention will be discussed in greater detail below, but may be briefly outlined as follows. When invoked, a client agent makes a connection to a facilitator, which is known as its parent facilitator. These connections are depicted as a double headed arrow between the client agent and the facilitator agent in FIGS. **3** and **4**, for example. Upon connection, an agent registers with its parent facilitator a specification of the capabilities and services it can provide. For example, a natural language agent may register the characteristics of its available natural language vocabulary. (For more details regarding client agent connections, see the discussion of FIG. **8** below.) Later during task completion, when a facilitator determines that the registered services **416** of one of its client agents will help satisfy a goal, the facilitator sends that client a request expressed in the Interagent Communication Language (ICL) **418**. (See FIG. **11** below for a more detailed discussion of the facilitator operations involved.) The agent parses this request, processes it, and returns answers or status reports to the facilitator. In processing a request, the client agent can make use of a variety of infrastructure capabilities provided in the preferred embodiment. For example, the client agent can use ICL **418** to request services of other agents, set triggers, and read or write shared data on the facilitator or other client agents that maintain shared data. (See the discussion of FIGS. **9**–**11** below for a more detailed discussion of request processing.)

The functionality of each client agent are made available to the agent community through registration of the client agent's capabilities with a facilitator **402**. A software "wrapper" essentially surrounds the underlying application program performing the services offered by each client. The common infrastructure for constructing agents is preferably supplied by an agent library. The agent library is preferably accessible in the runtime environment of several different programming languages. The agent library preferably minimizes the effort required to construct a new system and maximizes the ease with which legacy systems can be "wrapped" and made compatible with the agent-based architecture of the present invention.

By way of further illustration, a representative application is now briefly presented with reference to FIGS. **5** and **6**. In the Automated Office system depicted in FIG. **5**, a mobile user with a telephone and a laptop computer can access and task commercial applications such as calendars, databases, and email systems running back at the office. A user interface (UI) agent **408**, shown in FIG. **6**, runs on the user's local laptop and is responsible for accepting user input, sending requests to the facilitator **402** for delegation to appropriate agents, and displaying the results of the distributed computation. The user may interact directly with a specific remote application by clicking on active areas in the interface, calling up a form or window for that application, and making queries with standard interface dialog mechanisms. Conversely, a user may express a task to be executed by using typed, handwritten, or spoken (over the telephone) English sentences, without explicitly specifying which agent or agents should perform the task.

For instance, if the question "What is my schedule?" is written **420** in the user interface **408**, this request will be sent **422** by the UI **408** to the facilitator **402**, which in turn will ask **424** a natural language (NL) agent **426** to translate the query into JCL **18**. To accomplish this task, the NL agent **426** may itself need to make requests of the agent community to resolve unknown words such as "me" **428** (the UI agent **408** can respond **430** with the name of the current user) or "schedule" **432** (the calendar agent **434** defines this word **436**). The resulting ICL expression is then routed by the facilitator **402** to appropriate agents (in this case, the calendar agent **434**) to execute the request. Results are sent back **438** to the UI agent **408** for display.

The spoken request "When mail arrives for me about security, notify me immediately." produces a slightly more complex example involving communication among all agents in the system. After translation into ICL as described above, the facilitator installs a trigger **440** on the mail agent **442** to look for new messages about security. When one such message does arrive in its mail spool, the trigger fires, and the facilitator matches the action part of the trigger to capabilities published by the notification agent **446**. The notification agent **446** is a meta-agent, as it makes use of rules concerning the optimal use of different output modalities (email, fax, speech generation over the telephone) plus information about an individual user's preferences **448** to determine the best way of relaying a message through available media transfer application agents. After some competitive parallelism to locate the user (the calendar agent **434** and database agent **450** may have different guesses as to where to find the user) and some cooperative parallelism to produce required information (telephone number of location, user password, and an audio file containing a text-to-speech representation of the email message), a telephone agent **452** calls the user, verifying its identity through touchtones, and then play the message.

The above example illustrates a number of inventive features. As new agents connect to the facilitator, registering capability specifications and natural language vocabulary, what the user can say and do dynamically changes; in other words, the ICL is dynamically expandable. For example, adding a calendar agent to the system in the previous example and registering its capabilities enables users to ask natural language questions about their "schedule" without any need to revise code for the facilitator, the natural language agents, or any other client agents. In addition, the interpretation and execution of a task is a distributed process, with no single agent defining the set of possible inputs to the system. Further, a single request can produce cooperation and flexible communication among many agents, written in different programming languages and spread across multiple machines.

Design Philosophy and Considerations

One preferred embodiment provides an integration mechanism for heterogeneous applications in a distributed infrastructure, incorporating some of the dynamism and extensibility of blackboard approaches, the efficiency associated with mobile objects, plus the rich and complex interactions of communicating agents. Design goals for preferred embodiments of the present invention may be categorized under the general headings of interoperation and cooperation, user interfaces, and software engineering. These design goals are not absolute requirements, nor will they necessarily be satisfied by all embodiments of the present invention, but rather simply reflect the inventor's currently preferred design philosophy.

Versatile Mechanisms of Interoperation and Cooperation

Interoperation refers to the ability of distributed software components—agents—to communicate meaningfully. While every system-building framework must provide mechanisms of interoperation at some level of granularity, agent-based frameworks face important new challenges in this area. This is true primarily because autonomy, the hallmark of individual agents, necessitates greater flexibility in interactions within communities of agents. Coordination refers to the mechanisms by which a community of agents is able to work together productively on some task. In these areas, the goals for our framework are to provide flexibility in assembling communities of autonomous service providers, provide flexibility in structuring cooperative interactions, impose the right amount of structure, as well as include legacy and "owned-elsewhere" applications.

Provide flexibility in assembling communities of autonomous service providers—both at development time and at runtime. Agents that conform to the linguistic and ontological requirements for effective communication should be able to participate in an agent community, in various combinations, with minimal or near minimal prerequisite knowledge of the characteristics of the other players. Agents with duplicate and overlapping capabilities should be able to coexist within the same community, with the system making optimal or near optimal use of the redundancy.

Provide flexibility in structuring cooperative interactions among the members of a community of agents. A framework preferably provides an economical mechanism for setting up a variety of interaction patterns among agents, without requiring an inordinate amount of complexity or infrastructure within the individual agents. The provision of a service should be independent or minimally dependent upon a particular configuration of agents.

Impose the right amount of structure on individual agents. Different approaches to the construction of multi-agent systems impose different requirements on the individual agents. For example, because KQML is neutral as to the content of messages, it imposes minimal structural requirements on individual agents. On the other hand, the BDI paradigm tends to impose much more demanding requirements, by making assumptions about the nature of the programming elements that are meaningful to individual agents. Preferred embodiments of the present invention should fall somewhere between the two, providing a rich set of interoperation and coordination capabilities, without precluding any of the software engineering goals defined below.

Include legacy and "owned-elsewhere" applications. Whereas legacy usually implies reuse of an established system fully controlled by the agent-based system developer, owned-elsewhere refers to applications to which the developer has partial access, but no control. Examples of owned-elsewhere applications include data sources and services available on the World Wide Web, via simple form-based interfaces, and applications used cooperatively within a virtual enterprise, which remain the properties of separate corporate entities. Both classes of application must preferably be able to interoperate, more or less as full-fledged members of the agent community, without requiring an overwhelming integration effort.

Human-Oriented User Interfaces

Systems composed of multiple distributed components, and possibly dynamic configurations of components, require the crafting of intuitive user interfaces to provide conceptually natural interaction mechanisms, treat users as privileged members of the agent community and support collaboration.

Provide conceptually natural interaction mechanisms with multiple distributed components. When there are numerous disparate agents, and/or complex tasks implemented by the system, the user should be able to express requests without having detailed knowledge of the individual agents. With speech recognition, handwriting recognition, and natural language technologies becoming more mature, agent architectures should preferably support these forms of input playing increased roles in the tasking of agent communities.

Preferably treat users as privileged members of the agent community by providing an appropriate level of task specification within software agents, and reusable translation mechanisms between this level and the level of human requests, supporting constructs that seamlessly incorporate interactions between both human-interface and software types of agents.

Preferably support collaboration (simultaneous work over shared data and processing resources) between users and agents.

Realistic Software Engineering Requirements

System-building frameworks should preferably address the practical concerns of real-world applications by the specification of requirements which preferably include: Minimize the effort required to create new agents, and to wrap existing applications. Encourage reuse, both of domain-independent and domain-specific components. The concept of agent orientation, like that of object orientation, provides a natural conceptual framework for reuse, so long as mechanisms for encapsulation and interaction are structured appropriately. Support lightweight mobile platforms. Such platforms should be able to serve as hosts for agents, without requiring the installation of a massive environment. It should also be possible to construct individual agents that are relatively small and modest in their processing requirements. Minimize platform and language barriers. Creation of new agents, as well as wrapping of existing applications, should not require the adoption of a new language or environment.

Mechanisms of Cooperation

Cooperation among agents in accordance with the present invention is preferably achieved via messages expressed in a common language, ICL. Cooperation among agent is further preferably structured around a three-part approach: providers of services register capabilities specifications with a facilitator, requesters of services construct goals and relay them to a facilitator, and facilitators coordinate the efforts of the appropriate service providers in satisfying these goals.

The Interagent Communication Language (ICL)

Interagent Communication Language ("ICL") **418** refers to an interface, communication, and task coordination language preferably shared by all agents, regardless of what platform they run on or what computer language they are programmed in. ICL may be used by an agent to task itself or some subset of the agent community. Preferably, ICL allows agents to specify explicit control parameters while simultaneously supporting expression of goals in an underspecified, loosely constrained manner. In a further preferred embodiment, agents employ ICL to perform queries, execute actions, exchange information, set triggers, and manipulate data in the agent community.

In a further preferred embodiment, a program element expressed in ICL is the event. The activities of every agent, as well as communications between agents, are preferably structured around the transmission and handling of events. In communications, events preferably serve as messages between agents; in regulating the activities of individual agents, they may preferably be thought of as goals to be satisfied. Each event preferably has a type, a set of parameters, and content. For example, the agent library procedure oaa_Solve can be used by an agent to request services of other agents. A call to oaa_Solve, within the code of agent A, results in an event having the form

ev_post_solve(Goal, Params)

going from A to the facilitator, where ev_post_solve is the type, Goal is the content, and Params is a list of parameters. The allowable content and parameters preferably vary according to the type of the event.

The ICL preferably includes a layer of conversational protocol and a content layer. The conversational layer of ICL is defined by the event types, together with the parameter lists associated with certain of these event types. The content layer consists of the specific goals, triggers, and data elements that may be embedded within various events.

The ICL conversational protocol is preferably specified using an orthogonal, parameterized approach, where the conversational aspects of each element of an interagent conversation are represented by a selection of an event type and a selection of values from at least one orthogonal set of parameters. This approach offers greater expressiveness than an approach based solely on a fixed selection of speech acts, such as embodied in KQML. For example, in KQML, a request to satisfy a query can employ either of the performatives ask_all or ask_one. In ICL, on the other hand, this type of request preferably is expressed by the event type evost solve, together with the solution_limit(N) parameter—where N can be any positive integer. (A request for all solutions is indicated by the omission of the solution_limit parameter.) The request can also be accompanied by other parameters, which combine to further refine its semantics. In KQML, then, this example forces one to choose between two possible conversational options, neither of which may be precisely what is desired. In either case, the performative chosen is a single value that must capture the entire conversational characterization of the communication. This requirement raises a difficult challenge for the language designer, to select a set of performatives that provides the desired functionality without becoming unmanageably large. Consequently, the debate over the right set of performatives has consumed much discussion within the KQML community.

The content layer of the ICL preferably supports unification and other features found in logic programming language environments such as PROLOG. In some embodiments, the content layer of the ICL is simply an extension of at least one programming language. For example, the Applicants have found that PROLOG is suitable for implementing and extending into the content layer of the ICL. The agent libraries preferably provide support for constructing, parsing, and manipulating ICL expressions. It is possible to embed content expressed in other languages within an ICL event. However, expressing content in ICL simplifies the facilitator's access to the content, as well as the conversational layer, in delegating requests. This gives the facilitator more information about the nature of a request and helps the facilitator decompose compound requests and delegate the sub-requests.

Further, ICL expressions preferably include, in addition to events, at least one of the following: capabilities declarations, requests for services, responses to requests, trigger specifications, and shared data elements. A further preferred embodiment of the present invention incorporates ICL expressions including at least all of the following: events, capabilities declarations, requests for services, responses to requests, trigger specifications, and shared data elements.

Providing Services: Specifying "Solvables"

In a preferred embodiment of the present invention, every participating agent defines and publishes a set of capability declarations, expressed in ICL, describing the services that it provides. These declarations establish a high-level interface to the agent. This interface is used by a facilitator in communicating with the agent, and, most important, in delegating service requests (or parts of requests) to the agent. Partly due to the use of PROLOG as a preferred basis for ICL, these capability declarations are referred as solvables. The agent library preferably provides a set of procedures allowing an agent to add, remove, and modify its solvables, which it may preferably do at any time after connecting to its facilitator.

There are preferably at least two major types of solvables: procedure solvables and data solvables. Intuitively, a procedure solvable performs a test or action, whereas a data solvable provides access to a collection of data. For example, in creating an agent for a mail system, procedure solvables might be defined for sending a message to a person, testing whether a message about a particular subject has arrived in the mail queue, or displaying a particular message onscreen. For a database wrapper agent, one might define a distinct data solvable corresponding to each of the relations present in the database. Often, a data solvable is used to provide a shared data store, which may be not only queried, but also updated, by various agents having the required permissions.

There are several primary technical differences between these two types of solvables. First, each procedure solvable must have a handler declared and defined for it, whereas this is preferably not necessary for a data solvable. The handling of requests for a data solvable is preferably provided transparently by the agent library. Second, data solvables are preferably associated with a dynamic collection of facts (or clauses), which may be further preferably modified at runtime, both by the agent providing the solvable, and by other agents (provided they have the required permissions). Third, special features, available for use with data solvables, preferably facilitate maintaining the associated facts. In spite of these differences, it should be noted that the mechanism of use by which an agent requests a service is the same for the two types of solvables.

In one embodiment, a request for one of an agent's services normally arrives in the form of an event from the agent's facilitator. The appropriate handler then deals with this event. The handler may be coded in whatever fashion is most appropriate, depending on the nature of the task, and the availability of task-specific libraries or legacy code, if any. The only hard requirement is that the handler return an appropriate response to the request, expressed in ICL. Depending on the nature of the request, this response could be an indication of success or failure, or a list of solutions (when the request is a data query).

A solvable preferably has three parts: a goal, a list of parameters, and a list of permissions, which are declared using the format:

solvable(Goal, Parameters, Permissions)

The goal of a solvable, which syntactically takes the preferable form of an ICL structure, is a logical representation of the service provided by the solvable. (An ICL structure consists of a functor with 0 or more arguments. For example, in the structure a(b,c), 'a' is the functor, and 'b' and 'c' the arguments.) As with a PROLOG structure, the goal's arguments themselves may preferably be structures.

Various options can be included in the parameter list, to refine the semantics associated with the solvable. The type parameter is preferably used to say whether the solvable is data or procedure. When the type is procedure, another parameter may be used to indicate the handler to be associated with the solvable. Some of the parameters appropriate for a data solvable are mentioned elsewhere in this application. In either case (procedure or data solvable), the private parameter may be preferably used to restrict the use of a solvable to the declaring agent when the agent intends the solvable to be solely for its internal use but wishes to take advantage of the mechanisms in accordance with the present invention to access it, or when the agent wants the solvable to be available to outside agents only at selected times. In support of the latter case, it is preferable for the agent to change the status of a solvable from private to non-private at any time.

The permissions of a solvable provide mechanisms by which an agent may preferably control access to its services allowing the agent to restrict calling and writing of a solvable to itself and/or other selected agents. (Calling means requesting the service encapsulated by a solvable, whereas Writing means modifying the collection of facts associated with a data solvable.) The default permission for every solvable in a further preferred embodiment of the present invention is to be callable by anyone, and for data solvables to be writable by anyone. A solvable's permissions can preferably be changed at any time, by the agent providing the solvable.

For example, the solvables of a simple email agent might include:

```
solvable(send_message(email, +ToPerson, +Params),
  [type(procedure), callback(send_mail)],
    [ ]
solvable(last_message(email, −MessageId),
  [type(data), single_value(true)],
  [write(true)]),
solvable (get_message (email, +MessageId, −Msg),
  [type(procedure), callback(get_mail)], [ ])
```

The symbols '+' and '−', indicating input and output arguments, are at present used only for purposes of documentation. Most parameters and permissions have default values, and specifications of default values may be omitted from the parameters and permissions lists.

Defining an agent's capabilities in terms of solvable declarations effectively creates a vocabulary with which other agents can communicate with the new agent. Ensuring that agents will speak the same language and share a common, unambiguous semantics of the vocabulary involves ontology. Agent development tools and services (automatic translations of solvables by the facilitator) help address this issue; additionally, a preferred embodiment of the present invention will typically rely on vocabulary from either formally engineered ontologies for specific domains or from ontologies constructed during the incremental development of a body of agents for several applications or from both specific domain ontologies and incrementally developed ontologies. Several example tools and services are described in Cheyer et al.'s paper entitled "Development Tools for the Open Agent Architecture," as presented at the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 96), London, April 1996.

Although the present invention imposes no hard restrictions on the form of solvable declarations, two common usage conventions illustrate some of the utility associated with solvables.

Classes of services are often preferably tagged by a particular type. For instance, in the example above, the "last_message" and "get_message" solvables are specialized for email, not by modifying the names of the services, but rather by the use of the 'email' parameter, which serves during the execution of an ICL request to select (or not) a specific type of message.

Actions are generally written using an imperative verb as the functor of the solvable in a preferred embodiment of the present invention, the direct object (or item class) as the first argument of the predicate, required arguments following, and then an extensible parameter list as the last argument. The parameter list can hold optional information usable by the function. The ICL expression generated by a natural language parser often makes use of this parameter list to store prepositional phrases and adjectives.

As an illustration of the above two points, "Send mail to Bob about lunch" will be translated into an ICL request send message(email, 'Bob Jones', [subject(lunch)]), whereas "Remind Bob about lunch" would leave the transport unspecified (send_message(KIND, 'Bob Jones', [subject (lunch)])), enabling an available message transfer agents (e.g., fax, phone, mail, pager) to compete for the opportunity to carry out the request.

Requesting Services

An agent preferably requests services of the community of agent by delegating tasks or goals to its facilitator. Each request preferably contains calls to one or more agent solvables, and optionally specifies parameters containing advice to help the facilitator determine how to execute the task. Calling a solvable preferably does not require that the agent specify (or even know of) a particular agent or agents to handle the call. While it is possible to specify one or more agents using an address parameter (and there are situations in which this is desirable), in general it is advantageous to leave this delegation to the facilitator. This greatly reduces the hard-coded component dependencies often found in other distributed frameworks. The agent libraries of a preferred embodiment of the present invention provide an agent with a single, unified point of entry for requesting services of other agents: the library procedure oaa_Solve. In the style of logic programming, oaa_Solve may preferably be used both to retrieve data and to initiate actions, so that calling a data solvable looks the same as calling a procedure solvable.

Complex Goal Expressions

A powerful feature provided by preferred embodiments of the present invention is the ability of a client agent (or a user) to submit compound goals of an arbitrarily complex nature to a facilitator. A compound goal is a single goal expression that specifies multiple sub-goals to be performed. In speaking of a "complex goal expression" we mean that a single goal expression that expresses multiple sub-goals can potentially include more than one type of logical connector (e.g., AND, OR, NOT), and/or more than one level of logical nesting (e.g., use of parentheses), or the substantive equivalent. By way of further clarification, we note that when speaking of an "arbitrarily complex goal expression" we mean that goals are expressed in a language or syntax that allows expression of such complex goals when appropriate or when desired, not that every goal is itself necessarily complex.

It is contemplated that this ability is provided through an interagent communication language having the necessary syntax and semantics. In one example, the goals may take the form of compound goal expressions composed using operators similar to those employed by PROLOG, that is, the comma for conjunction, the semicolon for disjunction, the arrow for conditional execution, etc. The present invention also contemplates significant extensions to PROLOG syntax and semantics. For example, one embodiment incorporates a "parallel disjunction" operator indicating that the disjuncts are to be executed by different agents concurrently. A further embodiment supports the specification of whether a given sub-goal is to be executed breadth-first or depth-first.

A further embodiment supports each sub-goal of a compound goal optionally having an address and/or a set of parameters attached to it. Thus, each sub-goal takes the form

Address:Goal::Parameters

where both Address and Parameters are optional.

An address, if present, preferably specifies one or more agents to handle the given goal, and may employ several different types of referring expression: unique names, symbolic names, and shorthand names. Every agent has preferably a unique name, assigned by its facilitator, which relies upon network addressing schemes to ensure its global uniqueness. Preferably, agents also have self-selected symbolic names (for example, "mail"), which are not guaranteed to be unique. When an address includes a symbolic name, the facilitator preferably takes this to mean that all agents having that name should be called upon. Shorthand names include 'self' and 'parent' (which refers to the agent's facilitator). The address associated with a goal or sub-goal is preferably always optional. When an address is not present, it is the facilitator's job to supply an appropriate address.

The distributed execution of compound goals becomes particularly powerful when used in conjunction with natural language or speech-enabled interfaces, as the query itself may specify how functionality from distinct agents will be combined. As a simple example, the spoken utterance "Fax it to Bill Smith's manager." can be translated into the following compound ICL request:

oaa_Solve((manager('Bill Smith', M), fax(it,M,[ ])), [strategy(action)])

Note that in this ICL request there are two sub-goals, "manager('Bill Smith',M)" and "fax(it,M,[ ])," and a single global parameter "strategy(action)." According to the present invention, the facilitator is capable of mapping global parameters in order to apply the constraints or advice across the separate sub-goals in a meaningful way. In this instance, the global parameter strategy(action) implies a parallel constraint upon the first sub-goal; i.e., when there are multiple agents that can respond to the manager sub-goal, each agent should receive a request for service. In contrast, for the second sub-goal, parallelism should not be inferred from the global parameter strategy(action) because such an inference would possibly result in the transmission of duplicate facsimiles.

Refining Service Requests

In a preferred embodiment of the present invention, parameters associated with a goal (or sub-goal) can draw on useful features to refine the request's meaning. For example, it is frequently preferred to be able to specify whether or not solutions are to be returned synchronously; this is done using the reply parameter, which can take any of the values synchronous, asynchronous, or none. As another example, when the goal is a non-compound query of a data solvable, the cache parameter may preferably be used to request local caching of the facts associated with that solvable.

Many of the remaining parameters fall into two categories: feedback and advice. Feedback parameters allow a service requester to receive information from the facilitator about how a goal was handled. This feedback can include such things as the identities of the agents involved in satisfying the goal, and the amount of time expended in the satisfaction of the goal.

Advice parameters preferably give constraints or guidance to the facilitator in completing and interpreting the goal. For example, a solution_limit parameter preferably allows the requester to say how many solutions it is interested in; the facilitator and/or service providers are free to use this information in optimizing their efforts. Similarly, a time_limit is preferably used to say how long the requester is willing to wait for solutions to its request, and, in a multiple facilitator system, a level_limit may preferably be used to say how remote the facilitators may be that are consulted in the search for solutions. A priority parameter is preferably used to indicate that a request is more urgent than previous requests that have not yet been satisfied. Other preferred advice parameters include but are not limited to parameters used to tell the facilitator whether parallel satisfaction of the parts of a goal is appropriate, how to combine and filter results arriving from multiple solver agents, and whether the requester itself may be considered a candidate solver of the sub-goals of a request.

Advice parameters preferably provide an extensible set of low-level, orthogonal parameters capable of combining with the ICL goal language to fully express how information should flow among participants. In certain preferred embodiments of the present invention, multiple parameters can be grouped together and given a group name. The resulting high-level advice parameters can preferably be used to express concepts analogous to KQML's performatives, as well as define classifications of problem types. For instance, KQML's "ask_all" and "ask_one" performatives would be represented as combinations of values given to the parameters reply, parallel ok, and solution_limit. As an example of a higher-level problem type, the strategy "math_problem" might preferably send the query to all appropriate math solvers in parallel, collect their responses, and signal a conflict if different answers are returned. The strategy "essay_question" might preferably send the request to all appropriate participants, and signal a problem (i.e., cheating) if any of the returned answers are identical.

Facilitation

In a preferred embodiment of the present invention, when a facilitator receives a compound goal, its job is to construct a goal satisfaction plan and oversee its satisfaction in an optimal or near optimal manner that is consistent with the specified advice. The facilitator of the present invention maintains a knowledge base that records the capabilities of a collection of agents, and uses that knowledge to assist requesters and providers of services in making contact.

FIG. **7** schematically shows data structures **700** internal to a facilitator in accordance with one embodiment of the present invention. Consider the function of a Agent Registry **702** in the present invention. Each registered agent may be seen as associated with a collection of fields found within its parent facilitator such as shown in the figure. Each registered agent may optionally possess a Symbolic Name which would be entered into field **704**. As mentioned elsewhere, Symbolic Names need not be unique to each instance of an agent. Note that an agent may in certain preferred embodiments of the present invention possess more than one Symbolic Name. Such Symbolic Names would each be found through their associations in the Agent Registry entries. Each agent, when registered, must possess a Unique Address, which is entered into the Unique Address field **706**.

With further reference to FIG. **7**, each registered agent may be optionally associated with one or more capabilities, which have associated Capability Declaration fields **708** in the parent facilitator Agent Registry **702**. These capabilities may define not just functionality, but may further provide a utility parameter indicating, in some manner (e.g., speed, accuracy, etc), how effective the agent is at providing the declared capability. Each registered agent may be optionally associated with one or more data components, which have associated Data Declaration fields **710** in the parent facilitator Agent Registry **702**. Each registered agent may be optionally associated with one or more triggers, which preferably could be referenced through their associated Trigger Declaration fields **712** in the parent facilitator Agent Registry **702**. Each registered agent may be optionally associated with one or more tasks, which preferably could be referenced through their associated Task Declaration fields **714** in the parent facilitator Agent Registry **702**. Each registered agent may be optionally associated with one or more Process Characteristics, which preferably could be referenced through their associated Process Characteristics Declaration fields **716** in the parent facilitator Agent Registry **702**. Note that these characteristics in certain preferred embodiments of the present invention may include one or more of the following: Machine Type (specifying what type of computer may run the agent), Language (both computer and human interface).

A facilitator agent in certain preferred embodiments of the present invention further includes a Global Persistent Database **720**. The database **720** is composed of data elements which do not rely upon the invocation or instantiation of client agents for those data elements to persist. Examples of data elements which might be present in such a database include but are not limited to the network address of the facilitator agent's server, facilitator agent's server accessible network port list, firewalls, user lists, and security options regarding the access of server resources accessible to the facilitator agent.

A simplified walk through of operations involved in creating a client agent, a client agent initiating a service request, a client agent responding to a service request and a facilitator agent responding to a service request are including hereafter by way of illustrating the use of such a system. These figures and their accompanying discussion are provided by way of illustration of one preferred embodiment of the present invention and are not intended to limit the scope of the present invention.

FIG. **8** depicts operations involved in instantiating a client agent with its parent facilitator in accordance with a preferred embodiment of the present invention. The operations begin with starting the Agent Registration in a step **800**. In a next step **802**, the Installer, such as a client or facilitator agent, invokes a new client agent. It will be appreciated that any computer entity is capable of invoking a new agent. The system then instantiates the new client agent in a step **804**. This operation may involve resource allocations somewhere in the network on a local computer system for the client agent, which will often include memory as well as placement of references to the newly instantiated client agent in internal system lists of agents within that local computing system. Once instantiated, the new client and its parent facilitator establish a communications link in a step **806**. In certain preferred embodiments, this communications link involves selection of one or more physical transport mechanisms for this communication. Once established, the client agent transmits it profile to the parent facilitator in a step **808**. When received, the parent facilitator registers the client agent in a step **810**. Then, at a step **812**, a client agent has been instantiated in accordance with one preferred embodiment of the present invention.

FIG. **9** depicts operations involved in a client agent initiating a service request and receiving the response to that service request in accordance with a preferred embodiment of the present invention. The method of FIG. **9** begins in a step **900**, wherein any initialization or other such procedures may be performed. Then, in a step **902**, the client agent determines a goal to be achieved (or solved). This goal is then translated in a step **904** into ICL, if it is not already formulated in it. The goal, now stated in ICL, is then transmitted to the client agent's parent facilitator in a step **906**. The parent facilitator responds to this service request and at a later time, the client agent receives the results of the request in a step **908**, operations of FIG. **9** being complete in a done step **910**.

FIG. **10** depicts operations involved in a client agent responding to a service request in accordance with a preferred embodiment of the present invention. Once started in a step **1000**, the client agent receives the service request in a step **1002**. In a next step **1004**, the client agent parses the received request from ICL. The client agent then determines if the service is available in a step **1006**. If it is not, the client agent returns a status report to that effect in a step **1008**. If the service is available, control is passed to a step **1010** where the client performs the requested service. Note that in completing step **1010** the client may form complex goal expressions, requesting results for these solvables from the facilitator agent. For example, a fax agent might fax a document to a certain person only after requesting and receiving a fax number for that person. Subsequently, the client agent either returns the results of the service and/or a status report in a step **1012**. The operations of FIG. **10** are complete in a done step **1014**.

FIG. **11** depicts operations involved in a facilitator agent response to a service request in accordance with a preferred embodiment of the present invention. The start of such operations in step **1100** leads to the reception of a goal request in a step **1102** by the facilitator. This request is then parsed and interpreted by the facilitator in a step **1104**. The facilitator then proceeds to construct a goal satisfaction plan in a next step **1106**. In steps **1108** and **1110**, respectively, the facilitator determines the required sub-goals and then selects agents suitable for performing the required sub-goals. The facilitator then transmits the sub-goal requests to the selected agents in a step **1112** and receives the results of these transmitted requests in a step **1114**. It should be noted that the actual implementation of steps **1112** and **1114** are dependent upon the specific goal satisfaction plan. For instance, certain sub-goals may be sent to separate agents in parallel, while transmission of other sub-goals may be postponed until receipt of particular answers. Further, certain requests may generate multiple responses that generate additional sub-goals. Once the responses have been received, the facilitator determines whether the original requested goal has been completed in a step **1118**. If the original requested goal has not been completed, the facilitator recursively repeats the operations **1106** through **1116**. Once the original requested goal is completed, the facilitator returns the results to the requesting agent **1118** and the operations are done at **1120**.

A further preferred embodiment of the present invention incorporates transparent delegation, which means that a requesting agent can generate a request, and a facilitator can manage the satisfaction of that request, without the requester needing to have any knowledge of the identities or locations of the satisfying agents. In some cases, such as when the request is a data query, the requesting agent may also be oblivious to the number of agents involved in satisfying a request. Transparent delegation is possible because agents' capabilities (solvables) are treated as an abstract description of a service, rather than as an entry point into a library or body of code.

A further preferred embodiment of the present invention incorporates facilitator handling of compound goals, preferably involving three types of processing: delegation, optimization and interpretation.

Delegation processing preferably supports facilitator determination of which specific agents will execute a compound goal and how such a compound goal's sub-goals will be combined and the sub-goal results routed. Delegation involves selective application of global and local constraint and advice parameters onto the specific sub-goals. Delegation results in a goal that is unambiguous as to its meaning and as to the agents that will participate in satisfying it.

Optimization processing of the completed goal preferably includes the facilitator using sub-goal parallelization where appropriate. Optimization results in a goal whose interpretation will require as few exchanges as possible, between the facilitator and the satisfying agents, and can exploit parallel efforts of the satisfying agents, wherever this does not affect the goal's meaning.

Interpretation processing of the optimized goal. Completing the addressing of a goal involves the selection of one or more agents to handle each of its sub-goals (that is, each sub-goal for which this selection has not been specified by the requester). In doing this, the facilitator uses its knowledge of the capabilities of its client agents (and possibly of other facilitators, in a multi-facilitator system). It may also use strategies or advice specified by the requester, as explained below. The interpretation of a goal involves the coordination of requests to the satisfying agents, and assembling their responses into a coherent whole, for return to the requester.

A further preferred embodiment of present invention extends facilitation so the facilitator can employ strategies and advice given by the requesting agent, resulting in a variety of interaction patterns that may be instantiated in the satisfaction of a request.

A further preferred embodiment of present invention handles the distribution of both data update requests and requests for installation of triggers, preferably using some of the same strategies that are employed in the delegation of service requests.

Note that the reliance on facilitation is not absolute; that is, there is no hard requirement that requests and services be matched up by the facilitator, or that interagent communications go through the facilitator. There is preferably support in the agent library for explicit addressing of requests. However, a preferred embodiment of the present invention encourages employment the paradigm of agent communities, minimizing their development effort, by taking advantage of the facilitator's provision of transparent delegation and handling of compound goals.

A facilitator is preferably viewed as a coordinator, not a controller, of cooperative task completion. A facilitator preferably never initiates an activity. A facilitator preferably responds to requests to manage the satisfaction of some goal, the update of some data repository, or the installation of a trigger by the appropriate agent or agents. All agents can preferably take advantage of the facilitator's expertise in delegation, and its up-to-date knowledge about the current membership of a dynamic community. The facilitator's coordination services often allows the developer to lessen the complexity of individual agents, resulting in a more manageable software development process, and enabling the creation of lightweight agents.

Maintaining Data Repositories

The agent library supports the creation, maintenance, and use of databases, in the form of data solvables. Creation of a data solvable requires only that it be declared. Querying a data solvable, as with access to any solvable, is done using oaa__Solve.

A data solvable is conceptually similar to a relation in a relational database. The facts associated with each solvable are maintained by the agent library, which also handles incoming messages containing queries of data solvables. The default behavior of an agent library in managing these facts may preferably be refined, using parameters specified with the solvable's declaration. For example, the parameter single__value preferably indicates that the solvable should only contain a single fact at any given point in time. The parameter unique__values preferably indicates that no duplicate values should be stored.

Other parameters preferably allow data solvables use of the concepts of ownership and persistence. For implementing shared repositories, it is often preferable to maintain a record of which agent created each fact of a data solvable with the creating agent being preferably considered the fact's owner. In many applications, it is preferable to remove an agent's facts when that agent goes offline (for instance, when the agent is no longer participating in the agent community, whether by deliberate termination or by malfunction). When a data solvable is declared to be non-persistent, its facts are automatically maintained in this way, whereas a persistent data solvable preferably retains its facts until they are explicitly removed.

A further preferred embodiment of present invention supports an agent library through procedures by which agents can update (add, remove, and replace) facts belonging to data solvables, either locally or on other agents, given that they have preferably the required permissions. These procedures may preferably be refined using many of the same parameters that apply to service requests. For example, the address parameter preferably specifies one or more particular agents to which the update request applies. In its absence, just as with service requests, the update request preferably goes to all agents providing the relevant data solvable. This default behavior can be used to maintain coordinated "mirror" copies of a data set within multiple agents, and can be useful in support of distributed, collaborative activities.

Similarly, the feedback parameters, described in connection with oaa__Solve, are preferably available for use with data maintenance requests.

A further preferred embodiment of present invention supports ability to provide data solvables not just to client agents, but also to facilitator agents. Data solvables can preferably created, maintained and used by a facilitator. The facilitator preferably can, at the request of a client of the facilitator, create, maintain and share the use of data solvables with all the facilitator's clients. This can be useful with relatively stable collections of agents, where the facilitator's workload is predictable.

Using a Blackboard Style of Communication

In a further preferred embodiment of present invention, when a data solvable is publicly readable and writable, it acts essentially as a global data repository and can be used cooperatively by a group of agents. In combination with the use of triggers, this allows the agents to organize their efforts around a "blackboard" style of communication.

As an example, the "DCG-NL" agent (one of several existing natural language processing agents), provides natural language processing services for a variety of its peer agents, expects those other agents to record, on the facilitator, the vocabulary to which they are prepared to respond, with an indication of each word's part of speech, and of the logical form (ICL sub-goal) that should result from the use of that word. In a further preferred embodiment of present invention, the NL agent, preferably when it comes online, preferably installs a data solvable for each basic part of speech on its facilitator. For instance, one such solvable would be:

solvable(noun(Meaning, Syntax), [ ], [ ])

Note that the empty lists for the solvable's permissions and parameters are acceptable here, since the default permissions and parameters provide appropriate functionality.

A further preferred embodiment of present invention incorporating an Office Assistant system as discussed herein or similar to the discussion here supports several agents making use of these or similar services. For instance, the database agent uses the following call, to library procedure oaa__AddData, to post the noun 'boss', and to indicate that the "meaning" of boss is the concept 'manager':

oaa__AddData(noun(manager, atom(boss)), [address (parent)])

Autonomous Monitoring with Triggers

A further preferred embodiment of present invention includes support for triggers, providing a general mechanism for requesting some action be taken when a set of conditions is met. Each agent can preferably install triggers either locally, for itself, or remotely, on its facilitator or peer agents. There are preferably at least four types of triggers: communication, data, task, and time. In addition to a type, each trigger preferably specifies at least a condition and an action, both preferably expressed in ICL. The condition indicates under what circumstances the trigger should fire, and the action indicates what should happen when it fires. In addition, each trigger can be set to fire either an unlimited number of times, or a specified number of times, which can be any positive integer.

Triggers can be used in a variety of ways within preferred embodiments of the present invention. For example, triggers can be used for monitoring external sensors in the execution environment, tracking the progress of complex tasks, or coordinating communications between agents that are essential for the synchronization of related tasks. The installation of a trigger within an agent can be thought of as a representation of that agent's commitment to carry out the specified action, whenever the specified condition holds true.

Communication triggers preferably allow any incoming or outgoing event (message) to be monitored. For instance, a simple communication trigger may say something like: "Whenever a solution to a goal is returned from the facilitator, send the result to the presentation manager to be displayed to the user."

Data triggers preferably monitor the state of a data repository (which can be maintained on a facilitator or a client agent). Data triggers' conditions may be tested upon the addition, removal, or replacement of a fact belonging to a data solvable. An example data trigger is: "When 15 users are simultaneously logged on to a machine, send an alert message to the system administrator."

Task triggers preferably contain conditions that are tested after the processing of each incoming event and whenever a timeout occurs in the event polling. These conditions may specify any goal executable by the local ICL interpreter, and most often are used to test when some solvable becomes satisfiable. Task triggers are useful in checking for task-specific internal conditions. Although many cases such conditions are captured by solvables, in other cases they may not be. For example, a mail agent might watch for new incoming mail, or an airline database agent may monitor which flights will arrive later than scheduled. An example task trigger is: "When mail arrives for me about security, notify me immediately."

Time triggers preferably monitor time conditions. For instance, an alarm trigger can be set to fire at a single fixed point in time (e.g., "On December 23rd at 3 pm"), or on a recurring basis (e.g., "Every three minutes from now until noon").

Triggers are preferably implemented as data solvables, declared implicitly for every agent. When requesting that a trigger be installed, an agent may use many of the same parameters that apply to service and data maintenance requests.

A further preferred embodiment of present invention incorporates semantic support, in contrast with most programming methodologies, of the agent on which the trigger is installed only having to know how to evaluate the conditional part of the trigger, not the consequence. When the trigger fires, the action is delegated to the facilitator for execution. Whereas many commercial mail programs allow rules of the form "When mail arrives about XXX, [forward it, delete it, archive it]", the possible actions are hard-coded and the user must select from a fixed set.

A further preferred embodiment of present invention, the consequence of a trigger may be any compound goal executable by the dynamic community of agents. Since new agents preferably define both functionality and vocabulary, when an unanticipated agent (for example, a fax agent) joins the community, no modifications to existing code is required for a user to make use of it—"When mail arrives, fax it to Bill Smith."

The Agent Library

In a preferred embodiment of present invention, the agent library provides the infrastructure for constructing an agent-based system. The essential elements of protocol (involving the details of the messages that encapsulate a service request and its response) are preferably made transparent to simplify the programming applications. This enables the developer to focus functionality, rather than message construction details and communication details. For example, to request a service of another agent, an agent preferably calls the library procedure oaa__Solve. This call results in a message to a facilitator, which will exchange messages with one or more service providers, and then send a message containing the desired results to the requesting agent. These results are returned via one of the arguments of oaa__Solve. None of the messages involved in this scenario is explicitly constructed by the agent developer. Note that this describes the synchronous use of oaa__Solve.

In another preferred embodiment of present invention, an agent library provides both intraagent and interagent infrastructure; that is, mechanisms supporting the internal structure of individual agents, on the one hand, and mechanisms of cooperative interoperation between agents, on the other. Note that most of the infrastructure cuts across this boundary with many of the same mechanisms supporting both agent internals and agent interactions in an integrated fashion. For example, services provided by an agent preferably can be accessed by that agent through the same procedure (oaa__Solve) that it would employ to request a service of another agent (the only difference being in the address parameter accompanying the request). This helps the developer to reuse code and avoid redundant entry points into the same functionality.

Both of the preferred characteristics described above (transparent construction of messages and integration of intraagent with interagent mechanisms) apply to most other library functionality as well, including but not limited to data management and temporal control mechanisms.

Source Code Appendix

Source code for version 2.0 of the OAA software product is included as an appendix hereto, and is incorporated herein by reference. The code includes an agent library, which provides infrastructure for constructing an agent-based system. The library's several families of procedures provide the functionalities discussed above, as well as others that have not been discussed here but that will be sufficiently clear to the interested practitioner. For example, declarations of an agent's solvables, and their registration with a facilitator, are managed using procedures such as oaa__Declare, oaa__Undeclare, and oaa__Redeclare. Updates to data solvables can be accomplished with a family of procedures including oaa__AddData, oaa__RemoveData, and oaa__ReplaceData. Similarly, triggers are maintained using procedures such as oaa__AddTrigger, oaa__RemoveTrigger, and oaa__ReplaceTrigger. The provided source code also includes source code for an OAA Facilitator Agent.

The source code appendix is offered solely as a means of further helping practitioners to construct a preferred embodiment of the invention. By no means is the source code intended to limit the scope of the present invention.

Illustrative Applications

To further illustrate the technology of the preferred embodiment, we will next present and discuss two sample applications of the present inventions.

Unified Messaging

A further preferred embodiment of present invention incorporates a Unified Messaging application extending the Automated Office application presented previously herein with an emphasis on ubiquitous access and dynamic presentation of the information and services supported by the agent community. The agents used in this application are depicted in FIG. **12**.

A hypothetical example of realistic dialog using a preferred embodiment of the present invention can provide insight into how systems may preferably be built using the present invention. In this scenario, the user, with only a telephone as an interface, is planning a trip to Boston where he will soon give a presentation. Capitalized sentences are phrases spoken by the user into the telephone and processed by a phone agent **452**.

Responses, unless otherwise indicated, are spoken by the system using text-to-speech generation agent **454**.

1.1 Welcome to SRI International. Please enter your user ID and password.

<User enters touchtone ID and password>

Good to see you again Adam Cheyer. I am listening to you.

Every user interface agent **408**, including the telephone agent **452**, should know the identify of its user. This information is used in resolving anaphoric references such as "Me" and "I", and allows multiple user interfaces operated by the same user to work together.

1.2 WHAT IS TODAY'S SCHEDULE?

Here is today's schedule for Adam Cheyer:

At 10 am for 1 hour, meeting with Dave.

At 3 pm for 1 hour, presentation about software agents. End of schedule.

If the user is operating both a graphical user interface and a telephone, as described in conjunction with the Automated Office application, the result of this spoken request is to display a calendar window containing the current schedule. In this case, with no graphical display available, the GEN__NL agent **1202** is tasked to produce a spoken response that can be played over the phone. GEN__NL shares the same dynamic vocabulary and phrasal rules as the natural language parser DCG__NL **426**, and contains strategies for producing responses to queries using either simple or list-based multimedia utterances.

1.3 FIND FRIDAY'S WEATHER IN BOSTON.

The weather in Boston for Friday is as follows:

Sunny in the morning. Partly cloudy in the afternoon with a 20

percent chance of thunderstorms late. Highs in the mid 70s.

In addition to data accessible from legacy applications, content may be retrieved by web-reading agents which provide wrappers around useful websites.

1.4 FIND ALL NEW MAIL MESSAGES.

There are 2 messages available.

Message 1, from Mark Tierny, entitled "OAA meeting."

1.5 NEXT MESSAGE

Message 2, from Jennifer Schwefler, entitled "Presentation Summary."

1.6 PLAY IT.

This message is a multipart MIME-encoded message. There are two parts.

Part **1**. (Voicemail message, not text-to speech):

Thanks for taking part as a speaker in our conference.

The schedule will be posted soon on our homepage.

1.7 NEXT PART

Part **2**. (read using text-to-speech):

The presentation home page is http://www . . .

1.8 PRINT MESSAGE

Command executed.

Mail messages are no longer just simple text documents, but often consist of multiple subparts containing audio files, pictures, webpages, attachments and so forth. When a user asks to play a complex email message over the telephone, many different agents may be implicated in the translation process, which would be quite different given the request "print it." The challenge is to develop a system which will enable agents to cooperate in an extensible, flexible manner that alleviates explicit coding of agent interactions for every possible input/output combination.

In a preferred embodiment of the present invention, each agent concentrates only on what it can do and on what it knows, and leaves other work to be delegated to the agent community. For instance, a printer agent **1204**, defining the solvable print(Object,Parameters), can be defined by the following pseudo-code, which basically says, "If someone can get me a document, in either POSTSCRIPT or text form, I can print it.".

```
print(Object, Parameters) {
    ' If Object is reference to "it", find an appropriate document
    if (Object ="ref(it)")
        oaa__Solve(resolve__reference(the, document, Params, Object),[ ]);
    ' Given a reference to some document, ask for the document in POSTSCRIPT
    if (Object ="id(Pointer)")
        oaa__Solve(resolve__id__as(id(Pointer), postscript, [ ], Object), [ ]);
    ' If Object is of type text or POSTSCRIPT, we can print it.
    if ((Object is of type Text) or (Object is of type Postscript))
        do print (Object);
}
```

In the above example, since an email message is the salient document, the mail agent **442** will receive a request to produce the message as POSTSCRIPT. Whereas the mail agent **442** may know how to save a text message as POSTSCRIPT, it will not know what to do with a webpage or voicemail message. For these parts of the message, it will simply send oaa_Solve requests to see if another agent knows how to accomplish the task.

Until now, the user has been using only a telephone as user interface. Now, he moves to his desktop, starts a web browser **436**, and accesses the URL referenced by the mail message.

1.9 RECORD MESSAGE

Recording voice message. Start speaking now.

1.10 THIS IS THE UPDATED WEB PAGE CONTAINING THE PRESENTATION SCHEDULE.

Message one recorded.

1.11 IF THIS WEB PAGE CHANGES, GET IT TO ME WITH NOTE ONE.

Trigger added as requested.

In this example, a local agent **436** which interfaces with the web browser can return the current page as a solution to the request "oaa_Solve(resolve_reference(this, web_page, [ ], Ref),[ ])", sent by the NL agent **426**. A trigger is installed on a web agent **436** to monitor changes to the page, and when the page is updated, the notify agent **446** can find the user and transmit the webpage and voicemail message using the most appropriate media transfer mechanism.

This example based on the Unified Messaging application is intended to show how concepts in accordance with the present invention can be used to produce a simple yet extensible solution to a multi-agent problem that would be difficult to implement using a more rigid framework. The application supports adaptable presentation for queries across dynamically changing, complex information; shared context and reference resolution among applications; and flexible translation of multimedia data. In the next section, we will present an application which highlights the use of parallel competition and cooperation among agents during multi-modal fusion.

Multimodal Map

A further preferred embodiment of present invention incorporates the Multimodal Map application. This application demonstrates natural ways of communicating with a community of agents, providing an interactive interface on which the user may draw, write or speak. In a travel-planning domain illustrated by FIG. **13**, available information includes hotel, restaurant, and tourist-site data retrieved by distributed software agents from commercial Internet sites. Some preferred types of user interactions and multimodal issues handled by the application are illustrated by a brief scenario featuring working examples taken from the current system.

Sara is planning a business trip to San Francisco, but would like to schedule some activities for the weekend while she is there. She turns on her laptop PC, executes a map application, and selects San Francisco.

2.1 [Speaking] Where is downtown?

Map scrolls to appropriate area.

2.2 [Speaking and drawing region] Show me all hotels near here.

Icons representing hotels appear.

2.3 [Writes on a hotel] Info?

A textual description (price, attributes, etc.) appears.

2.4 [Speaking] I only want hotels with a pool. Some hotels disappear.

2.5 [Draws a crosscut on a hotel that is too close to a highway)

Hotel disappears

2.6 [Speaking and circling] Show me a photo of this hotel.

Photo appears.

2.7 (Points to another hotel]

Photo appears.

2.8 [Speaking] Price of the other hotel?

Price appears for previous hotel.

2.9 [Speaking and drawing an arrow] Scroll down.

Display adjusted.

2.10 [Speaking and drawing an arrow toward a hotel]

What is the distance from this hotel to Fisherman's Wharf?

Distance displayed.

2.11 [Pointing to another place and speaking] And the distance to here?

Distance displayed.

Sara decides she could use some human advice. She picks up the phone, calls Bob, her travel agent, and writes Start collaboration to synchronize his display with hers. At this point, both are presented with identical maps, and the input and actions of one will be remotely seen by the other.

3.1 [Sara speaks and circles two hotels]

Bob, I'm trying to choose between these two hotels. Any opinions?

3.2 [Bob draws an arrow, speaks, and points]

Well, this area is really nice to visit. You can walk there from

this hotel.

Map scrolls to indicated area. Hotel selected.

3.3 [Sara speaks] Do you think I should visit Alcatraz?

3.4 [Bob speaks] Map, show video of Alcatraz.

Video appears.

3.5 [Bob speaks] Yes, Alcatraz is a lot of fun.

A further preferred embodiment of present invention generates the most appropriate interpretation for the incoming streams of multimodal input. Besides providing a user interface to a dynamic set of distributed agents, the application is preferably built using an agent framework. The present invention also contemplates aiding the coordinate competition and cooperation among information sources, which in turn works in parallel to resolve the ambiguities arising at every level of the interpretation process: low-level processing of the data stream, anaphora resolution, cross-modality influences and addressee.

Low-level processing of the data stream: Pen input may be preferably interpreted as a gesture (e.g., 2.5: cross-out) by one algorithm, or as handwriting by a separate recognition process (e.g., 2.3: "info?"). Multiple hypotheses may preferably be returned by a modality recognition component.

Anaphora resolution: When resolving anaphoric references, separate information sources may contribute to resolving the reference: context by object type, deictic, visual context, database queries, discourse analysis. An example of information provided through context by object type is found in interpreting an utterance such as "show photo of the hotel", where the natural language component can return a list of the last hotels talked about. Deictic information in combination with a spoken utterance like "show photo of this hotel" may preferably include pointing, circling, or arrow gestures which might indicate the desired object (e.g., 2.7). Deictic references may preferably occur before, during, or after an accompanying verbal command. Information provided in a visual context, given for the request "display photo of the hotel" may preferably include the user interface agent might determine that only one hotel is currently visible on the map, and therefore this might be the desired reference object. Database queries preferably involving information from a database agent combined with results from other resolution strategies. Examples are "show me a photo of the hotel in Menlo Park" and 2.2. Discourse analysis preferably provides a source of information for phrases such as "No, the other one" (or 2.8).

The above list of preferred anaphora resolution mechanisms is not exhaustive. Examples of other preferred resolution methods include but are not limited to spatial reasoning ("the hotel between Fisherman's Wharf and Lombard Street") and user preferences ("near my favorite restaurant").

Cross-modality influences: When multiple modalities are used together, one modality may preferably reinforce or remove or diminish ambiguity from the interpretation of another. For instance, the interpretation of an arrow gesture may vary when accompanied by different verbal commands (e.g., "scroll left" vs. "show info about this hotel"). In the latter example, the system must take into account how accurately and unambiguously an arrow selects a single hotel.

Addressee: With the addition of collaboration technology, humans and automated agents all share the same workspace. A pen doodle or a spoken utterance may be meant for either another human, the system (3.1), or both (3.2).

The implementation of the Multimodal Map application illustrates and exploits several preferred features of the present invention: reference resolution and task delegation by parallel parameters of oaa_Solve, basic multi-user collaboration handled through built-in data management services, additional functionality readily achieved by adding new agents to the community, domain-specific code cleanly separated from other agents.

A further preferred embodiment of present invention provides reference resolution and task delegation handled in a distributed fashion by the parallel parameters of oaa_ Solve, with meta-agents encoding rules to help the facilitator make context- or user-specific decisions about priorities among knowledge sources.

A further preferred embodiment of present invention provides basic multi-user collaboration handled through at least one built-in data management service. The map user interface preferably publishes data solvables for elements such as icons, screen position, and viewers, and preferably defines these elements to have the attribute "shareable". For every update to this public data, the changes are preferably automatically replicated to all members of the collaborative session, with associated callbacks producing the visible effect of the data change (e.g., adding or removing an icon).

Functionality for recording and playback of a session is preferably implemented by adding agents as members of the collaborative community. These agents either record the data changes to disk, or read a log file and replicate the changes in the shared environment.

The domain-specific code for interpreting travel planning dialog is preferably separated from the speech, natural language, pen recognition, database and map user interface agents. These components were preferably reused without modification to add multimodal map capabilities to other applications for activities such as crisis management, multi-robot control, and the MVIEWS tools for the video analyst.

Improved Scalability and Fault Tolerance

Implementations of a preferred embodiment of present invention which rely upon simple, single facilitator architectures may face certain limitations with respect to scalability, because the single facilitator may become a communications bottleneck and may also represent a single, critical point for system failure.

Multiple facilitator systems as disclosed in the preferred embodiments to this point can be used to construct peer-to-peer agent networks as illustrated in FIG. **14**. While such embodiments are scalable, they do possess the potential for communication bottlenecks as discussed in the previous paragraph and they further possess the potential for reliability problems as central, critical points of vulnerability to systems failure.

A further embodiment of present invention supports a facilitator implemented as an agent like any other, whereby multiple facilitator network topologies can be readily constructed. One example configuration (but not the only possibility) is a hierarchical topology as depicted in FIG. **15**, where a top level Facilitator manages collections of both client agents **1508** and other Facilitators, **1504** and **1506**. Facilitator agents could be installed for individual users, for a group of users, or as appropriate for the task.

Note further, that network work topologies of facilitators can be seen as graphs where each node corresponds to an instance of a facilitator and each edge connecting two or more nodes corresponds to a transmission path across one or more physical transport mechanisms. Some nodes may represent facilitators and some nodes may represent clients. Each node can be further annotated with attributes corresponding to include triggers, data, capabilities but not limited to these attributes.

A further embodiment of present invention provides enhanced scalability and robustness by separating the planning and execution components of the facilitator. In contrast with the centralized facilitation schemes described above, the facilitator system **1600** of FIG. **16** separates the registry/planning component from the execution component. As a result, no single facilitator agent must carry all communications nor does the failure of a single facilitator agent shut down the entire system.

Turning directly to FIG. **16**, the facilitator system **1600** includes a registry/planner **1602** and a plurality of client agents **1612–1616**. The registry/planner **1604** is typically replicated in one or more locations accessible by the client agents. Thus if the registry/planner **1604** becomes unavailable, the client agents can access the replicated registry/planner(s).

This system operates, for example, as follows. An agent transmits a goal **1610** to the registry planner **1602**. The registry/planner **1604** translates the goal into an unambiguous execution plan detailing how to accomplish any sub-goals developed from the compound goal, as well as specifying the agents selected for performing the sub-goals. This execution plan is provided to the requesting agent which in turn initiates peer-to-peer interactions **1618** in order to implement the detailed execution plan, routing and combining information as specified within the execution plan. Communication is distributed thus decreasing sensitivity of the system to bandwidth limitations of a single facilitator agent. Execution state is likewise distributed thus enabling system operation even when a facilitator agent fails.

Further embodiments of present invention incorporate into the facilitator functionality such as load-balancing, resource management, and dynamic configuration of agent locations and numbers, using (for example) any of the topologies discussed. Other embodiments incorporate into a facilitator the ability to aid agents in establishing peer-to-peer communications. That is, for tasks requiring a sequence of exchanges between two agents, the facilitator assists the agents in finding one another and establishing communication, stepping out of the way while the agents communicate peer-to-peer over a direct, perhaps dedicated channel.

Further preferred embodiments of the present invention incorporate mechanisms for basic transaction management, such as periodically saving the state of agents (both facilitator and client) and rolling back to the latest saved state in the event of the failure of an agent.

What is claimed is:

1. A computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of:

registering a description of each active client agent's functional capabilities as corresponding registered functional capabilities, using an expandable, platform-independent, inter-agent language, wherein the inter-agent language includes:

a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events;

a content layer comprising one or more of goals, triggers and data elements associated with the events;

receiving a request for service as a base goal in the inter-agent language, in the form of an arbitrarily complex goal expression; and

dynamically interpreting the arbitrarily complex goal expression, said act of interpreting further comprising:

generating one or more sub-goals expressed in the inter-agent language;

constructing a goal satisfaction plan wherein the goal satisfaction plan includes:

a suitable delegation of sub-goal requests to best complete the requested service request—by using reasoning that includes one or more of domain-independent coordination strategies, domain-specific reasoning, and application-specific reasoning comprising rules and learning algorithms; and

dispatching each of the sub-goals to a selected client agent for performance, based on a match between the sub-goal being dispatched and the registered functional capabilities of the selected client agent.

2. A computer-implemented method as recited in claim **1**, further including the following acts of:

receiving a new request for service as a base goal using the inter-agent language, in the form of another arbitrarily complex goal expression, from at least one of the selected client agents in response to the sub-goal dispatched to said agent; and

recursively applying the step of dynamically interpreting the arbitrarily complex goal expression in order to perform the new request for service.

3. A computer-implemented method as recited in claim **2** wherein the act of registering a specific agent further includes:

invoking the specific agent in order to activate the specific agent;

instantiating an instance of the specific agent; and

transmitting the new agent profile from the specific agent to a facilitator agent in response to the instantiation of the specific agent.

4. A computer-implemented method as recited in claim **1** further including the act of deactivating a specific client agent no longer available to provide services by deleting the registration of the specific client agent.

5. A computer-implemented method as recited in claim **1** further comprising the act of providing an agent registry data structure.

6. A computer-implemented method as recited in claim **5** wherein the agent registry data structure includes at least one symbolic name for each active agent.

7. A computer-implemented method of recited in claim **5** wherein the agent registry data structure includes at least one data declaration for each active agent.

8. A computer-implemented method as recited in claim **5** wherein the agent registry data structure includes at least one trigger declaration for one active agent.

9. A computer-implemented method as recited in claim **5** wherein the agent registry data structure includes at least one task declaration, and process characteristics for each active agent.

10. A computer-implemented method as recited in claim **5** wherein the agent registry data structure includes at least one process characteristic for each active agent.

11. A computer-implemented method as recited in claim **1** further comprising the act of establishing communication between the plurality of distributed agents.

12. A computer-implemented method as recited in claim **1** further comprising the acts of:

receiving a request for service in a second language differing from the inter-agent language;

selecting a registered agent capable of converting the second language into the inter-agent language; and

forwarding the request for service in a second language to the registered agent capable of converting the second language into the inter-agent language, implicitly requesting that such a conversion be performed and the results returned.

13. A computer-implemented method as recited in claim **12** wherein the request include a natural language query, and the second registered agent capable of converting the second language into the inter-agent language service is a natural language agent.

14. A computer-implemented method as recited in claim **13** wherein the natural language query was generated by a user interface agent.

15. A computer-implemented method as recited in claim **1**, wherein the base goal requires setting a trigger having conditional functionality and consequential functionality.

16. A compute-implemented method as recited in claim **15** wherein the trigger is an outgoing communications trigger, the computer implemented method further including the acts of:

monitoring all outgoing communication events in order to determine whether a specific outgoing communication event has occurred; and

in response to the occurrence of the specific outgoing communication event, performing the particular action defined by the trigger.

17. A computer-implemented method as recited in claim **15** wherein the trigger is an incoming communications trigger, the computer implemented method further including the acts of:

monitoring all incoming communication events in order to determine whether a specific incoming communication event has occurred; and

in response to the occurrence of a specific incoming communication event satisfying the trigger conditional functionality, performing the particular consequential functionality defined by the trigger.

18. A computer-implemented method as recited in claim **15** wherein the trigger is a data trigger, the computer implemented method further including the acts of:

monitoring a state of a data repository; and

in response to a particular state event satisfying the trigger conditional functionality, performing the particular consequential functionality defined by the trigger.

19. A computer-implemented method as recited in claim **15** wherein the trigger is a time trigger, the computer implemented method further including the acts of:

monitoring for the occurrence of a particular time condition; and

in response to the occurrence of a particular time condition satisfying the trigger conditional functionality performing the particular consequential functionality defined by the trigger.

20. A computer-implemented method as recited in claim **15** wherein the trigger is installed and executed within the facilitator agent.

21. A computer-implemented method as recited in claim **15** wherein the trigger is installed and executed, within a first service-providing agent.

22. A computer-implemented method as recited in claim **15** wherein the conditional functionality of the trigger is installed on a facilitator agent.

23. A computer-implemented method as recited in claim **22** wherein the consequential functionality is installed on a specific service-providing agent other than a facilitator agent.

24. A computer-implemented method as recited in claim **15** wherein the conditional functionality of the trigger is installed on specific service-providing agent other than a facilitator agent.

25. A computer-implemented method as recited in claim **15** wherein the consequential functionality of the trigger is installed on a facilitator agent.

26. A computer-method as recited in claim **1** wherein the base goal is a compound goal having sub-goals separated by operators.

27. A computer-implemented method as recited in claim **26** wherein the type of available operators includes a conjunction operator, a disjunction operator, and a conditional execution operator.

28. A computer-implemented method as recited in claim **27** wherein the type of available operators further includes a parallel disjunction operator that indicates that disjunct goals are to be performed by different agents.

29. A computer program stored on a computer readable medium, the computer program executable to facilitate cooperative task completion within a distributed computing environment, the distributed computing environment including a plurality of autonomous electronic agents, the distributed computing environment supporting an Interagent Communication Language, the computer program comprising computer executable instructions for:

providing an agent registry that declares capabilities of service-providing electronic agents currently active within the distributed computing environment;

interpreting a service request in order to determine a base goal that may be a compound, arbitrarily complex base goal, the service request adhering to an Interagent Communication Language (ICL), where in the ICL includes:

a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events; and

a content layer comprising one or more of goals, triggers and data elements associated with the events;

the act of interpreting including the sub-acts of:

determining any task completion advice provided by the base goal, and

determining any task completion constraints provided by the base goal;

constructing a base goal satisfaction plan including the sub-acts of:

determining whether the request service is available,

determining sub-goals required in completing the base goal by using reasoning that includes one or more of domain-independent coordination strategies, domain-specific reasoning, and application-specific reasoning comprising rules and learning algorithms,

selecting service-providing electronic agents from the agent registry suitable for performing the determined sub-goals, and

ordering a delegation of sub-goal requests complete the requested service; and

implementing the base goal satisfaction plan.

30. A computer program as recited in claim **29** wherein the computer executable instruction for providing an agent registry includes the following computer executable instructions for registering a specific service-providing electronic agent into the agents registry

establishing a bi-directional communication link between the specific agent and a facilitator agent controlling the agent registry;

providing a new agent profile to the facilitator agent, the new agent profile defining publicly available capabilities of the specific agent; and

registering the specific agent together with the new agent profile within the agent registry, thereby making available to the facilitator agent the capabilities of the specific agent.

31. A computer program as recited in claim **30** wherein the computer executable instruction for registering a specific agent further includes:

invoking the specific agent in order to activate the specific agent;

instating an instance of the specific agent; and

transmitting the new agent profile from the specific agent to the facilitator agent in response to the instantiation of the specific agent.

32. A computer program as recited in claim **29** wherein the computer executable instruction for providing an agent registry includes a computer executable instruction for removing a specific service-providing electronic agent from the registry upon determining that the specific agent is no longer available to provide services.

33. A computer program as recited in claim **29** wherein the provided agent registry includes a symbolic name, a unique address, data declarations, trigger declarations, task declarations, and process characteristics for each active agent.

34. Computer program as recited in claim **29** further including computer executable instructions for receiving the service request via a communications link established with a client.

35. A computer program as recited in claim **29** wherein the computer executable instruction for providing a service request includes instructions for:

receiving a non-ICL format service request;

selecting an active agent capable of converting the non-ICL format service request into an ICL format service request;

forwarding the non-ICL format service request to the active agent capable of converting the non-ICL format service request, together with at request that such conversion be performed; and

receiving an ICL format service request corresponding to the non-ICL format service request.

36. A computer program as recited in claim **35** wherein the non-ICL format service request includes a natural language query, and the active agent capable of converting the non-ICL format service request into an ICL format service request is a natural language agent.

37. A computer program as recited in claim **36** wherein the natural language query is generated by a user in the agent.

38. A computer program as recited in claim **29**, the computer program further including computer executable instructions for implementing a base goal that requires setting a trigger having conditional and consequential functionality.

39. A computer program as recited in claim **38** wherein the trigger is an outgoing communications trigger, the computer program further including computer executable instructions for:

monitoring all outgoing communication events in order to determine whether a specific outgoing communication event has occurred; and

in response to the occurrence of the specific outgoing communication event, performing the particular action defined by the trigger.

40. A computer program as recited in claim **38** wherein the trigger is an incoming communications trigger, the computer program further including computer executable instructions for;

monitoring all incoming communication events in order to determine whether a specific incoming communication event has occurred; and

in response to the occurrence of the specific incoming communication event, performing the particular action defined by the trigger.

41. A computer program as recited in claim **38** wherein the trigger is a data trigger, the computer program further including computer executable instructions for:

monitoring a state of a data repository; and

in response to a particular state event, performing the particular action defined by the trigger.

42. A computer program as recited in claim **38** wherein the trigger is a time trigger, the computer program further including computer executable instructions for:

monitoring for the occurrence of a particular time condition; and

in response to the occurrence of the particular time condition, performing the particular action defined by the trigger.

43. A computer program as recited in claim **38** further including computer executable instructions for instating and executing the trigger within the facilitator agent.

44. A computer program as recited in claim **38** further including computer executable instructions for instating and executing the trigger within a first service-providing agent.

45. A computer program as recited in claim **29** further including computer executable instructions for interpreting compound goals having sub-goals separated by operators.

46. A computer program as recited in claim **45** wherein the type of available operators includes a conjunction operator, a disjunction operator, and a conditional execution operator.

47. A computer program as recited in claim **46** wherein the type of available operators further includes parallel disjunction operator that indicates that distinct goals are to be performed by different agents.

48. An Interagent Communication Language (ICL) providing a basis for facilitated cooperative task completion within a distributed computing environment having a facilitator agent and a plurality of autonomous service-providing electronic agents, wherein:

the ICL having:

a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events; and

a content layer comprising one or more of goals, triggers and data elements associated with the events;

the ICL having one or more features from a set of features comprising:

enabling agents perform queries of other agents;

enabling agents to exchange information with other gents; and

enabling agents to set triggers within other agents; and

the ICL having a syntax supporting compound goal expressions wherein said compound goal expressions are such that goals within a single request provided according to the ICL syntax may be coupled by one or more operators from a set of operators comprising:

a conditional execution operator; and

a parallel disjunctive operation that indicates that disjunct goals are to be performed by different agents.

49. An ICL as recited in claim **48**, wherein the ICL is computer platform independent.

50. An ICL as recited in claim **48** wherein the ICL is independent of computer programming languages which the plurality of agents are programmed in.

51. An ICL as recited in claim **48** wherein the ICL syntax supports explicit task completion constraints include use of specific agent constraints and response time constraints.

52. An ICL as recited in claim **51**, wherein possible types of task completion constraints include use of specific agent constraints and response time constraints.

53. An ICL as recited in claim **51** wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

54. An ICL as recited in claim **48** wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

55. An ICL as recited in claim **48** wherein each autonomous service-providing electronic agent defines and publishes a set of capability declarations or solvables, expressed in ICL, that describes services provided by such electronic agent.

56. An ICL as recited in claim **55** wherein an electronic agent's solvables define an interface for the electronic agent.

57. An ICL as recited in claim **56** wherein the facilitator agent maintains an agent registry making available plurality of electronic agent interfaces.

58. An ICL as recited in claim **57** wherein the possible types of solvables includes procedure solvables, a procedure solvable operable to implement a procedure such as a test or an action.

59. An ICL as recited in claim **58** wherein the possible types of solvables further includes data solvables, a data solvable operable to provide access to a collection of data.

60. An ICL as recited in claim **58** wherein the possible types of solvables includes data solvables, a data solvable operable to provide access to a collection of data.

61. A facilitator agent arranged to coordinate cooperative task completion within a distributed computing environment having a plurality of autonomous service-providing electronic agents, the facilitator agent comprising:

- an agent registry that declares capabilities of service-providing electronic agents currently active within the distributed computing environment; and
- a facilitating engine operable to parse a service requesting order to interpret a compound goal set forth therein, the compound goal including both local and global constraints and control parameters, the service request formed according to an Interagent Communication Language (ICL), wherein the ICL includes:
  - a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events; and
  - a content layer comprising one or more of goals, triggers and data elements associated with the events; and
- the facilitating engine further operable to construct a goal satisfaction plan by using reasoning that includes one or more of domain-independent coordination strategies, domain-specific reasoning, and application-specific reasoning comprising rules and learning algorithms.

62. A facilitator agent as recited in claim **61**, wherein the facilitating engine is capable of modifying the goal satisfaction plan during execution, the modifying initiated by events such as new agent declarations within the agent registry, decisions made by remote agents, and information, provided to the facilitating engine by remote agents.

63. A facilitator agent as recited in claim **61** wherein the agent registry includes a symbolic name, a unique address, data declarations, trigger declarations, task declarations, and process characteristics for each active agent.

64. A facilitator agent as recited in claim **61** wherein the facilitating engine is operable to install a trigger mechanism requesting that a certain action be taken when a certain set of conditions are met.

65. A facilitator agent as recited in claim **64** wherein the trigger mechanism is a communication trigger that monitors communication events and performs the certain action when a certain communication event occurs.

66. A facilitator agent as recited in claim **64** wherein the trigger mechanism is a data trigger that monitors a state of a data repository and performs the certain action when a certain data state is obtained.

67. A facilitator agent as recited in claim **66** wherein the data repository is local to the facilitator agent.

68. A facilitator agent as recited in claim **66** wherein the data repository is remote from the facilitator agent.

69. A facilitator agent as recited in claim **64** wherein the trigger mechanism is a task trigger having a set of conditions.

70. A facilitator agent as recited in claim **61**, the facilitator agent further including a global database accessible to at least one of the service-providing electronic agents.

71. A software-based, flexible computer architecture for communication and cooperation among distributed electronic agents, the architecture contemplating a distributed computing system comprising:

- a plurality of service-providing electronic agents;
- an Interagent Communication Language (ICL), wherein the inter-agent language includes:
  - a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events; and
  - a content layer comprising one or more of goals, triggers and data elements associated with the events; and
- a facilitator agent in bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including:
  - an agent registry that declares capabilities of service-providing electronic agents currently active within the distributed computing environment;
  - a facilitating engine operable to parse a service request in order to interpret an arbitrarily complex goal set forth therein, the facilitating engine further operable to construct a goal satisfaction plan including the coordination of a suitable delegation of sub-goal requests to best complete the requested service by using reasoning that includes one or more of domain-independent coordination strategies, domain-specific reasoning, and application-specific reasoning comprising rules and learning algorithms.

72. A computer architecture as recited in claim **71**, wherein the Interagent Communication Language (ICL) is for enabling agents to perform queries of other agents, exchange Information with other agents, and set triggers within other agents, the ICL further defined by an ICL syntax supporting compound goal expressions such that goals within single request provided according to the ICL syntax may be coupled by a conjunctive operator, a disjunctive operator, a conditional execution operator, and a parallel disjunctive operator parallel disjunctive operator that indicates that disjunct goals are to be performed by different agents.

73. A computer architecture as recited in claim **72**, wherein the ICL is computer platform independent.

74. A computer architecture as recited in claim **73** wherein the ICL is independent of computer programming languages in which the plurality of agents are programmed.

75. A computer architecture as recited in claim **73** wherein the ICL syntax supports explicit task completion constraints within goal expressions.

76. A computer architecture as recited in claim **75** wherein possible types of task completion constraints include use of specific agent constraints and response time constraints.

77. A computer architecture as recited in claim **75** wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

78. A computer architecture as recited in claim **73** wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

79. A computer architecture as recited in claim **73** wherein each autonomous service-providing electronic agent defines and publishes a set of capability declarations or solvables, expressed in ICL, that describes services provided by such electronic agent.

80. A computer architecture as recited in claim **79** wherein an electronic agent's solvables define an interface for the electronic agent.

81. A computer architecture as recited in claim **80** wherein the possible types of solvables includes procedure solvables, a procedure solvable operable to implement a procedure such as a test or an action.

82. A computer architecture as recited in claim **81** wherein the possible types of solvables further includes data solvables, a data solvable operable to provide access to a collection of data.

83. A computer architecture as recited in claim **82** wherein the possible types of solvables includes a data solvable operable to provide access to modify a collection of data.

84. A computer architecture as recited in claim **71** wherein a planning component of the facilitating engine are distributed across at least two computer processes.

85. A computer architecture as recited in claim **71** wherein an execution component of the facilitating engine is distributed across at least two computer process.

86. A data wave carrier providing a transport mechanism for information communication in a distributed computing environment having at least one facilitator agent and at least one active client agent, and an Interagent Communication Language (ICL), wherein the ICL includes:

a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events; and

a content layer comprising one or more of goals, triggers and data elements associated with the events;

wherein said at least one facilitator agent is operable to construct a goal satisfaction plan by using reasoning that includes one or more of domain-independent coordination strategies, domain-specific reasoning, and application-specific reasoning comprising rules and learning algorithms for satisfying one or more requests for service from said at least one active client agent, the data wave carrier comprising a signal representation of an inter-agent language description of an active client agent's functional capabilities.

87. A data wave carrier as recited in claim **86**, the data wave carrier further comprising a corresponding signal representation of said one or more requests for service in the inter-agent language from a first agent to a second agent.

88. A data wave carrier as recited in claim **86**, the data wave carrier further comprising a signal representation of a goal dispatched to an agent for performance from a facilitator agent.

89. A data wave carrier as recited in claim **88** wherein a later state of the data wave carrier comprises a signal representation of a response to the dispatched goal including results and/or a status report from the agent for performance to the facilitator agent.

* * * * *

A

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF EXPRESS MAILING

I hereby certify that this paper and the documents and/or fees referred to as attached therein are being deposited with the United States Postal Service on January 05, 1999 in an envelope as "Express Mail Post Office to Addressee" service under 37 CFR §1.10, Mailing Label Number EL221766053US, addressed to the Assistant Commissioner for Patents, Washington, DC 20231.

Michael L. Gough

Attorney Docket No.: SRI1P016

First Named Inventor:

CHEYER, Adam J.

UTILITY PATENT APPLICATION TRANSMITTAL (37 CFR § 1.53(b))

Assistant Commissioner for Patents
Box Patent Application
Washington, DC 20231

□ Duplicate for fee processing

Sir: This is a request for filing a patent application under 37 CFR § 1.53(b) in the name of inventors: Adam J. Cheyer and David L. Martin

For: **SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS**

Application Elements:

☒ 59 Pages of Specification, Claims and Abstract
☒ 16 Sheets of Drawings
☒ 01 Pages Combined Declaration and Power of Attorney

Accompanying Application Parts:

☒ Assignment and Assignment Recordation Cover Sheet (recording fee not enclosed)
☒ Return Receipt Postcard

Fee Calculation (37 CFR § 1.16)

	(Col. 1) NO. FILED		(Col. 2) NO. EXTRA	SMALL ENTITY RATE	 FEE	OR	LARGE ENTITY RATE	 FEE
BASIC FEE				$395	$	OR	$760	$ 760.00
TOTAL CLAIMS	89	-20 =	69	x11 =	$	OR	x18 =	$1242.00
INDEP CLAIMS	06	-03 =	03	x41 =	$	OR	x78 =	$ 234.00
* If the difference in Col. 1 is less than zero, enter "0" in Col. 2.				Total	$	OR	Total	$2236.00

Including filing fees and the assignment recordation fee of $40.00, the Commissioner is authorized to charge all required fees to Deposit Account No. 50-0384 (Order No. SRI1P016).

☒ The Commissioner is authorized to charge any fees beyond the amount enclosed which may be required, or to credit any overpayment, to Deposit Account No. 50-0384 (Order No. SRI1P016).

General Authorization for Petition for Extension of Time (37 CFR §1.136)

☒ Applicants hereby make and generally authorize any Petitions for Extensions of Time as may be needed for any subsequent filings. The Commissioner is also authorized to charge any extension fees under 37 CFR §1.17 as may be needed to Deposit Account No. 50-0384.

☒ Please send correspondence to the following address:

Brian R. Coleman
HICKMAN STEPHENS & COLEMAN, LLP
P.O. Box 52037
Palo Alto, CA 94303-0746

Tel (650) 470-7430
Fax (650) 470-7440

Date: 1/5/99

Brian R. Coleman
Registration No. **39,145**

Software-Based Architecture for Communication and Cooperation Among Distributed Electronic Agents

By:

Adam J. Cheyer and David L. Martin

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention is related to distributed computing environments and the completion of tasks within such environments. In particular, the present invention teaches a variety of software-based architectures for communication and cooperation among distributed electronic agents. Certain embodiments teach interagent communication languages enabling client agents to make requests in the form of arbitrarily complex goal expressions that are solved through facilitation by a facilitator agent.

Context and Motivation for Distributed Software Systems

The evolution of models for the design and construction of distributed software systems is being driven forward by several closely interrelated trends: the adoption of a *networked computing model*, rapidly rising expectations for *smarter, longer-lived, more autonomous software applications* and an ever increasing demand for *more accessible and intuitive user interfaces.*

Prior Art Figure 1 illustrates a *networked computing model* 100 having a plurality of client and server computer systems 120 and 122 coupled together over a physical transport mechanism 140. The adoption of the *networked computing model* 100 has lead to a greatly increased reliance on distributed sites for both data and processing resources. Systems such as the networked computing model 100 are based upon at least one physical transport mechanism 140 coupling the multiple computer systems 120 and 122 to support the transfer of information between these computers. Some of these computers basically support using the network and are known as *client* *computers* (*clients*). Some of these computers provide resources to other computers and are known as *server computers* (*servers*). The servers 122 can vary greatly in the resources they possess, access they provide and services made available to other computers across a network. Servers may service other servers as well as clients.

The Internet is a computing system based upon this network computing model. The Internet is continually growing, stimulating a paradigm shift for computing away from requiring all relevant data and programs to reside on the user's desktop machine. The data now routinely accessed from computers spread around the world has become increasingly rich in format, comprising multimedia documents, and audio and video streams. With the popularization of programming languages such as JAVA, data transported between local and remote machines may also include programs that can be downloaded and executed on the local machine. There is an ever increasing reliance on networked computing, necessitating software design approaches that allow for flexible composition of distributed processing elements in a dynamically changing and relatively unstable environment.

In an increasing variety of domains, application designers and users are coming to expect the deployment of *smarter, longer-lived, more autonomous, software applications.* Push technology, persistent monitoring of information sources, and the maintenance of user models, allowing for personalized responses and sharing of preferences, are examples of the simplest manifestations of this trend. Commercial enterprises are introducing significantly more advanced approaches, in many cases employing recent research results from artificial intelligence, data mining, machine learning, and other fields.

More than ever before, the increasing complexity of systems, the development of new technologies, and the availability of multimedia material and environments are creating a demand for *more accessible and intuitive user interfaces.* Autonomous, distributed, multi-component systems providing sophisticated services will no longer lend themselves to the familiar "direct manipulation" model of interaction, in which an individual user masters a fixed selection of commands provided by a single application. Ubiquitous computing, in networked environments, has brought about a situation in which the typical user of many software services is likely to be a non-expert, who may access a given service infrequently or only a few times.

Accommodating such usage patterns calls for new approaches. Fortunately, input modalities now becoming widely available, such as speech recognition and pen-based handwriting/gesture recognition, and the ability to manage the presentation of systems' responses by using multiple media provide an opportunity to fashion a style of human-computer interaction that draws much more heavily on our experience with human-human interactions.

PRIOR RELATED ART

Existing approaches and technologies for distributed computing include distributed objects, mobile objects, blackboard-style architectures, and agent-based software engineering.

The Distributed Object Approach

Object-oriented languages, such as C++ or JAVA, provide significant advances over standard procedural languages with respect to the reusability and modularity of code: *encapsulation*, *inheritance* and *polymorhpism*. Encapsulation encourages the creation of library interfaces that minimize dependencies on underlying algorithms or data structures. Changes to programming internals can be made at a later date with requiring modifications to the code that uses the library. Inheritance permits the extension and modification of a library of routines and data without requiring source code to the original library. Polymorphism allows one body of code to work on an arbitrary number of data types. For the sake of simplicity traditional objects may be seen to contain both methods and data. Methods provide the mechanisms by which the internal state of an object may be modified or by which communication may occur with another object or by which the instantiation or removal of objects may be directed.

With reference to Figure 2, a distributed object technology based around an Object Request Broker will now be described. Whereas "standard" object-oriented programming (OOP) languages can be used to build monolithic programs out of many object building blocks, distributed object technologies (DOOP) allow the creation of programs whose components may be spread across multiple machines. As shown in Figure 2, an object system 200 includes client objects 210 and server objects 220. To implement a client-server relationship between objects, the distributed object system

200 uses a registry mechanism (CORBA's registry is called an Object Request Broker, or ORB) 230 to store the interface descriptions of available objects. Through the services of the ORB 230, a client can transparently invoke a method on a remote server object. The ORB 230 is then responsible for finding the object 220 that can implement the request, passing it the parameters, invoking its method, and returning the results. In the most sophisticated systems, the client 210 does not have to be aware of where the object is located, its programming language, its operating system, or any other system aspects that are not part of the server object's interface.

Although distributed objects offer a powerful paradigm for creating networked applications, certain aspects of the approach are not perfectly tailored to the constantly changing environment of the Internet. A major restriction of the DOOP approach is that the interactions among objects are fixed through explicitly coded instructions by the application developer. It is often difficult to reuse an object in a new application without bringing along all its inherent dependencies on other objects (embedded interface definitions and explicit method calls). Another restriction of the DOOP approach is the result of its reliance on a remote procedure call (RPC) style of communication. Although easy to debug, this single thread of execution model does not facilitate programming to exploit the potential for parallel computation that one would expect in a distributed environment. In addition, RPC uses a blocking (synchronous) scheme that does not scale well for high-volume transactions.

Mobile Objects

Mobile objects, sometimes called mobile agents, are bits of code that can move to another execution site (presumably on a different machine) under their own programmatic control, where they can then interact with the local environment. For certain types of problems, the mobile object paradigm offers advantages over more traditional distributed object approaches. These advantages include network bandwidth and parallelism. Network bandwidth advantages exist for some database queries or electronic commerce applications, where it is more efficient to perform tests on data by bringing the tests to the data than by bringing large amounts of data to the testing program. Parallelism advantages include situations in which mobile agents can be spawned in parallel to accomplish many tasks at once.

Some of the disadvantages and inconveniences of the mobile agent approach include the programmatic specificity of the agent interactions, lack of coordination support between participant agents and execution environment irregularities regarding specific programming languages supported by host processors upon which agents reside. In a fashion similar to that of DOOP programming, an agent developer must programmatically specify where to go and how to interact with the target environment. There is generally little coordination support to encourage interactions among multiple (mobile) participants. Agents must be written in the programming language supported by the execution environment, whereas many other distributed technologies support heterogeneous communities of components, written in diverse programming languages.

Blackboard Architectures

Blackboard architectures typically allow multiple processes to communicate by reading and writing tuples from a global data store. Each process can watch for items of interest, perform computations based on the state of the blackboard, and then add partial results or queries that other processes can consider. Blackboard architectures provide a flexible framework for problem solving by a dynamic community of distributed processes. A blackboard architecture provides one solution to eliminating the tightly bound interaction links that some of the other distributed technologies require during interprocess communication. This advantage can also be a disadvantage: although a programmer does not need to refer to a specific process during computation, the framework does not provide programmatic control for doing so in cases where this would be practical.

Agent-based Software Engineering

Several research communities have approached distributed computing by casting it as a problem of modeling communication and cooperation among autonomous entities, or agents. Effective communication among independent agents requires four components: (1) a transport mechanism carrying messages in an asynchronous fashion, (2) an interaction protocol defining various types of communication interchange and their social implications (for instance, a response is expected of a question), (3) a content language permitting the expression and interpretation of utterances, and (4) an agreed-upon set of shared vocabulary and meaning for concepts (often called an *ontology*). Such mechanisms permit a much richer style of interaction among participants than can be expressed using a distributed object's RPC model or a blackboard architecture's centralized exchange approach.

Agent-based systems have shown much promise for flexible, fault-tolerant, distributed problem solving. Several agent-based projects have helped to evolve the notion of facilitation. However, existing agent-based technologies and architectures are typically very limited in the extent to which agents can specify complex goals or influence the strategies used by the facilitator. Further, such prior systems are not sufficiently attuned to the importance of integrating human agents (i.e., users) through natural language and other human-oriented user interface technologies.

The initial version of SRI International's Open Agent Architecture™ ("*OAA*®") technology provided only a very limited mechanism for dealing with compound goals. Fixed formats were available for specifying a flat list of either conjoined (AND) sub-goals or disjoined (OR) sub-goals; in both cases, parallel goal solving was hard-wired in, and only a single set of parameters for the entire list could be specified. More complex goal expressions involving (for example) combinations of different boolean connectors, nested expressions, or conditionally interdependent ("IF .. THEN") goals were not supported. Further, system scalability was not adequately addressed in this prior work.

SUMMARY OF INVENTION

A first embodiment of the present invention discloses a highly flexible, software-based architecture for constructing distributed systems. The architecture supports cooperative task completion by flexible, dynamic configurations of autonomous electronic agents. Communication and cooperation between agents are brokered by one or more facilitators, which are responsible for matching requests, from users and agents, with descriptions of the capabilities of other agents. It is not generally required that a user or agent know the identities, locations, or number of other agents involved in satisfying a request, and relatively minimal effort is involved in incorporating new agents and "wrapping" legacy applications. Extreme flexibility is achieved through an architecture organized around the declaration of capabilities by service-providing agents, the construction of arbitrarily complex goals by users and service-requesting agents, and the role of facilitators in delegating and coordinating the satisfaction of these goals, subject to advice and constraints that may accompany them. Additional mechanisms and features include facilities for creating and maintaining shared repositories of data; the use of triggers to instantiate commitments within and between agents; agent-based provision of multi-modal user interfaces, including natural language; and built-in support for including the user as a privileged member of the agent community. Specific embodiments providing enhanced scalability are also described.

BRIEF DESCRIPTION OF THE DRAWINGS

Prior Art

Prior Art FIGURE 1 depicts a networked computing model;

Prior Art FIGURE 2 depicts a distributed object technology based around an Object Resource Broker;

Examples of the Invention

FIGURE 3 depicts a distributed agent system based around a facilitator agent;

FIGURE 4 presents a structure typical of one small system of the present invention;

FIGURE 5 depicts an Automated Office system implemented in accordance with an example embodiment of the present invention supporting a mobile user with a laptop computer and a telephone;

FIGURE 6 schematically depicts an Automated Office system implemented as a network of agents in accordance with a preferred embodiment of the present invention;

FIGURE 7 schematically shows data structures internal to a facilitator in accordance with a preferred embodiment of the present invention;

FIGURE 8 depicts operations involved in instantiating a client agent with its parent facilitator in accordance with a preferred embodiment of the present invention;

FIGURE 9 depicts operations involved in a client agent initiating a service request and receiving the response to that service request in accordance with a certain preferred embodiment of the present invention;

FIGURE 10 depicts operations involved in a client agent responding to a service request in accordance with another preferable embodiment of the present invention;

FIGURE 11 depicts operations involved in a facilitator agent response to a service request in accordance with a preferred embodiment of the present invention;

FIGURE 12 depicts an Open Agent Architecture^{TM} based system of agents implementing a unified messaging application in accordance with a preferred embodiment of the present invention;

FIGURE 13 depicts a map oriented graphical user interface display as might be displayed by a multi-modal map application in accordance with a preferred embodiment of the present invention;

FIGURE 14 depicts a peer to peer multiple facilitator based agent system supporting distributed agents in accordance with a preferred embodiment of the present invention;

FIGURE 15 depicts a multiple facilitator agent system supporting at least a limited form of a hierarchy of facilitators in accordance with a preferred embodiment of the present invention; and

FIGURE 16 depicts a replicated facilitator architecture in accordance with one embodiment of the present invention.

BRIEF DESCRIPTION OF THE APPENDICES

The Appendices provide source code for an embodiment of the present invention written in the PROLOG programming language.

APPENDIX A: Source code file named compound.pl.

APPENDIX B: Source code file named fac.pl.

APPENDIX C: Source code file named libcom_tcp.pl.

APPENDIX D: Source code file named liboaa.pl.

APPENDIX E: Source code file named translations.pl.

DETAILED DESCRIPTION OF THE INVENTION

Figure 3 illustrates a distributed agent system 300 in accordance with one embodiment of the present invention. The agent system 300 includes a facilitator agent 310 and a plurality of agents 320. The illustration of Figure 3 provides a high level view of one simple system structure contemplated by the present invention. The facilitator agent 310 is in essence the "parent" facilitator for its "children" agents 320. The agents 320 forward service requests to the facilitator agent 310. The facilitator agent 310 interprets these requests, organizing a set of goals which are then delegated to appropriate agents for task completion.

The system 300 of Figure 3 can be expanded upon and modified in a variety of ways consistent with the present invention. For example, the agent system 300 can be distributed across a computer network such as that illustrated in Figure 1. The facilitator agent 310 may itself have its functionality distributed across several different computing platforms. The agents 320 may engage in interagent communication (also called peer to peer communications). Several different systems 300 may be coupled together for enhanced performance. These and a variety of other structural configurations are described below in greater detail.

Figure 4 presents the structure typical of a small system 400 in one embodiment of the present invention, showing user interface agents 408, several application agents 404 and meta-agents 406, the system 400 organized as a community of peers by their common relationship to a facilitator agent 402. As will be appreciated, Figure 4 places more structure upon the system 400 than shown in Figure 3, but both are valid representations of structures of the present invention. The facilitator 402 is a specialized server agent that is responsible for coordinating agent communications and cooperative problem-solving. The facilitator 402 may also provide a global data store for its client agents, allowing them to adopt a blackboard style of interaction. Note that certain advantages are found in utilizing two or more facilitator agents within the system 400. For example, larger systems can be assembled from multiple facilitator/client groups, each having the sort of structure shown in Figure 4. All agents that are not facilitators are referred to herein generically as *client* agents -- so called because each acts (in some respects) as a client of some facilitator, which provides communication and other essential services for the client.

The variety of possible client agents is essentially unlimited. Some typical categories of client agents would include application agents 404, meta-agents 406, and user interface agents 408, as depicted in Figure 4. Application agents 404 denote specialists that provide a collection of services of a particular sort. These services could be domain-independent technologies (such as speech recognition, natural language processing 410, email, and some forms of data retrieval and data mining) or user-specific or domain-specific (such as a travel planning and reservations agent). Application agents may be based on legacy applications or libraries, in which case the agent may be little more than a wrapper that calls a pre-existing API 412, for example. Meta-agents 406 are agents whose role is to assist the facilitator agent 402 in coordinating the activities of other agents. While the facilitator 402 possesses domain-independent coordination strategies, meta-agents 406 can augment these by using domain- and application-specific knowledge or reasoning (including but not limited to rules, learning algorithms and planning).

With further reference to Figure 4, user interface agents 408 can play an extremely important and interesting role in certain embodiments of the present invention. By way of explanation, in some systems, a user interface agent can be implemented as a collection of "micro-agents", each monitoring a different input modality (point-and-click, handwriting, pen gestures, speech), and collaborating to produce the best interpretation of the current inputs. These micro-agents are depicted in Figure 4, for example, as Modality Agents 414. While describing such subcategories of client agents is useful for purposes of illustration and understanding, they need not be formally distinguished within the system in preferred implementations of the present invention.

The operation of one preferred embodiment of the present invention will be discussed in greater detail below, but may be briefly outlined as follows. When invoked, a client agent makes a connection to a facilitator, which is known as its *parent facilitator*. These connections are depicted as a double headed arrow between the client agent and the facilitator agent in Figure 3 and 4, for example. Upon connection, an agent registers with its parent facilitator a specification of the capabilities and services it can provide. For example, a natural language agent may register the characteristics of its available natural language vocabulary. (For more details regarding client agent connections, see the discussion of Figure 8 below.) Later during task completion, when a facilitator determines that the registered services 416 of one of its client agents will help satisfy a goal, the facilitator sends that client a request expressed in the Interagent Communication Language (*ICL*) 418. (See Figure 11 below for a more detailed discussion of the facilitator operations involved.) The agent parses this request, processes it, and returns answers or status reports to the facilitator. In processing a request, the client agent can make use of a variety of infrastructure capabilities provided in the preferred embodiment. For example, the client agent can use *ICL* 418 to request services of other agents, set triggers, and read or write shared data on the facilitator or other client agents that maintain shared data. (See the discussion of Figures 9-11 below for a more detailed discussion of request processing.)

The functionality of each client agent are made available to the agent community through registration of the client agent's capabilities with a facilitator 402. A software "wrapper" essentially surrounds the underlying application program performing the services offered by each client. The common infrastructure for constructing agents is preferably supplied by an *agent library.* The agent library is preferably accessible in the runtime environment of several different programming languages. The agent library preferably minimizes the effort required to construct a new system and maximizes the ease with which legacy systems can be “wrapped” and made compatible with the agent-based architecture of the present invention.

By way of further illustration, a representative application is now briefly presented with reference to Figures 5 and 6. In the Automated Office system depicted in Figure 5, a mobile user with a telephone and a laptop computer can access and task commercial applications such as calendars, databases, and email systems running back at the office. A user interface (UI) agent 408, shown in Figure 6, runs on the user's local laptop and is responsible for accepting user input, sending requests to the facilitator 402 for delegation to appropriate agents, and displaying the results of the distributed computation. The user may interact directly with a specific remote application by clicking on active areas in the interface, calling up a form or window for that application, and making queries with standard interface dialog mechanisms. Conversely, a user may express a task to be executed by using typed, handwritten, or spoken (over the telephone) English sentences, without explicitly specifying which agent or agents should perform the task.

For instance, if the question "What is my schedule?" is written 420 in the user interface 408, this request will be sent 422 by the UI 408 to the facilitator 402, which in turn will ask 424 a natural language (NL) agent 426 to translate the query into *ICL* 18. To accomplish this task, the NL agent 426 may itself need to make requests of the agent community to resolve unknown words such as "me" 428 (the UI agent 408 can respond 430 with the name of the current user) or "schedule" 432 (the calendar agent 434 defines this word 436). The resulting *ICL* expression is then routed by the facilitator 402 to appropriate agents (in this case, the calendar agent 434) to execute the request. Results are sent back 438 to the UI agent 408 for display.

The spoken request "When mail arrives for me about security, notify me immediately." produces a slightly more complex example involving communication among all agents in the system. After translation into *ICL* as described above, the facilitator installs a trigger 440 on the mail agent 442 to look for new messages about security. When one such message does arrive in its mail spool, the trigger fires, and the facilitator matches the action part of the trigger to capabilities published by the notification agent 446. The notification agent 446 is a meta-agent, as it makes use of rules concerning the optimal use of different output modalities (email, fax, speech generation over the telephone) plus information about an individual user's preferences 448 to determine the best way of relaying a message through available media transfer application agents. After some competitive parallelism to locate the user (the calendar agent 434 and database agent 450 may have different guesses as to where to find the user) and some cooperative parallelism to produce required information (telephone number of location, user password, and an audio file containing a text-to-speech representation of the email message), a telephone agent 452 calls the user, verifying its identity through touchtones, and then play the message.

The above example illustrates a number of inventive features. As new agents connect to the facilitator, registering capability specifications and natural language vocabulary, what the user can say and do dynamically changes; in other words, the ICL is dynamically ***expandable***. For example, adding a calendar agent to the system in the previous example and registering its capabilities enables users to ask natural language questions about their "schedule" without any need to revise code for the facilitator, the natural language agents, or any other client agents. In addition, the interpretation and execution of a task is a distributed process, with no single agent defining the set of possible inputs to the system. Further, a single request can produce cooperation and flexible communication among many agents, written in different programming languages and spread across multiple machines.

Design Philosophy and Considerations

One preferred embodiment provides an integration mechanism for heterogeneous applications in a distributed infrastructure, incorporating some of the dynamism and extensibility of blackboard approaches, the efficiency associated with mobile objects, plus the rich and complex interactions of communicating agents. Design goals for preferred embodiments of the present invention may be categorized under the general headings of *interoperation and cooperation, user interfaces*, and *software engineering*. These design goals are not absolute requirements, nor will they necessarily be satisfied by all embodiments of the present invention, but rather simply reflect the inventor's currently preferred design philosophy.

Versatile mechanisms of interoperation and cooperation

Interoperation refers to the ability of distributed software components - agents - to communicate meaningfully. While every system-building framework must provide mechanisms of interoperation at some level of granularity, agent-based frameworks face important new challenges in this area. This is true primarily because autonomy, the hallmark of *individual* agents, necessitates greater flexibility in interactions within *communities* of agents. *Coordination* refers to the mechanisms by which a community of agents is able to work together productively on some task. In these areas, the goals for our framework are to *provide flexibility in assembling* *communities of autonomous service providers, provide flexibility in structuring cooperative interactions, impose the right amount of structure,* as well as *include legacy and "owned-elsewhere" applications.*

Provide flexibility in assembling communities of autonomous service providers -- both at development time and at runtime. Agents that conform to the linguistic and ontological requirements for effective communication should be able to participate in an agent community, in various combinations, with minimal or near minimal prerequisite knowledge of the characteristics of the other players. Agents with duplicate and overlapping capabilities should be able to coexist within the same community, with the system making optimal or near optimal use of the redundancy.

Provide flexibility in structuring cooperative interactions among the members of a community of agents. A framework preferably provides an economical mechanism for setting up a variety of interaction patterns among agents, without requiring an inordinate amount of complexity or infrastructure within the individual agents. The provision of a service should be independent or minimally dependent upon a particular configuration of agents.

Impose the right amount of structure on individual agents. Different approaches to the construction of multi-agent systems impose different requirements on the individual agents. For example, because KQML is neutral as to the content of messages, it imposes minimal structural requirements on individual agents. On the other hand, the BDI paradigm tends to impose much more demanding requirements, by making assumptions about the nature of the programming elements that are meaningful to individual agents. Preferred embodiments of the present invention should fall somewhere between the two, providing a rich set of interoperation and coordination capabilities, without precluding any of the software engineering goals defined below.

Include legacy and "owned-elsewhere" applications. Whereas *legacy* usually implies reuse of an established system fully controlled by the agent-based system developer, *owned-elsewhere* refers to applications to which the developer has partial access, but no control. Examples of owned-elsewhere applications include data sources and services available on the World Wide Web, via simple form-based interfaces, and applications used cooperatively within a virtual enterprise, which remain the properties of separate corporate entities. Both classes of application must preferably be able to interoperate, more or less as full-fledged members of the agent community, without requiring an overwhelming integration effort.

Human-oriented user interfaces

Systems composed of multiple distributed components, and possibly dynamic configurations of components, require the crafting of intuitive user interfaces to *provide conceptually natural interaction mechanisms, treat users as privileged members of the agent community* and *support collaboration.*

Provide conceptually natural interaction mechanisms with multiple distributed components. When there are numerous disparate agents, and/or complex tasks implemented by the system, the user should be able to express requests without having detailed knowledge of the individual agents. With speech recognition, handwriting recognition, and natural language technologies becoming more mature, agent architectures should preferably support these forms of input playing increased roles in the tasking of agent communities.

Preferably treat *users as privileged members* of the agent community by providing an appropriate level of task specification within *software* agents, and reusable translation mechanisms between this level and the level of *human* requests, supporting constructs that seamlessly incorporate interactions between both human-interface and software types of agents.

Preferably support *collaboration* (simultaneous work over shared data and processing resources) between users and agents.

Realistic software engineering requirements

System-building frameworks should preferably address the practical concerns of real-world applications by the specification of requirements which preferably include: *Minimize the effort* required to create new agents, and to wrap existing applications. *Encourage reuse,* both of domain-independent and domain-specific components. The concept of *agent orientation*, like that of object orientation, provides a natural conceptual framework for reuse, so long as mechanisms for encapsulation and interaction are structured appropriately. *Support lightweight, mobile platforms.* Such platforms should be able to serve as hosts for agents, without requiring the installation of a massive environment. It should also be possible to construct individual agents that are relatively small and modest in their processing requirements. *Minimize platform and language barriers.* Creation of new agents, as well as wrapping of existing applications, should not require the adoption of a new language or environment.

Mechanisms of Cooperation

Cooperation among agents in accordance with the present invention is preferably achieved via messages expressed in a common language, *ICL.* Cooperation among agent is further preferably structured around a three-part approach: providers of services register capabilities specifications with a facilitator, requesters of services construct goals and relay them to a facilitator, and facilitators coordinate the efforts of the appropriate service providers in satisfying these goals.

The Interagent Communication Language (ICL)

Interagent Communication Language (*"ICL"*) 418 refers to an interface, communication, and task coordination language preferably shared by all agents, regardless of what platform they run on or what computer language they are programmed in. *ICL* may be used by an agent to task itself or some subset of the agent community. Preferably, *ICL* allows agents to specify explicit control parameters while simultaneously supporting expression of goals in an underspecified, loosely constrained manner. In a further preferred embodiment, agents employ *ICL* to perform queries, execute actions, exchange information, set triggers, and manipulate data in the agent community.

In a further preferred embodiment, a program element expressed in *ICL* is the *event.* The activities of every agent, as well as communications between agents, are preferably structured around the transmission and handling of events. In communications, events preferably serve as messages between agents; in regulating the activities of individual agents, they may preferably be thought of as goals to be satisfied. Each event preferably has a type, a set of parameters, and content. For example, the agent library procedure *oaa_Solve* can be used by an agent to request services of other agents. A call to *oaa_Solve*, within the code of agent *A*, results in an event having the form

ev_post_solve(Goal, Params)

going from *A* to the facilitator, where *ev_post_solve* is the type, *Goal* is the content, and *Params* is a list of parameters. The allowable content and parameters preferably vary according to the type of the event.

The *ICL* preferably includes a layer of conversational protocol and a content layer. The conversational layer of *ICL* is defined by the event types, together with the parameter lists associated with certain of these event types. The content layer consists of the specific goals, triggers, and data elements that may be embedded within various events.

The *ICL* conversational protocol is preferably specified using an orthogonal, parameterized approach, where the conversational aspects of each element of an interagent conversation are represented by a selection of an event type and a selection of values from at least one orthogonal set of parameters. This approach offers greater expressiveness than an approach based solely on a fixed selection of *speech acts*, such as embodied in KQML. For example, in KQML, a request to satisfy a query can employ either of the performatives *ask_all* or *ask_one*. In *ICL*, on the other hand, this type of request preferably is expressed by the event type *ev_post_solve*, together with the *solution_limit(N)* parameter - where *N* can be any positive integer. (A request for all solutions is indicated by the omission of the *solution_limit* parameter.) The request can also be accompanied by other parameters, which combine to further refine its semantics. In KQML, then, this example forces one to choose between two possible conversational options, neither of which may be precisely what is desired. In either case, the performative chosen is a single value that must capture the entire conversational characterization of the communication. This requirement raises a difficult challenge for the language designer, to select a set of performatives that provides the desired functionality without becoming unmanageably large. Consequently, the debate over the right set of performatives has consumed much discussion within the KQML community.

The content layer of the *ICL* preferably supports unification and other features found in logic programming language environments such as PROLOG. In some embodiments, the content layer of the *ICL* is simply an extension of at least one programming language. For example, the Applicants have found that PROLOG is suitable for implementing and extending into the content layer of the *ICL*. The agent libraries preferably provide support for constructing, parsing, and manipulating *ICL* expressions. It is possible to embed content expressed in other languages within an *ICL* event. However, expressing content in *ICL* simplifies the facilitator's access to the content, as well as the conversational layer, in delegating requests. This gives the facilitator more information about the nature of a request and helps the facilitator decompose compound requests and delegate the sub-requests.

Further, *ICL* expressions preferably include, in addition to events, at least one of the following: capabilities declarations, requests for services, responses to requests, trigger specifications, and shared data elements. A further preferred embodiment of the present invention incorporates *ICL* expressions including at least all of the following: events, capabilities declarations, requests for services, responses to requests, trigger specifications, and shared data elements.

Providing Services: Specifying "Solvables"

In a preferred embodiment of the present invention, every participating agent defines and publishes a set of capability declarations, expressed in *ICL*, describing the services that it provides. These declarations establish a high-level interface to the agent. This interface is used by a facilitator in communicating with the agent, and, most important, in delegating service requests (or parts of requests) to the agent. Partly due to the use of PROLOG as a preferred basis for *ICL*, these capability declarations are referred as *solvables.* The agent library preferably provides a set of procedures allowing an agent to add, remove, and modify its solvables, which it may preferably do at any time after connecting to its facilitator.

There are preferably at least two major types of solvables: *procedure* solvables and *data* solvables. Intuitively, a procedure solvable performs a test or action, whereas a data solvable provides access to a collection of data. For example, in creating an agent for a mail system, procedure solvables might be defined for sending a message to a person, testing whether a message about a particular subject has arrived in the mail queue, or displaying a particular message onscreen. For a database wrapper agent, one might define a distinct data solvable corresponding to each of the relations present in the database. Often, a data solvable is used to provide a *shared* data store, which may be not only queried, but also updated, by various agents having the required permissions.

There are several primary technical differences between these two types of solvables. First, each procedure solvable must have a handler declared and defined for it, whereas this is preferably not necessary for a data solvable. The handling of requests for a data solvable is preferably provided transparently by the agent library. Second, data solvables are preferably associated with a dynamic collection of facts (or clauses), which may be further preferably modified at runtime, both by the agent providing the solvable, and by other agents (provided they have the required permissions). Third, special features, available for use with data solvables, preferably facilitate maintaining the associated facts. In spite of these differences, it should be noted that the mechanism of *use* by which an agent requests a service is the same for the two types of solvables.

In one embodiment, a request for one of an agent's services normally arrives in the form of an event from the agent's facilitator. The appropriate handler then deals with this event. The handler may be coded in whatever fashion is most appropriate, depending on the nature of the task, and the availability of task-specific libraries or legacy code, if any. The only hard requirement is that the handler return an appropriate response to the request, expressed in *ICL*. Depending on the nature of the request, this response could be an indication of success or failure, or a list of solutions (when the request is a data query).

A solvable preferably has three parts: a *goal*, a list of *parameters*, and a list of *permissions*, which are declared using the format:

solvable(Goal, Parameters, Permissions)

The goal of a solvable, which syntactically takes the preferable form of an *ICL* structure, is a logical representation of the service provided by the solvable. (An *ICL* structure consists of a *functor* with 0 or more arguments. For example, in the structure a(b,c), `a' is the functor, and `b' and `c' the arguments.) As with a PROLOG structure, the goal's arguments themselves may preferably be structures.

Various options can be included in the parameter list, to refine the semantics associated with the solvable. The *type* parameter is preferably used to say whether the solvable is *data* or *procedure*. When the type is *procedure*, another parameter may be used to indicate the handler to be associated with the solvable. Some of the parameters appropriate for a *data* solvable are mentioned elsewhere in this application. In either case (procedure or data solvable), the *private* parameter may be preferably used to restrict the use of a solvable to the declaring agent when the agent intends the solvable to be solely for its internal use but wishes to take advantage of the mechanisms in accordance with the present invention to access it, or when the agent wants the solvable to be available to outside agents only at selected times. In support of the latter case, it is preferable for the agent to change the status of a solvable from private to non-private at any time.

The permissions of a solvable provide mechanisms by which an agent may preferably control access to its services allowing the agent to restrict calling and writing of a solvable to itself and/or other selected agents. (*Calling* means requesting the service encapsulated by a solvable, whereas *writing* means modifying the collection of facts associated with a data solvable.) The default permission for every solvable in a further preferred embodiment of the present invention is to be callable by anyone, and for data solvables to be writable by anyone. A solvable's permissions can preferably be changed at any time, by the agent providing the solvable.

For example, the solvables of a simple email agent might include:

```
     solvable(send_message(email, +ToPerson, +Params),
          [type(procedure), callback(send_mail)],
            [])
     solvable(last_message(email, -MessageId),
            [type(data), single_value(true)],
            [write(true)]),
          solvable(get_message(email, +MessageId, -
Msg),
             [type(procedure), callback(get_mail)],
         [])
```

The symbols `+' and `-', indicating input and output arguments, are at present used only for purposes of documentation. Most parameters and permissions have default values, and specifications of default values may be omitted from the parameters and permissions lists.

Defining an agent's capabilities in terms of solvable declarations effectively creates a vocabulary with which other agents can communicate with the new agent. Ensuring that agents will speak the same language and share a common, unambiguous semantics of the vocabulary involves *ontology*. Agent development tools and services (automatic translations of solvables by the facilitator) help address this issue; additionally, a preferred embodiment of the present invention will typically rely on vocabulary from either formally engineered ontologies for specific domains or from ontologies constructed during the incremental development of a body of agents for several applications or from both specific domain ontologies and incrementally developed ontologies. Several example tools and services are described in Cheyer et al.'s paper entitled "Development Tools for the Open Agent Architecture," as presented at the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 96), London, April 1996.

Although the present invention imposes no hard restrictions on the form of solvable declarations, two common usage conventions illustrate some of the utility associated with solvables.

Classes of services are often preferably tagged by a particular type. For instance, in the example above, the "last_message" and "get_message" solvables are specialized for email, not by modifying the *names* of the services, but rather by the use of the `email' parameter, which serves during the execution of an *ICL* request to select (or not) a specific type of message.

Actions are generally written using an imperative verb as the functor of the solvable in a preferred embodiment of the present invention, the direct object (or item class) as the first argument of the predicate, required arguments following, and then an extensible parameter list as the last argument. The parameter list can hold optional information usable by the function. The *ICL* expression generated by a natural language parser often makes use of this parameter list to store prepositional phrases and adjectives.

As an illustration of the above two points, "Send mail to Bob about lunch" will be translated into an *ICL* request send_message(email, `Bob Jones', [subject(lunch)]), whereas "Remind Bob about lunch" would leave the transport unspecified (send_message(KIND, `Bob Jones', [subject(lunch)])), enabling all available message transfer agents (e.g., fax, phone, mail, pager) to compete for the opportunity to carry out the request.

Requesting Services

An agent preferably requests services of the community of agent by delegating tasks or goals to its facilitator. Each request preferably contains calls to one or more agent solvables, and optionally specifies parameters containing advice to help the facilitator determine how to execute the task. Calling a solvable preferably does *not* require that the agent specify (or even know of) a particular agent or agents to handle the call. While it is possible to specify one or more agents using an address parameter (and there are situations in which this is desirable), in general it is advantageous to leave this delegation to the facilitator. This greatly reduces the hard-coded component dependencies often found in other distributed frameworks. The agent libraries of a preferred embodiment of the present invention provide an agent with a single, unified point of entry for requesting services of other agents: the library procedure *oaa_Solve*. In the style of logic programming, *oaa_Solve* may preferably be used both to retrieve data and to initiate actions, so that calling a *data* solvable looks the same as calling a *procedure* solvable.

Complex Goal Expressions

A powerful feature provided by preferred embodiments of the present invention is the ability of a client agent (or a user) to submit compound goals of an arbitrarily complex nature to a facilitator. A compound goal is a single goal expression that specifies multiple sub-goals to be performed. In speaking of a *"complex goal expression"* we mean that a single goal expression that expresses multiple sub-goals can potentially include more than one type of logical connector (e.g., AND, OR, NOT), and/or more than one level of logical nesting (e.g., use of parentheses), or the substantive equivalent. By way of further clarification, we note that when speaking of an *"arbitrarily complex goal expression"* we mean that goals are expressed in a language or syntax that allows expression of such complex goals when appropriate or when desired, not that every goal is itself necessarily complex.

It is contemplated that this ability is provided through an interagent communication language having the necessary syntax and semantics. In one example, the goals may take the form of compound goal expressions composed using operators similar to those employed by PROLOG, that is, the comma for conjunction, the semicolon for disjunction, the arrow for conditional execution, etc. The present invention also contemplates significant extensions to PROLOG syntax and semantics. For example, one embodiment incorporates a "parallel disjunction" operator indicating that the disjuncts are to be executed by different agents concurrently. A further embodiment supports the specification of whether a given sub-goal is to be executed breadth-first or depth-first.

A further embodiment supports each sub-goal of a compound goal optionally having an address and/or a set of parameters attached to it. Thus, each sub-goal takes the form

Address:Goal::Parameters

where both *Address* and *Parameters* are optional.

An address, if present, preferably specifies one or more agents to handle the given goal, and may employ several different types of referring expression: unique names, symbolic names, and shorthand names. Every agent has preferably a unique name, assigned by its facilitator, which relies upon network addressing schemes to ensure its global uniqueness. Preferably, agents also have self-selected symbolic names (for example, "mail"), which are not guaranteed to be unique. When an address includes a symbolic name, the facilitator preferably takes this to mean that all agents having that name should be called upon. Shorthand names include `self' and `parent' (which refers to the agent's facilitator). The address associated with a goal or sub-goal is preferably always optional. When an address is not present, it is the facilitator's job to supply an appropriate address.

The distributed execution of compound goals becomes particularly powerful when used in conjunction with natural language or speech-enabled interfaces, as the query itself may specify how functionality from distinct agents will be combined. As a simple example, the spoken utterance "Fax it to Bill Smith's manager." can be translated into the following compound *ICL* request:

oaa_Solve((manager('Bill Smith', M), fax(it,M,[])), [strategy(action)])

Note that in this ICL request there are two sub-goals, "manager('Bill Smith',M)" and "fax(it,M,[])," and a single global parameter "strategy(action)." According to the present invention, the facilitator is capable of mapping global parameters in order to apply the constraints or advice across the separate sub-goals in a meaningful way. In this instance, the global parameter strategy(action) implies a parallel constraint upon the first sub-goal; i.e., when there are multiple agents that can respond to the manager sub-goal, each agent should receive a request for service. In contrast, for the second sub-goal, parallelism should not be inferred from the global parameter strategy(action) because such an inference would possibly result in the transmission of duplicate facsimiles.

Refining Service Requests

In a preferred embodiment of the present invention, parameters associated with a goal (or sub-goal) can draw on useful features to refine the request's meaning. For example, it is frequently preferred to be able to specify whether or not solutions are to be returned synchronously; this is done using the *reply* parameter, which can take any of the values *synchronous, asynchronous*, or *none*. As another example, when the goal is a non-compound query of a data solvable, the *cache* parameter may preferably be used to request local caching of the facts associated with that solvable. Many of the remaining parameters fall into two categories: feedback and advice.

Feedback parameters allow a service requester to receive information from the facilitator about how a goal was handled. This feedback can include such things as the identities of the agents involved in satisfying the goal, and the amount of time expended in the satisfaction of the goal.

Advice parameters preferably give constraints or guidance to the facilitator in completing and interpreting the goal. For example, a *solution_limit* parameter preferably allows the requester to say how many solutions it is interested in; the facilitator and/or service providers are free to use this information in optimizing their efforts. Similarly, a *time_limit* is preferably used to say how long the requester is willing to wait for solutions to its request, and, in a multiple facilitator system, a *level_limit* may preferably be used to say how remote the facilitators may be that are consulted in the search for solutions. A *priority* parameter is preferably used to indicate that a request is more urgent than previous requests that have not yet been satisfied. Other preferred advice parameters include but are not limited to parameters used to tell the facilitator whether parallel satisfaction of the parts of a goal is appropriate, how to combine and filter results arriving from multiple solver agents, and whether the requester itself may be considered a candidate solver of the sub-goals of a request.

Advice parameters preferably provide an extensible set of low-level, orthogonal parameters capable of combining with the *ICL* goal language to fully express how information should flow among participants. In certain preferred embodiments of the present invention, multiple parameters can be grouped together and given a group name. The resulting *high-level advice parameters* can preferably be used to express concepts analogous to KQML's performatives, as well as define classifications of problem types. For instance, KQML's "ask_all" and "ask_one" performatives would be represented as combinations of values given to the parameters *reply*, *parallel_ok*, and *solution_limit*. As an example of a higher-level problem type, the strategy "math_problem" might preferably send the query to all appropriate math solvers in parallel, collect their responses, and signal a conflict if different answers are returned. The strategy "essay_question" might preferably send the request to all appropriate participants, and signal a problem (i.e., cheating) if any of the returned answers are identical.

Facilitation

In a preferred embodiment of the present invention, when a facilitator receives a compound goal, its job is to construct a goal satisfaction plan and oversee its satisfaction in an optimal or near optimal manner that is consistent with the specified advice. The facilitator of the present invention maintains a knowledge base that records the capabilities of a collection of agents, and uses that knowledge to assist requesters and providers of services in making contact.

Figure 7 schematically shows data structures 700 internal to a facilitator in accordance with one embodiment of the present invention. Consider the function of a Agent Registry 702 in the present invention. Each registered agent may be seen as associated with a collection of fields found within its parent facilitator such as shown in the figure. Each registered agent may optionally possess a Symbolic Name which would be entered into field 704. As mentioned elsewhere, Symbolic Names need not be unique to each instance of an agent. Note that an agent may in certain preferred embodiments of the present invention possess more than one Symbolic Name. Such Symbolic Names would each be found through their associations in the Agent Registry entries. Each agent, when registered, must possess a Unique Address, which is entered into the Unique Address field 706.

With further reference to Figure 7, each registered agent may be optionally associated with one or more capabilities, which have associated Capability Declaration fields 708 in the parent facilitator Agent Registry 702. These capabilities may define not just functionality, but may further provide a utility parameter indicating, in some manner (e.g., speed, accuracy, etc), how effective the agent is at providing the declared capability. Each registered agent may be optionally associated with one or more data components, which have associated Data Declaration fields 710 in the parent facilitator Agent Registry 702. Each registered agent may be optionally associated with one or more triggers, which preferably could be referenced through their associated Trigger Declaration fields 712 in the parent facilitator Agent Registry 702. Each registered agent may be optionally associated with one or more tasks, which preferably could be referenced through their associated Task Declaration fields 714 in the parent facilitator Agent Registry 702. Each registered agent may be optionally associated with one or more Process Characteristics, which preferably could be referenced through their associated Process Characteristics Declaration fields 716 in the parent facilitator Agent Registry 702. Note that these characteristics in certain preferred embodiments of the present invention may include one or more of the following: Machine Type (specifying what type of computer may run the agent), Language (both computer and human interface).

A facilitator agent in certain preferred embodiments of the present invention further includes a Global Persistent Database 720. The database 720 is composed of data elements which do not rely upon the invocation or instantiation of client agents for those data elements to persist. Examples of data elements which might be present in such a database include but are not limited to the network address of the facilitator agent's server, facilitator agent's server accessible network port list, firewalls, user lists, and security options regarding the access of server resources accessible to the facilitator agent.

A simplified walk through of operations involved in creating a client agent, a client agent initiating a service request, a client agent responding to a service request and a facilitator agent responding to a service request are including hereafter by way of illustrating the use of such a system. These figures and their accompanying discussion are provided by way of illustration of one preferred embodiment of the present invention and are not intended to limit the scope of the present invention.

Figure 8 depicts operations involved in instantiating a client agent with its parent facilitator in accordance with a preferred embodiment of the present invention. The operations begin with starting the Agent Registration in a step 800. In a next step 802, the Installer, such as a client or facilitator agent, invokes a new client agent. It will be appreciated that any computer entity is capable of invoking a new agent. The system then instantiates the new client agent in a step 804. This operation may involve resource allocations somewhere in the network on a local computer system for the client agent, which will often include memory as well as placement of references to the newly instantiated client agent in internal system lists of agents within that local computing system. Once instantiated, the new client and its parent facilitator establish a communications link in a step 806. In certain preferred embodiments, this communications link involves selection of one or more physical transport mechanisms for this communication. Once established, the client agent transmits it profile to the parent facilitator in a step 808. When received, the parent facilitator registers the client agent in a step 810. Then, at a step 812, a client agent has been instantiated in accordance with one preferred embodiment of the present invention.

Figure 9 depicts operations involved in a client agent initiating a service request and receiving the response to that service request in accordance with a preferred embodiment of the present invention. The method of Figure 9 begins in a step 900, wherein any initialization or other such procedures may be performed. Then, in a step 902, the client agent determines a goal to be achieved (or solved). This goal is then translated in a step 904 into *ICL*, if it is not already formulated in it. The goal, now stated in *ICL*, is then transmitted to the client agent's parent facilitator in a step 906. The parent facilitator responds to this service request and at a later time, the client agent receives the results of the request in a step 908, operations of Figure 9 being complete in a done step 910.

FIGURE 10 depicts operations involved in a client agent responding to a service request in accordance with a preferred embodiment of the present invention. Once started in a step 1000, the client agent receives the service request in a step 1002. In a next step 1004, the client agent parses the received request from ICL. The client agent then determines if the service is available in a step 1006. If it is not, the client agent returns a status report to that effect in a step 1008. If the service is available, control is passed to a step 1010 where the client performs the requested service. Note that in completing step 1010 the client may form complex goal expressions, requesting results for these solvables from the facilitator agent. For example, a fax agent might fax a document to a certain person only after requesting and receiving a fax number for that person. Subsequently, the client agent either returns the results of the service and/or a status report in a step 1012. The operations of Figure 10 are complete in a done step 1014.

FIGURE 11 depicts operations involved in a facilitator agent response to a service request in accordance with a preferred embodiment of the present invention. The start of such operations in step 1100 leads to the reception of a goal request in a step 1102 by the facilitator. This request is then parsed and interpreted by the facilitator in a step 1104. The facilitator then proceeds to construct a goal satisfaction plan in a next step 1106. In steps 1108 and 1110, respectively, the facilitator determines the required sub-goals and then selects agents suitable for performing the required sub-goals. The facilitator then transmits the sub-goal requests to the selected agents in a step 1112 and receives the results of these transmitted requests in a step 1114. It should be noted that the actual implementation of steps 1112 and 1114 are dependent upon the specific goal satisfaction plan. For instance, certain sub-goals may be sent to separate agents in parallel, while transmission of other sub-goals may be postponed until receipt of particular answers. Further, certain requests may generate multiple responses that generate additional sub-goals. Once the responses have been received, the facilitator determines whether the original requested goal has been completed in a step 1118. If the original requested goal has not been completed, the facilitator recursively repeats the operations 1106 through 1116. Once the original requested goal is completed, the facilitator returns the results to the requesting agent 1118 and the operations are done at 1120.

A further preferred embodiment of the present invention incorporates *transparent delegation,* which means that a requesting agent can generate a request, and a facilitator can manage the satisfaction of that request, without the requester needing to have any knowledge of the identities or locations of the satisfying agents. In some cases, such as when the request is a data query, the requesting agent may also be oblivious to the *number* of agents involved in satisfying a request. Transparent delegation is possible because agents' capabilities (solvables) are treated as an abstract description of a service, rather than as an entry point into a library or body of code.

A further preferred embodiment of the present invention incorporates facilitator handling of compound goals, preferably involving three types of processing: delegation, optimization and interpretation.

Delegation processing preferably supports facilitator determination of which specific agents will execute a compound goal and how such a compound goal's sub-goals will be combined and the sub-goal results routed. *Delegation* involves selective application of global and local constraint and advice parameters onto the specific sub-goals. *Delegation* results in a goal that is unambiguous as to its meaning and as to the agents that will participate in satisfying it.

Optimization processing of the completed goal preferably includes the facilitator using sub-goal parallelization where appropriate. *Optimization* results in a goal whose interpretation will require as few exchanges as possible, between the facilitator and the satisfying agents, and can exploit parallel efforts of the satisfying agents, wherever this does not affect the goal's meaning.

Interpretation processing of the optimized goal. Completing the addressing of a goal involves the selection of one or more agents to handle each of its sub-goals (that is, each sub-goal for which this selection has not been specified by the requester). In doing this, the facilitator uses its knowledge of the capabilities of its client agents (and possibly of other facilitators, in a multi-facilitator system). It may also use strategies or advice specified by the requester, as explained below. The *interpretation* of a goal involves the coordination of requests to the satisfying agents, and assembling their responses into a coherent whole, for return to the requester.

A further preferred embodiment of present invention extends facilitation so the facilitator can employ strategies and advice given by the requesting agent, resulting in a variety of interaction patterns that may be instantiated in the satisfaction of a request.

A further preferred embodiment of present invention handles the distribution of both data update requests and requests for installation of triggers, preferably using some of the same strategies that are employed in the delegation of service requests.

Note that the reliance on facilitation is not absolute; that is, there is no hard requirement that requests and services be matched up by the facilitator, or that interagent communications go through the facilitator. There is preferably support in the agent library for explicit addressing of requests. However, a preferred embodiment of the present invention encourages employment the paradigm of agent communities, minimizing their development effort, by taking advantage of the facilitator's provision of transparent delegation and handling of compound goals.

A facilitator is preferably viewed as a *coordinator*, not a controller, of cooperative task completion. A facilitator preferably never initiates an activity. A facilitator preferably responds to requests to manage the satisfaction of some goal, the update of some data repository, or the installation of a trigger by the appropriate agent or agents. All agents can preferably take advantage of the facilitator's expertise in delegation, and its up-to-date knowledge about the current membership of a dynamic community. The facilitator's coordination services often allows the developer to lessen the complexity of individual agents, resulting in a more manageable software development process, and enabling the creation of lightweight agents.

Maintaining Data Repositories

The agent library supports the creation, maintenance, and use of databases, in the form of data solvables. Creation of a data solvable requires only that it be declared. Querying a data solvable, as with access to any solvable, is done using *oaa_Solve*.

A data solvable is conceptually similar to a relation in a relational database. The facts associated with each solvable are maintained by the agent library, which also handles incoming messages containing queries of data solvables. The default behavior of an agent library in managing these facts may preferably be refined, using parameters specified with the solvable's declaration. For example, the parameter *single_value* preferably indicates that the solvable should only contain a single fact at any given point in time. The parameter *unique_values* preferably indicates that no duplicate values should be stored.

Other parameters preferably allow data solvables use of the concepts of ownership and persistence. For implementing shared repositories, it is often preferable to maintain a record of which agent created each fact of a data solvable with the creating agent being preferably considered the fact's owner. In many applications, it is preferable to remove an agent's facts when that agent goes offline (for instance, when the agent is no longer participating in the agent community, whether by deliberate termination or by malfunction). When a data solvable is declared to be non-persistent, its facts are automatically maintained in this way, whereas a persistent data solvable preferably retains its facts until they are explicitly removed.

A further preferred embodiment of present invention supports an agent library through procedures by which agents can update (add, remove, and replace) facts belonging to data solvables, either locally or on other agents, given that they have preferably the required permissions. These procedures may preferably be refined using many of the same parameters that apply to service requests. For example, the *address* parameter preferably specifies one or more particular agents to which the update request applies. In its absence, just as with service requests, the update request preferably goes to *all* agents providing the relevant data solvable. This default behavior can be used to maintain coordinated "mirror" copies of a data set within multiple agents, and can be useful in support of distributed, collaborative activities.

Similarly, the *feedback* parameters, described in connection with *oaa_Solve*, are preferably available for use with data maintenance requests.

A further preferred embodiment of present invention supports ability to provide data solvables not just to client agents, but also to facilitator agents. Data solvables can preferably created, maintained and used by a facilitator. The facilitator preferably can, at the request of a client of the facilitator, create, maintain and share the use of data solvables with all the facilitator's clients. This can be useful with relatively stable collections of agents, where the facilitator's workload is predictable.

Using a Blackboard Style of Communication

In a further preferred embodiment of present invention, when a data solvable is publicly readable and writable, it acts essentially as a global data repository and can be used cooperatively by a group of agents. In combination with the use of triggers, this allows the agents to organize their efforts around a "blackboard" style of communication.

As an example, the "DCG-NL" agent (one of several existing natural language processing agents), provides natural language processing services for a variety of its peer agents, expects those other agents to record, on the facilitator, the vocabulary to which they are prepared to respond, with an indication of each word's part of speech, and of the logical form (*ICL* sub-goal) that should result from the use of that word. In a further preferred embodiment of present invention, the NL agent, preferably when it comes online, preferably installs a data solvable for each basic part of speech on its facilitator. For instance, one such solvable would be:

solvable(noun(Meaning, Syntax), [], [])

Note that the empty lists for the solvable's permissions and parameters are acceptable here, since the default permissions and parameters provide appropriate functionality.

A further preferred embodiment of present invention incorporating an Office Assistant system as discussed herein or similar to the discussion here supports several agents making use of these or similar services. For instance, the database agent uses the following call, to library procedure *oaa_AddData*, to post the noun `boss', and to indicate that the "meaning" of boss is the concept `manager':

oaa_AddData(noun(manager, atom(boss)), [address(parent)])

Autonomous Monitoring with Triggers

A further preferred embodiment of present invention includes support for triggers, providing a general mechanism for requesting some action be taken when a set of conditions is met. Each agent can preferably install triggers either locally, for itself, or remotely, on its facilitator or peer agents. There are preferably at least four types of triggers: communication, data, task, and time. In addition to a type, each trigger preferably specifies at least a condition and an action, both preferably expressed in *ICL*. The condition indicates under what circumstances the trigger should fire, and the action indicates what should happen when it fires. In addition, each trigger can be set to fire either an unlimited number of times, or a specified number of times, which can be any positive integer.

Triggers can be used in a variety of ways within preferred embodiments of the present invention. For example, triggers can be used for monitoring external sensors in the execution environment, tracking the progress of complex tasks, or coordinating communications between agents that are essential for the synchronization of related tasks. The installation of a trigger within an agent can be thought of as a representation of that agent's *commitment* to carry out the specified action, whenever the specified condition holds true.

Communication triggers preferably allow any incoming or outgoing event (message) to be monitored. For instance, a simple communication trigger may say something like: "Whenever a solution to a goal is returned from the facilitator, send the result to the presentation manager to be displayed to the user."

Data triggers preferably monitor the state of a data repository (which can be maintained on a facilitator or a client agent). Data triggers' conditions may be tested upon the addition, removal, or replacement of a fact belonging to a data solvable. An example data trigger is: "When 15 users are simultaneously logged on to a machine, send an alert message to the system administrator."

Task triggers preferably contain conditions that are tested after the processing of each incoming event and whenever a timeout occurs in the event polling. These conditions may specify any goal executable by the local *ICL* interpreter, and most often are used to test when some solvable becomes satisfiable. Task triggers are useful in checking for task-specific internal conditions. Although in many cases such conditions are captured by solvables, in other cases they may not be. For example, a mail agent might watch for new incoming mail, or an airline database agent may monitor which flights will arrive later than scheduled. An example task trigger is: "When mail arrives for me about security, notify me immediately."

Time triggers preferably monitor time conditions. For instance, an alarm trigger can be set to fire at a single fixed point in time (e.g., "On December 23rd at 3pm"), or on a recurring basis (e.g., "Every three minutes from now until noon").

Triggers are preferably implemented as data solvables, declared implicitly for every agent. When requesting that a trigger be installed, an agent may use many of the same parameters that apply to service and data maintenance requests.

A further preferred embodiment of present invention incorporates semantic support, in contrast with most programming methodologies, of the agent on which the trigger is installed only having to know how to evaluate the conditional part of the trigger, not the consequence. When the trigger fires, the action is delegated to the facilitator for execution. Whereas many commercial mail programs allow rules of the form "When mail arrives about XXX, [forward it, delete it, archive it]", the possible actions are hard-coded and the user must select from a fixed set.

A further preferred embodiment of present invention, the consequence of a trigger may be any compound goal executable by the dynamic community of agents. Since new agents preferably define both functionality and vocabulary, when an unanticipated agent (for example, a fax agent) joins the community, no modifications to existing code is required for a user to make use of it - "When mail arrives, fax it to Bill Smith."

The Agent Library

In a preferred embodiment of present invention, the agent library provides the infrastructure for constructing an agent-based system. The essential elements of protocol (involving the details of the messages that encapsulate a service request and its response) are preferably made transparent to simplify the programming applications. This enables the developer to focus functionality, rather than message construction details and communication details. For example, to request a service of another agent, an agent preferably calls the library procedure *oaa_Solve*. This call results in a message to a facilitator, which will exchange messages with one or more service providers, and then send a message containing the desired results to the requesting agent. These results are returned via one of the arguments of *oaa_Solve*. None of the messages involved in this scenario is explicitly constructed by the agent developer. Note that this describes the *synchronous* use of *oaa_Solve*.

In another preferred embodiment of present invention, an agent library provides both *intra*agent and *inter*agent infrastructure; that is, mechanisms supporting the internal structure of individual agents, on the one hand, and mechanisms of cooperative interoperation between agents, on the other. Note that most of the infrastructure cuts across this boundary with many of the same mechanisms supporting both agent internals and agent interactions in an integrated fashion. For example, services provided by an agent preferably can be accessed by that agent through the same procedure (*oaa_Solve*) that it would employ to request a service of another agent (the only difference being in the *address* parameter accompanying the request). This helps the developer to reuse code and avoid redundant entry points into the same functionality.

Both of the preferred characteristics described above (transparent construction of messages and integration of *intra*agent with *inter*agent mechanisms) apply to most other library functionality as well, including but not limited to data management and temporal control mechanisms.

Source Code Appendix

Source code for version 2.0 of the*OAA* software product is included as an appendix hereto, and is incorporated herein by reference. The code includes an agent library, which provides infrastructure for constructing an agent-based system. The library's several families of procedures provide the functionalities discussed above, as well as others that have not been discussed here but that will be sufficiently clear to the interested practitioner. For example, declarations of an agent's solvables, and their registration with a facilitator, are managed using procedures such as *oaa_Declare*, *oaa_Undeclare*, and *oaa_Redeclare*. Updates to data solvables can be accomplished with a family of procedures including *oaa_AddData*, *oaa_RemoveData*, and *oaa_ReplaceData*. Similarly, triggers are maintained using procedures such as *oaa_AddTrigger*, *oaa_RemoveTrigger*, and *oaa_ReplaceTrigger*. The provided source code also includes source code for an OAA Facilitator Agent.

The source code appendix is offered solely as a means of further helping practitioners to construct a preferred embodiment of the invention. By no means is the source code intended to limit the scope of the present invention.

Illustrative Applications

To further illustrate the technology of the preferred embodiment, we will next present and discuss two sample applications of the present inventions.

Unified Messaging

A further preferred embodiment of present invention incorporates a Unified Messaging application extending the Automated Office application presented previously herein with an emphasis on ubiquitous access and dynamic presentation of the information and services supported by the agent community. The agents used in this application are depicted in Figure 12.

A hypothetical example of realistic dialog using a preferred embodiment of the present invention can provide insight into how systems may preferably be built using the present invention. In this scenario. the user, with only a telephone as an interface, is planning a trip to Boston where he will soon give a presentation. Capitalized sentences are phrases spoken by the user into the telephone and processed by a phone agent 452.

Responses, unless otherwise indicated, are spoken by the system using text-to-speech generation agent 454.

```
1.1  Welcome to SRI International.  Please enter your
user ID and password.
     <User enters touchtone ID and password>

          Good to see you again Adam Cheyer.  I am
listening to you.
```

Every user interface agent 408, including the telephone agent 452, should know the identify of its user. This information is used in resolving anaphoric references such as "me" and "I", and allows multiple user interfaces operated by the same user to work together.

```
1.2  WHAT IS TODAY'S SCHEDULE?
     Here is today's schedule for Adam Cheyer:
       At 10am for 1 hour, meeting with Dave.
       At 3pm for 1 hour, presentation about software
agents.

          End of schedule.
```

If the user is operating both a graphical user interface and a telephone, as described in conjunction with the Automated Office application, the result of this spoken request is to display a calendar window containing the current schedule. In this case, with no graphical display available, the GEN_NL agent 1202 is tasked to produce a spoken response that can be played over the phone. GEN_NL shares the same dynamic vocabulary and phrasal rules as the natural language parser DCG_NL 426, and contains strategies for producing responses to queries using either simple or list-based multimedia utterances.

```
1.3  FIND FRIDAY'S WEATHER IN BOSTON.
     The weather in Boston for Friday is as follows:
       Sunny in the morning. Partly cloudy in the
afternoon with a 20
       percent chance of thunderstorms late. Highs in the
mid 70s.
```

In addition to data accessible from legacy applications, content may be retrieved by web-reading agents which provide wrappers around useful websites.

```
1.4  FIND ALL NEW MAIL MESSAGES.
     There are 2 messages available.
     Message 1, from Mark Tierny, entitled "OAA meeting."
1.5  NEXT MESSAGE
     Message 2, from Jennifer Schwefler, entitled
"Presentation Summary."
1.6  PLAY IT.
     This message is a multipart MIME-encoded message.
There are two parts.
     Part 1.  (Voicemail message, not text-to speech):
     Thanks for taking part as a speaker in our
conference.
     The schedule will be posted soon on our homepage.
1.7  NEXT PART
     Part 2. (read using text-to-speech):
     The presentation home page is http://www....
1.8  PRINT MESSAGE
     Command executed.
```

Mail messages are no longer just simple text documents, but often consist of multiple subparts containing audio files, pictures, webpages, attachments and so forth. When a user asks to play a complex email message over the telephone, many different agents may be implicated in the translation process, which would be quite different given the request "print it." The challenge is to develop a system which will enable agents to cooperate in an extensible, flexible manner that alleviates explicit coding of agent interactions for every possible input/output combination.

In a preferred embodiment of the present invention, each agent concentrates only on what it can do and on what it knows, and leaves other work to be delegated to the agent community. For instance, a printer agent 1204, defining the solvable print(Object,Parameters), can be defined by the following pseudo-code, which basically says, "If someone can get me a document, in either POSTSCRIPT or text form, I can print it.".

```
print(Object, Parameters) {
   ' If Object is reference to "it", find an appropriate
document
   if (Object = "ref(it)")
      oaa_Solve(resolve_reference(the, document, Params,
Object),[]);
   ' Given a reference to some document, ask for the
document in POSTSCRIPT
   if (Object = "id(Pointer)")
      oaa_Solve(resolve_id_as(id(Pointer), postscript,
[], Object),[]);
   ' If Object is of type text or POSTSCRIPT, we can
print it.
   if ((Object is of type Text) or (Object is of type
Postscript))
      do_print(Object);
}
```

In the above example, since an email message is the salient document, the mail agent 442 will receive a request to produce the message as POSTSCRIPT. Whereas the mail agent 442 may know how to save a text message as POSTSCRIPT, it will not know what to do with a webpage or voicemail message. For these parts of the message, it will simply send oaa_Solve requests to see if another agent knows how to accomplish the task.

Until now, the user has been using only a telephone as user interface. Now, he moves to his desktop, starts a web browser 436, and accesses the URL referenced by the mail message.

```
1.9  RECORD MESSAGE
     Recording voice message.  Start speaking now.
1.10 THIS IS THE UPDATED WEB PAGE CONTAINING THE
PRESENTATION SCHEDULE.
     Message one recorded.
1.11 IF THIS WEB PAGE CHANGES, GET IT TO ME WITH NOTE
ONE.
     Trigger added as requested.
```

In this example, a local agent 436 which interfaces with the web browser can return the current page as a solution to the request "oaa_Solve(resolve_reference(this, web_page, [], Ref),[])", sent by the NL agent 426. A trigger is installed on a web agent 436 to monitor changes to the page, and when the page is updated, the notify agent 446 can find the user and transmit the webpage and voicemail message using the most appropriate media transfer mechanism.

This example based on the Unified Messaging application is intended to show how concepts in accordance with the present invention can be used to produce a simple yet extensible solution to a multi-agent problem that would be difficult to implement using a more rigid framework. The application supports adaptable presentation for queries across dynamically changing, complex information; shared context and reference resolution among applications; and flexible translation of multimedia data. In the next section, we will present an application which highlights the use of parallel competition and cooperation among agents during multi-modal fusion.

Multimodal Map

A further preferred embodiment of present invention incorporates the Multimodal Map application. This application demonstrates natural ways of communicating with a community of agents, providing an interactive interface on which the user may draw, write or speak. In a travel-planning domain illustrated by Figure 13, available information includes hotel, restaurant, and tourist-site data retrieved by distributed software agents from commercial Internet sites. Some preferred types of user interactions and multimodal issues handled by the application are illustrated by a brief scenario featuring working examples taken from the current system.

Sara is planning a business trip to San Francisco, but would like to schedule some activities for the weekend while she is there. She turns on her laptop PC, executes a map application, and selects San Francisco.

2.1 [Speaking] Where is downtown?

Map scrolls to appropriate area.

2.2 [Speaking and drawing region] Show me all hotels near here.

Icons representing hotels appear.

2.3 [Writes on a hotel] Info?

A textual description (price, attributes, etc.) appears.

2.4 [Speaking] I only want hotels with a pool.

Some hotels disappear.

2.5 [Draws a crossout on a hotel that is too close to a highway]

Hotel disappears

2.6 [Speaking and circling] Show me a photo of this hotel.

Photo appears.

2.7 [Points to another hotel]

Photo appears.

2.8 [Speaking] Price of the other hotel?

Price appears for previous hotel.

2.9 [Speaking and drawing an arrow] Scroll down.

Display adjusted.

2.10 [Speaking and drawing an arrow toward a hotel]

What is the distance from this hotel to Fisherman's Wharf?

Distance displayed.

2.11 [Pointing to another place and speaking] And the distance to here?

Distance displayed.

Sara decides she could use some human advice. She picks up the phone, calls Bob, her travel agent, and writes Start collaboration to synchronize his display with hers. At this point, both are presented with identical maps, and the input and actions of one will be remotely seen by the other.

3.1 [Sara speaks and circles two hotels]

Bob, I'm trying to choose between these two hotels. Any opinions?

3.2 [Bob draws an arrow, speaks, and points]

Well, this area is really nice to visit. You can walk there from

this hotel.
Map scrolls to indicated area. Hotel selected.
3.3 [Sara speaks] Do you think I should visit Alcatraz?
3.4 [Bob speaks] Map, show video of Alcatraz.
Video appears.
3.5 [Bob speaks] Yes, Alcatraz is a lot of fun.

A further preferred embodiment of present invention generates the most appropriate interpretation for the incoming streams of multimodal input. Besides providing a user interface *to* a dynamic set of distributed agents, the application is preferably built *using* an agent framework. The present invention also contemplates aiding the coordinate competition and cooperation among information sources, which in turn works in parallel to resolve the ambiguities arising at every level of the interpretation process: *low-level processing of the data stream, anaphora resolution, cross-modality influences* and *addressee.*

Low-level processing of the data stream: Pen input may be preferably interpreted as a gesture (e.g., 2.5: cross-out) by one algorithm, or as handwriting by a separate recognition process (e.g., 2.3: "info?"). Multiple hypotheses may preferably be returned by a modality recognition component.

Anaphora resolution: When resolving anaphoric references, separate information sources may contribute to resolving the reference: context by object type, deictic, visual context, database queries, discourse analysis. An example of information provided through context by object type is found in interpreting an utterance such as "show photo of the hotel", where the natural language component can return a list of the last hotels talked about. Deictic information in combination with a spoken utterance like "show photo of this hotel" may preferably include pointing, circling, or arrow gestures which might indicate the desired object (e.g., 2.7). Deictic references may preferably occur before, during, or after an accompanying verbal command. Information provided in a visual context, given for the request "display photo of the hotel" may preferably include the user interface agent might determine that only one hotel is currently visible on the map, and therefore this might be the desired reference object. Database queries preferably involving information from a database agent combined with results from other resolution strategies. Examples are "show me a photo of the hotel in Menlo Park" and

2.2. Discourse analysis preferably provides a source of information for phrases such as "No, the other one" (or 2.8).

The above list of preferred anaphora resolution mechanisms is not exhaustive. Examples of other preferred resolution methods include but are not limited to spatial reasoning ("the hotel between Fisherman's Wharf and Lombard Street") and user preferences ("near my favorite restaurant").

Cross-modality influences: When multiple modalities are used together, one modality may preferably reinforce or remove or diminish ambiguity from the interpretation of another. For instance, the interpretation of an arrow gesture may vary when accompanied by different verbal commands (e.g., "scroll left" vs. "show info about this hotel"). In the latter example, the system must take into account how accurately and unambiguously an arrow selects a single hotel.

Addressee: With the addition of collaboration technology, humans and automated agents all share the same workspace. A pen doodle or a spoken utterance may be meant for either another human, the system (3.1), or both (3.2).

The implementation of the Multimodal Map application illustrates and exploits several preferred features of the present invention: · reference resolution and task delegation by parallel parameters of oaa_Solve, basic multi-user collaboration handled through built-in data management services, additional functionality readily achieved by adding new agents to the community, domain-specific code cleanly separated from other agents.

A further preferred embodiment of present invention provides reference resolution and task delegation handled in a distributed fashion by the parallel parameters of oaa_Solve, with meta-agents encoding rules to help the facilitator make context- or user-specific decisions about priorities among knowledge sources.

A further preferred embodiment of present invention provides basic multi-user collaboration handled through at least one built-in data management service. The map user interface preferably publishes data solvables for elements such as icons, screen position, and viewers, and preferably defines these elements to have the attribute "shareable". For every update to this public data, the changes are preferably automatically replicated to all members of the collaborative session, with associated callbacks producing the visible effect of the data change (e.g., adding or removing an icon).

Functionality for recording and playback of a session is preferably implemented by adding agents as members of the collaborative community. These agents either record the data changes to disk, or read a log file and replicate the changes in the shared environment.

The domain-specific code for interpreting travel planning dialog is preferably separated from the speech, natural language, pen recognition, database and map user interface agents. These components were preferably reused without modification to add multimodal map capabilities to other applications for activities such as crisis management, multi-robot control, and the MVIEWS tools for the video analyst.

Improved Scalability and Fault Tolerance

Implementations of a preferred embodiment of present invention which rely upon simple, single facilitator architectures may face certain limitations with respect to scalability, because the single facilitator may become a communications bottleneck and may also represent a single, critical point for system failure.

Multiple facilitator systems as disclosed in the preferred embodiments to this point can be used to construct peer-to-peer agent networks as illustrated in Figure 14. While such embodiments are scalable, they do possess the potential for communication bottlenecks as discussed in the previous paragraph and they further possess the potential for reliability problems as central, critical points of vulnerability to systems failure.

A further embodiment of present invention supports a facilitator implemented as an agent like any other, whereby multiple facilitator network topologies can be readily constructed. One example configuration (but not the only possibility) is a hierarchical topology as depicted in Figure 15, where a top level Facilitator manages collections of both client agents 1508 and other Facilitators, 1504 and 1506. Facilitator agents could be installed for individual users, for a group of users, or as appropriate for the task.

Note further, that network work topologies of facilitators can be seen as graphs where each node corresponds to an instance of a facilitator and each edge connecting two or more nodes corresponds to a transmission path across one or more physical transport mechanisms. Some nodes may represent facilitators and some nodes may represent clients. Each node can be further annotated with attributes corresponding to include triggers, data, capabilities but not limited to these attributes.

A further embodiment of present invention provides enhanced scalability and robustness by separating the planning and execution components of the facilitator. In contrast with the centralized facilitation schemes described above, the facilitator system 1600 of Figure 16 separates the registry/planning component from the execution component. As a result, no single facilitator agent must carry all communications nor does the failure of a single facilitator agent shut down the entire system.

Turning directly to Figure 16, the facilitator system 1600 includes a registry/planner 1602 and a plurality of client agents 1612-1616. The registry/planner 1604 is typically replicated in one or more locations accessible by the client agents. Thus if the registry/planner 1604 becomes unavailable, the client agents can access the replicated registry/planner(s).

This system operates, for example, as follows. An agent transmits a goal 1610 to the registry planner 1602. The registry/planner 1604 translates the goal into an unambiguous execution plan detailing how to accomplish any sub-goals developed from the compound goal, as well as specifying the agents selected for performing the sub-goals. This execution plan is provided to the requesting agent which in turn initiates peer-to-peer interactions 1618 in order to implement the detailed execution plan, routing and combining information as specified within the execution plan. Communication is distributed thus decreasing sensitivity of the system to bandwidth limitations of a single facilitator agent. Execution state is likewise distributed thus enabling system operation even when a facilitator agent fails.

Further embodiments of present invention incorporate into the facilitator functionality such as load-balancing, resource management, and dynamic configuration of agent locations and numbers, using (for example) any of the topologies discussed. Other embodiments incorporate into a facilitator the ability to aid agents in establishing peer-to-peer communications. That is, for tasks requiring a sequence of exchanges between two agents, the facilitator assist the agents in finding one another and establishing communication, stepping out of the way while the agents communicate peer-to-peer over a direct, perhaps dedicated channel.

Further preferred embodiments of the present invention incorporate mechanisms for basic transaction management, such as periodically saving the state of agents (both facilitator and client) and rolling back to the latest saved state in the event of the failure of an agent.

IN THE CLAIMS:

1. A computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of:

registering a description of each active client agent's functional capabilities, using an expandable, platform-independent, inter-agent language;

receiving a request for service as a base goal in the inter-agent language, in the form of an arbitrarily complex goal expression; and

dynamically interpreting the goal expression, said act of interpreting further comprising:

generating one or more sub-goals using the inter-agent language; and

dispatching each of the sub-goals to a selected client agent for performance, based on a match between the sub-goal being dispatched and the registered functional capabilities of the selected client agent.

2. A computer-implemented method as recited in claim 1, further including the following acts of:

receiving a new request for service as a base goal using the inter-agent language, in the form of another arbitrarily complex goal expression, from at least one of the selected client agents in response to the sub-goal dispatched to said agent; and

recursively applying the last step of claim 1 in order to perform the new request for service.

3. A computer implemented method as recited in claim 2 wherein the act of registering a specific agent further includes:

invoking the specific agent in order to activate the specific agent;

instantiating an instance of the specific agent; and

transmitting the new agent profile from the specific agent to the facilitator agent in response to the instantiation of the specific agent.

4. A computer implemented method as recited in claim 1 further including the act of deactivating a specific client agent no longer available to provide services by deleting the registration of the specific client agent.

5. A computer implemented method as recited in claim 1 further comprising the act of providing an agent registry data structure.

6. A computer implemented method as recited in claim 5 wherein the agent registry data structure includes at least one symbolic name for each active agent.

7. A computer implemented method as recited in claim 5 wherein the agent registry data structure includes at least one data declaration for each active agent.

8. A computer implemented method as recited in claim 5 wherein the agent registry data structure includes at least one trigger declaration for one active agent.

9. A computer implemented method as recited in claim 5 wherein the agent registry data structure includes at least one task declaration, and process characteristics for each active agent.

10. A computer implemented method as recited in claim 5 wherein the agent registry data structure includes at least one process characteristic for each active agent.

11. A computer implemented method as recited in claim 1 further comprising the act of establishing communication between the plurality of distributed agents.

12. A computer implemented method as recited in claim 1 further comprising the acts of:

receiving a request for service in a second language differing from the inter-agent language;

selecting a registered agent capable of converting the second language into the inter-agent language; and

forwarding the request for service in a second language to the registered agent capable of converting the second language into the inter-agent language, implicitly requesting that such a conversion be performed and the results returned.

13. A computer implemented method as recited in claim 12 wherein the request includes a natural language query, and the registered agent capable of converting the second language into the inter-agent language service is a natural language agent.

14. A computer implemented method as recited in claim 13 wherein the natural language query was generated by a user interface agent.

15. A computer implemented method as recited in claim 1, wherein the base goal requires setting a trigger having conditional functionality and consequential functionality.

16. A computer implemented method as recited in claim 15 wherein the trigger is an outgoing communications trigger, the computer implemented method further including the acts of:

monitoring all outgoing communication events in order to determine whether a specific outgoing communication event has occurred; and

in response to the occurrence of the specific outgoing communication event, performing the particular action defined by the trigger.

17. A computer implemented method as recited in claim 15 wherein the trigger is an incoming communications trigger, the computer implemented method further including the acts of:

monitoring all incoming communication events in order to determine whether a specific incoming communication event has occurred; and

in response to the occurrence of a specific incoming communication event satisfying the trigger conditional functionality, performing the particular consequential functionality defined by the trigger.

18. A computer implemented method as recited in claim 15 wherein the trigger is a data trigger, the computer implemented method further including the acts of:

monitoring a state of a data repository; and

in response to a particular state event satisfying the trigger conditional functionality, performing the particular consequential functionality defined by the trigger.

19. A computer implemented method as recited in claim 15 wherein the trigger is a time trigger, the computer implemented method further including the acts of:

monitoring for the occurrence of a particular time condition; and

in response to the occurrence of a particular time condition satisfying the trigger conditional functionality, performing the particular consequential functionality defined by the trigger.

20. A computer implemented method as recited in claim 15 wherein the trigger is installed and executed within the facilitator agent.

21. A computer implemented method as recited in claim 15 wherein the trigger is installed and executed within a first service-providing agent.

22. A computer implemented method as recited in claim 15 wherein the conditional functionality of the trigger is installed on a facilitator agent.

23. A computer implemented method as recited in claim 22 wherein the consequential functionality is installed on a specific service-providing agent other than a facilitator agent.

24. A computer implemented method as recited in claim 15 wherein the conditional functionality of the trigger is installed on a specific service-providing agent other than a facilitator agent.

25. A computer implemented method as recited in claim 15 wherein the consequential functionality of the trigger is installed on a facilitator agent.

26. A computer implemented method as recited in claim 1 wherein the base goal is a compound goal having sub-goals separated by operators.

27. A computer implemented method as recited in claim 26 wherein the type of available operators includes a conjunction operator, a disjunction operator, and a conditional execution operator.

28. A computer implemented method as recited in claim 27 wherein the type of available operators further includes a parallel disjunction operator that indicates that disjunct goals are to be performed by different agents.

29. A computer program stored on a computer readable medium, the computer program executable to facilitate cooperative task completion within a distributed computing environment, the distributed computing environment including a plurality of autonomous electronic agents, the distributed computing environment supporting an Interagent Communication Language, the computer program comprising computer executable instructions for:

providing an agent registry that declares capabilities of service-providing electronic agents currently active within the distributed computing environment;

interpreting a service request in order to determine a base goal that may be a compound, arbitrarily complex base goal, the service request adhering to an Interagent Communication Language (ICL), the act of interpreting including the sub-acts of:

determining any task completion advice provided by the base goal, and

determining any task completion constraints provided by the base goal;

constructing a base goal satisfaction plan including the sub-acts of:

determining whether the requested service is available,

determining sub-goals required in completing the base goal,

selecting service-providing electronic agents from the agent registry suitable for performing the determined sub-goals, and

ordering a delegation of sub-goal requests to best complete the requested service; and

implementing the base goal satisfaction plan.

30. A computer program as recited in claim 29 wherein the computer executable instruction for providing an agent registry includes the following computer executable instructions for registering a specific service-providing electronic agent into the agent registry:

establishing a bi-directional communications link between the specific agent and a facilitator agent controlling the agent registry;

providing a new agent profile to the facilitator agent, the new agent profile defining publicly available capabilities of the specific agent; and

registering the specific agent together with the new agent profile within the agent registry, thereby making available to the facilitator agent the capabilities of the specific agent.

31. A computer program as recited in claim 30 wherein the computer executable instruction for registering a specific agent further includes:

invoking the specific agent in order to activate the specific agent;

instantiating an instance of the specific agent; and

transmitting the new agent profile from the specific agent to the facilitator agent in response to the instantiation of the specific agent.

32. A computer program as recited in claim 29 wherein the computer executable instruction for providing an agent registry includes a computer executable instruction for removing a specific service-providing electronic agent from the registry upon determining that the specific agent is no longer available to provide services.

33. A computer program as recited in claim 29 wherein the provided agent registry includes a symbolic name, a unique address, data declarations, trigger declarations, task declarations, and process characteristics for each active agent.

34. A computer program as recited in claim 29 further including computer executable instructions for receiving the service request via a communications link established with a client.

35. A computer program as recited in claim 29 wherein the computer executable instruction for providing a service request includes instructions for:

receiving a non-ICL format service request;

selecting an active agent capable of converting the non-ICL formal service request into an ICL format service request;

forwarding the non-ICL format service request to the active agent capable of converting the non-ICL format service request, together with a request that such conversion be performed; and

receiving an ICL format service request corresponding to the non-ICL format service request.

36. A computer program as recited in claim 35 wherein the non-ICL format service request includes a natural language query, and the active agent capable of converting the non-ICL formal service request into an ICL format service request is a natural language agent.

37. A computer program as recited in claim 36 wherein the natural language query is generated by a user interface agent.

38. A computer program as recited in claim 29, the computer program further including computer executable instructions for implementing a base goal that requires setting a trigger having conditional and consequential functionality.

39. A computer program as recited in claim 38 wherein the trigger is an outgoing communications trigger, the computer program further including computer executable instructions for:

monitoring all outgoing communication events in order to determine whether a specific outgoing communication event has occurred; and

in response to the occurrence of the specific outgoing communication event, performing the particular action defined by the trigger.

40. A computer program as recited in claim 38 wherein the trigger is an incoming communications trigger, the computer program further including computer executable instructions for:

monitoring all incoming communication events in order to determine whether a specific incoming communication event has occurred; and

in response to the occurrence of the specific incoming communication event, performing the particular action defined by the trigger.

41. A computer program as recited in claim 38 wherein the trigger is a data trigger, the computer program further including computer executable instructions for:

monitoring a state of a data repository; and

in response to a particular state event, performing the particular action defined by the trigger.

42. A computer program as recited in claim 38 wherein the trigger is a time trigger, the computer program further including computer executable instructions for:

monitoring for the occurrence of a particular time condition; and

in response to the occurrence of the particular time condition, performing the particular action defined by the trigger.

43. A computer program as recited in claim 38 further including computer executable instructions for installing and executing the trigger within the facilitator agent.

44. A computer program as recited in claim 38 further including computer executable instructions for installing and executing the trigger within a first service-providing agent.

45. A computer program as recited in claim 29 further including computer executable instructiòns for interpreting compound goals having sub-goals separated by operators.

46. A computer program as recited in claim 45 wherein the type of available operators includes a conjunction operator, a disjunction operator, and a conditional execution operator.

47. A computer program as recited in claim 46 wherein the type of available operators further includes a parallel disjunction operator that indicates that disjunct goals are to be performed by different agents.

48. An Interagent Communication Language (ICL) providing a basis for facilitated cooperative task completion within a distributed computing environment having a facilitator agent and a plurality of autonomous service-providing electronic agents, the ICL enabling agents to perform queries of other agents, exchange information with other agents, set triggers within other agents, an ICL syntax supporting compound goal expressions such that goals within a single request provided according to the ICL syntax may be coupled by a conjunctive operator, a disjunctive operator, a conditional execution operator, and a parallel disjunctive operator parallel disjunctive operator that indicates that disjunct goals are to be performed by different agents.

49. An ICL as recited in claim 48, wherein the ICL is computer platform independent.

50. An ICL as recited in claim 48 wherein the ICL is independent of computer programming languages which the plurality of agents are programmed in.

51. An ICL as recited in claim 48 wherein the ICL syntax supports explicit task completion constraints within goal expressions.

52. An ICL as recited in claim 51 wherein possible types of task completion constraints include use of specific agent constraints and response time constraints.

53. An ICL as recited in claim 51 wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

54. An ICL as recited in claim 48 wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

55. An ICL as recited in claim 48 wherein each autonomous service-providing electronic agent defines and publishes a set of capability declarations or solvables, expressed in ICL, that describes services provided by such electronic agent.

56. An ICL as recited in claim 55 wherein an electronic agent's solvables define an interface for the electronic agent.

57. An ICL as recited in claim 56 wherein the facilitator agent maintains an agent registry making available a plurality of electronic agent interfaces.

58. An ICL as recited in claim 57 wherein the possible types of solvables includes procedure solvables, a procedure solvable operable to implement a procedure such as a test or an action.

59. An ICL as recited in claim 58 wherein the possible types of solvables further includes data solvables, a data solvable operable to provide access to a collection of data.

60. An ICL as recited in claim 58 wherein the possible types of solvables includes data solvables, a data solvable operable to provide access to a collection of data.

61. A facilitator agent arranged to coordinate cooperative task completion within a distributed computing environment having a plurality of autonomous service-providing electronic agents, the facilitator agent comprising:

an agent registry that declares capabilities of service-providing electronic agents currently active within the distributed computing environment; and

a facilitating engine operable to parse a service request in order to interpret a compound goal set forth therein, the compound goal including both local and global constraints and control parameters, the service request formed according to an Interagent Communication Language (ICL), the facilitating engine further operable to construct a goal satisfaction plan specifying the coordination of a suitable delegation of sub-goal requests to complete the requested service satisfying both the local and global constraints and control parameters.

62. A facilitator agent as recited in claim 61, wherein the facilitating engine is capable of modifying the goal satisfaction plan during execution, the modifying initiated by events such as new agent declarations within the agent registry, decisions made by remote agents, and information provided to the facilitating engine by remote agents.

63. A facilitator agent as recited in claim 61 wherein the agent registry includes a symbolic name, a unique address, data declarations, trigger declarations, task declarations, and process characteristics for each active agent.

64. A facilitator agent as recited in claim 61 wherein the facilitating engine is operable to install a trigger mechanism requesting that a certain action be taken when a certain set of conditions are met.

65. A facilitator agent as recited in claim 64 wherein the trigger mechanism is a communication trigger that monitors communication events and performs the certain action when a certain communication event occurs.

66. A facilitator agent as recited in claim 64 wherein the trigger mechanism is a data trigger that monitors a state of a data repository and performs the certain action when a certain data state is obtained.

67. A facilitator agent as recited in claim 66 wherein the data repository is local to the facilitator agent.

68. A facilitator agent as recited in claim 66 wherein the data repository is remote from the facilitator agent.

69. A facilitator agent as recited in claim 64 wherein the trigger mechanism is a task trigger having a set of conditions.

70. A facilitator agent as recited in claim 61, the facilitator agent further including a global database accessible to at least one of the service-providing electronic agents.

71. A software-based, flexible computer architecture for communication and cooperation among distributed electronic agents, the architecture contemplating a distributed computing system comprising:

a plurality of service-providing electronic agents; and

a facilitator agent in bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including:

an agent registry that declares capabilities of service-providing electronic agents currently active within the distributed computing environment;

a facilitating engine operable to parse a service request in order to interpret an arbitrarily complex goal set forth therein, the facilitating engine further operable to construct a goal satisfaction plan including the coordination of a suitable delegation of sub-goal requests to best complete the requested service.

72. A computer architecture as recited in claim 71, wherein the basis for the computer architect is an Interagent Communication Language (ICL) enabling agents to perform queries of other agents, exchange information with other agents, and set triggers within other agents, the ICL further defined by an ICL syntax supporting compound goal expressions such that goals within a single request provided according to the ICL syntax may be coupled by a conjunctive operator, a disjunctive operator, a conditional execution operator, and a parallel disjunctive operator parallel disjunctive operator that indicates that disjunct goals are to be performed by different agents.

73. A computer architecture as recited in claim 72, wherein the ICL is computer platform independent.

74. A computer architecture as recited in claim 73 wherein the ICL is independent of computer programming languages in which the plurality of agents are programmed.

75. A computer architecture as recited in claim 73 wherein the ICL syntax supports explicit task completion constraints within goal expressions.

76. A computer architecture as recited in claim 75 wherein possible types of task completion constraints include use of specific agent constraints and response time constraints.

77. A computer architecture as recited in claim 75 wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

78. A computer architecture as recited in claim 73 wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

79. A computer architecture as recited in claim 73 wherein each autonomous service-providing electronic agent defines and publishes a set of capability declarations or solvables, expressed in ICL, that describes services provided by such electronic agent.

80. A computer architecture as recited in claim 79 wherein an electronic agent's solvables define an interface for the electronic agent.

81. A computer architecture as recited in claim 80 wherein the possible types of solvables includes procedure solvables, a procedure solvable operable to implement a procedure such as a test or an action.

82. A computer architecture as recited in claim 81 wherein the possible types of solvables further includes data solvables, a data solvable operable to provide access to a collection of data.

83. A computer architecture as recited in claim 82 wherein the possible types of solvables includes a data solvable operable to provide access to modify a collection of data.

84.· A computer architecture as recited in claim 71 wherein the planning component of the facilitating engine are distributed across at least two computer processes.

85. A computer architecture as recited in claim 71 wherein the execution component of the facilitating engine is distributed across at least two computer processes.

86. A data wave carrier providing a transport mechanism for information communication in a distributed computing environment having at least one facilitator agent and at least one active client agent, the data wave carrier comprising a signal representation of an inter-agent language description of an active client agent's functional capabilities.

87. A data wave carrier as recited in claim 85, the data wave carrier further comprising a signal representation of a request for service in the inter-agent language from a first agent to a second agent.

88. A data wave carrier as recited in claim 85, the data wave carrier further comprising a signal representation of a goal dispatched to an agent for performance from a facilitator agent.

89. A data wave carrier as recited in claim 88 wherein a later state of the data wave carrier comprises a signal representation of a response to the dispatched goal including results and/or a status report from the agent for performance to the facilitator agent.

Software-Based Architecture for Communication and Cooperation Among Distributed Electronic Agents

ABSTRACT

A highly flexible, software-based architecture is disclosed for constructing distributed systems. The architecture supports cooperative task completion by flexible, dynamic configurations of autonomous electronic agents. Communication and cooperation between agents are brokered by one or more facilitators, which are responsible for matching requests, from users and agents, with descriptions of the capabilities of other agents. It is not generally required that a user or agent know the identities, locations, or number of other agents involved in satisfying a request, and relatively minimal effort is involved in incorporating new agents and "wrapping" legacy applications. Extreme flexibility is achieved through an architecture organized around the declaration of capabilities by service-providing agents, the construction of arbitrarily complex goals by users and service-requesting agents, and the role of facilitators in delegating and coordinating the satisfaction of these goals, subject to advice and constraints that may accompany them. Additional mechanisms and features include facilities for creating and maintaining shared repositories of data; the use of triggers to instantiate commitments within and between agents; agent-based provision of multi-modal user interfaces, including natural language; and built-in support for including the user as a privileged member of the agent community. Specialized embodiments providing enhanced scalability are also described.

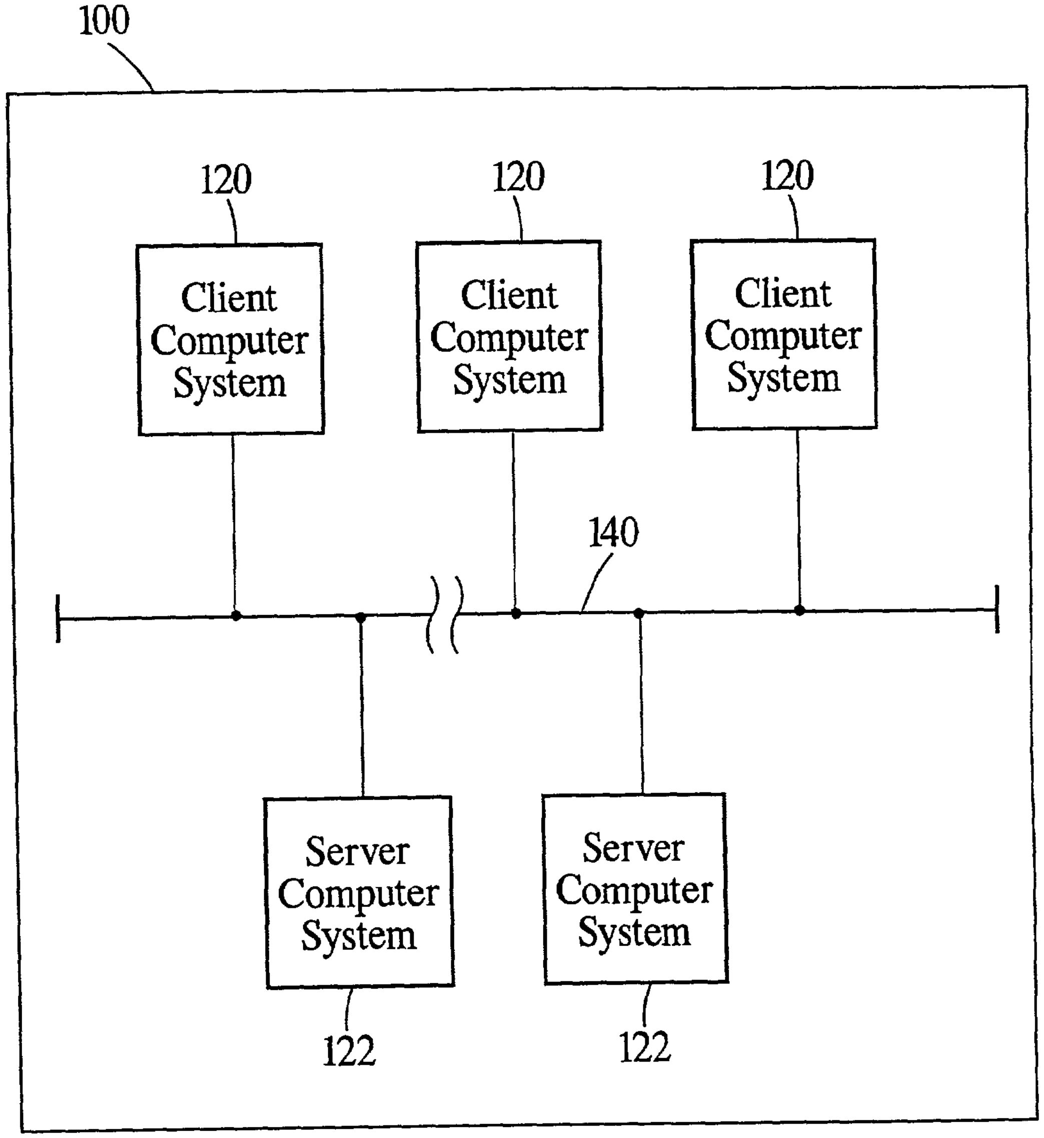


Fig. 1
(Prior Art)

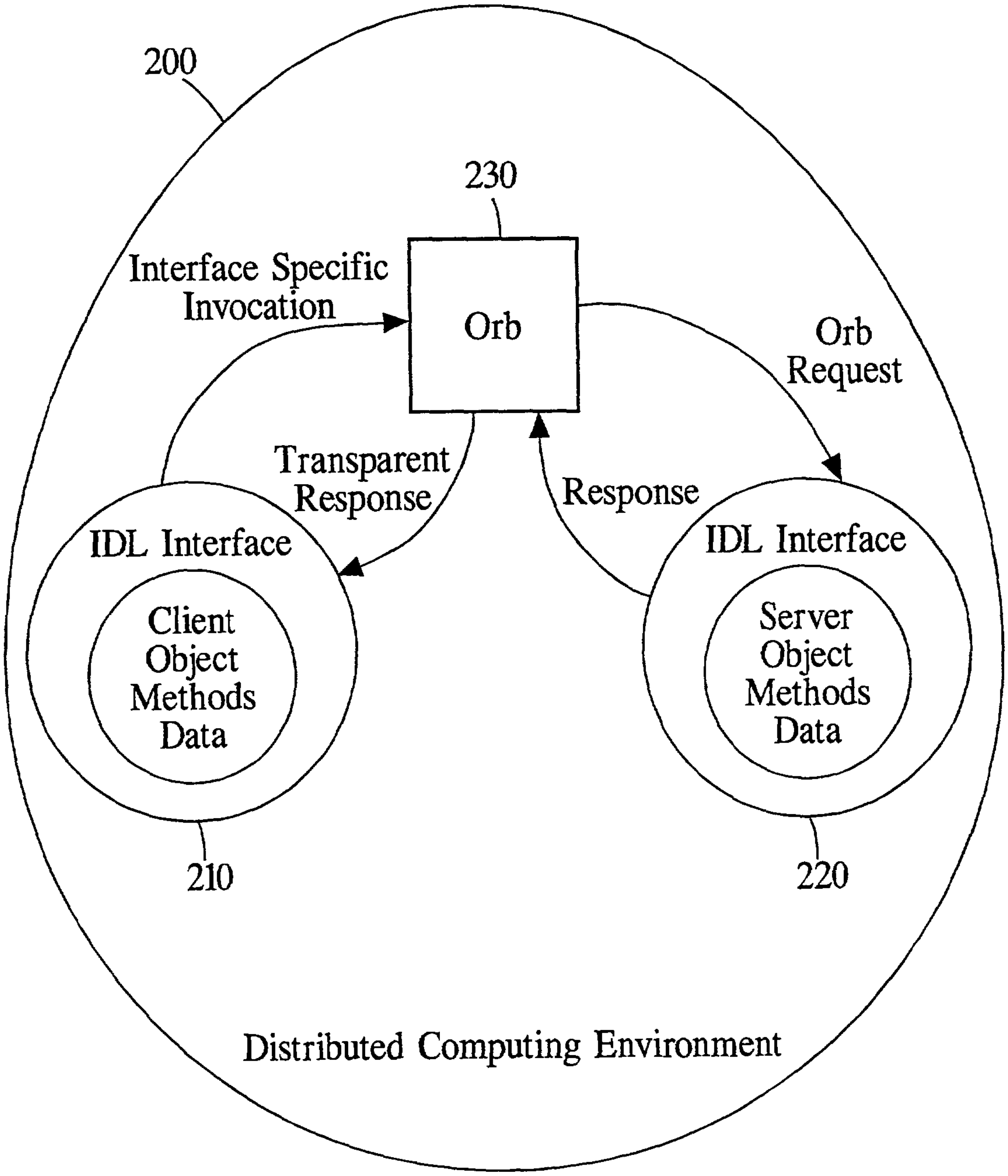

200
230
Interface Specific Invocation
Orb
Orb Request
Transparent Response
Response
IDL Interface
Client Object Methods Data
IDL Interface
Server Object Methods Data
210
220
Distributed Computing Environment

Fig. 2
(Prior Art)

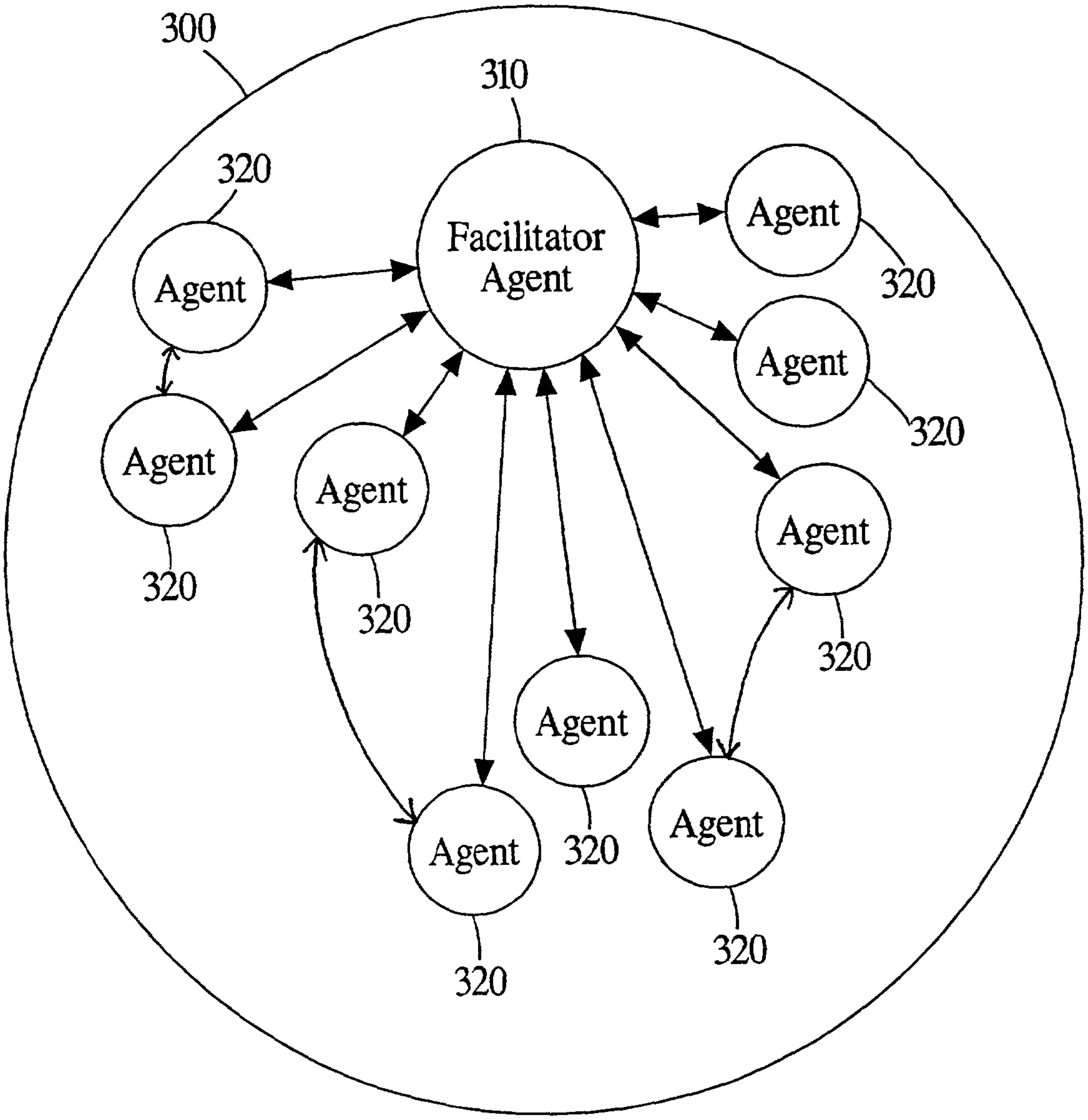

300
310
Facilitator Agent
Agent
Agent
Agent
Agent
Agent
Agent
Agent
Agent
Agent
Agent
320
320
320
320
320
320
320
320
320
320

Fig. 3

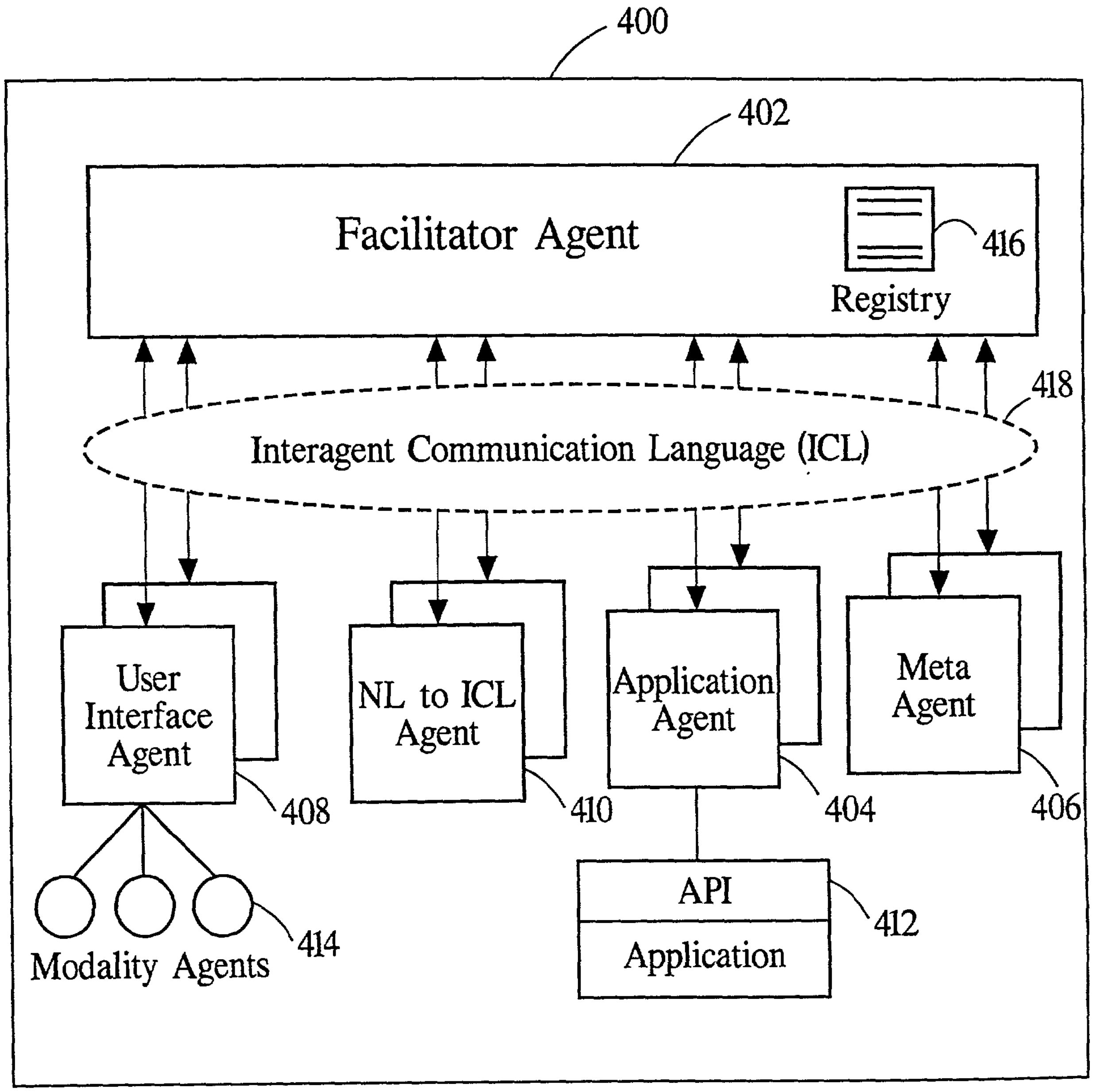
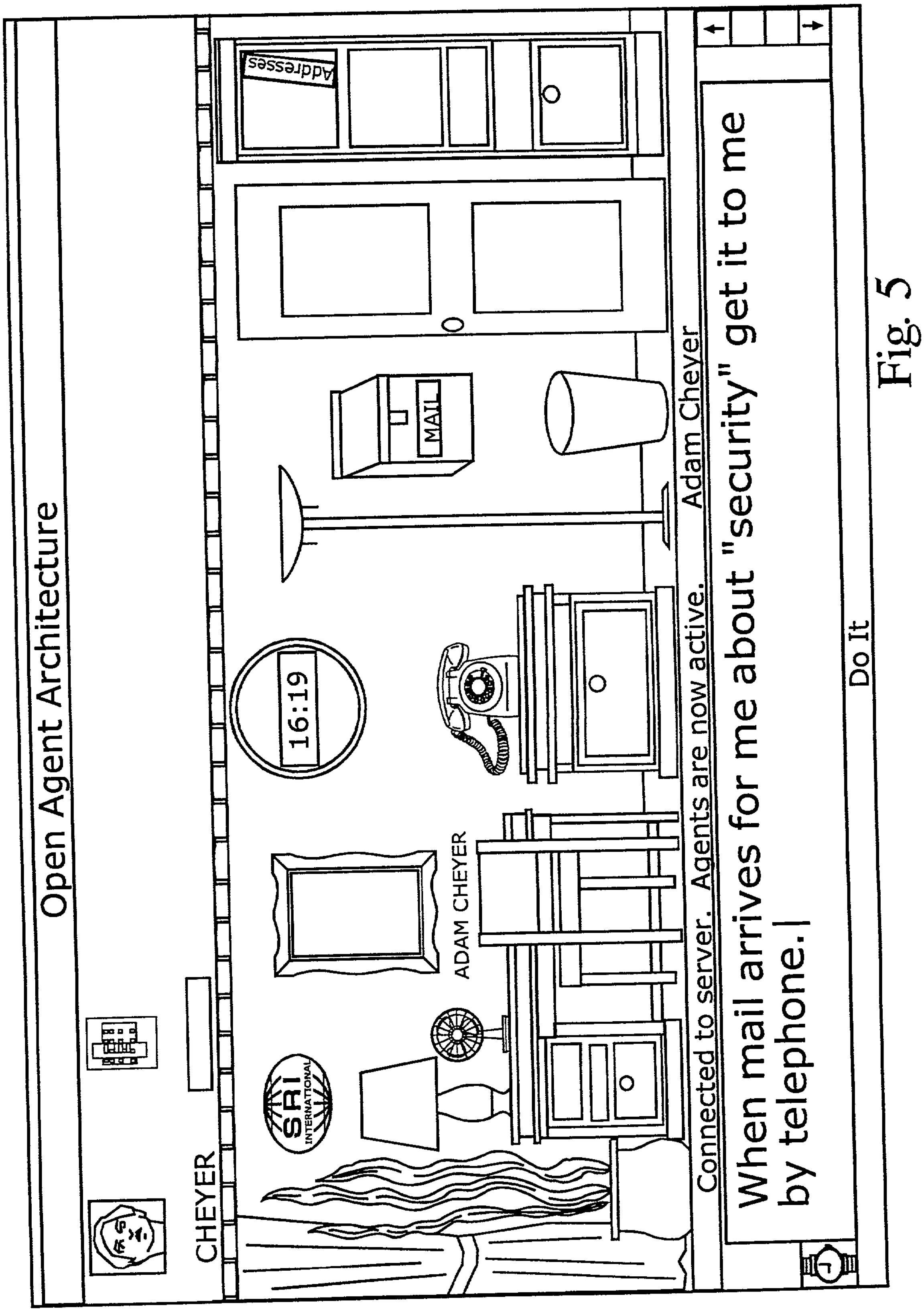

400
402
Facilitator Agent
416
Registry
418
Interagent Communication Language (ICL)
User Interface Agent
408
NL to ICL Agent
410
Application Agent
404
Meta Agent
406
API
Application
412
Modality Agents
414

Fig. 4



Connected to server. Agents are now active.

When mail arrives for me about "security" get it to me by telephone.|

Fig. 5

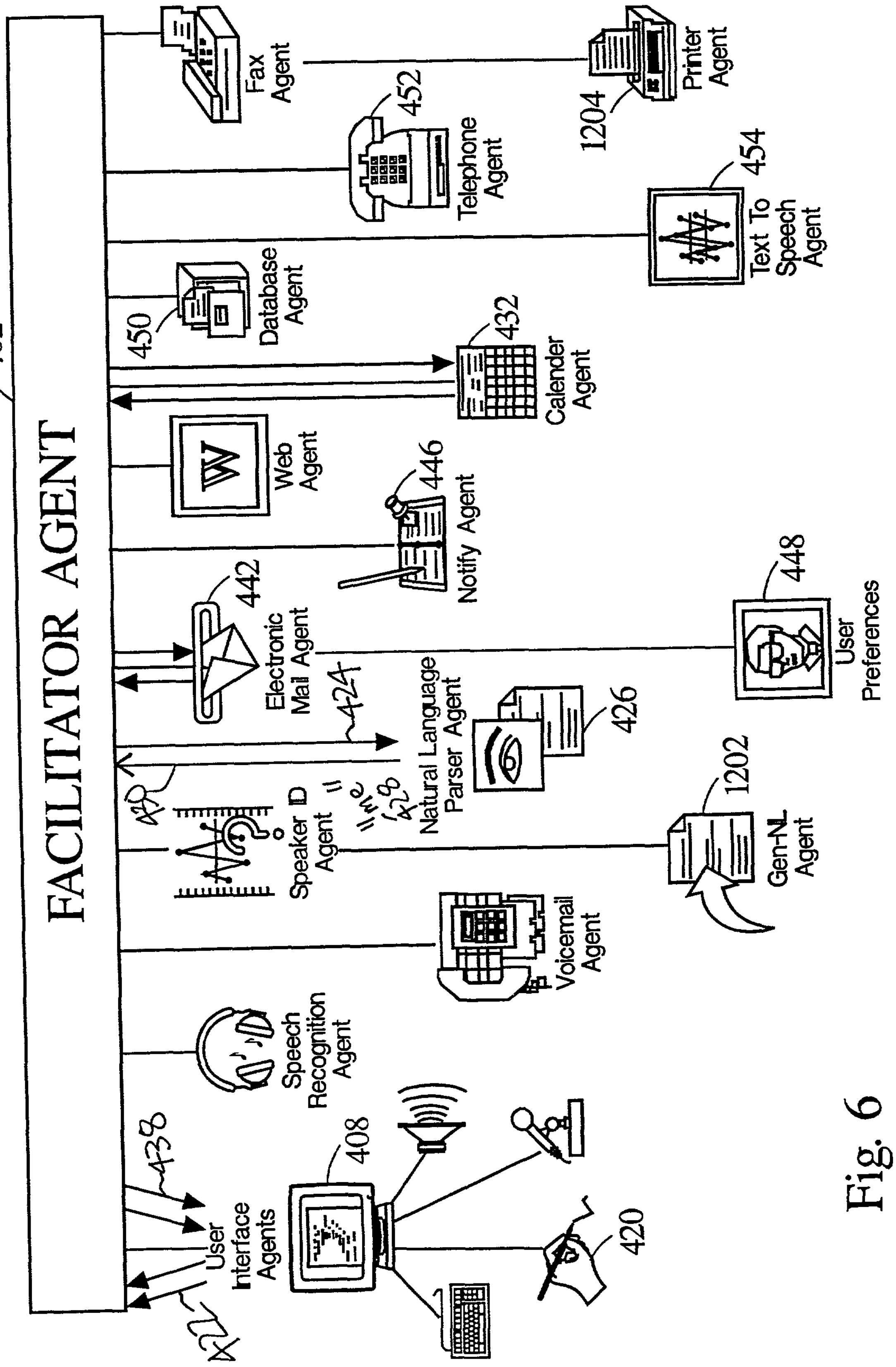

FACILITATOR AGENT
402
Fax Agent
Printer Agent
1204
Telephone Agent
452
Text To Speech Agent
454
Database Agent
450
Calender Agent
432
Web Agent
Notify Agent
446
Electronic Mail Agent
442
User Preferences
448
Natural Language Parser Agent
424
426
430
Speaker ID Agent
"me" 428
Gen-NL Agent
1202
Voicemail Agent
Speech Recognition Agent
User Interface Agents
438
422
408
420

Fig. 6

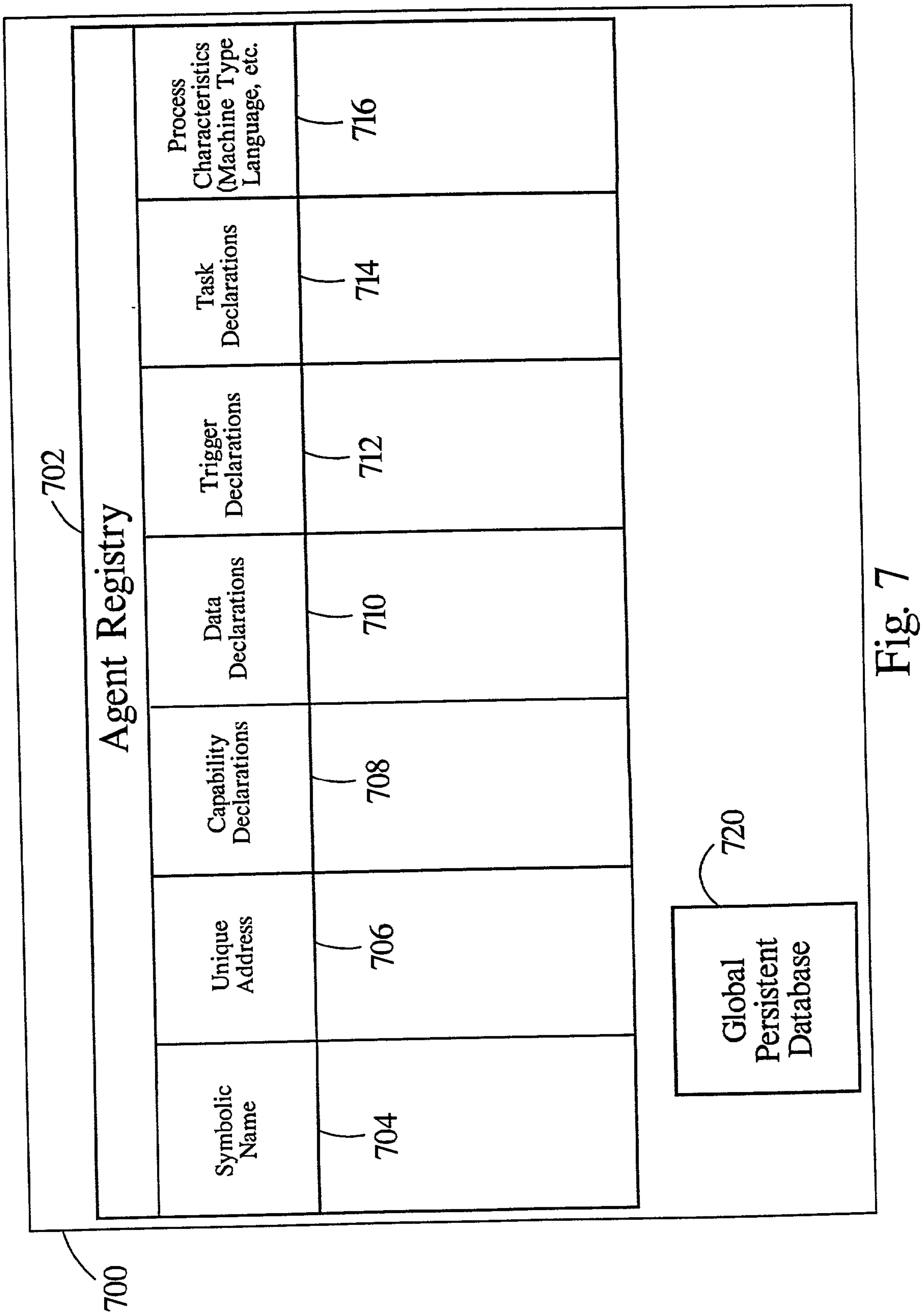

Agent Registry
Symbolic Name
Unique Address
Capability Declarations
Data Declarations
Trigger Declarations
Task Declarations
Process Characteristics (Machine Type Language, etc.
704
706
708
710
712
714
716
702
Global Persistent Database
720
700

Fig. 7

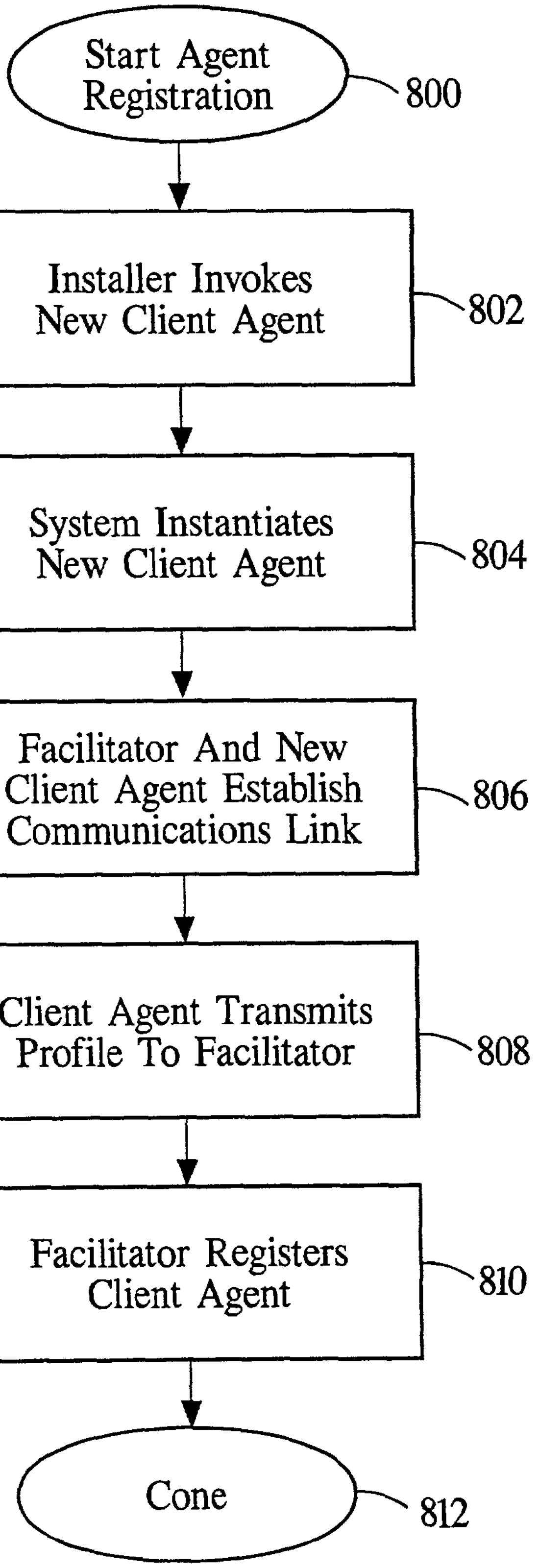

Start Agent Registration
800
Installer Invokes New Client Agent
802
System Instantiates New Client Agent
804
Facilitator And New Client Agent Establish Communications Link
806
Client Agent Transmits Profile To Facilitator
808
Facilitator Registers Client Agent
810
Cone
812

Fig. 8

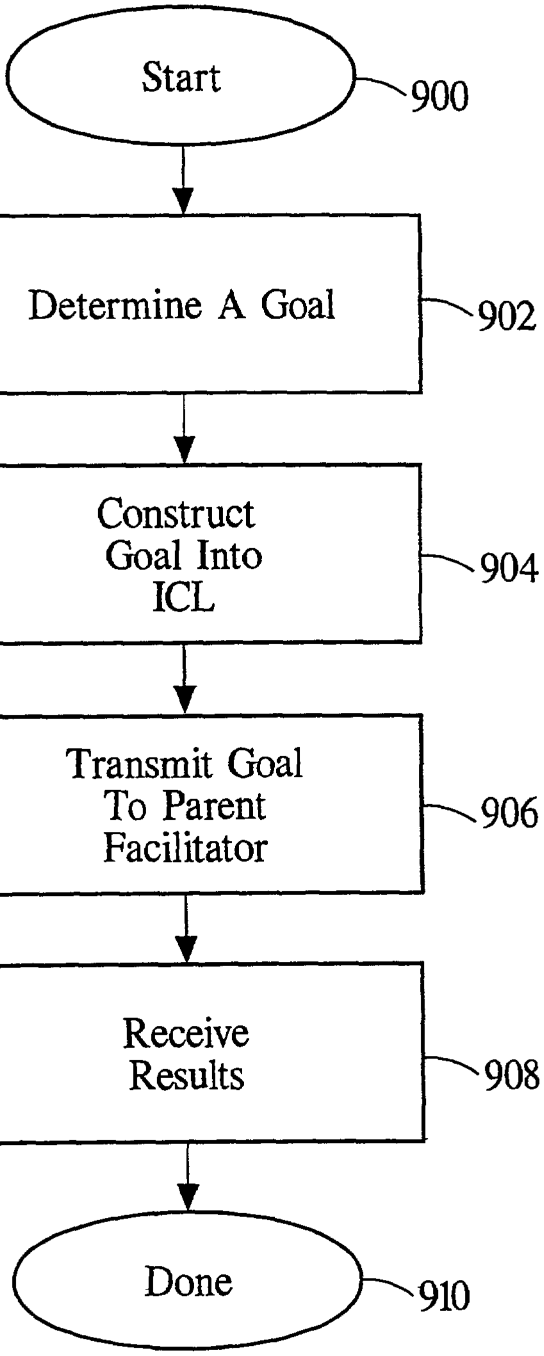

Start
900
Determine A Goal
902
Construct Goal Into ICL
904
Transmit Goal To Parent Facilitator
906
Receive Results
908
Done
910

Fig. 9

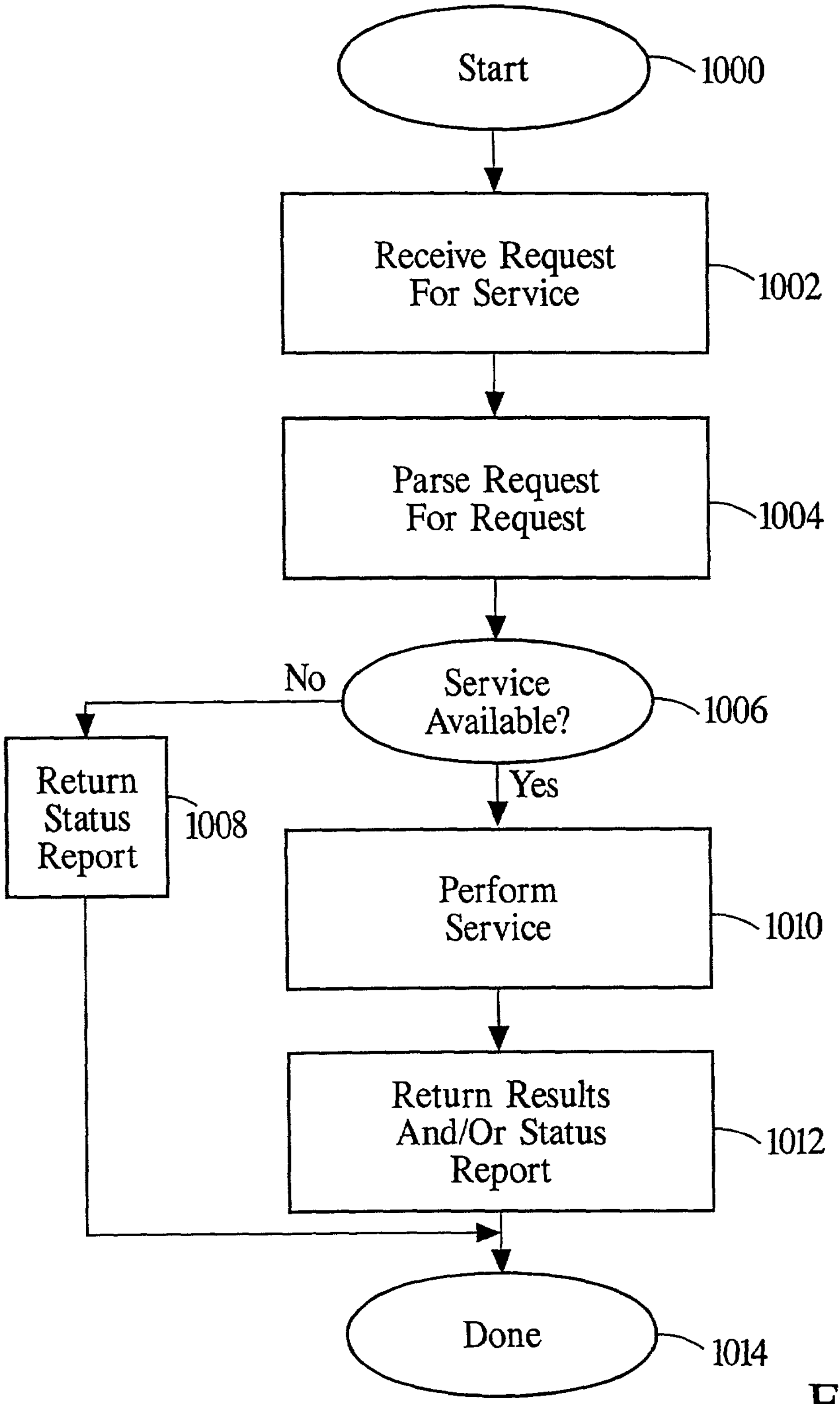

Start
1000
Receive Request For Service
1002
Parse Request For Request
1004
No
Service Available?
1006
Return Status Report
1008
Yes
Perform Service
1010
Return Results And/Or Status Report
1012
Done
1014

Fig. 10

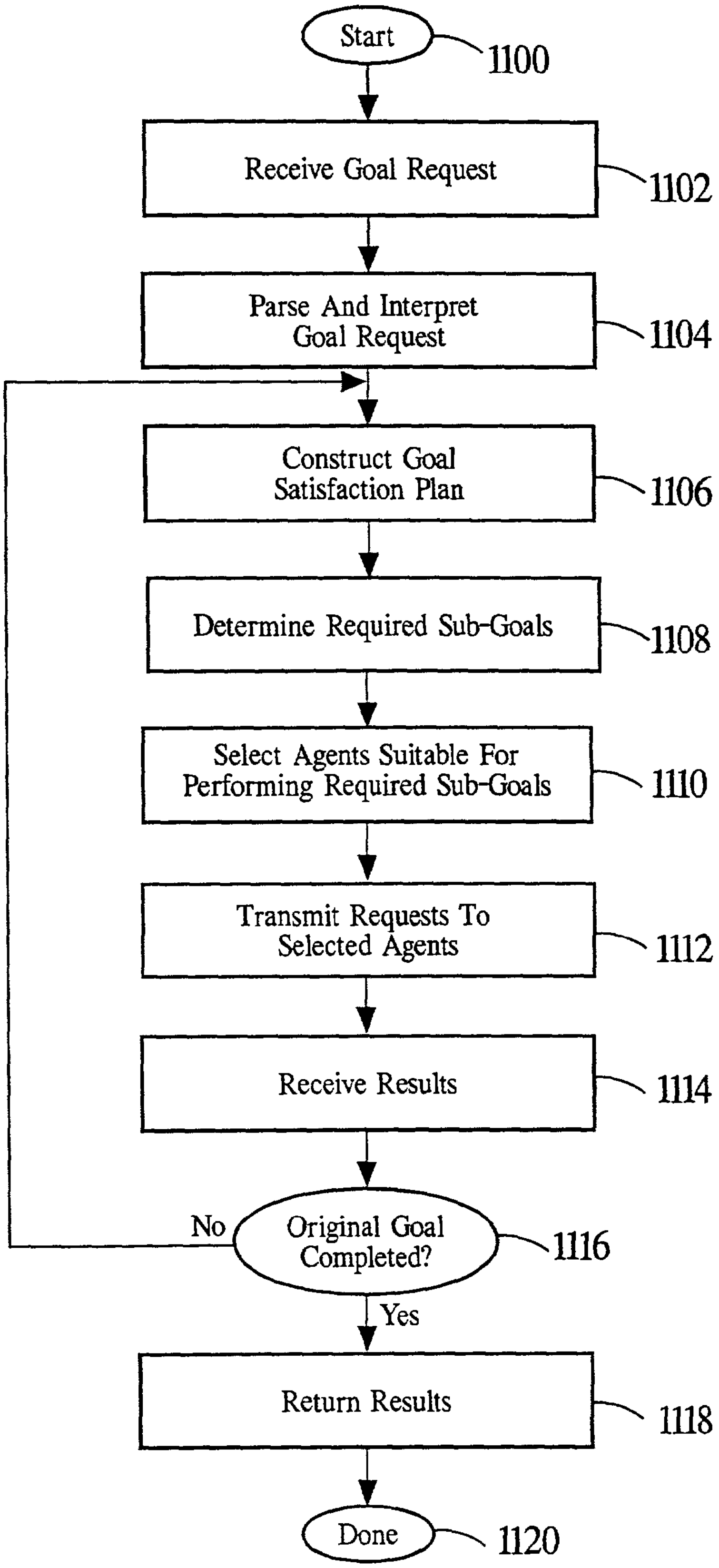

Start
1100
Receive Goal Request
1102
Parse And Interpret Goal Request
1104
Construct Goal Satisfaction Plan
1106
Determine Required Sub-Goals
1108
Select Agents Suitable For Performing Required Sub-Goals
1110
Transmit Requests To Selected Agents
1112
Receive Results
1114
No
Original Goal Completed?
1116
Yes
Return Results
1118
Done
1120

Fig. 11

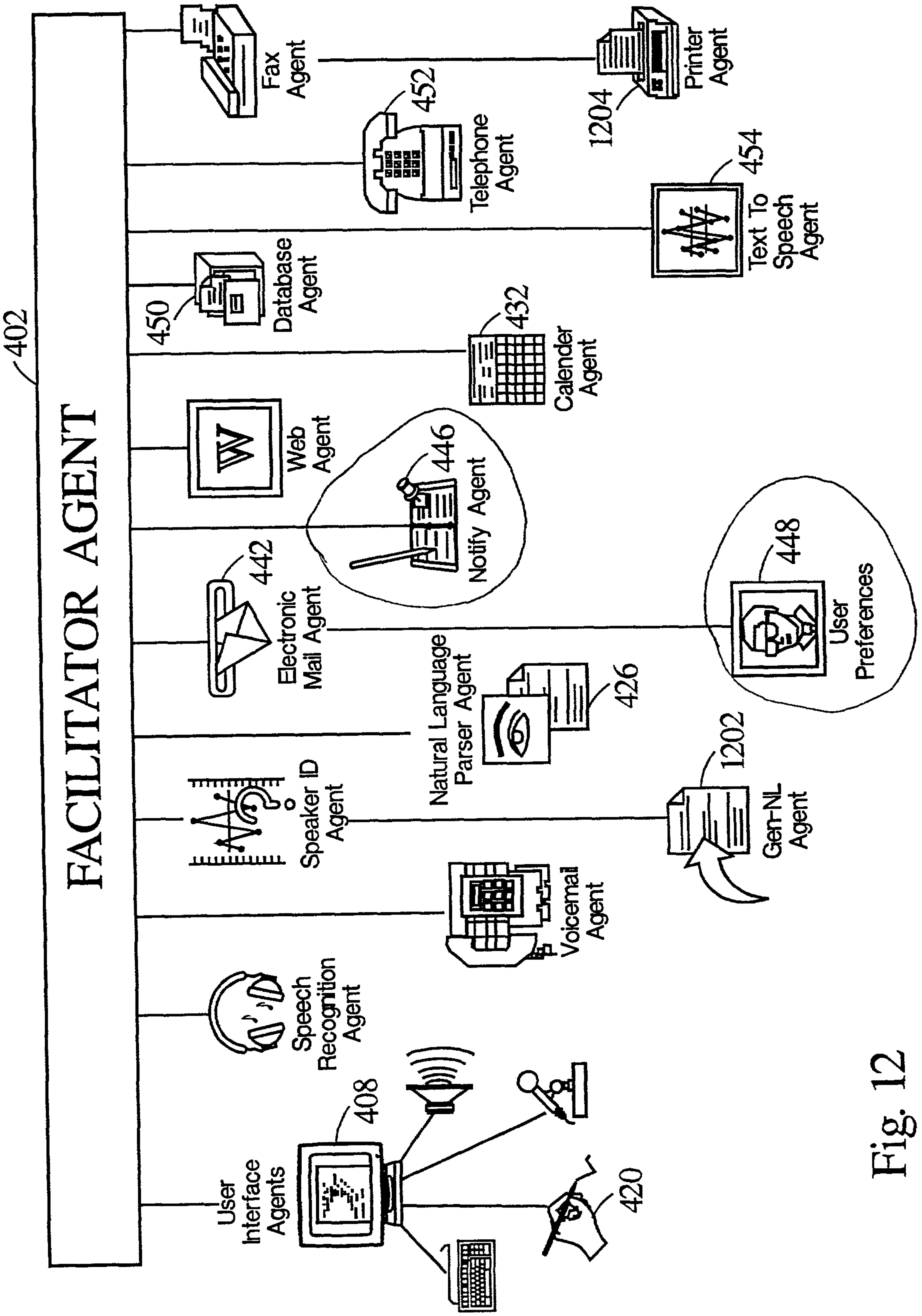

FACILITATOR AGENT
402
User Interface Agents
408
420
Speech Recognition Agent
Voicemail Agent
Speaker ID Agent
Gen-NL Agent
1202
Natural Language Parser Agent
426
Electronic Mail Agent
442
User Preferences
448
Notify Agent
446
Web Agent
Calender Agent
432
Database Agent
450
Text To Speech Agent
454
Telephone Agent
452
Fax Agent
Printer Agent
1204

Fig. 12

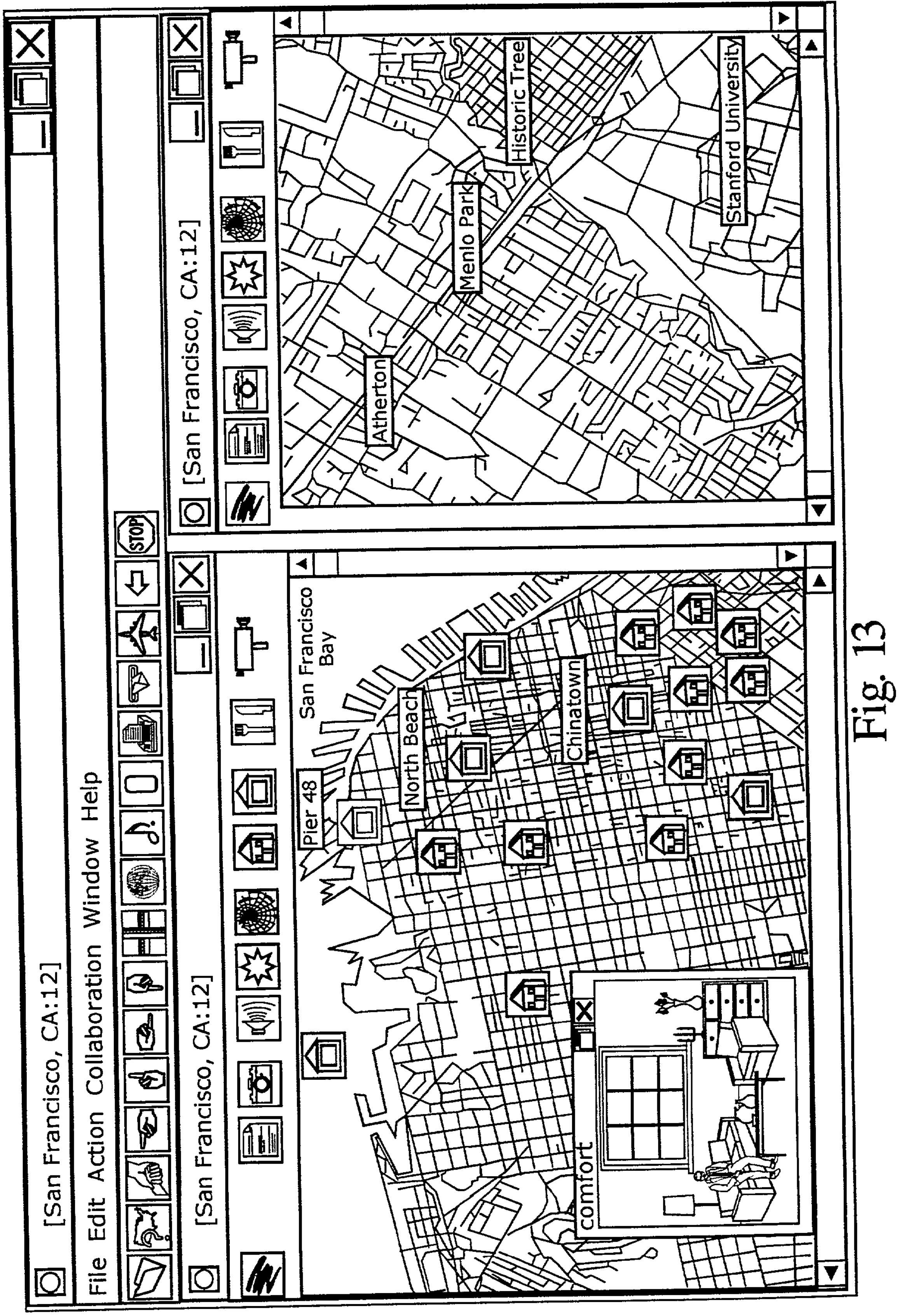

[San Francisco, CA:12]
File Edit Action Collaboration Window Help
STOP
[San Francisco, CA:12]
San Francisco Bay
Pier 48
North Beach
Chinatown
comfort
[San Francisco, CA:12]
Atherton
Menlo Park
Historic Tree
Stanford University

Fig. 13

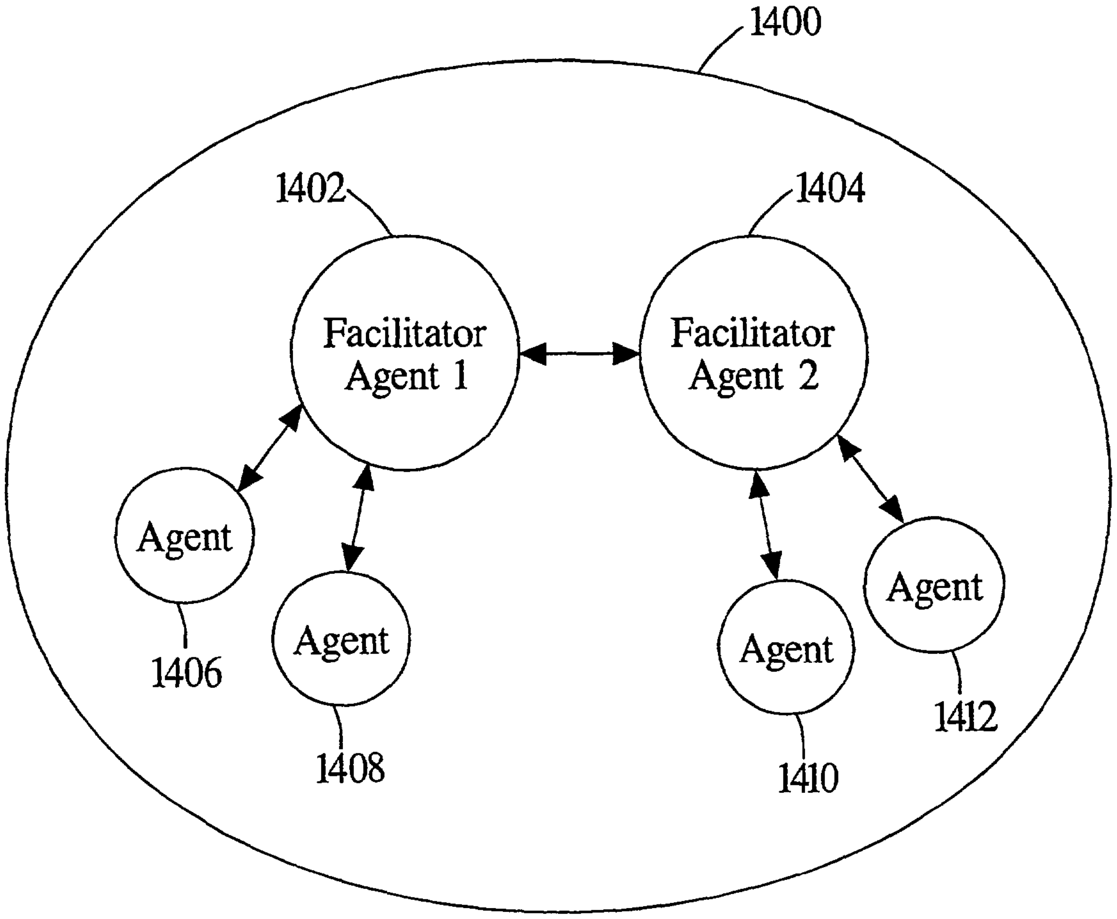

1400
1402
1404
Facilitator Agent 1
Facilitator Agent 2
Agent
Agent
Agent
Agent
1406
1408
1410
1412

Fig. 14

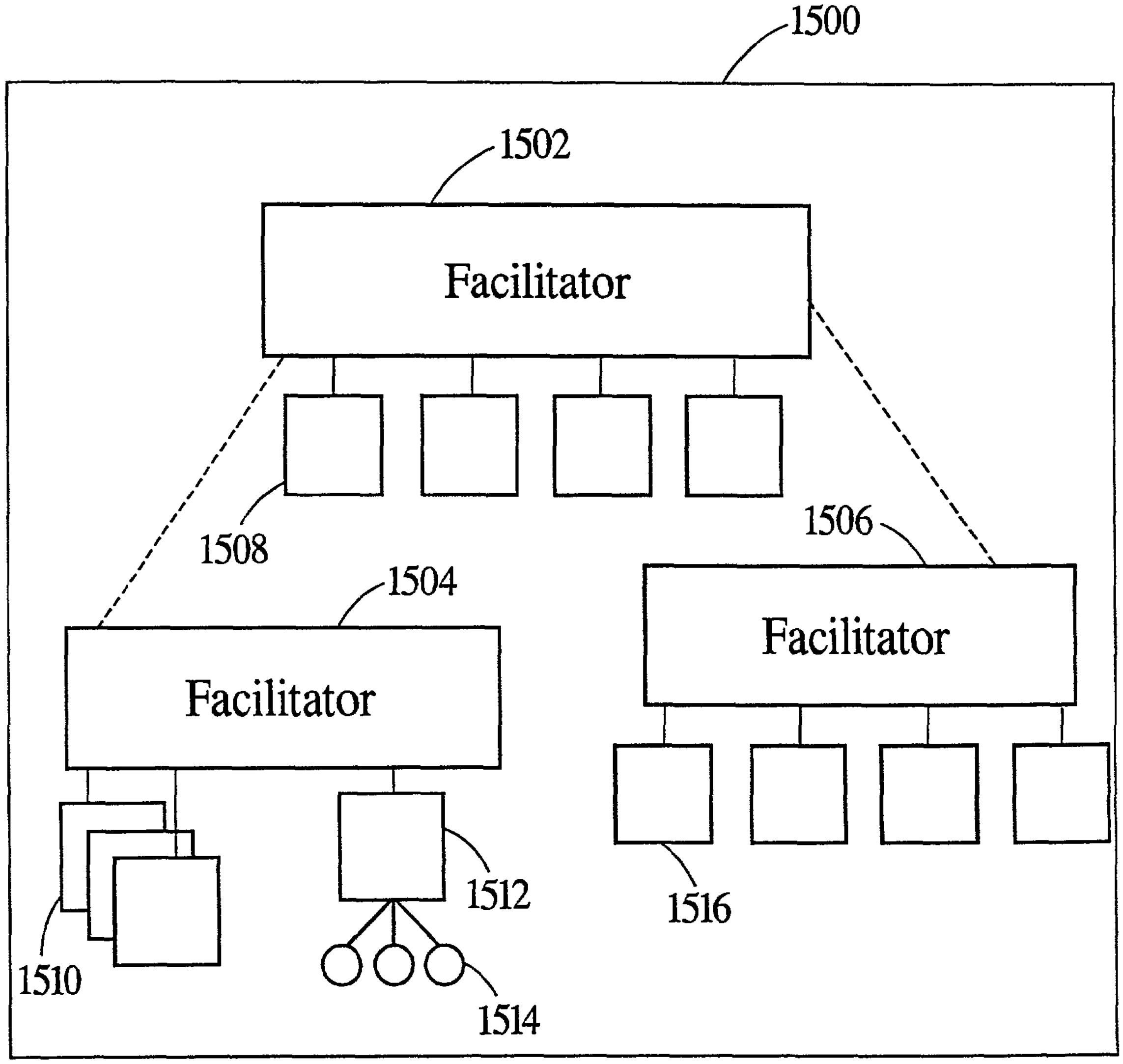

1500
1502
Facilitator
1508
1506
1504
Facilitator
Facilitator
1512
1516
1510
1514

Fig. 15

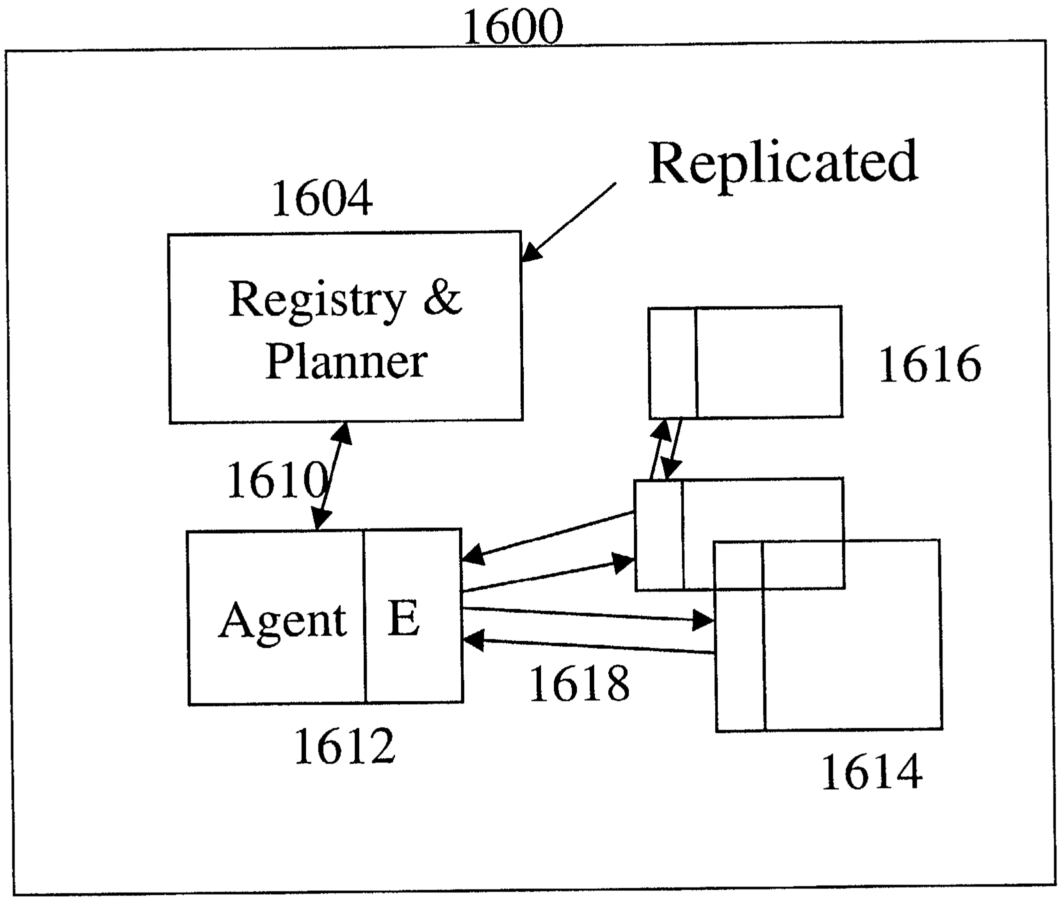

1600
Replicated
1604
Registry & Planner
1616
1610
Agent
E
1618
1612
1614

Figure 16

DECLARATION AND POWER OF ATTORNEY
FOR ORIGINAL U.S. PATENT APPLICATION

Attorney's Docket No. SRI1P016

As a below-named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe that I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled: SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS, the specification of which is attached hereto.

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, CFR § 1.56.

AJC DLM Stephens & Coleman, LLP,

And I hereby appoint the law firm of Hickman ~~& Martin~~ including **Paul L. Hickman (Reg. No. 28, 516); L. Keith Stephens (Reg. No. 32,632); Brian R. Coleman (Reg. No. 39,145); Dawn L. Palmer (Reg. No. 41,238); Jerray Wei (Reg. No. 43,247); and Ian L. Cartier (Reg. No. 38,406)** as my principal attorneys to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith:

Send Correspondence To:

Brian R. Coleman
HICKMAN STEPHENS & COLEMAN, LLP
P.O. BOX 52037
Palo Alto, California 94303-0746

Direct Telephone Calls To: **Brian R. Coleman at telephone number (650) 470-7430**

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

Typewritten Full Name of Sole or First Inventor: Adam J. Cheyer Citizenship: USA
Inventor's signature: Adam J Cheyer **Date of Signature:** 1/5/99
Residence: (City) Palo Alto (State/Country) CA
Post Office Address: 757 Cereza Drive Palo Alto CA 94306

Typewritten Full Name of Second Inventor: David L. Martin Citizenship: USA
Inventor's signature: David L. Martin **Date of Signature:** 1/5/99
Residence: (City) Santa Clara (State/Country) CA
Post Office Address: 167 CRONIN DR. Santa Clara, CA 95051

Jc618 U.S. PTO
09/225198
01/05/99

ISSUE CLASSIFICATION	
Subclass	
Class	

PATENT NUMBER

U.S. UTILITY PATENT APPLICATION

O.I.P.E.	PATENT DATE
TM SCANNED A.G.@ Q.A. KB	

SECTOR	CLASS	SUBCLASS	ART UNIT	EXAMINER
	~~[illegible]~~ 709	317	~~[illegible]~~ 2151 2126	LCOCK

FILED WITH: □ DISK (CRF) □ FICHE
(Attached in pocket on right inside flap)

PREPARED AND APPROVED FOR ISSUE

ISSUING CLASSIFICATION

ORIGINAL		CROSS REFERENCE(S)						
CLASS	SUBCLASS	CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)					
INTERNATIONAL CLASSIFICATION								
/								
/								
/								
/								
/				□ Continued on Issue Slip Inside File Jacket				

□ TERMINAL DISCLAIMER	DRAWINGS			CLAIMS ALLOWED	
	Sheets Drwg.	Figs. Drwg.	Print Fig.	Total Claims	Print Claim for O.G.

□ a) The term of this patent subsequent to ____________ (date) has been disclaimed.

____________________ (Assistant Examiner) ________ (Date)

NOTICE OF ALLOWANCE MAILED

□ b) The term of this patent shall not extend beyond the expiration date of U.S Patent. No. ____________

____

____

____________________ (Primary Examiner) ________ (Date)

ISSUE FEE	
Amount Due	Date Paid

□ c) The terminal ____months of this patent have been disclaimed.

____________________ (Legal Instruments Examiner) ________ (Date)

ISSUE BATCH NUMBER

WARNING:
The information disclosed herein may be restricted. Unauthorized disclosure may be prohibited by the United States Code Title 35, Sections 122, 181 and 368. Possession outside the U.S. Patent & Trademark Office is restricted to authorized employees and contractors only.

Form PTO-436A
(Rev. 6/98)

Best Available Copy

(LABEL AREA)

(FACE)

[redacted]

Best Available Copy

SEARCHED

Class	Sub.	Date	Exmr.
709	317 202	7/10/02	jfs
709	317 202	2/20/03	jfs
UPDATED		11/20/03	jfs

SEARCH NOTES (INCLUDING SEARCH STRATEGY)

	Date	Exmr.
EAST WEST ACM IEEE INTERNET	7/10/02	jfs
UPDATED	2/20/03	jfs
UPDATED.	11/20/03	jfs

INTERFERENCE SEARCHED

Class	Sub.	Date	Exmr.

(RIGHT OUTSIDE)

ISSUE SLIP S⁻⁻·ΓʼE AREA (for additional cross references)

POSITION	INITIALS	ID NO.	DATE
FEE DETERMINATI N	NW	71534	01-19/99
O.I.P.E. CLASSIFIER		10	1/20
FORMALITY REVIEW	YC	71470	1/28/99

INDEX OF CLAIMS

✔ .... Rejected		N .... Non-elected	
= .... Allowed		I .... Interference	
— (Through numeral)... Canceled		A .... Appeal	
÷ .... Restricted		0 .... Objected	

Claim		Date									
Final	Original	7/11/02	2/21/03	11/20/03							
	1	✓	✓	✓							
	2										
	3										
	4										
	5										
	6										
	7										
	8										
	9										
	10										
	11										
	12										
	13										
	14										
	15										
	16										
	17										
	18										
	19										
	20										
	21										
	22										
	23										
	24										
	25										
	26										
	27										
	28										
	29										
	30										
	31										
	32										
	33										
	34										
	35										
	36										
	37										
	38										
	39										
	40										
	41										
	42										
	43										
	44										
	45										
	46										
	47										
	48										
	49										
	50	✓	✓	✓							

Claim		Date									
Final	Original	7/11/02	2/21/03	11/20/03							
	51	✓	✓	✓							
	52										
	53										
	54										
	55										
	56										
	57										
	58										
	59										
	60										
	61										
	62										
	63										
	64										
	65										
	66										
	67										
	68										
	69										
	70										
	71										
	72										
	73										
	74										
	75										
	76										
	77										
	78										
	79										
	80										
	81										
	82										
	83										
	84										
	85										
	86										
	87										
	88										
	89	✓	✓	✓							
	90										
	91										
	92										
	93										
	94										
	95										
	96										
	97										
	98										
	99										
	100										

Claim		Date									
Final	Original										
	101										
	102										
	103										
	104										
	105										
	106										
	107										
	108										
	109										
	110										
	111										
	112										
	113										
	114										
	115										
	116										
	117										
	118										
	119										
	120										
	121										
	122										
	123										
	124										
	125										
	126										
	127										
	128										
	129										
	130										
	131										
	132										
	133										
	134										
	135										
	136										
	137										
	138										
	139										
	140										
	141										
	142										
	143										
	144										
	145										
	146										
	147										
	148										
	149										
	150										

If more than 150 claims or 10 actions
staple additional sheet here

(LEFT INSIDE)

A

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF EXPRESS MAILING
I hereby certify that this paper and the documents and/or fees referred to as attached therein are being deposited with the United States Postal Service on January 05, 1999 in an envelope as "Express Mail Post Office to Addressee" service under 37 CFR §1.10, Mailing Label Number EL221766053US, addressed to the Assistant Commissioner for Patents, Washington, DC 20231.

Michael L. Gough

Attorney Docket No.: SRI1P016
First Named Inventor:
CHEYER, Adam J.

UTILITY PATENT APPLICATION TRANSMITTAL (37 CFR § 1.53(b))

Assistant Commissioner for Patents
Box Patent Application
Washington, DC 20231

□ Duplicate for fee processing

Sir: This is a request for filing a patent application under 37 CFR § 1.53(b) in the name of inventors: Adam J. Cheyer and David L. Martin

For: **SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS**

Application Elements:

☒ 59 Pages of Specification, Claims and Abstract
☒ 16 Sheets of Drawings
☒ 01 Pages Combined Declaration and Power of Attorney

Accompanying Application Parts:

☒ Assignment and Assignment Recordation Cover Sheet (recording fee not enclosed)
☒ Return Receipt Postcard

Fee Calculation (37 CFR § 1.16)

	(Col. 1)	(Col. 2)	SMALL ENTITY		OR	LARGE ENTITY	
	NO. FILED	NO. EXTRA	RATE	FEE		RATE	FEE
BASIC FEE			$395	$	OR	$760	$ 760.00
TOTAL CLAIMS	89 -20 =	69	x11 =	$	OR	x18 =	$1242.00
INDEP CLAIMS	06 -03 =	03	x41 =	$	OR	x78 =	$ 234.00
* If the difference in Col. 1 is less than zero, enter "0" in Col. 2.			Total	$	OR	Total	$2236.00

Including filing fees and the assignment recordation fee of $40.00, the Commissioner is authorized to charge all required fees to Deposit Account No. 50-0384 (Order No. SRI1P016).

☒ The Commissioner is authorized to charge any fees beyond the amount enclosed which may be required, or to credit any overpayment, to Deposit Account No. 50-0384 (Order No. SRI1P016).

General Authorization for Petition Extension of Time (37 CFR §1.136)

☒ Applicants hereby make and generally authorize any Petitions for Extensions of Time as may be needed for any subsequent filings. The Commissioner is also authorized to charge any extension fees under 37 CFR §1.17 as may be needed to Deposit Account No. 50-0384.

☒ Please send correspondence to the following address:

Brian R. Coleman
HICKMAN STEPHENS & COLEMAN, LLP
P.O. Box 52037
Palo Alto, CA 94303-0746

Tel (650) 470-7430
Fax (650) 470-7440

Date: 1/5/99

Brian R. Coleman
Registration No. **39,145**

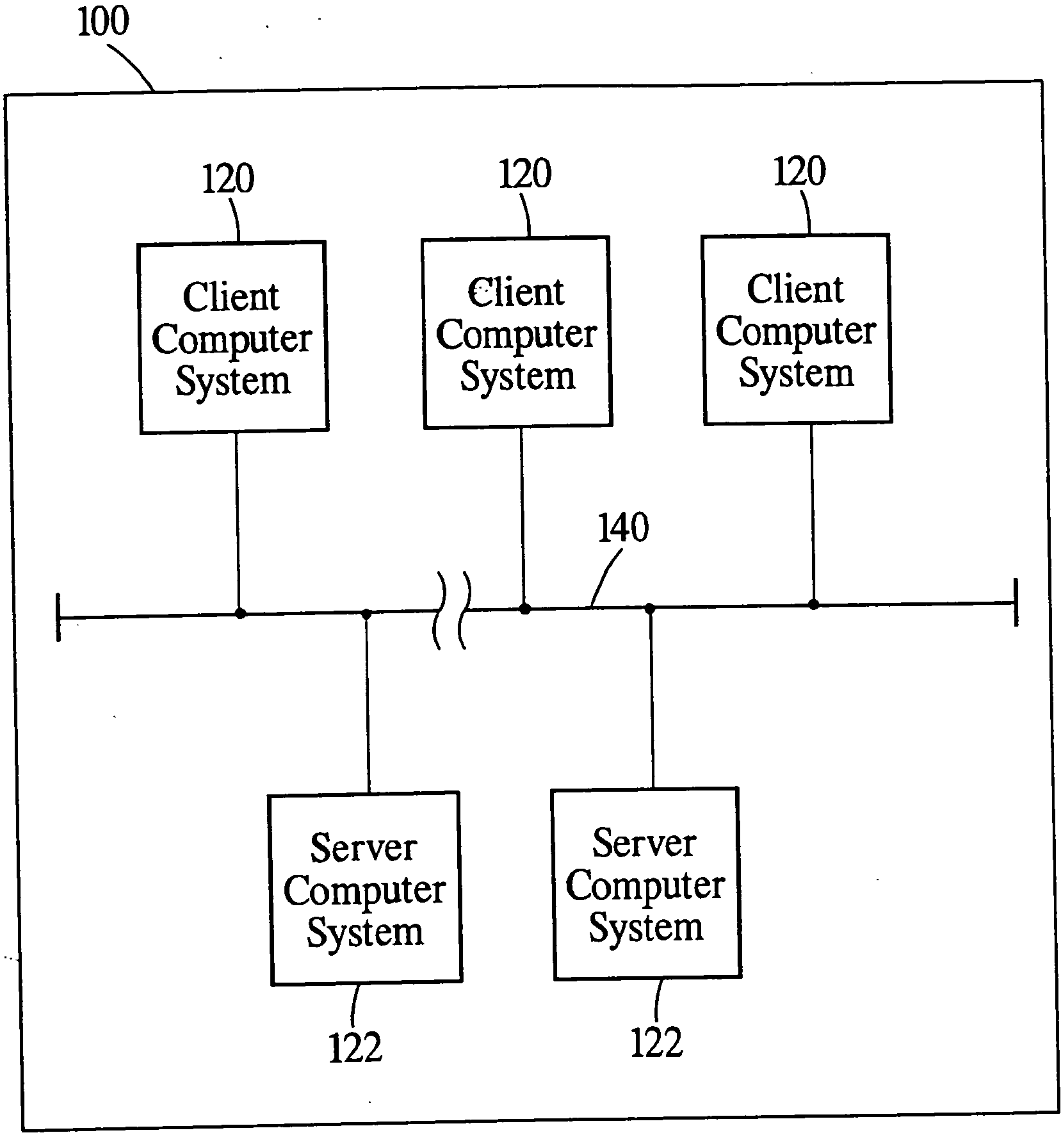


Fig. 1
(Prior Art)

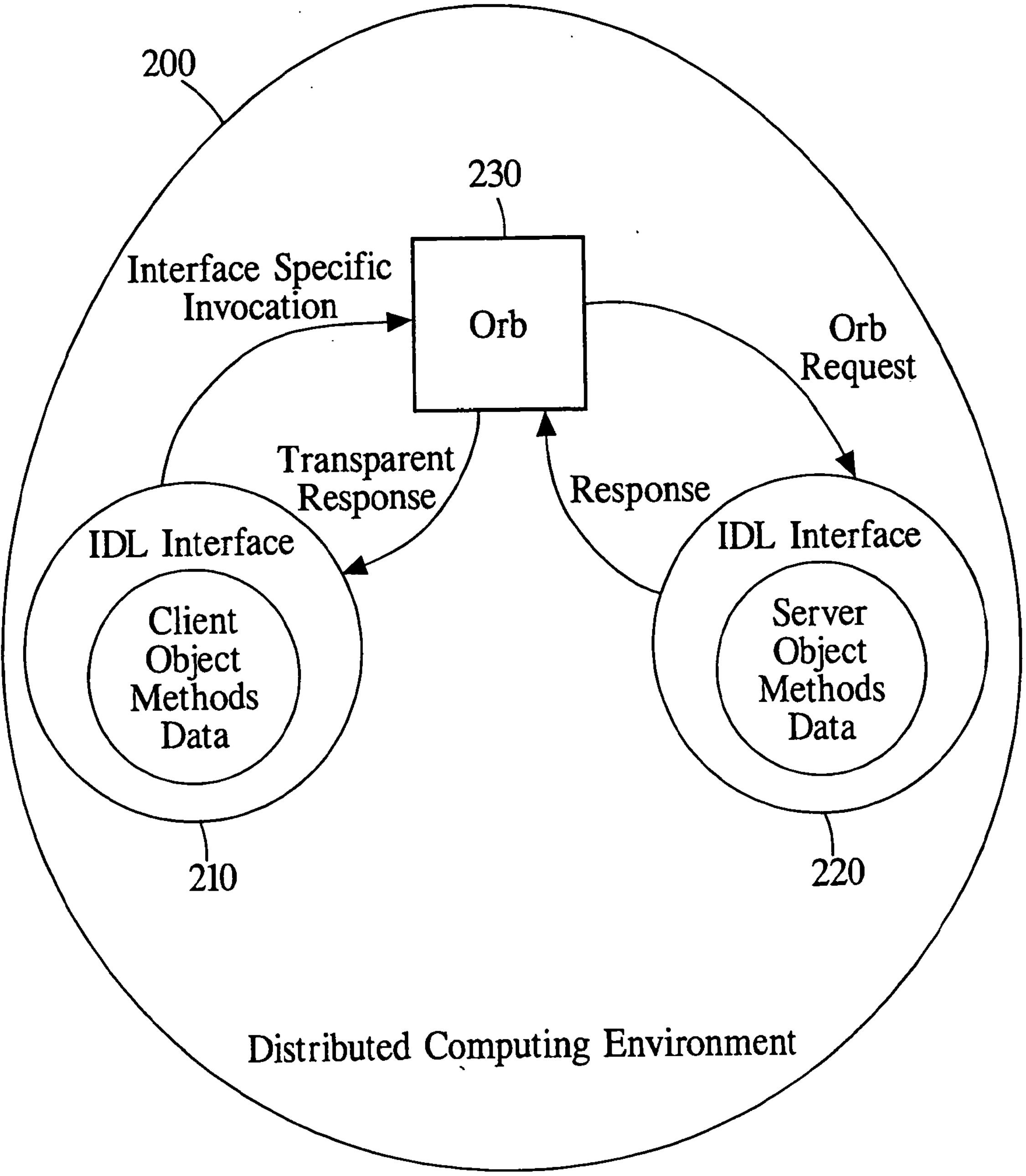

200
230
Orb
Interface Specific Invocation
Orb Request
Transparent Response
Response
IDL Interface
Client Object Methods Data
IDL Interface
Server Object Methods Data
210
220
Distributed Computing Environment

Fig. 2
(Prior Art)

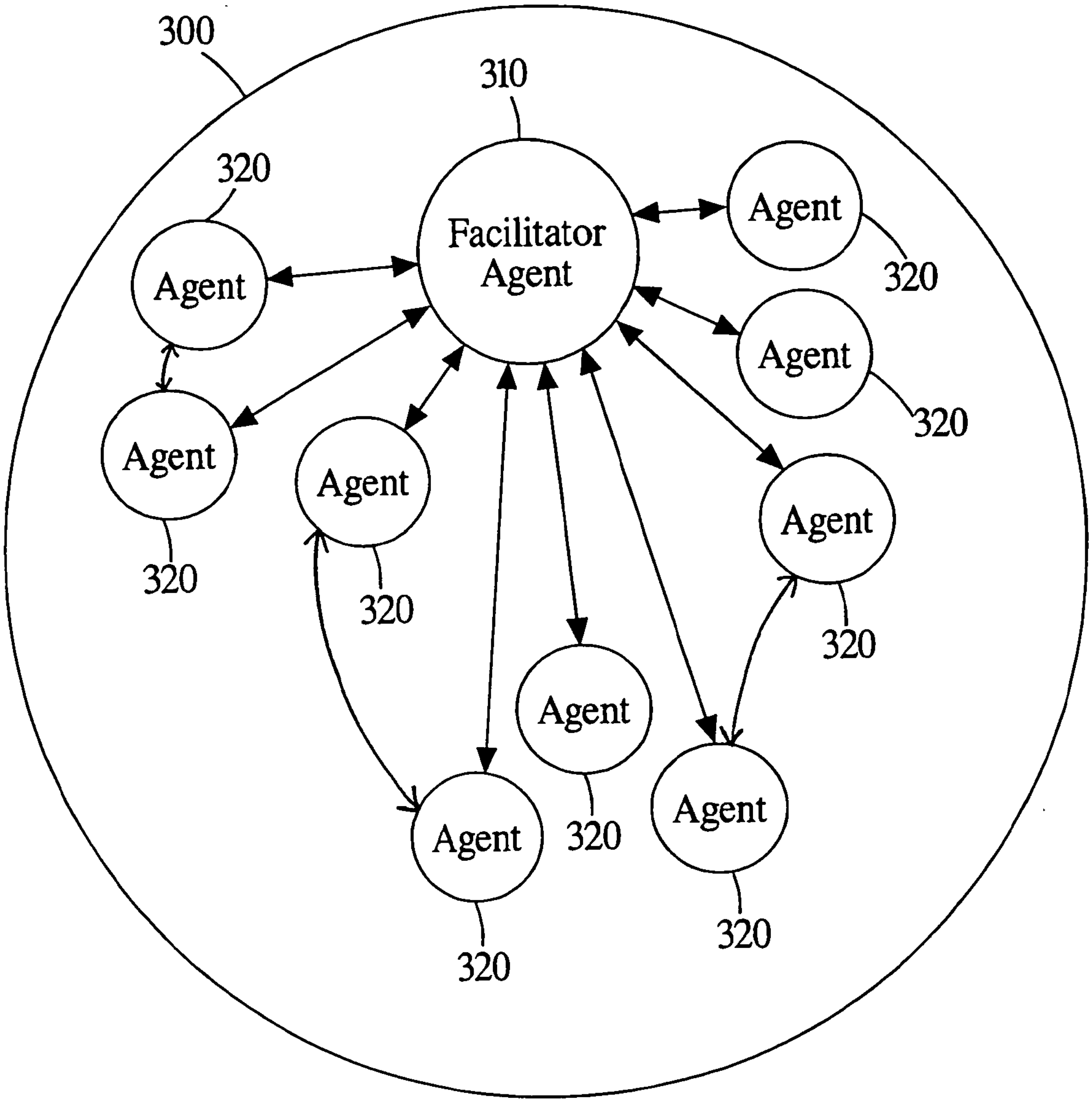

300
310
Facilitator Agent
Agent
Agent
Agent
Agent
Agent
Agent
Agent
Agent
Agent
Agent
320
320
320
320
320
320
320
320
320
320

Fig. 3

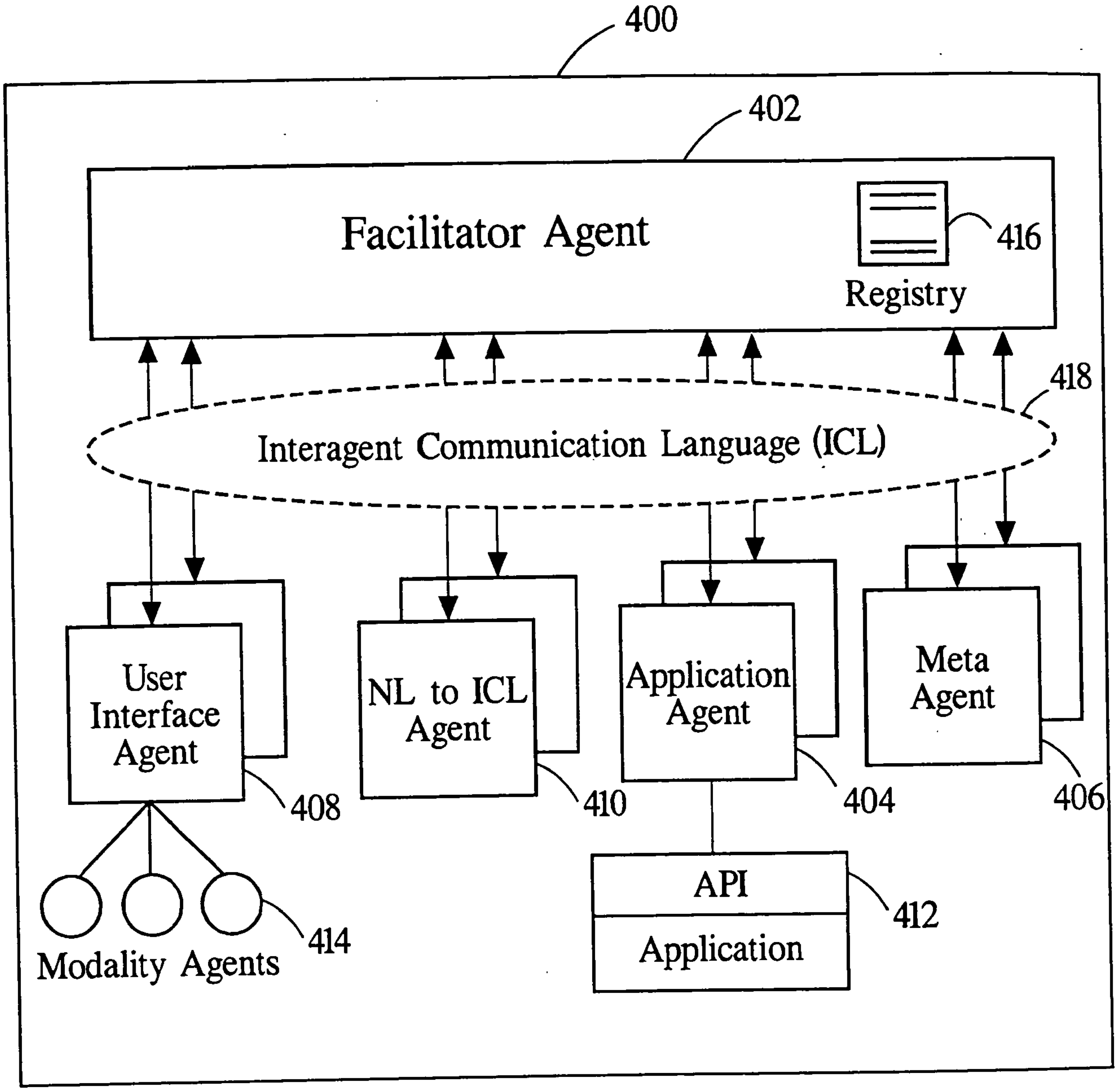
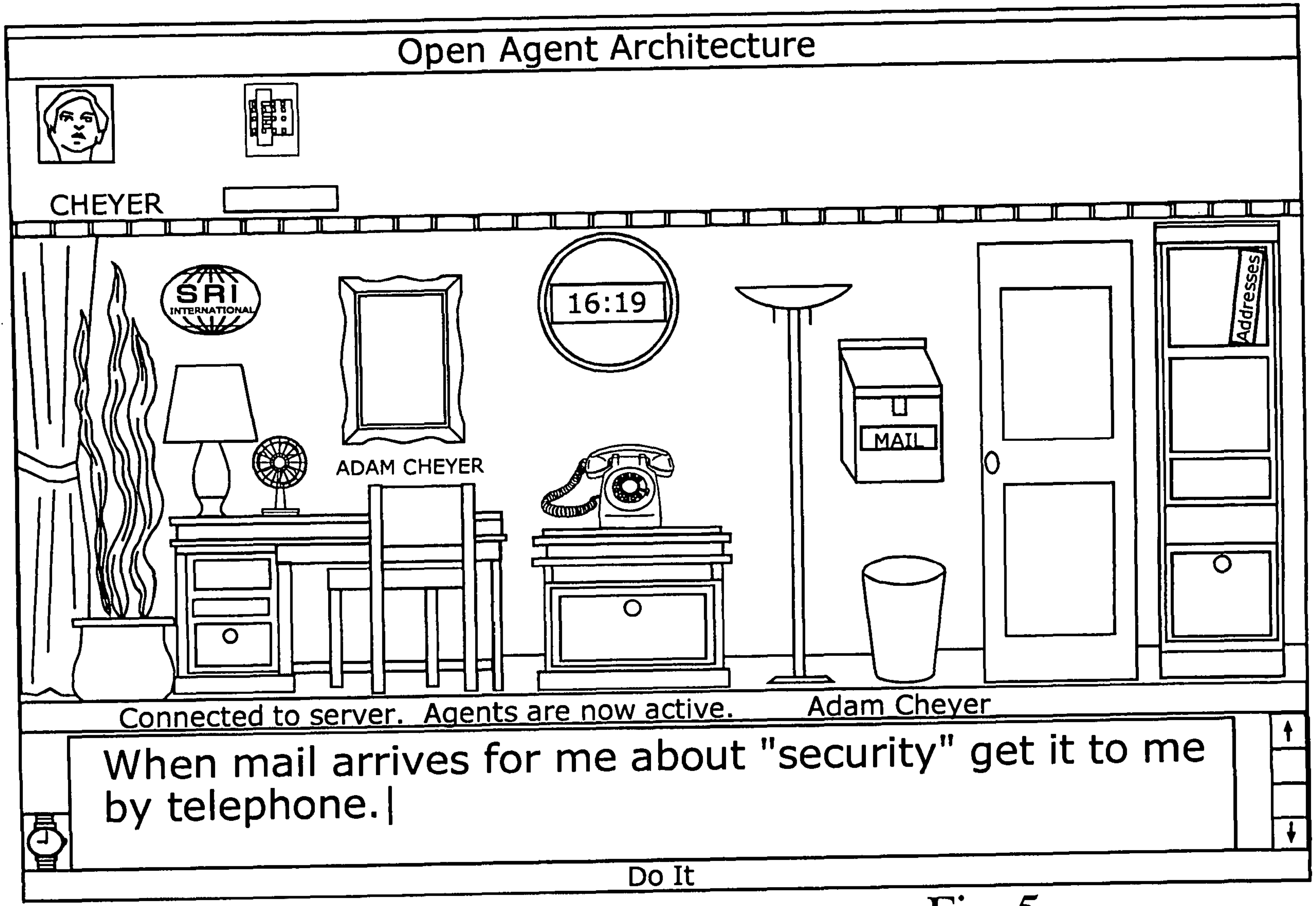

400
402
Facilitator Agent
Registry
416
418
Interagent Communication Language (ICL)
User Interface Agent
408
NL to ICL Agent
410
Application Agent
404
Meta Agent
406
Modality Agents
414
API
Application
412

Fig. 4



Connected to server. Agents are now active. Adam Cheyer

When mail arrives for me about "security" get it to me by telephone.|

Fig. 5

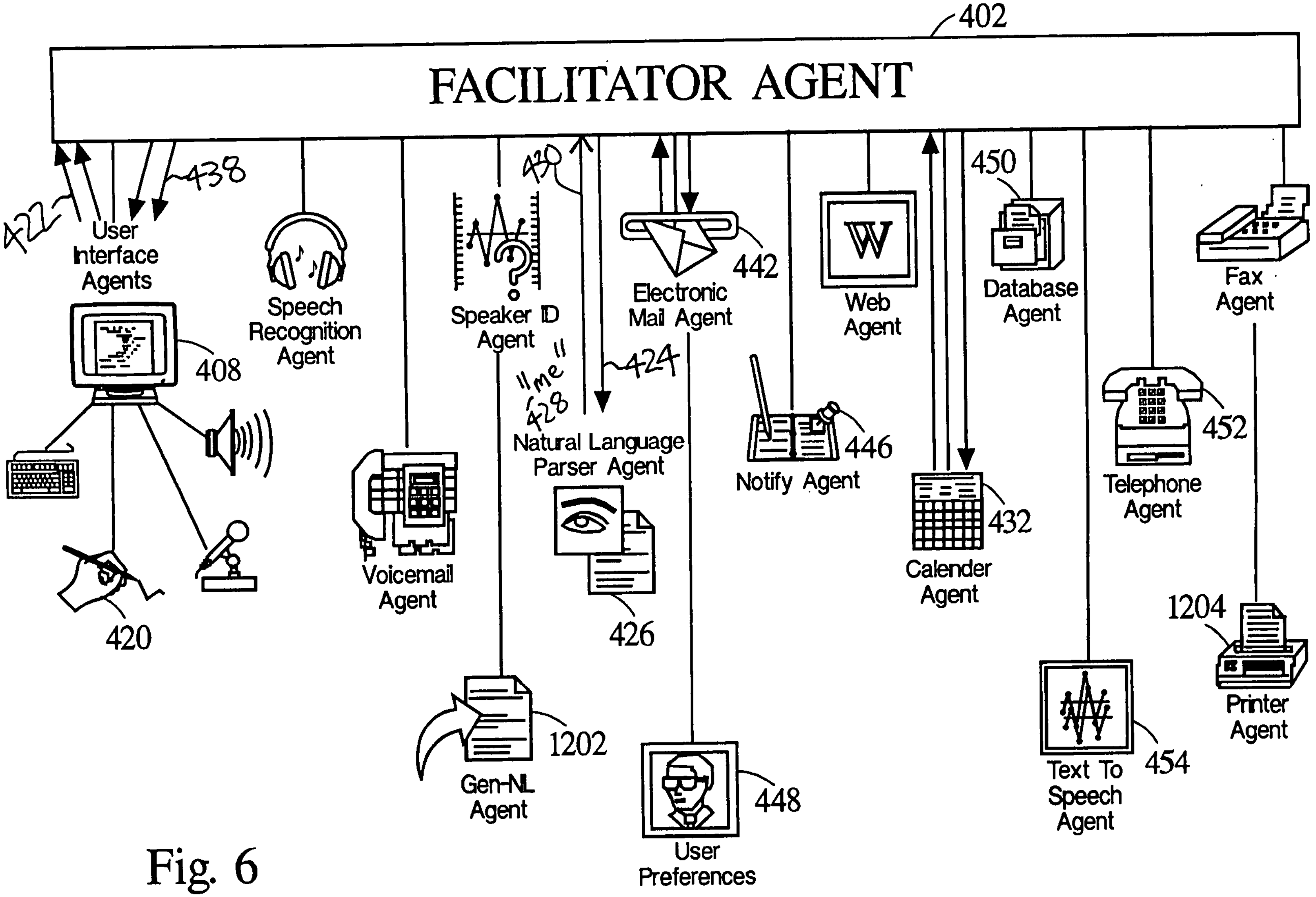

402
FACILITATOR AGENT
422
User Interface Agents
438
408
420
Speech Recognition Agent
Voicemail Agent
Speaker ID Agent
430
"me"
428
424
Natural Language Parser Agent
426
Gen-NL Agent
1202
Electronic Mail Agent
442
User Preferences
448
Web Agent
Notify Agent
446
Calender Agent
432
450
Database Agent
Text To Speech Agent
454
Telephone Agent
452
Fax Agent
Printer Agent
1204

Fig. 6

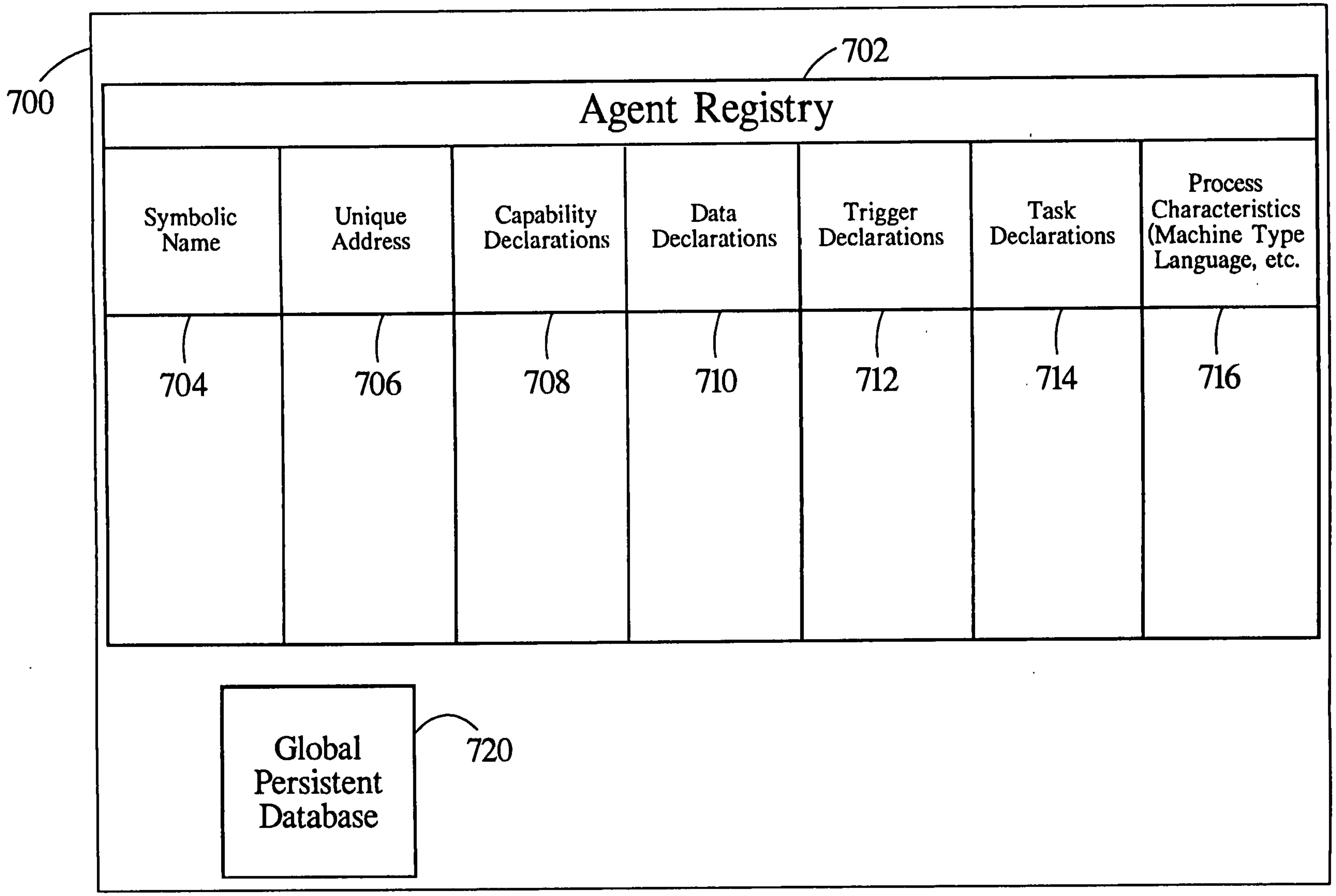

700
702
Agent Registry
Symbolic Name
Unique Address
Capability Declarations
Data Declarations
Trigger Declarations
Task Declarations
Process Characteristics (Machine Type Language, etc.
704
706
708
710
712
714
716
Global Persistent Database
720

Fig. 7

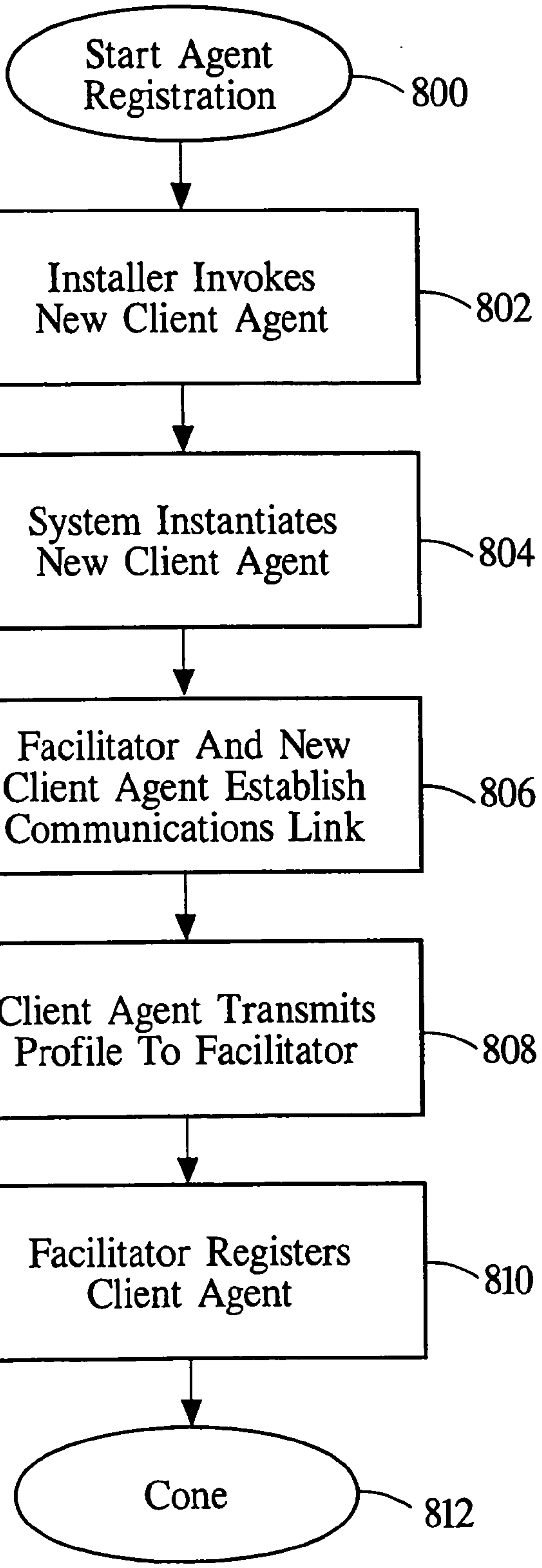

Start Agent Registration
800
Installer Invokes New Client Agent
802
System Instantiates New Client Agent
804
Facilitator And New Client Agent Establish Communications Link
806
Client Agent Transmits Profile To Facilitator
808
Facilitator Registers Client Agent
810
Cone
812

Fig. 8

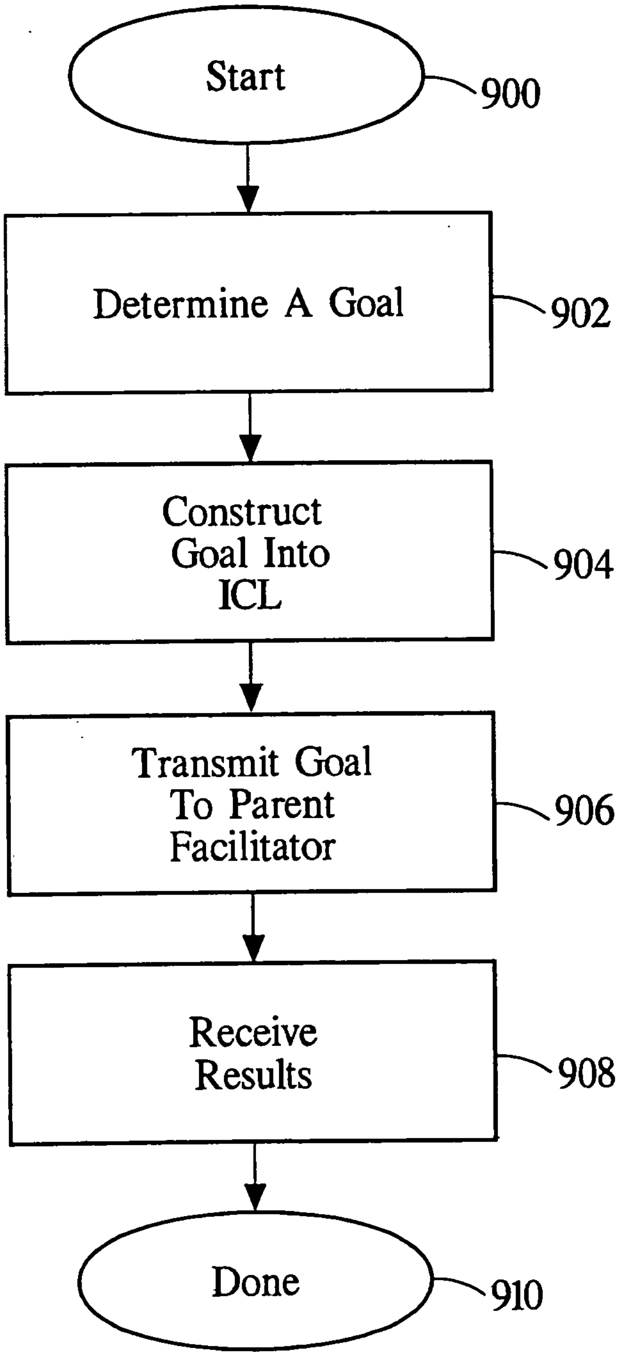

Start
900
Determine A Goal
902
Construct Goal Into ICL
904
Transmit Goal To Parent Facilitator
906
Receive Results
908
Done
910

Fig. 9

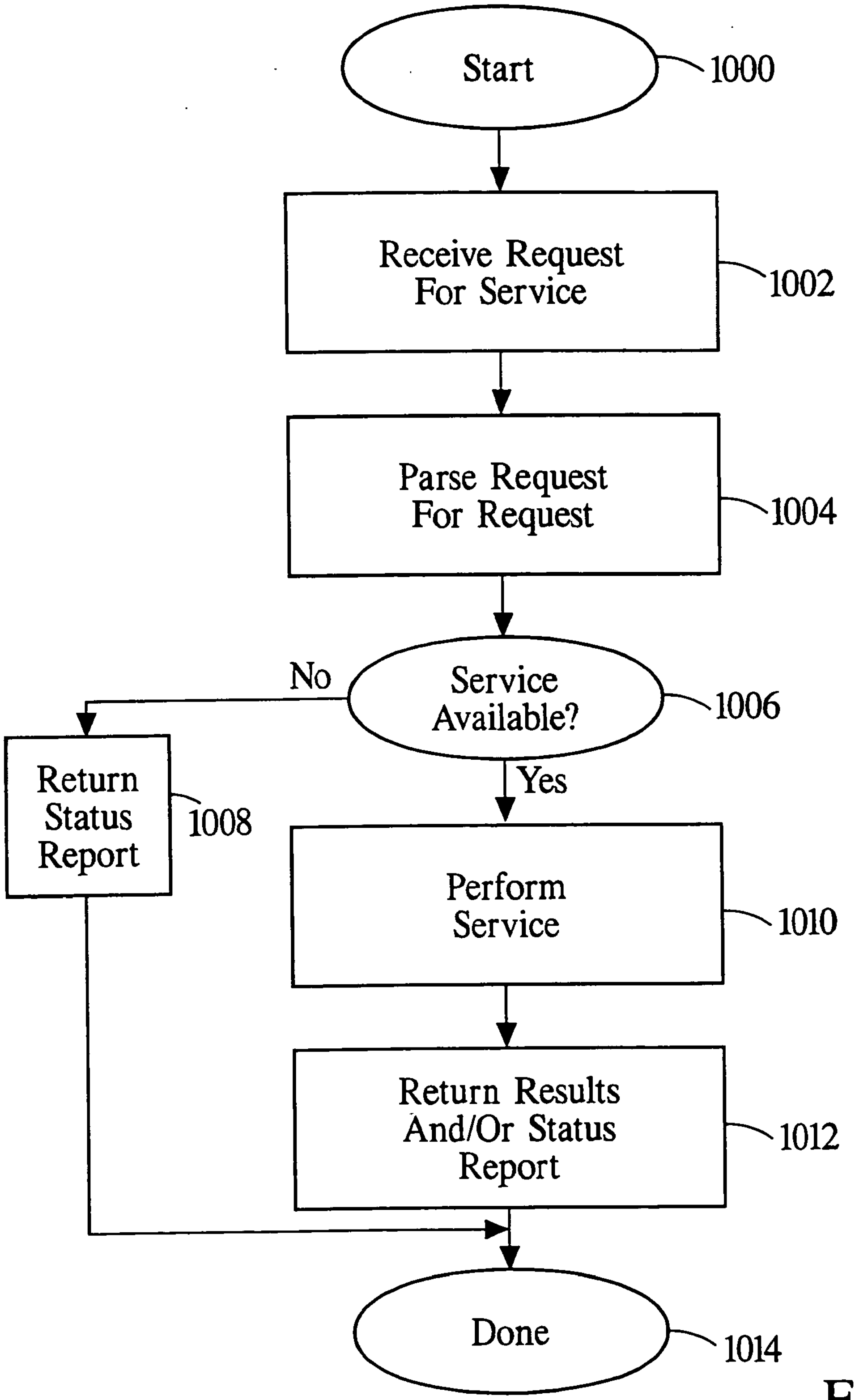

Start
1000
Receive Request For Service
1002
Parse Request For Request
1004
Service Available?
1006
No
Yes
Return Status Report
1008
Perform Service
1010
Return Results And/Or Status Report
1012
Done
1014

Fig. 10

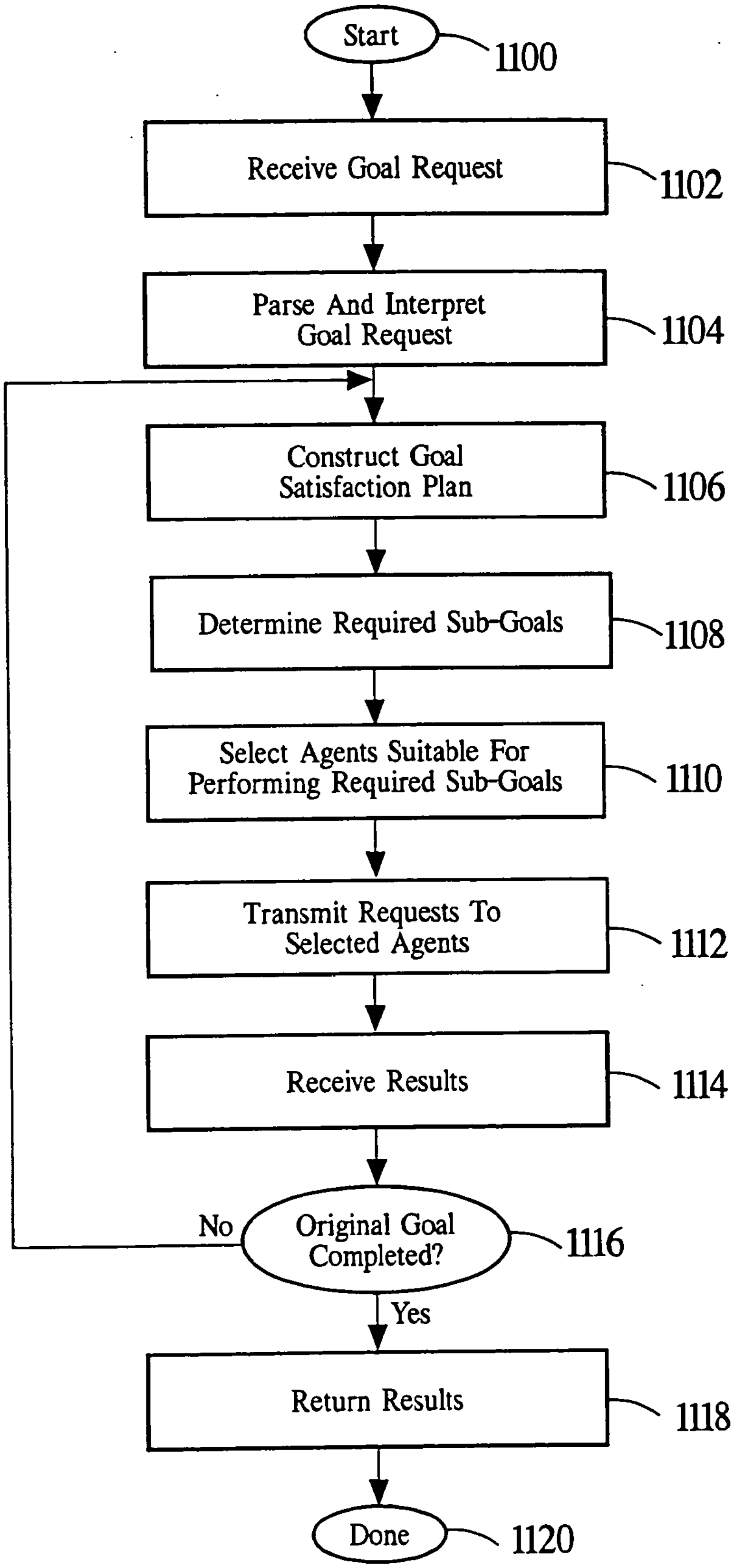

Start
1100
Receive Goal Request
1102
Parse And Interpret Goal Request
1104
Construct Goal Satisfaction Plan
1106
Determine Required Sub-Goals
1108
Select Agents Suitable For Performing Required Sub-Goals
1110
Transmit Requests To Selected Agents
1112
Receive Results
1114
No
Original Goal Completed?
1116
Yes
Return Results
1118
Done
1120

Fig. 11

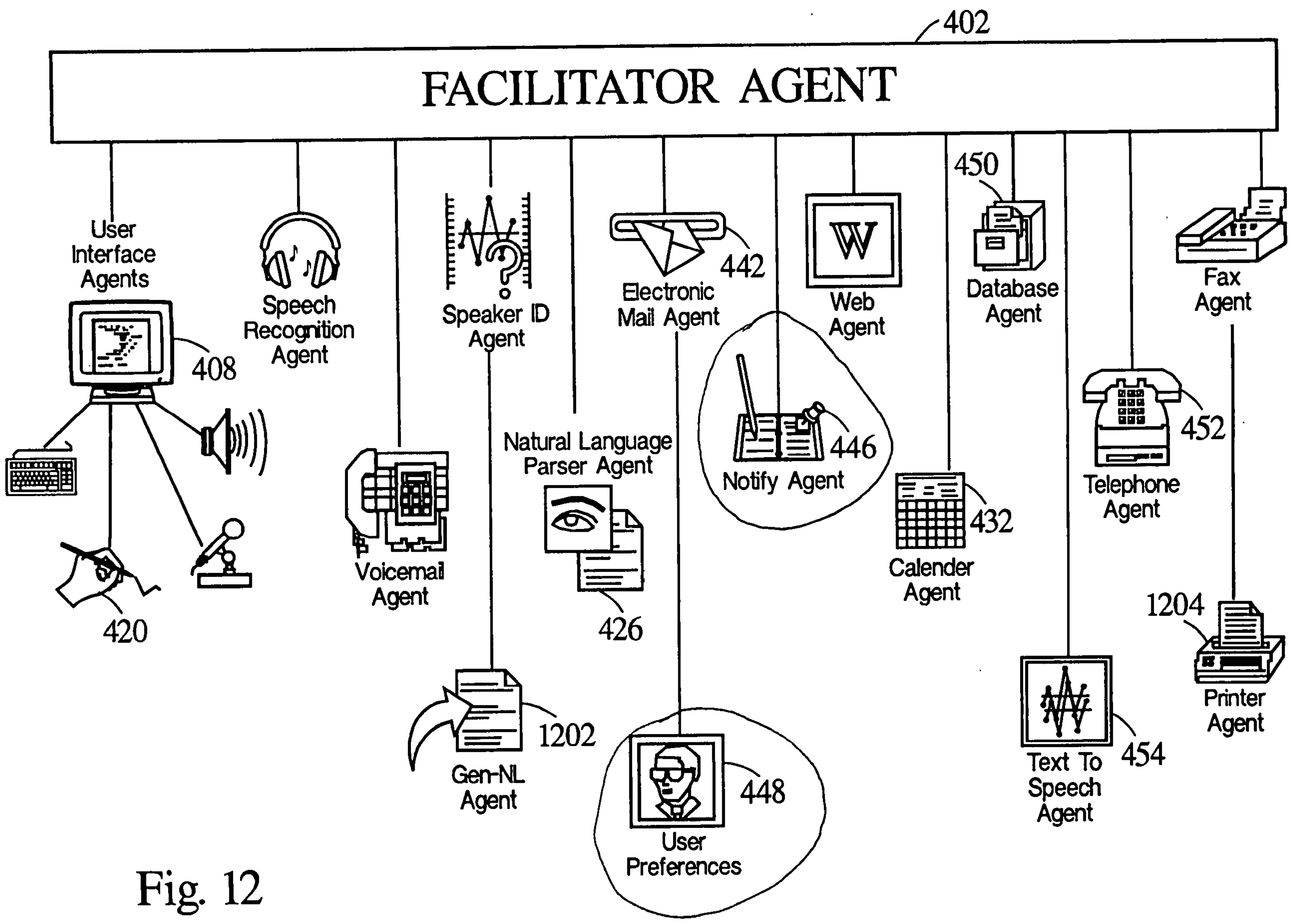

402
FACILITATOR AGENT
User Interface Agents
408
420
Speech Recognition Agent
Voicemail Agent
Speaker ID Agent
Gen-NL Agent
1202
Natural Language Parser Agent
426
Electronic Mail Agent
442
User Preferences
448
Notify Agent
446
Web Agent
Calender Agent
432
450
Database Agent
Text To Speech Agent
454
Telephone Agent
452
Fax Agent
Printer Agent
1204

Fig. 12

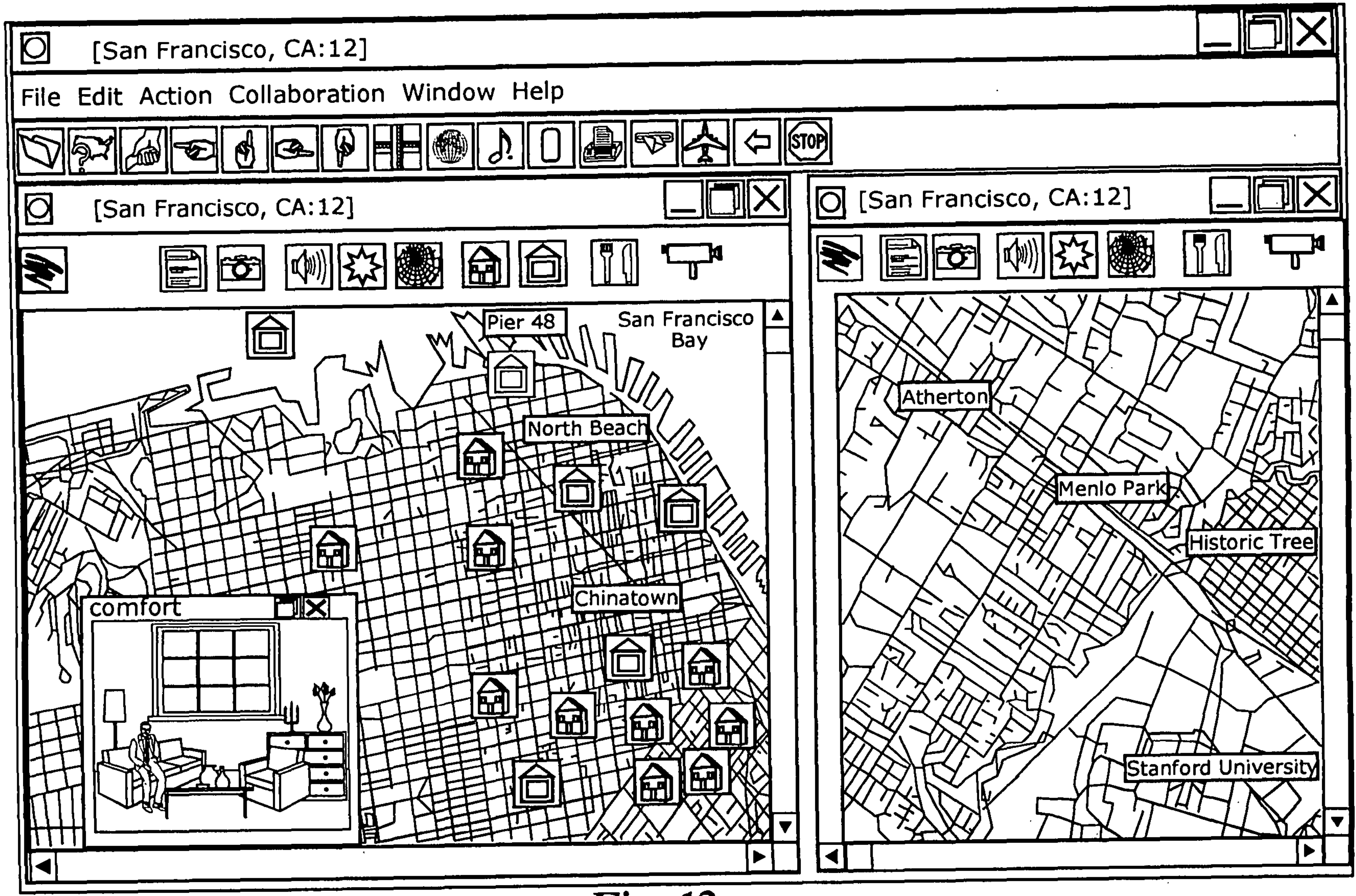

[San Francisco, CA:12]
File Edit Action Collaboration Window Help
STOP
[San Francisco, CA:12]
Pier 48
San Francisco Bay
North Beach
Chinatown
comfort
[San Francisco, CA:12]
Atherton
Menlo Park
Historic Tree
Stanford University

Fig. 13

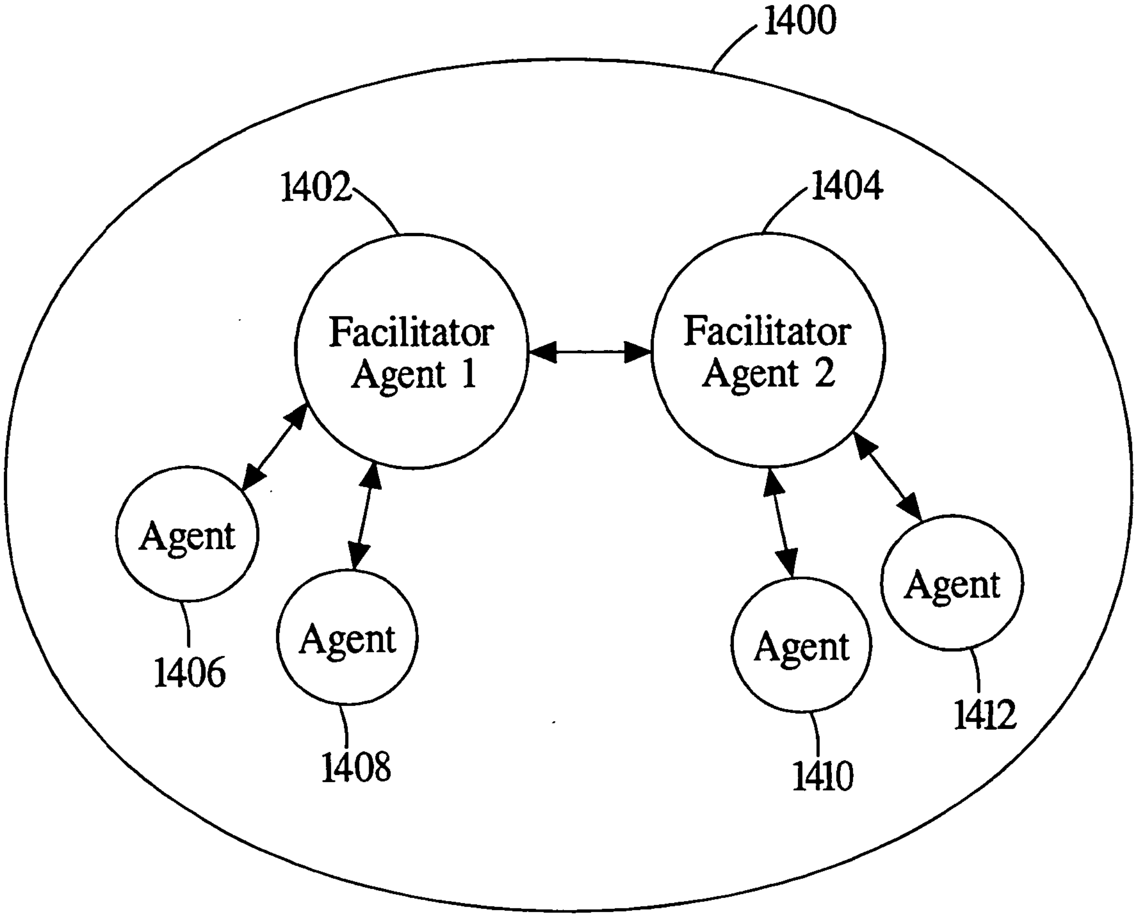

1400
1402
1404
Facilitator Agent 1
Facilitator Agent 2
Agent
1406
Agent
1408
Agent
1410
Agent
1412

Fig. 14

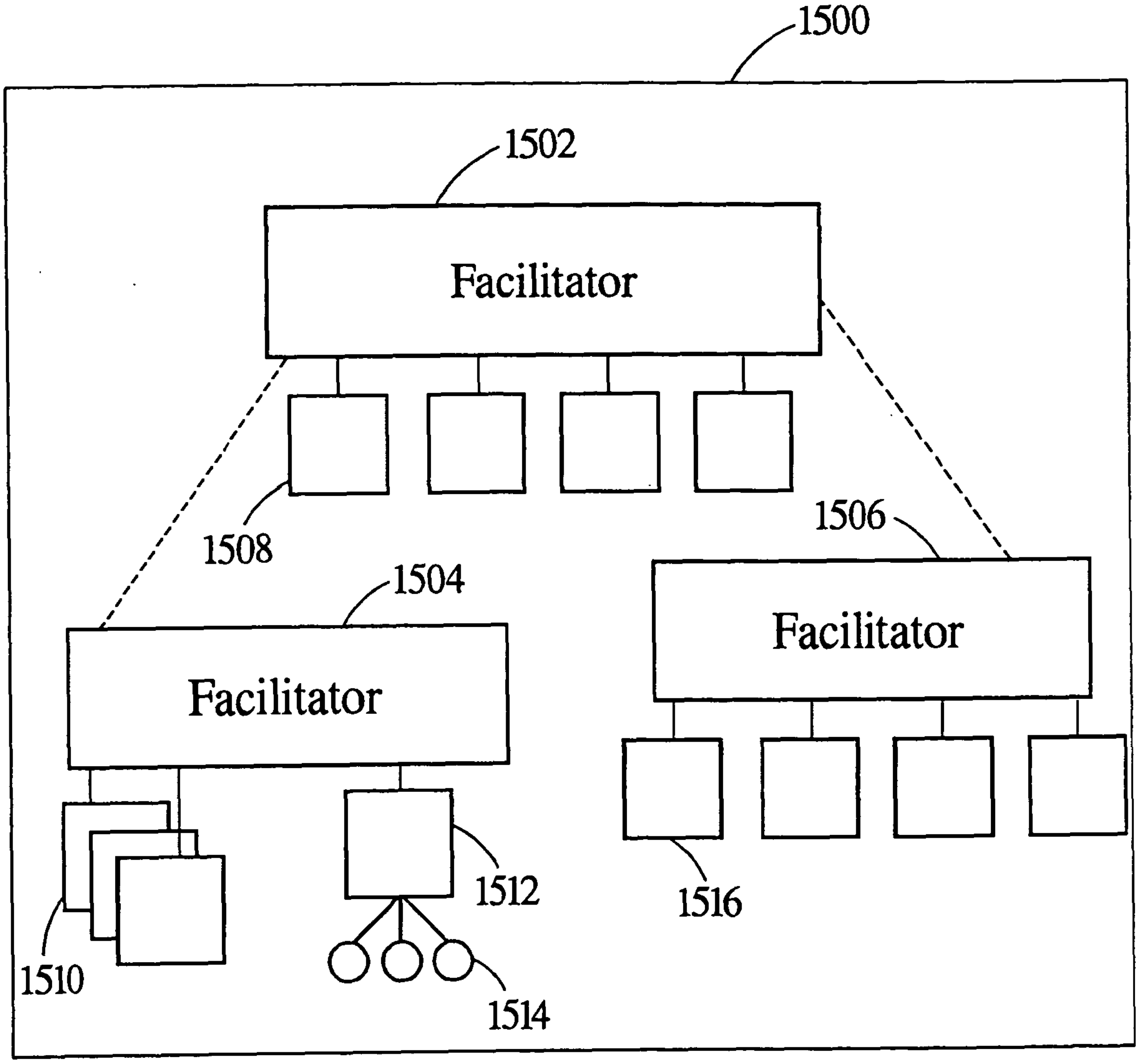

1500
1502
Facilitator
1508
1506
1504
Facilitator
Facilitator
1510
1512
1514
1516

Fig. 15

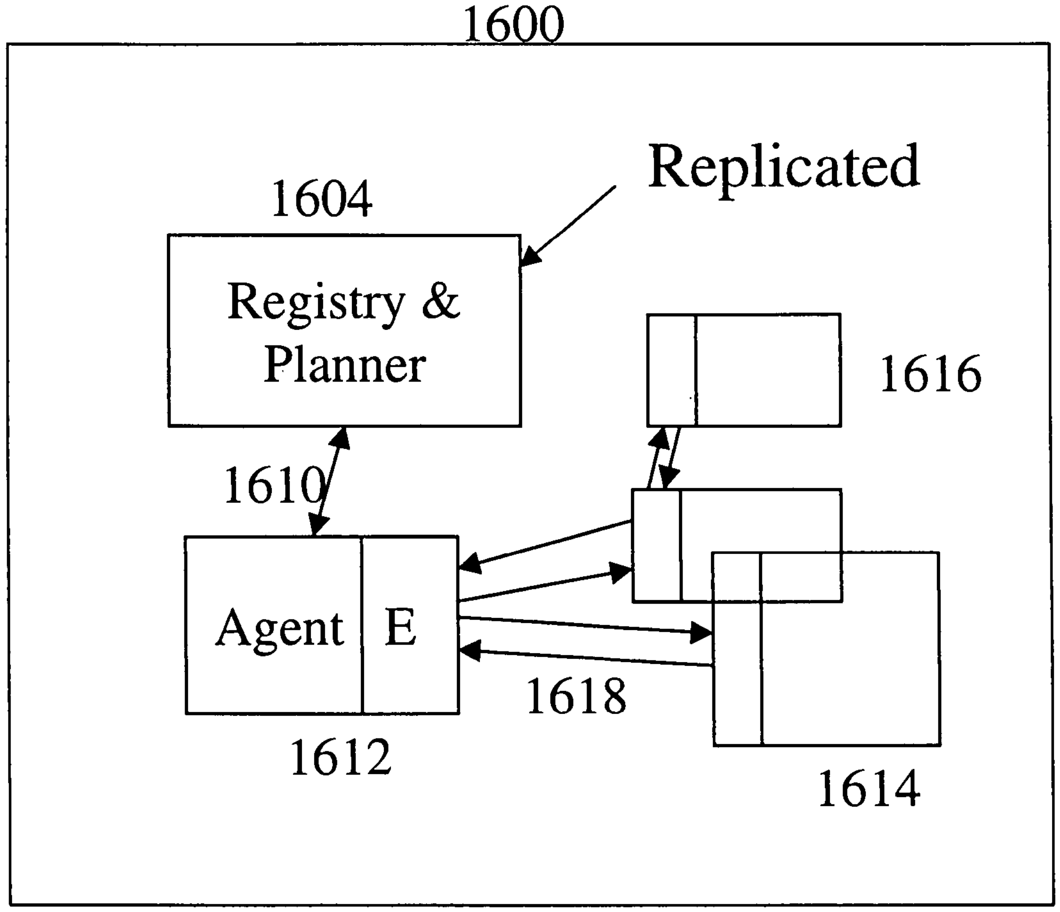


Figure 16

SRI1016 16/

Software-Based Architecture for Communication and Cooperation Among Distributed Electronic Agents

By:

Adam J. Cheyer and David L. Martin

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention is related to distributed computing environments and the completion of tasks within such environments. In particular, the present invention teaches a variety of software-based architectures for communication and cooperation among distributed electronic agents. Certain embodiments teach interagent communication languages enabling client agents to make requests in the form of arbitrarily complex goal expressions that are solved through facilitation by a facilitator agent.

Context and Motivation for Distributed Software Systems

The evolution of models for the design and construction of distributed software systems is being driven forward by several closely interrelated trends: the adoption of a *networked computing model*, rapidly rising expectations for *smarter, longer-lived, more autonomous software applications* and an ever increasing demand for *more accessible and intuitive user interfaces.*

Prior Art Figure 1 illustrates a *networked computing model* 100 having a plurality of client and server computer systems 120 and 122 coupled together over a physical transport mechanism 140. The adoption of the *networked computing model* 100 has lead to a greatly increased reliance on distributed sites for both data and processing resources. Systems such as the networked computing model 100 are based upon at least one physical transport mechanism 140 coupling the multiple computer systems 120 and 122 to support the transfer of information between these computers. Some of these computers basically support using the network and are known as *client* *computers* (*clients*). Some of these computers provide resources to other computers and are known as *server computers* (*servers*). The servers 122 can vary greatly in the resources they possess, access they provide and services made available to other computers across a network. Servers may service other servers as well as clients.

The Internet is a computing system based upon this network computing model. The Internet is continually growing, stimulating a paradigm shift for computing away from requiring all relevant data and programs to reside on the user's desktop machine. The data now routinely accessed from computers spread around the world has become increasingly rich in format, comprising multimedia documents, and audio and video streams. With the popularization of programming languages such as JAVA, data transported between local and remote machines may also include programs that can be downloaded and executed on the local machine. There is an ever increasing reliance on networked computing, necessitating software design approaches that allow for flexible composition of distributed processing elements in a dynamically changing and relatively unstable environment.

In an increasing variety of domains, application designers and users are coming to expect the deployment of *smarter, longer-lived, more autonomous, software applications.* Push technology, persistent monitoring of information sources, and the maintenance of user models, allowing for personalized responses and sharing of preferences, are examples of the simplest manifestations of this trend. Commercial enterprises are introducing significantly more advanced approaches, in many cases employing recent research results from artificial intelligence, data mining, machine learning, and other fields.

More than ever before, the increasing complexity of systems, the development of new technologies, and the availability of multimedia material and environments are creating a demand for *more accessible and intuitive user interfaces.* Autonomous, distributed, multi-component systems providing sophisticated services will no longer lend themselves to the familiar "direct manipulation" model of interaction, in which an individual user masters a fixed selection of commands provided by a single application. Ubiquitous computing, in networked environments, has brought about a situation in which the typical user of many software services is likely to be a non-expert, who may access a given service infrequently or only a few times.

Accommodating such usage patterns calls for new approaches. Fortunately, input modalities now becoming widely available, such as speech recognition and pen-based handwriting/gesture recognition, and the ability to manage the presentation of systems' responses by using multiple media provide an opportunity to fashion a style of human-computer interaction that draws much more heavily on our experience with human-human interactions.

PRIOR RELATED ART

Existing approaches and technologies for distributed computing include distributed objects, mobile objects, blackboard-style architectures, and agent-based software engineering.

The Distributed Object Approach

Object-oriented languages, such as C++ or JAVA, provide significant advances over standard procedural languages with respect to the reusability and modularity of code: *encapsulation*, *inheritance* and *polymorhpism*. Encapsulation encourages the creation of library interfaces that minimize dependencies on underlying algorithms or data structures. Changes to programming internals can be made at a later date with requiring modifications to the code that uses the library. Inheritance permits the extension and modification of a library of routines and data without requiring source code to the original library. Polymorphism allows one body of code to work on an arbitrary number of data types. For the sake of simplicity traditional objects may be seen to contain both methods and data. Methods provide the mechanisms by which the internal state of an object may be modified or by which communication may occur with another object or by which the instantiation or removal of objects may be directed.

With reference to Figure 2, a distributed object technology based around an Object Request Broker will now be described. Whereas "standard" object-oriented programming (OOP) languages can be used to build monolithic programs out of many object building blocks, distributed object technologies (DOOP) allow the creation of programs whose components may be spread across multiple machines. As shown in Figure 2, an object system 200 includes client objects 210 and server objects 220. To implement a client-server relationship between objects, the distributed object system

200 uses a registry mechanism (CORBA's registry is called an object Request Broker, or ORB) 230 to store the interface descriptions of available objects. Through the services of the ORB 230, a client can transparently invoke a method on a remote server object. The ORB 230 is then responsible for finding the object 220 that can implement the request, passing it the parameters, invoking its method, and returning the results. In the most sophisticated systems, the client 210 does not have to be aware of where the object is located, its programming language, its operating system, or any other system aspects that are not part of the server object's interface.

Although distributed objects offer a powerful paradigm for creating networked applications, certain aspects of the approach are not perfectly tailored to the constantly changing environment of the Internet. A major restriction of the DOOP approach is that the interactions among objects are fixed through explicitly coded instructions by the application developer. It is often difficult to reuse an object in a new application without bringing along all its inherent dependencies on other objects (embedded interface definitions and explicit method calls). Another restriction of the DOOP approach is the result of its reliance on a remote procedure call (RPC) style of communication. Although easy to debug, this single thread of execution model does not facilitate programming to exploit the potential for parallel computation that one would expect in a distributed environment. In addition, RPC uses a blocking (synchronous) scheme that does not scale well for high-volume transactions.

Mobile Objects

Mobile objects, sometimes called mobile agents, are bits of code that can move to another execution site (presumably on a different machine) under their own programmatic control, where they can then interact with the local environment. For certain types of problems, the mobile object paradigm offers advantages over more traditional distributed object approaches. These advantages include network bandwidth and parallelism. Network bandwidth advantages exist for some database queries or electronic commerce applications, where it is more efficient to perform tests on data by bringing the tests to the data than by bringing large amounts of data to the testing program. Parallelism advantages include situations in which mobile agents can be spawned in parallel to accomplish many tasks at once.

Some of the disadvantages and inconveniences of the mobile agent approach include the programmatic specificity of the agent interactions, lack of coordination support between participant agents and execution environment irregularities regarding specific programming languages supported by host processors upon which agents reside. In a fashion similar to that of DOOP programming, an agent developer must programmatically specify where to go and how to interact with the target environment. There is generally little coordination support to encourage interactions among multiple (mobile) participants. Agents must be written in the programming language supported by the execution environment, whereas many other distributed technologies support heterogeneous communities of components, written in diverse programming languages.

Blackboard Architectures

Blackboard architectures typically allow multiple processes to communicate by reading and writing tuples from a global data store. Each process can watch for items of interest, perform computations based on the state of the blackboard, and then add partial results or queries that other processes can consider. Blackboard architectures provide a flexible framework for problem solving by a dynamic community of distributed processes. A blackboard architecture provides one solution to eliminating the tightly bound interaction links that some of the other distributed technologies require during interprocess communication. This advantage can also be a disadvantage: although a programmer does not need to refer to a specific process during computation, the framework does not provide programmatic control for doing so in cases where this would be practical.

Agent-based Software Engineering

Several research communities have approached distributed computing by casting it as a problem of modeling communication and cooperation among autonomous entities, or agents. Effective communication among independent agents requires four components: (1) a transport mechanism carrying messages in an asynchronous fashion, (2) an interaction protocol defining various types of communication interchange and their social implications (for instance, a response is expected of a question), (3) a content language permitting the expression and interpretation of utterances, and (4) an agreed-upon set of shared vocabulary and meaning for concepts (often called an *ontology*). Such mechanisms permit a much richer style of interaction among participants than can be expressed using a distributed object's RPC model or a blackboard architecture's centralized exchange approach.

Agent-based systems have shown much promise for flexible, fault-tolerant, distributed problem solving. Several agent-based projects have helped to evolve the notion of facilitation. However, existing agent-based technologies and architectures are typically very limited in the extent to which agents can specify complex goals or influence the strategies used by the facilitator. Further, such prior systems are not sufficiently attuned to the importance of integrating human agents (i.e., users) through natural language and other human-oriented user interface technologies.

The initial version of SRI International's Open Agent Architecture™ ("*OAA*®") technology provided only a very limited mechanism for dealing with compound goals. Fixed formats were available for specifying a flat list of either conjoined (AND) sub-goals or disjoined (OR) sub-goals; in both cases, parallel goal solving was hard-wired in, and only a single set of parameters for the entire list could be specified. More complex goal expressions involving (for example) combinations of different boolean connectors, nested expressions, or conditionally interdependent ("IF .. THEN") goals were not supported. Further, system scalability was not adequately addressed in this prior work.

SUMMARY OF INVENTION

A first embodiment of the present invention discloses a highly flexible, software-based architecture for constructing distributed systems. The architecture supports cooperative task completion by flexible, dynamic configurations of autonomous electronic agents. Communication and cooperation between agents are brokered by one or more facilitators, which are responsible for matching requests, from users and agents, with descriptions of the capabilities of other agents. It is not generally required that a user or agent know the identities, locations, or number of other agents involved in satisfying a request, and relatively minimal effort is involved in incorporating new agents and "wrapping" legacy applications. Extreme flexibility is achieved through an architecture organized around the declaration of capabilities by service-providing agents, the construction of arbitrarily complex goals by users and service-requesting agents, and the role of facilitators in delegating and coordinating the satisfaction of these goals, subject to advice and constraints that may accompany them. Additional mechanisms and features include facilities for creating and maintaining shared repositories of data; the use of triggers to instantiate commitments within and between agents; agent-based provision of multi-modal user interfaces, including natural language; and built-in support for including the user as a privileged member of the agent community. Specific embodiments providing enhanced scalability are also described.

BRIEF DESCRIPTION OF THE DRAWINGS

Prior Art

Prior Art FIGURE 1 depicts a networked computing model;

Prior Art FIGURE 2 depicts a distributed object technology based around an Object Resource Broker;

Examples of the Invention

FIGURE 3 depicts a distributed agent system based around a facilitator agent;

FIGURE 4 presents a structure typical of one small system of the present invention;

FIGURE 5 depicts an Automated Office system implemented in accordance with an example embodiment of the present invention supporting a mobile user with a laptop computer and a telephone;

FIGURE 6 schematically depicts an Automated Office system implemented as a network of agents in accordance with a preferred embodiment of the present invention;

FIGURE 7 schematically shows data structures internal to a facilitator in accordance with a preferred embodiment of the present invention;

FIGURE 8 depicts operations involved in instantiating a client agent with its parent facilitator in accordance with a preferred embodiment of the present invention;

FIGURE 9 depicts operations involved in a client agent initiating a service request and receiving the response to that service request in accordance with a certain preferred embodiment of the present invention;

FIGURE 10 depicts operations involved in a client agent responding to a service request in accordance with another preferable embodiment of the present invention;

FIGURE 11 depicts operations involved in a facilitator agent response to a service request in accordance with a preferred embodiment of the present invention;

FIGURE 12 depicts an Open Agent Architecture™ based system of agents implementing a unified messaging application in accordance with a preferred embodiment of the present invention;

FIGURE 13 depicts a map oriented graphical user interface display as might be displayed by a multi-modal map application in accordance with a preferred embodiment of the present invention;

FIGURE 14 depicts a peer to peer multiple facilitator based agent system supporting distributed agents in accordance with a preferred embodiment of the present invention;

FIGURE 15 depicts a multiple facilitator agent system supporting at least a limited form of a hierarchy of facilitators in accordance with a preferred embodiment of the present invention; and

FIGURE 16 depicts a replicated facilitator architecture in accordance with one embodiment of the present invention.

BRIEF DESCRIPTION OF THE APPENDICES

The Appendices provide source code for an embodiment of the present invention written in the PROLOG programming language.

APPENDIX A: Source code file named compound.pl.

APPENDIX B: Source code file named fac.pl.

APPENDIX C: Source code file named libcom_tcp.pl.

APPENDIX D: Source code file named liboaa.pl.

APPENDIX E: Source code file named translations.pl.

DETAILED DESCRIPTION OF THE INVENTION

Figure 3 illustrates a distributed agent system 300 in accordance with one embodiment of the present invention. The agent system 300 includes a facilitator agent 310 and a plurality of agents 320. The illustration of Figure 3 provides a high level view of one simple system structure contemplated by the present invention. The facilitator agent 310 is in essence the "parent" facilitator for its "children" agents 320. The agents 320 forward service requests to the facilitator agent 310. The facilitator agent 310 interprets these requests, organizing a set of goals which are then delegated to appropriate agents for task completion.

The system 300 of Figure 3 can be expanded upon and modified in a variety of ways consistent with the present invention. For example, the agent system 300 can be distributed across a computer network such as that illustrated in Figure 1. The facilitator agent 310 may itself have its functionality distributed across several different computing platforms. The agents 320 may engage in interagent communication (also called peer to peer communications). Several different systems 300 may be coupled together for enhanced performance. These and a variety of other structural configurations are described below in greater detail.

Figure 4 presents the structure typical of a small system 400 in one embodiment of the present invention, showing user interface agents 408, several application agents 404 and meta-agents 406, the system 400 organized as a community of peers by their common relationship to a facilitator agent 402. As will be appreciated, Figure 4 places more structure upon the system 400 than shown in Figure 3, but both are valid representations of structures of the present invention. The facilitator 402 is a specialized server agent that is responsible for coordinating agent communications and cooperative problem-solving. The facilitator 402 may also provide a global data store for its client agents, allowing them to adopt a blackboard style of interaction. Note that certain advantages are found in utilizing two or more facilitator agents within the system 400. For example, larger systems can be assembled from multiple facilitator/client groups, each having the sort of structure shown in Figure 4. All agents that are not facilitators are referred to herein generically as *client* agents -- so called because each acts (in some respects) as a client of some facilitator, which provides communication and other essential services for the client.

The variety of possible client agents is essentially unlimited. Some typical categories of client agents would include application agents 404, meta-agents 406, and user interface agents 408, as depicted in Figure 4. Application agents 404 denote specialists that provide a collection of services of a particular sort. These services could be domain-independent technologies (such as speech recognition, natural language processing 410, email, and some forms of data retrieval and data mining) or user-specific or domain-specific (such as a travel planning and reservations agent). Application agents may be based on legacy applications or libraries, in which case the agent may be little more than a wrapper that calls a pre-existing API 412, for example. Meta-agents 406 are agents whose role is to assist the facilitator agent 402 in coordinating the activities of other agents. While the facilitator 402 possesses domain-independent coordination strategies, meta-agents 406 can augment these by using domain- and application-specific knowledge or reasoning (including but not limited to rules, learning algorithms and planning).

With further reference to Figure 4, user interface agents 408 can play an extremely important and interesting role in certain embodiments of the present invention. By way of explanation, in some systems, a user interface agent can be implemented as a collection of "micro-agents", each monitoring a different input modality (point-and-click, handwriting, pen gestures, speech), and collaborating to produce the best interpretation of the current inputs. These micro-agents are depicted in Figure 4, for example, as Modality Agents 414. While describing such subcategories of client agents is useful for purposes of illustration and understanding, they need not be formally distinguished within the system in preferred implementations of the present invention.

The operation of one preferred embodiment of the present invention will be discussed in greater detail below, but may be briefly outlined as follows. When invoked, a client agent makes a connection to a facilitator, which is known as its *parent facilitator*. These connections are depicted as a double headed arrow between the client agent and the facilitator agent in Figure 3 and 4, for example. Upon connection, an agent registers with its parent facilitator a specification of the capabilities and services it can provide. For example, a natural language agent may register the characteristics of its available natural language vocabulary. (For more details regarding client agent connections, see the discussion of Figure 8 below.) Later during task completion, when a facilitator determines that the registered services 416 of one of its client agents will help satisfy a goal, the facilitator sends that client a request expressed in the Interagent Communication Language (*ICL*) 418. (See Figure 11 below for a more detailed discussion of the facilitator operations involved.) The agent parses this request, processes it, and returns answers or status reports to the facilitator. In processing a request, the client agent can make use of a variety of infrastructure capabilities provided in the preferred embodiment. For example, the client agent can use *ICL* 418 to request services of other agents, set triggers, and read or write shared data on the facilitator or other client agents that maintain shared data. (See the discussion of Figures 9-11 below for a more detailed discussion of request processing.)

The functionality of each client agent are made available to the agent community through registration of the client agent's capabilities with a facilitator 402. A software "wrapper" essentially surrounds the underlying application program performing the services offered by each client. The common infrastructure for constructing agents is preferably supplied by an *agent library*. The agent library is preferably accessible in the runtime environment of several different programming languages. The agent library preferably minimizes the effort required to construct a new system and maximizes the ease with which legacy systems can be “wrapped” and made compatible with the agent-based architecture of the present invention.

By way of further illustration, a representative application is now briefly presented with reference to Figures 5 and 6. In the Automated Office system depicted in Figure 5, a mobile user with a telephone and a laptop computer can access and task commercial applications such as calendars, databases, and email systems running back at the office. A user interface (UI) agent 408, shown in Figure 6, runs on the user's local laptop and is responsible for accepting user input, sending requests to the facilitator 402 for delegation to appropriate agents, and displaying the results of the distributed computation. The user may interact directly with a specific remote application by clicking on active areas in the interface, calling up a form or window for that application, and making queries with standard interface dialog mechanisms. Conversely, a user may express a task to be executed by using typed, handwritten, or spoken (over the telephone) English sentences, without explicitly specifying which agent or agents should perform the task.

For instance, if the question "What is my schedule?" is written 420 in the user interface 408, this request will be sent 422 by the UI 408 to the facilitator 402, which in turn will ask 424 a natural language (NL) agent 426 to translate the query into *ICL* 18. To accomplish this task, the NL agent 426 may itself need to make requests of the agent community to resolve unknown words such as "me" 428 (the UI agent 408 can respond 430 with the name of the current user) or "schedule" 432 (the calendar agent 434 defines this word 436). The resulting *ICL* expression is then routed by the facilitator 402 to appropriate agents (in this case, the calendar agent 434) to execute the request. Results are sent back 438 to the UI agent 408 for display.

The spoken request "When mail arrives for me about security, notify me immediately." produces a slightly more complex example involving communication among all agents in the system. After translation into *ICL* as described above, the facilitator installs a trigger 440 on the mail agent 442 to look for new messages about security. When one such message does arrive in its mail spool, the trigger fires, and the facilitator matches the action part of the trigger to capabilities published by the notification agent 446. The notification agent 446 is a meta-agent, as it makes use of rules concerning the optimal use of different output modalities (email, fax, speech generation over the telephone) plus information about an individual user's preferences 448 to determine the best way of relaying a message through available media transfer application agents. After some competitive parallelism to locate the user (the calendar agent 434 and database agent 450 may have different guesses as to where to find the user) and some cooperative parallelism to produce required information (telephone number of location, user password, and an audio file containing a text-to-speech representation of the email message), a telephone agent 452 calls the user, verifying its identity through touchtones, and then play the message.

The above example illustrates a number of inventive features. As new agents connect to the facilitator, registering capability specifications and natural language vocabulary, what the user can say and do dynamically changes; in other words, the ICL is dynamically ***expandable***. For example, adding a calendar agent to the system in the previous example and registering its capabilities enables users to ask natural language questions about their "schedule" without any need to revise code for the facilitator, the natural language agents, or any other client agents. In addition, the interpretation and execution of a task is a distributed process, with no single agent defining the set of possible inputs to the system. Further, a single request can produce cooperation and flexible communication among many agents, written in different programming languages and spread across multiple machines.

Design Philosophy and Considerations

One preferred embodiment provides an integration mechanism for heterogeneous applications in a distributed infrastructure, incorporating some of the dynamism and extensibility of blackboard approaches, the efficiency associated with mobile objects, plus the rich and complex interactions of communicating agents. Design goals for preferred embodiments of the present invention may be categorized under the general headings of *interoperation and cooperation, user interfaces*, and *software engineering*. These design goals are not absolute requirements, nor will they necessarily be satisfied by all embodiments of the present invention, but rather simply reflect the inventor's currently preferred design philosophy.

Versatile mechanisms of interoperation and cooperation

Interoperation refers to the ability of distributed software components - agents - to communicate meaningfully. While every system-building framework must provide mechanisms of interoperation at some level of granularity, agent-based frameworks face important new challenges in this area. This is true primarily because autonomy, the hallmark of *individual* agents, necessitates greater flexibility in interactions within *communities* of agents. *Coordination* refers to the mechanisms by which a community of agents is able to work together productively on some task. In these areas, the goals for our framework are to *provide flexibility in assembling* *communities of autonomous service providers, provide flexibility in structuring cooperative interactions, impose the right amount of structure*, as well as *include legacy and "owned-elsewhere" applications.*

Provide flexibility in assembling communities of autonomous service providers -- both at development time and at runtime. Agents that conform to the linguistic and ontological requirements for effective communication should be able to participate in an agent community, in various combinations, with minimal or near minimal prerequisite knowledge of the characteristics of the other players. Agents with duplicate and overlapping capabilities should be able to coexist within the same community, with the system making optimal or near optimal use of the redundancy.

Provide flexibility in structuring cooperative interactions among the members of a community of agents. A framework preferably provides an economical mechanism for setting up a variety of interaction patterns among agents, without requiring an inordinate amount of complexity or infrastructure within the individual agents. The provision of a service should be independent or minimally dependent upon a particular configuration of agents.

Impose the right amount of structure on individual agents. Different approaches to the construction of multi-agent systems impose different requirements on the individual agents. For example, because KQML is neutral as to the content of messages, it imposes minimal structural requirements on individual agents. On the other hand, the BDI paradigm tends to impose much more demanding requirements, by making assumptions about the nature of the programming elements that are meaningful to individual agents. Preferred embodiments of the present invention should fall somewhere between the two, providing a rich set of interoperation and coordination capabilities, without precluding any of the software engineering goals defined below.

Include legacy and "owned-elsewhere" applications. Whereas *legacy* usually implies reuse of an established system fully controlled by the agent-based system developer, *owned-elsewhere* refers to applications to which the developer has partial access, but no control. Examples of owned-elsewhere applications include data sources and services available on the World Wide Web, via simple form-based interfaces, and applications used cooperatively within a virtual enterprise, which remain the properties of separate corporate entities. Both classes of application must preferably be able to interoperate, more or less as full-fledged members of the agent community, without requiring an overwhelming integration effort.

Human-oriented user interfaces

Systems composed of multiple distributed components, and possibly dynamic configurations of components, require the crafting of intuitive user interfaces to *provide conceptually natural interaction mechanisms, treat users as privileged members of the agent community* and *support collaboration.*

Provide conceptually natural interaction mechanisms with multiple distributed components. When there are numerous disparate agents, and/or complex tasks implemented by the system, the user should be able to express requests without having detailed knowledge of the individual agents. With speech recognition, handwriting recognition, and natural language technologies becoming more mature, agent architectures should preferably support these forms of input playing increased roles in the tasking of agent communities.

Preferably treat *users as privileged members* of the agent community by providing an appropriate level of task specification within *software* agents, and reusable translation mechanisms between this level and the level of *human* requests, supporting constructs that seamlessly incorporate interactions between both human-interface and software types of agents.

Preferably support *collaboration* (simultaneous work over shared data and processing resources) between users and agents.

Realistic software engineering requirements

System-building frameworks should preferably address the practical concerns of real-world applications by the specification of requirements which preferably include: *Minimize the effort* required to create new agents, and to wrap existing applications. *Encourage reuse,* both of domain-independent and domain-specific components. The concept of *agent orientation,* like that of object orientation, provides a natural conceptual framework for reuse, so long as mechanisms for encapsulation and interaction are structured appropriately. *Support lightweight, mobile platforms.* Such platforms should be able to serve as hosts for agents, without requiring the installation of a massive environment. It should also be possible to construct individual agents that are relatively small and modest in their processing requirements. *Minimize platform and language barriers.* Creation of new agents, as well as wrapping of existing applications, should not require the adoption of a new language or environment.

Mechanisms of Cooperation

Cooperation among agents in accordance with the present invention is preferably achieved via messages expressed in a common language, *ICL*. Cooperation among agent is further preferably structured around a three-part approach: providers of services register capabilities specifications with a facilitator, requesters of services construct goals and relay them to a facilitator, and facilitators coordinate the efforts of the appropriate service providers in satisfying these goals.

The Interagent Communication Language (ICL)

Interagent Communication Language (*"ICL"*) 418 refers to an interface, communication, and task coordination language preferably shared by all agents, regardless of what platform they run on or what computer language they are programmed in. *ICL* may be used by an agent to task itself or some subset of the agent community. Preferably, *ICL* allows agents to specify explicit control parameters while simultaneously supporting expression of goals in an underspecified, loosely constrained manner. In a further preferred embodiment, agents employ *ICL* to perform queries, execute actions, exchange information, set triggers, and manipulate data in the agent community.

In a further preferred embodiment, a program element expressed in *ICL* is the *event*. The activities of every agent, as well as communications between agents, are preferably structured around the transmission and handling of events. In communications, events preferably serve as messages between agents; in regulating the activities of individual agents, they may preferably be thought of as goals to be satisfied. Each event preferably has a type, a set of parameters, and content. For example, the agent library procedure *oaa_Solve* can be used by an agent to request services of other agents. A call to *oaa_Solve*, within the code of agent *A*, results in an event having the form

ev_post_solve(Goal, Params)

going from *A* to the facilitator, where *ev_post_solve* is the type, *Goal* is the content, and *Params* is a list of parameters. The allowable content and parameters preferably vary according to the type of the event.

The *ICL* preferably includes a layer of conversational protocol and a content layer. The conversational layer of *ICL* is defined by the event types, together with the parameter lists associated with certain of these event types. The content layer consists of the specific goals, triggers, and data elements that may be embedded within various events.

The *ICL* conversational protocol is preferably specified using an orthogonal, parameterized approach, where the conversational aspects of each element of an interagent conversation are represented by a selection of an event type and a selection of values from at least one orthogonal set of parameters. This approach offers greater expressiveness than an approach based solely on a fixed selection of *speech acts*, such as embodied in KQML. For example, in KQML, a request to satisfy a query can employ either of the performatives *ask_all* or *ask_one*. In *ICL*, on the other hand, this type of request preferably is expressed by the event type *ev_post_solve*, together with the *solution_limit(N)* parameter - where *N* can be any positive integer. (A request for all solutions is indicated by the omission of the *solution_limit* parameter.) The request can also be accompanied by other parameters, which combine to further refine its semantics. In KQML, then, this example forces one to choose between two possible conversational options, neither of which may be precisely what is desired. In either case, the performative chosen is a single value that must capture the entire conversational characterization of the communication. This requirement raises a difficult challenge for the language designer, to select a set of performatives that provides the desired functionality without becoming unmanageably large. Consequently, the debate over the right set of performatives has consumed much discussion within the KQML community.

The content layer of the *ICL* preferably supports unification and other features found in logic programming language environments such as PROLOG. In some embodiments, the content layer of the *ICL* is simply an extension of at least one programming language. For example, the Applicants have found that PROLOG is suitable for implementing and extending into the content layer of the *ICL*. The agent libraries preferably provide support for constructing, parsing, and manipulating *ICL* expressions. It is possible to embed content expressed in other languages within an *ICL* event. However, expressing content in *ICL* simplifies the facilitator's access to the content, as well as the conversational layer, in delegating requests. This gives the facilitator more information about the nature of a request and helps the facilitator decompose compound requests and delegate the sub-requests.

Further, *ICL* expressions preferably include, in addition to events, at least one of the following: capabilities declarations, requests for services, responses to requests, trigger specifications, and shared data elements. A further preferred embodiment of the present invention incorporates *ICL* expressions including at least all of the following: events, capabilities declarations, requests for services, responses to requests, trigger specifications, and shared data elements.

Providing Services: Specifying "Solvables"

In a preferred embodiment of the present invention, every participating agent defines and publishes a set of capability declarations, expressed in *ICL*, describing the services that it provides. These declarations establish a high-level interface to the agent. This interface is used by a facilitator in communicating with the agent, and, most important, in delegating service requests (or parts of requests) to the agent. Partly due to the use of PROLOG as a preferred basis for *ICL*, these capability declarations are referred as *solvables*. The agent library preferably provides a set of procedures allowing an agent to add, remove, and modify its solvables, which it may preferably do at any time after connecting to its facilitator.

There are preferably at least two major types of solvables: *procedure* solvables and *data* solvables. Intuitively, a procedure solvable performs a test or action, whereas a data solvable provides access to a collection of data. For example, in creating an agent for a mail system, procedure solvables might be defined for sending a message to a person, testing whether a message about a particular subject has arrived in the mail queue, or displaying a particular message onscreen. For a database wrapper agent, one might define a distinct data solvable corresponding to each of the relations present in the database. Often, a data solvable is used to provide a *shared* data store, which may be not only queried, but also updated, by various agents having the required permissions.

There are several primary technical differences between these two types of solvables. First, each procedure solvable must have a handler declared and defined for it, whereas this is preferably not necessary for a data solvable. The handling of requests for a data solvable is preferably provided transparently by the agent library. Second, data solvables are preferably associated with a dynamic collection of facts (or clauses), which may be further preferably modified at runtime, both by the agent providing the solvable, and by other agents (provided they have the required permissions). Third, special features, available for use with data solvables, preferably facilitate maintaining the associated facts. In spite of these differences, it should be noted that the mechanism of *use* by which an agent requests a service is the same for the two types of solvables.

In one embodiment, a request for one of an agent's services normally arrives in the form of an event from the agent's facilitator. The appropriate handler then deals with this event. The handler may be coded in whatever fashion is most appropriate, depending on the nature of the task, and the availability of task-specific libraries or legacy code, if any. The only hard requirement is that the handler return an appropriate response to the request, expressed in *ICL*. Depending on the nature of the request, this response could be an indication of success or failure, or a list of solutions (when the request is a data query).

A solvable preferably has three parts: a *goal*, a list of *parameters*, and a list of *permissions*, which are declared using the format:

solvable(Goal, Parameters, Permissions)

The goal of a solvable, which syntactically takes the preferable form of an *ICL* structure, is a logical representation of the service provided by the solvable. (An *ICL* structure consists of a *functor* with 0 or more arguments. For example, in the structure a(b,c), `a' is the functor, and `b' and `c' the arguments.) As with a PROLOG structure, the goal's arguments themselves may preferably be structures.

Various options can be included in the parameter list, to refine the semantics associated with the solvable. The *type* parameter is preferably used to say whether the solvable is *data* or *procedure*. When the type is *procedure*, another parameter may be used to indicate the handler to be associated with the solvable. Some of the parameters appropriate for a *data* solvable are mentioned elsewhere in this application. In either case (procedure or data solvable), the *private* parameter may be preferably used to restrict the use of a solvable to the declaring agent when the agent intends the solvable to be solely for its internal use but wishes to take advantage of the mechanisms in accordance with the present invention to access it, or when the agent wants the solvable to be available to outside agents only at selected times. In support of the latter case, it is preferable for the agent to change the status of a solvable from private to non-private at any time.

The permissions of a solvable provide mechanisms by which an agent may preferably control access to its services allowing the agent to restrict calling and writing of a solvable to itself and/or other selected agents. (*Calling* means requesting the service encapsulated by a solvable, whereas *writing* means modifying the collection of facts associated with a data solvable.) The default permission for every solvable in a further preferred embodiment of the present invention is to be callable by anyone, and for data solvables to be writable by anyone. A solvable's permissions can preferably be changed at any time, by the agent providing the solvable.

For example, the solvables of a simple email agent might include:

```
    solvable(send_message(email, +ToPerson, +Params),
           [type(procedure), callback(send_mail)],
             [])
    solvable(last_message(email, -MessageId),
             [type(data), single_value(true)],
             [write(true)]),
           solvable(get_message(email, +MessageId, -
Msg),
               [type(procedure), callback(get_mail)],
           [])
```

The symbols `+' and `-', indicating input and output arguments, are at present used only for purposes of documentation. Most parameters and permissions have default values, and specifications of default values may be omitted from the parameters and permissions lists.

Defining an agent's capabilities in terms of solvable declarations effectively creates a vocabulary with which other agents can communicate with the new agent. Ensuring that agents will speak the same language and share a common, unambiguous semantics of the vocabulary involves *ontology.* Agent development tools and services (automatic translations of solvables by the facilitator) help address this issue; additionally, a preferred embodiment of the present invention will typically rely on vocabulary from either formally engineered ontologies for specific domains or from ontologies constructed during the incremental development of a body of agents for several applications or from both specific domain ontologies and incrementally developed ontologies. Several example tools and services are described in Cheyer et al.'s paper entitled "Development Tools for the Open Agent Architecture," as presented at the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 96), London, April 1996.

Although the present invention imposes no hard restrictions on the form of solvable declarations, two common usage conventions illustrate some of the utility associated with solvables.

Classes of services are often preferably tagged by a particular type. For instance, in the example above, the "last_message" and "get_message" solvables are specialized for email, not by modifying the *names* of the services, but rather by the use of the `email' parameter, which serves during the execution of an *ICL* request to select (or not) a specific type of message.

Actions are generally written using an imperative verb as the functor of the solvable in a preferred embodiment of the present invention, the direct object (or item class) as the first argument of the predicate, required arguments following, and then an extensible parameter list as the last argument. The parameter list can hold optional information usable by the function. The *ICL* expression generated by a natural language parser often makes use of this parameter list to store prepositional phrases and adjectives.

As an illustration of the above two points, "Send mail to Bob about lunch" will be translated into an *ICL* request send_message(email, `Bob Jones', [subject(lunch)]), whereas "Remind Bob about lunch" would leave the transport unspecified (send_message(KIND, 'Bob Jones', [subject(lunch)])), enabling all available message transfer agents (e.g., fax, phone, mail, pager) to compete for the opportunity to carry out the request.

Requesting Services

An agent preferably requests services of the community of agent by delegating tasks or goals to its facilitator. Each request preferably contains calls to one or more agent solvables, and optionally specifies parameters containing advice to help the facilitator determine how to execute the task. Calling a solvable preferably does *not* require that the agent specify (or even know of) a particular agent or agents to handle the call. While it is possible to specify one or more agents using an address parameter (and there are situations in which this is desirable), in general it is advantageous to leave this delegation to the facilitator. This greatly reduces the hard-coded component dependencies often found in other distributed frameworks. The agent libraries of a preferred embodiment of the present invention provide an agent with a single, unified point of entry for requesting services of other agents: the library procedure *oaa_Solve*. In the style of logic programming, *oaa_Solve* may preferably be used both to retrieve data and to initiate actions, so that calling a *data* solvable looks the same as calling a *procedure* solvable.

Complex Goal Expressions

A powerful feature provided by preferred embodiments of the present invention is the ability of a client agent (or a user) to submit compound goals of an arbitrarily complex nature to a facilitator. A compound goal is a single goal expression that specifies multiple sub-goals to be performed. In speaking of a *"complex goal expression"* we mean that a single goal expression that expresses multiple sub-goals can potentially include more than one type of logical connector (e.g., AND, OR, NOT), and/or more than one level of logical nesting (e.g., use of parentheses), or the substantive equivalent. By way of further clarification, we note that when speaking of an *"arbitrarily complex goal expression"* we mean that goals are expressed in a language or syntax that allows expression of such complex goals when appropriate or when desired, not that every goal is itself necessarily complex.

It is contemplated that this ability is provided through an interagent communication language having the necessary syntax and semantics. In one example, the goals may take the form of compound goal expressions composed using operators similar to those employed by PROLOG, that is, the comma for conjunction, the semicolon for disjunction, the arrow for conditional execution, etc. The present invention also contemplates significant extensions to PROLOG syntax and semantics. For example, one embodiment incorporates a "parallel disjunction" operator indicating that the disjuncts are to be executed by different agents concurrently. A further embodiment supports the specification of whether a given sub-goal is to be executed breadth-first or depth-first.

A further embodiment supports each sub-goal of a compound goal optionally having an address and/or a set of parameters attached to it. Thus, each sub-goal takes the form

Address:Goal::Parameters

where both *Address* and *Parameters* are optional.

An address, if present, preferably specifies one or more agents to handle the given goal, and may employ several different types of referring expression: unique names, symbolic names, and shorthand names. Every agent has preferably a unique name, assigned by its facilitator, which relies upon network addressing schemes to ensure its global uniqueness. Preferably, agents also have self-selected symbolic names (for example, "mail"), which are not guaranteed to be unique. When an address includes a symbolic name, the facilitator preferably takes this to mean that all agents having that name should be called upon. Shorthand names include `self' and `parent' (which refers to the agent's facilitator). The address associated with a goal or sub-goal is preferably always optional. When an address is not present, it is the facilitator's job to supply an appropriate address.

The distributed execution of compound goals becomes particularly powerful when used in conjunction with natural language or speech-enabled interfaces, as the query itself may specify how functionality from distinct agents will be combined. As a simple example, the spoken utterance "Fax it to Bill Smith's manager." can be translated into the following compound *ICL* request:

oaa_Solve((manager('Bill Smith', M), fax(it,M,[])), [strategy(action)])

Note that in this ICL request there are two sub-goals, "manager('Bill Smith',M)" and "fax(it,M,[])," and a single global parameter "strategy(action)." According to the present invention, the facilitator is capable of mapping global parameters in order to apply the constraints or advice across the separate sub-goals in a meaningful way. In this instance, the global parameter strategy(action) implies a parallel constraint upon the first sub-goal; i.e., when there are multiple agents that can respond to the manager sub-goal, each agent should receive a request for service. In contrast, for the second sub-goal, parallelism should not be inferred from the global parameter strategy(action) because such an inference would possibly result in the transmission of duplicate facsimiles.

Refining Service Requests

In a preferred embodiment of the present invention, parameters associated with a goal (or sub-goal) can draw on useful features to refine the request's meaning. For example, it is frequently preferred to be able to specify whether or not solutions are to be returned synchronously; this is done using the *reply* parameter, which can take any of the values *synchronous, asynchronous*, or *none.* As another example, when the goal is a non-compound query of a data solvable, the *cache* parameter may preferably be used to request local caching of the facts associated with that solvable. Many of the remaining parameters fall into two categories: feedback and advice.

Feedback parameters allow a service requester to receive information from the facilitator about how a goal was handled. This feedback can include such things as the identities of the agents involved in satisfying the goal, and the amount of time expended in the satisfaction of the goal.

Advice parameters preferably give constraints or guidance to the facilitator in completing and interpreting the goal. For example, a *solution_limit* parameter preferably allows the requester to say how many solutions it is interested in; the facilitator and/or service providers are free to use this information in optimizing their efforts. Similarly, a *time_limit* is preferably used to say how long the requester is willing to wait for solutions to its request, and, in a multiple facilitator system, a *level_limit* may preferably be used to say how remote the facilitators may be that are consulted in the search for solutions. A *priority* parameter is preferably used to indicate that a request is more urgent than previous requests that have not yet been satisfied. Other preferred advice parameters include but are not limited to parameters used to tell the facilitator whether parallel satisfaction of the parts of a goal is appropriate, how to combine and filter results arriving from multiple solver agents, and whether the requester itself may be considered a candidate solver of the sub-goals of a request.

Advice parameters preferably provide an extensible set of low-level, orthogonal parameters capable of combining with the *ICL* goal language to fully express how information should flow among participants. In certain preferred embodiments of the present invention, multiple parameters can be grouped together and given a group name. The resulting *high-level advice parameters* can preferably be used to express concepts analogous to KQML's performatives, as well as define classifications of problem types. For instance, KQML's "ask_all" and "ask_one" performatives would be represented as combinations of values given to the parameters *reply*, *parallel_ok*, and *solution_limit*. As an example of a higher-level problem type, the strategy "math_problem" might preferably send the query to all appropriate math solvers in parallel, collect their responses, and signal a conflict if different answers are returned. The strategy "essay_question" might preferably send the request to all appropriate participants, and signal a problem (i.e., cheating) if any of the returned answers are identical.

Facilitation

In a preferred embodiment of the present invention, when a facilitator receives a compound goal, its job is to construct a goal satisfaction plan and oversee its satisfaction in an optimal or near optimal manner that is consistent with the specified advice. The facilitator of the present invention maintains a knowledge base that records the capabilities of a collection of agents, and uses that knowledge to assist requesters and providers of services in making contact.

Figure 7 schematically shows data structures 700 internal to a facilitator in accordance with one embodiment of the present invention. Consider the function of a Agent Registry 702 in the present invention. Each registered agent may be seen as associated with a collection of fields found within its parent facilitator such as shown in the figure. Each registered agent may optionally possess a Symbolic Name which would be entered into field 704. As mentioned elsewhere, Symbolic Names need not be unique to each instance of an agent. Note that an agent may in certain preferred embodiments of the present invention possess more than one Symbolic Name. Such Symbolic Names would each be found through their associations in the Agent Registry entries. Each agent, when registered, must possess a Unique Address, which is entered into the Unique Address field 706.

With further reference to Figure 7, each registered agent may be optionally associated with one or more capabilities, which have associated Capability Declaration fields 708 in the parent facilitator Agent Registry 702. These capabilities may define not just functionality, but may further provide a utility parameter indicating, in some manner (e.g., speed, accuracy, etc), how effective the agent is at providing the declared capability. Each registered agent may be optionally associated with one or more data components, which have associated Data Declaration fields 710 in the parent facilitator Agent Registry 702. Each registered agent may be optionally associated with one or more triggers, which preferably could be referenced through their associated Trigger Declaration fields 712 in the parent facilitator Agent Registry 702. Each registered agent may be optionally associated with one or more tasks, which preferably could be referenced through their associated Task Declaration fields 714 in the parent facilitator Agent Registry 702. Each registered agent may be optionally associated with one or more Process Characteristics, which preferably could be referenced through their associated Process Characteristics Declaration fields 716 in the parent facilitator Agent Registry 702. Note that these characteristics in certain preferred embodiments of the present invention may include one or more of the following: Machine Type (specifying what type of computer may run the agent), Language (both computer and human interface).

A facilitator agent in certain preferred embodiments of the present invention further includes a Global Persistent Database 720. The database 720 is composed of data elements which do not rely upon the invocation or instantiation of client agents for those data elements to persist. Examples of data elements which might be present in such a database include but are not limited to the network address of the facilitator agent's server, facilitator agent's server accessible network port list, firewalls, user lists, and security op~~ns regarding the access of server resou~~s accessible to the facilitator agent.

A simplified walk through of operations involved in creating a client agent, a client agent initiating a service request, a client agent responding to a service request and a facilitator agent responding to a service request are including hereafter by way of illustrating the use of such a system. These figures and their accompanying discussion are provided by way of illustration of one preferred embodiment of the present invention and are not intended to limit the scope of the present invention.

Figure 8 depicts operations involved in instantiating a client agent with its parent facilitator in accordance with a preferred embodiment of the present invention. The operations begin with starting the Agent Registration in a step 800. In a next step 802, the Installer, such as a client or facilitator agent, invokes a new client agent. It will be appreciated that any computer entity is capable of invoking a new agent. The system then instantiates the new client agent in a step 804. This operation may involve resource allocations somewhere in the network on a local computer system for the client agent, which will often include memory as well as placement of references to the newly instantiated client agent in internal system lists of agents within that local computing system. Once instantiated, the new client and its parent facilitator establish a communications link in a step 806. In certain preferred embodiments, this communications link involves selection of one or more physical transport mechanisms for this communication. Once established, the client agent transmits it profile to the parent facilitator in a step 808. When received, the parent facilitator registers the client agent in a step 810. Then, at a step 812, a client agent has been instantiated in accordance with one preferred embodiment of the present invention.

Figure 9 depicts operations involved in a client agent initiating a service request and receiving the response to that service request in accordance with a preferred embodiment of the present invention. The method of Figure 9 begins in a step 900, wherein any initialization or other such procedures may be performed. Then, in a step 902, the client agent determines a goal to be achieved (or solved). This goal is then translated in a step 904 into *ICL*, if it is not already formulated in it. The goal, now stated in *ICL*, is then transmitted to the client agent's parent facilitator in a step 906. The parent facilitator responds to this service request and at a later time, the client agent receives the results of the request in a step 908, operations of Figure 9 being complete in a done step 910.

FIGURE 10 depicts operations involved in a client agent responding to a service request in accordance with a preferred embodiment of the present invention. Once started in a step 1000, the client agent receives the service request in a step 1002. In a next step 1004, the client agent parses the received request from ICL. The client agent then determines if the service is available in a step 1006. If it is not, the client agent returns a status report to that effect in a step 1008. If the service is available, control is passed to a step 1010 where the client performs the requested service. Note that in completing step 1010 the client may form complex goal expressions, requesting results for these solvables from the facilitator agent. For example, a fax agent might fax a document to a certain person only after requesting and receiving a fax number for that person. Subsequently, the client agent either returns the results of the service and/or a status report in a step 1012. The operations of Figure 10 are complete in a done step 1014.

FIGURE 11 depicts operations involved in a facilitator agent response to a service request in accordance with a preferred embodiment of the present invention. The start of such operations in step 1100 leads to the reception of a goal request in a step 1102 by the facilitator. This request is then parsed and interpreted by the facilitator in a step 1104. The facilitator then proceeds to construct a goal satisfaction plan in a next step 1106. In steps 1108 and 1110, respectively, the facilitator determines the required sub-goals and then selects agents suitable for performing the required sub-goals. The facilitator then transmits the sub-goal requests to the selected agents in a step 1112 and receives the results of these transmitted requests in a step 1114. It should be noted that the actual implementation of steps 1112 and 1114 are dependent upon the specific goal satisfaction plan. For instance, certain sub-goals may be sent to separate agents in parallel, while transmission of other sub-goals may be postponed until receipt of particular answers. Further, certain requests may generate multiple responses that generate additional sub-goals. Once the responses have been received, the facilitator determines whether the original requested goal has been completed in a step 1118. If the original requested goal has not been completed, the facilitator recursively repeats the operations 1106 through 1116. Once the original requested goal is completed, the facilitator returns the results to the requesting agent 1118 and the operations are done at 1120.

A further preferred embodiment of the present invention incorporates *transparent delegation*, which means that a requesting agent can generate a request, and a facilitator can manage the satisfaction of that request, without the requester needing to have any knowledge of the identities or locations of the satisfying agents. In some cases, such as when the request is a data query, the requesting agent may also be oblivious to the *number* of agents involved in satisfying a request. Transparent delegation is possible because agents' capabilities (solvables) are treated as an abstract description of a service, rather than as an entry point into a library or body of code.

A further preferred embodiment of the present invention incorporates facilitator handling of compound goals, preferably involving three types of processing: delegation, optimization and interpretation.

Delegation processing preferably supports facilitator determination of which specific agents will execute a compound goal and how such a compound goal's sub-goals will be combined and the sub-goal results routed. *Delegation* involves selective application of global and local constraint and advice parameters onto the specific sub-goals. *Delegation* results in a goal that is unambiguous as to its meaning and as to the agents that will participate in satisfying it.

Optimization processing of the completed goal preferably includes the facilitator using sub-goal parallelization where appropriate. *Optimization* results in a goal whose interpretation will require as few exchanges as possible, between the facilitator and the satisfying agents, and can exploit parallel efforts of the satisfying agents, wherever this does not affect the goal's meaning.

Interpretation processing of the optimized goal. Completing the addressing of a goal involves the selection of one or more agents to handle each of its sub-goals (that is, each sub-goal for which this selection has not been specified by the requester). In doing this, the facilitator uses its knowledge of the capabilities of its client agents (and possibly of other facilitators, in a multi-facilitator system). It may also use strategies or advice specified by the requester, as explained below. The *interpretation* of a goal involves the coordination of requests to the satisfying agents, and assembling their responses into a coherent whole, for return to the requester.

A further preferred embodiment of present invention extends facilitation so the facilitator can employ strategies and advice given by the requesting agent, resulting in a variety of interaction patterns that may be instantiated in the satisfaction of a request.

A further preferred embodiment of present invention handles the distribution of both data update requests and requests for installation of triggers, preferably using some of the same strategies that are employed in the delegation of service requests.

Note that the reliance on facilitation is not absolute; that is, there is no hard requirement that requests and services be matched up by the facilitator, or that interagent communications go through the facilitator. There is preferably support in the agent library for explicit addressing of requests. However, a preferred embodiment of the present invention encourages employment the paradigm of agent communities, minimizing their development effort, by taking advantage of the facilitator's provision of transparent delegation and handling of compound goals.

A facilitator is preferably viewed as a *coordinator*, not a controller, of cooperative task completion. A facilitator preferably never initiates an activity. A facilitator preferably responds to requests to manage the satisfaction of some goal, the update of some data repository, or the installation of a trigger by the appropriate agent or agents. All agents can preferably take advantage of the facilitator's expertise in delegation, and its up-to-date knowledge about the current membership of a dynamic community. The facilitator's coordination services often allows the developer to lessen the complexity of individual agents, resulting in a more manageable software development process, and enabling the creation of lightweight agents.

Maintaining Data Repositories

The agent library supports the creation, maintenance, and use of databases, in the form of data solvables. Creation of a data solvable requires only that it be declared. Querying a data solvable, as with access to any solvable, is done using *oaa_Solve.*

A data solvable is conceptually similar to a relation in a relational database. The facts associated with each solvable are maintained by the agent library, which also handles incoming messages containing queries of data solvables. The default behavior of an agent library in managing these facts may preferably be refined, using parameters specified with the solvable's declaration. For example, the parameter *single_value* preferably indicates that the solvable should only contain a single fact at any given point in time. The parameter *unique_values* preferably indicates that no duplicate values should be stored.

Other parameters preferably allow data solvables use of the concepts of ownership and persistence. For implementing shared repositories, it is often preferable to maintain a record of which agent created each fact of a data solvable with the creating agent being preferably considered the fact's owner. In many applications, it is preferable to remove an agent's facts when that agent goes offline (for instance, when the agent is no longer participating in the agent community, whether by deliberate termination or by malfunction). When a data solvable is declared to be non-persistent, its facts are automatically maintained in this way, whereas a persistent data solvable preferably retains its facts until they are explicitly removed.

A further preferred embodiment of present invention supports an agent library through procedures by which agents can update (add, remove, and replace) facts belonging to data solvables, either locally or on other agents, given that they have preferably the required permissions. These procedures may preferably be refined using many of the same parameters that apply to service requests. For example, the *address* parameter preferably specifies one or more particular agents to which the update request applies. In its absence, just as with service requests, the update request preferably goes to *all* agents providing the relevant data solvable. This default behavior can be used to maintain coordinated "mirror" copies of a data set within multiple agents, and can be useful in support of distributed, collaborative activities.

Similarly, the *feedback* parameters, described in connection with *oaa_Solve*, are preferably available for use with data maintenance requests.

A further preferred embodiment of present invention supports ability to provide data solvables not just to client agents, but also to facilitator agents. Data solvables can preferably created, maintained and used by a facilitator. The facilitator preferably can, at the request of a client of the facilitator, create, maintain and share the use of data solvables with all the facilitator's clients. This can be useful with relatively stable collections of agents, where the facilitator's workload is predictable.

Using a Blackboard Style of Communication

In a further preferred embodiment of present invention, when a data solvable is publicly readable and writable, it acts essentially as a global data repository and can be used cooperatively by a group of agents. In combination with the use of triggers, this allows the agents to organize their efforts around a "blackboard" style of communication.

As an example, the "DCG-NL" agent (one of several existing natural language processing agents), provides natural language processing services for a variety of its peer agents, expects those other agents to record, on the facilitator, the vocabulary to which they are prepared to respond, with an indication of each word's part of speech, and of the logical form (*ICL* sub-goal) that should result from the use of that word. In a further preferred embodiment of present invention, the NL agent, preferably when it comes online, preferably installs a data solvable for each basic part of speech on its facilitator. For instance, one such solvable would be:

```
solvable(noun(Meaning, Syntax), [], [])
```

Note that the empty lists for the solvable's permissions and parameters are acceptable here, since the default permissions and parameters provide appropriate functionality.

A further preferred embodiment of present invention incorporating an Office Assistant system as discussed herein or similar to the discussion here supports several agents making use of these or similar services. For instance, the database agent uses the following call, to library procedure *oaa_AddData*, to post the noun `boss', and to indicate that the "meaning" of boss is the concept `manager':

```
oaa_AddData(noun(manager, atom(boss)), [address(parent)])
```

Autonomous Monitoring with Triggers

A further preferred embodiment of present invention includes support for triggers, providing a general mechanism for requesting some action be taken when a set of conditions is met. Each agent can preferably install triggers either locally, for itself, or remotely, on its facilitator or peer agents. There are preferably at least four types of triggers: communication, data, task, and time. In addition to a type, each trigger preferably specifies at least a condition and an action, both preferably expressed in *ICL*. The condition indicates under what circumstances the trigger should fire, and the action indicates what should happen when it fires. In addition, each trigger can be set to fire either an unlimited number of times, or a specified number of times, which can be any positive integer.

Triggers can be used in a variety of ways within preferred embodiments of the present invention. For example, triggers can be used for monitoring external sensors in the execution environment, tracking the progress of complex tasks, or coordinating communications between agents that are essential for the synchronization of related tasks. The installation of a trigger within an agent can be thought of as a representation of that agent's *commitment* to carry out the specified action, whenever the specified condition holds true.

Communication triggers preferably allow any incoming or outgoing event (message) to be monitored. For instance, a simple communication trigger may say something like: "Whenever a solution to a goal is returned from the facilitator, send the result to the presentation manager to be displayed to the user."

Data triggers preferably monitor the state of a data repository (which can be maintained on a facilitator or a client agent). Data triggers' conditions may be tested upon the addition, removal, or replacement of a fact belonging to a data solvable. An example data trigger is: "When 15 users are simultaneously logged on to a machine, send an alert message to the system administrator."

Task triggers preferably contain conditions that are tested after the processing of each incoming event and whenever a timeout occurs in the event polling. These conditions may specify any goal executable by the local *ICL* interpreter, and most often are used to test when some solvable becomes satisfiable. Task triggers are useful in checking for task-specific internal conditions. Although in many cases such conditions are captured by solvables, in other cases they may not be. For example, a mail agent might watch for new incoming mail, or an airline database agent may monitor which flights will arrive later than scheduled. An example task trigger is: "When mail arrives for me about security, notify me immediately."

Time triggers preferably monitor time conditions. For instance, an alarm trigger can be set to fire at a single fixed point in time (e.g., "On December 23rd at 3pm"), or on a recurring basis (e.g., "Every three minutes from now until noon").

Triggers are preferably implemented as data solvables, declared implicitly for every agent. When requesting that a trigger be installed, an agent may use many of the same parameters that apply to service and data maintenance requests.

A further preferred embodiment of present invention incorporates semantic support, in contrast with most programming methodologies, of the agent on which the trigger is installed only having to know how to evaluate the conditional part of the trigger, not the consequence. When the trigger fires, the action is delegated to the facilitator for execution. Whereas many commercial mail programs allow rules of the form "When mail arrives about XXX, [forward it, delete it, archive it]", the possible actions are hard-coded and the user must select from a fixed set.

A further preferred embodiment of present invention, the consequence of a trigger may be any compound goal executable by the dynamic community of agents. Since new agents preferably define both functionality and vocabulary, when an unanticipated agent (for example, a fax agent) joins the community, no modifications to existing code is required for a user to make use of it - "When mail arrives, fax it to Bill Smith."

The Agent Library

In a preferred embodiment of present invention, the agent library provides the infrastructure for constructing an agent-based system. The essential elements of protocol (involving the details of the messages that encapsulate a service request and its response) are preferably made transparent to simplify the programming applications. This enables the developer to focus functionality, rather than message construction details and communication details. For example, to request a service of another agent, an agent preferably calls the library procedure *oaa_Solve*. This call results in a message to a facilitator, which will exchange messages with one or more service providers, and then send a message containing the desired results to the requesting agent. These results are returned via one of the arguments of *oaa_Solve*. None of the messages involved in this scenario is explicitly constructed by the agent developer. Note that this describes the *synchronous* use of *oaa_Solve*.

In another preferred embodiment of present invention, an agent library provides both *intra*agent and *inter*agent infrastructure; that is, mechanisms supporting the internal structure of individual agents, on the one hand, and mechanisms of cooperative interoperation between agents, on the other. Note that most of the infrastructure cuts across this boundary with many of the same mechanisms supporting both agent internals and agent interactions in an integrated fashion. For example, services provided by an agent preferably can be accessed by that agent through the same procedure (*oaa_Solve*) that it would employ to request a service of another agent (the only difference being in the *address* parameter accompanying the request). This helps the developer to reuse code and avoid redundant entry points into the same functionality.

Both of the preferred characteristics described above (transparent construction of messages and integration of *intra*agent with *inter*agent mechanisms) apply to most other library functionality as well, including but not limited to data management and temporal control mechanisms.

Source Code Appendix

Source code for version 2.0 of the*OAA* software product is included as an appendix hereto, and is incorporated herein by reference. The code includes an agent library, which provides infrastructure for constructing an agent-based system. The library's several families of procedures provide the functionalities discussed above, as well as others that have not been discussed here but that will be sufficiently clear to the interested practitioner. For example, declarations of an agent's solvables, and their registration with a facilitator, are managed using procedures such as *oaa_Declare*, *oaa_Undeclare*, and *oaa_Redeclare*. Updates to data solvables can be accomplished with a family of procedures including *oaa_AddData*, *oaa_RemoveData*, and *oaa_ReplaceData*. Similarly, triggers are maintained using procedures such as *oaa_AddTrigger*, *oaa_RemoveTrigger*, and *oaa_ReplaceTrigger*. The provided source code also includes source code for an OAA Facilitator Agent.

The source code appendix is offered solely as a means of further helping practitioners to construct a preferred embodiment of the invention. By no means is the source code intended to limit the scope of the present invention.

Illustrative Applications

To further illustrate the technology of the preferred embodiment, we will next present and discuss two sample applications of the present inventions.

Unified Messaging

A further preferred embodiment of present invention incorporates a Unified Messaging application extending the Automated Office application presented previously herein with an emphasis on ubiquitous access and dynamic presentation of the information and services supported by the agent community. The agents used in this application are depicted in Figure 12.

A hypothetical example of realistic dialog using a preferred embodiment of the present invention can provide insight into how systems may preferably be built using the present invention. In this scenario, the user, with only a telephone as an interface, is planning a trip to Boston where he will soon give a presentation. Capitalized sentences are phrases spoken by the user into the telephone and processed by a phone agent 452.

Responses, unless otherwise indicated, are spoken by the system using text-to-speech generation agent 454.

```
1.1  Welcome to SRI International.  Please enter your
user ID and password.
     <User enters touchtone ID and password>

          Good to see you again Adam Cheyer.  I am
listening to you.
```

Every user interface agent 408, including the telephone agent 452, should know the identify of its user. This information is used in resolving anaphoric references such as "me" and "I", and allows multiple user interfaces operated by the same user to work together.

```
1.2  WHAT IS TODAY'S SCHEDULE?
     Here is today's schedule for Adam Cheyer:
       At 10am for 1 hour, meeting with Dave.
       At 3pm for 1 hour, presentation about software
agents.

          End of schedule.
```

If the user is operating both a graphical user interface and a telephone, as described in conjunction with the Automated Office application, the result of this spoken request is to display a calendar window containing the current schedule. In this case, with no graphical display available, the GEN_NL agent 1202 is tasked to produce a spoken response that can be played over the phone. GEN_NL shares the same dynamic vocabulary and phrasal rules as the natural language parser DCG_NL 426, and contains strategies for producing responses to queries using either simple or list-based multimedia utterances.

```
1.3  FIND FRIDAY'S WEATHER IN BOSTON.
     The weather in Boston for Friday is as follows:
       Sunny in the morning. Partly cloudy in the
afternoon with a 20
       percent chance of thunderstorms late. Highs in the
mid 70s.
```

In addition to data accessible from legacy applications, content may be retrieved by web-reading agents which provide wrappers around useful websites.

```
1.4  FIND ALL NEW MAIL MESSAGES.
     There are 2 messages available.
     Message 1, from Mark Tierny, entitled "OAA meeting."
1.5  NEXT MESSAGE
     Message 2, from Jennifer Schwefler, entitled
"Presentation Summary."
1.6  PLAY IT.
     This message is a multipart MIME-encoded message.
There are two parts.
     Part 1.  (Voicemail message, not text-to speech):
     Thanks for taking part as a speaker in our
conference.
     The schedule will be posted soon on our homepage.
1.7  NEXT PART
     Part 2. (read using text-to-speech):
     The presentation home page is http://www....
1.8  PRINT MESSAGE
     Command executed.
```

Mail messages are no longer just simple text documents, but often consist of multiple subparts containing audio files, pictures, webpages, attachments and so forth. When a user asks to play a complex email message over the telephone, many different agents may be implicated in the translation process, which would be quite different given the request "print it." The challenge is to develop a system which will enable agents to cooperate in an extensible, flexible manner that alleviates explicit coding of agent interactions for every possible input/output combination.

In a preferred embodiment of the present invention, each agent concentrates only on what it can do and on what it knows, and leaves other work to be delegated to the agent community. For instance, a printer agent 1204, defining the solvable print(Object,Parameters), can be defined by the following pseudo-code, which basically says, "If someone can get me a document, in either POSTSCRIPT or text form, I can print it.".

```
print(Object, Parameters) {
   ' If Object is reference to "it", find an appropriate
document
   if (Object = "ref(it)")
      oaa_Solve(resolve_reference(the, document, Params,
Object),[]);
   ' Given a reference to some document, ask for the
document in POSTSCRIPT
   if (Object = "id(Pointer)")
      oaa_Solve(resolve_id_as(id(Pointer), postscript,
[], Object),[]);
   ' If Object is of type text or POSTSCRIPT, we can
print it.
   if ((Object is of type Text) or (Object is of type
Postscript))
      do_print(Object);
}
```

In the above example, since an email message is the salient document, the mail agent 442 will receive a request to produce the message as POSTSCRIPT. Whereas the mail agent 442 may know how to save a text message as POSTSCRIPT, it will not know what to do with a webpage or voicemail message. For these parts of the message, it will simply send oaa_Solve requests to see if another agent knows how to accomplish the task.

Until now, the user has been using only a telephone as user interface. Now, he moves to his desktop, starts a web browser 436, and accesses the URL referenced by the mail message.

```
1.9  RECORD MESSAGE
     Recording voice message.  Start speaking now.
1.10 THIS IS THE UPDATED WEB PAGE CONTAINING THE
PRESENTATION SCHEDULE.
     Message one recorded.
1.11 IF THIS WEB PAGE CHANGES, GET IT TO ME WITH NOTE
ONE.
     Trigger added as requested.
```

In this example, a local agent 436 which interfaces with the web browser can return the current page as a solution to the request "oaa_Solve(resolve_reference(this, web_page, [], Ref),[])", sent by the NL agent 426. A trigger is installed on a web agent 436 to monitor changes to the page, and when the page is updated, the notify agent 446 can find the user and transmit the webpage and voicemail message using the most appropriate media transfer mechanism.

This example based on the Unified Messaging application is intended to show how concepts in accordance with the present invention can be used to produce a simple yet extensible solution to a multi-agent problem that would be difficult to implement using a more rigid framework. The application supports adaptable presentation for queries across dynamically changing, complex information; shared context and reference resolution among applications; and flexible translation of multimedia data. In the next section, we will present an application which highlights the use of parallel competition and cooperation among agents during multi-modal fusion.

Multimodal Map

A further preferred embodiment of present invention incorporates the Multimodal Map application. This application demonstrates natural ways of communicating with a community of agents, providing an interactive interface on which the user may draw, write or speak. In a travel-planning domain illustrated by Figure 13, available information includes hotel, restaurant, and tourist-site data retrieved by distributed software agents from commercial Internet sites. Some preferred types of user interactions and multimodal issues handled by the application are illustrated by a brief scenario featuring working examples taken from the current system.

Sara is planning a business trip to San Francisco, but would like to schedule some activities for the weekend while she is there. She turns on her laptop PC, executes a map application, and selects San Francisco.

2.1 [Speaking] Where is downtown?
Map scrolls to appropriate area.
2.2 [Speaking and drawing region] Show me all hotels near here.
Icons representing hotels appear.
2.3 [Writes on a hotel] Info?
A textual description (price, attributes, etc.) appears.
2.4 [Speaking] I only want hotels with a pool.
Some hotels disappear.
2.5 [Draws a crossout on a hotel that is too close to a highway]
Hotel disappears
2.6 [Speaking and circling] Show me a photo of this hotel.
Photo appears.
2.7 [Points to another hotel]
Photo appears.
2.8 [Speaking] Price of the other hotel?
Price appears for previous hotel.
2.9 [Speaking and drawing an arrow] Scroll down.
Display adjusted.
2.10 [Speaking and drawing an arrow toward a hotel]
What is the distance from this hotel to Fisherman's Wharf?
Distance displayed.
2.11 [Pointing to another place and speaking] And the distance to here?
Distance displayed.

Sara decides she could use some human advice. She picks up the phone, calls Bob, her travel agent, and writes Start collaboration to synchronize his display with hers. At this point, both are presented with identical maps, and the input and actions of one will be remotely seen by the other.

3.1 [Sara speaks and circles two hotels]
Bob, I'm trying to choose between these two hotels. Any opinions?
3.2 [Bob draws an arrow, speaks, and points]
Well, this area is really nice to visit. You can walk there from

this hotel.

Map scrolls to indicated area. Hotel selected.

3.3 [Sara speaks] Do you think I should visit Alcatraz?

3.4 [Bob speaks] Map, show video of Alcatraz.

Video appears.

3.5 [Bob speaks] Yes, Alcatraz is a lot of fun.

A further preferred embodiment of present invention generates the most appropriate interpretation for the incoming streams of multimodal input. Besides providing a user interface *to* a dynamic set of distributed agents, the application is preferably built *using* an agent framework. The present invention also contemplates aiding the coordinate competition and cooperation among information sources, which in turn works in parallel to resolve the ambiguities arising at every level of the interpretation process: *low-level processing of the data stream*, *anaphora resolution*, *cross-modality influences* and *addressee.*

Low-level processing of the data stream: Pen input may be preferably interpreted as a gesture (e.g., 2.5: cross-out) by one algorithm, or as handwriting by a separate recognition process (e.g., 2.3: "info?"). Multiple hypotheses may preferably be returned by a modality recognition component.

Anaphora resolution: When resolving anaphoric references, separate information sources may contribute to resolving the reference: context by object type, deictic, visual context, database queries, discourse analysis. An example of information provided through context by object type is found in interpreting an utterance such as "show photo of the hotel", where the natural language component can return a list of the last hotels talked about. Deictic information in combination with a spoken utterance like "show photo of this hotel" may preferably include pointing, circling, or arrow gestures which might indicate the desired object (e.g., 2.7). Deictic references may preferably occur before, during, or after an accompanying verbal command. Information provided in a visual context, given for the request "display photo of the hotel" may preferably include the user interface agent might determine that only one hotel is currently visible on the map, and therefore this might be the desired reference object. Database queries preferably involving information from a database agent combined with results from other resolution strategies. Examples are "show me a photo of the hotel in Menlo Park" and

2.2. Discourse analysis preferably provides a source of information for phrases such as "No, the other one" (or 2.8).

The above list of preferred anaphora resolution mechanisms is not exhaustive. Examples of other preferred resolution methods include but are not limited to spatial reasoning ("the hotel between Fisherman's Wharf and Lombard Street") and user preferences ("near my favorite restaurant").

Cross-modality influences: When multiple modalities are used together, one modality may preferably reinforce or remove or diminish ambiguity from the interpretation of another. For instance, the interpretation of an arrow gesture may vary when accompanied by different verbal commands (e.g., "scroll left" vs. "show info about this hotel"). In the latter example, the system must take into account how accurately and unambiguously an arrow selects a single hotel.

Addressee: With the addition of collaboration technology, humans and automated agents all share the same workspace. A pen doodle or a spoken utterance may be meant for either another human, the system (3.1), or both (3.2).

The implementation of the Multimodal Map application illustrates and exploits several preferred features of the present invention: reference resolution and task delegation by parallel parameters of oaa_Solve, basic multi-user collaboration handled through built-in data management services, additional functionality readily achieved by adding new agents to the community, domain-specific code cleanly separated from other agents.

A further preferred embodiment of present invention provides reference resolution and task delegation handled in a distributed fashion by the parallel parameters of oaa_Solve, with meta-agents encoding rules to help the facilitator make context- or user-specific decisions about priorities among knowledge sources.

A further preferred embodiment of present invention provides basic multi-user collaboration handled through at least one built-in data management service. The map user interface preferably publishes data solvables for elements such as icons, screen position, and viewers, and preferably defines these elements to have the attribute "shareable". For every update to this public data, the changes are preferably automatically replicated to all members of the collaborative session, with associated callbacks producing the visible effect of the data change (e.g., adding or removing an icon).

Functionality for recording and playback of a session is preferably implemented by adding agents as members of the collaborative community. These agents either record the data changes to disk, or read a log file and replicate the changes in the shared environment.

The domain-specific code for interpreting travel planning dialog is preferably separated from the speech, natural language, pen recognition, database and map user interface agents. These components were preferably reused without modification to add multimodal map capabilities to other applications for activities such as crisis management, multi-robot control, and the MVIEWS tools for the video analyst.

Improved Scalability and Fault Tolerance

Implementations of a preferred embodiment of present invention which rely upon simple, single facilitator architectures may face certain limitations with respect to scalability, because the single facilitator may become a communications bottleneck and may also represent a single, critical point for system failure.

Multiple facilitator systems as disclosed in the preferred embodiments to this point can be used to construct peer-to-peer agent networks as illustrated in Figure 14. While such embodiments are scalable, they do possess the potential for communication bottlenecks as discussed in the previous paragraph and they further possess the potential for reliability problems as central, critical points of vulnerability to systems failure.

A further embodiment of present invention supports a facilitator implemented as an agent like any other, whereby multiple facilitator network topologies can be readily constructed. One example configuration (but not the only possibility) is a hierarchical topology as depicted in Figure 15, where a top level Facilitator manages collections of both client agents 1508 and other Facilitators, 1504 and 1506. Facilitator agents could be installed for individual users, for a group of users, or as appropriate for the task.

Note further, that network work topologies of facilitators can be seen as graphs where each node corresponds to an instance of a facilitator and each edge connecting two or more nodes corresponds to a transmission path across one or more physical transport mechanisms. Some nodes may represent facilitators and some nodes may represent clients. Each node can be further annotated with attributes corresponding to include triggers, data, capabilities but not limited to these attributes.

A further embodiment of present invention provides enhanced scalability and robustness by separating the planning and execution components of the facilitator. In contrast with the centralized facilitation schemes described above, the facilitator system 1600 of Figure 16 separates the registry/planning component from the execution component. As a result, no single facilitator agent must carry all communications nor does the failure of a single facilitator agent shut down the entire system.

Turning directly to Figure 16, the facilitator system 1600 includes a registry/planner 1602 and a plurality of client agents 1612-1616. The registry/planner 1604 is typically replicated in one or more locations accessible by the client agents. Thus if the registry/planner 1604 becomes unavailable, the client agents can access the replicated registry/planner(s).

This system operates, for example, as follows. An agent transmits a goal 1610 to the registry planner 1602. The registry/planner 1604 translates the goal into an unambiguous execution plan detailing how to accomplish any sub-goals developed from the compound goal, as well as specifying the agents selected for performing the sub-goals. This execution plan is provided to the requesting agent which in turn initiates peer-to-peer interactions 1618 in order to implement the detailed execution plan, routing and combining information as specified within the execution plan. Communication is distributed thus decreasing sensitivity of the system to bandwidth limitations of a single facilitator agent. Execution state is likewise distributed thus enabling system operation even when a facilitator agent fails.

Further embodiments of present invention incorporate into the facilitator functionality such as load-balancing, resource management, and dynamic configuration of agent locations and numbers, using (for example) any of the topologies discussed. Other embodiments incorporate into a facilitator the ability to aid agents in establishing peer-to-peer communications. That is, for tasks requiring a sequence of exchanges between two agents, the facilitator assists the agents in finding one another and establishing communication, stepping out of the way while the agents communicate peer-to-peer over a direct, perhaps dedicated channel.

Further preferred embodiments of the present invention incorporate mechanisms for basic transaction management, such as periodically saving the state of agents (both facilitator and client) and rolling back to the latest saved state in the event of the failure of an agent.

APPENDIX A.I

Source code file named compound.pl.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   File    : compound.pl
%   Primary Authors  : David Martin, Adam Cheyer
%   Purpose : Provides handling of compound goals by the facilitator.
%
%   -------------------------------------------------------------------------
%   Unpublished-rights reserved under the copyright laws of the United States.
%
%
%   Unpublished Copyright (c) 1998, SRI International.
%   "Open Agent Architecture" and "OAA" are Trademarks of SRI International.
%   -------------------------------------------------------------------------

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This is just here so this file can be compiled separately (but its
% official declaration is in oaa.pl):
:- op(599,yfx,::).

:- dynamic
    binding_num/1,
    ks_num/1,
    multiple_continuation/7
.

% This file is loaded by facilitator code, and thus no
% module imports are needed here.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% OVERVIEW
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

/*\

These facilitator routines support the use of compound "ICL goals".
An ICLGoal is of the form Sources:Goal::Params, where both Sources and
Params are optional.  Each subgoal of ICLGoal is also of that form.

When an agent calls solve/2, it may specify an ICL goal which is
"incomplete"; that is, ambiguous as to which agents are to solve the
various subgoals.  The facilitator then completes the ICL goal, if
necessary, and executes it.  Execution involves having all the
subgoals solved by the appropriate agents, assembling the solutions,
and returning them to the requesting agent.

If a agent wants to construct a complete ICL goal, and is willing to
guarantee that it's complete and that all solvers mentioned in it are
currently valid, then that agent (usually a "meta-agent") may call
execute_goal directly.  @@ We haven't yet provided library calls for
this.

IMPORTANT NOTE: : has higher precedence than ::.  This means that
a:b::c will unify with X:Y and X:Y::Z, but NOT with Y::Z.

Wherever a Sources field appears, it may be any of the following:
    built_in
    facilitator
    parent
    KS
    [KS1, KS2, ...]

'built_in' isn't normally specified by a requesting agent - although
there's no harm in doing so - but is used internally by the
facilitator.  KS, KS1, KS2, etc. may be either the name or address of
an agent (client or facilitator).  'facilitator' or 'parent' may also
appear in a list of KS's.  If Sources is an empty list or a var, it is
handled just as if there were no Sources field, in which case the
facilitator determines what sources are relevant.

Note that when an ICL goal includes a Sources field, there should not be
Sources fields for any of its subgoals.  If there are, they will be
ignored.  (@@Need to make sure this works ok.)  However, Params fields
may be usefully nested within goals that have Params fields.  Certain
nested parameters, such as solution_limit/1, can be used by the
solving agent.

If an ICL goal has parameters, some of them are "inherited" by
subgoals.  If there's a conflicting parameter on a subgoal, however,
it overrides an inherited parameter.

PARAMETERS
----------

address(+A) [embedded or global] - Used precisely as if A: prefixes
the relevant goal.

get_address(-S) [embedded] - bind S to indicate who provided the
solution.  Solver identities will be given as numeric ids.  Currently
only works when attached to non-compound (sub)goals.

get_address(-S) [global] - bind S to indicate all sources that were queried
in finding solutions (even if they returned none).

\*/

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% GOAL COMPLETION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

/*\

complete_goal(RequestingKS, Goal, GlobalParams, CompletedGoal).

complete_goal takes in an ICL goal and produces a "complete ICL goal"
(sometimes known as a "plan", but I think we'll reserve that term for
future developments).  The goal and the complete goal have precisely
the same variables - but are not necessarily unifiable.

\*/


complete_goal(RequestingKS, Goal, GlobalParams, CompletedGoal) :-
    complete_addressing(RequestingKS, Goal, GlobalParams, AddressedGoal),
    complete_concurrency(AddressedGoal, CompletedGoal).
```

```
/*\

complete_addressing(+RequestingKS, +ICLGoal, +GlobalParams, -AddressedGoal).

AddressedGoal has more-or-less the same form as ICLGoal, but possibly
with some regrouping of subgoals, and the addition of Sources fields
to ICLGoal or its subgoals.  The idea is that AddressedGoal contains
complete information as to where its various subgoals are to be sent,
so that no further analysis is needed.  Any regrouping of subgoals is
done as an optimization.  AddressedGoal shares all variables with
ICLGoal.

@@What other operators (e.g., negation) might we want to support?

\*/

complete_addressing(RequestingKS, ICLGoal, GlobalParams, AddressedGoal) :-
    % @@ verify_params(GlobalParams, global, Verified),
    complete_sources(RequestingKS, ICLGoal, GlobalParams,
                 AddressedGoalWithParamsEverywhere),
    % @@Here, propagate params, instantiate address request in GlobalParams. ?
    remove_empty_params(AddressedGoalWithParamsEverywhere, AddressedGoal).

/*\

complete_sources(+RequestingKS, +ICLGoal, +GlobalParams, -AddressedGoal).

Ensures that every subgoal is explicitly covered by one or more
sources.  Determines the largest subgoals that can be "chunked"; that
is, grouped together for submission to a source.

In the process, every goal acquires a Params field (wherever there was
no Params field before, the empty list is added).  This is done just
to make the definition of complete_sources more readable.

\*/

    % Here we assume that the goal-writer didn't really mean to put a var,
    % because it's not meaningful to do so:
complete_sources(KS, Sources:Goal, GlobalParams, AddressedGoal) :-
    var(Sources),
    !,
    complete_sources(KS, Goal, GlobalParams, AddressedGoal).

/*
    ( AddressedGoal = A:_ ->
      Sources = A
    | otherwise ->
      findall(A, sub_term(A:_, AddressedGoal), SubSources),
      % @@More work needed here:
      Sources = SubSources
    ).
*/

    % Here we assume that the goal-writer didn't really mean to put [],
    % because it's not meaningful to do so:
```

```
complete_sources(KS, []:Goal, GlobalParams, AddressedGoal) :-
    !,
    complete_sources(KS, Goal, GlobalParams, AddressedGoal).

    % Sources and Params already specified; we're done:
    % @@But let's verify the sources are valid!
complete_sources(_KS, Sources:Goal::Params, _GlobalParams,
             Sources:Goal::Params) :-
    !.

    % Sources already specified; add empty Params list:
complete_sources(_KS, Sources:Goal, _GlobalParams, Sources:Goal::[]) :-
    !.

    % Sure, we'll continue to support an address in Params or GlobalParams:
complete_sources(KS, Goal::Params, GlobalParams, AddressedGoal) :-
    % @@ verify_params(...),
    ( memberchk(address(Sources), Params) ;
      memberchk(address(Sources), GlobalParams) ),
    \+ var(Sources),
    !,
    complete_sources(KS, Sources:Goal::Params, GlobalParams, AddressedGoal).

    % No Sources or Params specified; add empty Params list before
    % proceeding:
complete_sources(KS, Goal, GlobalParams, AddressedGoal) :-
    \+ (Goal = _::_),
    !,
    complete_sources(KS, Goal::[], GlobalParams, AddressedGoal).

    % Here we get down to the real work: determining solvers and
    % chunking of subgoals:

complete_sources(KS, (\+ Goal1)::Params, GlobalParams, AddressedGoal) :-
    !,
    oaa_Name(Facilitator),
    complete_sources(KS, Goal1, GlobalParams, AddressedGoal1),
      % If S1 is a SINGLE source, it's OK to send the negation to the source.
      % This case also works if S1 == built_in.
    ( (AddressedGoal1 = [S1]:G1::P1,
       S1 \== Facilitator,
       S1 \== facilitator) ->
        AddressedGoal = S1:((\+ G1)::P1)::Params
    | otherwise ->
        AddressedGoal = (\+ AddressedGoal1::Params)
    ).

complete_sources(KS, (Goal1, Goal2, Goal3)::Params, GlobalParams,
             AddressedGoal) :-
    % This clause is needed because we want built_in pred's to be grouped
    % with what comes before, not after.
    !,
    complete_sources(KS, Goal1, GlobalParams, AddressedGoal1),
    complete_sources(KS, Goal2, GlobalParams, AddressedGoal2),
    complete_sources(KS, Goal3, GlobalParams, AddressedGoal3),
    ( (AddressedGoal1 = S1:G1::P1,
       AddressedGoal2 = S2:G2::P2,
       AddressedGoal3 = S3:G3::P3,
       chunkable_sources([S1, S2, S3], Sources),
       compatible_params([P1, P2, P3])) ->
       AddressedGoal = Sources:(G1::P1, G2::P2, G3::P3)::Params
    | (AddressedGoal1 = S1:G1::P1,
       AddressedGoal2 = S2:G2::P2,
       AddressedGoal3 = (S3A:G3A::P3A, Goal3B)::P3,
       % Goal3B may or may not begin with Source:.  icl_GoalComponents
       % deals with the precedence issues.
       icl_GoalComponents(Goal3B, _, G3B, P3B),
       chunkable_sources([S1, S2, S3A], Sources),
       append(P3A, P3, NewP3A),
       append(P3B, P3, NewP3B),
       compatible_params([P1, P2, NewP3A])) ->
       AddressedGoal = (Sources:(G1::P1, G2::P2, G3A::NewP3A)::[],
                        G3B::NewP3B)::Params
    | (AddressedGoal1 = S1:G1::P1,
       AddressedGoal2 = S2:G2::P2,
       chunkable_sources(S1, S2, Sources),
       compatible_params([P1, P2])) ->
       AddressedGoal = (Sources:(G1::P1, G2::P2)::[], AddressedGoal3)::Params
    | (AddressedGoal2 = S2:G2::P2,
       AddressedGoal3 = S3:G3::P3,
       chunkable_sources(S2, S3, Sources),
       compatible_params([P2, P3])) ->
       AddressedGoal = (AddressedGoal1, Sources:(G2::P2, G3::P3)::[])::Params
    | (AddressedGoal2 = S2:G2::P2,
       AddressedGoal3 = (S3A:G3A::P3A, Goal3B)::P3,
       icl_GoalComponents(Goal3B, _, G3B, P3B),
       chunkable_sources([S2, S3A], Sources),
       append(P3A, P3, NewP3A),
       append(P3B, P3, NewP3B),
       compatible_params([P2, NewP3A])) ->
       AddressedGoal = (AddressedGoal1, Sources:(G2::P2, G3A::NewP3A)::[],
                        G3B:NewP3B)::Params
    | otherwise ->
       AddressedGoal =
           (AddressedGoal1, AddressedGoal2, AddressedGoal3)::Params
    ).
complete_sources(KS, (Goal1, Goal2)::Params, GlobalParams, AddressedGoal) :-
    !,
    complete_sources(KS, Goal1, GlobalParams, AddressedGoal1),
    complete_sources(KS, Goal2, GlobalParams, AddressedGoal2),
    ( (AddressedGoal1 = S1:G1::P1,
       AddressedGoal2 = S2:G2::P2,
       chunkable_sources(S1, S2, Sources),
       compatible_params([P1, P2])) ->
       AddressedGoal = Sources:(G1::P1, G2::P2)::Params
    | otherwise ->
       AddressedGoal = (AddressedGoal1, AddressedGoal2)::Params
    ).
    % Note: this clause must precede that for disjunction.
complete_sources(KS, (Goal1 -> Goal2 ; Goal3)::Params, GlobalParams,
             AddressedGoal) :-
    !,
    complete_sources(KS, Goal1, GlobalParams, AddressedGoal1),
    complete_sources(KS, Goal2, GlobalParams, AddressedGoal2),
    complete_sources(KS, Goal3, GlobalParams, AddressedGoal3),
    ( (AddressedGoal1 = S1:G1::P1,
       AddressedGoal2 = S2:G2::P2,
       AddressedGoal3 = S3:G3::P3,
       chunkable_sources([S1, S2, S3], Sources),
       compatible_params([P1, P2, P3])) ->
       AddressedGoal = Sources:(G1::P1 -> G2::P2 | G3::P3)::Params
    | otherwise ->
       AddressedGoal =
         (AddressedGoal1 -> AddressedGoal2 | AddressedGoal3)::Params
    ).
complete_sources(KS, (Goal1 -> Goal2)::Params, GlobalParams, AddressedGoal) :-
    !,
    complete_sources(KS, Goal1, GlobalParams, AddressedGoal1),
    complete_sources(KS, Goal2, GlobalParams, AddressedGoal2),
    ( (AddressedGoal1 = S1:G1::P1,
       AddressedGoal2 = S2:G2::P2,
       chunkable_sources([S1, S2], Sources),
       compatible_params([P1, P2])) ->
       AddressedGoal = Sources:(G1::P1 -> G2::P2)::Params
    | otherwise ->
       AddressedGoal =
         (AddressedGoal1 -> AddressedGoal2)::Params
    ).
complete_sources(KS, (Goal1 ; Goal2)::Params, GlobalParams, AddressedGoal) :-
    !,
    complete_sources(KS, Goal1, GlobalParams, AddressedGoal1),
    complete_sources(KS, Goal2, GlobalParams, AddressedGoal2),
    ( (AddressedGoal1 = S1:G1::P1,
       AddressedGoal2 = S2:G2::P2,
       chunkable_sources(S1, S2, Sources),
       compatible_params([P1, P2])) ->
       AddressedGoal = Sources:(G1::P1; G2::P2)::Params
    | otherwise ->
       AddressedGoal = (AddressedGoal1; AddressedGoal2)::Params
    ).
    % To be complete, we will allow for this nonstandard goal form:
complete_sources(KS, Goal::Params1::Params2, GlobalParams,
                 AddressedGoal::Params2) :-
    !,
    complete_sources(KS, Goal::Params1, GlobalParams, AddressedGoal).
complete_sources(_KS, Goal::Params, _GlobalParams, built_in:Goal::Params) :-
    icl_BuiltIn(Goal),
    !.
    % Here, finally, we determine the agents (or parent facilitator) that
    % can solve a non-compound Goal:
complete_sources(KS, Goal, GlobalParams, Sources:Goal) :-
    sources_for_goal(KS, Goal, GlobalParams, Sources).

remove_empty_params(Addr:Goal::[], Addr:NewGoal) :-
    !,
    remove_empty_params(Goal, NewGoal).
remove_empty_params(Addr:Goal::Params, Addr:NewGoal::Params) :-
    !,
    remove_empty_params(Goal, NewGoal).
remove_empty_params(Goal::[], NewGoal) :-
    !,
    remove_empty_params(Goal, NewGoal).
remove_empty_params(Goal::Params, NewGoal::Params) :-
    !,
    remove_empty_params(Goal, NewGoal).
remove_empty_params(Sources:Goal, Sources:NewGoal) :-
    !,
    remove_empty_params(Goal, NewGoal).
remove_empty_params((\+ Goal)::[], (\+ NewGoal)) :-
    !,
    remove_empty_params(Goal, NewGoal).
remove_empty_params((Goal1, Goal2), (NewGoal1, NewGoal2)) :-
    !,
    remove_empty_params(Goal1, NewGoal1),
    remove_empty_params(Goal2, NewGoal2).
remove_empty_params((Goal1 ; Goal2), (NewGoal1 ; NewGoal2)) :-
    !,
    remove_empty_params(Goal1, NewGoal1),
    remove_empty_params(Goal2, NewGoal2).
remove_empty_params((Goal1 -> Goal2), (NewGoal1 -> NewGoal2)) :-
    !,
    remove_empty_params(Goal1, NewGoal1),
    remove_empty_params(Goal2, NewGoal2).
    % Primitive (non-compound) goal:
remove_empty_params(Goal, Goal).

remove_addresses(_Sources:Goal, NewGoal) :-
    !,
    remove_addresses(Goal, NewGoal).
remove_addresses((Goal1, Goal2), (NewGoal1, NewGoal2)) :-
    !,
    remove_addresses(Goal1, NewGoal1),
    remove_addresses(Goal2, NewGoal2).
remove_addresses((Goal1 ; Goal2), (NewGoal1 ; NewGoal2)) :-
    !,
    remove_addresses(Goal1, NewGoal1),
    remove_addresses(Goal2, NewGoal2).
remove_addresses((Goal1 -> Goal2), (NewGoal1 -> NewGoal2)) :-
    !,
    remove_addresses(Goal1, NewGoal1),
    remove_addresses(Goal2, NewGoal2).
    % Primitive (non-compound) goal:
remove_addresses(Goal, Goal).

/*\

chunkable_sources(+Sources1, +Sources2, -Sources).

Each argument is either: a single KS name (or numeric id); a list of
KS names (where 'facilitator' or 'parent' also count as KS
names), or the atom 'built_in'.  (Empty list is OK.)

Sources1 gives the sources that can solve some goal, Sources2
gives the sources that can solve some other goal, and if this
pred. succeeds, Sources gives a set of sources that can solve
both together.

NOTES ON CHUNKING:
```

```
%1 A chunk is a sub-goal SG of a Goal such that
  (1) There is a nonempty set S of client agents each of which can solve
the entire chunk (that is, every predicate in the chunk is either an
icl_BuiltIn or one of the agent's solvables), and
  (2) Performing the subgoal as (ks1:SQ ; ks2:SQ ; ... ; ksN:SQ), where
ks1 ... ksN are all the agents in S, does not in any way violate the
intended semantics of the overall Goal.

NOTE:  chunking is done "conservatively", so as to preserve Prolog
semantics.  So, for example, the following Goal:
    (a(1), b(2)),
where a and b are both solvable by ks1 and ks2, will be chunked as
follows:
    chunk(a(1), [ks1, ks2]), chunk(b(2), [ks1, ks2])
which amounts to no chunking at all, instead of
    chunk((a(1), b(2)), [ks1, ks2]).

The former results in execution
    (ks1:a(1) ; ks2:a2), (ks1:b(2) ; ks2:b(2))
whereas the latter would result in execution
    ks1:(a(1), b(2)) ; ks2:(a(1), b(2))
We might want to explore under what conditions more extensive chunking
can be done.

\*/

    % This just allows for single sources, not in a list:
chunkable_sources(Source1, Source2, Sources) :-
    ( atomic(Source1) ->
      S1 = [Source1]
    | otherwise ->
      S1 = Source1
    ),
    ( atomic(Source2) ->
      S2 = [Source2]
    | otherwise ->
      S2 = Source2
    ),
    chunkable_srcs(S1, S2, Sources).

chunkable_srcs(built_in, Sources, Sources) :-
    % at least one element:
    Sources = [_ | _],
    !.
chunkable_srcs(Sources, built_in, Sources) :-
    Sources = [_ | _],
    !.
chunkable_srcs([], [], []) :-
    !.
chunkable_srcs([Source], [Source], [Source]) :-
    !.
chunkable_srcs([Source1], [Source2], [Source1]) :-
    ( number(Source1), atom(Source2) ;
      number(Source2), atom(Source1) ),
    !,
    find_address(Source1, Source),
    find_address(Source2, Source).
    % chunkable_sources(+SourcesIn, -SourcesOut).
    %   Does the same as chunkable_sources/3, but allows for a list
    % of sources (length >= 1) as arg 1.

chunkable_sources([Sources], Sources).
chunkable_sources([Sources1, Sources2 | RestSources], SourcesOut) :-
    chunkable_sources(Sources1, Sources2, SourcesTemp),
    chunkable_sources([SourcesTemp | RestSources], SourcesOut).

    % compatible_params(+ParamLists).
    %   ParamLists is a list of 2 or more ParamLists.  This predicate
    % succeeds IFF the ParamLists are compatible for purposes of
    % chunking.
compatible_params(_).

    % sources_for_goal(+RequestingKS, +Goal, +Params, -Sources).
    % @@ Here, depending on how the treatment of multiple facilitators evolves,
    % we may need to revisit the default use of the facilitator.

sources_for_goal(RequestingKS, ICLGoal, GlobalParams, Sources) :-
    icl_GoalComponents(ICLGoal, _, Goal, Params),
    append(Params, GlobalParams, AllParams),
    findall(SomeKS,
          choose_ks_for_goal(RequestingKS,Goal,_,AllParams,SomeKS,_),
          KSList),
    ( KSList = [] ->
      % @@Determine if there's a parent facilitator that can handle
      % the goal.  This needs work; probably should have a local record
      % of what the parent can handle.
      find_level(AllParams, Level, _NewParams),
      ( (on_exception(_, com:com_GetInfo(parent, fac_id(ParentBB)), fail), Level
> 0) ->
          Sources = [ParentBB]
      | otherwise ->
          Sources = []
      )
    | otherwise ->
        Sources = KSList
    ).

    % If Sources is bound, VERIFIES that all the Sources can be used
    % on the ICLGoal.  If var(Sources), finds all the Sources that can
    % be used.

% sources_for_compound_goal(RKS, ICLGoal, GlobalParams, Sources) :-

/*\

complete_concurrency(+Goal, -ConcurrentGoal).

TBD.

\*/

complete_concurrency(Goal, Goal).
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% GOAL EXECUTION: TOP LEVEL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

/*\
    execute_goal(+RequestingKS, +OrigGoal, +OrigParams, +CompleteGoal).

OrigGoal are OrigParams are exactly as submitted by some client agent
(RequestingKS).  CompleteGoal is the rewriting of OrigGoal that
ensures complete addressing.  OrigGoal and ICLGoal contain precisely
the same var's.

See global comments near the top of this file.

Note: the meaning of variable "Goal" and other variables ending in
"Goal" varies with context.  In some places they indicate an ICL
goal Source:Goal::Params (where Source and Params are both optional);
in other places, they indicate just the Goal part of an ICL goal.

\*/


execute_goal(RKS, OrigGoal, OrigParams, ICLGoal) :-
    % Here, ICLGoal may or may not include a Sources component.  Either
    % way, it gets handled by execute/7.
    % @@ What if OrigGoal's Params or GlobalParams has vars?
    % We remove addresses before calling term_vars only so as to avoid
    % a syntax error exception that comes up when ICLGoal = Addr:\+Goal
    remove_addresses(ICLGoal, TempGoal),
    term_vars(TempGoal, AllVars, _Singletons, _NonSingletons),
    new_goal_id(Id),
    % This means simply, "When the Solvers and solutions (in the form of
    % Bindings for AllVars) are known for Goal, call
    % unify_and_return_solutions(...)."
    assert(continuation(Id, Requestees, Solvers, Bindings,
           unify_and_return_solutions(Id,RKS,OrigGoal,OrigParams,AllVars,
                                      Requestees,Solvers,Bindings))),
    % This means: Find the Solvers and solutions:
    execute(Id, RKS, [], [], ICLGoal, OrigParams, AllVars).

/*\
 *     execute(Id, RKS, Requestees, Solvers, Goal, InheritedParams, Vars).

execute/7 satisfies the ICL goal Goal.  Id is an integer that
identifies a continuation assertion.  When the satisfaction of Goal
has been completed, the continuation assertion tells what to do next.
The satisfaction of Goal may be very simple, or may involve a number
of steps, depending on the form of Goal.

Requestees is a list of source id's of all sources asked to
participate in the satisfaction of whatever request contained Goal,
and Solvers is a list of source id's of sources that succeeded in
satisfying some part of the request (so Solvers is a subset of
Requestees.  These lists are being accumulated for return to the agent
that submitted the request.

Conceptually, execute/7 does this:
    findall(Vars, Goal, Bindings),
    append(Requestees, <list of KSs called on in the findall>, NewRequestees),
    append(Solvers, <list of KSs providing solutions in the findall>,
           NewSolvers),
    continue_execution(Id, RKS, NewRequestees, NewSolvers, Bindings)

The behavior of continue_execution, then, depends on a continuation/5
assertion, with Id as the first arg.

The important details have to do with how the satisfaction of the
"findall" part of this strategy may be delayed.

 *
\*/

execute(Id, RKS, Requestees, Solvers, built_in:ICLGoal, InheritedParams, Vars)
:-
    % This handles ICL built-ins, such as <, >, =, member/2, true, false, ...
    !,
    icl_GoalComponents(ICLGoal, _, Goal, Params),
    append(Params, InheritedParams, AllParams),
    oaa_Name(Facilitator),
    add_element(Facilitator, Requestees, NewRequestees),
    % If the requestor wants to know the solver, bind it here:
    ( memberchk(get_address(Facilitator), Params) -> true | true),

    ( oaa:passes_tests(Params) ->
        % @@The use of solution_limit and elsewhere here needs a close look:
        ( memberchk(solution_limit(N), AllParams) ->
          oaa:findNSolutions(N, Vars, call(Goal), Bindings)
      | otherwise ->
          findall(Vars, call(Goal), Bindings)
      )
    | otherwise ->
        Bindings = []
    ),
    ( Bindings == [] ->
        NewSolvers = Solvers
    | otherwise ->
        add_element(Facilitator, Solvers, NewSolvers)
    ),
    ( memberchk(reply(none), AllParams) ->
        continue_execution(Id, RKS, NewRequestees, NewSolvers, [Vars])
    | otherwise ->
        continue_execution(Id, RKS, NewRequestees, NewSolvers, Bindings)
    ).

    % Empty list of sources:
execute(Id, RKS, Requestees, Solvers, []:ICLGoal, _InheritedParams, _Vars) :-
    format('WARNING: No solvers for ICL goal or subgoal:~n  ~q~n',
         ICLGoal),
    continue_execution(Id, RKS, Requestees, Solvers, []).

    % Single KS in a list:
execute(Id, RKS, Requestees, Solvers, [KS]:G, Params, Vars) :-
    !,
    execute(Id, RKS, Requestees, Solvers, KS:G, Params, Vars).

    % Multiple KSs in a list:
execute(Id, RKS, Requestees, Solvers, [KS | Rest]:G, Params, Vars) :-
    !,
    execute_for_each_ks(Id, RKS, Requestees, Solvers, G, Params,
                  Vars, [KS | Rest]).

    % Solver is facilitator (me):
execute(Id, RKS, Requestees, Solvers, Source:ICLGoal, InheritedParams, Vars) :-
    oaa_Name(Facilitator),
    (Source = facilitator ; Source = Facilitator),
    !,
    icl_GoalComponents(ICLGoal, _, Goal, Params),
    % If the requestor wants to know the solver, bind it here:
    ( memberchk(get_address(Facilitator), Params) -> true | true),
    append(Params, InheritedParams, AllParams),
    findall(Vars,
            oaa:oaa_solve_local(Goal, InheritedParams),
          Bindings),
    ( memberchk(reply(none), AllParams) ->
        true
    | otherwise ->
      oaa_Name(KSName),
      add_element(KSName, Requestees, NewRequestees),
      ( Bindings == [] ->
            NewSolvers = Solvers
      | otherwise ->
            add_element(KSName, Solvers, NewSolvers)
      ),
      continue_execution(Id, RKS, NewRequestees, NewSolvers, Bindings)
    ).

% Note: this code was inherited from pre-compound-query facilitator.
% One significant change: when a goal is sent to a parent, we used to
% automatically include local blackboard solutions also.  We don't
% do this anymore.
%
% @@ Strategy should be re-evaluated at some point.  For instance,
% the use of var P2 might now cause things to break (the requesting
% agent might try to unify its copy of Params with P2).

execute(Id, RKS, Requestees, Solvers, Sources:ICLGoal, InheritedParams, Vars) :-
    on_exception(_, com:com_GetInfo(parent, fac_id(ParentBB)), fail),
    (Sources == parent ; Sources == ParentBB),
    !,

    icl_GoalComponents(ICLGoal, _, _Goal, Params),
    % If the requestor wants to know the solver, bind it here:
    % NO - it gets bound by the parent facilitator.
    % ( memberchk(get_address(ParentBB), Params) -> true | true),

    append(Params, InheritedParams, AllParams),
    % We don't need to check the level here (that's already been done),
    % but we do need to decrement its value by 1:
    find_level(AllParams, _Level, NewParams),
    oaa_TraceMsg('~nRouting goal "solve(~p)" to parent ~p.~n',
    [ICLGoal, ParentBB]),
    new_goal_id(NewId),
    oaa_PostEvent(ev_post_solve_from_bb(NewId, ICLGoal, NewParams),
                  [address(ParentBB)]),
    ( memberchk(reply(none), NewParams) ->
        unify_and_continue_execution(Id, RKS, ICLGoal, Vars,
          ParentBB, Requestees, Solvers, [ICLGoal])
    | otherwise ->
      % @@Shouldn't there be a time-check here?
      oaa:oaa_add_trigger_local(
              comm,
              event(ev_reply_solved_by_bb(NewId, _KS, ICLGoal, _P2,
                                          Solutions),
                    _),
            ev_unify_and_continue_execution(Id, RKS, ICLGoal, Vars,
                     ParentBB, Requestees, Solvers, Solutions),
              [recurrence(when), on(receive)])
    ).

    % Send the goal to an agent:
execute(Id, RKS, Requestees, Solvers, KS:ICLGoal, InheritedParams, Vars) :-
    !,
    icl_GoalComponents(ICLGoal, _, Goal, Params),
    append(Params, InheritedParams, AllParams),
    % @@What if the KS' status has changed since it was specified?
    % find_address allows for KS to be either numeric or symbolic.
    find_address(KS, KSId),
    % If the requestor wants to know the solver, bind it here:
    ( memberchk(get_address(KSId), Params) -> true | true),
    % Could do another check of the agent's validity:
    % ks_ready(KSId, _),

%     relevant_vars(Vars, Goal, GVars),
%     OptimizedG = findall(GVars, Goal, All),

    % Output trace message:
    ( oaa:oaa_trace(on) ->
        copy_term(ICLGoal, TraceCopy),
      numbervars(TraceCopy, 0, _),
        copy_term(InheritedParams, ParamsCopy),
      numbervars(ParamsCopy, 0, _),
      oaa_TraceMsg(
        '% Routing goal to ~w:~n%   ~w ~w~n~n',
        [KS, TraceCopy, ParamsCopy])
    | otherwise ->
        true
    ),

    new_goal_id(NewId),
%     oaa_PostEvent(KS, RKS, solve(NewId, OptimizedG::Params, [])),
    oaa_PostEvent(ev_solve(NewId, ICLGoal, InheritedParams),
                  [from(RKS), address(KSId)]),

    ( memberchk(reply(none), AllParams) ->
        unify_and_continue_execution(Id, RKS, ICLGoal, Vars,
                           KSId, Requestees, Solvers, [ICLGoal])
         % If time_limit specified in parameters, setup
        % time_trigger to wakeup if solutions hasn't been returned
        % in specified time.
    | otherwise ->
       ( memberchk(time_limit(NSecs), AllParams) ->
           add_time_check(NSecs, NewId, RKS, Goal,AllParams)
      | true),
      oaa:oaa_add_trigger_local(
        comm,
        event(ev_solved(NewId, _KS, ICLGoal, _P2, Solutions), _),
        ev_unify_and_continue_execution(Id, RKS, ICLGoal, Vars,
                                 KSId, Requestees, Solvers, Solutions),
        [recurrence(when), on(receive)])
%     poll_until_all_events([solved(Id, _KS, OptimizedG, P2, Solutions)]),
%     Solutions = [findall(GVars, Goal, All)],
%     respond_query(Id, RKS, Solvers, KS, Goal, P2, Solutions)
    % Backtrack over solutions:
%     member(GVars, All).
    ).

    % Negation:
execute(Id, RKS, Requestees, Solvers, ICLGoal, InheritedParams, Vars) :-
    icl_GoalComponents(ICLGoal, _, (\+ G1), Params),
    !,
    append(Params, InheritedParams, NewIParams),
    new_goal_id(NewId),
    assert(
        continuation(NewId, NewRequestees, NewSolvers, Bindings,
          continue_negation(Id, RKS, NewRequestees, NewSolvers, NewIParams,
                        Vars, Bindings))),
    execute(NewId, RKS, Requestees, Solvers, G1, NewIParams, Vars).

    % Conjunction:
execute(Id, RKS, Requestees, Solvers, ICLGoal, InheritedParams, Vars) :-
    icl_GoalComponents(ICLGoal, _, (G1, G2), Params),
    !,
    append(Params, InheritedParams, NewIParams),
    new_goal_id(NewId),
    assert(
        continuation(NewId, NewRequestees, NewSolvers, Bindings,
          continue_conjunction(Id, RKS, NewRequestees, NewSolvers, G2,
NewIParams,
                            Vars, Bindings))),
    execute(NewId, RKS, Requestees, Solvers, G1, NewIParams, Vars).

    % Local cut with alternative.  Note: this clause must precede
    % that for disjunction.
execute(Id, RKS, Requestees, Solvers, ICLGoal, InheritedParams, Vars) :-
    icl_GoalComponents(ICLGoal, _, (G1 -> G2 | G3), Params),
    !,
    append(Params, InheritedParams, NewIParams),
    new_goal_id(NewId),
    assert(
        continuation(NewId, NewRequestees, NewSolvers, Bindings,
          continue_local_cut(Id, RKS, NewRequestees, NewSolvers, G2, G3,
NewIParams,
                        Vars, Bindings))),
    execute(NewId, RKS, Requestees, Solvers, G1, NewIParams, Vars).
    % Local cut:
execute(Id, RKS, Requestees, Solvers, ICLGoal, InheritedParams, Vars) :-
    icl_GoalComponents(ICLGoal, _, (G1 -> G2), Params),
    !,
    append(Params, InheritedParams, NewIParams),
    new_goal_id(NewId),
    assert(
        continuation(NewId, NewRequestees, NewSolvers, Bindings,
          continue_local_cut(Id, RKS, NewRequestees, NewSolvers, G2, false,
NewIParams,
                       Vars, Bindings))),
    execute(NewId, RKS, Requestees, Solvers, G1, NewIParams, Vars).

    % Disjunction:
execute(Id, RKS, Requestees, Solvers, ICLGoal, InheritedParams, Vars) :-
    icl_GoalComponents(ICLGoal, _, (G1; G2), Params),
    !,
    append(Params, InheritedParams, NewIParams),
    new_goal_id(Id1),
    new_goal_id(Id2),
    assert(
        multiple_continuation([Id1, Id2], Requestees, AllRequestees,
                        Solvers, AllSolvers,
                        [], AllBindings,
          continue_execution(Id, RKS, AllRequestees, AllSolvers, AllBindings))),
    execute(Id1, RKS, Requestees, Solvers, G1, NewIParams, Vars),
    execute(Id2, RKS, Requestees, Solvers, G2, NewIParams, Vars).

    % Occasionally, a goal may have the form G::P (that is, no
    % address, and P is not compound), but it is still valid, so
    % long as G is valid.
    %
    % Ex.: ([7]:a1(1)::[...])::[...]
execute(Id, RKS, Requestees, Solvers, Goal::Params, InheritedParams, Vars) :-
    !,
    append(Params, InheritedParams, NewIParams),
    execute(Id, RKS, Requestees, Solvers, Goal, NewIParams, Vars).

execute(Id, RKS, Requestees, Solvers, G, _Params, _Vars) :-
    format('WARNING (execute/7): unrecognized goal form:~n    ~w~n', [G]),
    continue_execution(Id, RKS, Requestees, Solvers, []).

execute_for_each_ks(Id, RKS, Requestees, Solvers, Goal, Params, Vars, KSs) :-
    length(KSs, NumKSs),
    new_goal_ids(NumKSs, Ids),
    assert(
        multiple_continuation(Ids, Requestees, AllRequestees, Solvers,
AllSolvers, [], AllBindings,
          continue_execution(Id, RKS, AllRequestees, AllSolvers, AllBindings))),
    exec_for_each_ks(NumKSs, Ids, KSs, RKS, Requestees, Solvers, Goal,
                 Params, Vars).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% GOAL EXECUTION: INTERMEDIATE STEPS
% The predicates in this group define intermediate steps in the satisfaction
% of various ICL goal forms.
```

```
%
% Note: intermediate steps in handling of DISJUNCTION are handled by
% continue_execution, using the multiple_continuation assertion.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % This is used in satisfying [KS1, KS2, ...]:Goal.  Note that this is
    % equivalent to a disjunction (KS1:Goal ; KS2:Goal ; ....).  So we
    % are able to use the multiple_continuation assertion to accumulate
    % the solutions.
    %
    % We don't need Solvers, because ...

exec_for_each_ks(NumKSs, Ids, KSs, RKS, _Requestees, _Solvers,
             Goal, Params, Vars) :-
    retractall( ks_num(_) ),
    assert( ks_num(1) ),
    repeat,
    ks_num(Num),
    ( Num > NumKSs ->
      !
    | otherwise ->
      nth1(Num, KSs, KS),
      nth1(Num, Ids, Id),
      % We use a local cut to prevent some (harmless) backtracking.
      % This is one place where we don't need to pass Requestees and
      % Solvers through to execute (3rd and 4th args), because they are
      % filled in by handle_multiple_continuation.

      ( execute(Id, RKS, [], [], KS:Goal, Params, Vars) -> true ),
      NextNum is Num + 1,
      retractall( ks_num(_) ),
      assert( ks_num(NextNum) ),
      fail
    ).

    % This is used in satisfying (\+ Goal).  When this
    % pred. is called, Goal has just been completed.  Bindings gives
    % the solutions to  Goal.

continue_negation(Id, RKS, Requestees, Solvers, _Params, Vars, []) :-
    !,
    continue_execution(Id, RKS, Requestees, Solvers, [Vars]).
continue_negation(Id, RKS, Requestees, Solvers, _Params, _Vars, _Bindings) :-
    continue_execution(Id, RKS, Requestees, Solvers, []).

    % This is used in satisfying (Goal1, Goal2).  When this
    % pred. is called, Goal1 has just been completed.  Bindings gives
    % the solutions to Goal1.

continue_conjunction(Id, RKS, Requestees, Solvers, _Goal2, _Params, _Vars, [])
:-
    !,
    continue_execution(Id, RKS, Requestees, Solvers, []).
continue_conjunction(Id, RKS, Requestees, Solvers, Goal2, Params, Vars,
Bindings) :-
    length(Bindings, NumBindings),
    new_goal_ids(NumBindings, Ids),
    assert(
        multiple_continuation(Ids, Requestees, AllRequestees, Solvers,
AllSolvers, [], AllBindings,
          continue_execution(Id, RKS, AllRequestees, AllSolvers, AllBindings))),
    exec_for_each_binding(NumBindings, Ids, Bindings, RKS, Requestees, Solvers,
Goal2,
              Params, Vars).

    % We don't need Requestees or Solvers, because they are filled in
    % by handle_multiple_continuation.

exec_for_each_binding(NumBindings, Ids, Bindings, RKS, _Requestees, _Solvers,
            Goal, Params, Vars) :-
    retractall( binding_num(_) ),
    assert( binding_num(1) ),
    repeat,
    binding_num(Num),
    ( Num > NumBindings ->
      !
    | otherwise ->
      nth1(Num, Bindings, Binding),
      nth1(Num, Ids, Id),
      Vars = Binding,
      % We use a local cut to prevent some (harmless) backtracking.
      % This is one place where we don't need to pass Solvers through
      % to execute (3rd arg):
      ( execute(Id, RKS, [], [], Goal, Params, Binding) -> true ),
      NextNum is Num + 1,
      retractall( binding_num(_) ),
      assert( binding_num(NextNum) ),
      fail
    ).

    % This is used in satisfying Goal1 -> Goal2 | Goal3.  When this
    % pred. is called, Goal1 has just been completed.  Bindings gives
    % the solutions to Goal1.

    % No solutions to Goal1:
continue_local_cut(Id, RKS, Requestees, Solvers, _Goal2, Goal3, Params,
               Vars, []) :-
    !,
    ( Goal3 = false ->
        continue_execution(Id, RKS, Requestees, Solvers, [])
    | otherwise ->
      execute(Id, RKS, Requestees, Solvers, Goal3, Params, Vars)
    ).
    % Some solutions:
continue_local_cut(Id, RKS, Requestees, Solvers, Goal2, _Goal3, Params,
               Vars, [Binding1 | _]) :-
    new_goal_id(NewId),
    assert(
        continuation(NewId, NewRequestees, NewSolvers, Bindings,
          continue_execution(Id, RKS, NewRequestees, NewSolvers, Bindings))),
    Vars = Binding1,
    % local cut to prevent some (harmless) backtracking:
    ( execute(NewId, RKS, Requestees, Solvers, Goal2, Params, Binding1) -> true
).
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% GOAL EXECUTION: COMPLETION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This is called when the goal associated with Id has been completely
% satisfied.

continue_execution(Id, _RKS, Requestees, Solvers, Bindings) :-
      % Here we are BINDING the Solvers and Bindings var's. in the
      % continuation assertion.  The var. also appears in Continuation:
    ( retract(continuation(Id, Requestees, Solvers, Bindings, Continuation)) ->
      call(Continuation)
    | multiple_continuation(Ids, _, _, _, _, _, _),
      memberchk(Id, Ids) ->
      handle_multiple_continuation(Id, Requestees, Solvers, Bindings, Ids)
    | otherwise ->
      format('Internal Error: no continuation with id ~w~n', [Id])
    ).

handle_multiple_continuation(Id, Requestees, Solvers, Bindings, Ids) :-
    retract(multiple_continuation(Ids, PrevRequestees,
                          AllRequestees, PrevSolvers, AllSolvers,
                          PrevBindings, AllBindings,
                          Continuation)),
    del_element(Id, Ids, NewIds),
    append(PrevBindings, Bindings, NewBindings),
    append(PrevRequestees, Requestees, NewRequestees),
    append(PrevSolvers, Solvers, NewSolvers),
    ( NewIds = [] ->
      AllBindings = NewBindings,
      AllRequestees = NewRequestees,
      AllSolvers = NewSolvers,
      call(Continuation)
    | otherwise ->
      assert(multiple_continuation(NewIds, NewRequestees, AllRequestees,
                             NewSolvers, AllSolvers,
                             NewBindings, AllBindings,
                             Continuation))
    ).

% @@Let's see, if these args included the vars for any
% nested solvers params, we could probably instantiate solvers
% params in Goal...

unify_and_continue_execution(Id, RKS, Goal, Vars, Requestee, Requestees,
                       Solvers, Solutions) :-
    add_element(Requestee, Requestees, NewRequestees),
    ( Solutions == [] ->
        NewSolvers = Solvers
    | otherwise ->
        add_element(Requestee, Solvers, NewSolvers)
    ),
    findall(Vars,
          member(Goal, Solutions),
          Bindings),
    continue_execution(Id, RKS, NewRequestees, NewSolvers, Bindings).
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% GENERAL UTILITIES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

term_vars(Term, AllVars, SingletonVars, NonSingletonVars) :-
    with_output_to_chars(portray_clause(Term), Chars),
    with_input_from_chars(
                read_term([variable_names(Names), singletons(Singletons)],
                    Term1),
                Chars),
    extract_vars(Names, Singletons, AllVars, SingletonVars, NonSingletonVars),
    Term = Term1.

extract_vars([], _Singletons, [], [], []).
extract_vars([Name = Var | RestNames], Singletons, [Var | RestVars],
           [Var | RestSV], NonSingletonVars) :-
    memberchk(Name = Var, Singletons),
    !,
    extract_vars(RestNames, Singletons, RestVars, RestSV, NonSingletonVars).
extract_vars([_Name = Var | RestNames], Singletons, [Var | RestVars],
           RestSV, [Var | NonSingletonVars]) :-
    extract_vars(RestNames, Singletons, RestVars, RestSV, NonSingletonVars).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DEBUGGING UTILITIES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% static_test :-
%     Class = root,
%     KSName = dontcare,
%     BBName = dontcare,
%     oaa_read_setup_file,
%     oaa_init_flags,
%     assert(oaa_class(Class)),
%     oaa_SetupCommunication(Class, KSName, BBName, []),
%     on_exception(_, oaa_AppInit, true),
%     oaa_Ready(true).
%
% connect :-
%     % go(leaf, shell, root).
%     static_test.
%
% ce :-
%        repeat,
%        oaa_GetEvent(CallingKS, Event, 0),
%        ( Event = timeout ->
%        !,
%            format('No events~n', [])
%        | otherwise ->
%            oaa_process_event(CallingKS, Event),
%        fail
%        ).
% ce :-
%     format('No events~n', []).
%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% OrigGoal must be used in the return event, so that the
% requesting KS will identify it correctly.

unify_and_return_solutions(Id,RKS,OrigGoal,OrigParams,Vars,Requestees,Solvers,Bi
ndings) :-
    findall(OrigGoal,
          member(Vars, Bindings),
          Solutions),
    oaa_TraceMsg('~nRouting answers back to ~p:~n   ~p~n',
           [RKS,Solutions]),
    cancel_time_check(Id),
    remove_dups(Requestees, RequesteesSet),
    remove_dups(Solvers, SolversSet),
    % If present, bind solvers request in OrigParams:
    ( memberchk(get_address(RequesteesSet), OrigParams) -> true | true ),
    ( memberchk(get_satisfiers(SolversSet), OrigParams) -> true | true ),
    oaa_PostEvent(ev_reply_solved(RequesteesSet, SolversSet, OrigGoal,
OrigParams, Solutions),
                   [address(RKS)]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

APPENDIX A.II

Source code file named fac.pl.

```
%*****************************************************************************
%   File    : fac.pl
%   Primary Authors  : Adam Cheyer, David Martin
%   Purpose :  Provides communications and coordination of the activities
%            of a dynamic collection of client agents.
%   Updated : 12/98
%
%   ------------------------------------------------------------------------
%   Unpublished-rights reserved under the copyright laws of the United States.
%
%
%   Unpublished Copyright (c) 1998, SRI International.
%   "Open Agent Architecture" and "OAA" are Trademarks of SRI International.
%   ------------------------------------------------------------------------
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% fac.pl : the facilitator agent          Adam Cheyer
%                                               David Martin
%
% Provides communications and coordination of the activities of a
% dynamic collection of client agents.
%
% The blackboard can respond to the following external requests:
%
%     ev_post_event(AgentID, Cmd)    : sends event to the agent
%     ev_post_event(Cmd)      : sends event to all
%     ev_post_declare(Mode, Solvables, Params)
%                               : adds, removes or replaces solvables ON
%                               : the facilitator
%     ev_post_update(Mode, Clause, Params)
%                               : adds, removes, or replaces data
%                                 on appropriate agents
%     ev_post_trigger_update(Mode,TriggerType,Condition,Action,Params)
%                               : adds or removes a trigger
%                                 on appropriate agents
%     ev_post_solve(Goal, Params): finds agent(s) to solve Goal
%     connected(Connection)   : records that a client agent has connected
%       ev_connect(AgentInfo)
%                               : additional information from a client
%                               : agent (having version > 3.0)
%     end_of_file(Connection) : records that a client has closed its
%                         connection
%     ev_register_solvables : records the goals that an agent can solve.
%
% A facilitator uses the following events internally as trigger actions:
%
%     ev_respond_query(Id,ToKS,ByKS,G,OrigParams,Params,S)
%                               : Sends the result of a query back to KS
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

:- use_module(library(lists)).
:- use_module(library(basics)).
:- use_module(library(strings)).
:- use_module(library(charsio)).
:- use_module(library(sets)).
:- use_module(library(samsort)).  % for samsort(Ordered,Raw,Sort)
:- use_module(library(tcp), [tcp_now/1, tcp_time_plus/3,
                       tcp_schedule_wakeup/2, tcp_cancel_wakeup/2]).


% The file containing the com module is normally specified here.  For
% more info, see comments near the top of oaa.pl.
:- use_module(com_tcp, all).
:- use_module(oaa, all).

% Whether or not to load translations and compound query code
% is determined right here:
% :- [compound].
:- [translations].


:- multifile oaa_AppDoEvent/2.

:- dynamic time_limit_trigger/5.   % time_limit_trigger(Id,When,KS,Goal,Params)
:- dynamic goal_count/10.        % goal_count(GoalId,Goal,Params,EvParams,
                                   %   ToBeCalled,Called,Responders,Solvers,
                                   %   Answers,NumAnswers)
:- dynamic update_count/4.       % update_count(GoalId,NumAgentsRequested,
                           %   KSs, Updaters)
initial_solvables([
  solvable(agent_data(_Id, _Status, _Solvables, _Name), [type(data)],
                                                      [write(true)]),
    % Locations of all facilitators (currently maintained only by the 'root'
    % facilitator:
  solvable(agent_location(_Id2, _Name2, _Host2, _Port2), [type(data)],
                                                         [write(true)]),
    % Host (if known) of each client agent:
  solvable(agent_host(_Id3, _Name3, _Host3), [type(data)], [write(true)]),
  agent_version(_Id1, _Language1, _Version1),
  can_solve(_Goal4, _IdList4),
    %  For backwards compatibility.  In translations.pl, some events
    %  (write_bb, etc.) specify updates to this solvable.  Also, old-style
    % data triggers refer to it:
  solvable(data(_Item, _Data), [type(data)], [write(true)])
]).

/* Agent specific declarations */

oaa_AppInit :-
    oaa_SetTimeout(0).

/* This is the event generated by the TCP library.  Will be followed
   immediately by ev_connect/4, which is constructed by the client agent */
oaa_AppDoEvent(connected(Connection), _) :-
        !,
      format('~nKnowledge source connected: ~p~n~n', [Connection]),
      Id = Connection,
      oaa:oaa_add_data_local(agent_data(Id, open, [], Id), []),
      %% Maintain information of currently connected data.
      add_connected(Id, Connection).
```

```
/* For now, the ID of a client agent is the same as its connection (socket).
   This could change in the future, so we store Id and Connection
   as two separate entities. */
oaa_AppDoEvent(ev_connect(AgentInfoList), Params) :-
        memberchk( connection_id(Id), Params),
      oaa_Name(MyName),
      oaa_Id(MyId),
      MyLanguage = prolog,
      oaa_LibraryVersion(MyVersion),

      update_connected(Id, AgentInfoList),

      % preferred TCP transfer mechanism
      MyFormat = quintus_binary,

      % Inform the client of his Id, and info about me.
        com_SendData(Id,
         event(ev_connected([oaa_id(Id), fac_id(MyId), fac_name(MyName),
           fac_lang(MyLanguage), fac_version(MyVersion),
           format(MyFormat)]),
             [])).

/* Removes meta-data for KS when the KS deconnects */
oaa_AppDoEvent(end_of_file(Connection), _) :-
        Id = Connection,
      remove_connected(Id),
      oaa:oaa_remove_data_local(agent_data(Id, _Status, _Solvable, AgentName),
[]),
      format('~nKnowledge source disconnected: ~p (~p)~n~n', [Id,AgentName]),
        % remove all facts written by the agent
      % TBD: Is this getting all relevant triggers (see commented code below)?
        oaa:oaa_remove_data_owned_by(Id),

% Do we really want to do this?  I think clients who are interested could
% register a trigger on the agent_data predicate.
% Rather, I think we should check to see if any agents are currently waiting
% for this agent to solve some goal -- if the agent disconnects, we can assume
% that it won't be solving the goal anytime soon, and we should send back
% failure to the requesting agent.  See OAA 1.0 Facilitator, end_of_file()
% method.  [AJC, 11/24/97]
      post_to_all_clients(ev_agent_disconnected(Id)).

%     fail.
% TBD: This needs update to look at the persistence param.
% oaa_AppDoEvent(end_of_file(KS), _) :-
%     % remove all triggers for KS
%     on_exception(_, trigger(KS, Type, Kind, OpMask, Template, Cond, Action),
fail),
%     retract(trigger(KS, Type, Kind, OpMask, Template, Cond, Action)),
%     fail.
% oaa_AppDoEvent(end_of_file(_KS), _) :- !.

oaa_AppDoEvent(ev_ready(Name), Params) :-
        memberchk(from(Id), Params),
      % TBD: Let's have an error message if this fails:
      oaa:oaa_remove_data_local(agent_data(Id, _OldStatus, Solvables, _Name),
Params),
      oaa:oaa_add_data_local(agent_data(Id, ready, Solvables, Name), Params).

/* Stores the goals that a KS knows how to solve */
% Is this obsolete?
oaa_AppDoEvent(ev_register_solvables(Goals), Params) :-
        memberchk(from(KS), Params),
      oaa_AppDoEvent(ev_register_solvables(add,Goals,KS,[]), Params), !.

    % IMPORTANT: We assume the Solvables are in standard form and can
    % legally be added/removed/replaced for this agent.  Also, we take
    % care to keep the facilitator's copy of each client's solvables
    % identical to that stored at the client.  (Compare to code in
    % liboaa.pl, pred. oaa_declare_local).
oaa_AppDoEvent(ev_register_solvables(Mode,Solvs,AgentName,EvParams), Params) :-
        memberchk(from(KS), Params),
      oaa_Name(KSName),
      (oaa:oaa_remove_data_local(agent_data(KS, Status, List, _AgentName),
Params)
         ;
       format('STRANGE! register_solvables called by unknown KS!!!: ~p~n',
            [KS]),
       Status = ready,
       List = []
      ),
      icl_ConvertSolvables(PrettySolvs, Solvs),
      ( Mode == add, memberchk(if_exists(overwrite), EvParams) ->
          NewList = Solvs,
          format('~p (~p) can solve: ~n  ~p~n~n', [KS, AgentName,
                                          PrettySolvs])
      | Mode == add ->
          append(List, Solvs, NewList),
          format('~p (~p) has added solvables: ~n  ~p~n~n',
               [KS, AgentName, PrettySolvs])
      | Mode == remove ->
          subtract(List, Solvs, NewList),
          format('~p (~p) has removed solvables: ~n  ~p~n~n',
               [KS, AgentName, PrettySolvs])
      | Mode == replace ->
          memberchk(with(NewSolvable), EvParams),
          Solvs = [Solvable],
          oaa:replace_element(Solvable, List, NewSolvable, NewList),
          format('~p (~p) has replaced solvable:~n  ~p~nwith solvable:~n
~p~n~n',
               [KS, AgentName, Solvable, NewSolvable])
      ),
      oaa:oaa_add_data_local(agent_data(KS, Status, NewList, AgentName),
Params),

      % if a parent exists (not root), pass goals upward.
      (com:com_GetInfo(parent, connection(_C)) ->
         oaa_PostEvent( ev_register_solvables(Mode,Solvs,EvParams,KSName),
                        [address(parent)])
      |
         true),
      !.
```

```
/* A client has requested that I declare certain solvables.
   TBD: This is still sketchy; should include some validation of the
   request, and should ensure the perms and params are right. */
oaa_AppDoEvent(ev_post_declare(Mode, Solvables, Params), EvParams) :-
    memberchk(from(RequestingKS), EvParams),
    oaa:oaa_declare_local(Mode, Solvables, Params, NewSolvables),
    icl_ConvertSolvables(PrettySolvs, NewSolvables),
    oaa_Id(MyId),
    oaa_Name(MyName),
    format('~p (~p) has added solvables: ~n  ~p~n~n',
         [MyId, MyName, PrettySolvs]),
    oaa_PostEvent(
             ev_reply_declared(Mode, Solvables,Params, NewSolvables),
             [address(RequestingKS)]).

% A client requests a data solvable update operation (add, remove, replace)
% on the appropriate agents.
oaa_AppDoEvent(ev_post_update(Mode, Clause, Params), EvParams) :-
    ( Clause = (Head :- _Body) ->
      true
    | otherwise ->
      Head = Clause
    ),
    memberchk(from(RequestingKS), EvParams),
    % see if the query is addressed using address(KS) in Params
    check_address(Params, AddrKS),
    choose_agents_for_data(RequestingKS,Head,AddrKS,write,false,KSList),
    dispatch_update_request(RequestingKS, Mode, Clause, Params, KSList).

% A client requests a trigger update operation (Mode = add, remove, replace)
% on the appropriate agents.  For triggers of type comm' and time', the
% address parameter must be present (otherwise, the request should not
% have come to the facilitator). For the other types, the address is
% optional.

oaa_AppDoEvent(ev_post_trigger_update(Mode, data, Condition,
                              Action, Params), EvParams) :-
    !,
    memberchk(from(RequestingKS), EvParams),
    % see if the query is addressed using address(KS) in Params
    check_address(Params, AddrKS),
    choose_agents_for_data(RequestingKS,Condition,AddrKS,call,false,KSList),
    append(Params, EvParams, AllParams),
    dispatch_trigger_request(RequestingKS, Mode, data, Condition, Action,
                             AllParams, KSList).
oaa_AppDoEvent(ev_post_trigger_update(Mode, task, Condition,
                              Action, Params), EvParams) :-
    !,
    memberchk(from(RequestingKS), EvParams),
    % see if the query is addressed using address(KS) in Params
    check_address(Params, AddrKS),
    choose_agents_for_goal(RequestingKS,Condition,AddrKS,Params,false,KSList),
    append(Params, EvParams, AllParams),
    dispatch_trigger_request(RequestingKS, Mode, task, Condition, Action,
                             AllParams, KSList).

oaa_AppDoEvent(ev_post_trigger_update(Mode, Type, Condition,
                               Action, Params), EvParams) :-
    memberchk(from(RequestingKS), EvParams),
    check_address(Params, KSList),
    is_list(KSList),
    append(Params, EvParams, AllParams),
    dispatch_trigger_request(RequestingKS, Mode, Type, Condition, Action,
                       AllParams, KSList).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TBD: New for compound goals:

% If satisfaction of a compound goal is requested, and the compound query
% interpreter is not included, signal error condition:
oaa_AppDoEvent(ev_post_solve(Goal, Params), EvParams) :-
    \+ current_predicate(complete_goal, complete_goal(_,_,_,_)),
    \+ icl_BasicGoal(Goal),
    !,
    format('ERROR: This facilitator does not support compound goals~n', []),
    format('  Returning 0 solutions for goal:~n  ~w~n', [Goal]),
    oaa_Id(Facilitator),
    memberchk(from(RequestingKS), EvParams),
    oaa_PostEvent(
          ev_reply_solved([Facilitator],[],Goal,Params,[]),
[address(RequestingKS)]).

% If compound goal capabilities are included, ALL ev_post_solve events are
% handled here.  Otherwise, they fall through to later clauses.
oaa_AppDoEvent(ev_post_solve(Goal, Params), EvParams) :-
    current_predicate(complete_goal, complete_goal(_,_,_,_)),
    !,
    memberchk(from(RequestingKS), EvParams),
    complete_goal(RequestingKS, Goal, Params, CompletedGoal),
    execute_goal(RequestingKS, Goal, Params, CompletedGoal).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


/* Finds all KSs for a goal, asks them to solve it, then returns */
/* the answers to the calling KS                               */
oaa_AppDoEvent(ev_post_solve(Goal, Params), EvParams) :-
        memberchk(from(RequestingKS), EvParams),
      % see if the query is addressed using address(KS) in Params
      check_address(Params, AddrKS),
        choose_agents_for_goal(RequestingKS,Goal,AddrKS,Params,true,KSList),

      % if none of my agents know how to solve goal, send to parent
      (KSList = [] ->

         find_level(Params, Level, NewParams),
         ((com:com_GetInfo(parent, fac_name(ParentName)),
           Level > 0) ->
              oaa_TraceMsg('~nRouting goal "ev_solve(~p)" to parent ~p.~n',
             [Goal, ParentName]),

           new_goal_id(Id),
           oaa_PostEvent( ev_post_solve_from_bb(Id, Goal, NewParams),
                          [address(parent)]),
            % if answers requested,
            % send parent's answers directly back to requestingKS
            % as well as blackboard solutions
            (memberchk(reply(none), NewParams) -> true |
             % No longer valid:
             % send_blackboard_solutions(RequestingKS, Goal, Params),
             oaa:oaa_add_trigger_local(
                 comm,
               event(ev_reply_solved_by_bb(Id,SomeKS,Goal,Params2,Solutions),
                   _),
             ev_respond_query(Id,RequestingKS,SomeKS,Goal,Params,Params2,
                           Solutions),
             [recurrence(when), on(receive)])
            )
         |
            % root blackboard: doesn't know anyone who can solve goal
           (memberchk(reply(none), NewParams) -> true |
              oaa_Id(KSID),
              oaa_PostEvent(
               ev_reply_solved([KSID],[],Goal,Params,[]),
               [address(RequestingKS)])
           )
        )

      | otherwise ->
          dispatch_solve_request(RequestingKS, Goal, Params, EvParams, KSList)
      ).

/* Finds all KSs for a goal, asks them to solve it, then returns */
/* the answers to the calling BB                               */
oaa_AppDoEvent(ev_post_solve_from_bb(Id, Goal, Params), EvParams) :-
        memberchk(from(RequestingKS), EvParams),
      % see if the query is addressed using address(KS) in Params
      check_address(Params, AddrKS),
      choose_agents_for_goal(RequestingKS,Goal,AddrKS,Params,true,KSList),

      % if none of my agents know how to solve goal, send to parent
      (KSList = [] ->

         find_level(Params, Level, NewParams),
         % try to ask parent
         ((com:com_GetInfo(parent, fac_name(ParentName)),
           com:com_GetInfo(parent, fac_id(ParentId)), Level > 0) ->
              oaa_TraceMsg('~nRouting goal "ev_solve(~p)" to parent ~p.~n',
            [Goal, ParentName]),

            oaa_PostEvent( ev_post_solve_from_bb(Id, Goal, NewParams),
                           [address(parent)]),

            (memberchk(reply(none), NewParams) -> true |
              oaa:oaa_add_trigger_local(
                 comm,
               event(ev_reply_solved_by_bb(Id, _SomeKS, Goal, P2, Solutions),
                   _),
               ev_respond_bb_query(RequestingKS,ParentId,Id,Goal,Params,
                              P2, Solutions),
              [recurrence(when), on(receive)])
            )
         |
            % root blackboard : knows no solvers
            (memberchk(reply(none), Params) -> true |
              oaa_Name(KSName),
              oaa_PostEvent(
               ev_reply_solved_by_bb(Id, KSName,Goal,Params, []),
               [address(RequestingKS)])
            )
         )

      |
         member(SomeKS, KSList),     % backtrack over all KSs.
           oaa_TraceMsg('~nRouting goal to ~p: ~p~n',
                      [SomeKS, Goal]),

         oaa_PostEvent( ev_solve(Id, Goal, Params),
                        [address(SomeKS), from(RequestingKS)]),
         (memberchk(reply(none), Params) -> fail |
           oaa:oaa_add_trigger_local(
              comm,
              event(ev_solved(Id, _SomeKS, Goal, P2, Solutions), _),
              ev_respond_bb_or_post_higher(RequestingKS,SomeKS,Id,
              Goal,P2,Solutions),
            [recurrence(when), on(receive)])
         ),
         fail     % send events to all KSs that can solve goal.
      ).


oaa_AppDoEvent(wakeup(time_limit(Id)), _EvParams) :-
      retract(time_limit_trigger(Id,_When,RequestingKS,Goal,Params)),
      oaa_TraceMsg('~nTime limit expired. Goal failed:~n   ~p~n', [Goal]),
      oaa_Id(KSId),     % get local ksid
%      interpret(KSId,
%         ev_respond_query(-1,RequestingKS, KSId, Goal, Params, Params, [])).
      oaa_Interpret(
          ev_respond_query(-1,RequestingKS, KSId, Goal, Params, Params, []),
        [from(KSId)]).


% When asked by parent blackboard to solve a goal,
% route all answers back using "ev_solved(Id, KS, Goal, Params, Solutions)".
oaa_AppDoEvent(ev_solve(Id, Goal, Params), EvParams) :-
        memberchk(from(ParentBB), EvParams),
      oaa_Name(KSName),

      % see if the query is addressed using address(KS) in Params
      check_address(Params, AddrKS),
      choose_agents_for_goal(KSName,Goal,AddrKS,Params,true,KSList),

      % if none of my agents know how to solve goal, send empty solutions
      (KSList = [] ->
         (memberchk(reply(none), Params) -> true |
            oaa_PostEvent( ev_solved(Id,KSName,Goal,Params,[]),
                           [address(ParentBB)])
         )
      |
         member(SomeKS, KSList),    % backtrack over all KSs.
           oaa_TraceMsg('~nRouting goal "ev_solve(~p)" to ~p.~n', [Goal,
SomeKS]),

         oaa_PostEvent( ev_solve(Id, Goal, Params),
                        [address(SomeKS), from(ParentBB)]),
         (memberchk(reply(none), Params) -> fail |
           oaa:oaa_add_trigger_local(
              comm,
              event(ev_solved(Id, _SomeKS, Goal, P2, Solutions), _),
              ev_respond_to_parent(ParentBB,KSName,Id,Goal,Params,
                                   P2, Solutions),
            [recurrence(when), on(receive)])
         ),
         fail     % send events to all KSs that can solve goal.
      ).

/* If a KS is available, send it the message */
oaa_AppDoEvent(ev_post_event(Event), EvParams) :-
        memberchk(from(KS), EvParams),
        choose_ks_for_goal(KS, Event, _, [], SomeKS, _),
      oaa_PostEvent(Event, [address(SomeKS), from(KS)]),
      fail.

/* If a KS is available, send it the message */
oaa_AppDoEvent(ev_post_event(KSName, Event), EvParams) :-
      oaa_Name(KSName), !,
      % interpret(KS, Event).
      oaa_Interpret(Event, EvParams).
oaa_AppDoEvent(ev_post_event(KSName, Event), EvParams) :-
        memberchk(from(KS), EvParams),
        % agent must be "ready" to receive messages, or just
      % open if it is an agent compiled with old agentlib.
        (oaa:oaa_solve_local(agent_data(RealKS, ready, _Solvable,AgentName), [])
;
       oaa:oaa_solve_local(agent_data(RealKS, open, _Solvable,AgentName), []),
       oaa_Version(RealKS, _Language, Version),
       Version < 2.0),
      (match_ks(KSName, RealKS) ; KSName = AgentName),
      oaa_PostEvent(Event, [address(RealKS), from(KS)] ),
      fail.
% oaa_AppDoEvent(ev_post_event(_KS, _Event), _KS) :- !.
oaa_AppDoEvent(ev_post_event(_KS, _Event), _EvParams) :- !.



% Send back solutions to KS who originally requested them (with ev_post_solve)
%
% 970219: DLM: Added arg. OrigParams.  There is now a requirement that
% the params returned in a ev_reply_solved event must be unifiable with the
original
% params (from the corresponding solve event).
oaa_AppDoEvent(ev_respond_query(Id,RequestingKS, Requestee, Goal, OrigParams,
                      Params,Solutions), _EvParams) :-
        oaa_TraceMsg('~nRouting answers back to ~p:~n   ~p~n',
                [RequestingKS,Solutions]),
      cancel_time_check(Id),
      unify_params(OrigParams, Params, UParams),
      ( Solutions == [] ->
          Solvers = []
      | otherwise ->
          Solvers = [Requestee]
      ),
      oaa_PostEvent( ev_reply_solved([Requestee], Solvers, Goal, UParams,
Solutions),
                   [address(RequestingKS)]), !.

% Send back solutions to KS who originally requested them (with ev_post_solve)
% If no solutions, ask a higher blackboard
oaa_AppDoEvent(
   ev_respond_or_post_higher(RequestingKS, Solver,Id,Goal,P,Solutions),
   _EvParams) :-
      ((Solutions \== [] ; oaa:oaa_class(root)) ->
         cancel_time_check(Id), !,
         return_solutions(RequestingKS, Solver, Id, Goal,P,Solutions)
      |
         % @@DLM: The following needs work.  Must check goal_count status
         % before posting higher
         % sub-agents found no solutions: post higher
         com:com_GetInfo(parent, fac_id(ParentId)),
         find_level(P, Level, NewParams),
         Level > 0,
         oaa_PostEvent( ev_post_solve_from_bb(Id, Goal, NewParams),
                        [address(parent)]),
         oaa:oaa_add_trigger_local(
            comm,
            event(ev_reply_solved_by_bb(Id, _SomeKS, Goal, P2, Solutions),
                  _),
            ev_respond_query(Id,RequestingKS,ParentId,Goal,P,P2, Solutions),
            [recurrence(when), on(receive)])
      ).


% Send back acknowledgement to agent that originally requested an update.
oaa_AppDoEvent(
   ev_return_update(RequestingKS, Mode, Solver, Id, Clause, Params, Updaters),
   _EvParams) :-
      return_update(RequestingKS, Mode, Solver, Id, Clause, Params, Updaters).
% Send back acknowledgement to agent that originally requested a trigger
% update.
oaa_AppDoEvent(
   ev_return_trigger_update(RequestingKS, Mode, Solver, Id, Type, Condition,
                      Action, Params, Updaters),
   _EvParams) :-
        oaa_TraceMsg('~nRouting trigger updaters back to ~p:~n    ~p~n',
                [RequestingKS,Updaters]),
      return_trigger_update(RequestingKS, Mode, Solver, Id, Type, Condition,
                        Action, Params, Updaters).

% Send back solutions to a blackboard who requested them
%   (with ev_post_solve_from_bb)
%
```

```
% 970219: DLM: Added arg. OrigP.  There is now a requirement that
% the params returned in a ev_solved event must be unifiable with the original
% params (from the corresponding solve event).
oaa_AppDoEvent(ev_respond_bb_query(RequestingBB, Solver, Id,Goal,
                         OrigP, P,Solutions), _EvParams) :-
        unify_params(OrigP, P, UP),
        oaa_TraceMsg('~nRouting answers back to blackboard ~p:~n   ~p~n',
                [RequestingBB,Solutions]),
      oaa_PostEvent( ev_reply_solved_by_bb(Id,Solver,Goal,UP,Solutions),
                   [address(RequestingBB)]), !.

% Send back solutions to a blackboard who requested them
oaa_AppDoEvent(
    ev_respond_bb_or_post_higher(RequestingBB,Solver,Id,Goal,P,Solutions),
    _EvParams) :-
      ((Solutions \== [] ; oaa:oaa_class(root)) ->
           oaa_TraceMsg('~nRouting answers back to blackboard ~p:~n   ~p~n',
                [RequestingBB,Solutions]),
         oaa_PostEvent( ev_reply_solved_by_bb(Id, Solver, Goal, P,Solutions),
                    [address(RequestingBB)])
      |
         % sub-agents found no solutions: post higher
         com:com_GetInfo(parent, fac_id(ParentId)),
         find_level(P, Level, NewParams),
         Level > 0,
         oaa_PostEvent( ev_post_solve_from_bb(Id, Goal, NewParams),
                        [address(parent)]),
         oaa:oaa_add_trigger_local(
            comm,
            event(ev_reply_solved_by_bb(Id, _SomeKS, Goal, P2, Solutions),
                  _),
            ev_respond_bb_query(RequestingBB,ParentId,Id,Goal,P,P2,Solutions),
            [recurrence(when), on(receive)])
      ).


% Send back solutions to KS who originally requested them (with ev_post_solve)
%
% 970219: DLM: Added arg. OrigP.  There is now a requirement that
% the params returned in a ev_solved event must be unifiable with the original
% params (from the corresponding solve event).
oaa_AppDoEvent(ev_respond_to_parent(ParentBB,Solver,Id,Goal, OrigP,
                           P, Solutions), _EvParams) :-
        unify_params(OrigP, P, UP),
        oaa_TraceMsg('~nRouting answers back to parent bb ~p:~n   ~p~n',
                [ParentBB,Solutions]),
      oaa_PostEvent( ev_solved(Id, Solver, Goal, UP, Solutions),
                   [address(ParentBB)]), !.

oaa_AppDoEvent(ev_check_agent_name(KSName), EvParams):-
        memberchk(from(KS), EvParams),
        findall(KSName, oaa:oaa_solve_local(agent_location(_KSID, KSName ,_,_),
[]), L),
        (L==[] ->
         % @@tcp_send shouldn't be used:
        tcp_send(KS, 'UNIQUE');
        findall(KS1, oaa:oaa_solve_local(agent_location(_, KS1, _,_), []), R),
        tcp_send(KS, R)),!.

oaa_AppDoEvent(ev_register_port_number(Name,Address), EvParams) :- %+KS, +Port,
+Host
        memberchk(from(KS), EvParams),
        Address =.. [address, Port, Host],
        oaa:oaa_remove_data_local(agent_location(KS, _Name, _Port, _Host),
[]),!,
        oaa:oaa_add_data_local(agent_location(KS, Name, Port, Host), []),
        format('Agent ~p has Port: ~p , Host: ~p ~n', [KS, Port, Host]),
        !.

oaa_AppDoEvent(ev_register_port_number(Name,Address), EvParams) :- %+KS, +Port,
+Host
        memberchk(from(KS), EvParams),
        Address =.. [address, Port, Host],
        oaa:oaa_add_data_local(agent_location(KS, Name, Port, Host), []),
        format('Agent ~p has Port: ~p , Host: ~p ~n', [KS, Port, Host]),
        !.

oaa_AppDoEvent(ev_continue_execution(Id, RKS, Requestees, Solvers, Solutions),
             _EvParams) :-
        continue_execution(Id, RKS, Requestees, Solvers, Solutions).

% This is called from a trigger set in compound.pl.
oaa_AppDoEvent(
    ev_unify_and_continue_execution(Id, RKS, Goal, Vars, Requestee, Requestees,
Solvers, Solutions),
    _) :-
    unify_and_continue_execution(Id, RKS, Goal, Vars, Requestee, Requestees,
Solvers, Solutions).

/* Facilitator solvable: report the version and language of some
   connected agent. */
oaa_AppDoEvent(agent_version(Id, Language, Version), _EvParams) :-
    !,
    oaa_Version(Id, Language, Version).

/* Facilitator solvable: Find all agents who can solve goal */
oaa_AppDoEvent(can_solve(Goal, KSList), EvParams) :-
        ( memberchk(from(KS), EvParams) -> true | oaa_Id(KS) ),
      findall(SomeKS, choose_ks_for_goal(KS, Goal, _, [], SomeKS, _), KSList).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % choose_agents_for_goal(RequestingKS,Goal,AddrKS,Params,Sort,Agents).
  %
  % The first 4 arguments are exactly as expected by choose_ks_for_goal.
  % Sort, a boolean, tells whether to sort on utility.
choose_agents_for_goal(RequestingKS,Goal,AddrKS,Params,Sort,Agents) :-
    findall(
      p(Agent,Utility),
      choose_ks_for_goal(RequestingKS,Goal,AddrKS,Params,Agent,Utility),
      Pairs
    ),
    ( Sort ->
      samsort(oaa_utility_compare, Pairs, SortedPairs)
    | otherwise ->
```

```
        SortedPairs = Pairs
    ),
    findall(Agent, member(p(Agent,_Utility), SortedPairs), Agents).

  % choose_agents_for_data(RequestingKS,Goal,AddrKS,Perm,Sort,Agents).
  %
  % The first 4 arguments are exactly as expected by choose_ks_for_data.
  % Sort, a boolean, tells whether to sort on utility.
choose_agents_for_data(RequestingKS,Goal,AddrKS,Perm,Sort,Agents) :-
    findall(
      p(Agent,Utility),
      choose_ks_for_data(RequestingKS,Goal,AddrKS,Perm,Agent,Utility),
      Pairs
    ),
    ( Sort ->
      samsort(oaa_utility_compare, Pairs, SortedPairs)
    | otherwise ->
      SortedPairs = Pairs
    ),
    findall(Agent, member(p(Agent,_Utility), SortedPairs), Agents).

oaa_utility_compare(p(_Agent1,Utility1), p(_Agent2,Utility2)) :-
    Utility1 >= Utility2.


/* Finds a KS that knows how to solve Goal */

% backtracks over all KSs that know how to solve
%   a particular goal, except for RequestingKS, which is the
%   KS who asked for the goal to be solved in the
%   first place.  (RequestingKS is included if the 'reflexive' Param
%   is present.)
% MemberList can be a list used to reduce the set to at most MemberList
%   or can be a specific KS to try, or a variable.
% If an address is specified in MemberList, it can be the same as
%   RequestingKS (DLM, 96/10/30).
% Solvable lists can contain complex tests (AC, 97/2/5)
%     e.g. [goal1(Y),(g(X) :- X > 1,X < 10),goal2]
% Params is now used to check for 'reflexive' (DLM, 97/03/06).
% Utility is the numeric value the KS has associated with the
%   solvable.
choose_ks_for_goal(RequestingKS, Goal, MemberList, Params, SomeKS, Utility) :-
        var(MemberList),
      !,
        ks_ready(SomeKS, ListOfGoals),
      ( icl_GetParamValue(reflexive(true), Params) ->
            true
      | otherwise ->
          SomeKS \== RequestingKS
      ),
      oaa:oaa_goal_matches_solvables(Goal, ListOfGoals, _, Matched),
      Matched = solvable(_, SolveParams, _),
      icl_GetParamValue(utility(Utility), SolveParams).
choose_ks_for_goal(_RequestingKS, Goal, MemberList, _Params, SomeKS, Utility) :-
      (is_list(MemberList) ->
         member(SomeKS, MemberList)
      | SomeKS = MemberList),
   oaa:icl_true_id(SomeKS, TrueId),
    ks_ready(TrueId, ListOfGoals),
   oaa:oaa_goal_matches_solvables(Goal, ListOfGoals, _, Matched),
   Matched = solvable(_, SolveParams, _),
   icl_GetParamValue(utility(Utility), SolveParams).

% backtracks over all KSs that know how to write a particular goal (or
% read, though that's not currently used), except for RequestingKS,
% which is the KS who asked for the goal to be solved in the first
% place.  RequestingKS is never included, because he does the
% appropriate asserts locally, when appropriate.
%
% Perm is 'read' or 'write'.

choose_ks_for_data(RequestingKS, Goal, MemberList, Perm, SomeKS, Utility) :-
    var(MemberList),
   !,
    ks_ready(SomeKS, ListOfGoals),
   SomeKS \== RequestingKS,
   oaa:oaa_data_matches_solvables(Goal, ListOfGoals, Perm, _, Matched),
   Matched = solvable(_, SolveParams, _),
   icl_GetParamValue(utility(Utility), SolveParams).
choose_ks_for_data(_RequestingKS, Goal, MemberList, Perm, SomeKS, Utility) :-
   (is_list(MemberList) ->
     member(SomeKS, MemberList)
   | SomeKS = MemberList),
    ks_ready(SomeKS, ListOfGoals),
   oaa:oaa_data_matches_solvables(Goal, ListOfGoals, Perm, _, Matched),
   Matched = solvable(_, SolveParams, _),
   icl_GetParamValue(utility(Utility), SolveParams).

% ks_ready(*SomeKS, *ListOfGoals).
%   Backtracks over all agents that are ready to solve goals.
%   If SomeKS is bound (with an agent's local ID), only that agent is
%   considered.
ks_ready(SomeKS, ListOfGoals) :-
    % agent must be "ready" to receive messages, or just
   % open if it is an agent compiled with old agentlib.
    (oaa:oaa_solve_local(agent_data(SomeKS, ready, ListOfGoals,_AgentName),
[]) ;
   oaa:oaa_solve_local(agent_data(SomeKS, open, ListOfGoals,_AgentName),
[]),
   oaa_Version(SomeKS, _Language, Version),
   Version < 2.0).
% Facilitator agents look up their own solvables in oaa_solvables/1.
ks_ready(SomeKS, ListOfGoals) :-
  oaa_Id(SomeKS),
  oaa:oaa_solvables(ListOfGoals).

match_ks(all, _KS).
match_ks(KS, KS).

% If params contains a VALID address (symbolic name or id) for one or more
% agents, return the agents' ids.
% If params contains an INVALID address, remove it from the list returned.
% Otherwise, KSAddr should return a variable.
% 97-05-23 (DLM): The address param now should always contain a list,
% but we'll check just to be safe.

check_address(Params, KSAddr) :-
      memberchk(address(Addr), Params),
      ( is_list(Addr) ->
          AddrList = Addr
      |   AddrList = [Addr]),
      find_addresses(AddrList, KSAddr),
      !.
check_address(_Params, _SomeKS).

find_addresses([], []).
find_addresses([Addr | Addrs], [Id | Ids]) :-
    find_address(Addr, Id),
    !,
    find_addresses(Addrs, Ids).
find_addresses([_Addr | Addrs], Ids) :-
    find_addresses(Addrs, Ids).

% Given an agent id (eg. 5) or a symbolic name (eg. 'interface')
% returns the local id for the reference.
%
% TBD: This does not yet handle remote addresses (associated with a different
% facilitator).

find_address(addr(Addr), SomeKS) :-
    com:com_GetInfo(incoming, oaa_addr(Addr)),
    % That's me, the facilitator.
    !,
    oaa_Id(SomeKS).
find_address(addr(Addr, SomeKS), SomeKS) :-
    com:com_GetInfo(incoming, oaa_addr(Addr)),
    % One of my clients.
    !,
    % Make sure it's current:
    oaa:oaa_solve_local(agent_data(SomeKS, _, _ListOfGoals, _AgentName), []).
find_address(name(Name), SomeKS) :-
    !,
    atom(Name),
    oaa:oaa_solve_local(agent_data(SomeKS, _, _ListOfGoals, Name), []).
find_address(SomeKS, SomeKS) :-
    oaa:oaa_solve_local(agent_data(SomeKS, _, _ListOfGoals, _AgentName), []),
    !.


find_level(Params, Level, NewParams) :-
      oaa:remove_element(level_limit(Level), Params, Params2), !,
      (Level > 0 ->
         NewLevel is Level - 1
      |  NewLevel is 0),
      NewParams = [level_limit(NewLevel)|Params2].
find_level(Params, 1, Params).

post_to_all_clients(Event) :-
        oaa_Id(FacId),
        oaa:oaa_solve_local(agent_data(ClientId, ready, _Solvable,_AgentName),
[]),
      ClientId \== FacId,
      oaa_PostEvent(Event, [address(ClientId), from(FacId)] ),
      fail.
post_to_all_clients(_Event).

    % This is called when length of KSList is > 0.
    %
    % goal_count(GoalId,Goal,Params,EvParams,ToBeCalled,Called,
    %            Responders,Solvers,Answers,NumAnswers)

dispatch_solve_request(RequestingKS, Goal, Params, EvParams, KSList) :-
    new_goal_id(Id),
    % Note that reply (none) overrides parallel_ok (false).  We can't
    % provide parallel_ok (false) if no replies come back from solvers.
    ( memberchk(reply(none), Params) ->
      dispatch_solve_events(KSList, Id, RequestingKS, Goal, Params, EvParams)
    | memberchk(parallel_ok(false), Params) ->
      % Dispatch to one KS; save the rest for later.
      KSList = [FirstKS | Rest],
        assert(goal_count(Id, Goal, Params, EvParams, Rest,
                        [FirstKS], [], [], [], 0)),
      dispatch_solve_event(Id, RequestingKS, Goal, Params, EvParams, FirstKS)
    | otherwise ->
      % Dispatch to all KSs.
      assert(goal_count(Id, Goal, Params, EvParams, [],
                        KSList, [], [], [], 0)),
      dispatch_solve_events(KSList, Id, RequestingKS, Goal, Params, EvParams)
    ).


dispatch_solve_events([], _Id, _RequestingKS, _Goal, _Params, _EvParams).
dispatch_solve_events([SomeKS | Rest], Id, RequestingKS, Goal,
                  Params, EvParams) :-
    dispatch_solve_event(Id, RequestingKS, Goal, Params, EvParams, SomeKS),
    dispatch_solve_events(Rest, Id, RequestingKS, Goal, Params, EvParams).

dispatch_solve_event(Id, RequestingKS, Goal, Params, EvParams, SomeKS) :-
    oaa_Id(SomeKS),
    % That's me, the facilitator.
    !,
    icl_GoalComponents(Goal, _, _, GoalParams),
    append(Params, EvParams, InheritedParams),
    append(GoalParams, InheritedParams, AllParams),
    findall(Goal,
              % InheritedParams here is right, not AllParams:
            oaa:oaa_solve_local(Goal, InheritedParams),
          Solutions),
    ( memberchk(reply(none), AllParams) ->
      true
    | otherwise ->
        oaa_AppDoEvent(

ev_respond_or_post_higher(RequestingKS,SomeKS,Id,Goal,Params,Solutions),
        [])
    ).
dispatch_solve_event(Id, RequestingKS, Goal, Params, _EvParams, SomeKS) :-
    oaa_TraceMsg('~nRouting goal "ev_solve(~p)" to ~p.~n', [Goal, SomeKS]),
         % ask a sub-agent to try and solve goal.
         %   if solutions are returned, pass them to requestingKS.
         %   otherwise, ask higher blackboard to try and solve goals.
         % note: send ev_solve(id(Id,SomeKS), ...) as a means of insuring
         %   that each ev_solved() trigger is unique and only matches
         %   exactly one response.  We use _SomeKS in the field indicating
         %   which agent actually solved the goal because individual
         %   agents don't necessarily know their internal unique ID #.
    oaa_PostEvent( ev_solve(id(Id,SomeKS), Goal, Params),
                   [address(SomeKS), from(RequestingKS)]),
    ( memberchk(reply(none), Params) ->
        true
    | otherwise ->
           % If time_limit specified in parameters, setup
           % time_trigger to wakeup if solutions hasn't been returned
           % in specified time.
        ( memberchk(time_limit(NSecs), Params) ->
          add_time_check(NSecs, Id, RequestingKS, Goal,Params)
      | true),
      oaa:oaa_add_trigger_local(
        comm,
        event(ev_solved(id(Id,SomeKS), _SomeKS, Goal, P2, Solutions), _),
        ev_respond_or_post_higher(RequestingKS,SomeKS,Id,Goal,P2,Solutions),
        [recurrence(when), on(receive)])
    ).

% return_solutions(+RequestingKS, +Responder, +Id, +Goal, +P, +NewSolutions).
% Having just received solutions from a Responder, take the appropriate action.
%
% Even though the Responder has returned copies of the goal and params,
% we don't need them because we have a local copy in goal_count.
%
% @@DLM: Unresolved question about streaming: Should we stream the
% responses with 0 solutions?  [My thinking is "yes".]
return_solutions(RequestingKS, Responder, Id, _Goal, _P, NewSolutions) :-
    % ToBeCalled lists solvers not yet called.  PrevCalled lists
    % the called solvers that have yet to respond.
    retract(goal_count(Id, Goal, Params, EvParams,
                       ToBeCalled, PrevCalled, PrevResponders,
                       PrevSolvers, PrevSolutions, PrevNumSol)),
    !,
      % Take Responder out of the called list:
    ( selectchk(Responder, PrevCalled, Called) ->
        true
    | otherwise ->
      format('ERROR: Inappropriate ev_solved event received:~n', []),
      format('  ~w ~w ~w ~w~n', [RequestingKS, Responder, Id, Goal]),
      Called = PrevCalled
    ),
      % and put him into the responder list:
    append(PrevResponders, [Responder], Responders),
      % The solvers are just the responders that succeeded:
    ( NewSolutions = [] ->
      NewSolvers = []
    | otherwise ->
      NewSolvers = [Responder]
    ),
append(PrevSolvers, NewSolvers, Solvers),
append(PrevSolutions, NewSolutions, Solutions),
length(NewSolutions, NewNumSol),
NumSol is PrevNumSol + NewNumSol,

  % This case means that either: (1) we've gotten responses from all
  % solvers; and/or (2) we have reached the desired number of solutions.
  % By not saving goal_count, we ensure that any additional returned
  % solutions are ignored:
( ((ToBeCalled == [], Called == []) ;
   (memberchk(solution_limit(Limit), Params), NumSol >= Limit)) ->
      % This test is a place-holder; streaming not yet official:
    ( memberchk(reply(streaming), Params) ->
        Return = ev_reply_solved([Responder], NewSolvers, Goal, Params,
                               NewSolutions)
  | otherwise ->
        Return = ev_reply_solved(Responders, Solvers, Goal, Params,
                               Solutions)
  ),
  Save = false
  % This case happens with parallel_ok(false):
| ToBeCalled = [Next | Rest] ->
  dispatch_solve_event(Id, RequestingKS, Goal, Params, EvParams, Next),
      % This test is a place-holder; streaming not yet official:
    ( memberchk(reply(streaming), Params) ->
        Return = ev_reply_solved([Responder], NewSolvers, Goal, Params,
                               NewSolutions),
        Save = goal_count(Id, Goal, Params, EvParams,
                        Rest, [Next|Called], [], [], [], NumSol)
    | otherwise ->
      Return = false,
        Save = goal_count(Id, Goal, Params, EvParams,
                        Rest, [Next|Called], Responders, Solvers,
                    Solutions, NumSol)
    )
  % Still waiting for some called solvers to respond:
| Called = [_ | _] ->
      % This test is a place-holder; streaming not yet official:
    ( memberchk(reply(streaming), Params) ->
        Return = ev_reply_solved([Responder], NewSolvers, Goal, Params,
                               NewSolutions),
        Save = goal_count(Id, Goal, Params, EvParams,
                        ToBeCalled, Called, [], [], [], NumSol)
    | otherwise ->
      Return = false,
        Save = goal_count(Id, Goal, Params, EvParams,
                        ToBeCalled, Called, Responders, Solvers,
                    Solutions, NumSol)
    )
),
( Save == false ->
    true
| otherwise ->
   assert(Save)
),
( Return == false ->
    true
    | otherwise ->
        oaa_TraceMsg('~nRouting answers back to ~p:~n   ~p~n',
                  [RequestingKS,Return]),
        oaa_PostEvent(Return, [address(RequestingKS)])
    ).
return_solutions(_RequestingKS, _Responder, _Id, _Goal, _P, _NewSolutions).

dispatch_update_request(RequestingKS, Mode, Clause, Params, []) :-
    % No agents able to perform the requested update:
    !,
    ( memberchk(reply(none), Params) ->
      true
    | otherwise ->
      Event = ev_reply_updated(Mode, Clause, Params, [], []),
      oaa_PostEvent(Event, [address(RequestingKS)])
    ).
dispatch_update_request(RequestingKS, Mode, Clause, Params, KSList) :-
    new_goal_id(Id),
    length(KSList,NumKSsForGoal),
    % if more than one KS can solve the goal, remember so that
    % we can collect answers from all of them later
    ( NumKSsForGoal > 1 ->
        assert(update_count(Id, NumKSsForGoal, [], []))
    | otherwise ->
        true
    ),
    member(SomeKS, KSList),    % backtrack over all KSs.
    dispatch_update_event(Id, RequestingKS, Mode, Clause, Params, SomeKS),
    fail.
dispatch_update_request(_RequestingKS, _Mode, _Clause, _Params, _KSList).

dispatch_update_event(Id, RequestingKS, Mode, Clause, Params, SomeKS) :-
    oaa_Id(SomeKS),
    % That's me, the facilitator.
    !,
    ( Mode == add ->
        Functor = oaa_add_data_local
    | Mode == replace ->
        Functor = oaa_replace_data_local
    | otherwise ->
        Functor = oaa_remove_data_local
    ),
    append(Params, [from(RequestingKS)], AllParams),
    Goal =.. [Functor, Clause, AllParams],
    ( call(oaa:Goal) ->
        Updaters = [SomeKS]
    | otherwise ->
        Updaters = []
    ),
    ( memberchk(reply(none), Params) ->
      true
    | otherwise ->
        % Params must be returned here (not AllParams):
        return_update(RequestingKS,Mode,SomeKS,Id, Clause,Params,Updaters)
    ).
dispatch_update_event(Id, RequestingKS, Mode, Clause, Params, SomeKS) :-
    oaa_TraceMsg('~nRouting request "ev_update(~p, ~p, ~p)" to ~p.~n',
                [Mode, Clause, Params, SomeKS]),
     append(Params, [from(RequestingKS)], AllParams),
     oaa_PostEvent(
              ev_update(id(Id,SomeKS), Mode, Clause, AllParams),
              [address(SomeKS)]),
     ( memberchk(reply(none), Params) ->
       true
     | otherwise ->
       % TBD: Do we want to set a time trigger here?
       oaa:oaa_add_trigger_local(
           comm,
           event(ev_updated(id(Id,SomeKS), _Mode, _Clause, _P2, Updaters), _),
             % Params must be returned here (not AllParams):
           ev_return_update(RequestingKS,Mode,SomeKS,Id,
                                     Clause,Params,Updaters),
           [recurrence(when), on(receive)])
     ).


% Returns, to requesting KS, the addresses of all agents (including
% facilitator if appropriate), that attempted (NewKSs) and that actually
% satisfied (Updaters) an update request.
%
% NewUpdaters is always either [], or a singleton list.
%
% Possible values for Mode: add, remove, replace.
%
% Note: Params must be returned in ev_reply_updated, so it must be
% unifiable with the params embedded in the requesting event (ev_post_event).

return_update(RequestingKS, Mode, Responder, Id, Clause, Params,
            NewUpdaters) :-
       retract(update_count(Id, AgentsLeft, PrevKSs, PrevUpdaters)),
       append(PrevUpdaters, NewUpdaters, Updaters),
       append(PrevKSs, [Responder], NewKSs),
       ( AgentsLeft > 1 ->
          NewAgentsLeft is AgentsLeft - 1,
          assert(update_count(Id, NewAgentsLeft, NewKSs, Updaters))
       | otherwise ->
           oaa_TraceMsg('~nRouting updaters back to ~p:~n   ~p~n',
                    [RequestingKS,Updaters]),
           Event = ev_reply_updated(Mode, Clause, Params, NewKSs, Updaters),
           oaa_PostEvent(Event, [address(RequestingKS)])
       ), !.
return_update(RequestingKS, Mode, Responder, _Id, Clause, Params, Updaters) :-
     oaa_TraceMsg('~nRouting updaters back to ~p:~n   ~p~n',
              [RequestingKS,Updaters]),
     Event = ev_reply_updated(Mode, Clause, Params, [Responder], Updaters),
     oaa_PostEvent(Event, [address(RequestingKS)]).

     % No agents able to install this trigger:
dispatch_trigger_request(RKS, Mode, Type, Condition, Action, Params, []) :-
     !,
     ( memberchk(reply(none), Params) ->
       true
     | otherwise ->
       Event = ev_reply_trigger_updated(Mode, Type, Condition, Action, Params,
                                   [], []),
      oaa_PostEvent(Event, [address(RKS)])
    ).
dispatch_trigger_request(RKS, Mode, Type, Condition, Action, Params, KSList) :-
    new_goal_id(Id),
    length(KSList,NumKSsForGoal),
    % if more than one KS can solve the goal, remember so that
    % we can collect answers from all of them later
    ( NumKSsForGoal > 1 ->
        assert(update_count(Id, NumKSsForGoal, [], []))
    | otherwise ->
        true
    ),
    member(SomeKS, KSList),     % backtrack over all KSs.
    dispatch_trigger_event(Id, RKS, Mode, Type, Condition, Action, Params,
SomeKS),
    fail.
dispatch_trigger_request(_RKS, _Mode, _Type, _Condition, _Action, _Params,
                   _KSList).

dispatch_trigger_event(Id, RKS, Mode, Type, Condition, Action, Params,
                   SomeKS) :-
    oaa_Id(SomeKS),
    % That's me, the facilitator.
    !,
    ( Mode == add ->
        Functor = oaa_add_trigger_local
    | otherwise ->
        Functor = oaa_remove_trigger_local
    ),
    Goal =.. [Functor, Type, Condition, Action, Params],
    ( call(oaa:Goal) ->
        Updaters = [SomeKS]
    | otherwise ->
        Updaters = []
    ),
    ( memberchk(reply(none), Params) ->
      true
    | otherwise ->
        return_trigger_update(RKS, Mode, SomeKS, Id, Type,
                          Condition, Action, Params, Updaters)
    ).
dispatch_trigger_event(Id, RKS, Mode, Type, Condition, Action, Params,
                   SomeKS) :-
    oaa_TraceMsg('~nRouting request~n  ev_update_trigger(~p, ~p, ~p, ~p, ~p)~nto
~p.~n',
              [Mode, Type, Condition, Action, Params, SomeKS]),
    oaa_PostEvent(
       ev_update_trigger(id(Id,SomeKS), Mode, Type, Condition, Action, Params),
       [address(SomeKS), from(RKS)]),
    ( memberchk(reply(none), Params) ->
      true
    | otherwise ->
      % TBD: Do we want to set a time trigger here?
      oaa:oaa_add_trigger_local(
          comm,
          event(ev_trigger_updated(id(Id,SomeKS), _Mode, _Type, _Condition,
_Action, P2, Updaters), _),
          ev_return_trigger_update(RKS,Mode,SomeKS,Id,
                              Type,Condition,Action,P2,Updaters),
          [recurrence(when), on(receive)])
    ).


% Returns, to requesting KS, the addresses of all agents (including
% facilitator if appropriate), that attempted (NewKSs) and that actually
% satisfied (Updaters) a trigger update request.
%
% NewUpdaters is always either [], or a singleton list.
%
% Possible values for Mode: add, remove.

return_trigger_update(RequestingKS, Mode, Responder, Id,
                  Type, Condition, Action, Params, NewUpdaters) :-
      retract(update_count(Id, AgentsLeft, PrevKSs, PrevUpdaters)),
      append(PrevUpdaters, NewUpdaters, Updaters),
      append(PrevKSs, [Responder], NewKSs),
      ( AgentsLeft > 1 ->
         NewAgentsLeft is AgentsLeft - 1,
         assert(update_count(Id, NewAgentsLeft, NewKSs, Updaters))
      | otherwise ->
          Event = ev_reply_trigger_updated(Mode,Type,Condition,Action,
                                     Params, NewKSs, Updaters),
          oaa_PostEvent(Event, [address(RequestingKS)])
      ), !.
return_trigger_update(RequestingKS, Mode, Responder, _Id,
                  Type, Condition, Action, Params, Updaters) :-
    Event = ev_reply_trigger_updated(Mode, Type, Condition, Action,
                             Params, [Responder], Updaters),
    oaa_PostEvent(Event, [address(RequestingKS)]).

    % unify_params(+OrigParams, +Params, -UnifiedParams).
    %
    % There is now (970219) a requirement that the params returned in
    % a ev_solved or ev_solved_by_bb event must be unifiable with the original
    % params from the corresponding solve request.  In some situations*, the
    % Params returned to the facilitator by a solver may not unify with
    % the OrigParams, but may contain individual elements with variables
    % instantiated by the solver.  This pred. can be used to save these
    % instantiations.
    %
    % *Such as, when find_level has been used to create a new params list.
unify_params([], _Params, []).
unify_params([OrigParam | Rest], Params, [OrigParam | UnifiedRest]) :-
    ( memberchk(OrigParam, Params) | true ),
    !,
    unify_params(Rest, Params, UnifiedRest).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% These are extremely simple predicates for maintaining com_connection_info/5,
% which keeps info about the agents to which this agent currently has
% a communications channel.
```

```
add_connected(Id, Connection) :-
   assert(com:com_connection_info(Id, unknown, child,
                     [connection(Connection),oaa_id(Id)], connected)).

update_connected(Id, AddInfo) :-
   com_AddInfo(Id, AddInfo).

% remove_connected(+Id).
remove_connected(Id) :-
   retractall(com:com_connection_info(Id, _, _, _, _)).

% if the time_limit(NSec) parameter is sent, install wakeup on server
% to indicate the request has failed if not achieved in the correct time.
add_time_check(NSecs, Id, RequestingKS, Goal,Params) :-
   (time_limit_trigger(Id,_When, RequestingKS,_Goal,_Params) ->
      true   % already added for this goal request
   |
      tcp_now(Now),
      tcp_time_plus(Now,NSecs,Soon),
      tcp_schedule_wakeup(Soon, time_limit(Id)),
      assert(time_limit_trigger(Id,Soon,RequestingKS,Goal,Params)),
      oaa_TraceMsg('~nTime limit check added for ~p~n',[Goal])
   ), !.


% if solutions are returned before a time_limit_trigger has expired,
%  remove the trigger.
cancel_time_check(Id) :-
   retract(time_limit_trigger(Id,When,_RequestingKS,Goal,_Params)),
   tcp_cancel_wakeup(When,time_limit(Id)),
   oaa_TraceMsg('~nTime limit check removed because solution returned.~n
~p~n',
      [Goal]), !.
cancel_time_check(_Id).



/* Generates a unique ID for a goal.        */
/* ID's should be unique across blackboards*/
/*   which is why we use the KSName prefix */
/* Goal counters are used to make sure the */
/* solution really matches the query.       */

new_goal_id(NewId) :-
      oaa_Name(KSName),
      concat(KSName, '_', Tmp),
      gensym(Tmp, NewId).

% Returns a list containing Num new goal ids.

new_goal_ids(Num, [NewId | RestIds]) :-
    Num > 0,
    !,
    new_goal_id(NewId),
    NewNum is Num - 1,
    new_goal_ids(NewNum, RestIds).
new_goal_ids(_Num, []).
```

```
start :-
    runtime_entry(start).

runtime_entry(start) :-
    initial_solvables(Solvables),
    com_ListenAt(incoming, CInfo),
    format('Listening at ~p~n~n', [CInfo]),
    oaa_RegisterCallback(app_do_event, user:oaa_AppDoEvent),
    oaa_Register(incoming, 'root', Solvables),
    on_exception(_, oaa_AppInit, true),
    oaa_MainLoop(true).


runtime_entry(abort) :- !.
%     format('Closing all connections...~n',[]),
%     close_all_connections.

% If the Facilitator is killed (ctrl-c) before disconnecting
% all clients, it will not free the port.
% This code is an attempt to fix this problem, but it doesn't
% help.  Why not???
% close_all_connections :-
%     tcp_connected(X,Y),
%     tcp_destroy_listener(Y),
%     tcp_shutdown(X),
%     fail.
% close_all_connections :-
%     tcp_reset, fail.
```

APPENDIX A.III

Source code file named libcom_tcp.pl.

```
%******************************************************************************
%   File    : libcom_tcp.pl
%   Primary Authors  : Adam Cheyer, David Martin
%   Purpose : TCP instantiation of lowlevel communication primitives for OAA
%   Updated : 01/98
%
%   -------------------------------------------------------------------------
%   Unpublished-rights reserved under the copyright laws of the United States.
%
%
%   Unpublished Copyright (c) 1993-98, SRI International.
%   "Open Agent Architecture" and "OAA" are Trademarks of SRI International.
%   -------------------------------------------------------------------------
%
%



%******************************************************************************
%* RCS Header and internal version
%******************************************************************************

:- module(com,
        [com_Connect/2,
         com_Disconnect/1,
         com_ListenAt/2,
         com_SendData/2,
         com_SelectEvent/2,
         com_AddInfo/2,
         com_GetInfo/2]).

% rcs version number
rcsid(libcom_tcp, '$Header:
/tmp_mnt/home/zuma1/martin/OAA/agents/beta/prolog/RCS/com_tcp.pl,v 1.10
1998/05/06 22:35:36 martin Exp $').

:- use_module(library(sets)).
:- use_module(library(tcp)).
:- use_module(library(basics)).
:- use_module(library(lists)).
:- use_module(library(charsio)).  % for sprintf and with_output_to_chars
:- use_module(library(ask)).      % for ask_oneof
:- use_module(library(environ)).  % read environment vars
:- use_module(library(files)).        % can_open_file
:- use_module(library(strings)).  % for concat

:- dynamic
      com_connection_info/5,  % id, commtype, client/server, commInfo, status
        com_already_loaded/1. % filename



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    com_Connect(+ConnectionId, ?Address)
% purpose: Given a connection ID and an address, initiates a client connection
% remarks:
```

```
%  - if Address is a variable, instantiates the Address by using
%     com_ResolveVariables, which looks in a setup file, command line, and
%     environment variables for the required info.
%  - stores the connection info for connection ID in com_connection_info/5.
%  - fails if connection can't be made
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
com_Connect(ConnectionId, tcp(Host,Port)) :-
      ground(ConnectionId),
      % if variable address, look it up...
      ((var(Host) ; var(Port)) ->
         com_ResolveVariables([
            [cmd('-oaa_host',Host), cmd('-oaa_port', Port)],
            [env('OAA_HOST', Host), env_int('OAA_PORT', Port)],
            [setup('setup.pl', oaa_host, Host),
             setup('setup.pl',oaa_port, Port)]
          ])
      | true),

      tcp_connect(address(Port, Host), RootConnection),
      assert(com_connection_info(ConnectionId, tcp, client,
            [addr(tcp(Host,Port)),
             oaa_host(Host),oaa_port(Port),connection(RootConnection)],
            connected)).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    com_Disconnect(+ConnectionId)
% purpose: Given a connection ID of type 'client', shuts down the connection.
% remarks: Succeeds silently if there is not an open connection having the
%          given id.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
com_Disconnect(ConnectionId) :-
      ground(ConnectionId),
      com_connection_info(ConnectionId, tcp, client, _Info, connected),
      com_GetInfo(ConnectionId, connection(Connection)),
      tcp_shutdown(Connection),
      retract(com_connection_info(ConnectionId,tcp,client,_Info,connected)),
      !.
com_Disconnect(_ConnectionId).



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    com_ListenAt(+ConnectionId, ?Address)
% purpose: Given a connection ID and an address, initiate a server connection
% remarks:
%  - if Address is a variable, instantiates the Address by using
%     com_ResolveVariables, which looks in a setup file, command line, and
%     environment variables for the required info.
%  - stores the connection info for connection ID in com_connection_info/5.
%  - fails if connection can't be made
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
com_ListenAt(ConnectionId, tcp(Host,Port)) :-
      ground(ConnectionId),
      % if variable address, look it up...
      ((var(Host) ; var(Port)) ->
         com_ResolveVariables([
            [cmd('-oaa_host',Host), cmd('-oaa_port', Port)],
            [env('OAA_HOST', Host), env_int('OAA_PORT', Port)],
            [setup('setup.pl',oaa_host, Host),
             setup('setup.pl',oaa_port, Port)]
          ])
      | true),

      repeat,
      (on_exception(E,
                 tcp_listen_at_port(Port, Host),
                 Exception = E) ->
         ( var(Exception) ->
            assert(com_connection_info(ConnectionId, tcp, server,
               [addr(tcp(Host,Port)),oaa_host(Host),oaa_port(Port)],
               connected)),
            !
         | otherwise ->
            com_ask_about_tcp_exception(Port, Host, Response),
            ( Response == yes ->
                  fail
            | otherwise ->
                  halt
            )
         )
      |
            com_ask_about_tcp_exception(Port, Host, Response),
            ( Response == yes ->
                  fail
            | otherwise ->
                  halt
            )
      ).


com_ask_about_tcp_exception(Port, Host, Response) :-
      repeat,
      with_output_to_chars(
             format('Currently unable to access ~w port ~w.~n  Try again? ~w',
                  [Host, Port, '[y)es, n)o, h)elp]']),
            Chars),
      name(Prompt, Chars),
      ask_oneof(Prompt, [yes, no, help], Response),
      ( Response == help ->
            com_print_tcp_exception_help,
            fail
      | otherwise ->
            !
      ).

com_print_tcp_exception_help :-
    write('
I''ve just attempted to listen on the specified port, but was unable
to gain control of it.  This could be because there''s already a
Facilitator, or some other program, making use of that port.  Or, it
could be that a Facilitator using that port was just terminated.  In
such cases, the port may be inaccessible for a brief period (usually
only a few seconds, but sometimes more).  It may help to kill any
```

```
client agents which may still be connected to the defunct Facilitator.

If you think the specified port may now be accessible, enter "y" and
I''ll try again.  You may request retry any number of times.

If you want me to listen on a different port, enter "n", which will
cause me to terminate.  Then change your port specification (it''s
either in a setup file or an environment variable).  Then restart me.

').



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    com_SendData(+ConnectionId, +Data)
% purpose: Sends data to the specified connection ID
% remarks:
%  - Checks format for destination connection
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
com_SendData(ConnectionId, Data) :-
      ground(ConnectionId),
      ( com_connection_info(ConnectionId, Type, _ClientServer, InfoList,
           connected),
        (Type = tcp ; Type = unknown), !,
        memberchk(connection(Dest), InfoList)
        ;
        format('~nError: cannot find open connection for ~p!~n',
           [ConnectionId]),
        fail
      ),
      ( memberchk(format(F), InfoList) ->
          true
      | memberchk(agent_language(c), InfoList) ->
          F = special_case_c
      | otherwise ->
          F = default
      ),
      !,
      com_send_data_by_format(Dest, F, Data).


% quintus_binary: for inter-quintus communication
com_send_data_by_format(Dest, quintus_binary, Data) :- !,
      tcp_send(Dest, Data).
% prolog: a synonym for quintus_binary
com_send_data_by_format(Dest, prolog, Data) :- !,
      tcp_send(Dest, Data).


% pure_ascii: don't wrap data in term() wrapper
com_send_data_by_format(Dest, pure_ascii, Data) :- !,
        current_output(CurrentOutput),
        flush_output(CurrentOutput),
        tcp_output_stream(Dest, TcpOutput),
        set_output(TcpOutput),
        WriteParams =
           [quoted(true),            % make input acceptable for read
            ignore_ops(false),       % false so list will be printed as '[1,2]'
            % !!! could be a problem with +, other opts.
            numbervars(true),        % print vars as f(A).
            character_escapes(false),% write actual character, not \255
            max_depth(0)],           % no depth limit

        write_term(Data, WriteParams),

        flush_output(TcpOutput),
        set_output(CurrentOutput), !.

% special_case_c: This is the same as default, EXCEPT for the use of
% nl, nl.  See comments within the clause for default format.
% Currently we don't understand why it matters.
com_send_data_by_format(Dest, special_case_c, Data) :- !,
        current_output(CurrentOutput),
        flush_output(CurrentOutput),
        tcp_output_stream(Dest, TcpOutput),
        set_output(TcpOutput),

        WriteParams =
           [quoted(true),            % make input acceptable for read
            ignore_ops(false),       % false so list will be printed as '[1,2]'
            % !!! could be a problem with +, other opts.
            numbervars(true),        % print vars as f(A).
            character_escapes(false),% write actual character, not \255
            max_depth(0)],           % no depth limit

        write_term(term(Data), WriteParams),
        write('.'),
        nl, nl,
        flush_output(TcpOutput),
        set_output(CurrentOutput), !.

% DefaultOAA: wrap in term() wrapper for easy parsing
com_send_data_by_format(Dest, _DefaultOAA, Data) :-
        current_output(CurrentOutput),
        flush_output(CurrentOutput),
        tcp_output_stream(Dest, TcpOutput),
        set_output(TcpOutput),

        WriteParams =
           [quoted(true),            % make input acceptable for read
            ignore_ops(false),       % false so list will be printed as '[1,2]'
            % !!! could be a problem with +, other opts.
            numbervars(true),        % print vars as f(A).
            character_escapes(false),% write actual character, not \255
            max_depth(0)],           % no depth limit

        write_term(term(Data), WriteParams),
        write('.'),
        % nl, nl,
% The preceding does not work between two Quintus agents
% (neither does a single nl, nor does it help to use nl(TcpOutput)),
% so we went to the following.  However, the following does not work
% when a QP facilitator sends to the C interface agent.  For now,
% we'll solve this problem by defining the special_case_c format.
% (DLM, 97-04-09)
      put(TcpOutput, 10),
      % This causes the agents to disconnect (at least under UNIX):
        % put(TcpOutput, 13),

        flush_output(TcpOutput),
        set_output(CurrentOutput), !.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    com_SelectEvent(+TimeOut, -Event)
% purpose: Waits and returns an incoming event, or 'timeout' if TimeOut expires
% remarks:
%  - TimeOut may be a real number, and represents seconds.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
com_SelectEvent(0, Event) :- !,
      on_exception(E,tcp_select(Event), com_print_err(E)).
com_SelectEvent(Seconds, Event) :-
      on_exception(E,tcp_select(Seconds, Event),com_print_err(E)).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    com_print_err
% purpose:  Print error message if problem reading the event
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
com_print_err(E) :-
   format('~n========== READ ERROR !!! ===========~n',[]),
   format('| Messages in this block are rejected~n',[]),
   format('| by the system.~n',[]),
   format('--------------------------------------~n',[]),
   print_message(error, E),
   format('======================================~n',[]), fail.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    com_AddInfo
% purpose:  Adds or changes information about connection
% remarks:
%    Info may be status(S), type(T), protocol(P) or any element (or list
%    of elements) to be stored in InfoList.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
com_AddInfo(ConnectionId, NewInfo) :-
      retract(com_connection_info(ConnectionId, Protocol, Type,
            InfoList, Status)),
      (NewInfo = status(NewStatus), C = true ; NewStatus = Status),
      (NewInfo = protocol(NewProtocol), C = true ; NewProtocol = Protocol),
      (NewInfo = type(NewType), C = true ; NewType = Type),
      (NewInfo = [_H|_T] ->
         union([InfoList, NewInfo], NewInfoList)
       | (ground(C) ; union([InfoList, [NewInfo]], NewInfoList))
      ),
      assert(com_connection_info(ConnectionId, NewProtocol, NewType,
            NewInfoList, NewStatus)), !.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    com_GetInfo
% purpose:  Looks up information about connection
% remarks:
%    Info may be status(S), type(T), protocol(P) or any element stored
%    in InfoList.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
com_GetInfo(ConnectionId, Info) :-
      com_connection_info(ConnectionId, Protocol, Type,
            InfoList, Status),
      (Info = status(Status) ;
       Info = type(Type) ;
       Info = protocol(Protocol) ;
       memberchk(Info, InfoList)),
      !.



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% name:    com_ResolveVariables
% purpose: Tries to instantiate the arguments by looking in the command
%        line arguments, environment variables, and setup files
%  inputs:
%   - VarList:   A list of lists: the first sublist that completely resolves
%        provides the value for com_ResolveVariables.
% remarks:
%    sublists may contain elements in the following format:
%       env(EnvVar, Val)      : looks for "EnvVar" in environment vars
%     env_int(EnvVar, Val)    : Returns value for EnvVar as an integer
%     cmd(CmdVar, Val)  : looks for "CmdVar <Val>" on command line
%     setup(File,SVar, Val)   : reads SVar from setup file File
% example:
%    resolves host and port by searching first commandline, then environment
%    variables, finally reads setup file.
%
%     com_ResolveVariables([
%         [cmd('-oaa_host',Host), cmd('-oaa_port', Port)],
%       [env('OAA_HOST', Host), env_int('OAA_PORT', Port)],
%       [setup('setup.pl',oaa_host, Host),
%        setup('setup.pl',oaa_port, Port)]
%     ])
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
com_ResolveVariables([VarList|_]) :-
      com_resolve_variables(VarList), !.
com_ResolveVariables([_VarList|Rest]) :-
      com_ResolveVariables(Rest).

com_resolve_variables([]).

com_resolve_variables([env_int(EnvVar, Val)|Rest]) :- !,
      environ(EnvVar, EnvAtom),
      name(EnvAtom, EnvChars),
      number_chars(Val, EnvChars),
      com_resolve_variables(Rest).

com_resolve_variables([env(EnvVar, Val)|Rest]) :- !,
      environ(EnvVar, Val),
      com_resolve_variables(Rest).

com_resolve_variables([cmd(CmdVar, Val)|Rest]) :- !,
      % get command line arguments
      unix(argv(ListOfArgs)),
      append(_, [CmdVar, Val|_], ListOfArgs),
      com_resolve_variables(Rest).

com_resolve_variables([setup(File,SVar, Val)|Rest]) :- !,
      % read setup file to load all values
      com_read_setup_file(File),
      Pred =.. [SVar, Val],
        on_exception(_, Pred, fail),
      com_resolve_variables(Rest).



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    com_read_setup_file
% purpose: Finds and loads setup file
% remarks:
%    Always succeeds.
%    The search path for 'setup.pl' is as follows:
%      1. Current directory
%      2. Home directory for user
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
com_read_setup_file(File) :-
      com_already_loaded(File), !.
com_read_setup_file(File) :-
      ( absolute_file_name(File, LocalSetupFile),
        can_open_file(LocalSetupFile, read, fail) ->
          SetupFile = LocalSetupFile
      |
        concat('~/',File, HomeName),
        absolute_file_name(HomeName, UserSetupFile),
          can_open_file(UserSetupFile, read, fail) ->
             SetupFile = UserSetupFile
      ),

      (ground(SetupFile) ->
         format('Loading setup file:~n  ~w~n~n', [SetupFile]),
         ( com_consult(SetupFile, _) ->
             assert(com_already_loaded(File))
         | otherwise ->
             format('~w: A problem was encountered in loading the setup file~n',
                  ['WARNING'])
         )
      | true).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% name:    com_consult(+FilePath, -AbsFileName).
% purpose:
% remarks: We don't use Quintus' builtin consult, because it's too picky
%          about associating predicates with files.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
com_consult(FilePath, AbsFileName) :-
    absolute_file_name(FilePath, AbsFileName),
    can_open_file(AbsFileName, read, fail),
    open(AbsFileName, read, Stream),
    load_clauses(Stream),
    close(Stream).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    load_clauses(+Stream).
% purpose:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load_clauses(Stream) :-
    repeat,
    read_term(Stream, [], Term),
    ( Term = ':-'(_Body) ->
      true
    | Term = end_of_file ->
      true
    | otherwise ->
        load_clause(Term)
    ),
    ( at_end_of_file(Stream) ->
      !
    | otherwise ->
      fail
    ).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    load_clause(+Term).
% purpose:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load_clause(Term) :-
    assert( Term ).
```

APPENDIX A.IV

Source code file named liboaa.pl.

```
%*****************************************************************************
%   File    : liboaa.pl
%   Primary Authors  : Adam Cheyer, David Martin
%   Purpose : Prolog version of library for the Open Agent Architecture
%   Updated : 12/98
%
%   -----------------------------------------------------------------------
%   Unpublished-rights reserved under the copyright laws of the United States.
%


%
%   Unpublished Copyright (c) 1998, SRI International.
%   "Open Agent Architecture" and "OAA" are Trademarks of SRI International.
%   -----------------------------------------------------------------------
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Note: internal functions use the naming convention oaa_function_name(),
%     while public predicates use oaa_PublicPredicate().
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Version 2.0 (change oaa_version assertion)
%   - corrects FromKS in do_events by changing event format to include this
%     info.
%   - messages are only sent to READY agents.  For previous versions, an
%     agent may be either READY or just OPEN.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Version 2.1 (change oaa_version assertion)
%   - triggers have 2 new arguments, OpMask and Template, and
%     more general semantics.  Backwards compatibility is provided.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Version 3.0 (change oaa_version assertion)
%   - primitives changed to start with oaa_ (and _icl) prefixes
%   - Major restructuring and cleanup, including many new capabilities,
%     for first public release (a.k.a. "OAA 2")
%*****************************************************************************

:- module(oaa,
        [icl_GetParamValue/2,
         icl_GetPermValue/2,
         icl_BasicGoal/1,
         icl_GoalComponents/4,
         icl_ConsistentParams/2,
         icl_BuiltIn/1,
         icl_ConvertSolvables/2,
         oaa_LibraryVersion/1,
         oaa_Register/3,
         oaa_RegisterCallback/2,
         oaa_ResolveVariables/1,
         oaa_Ready/1,
         oaa_MainLoop/1,
         oaa_SetTimeout/1,
         oaa_GetEvent/3,
         oaa_ProcessEvent/2,
         oaa_Interpret/2,
         oaa_DelaySolution/1,
         oaa_ReturnDelayedSolutions/2,
         oaa_AddDelayedContextParams/3,
        oaa_PostEvent/2,
        oaa_CanSolve/2,
        oaa_Version/3,
        oaa_Ping/3,
        oaa_Declare/5,
        oaa_DeclareData/3,
        oaa_Undeclare/3,
        oaa_Redeclare/3,
        oaa_AddData/2,
        oaa_RemoveData/2,
        oaa_ReplaceData/3,
        oaa_CheckTriggers/3,
        oaa_AddTrigger/4,
        oaa_RemoveTrigger/4,
        oaa_Solve/2,
        oaa_InCache/2,
        oaa_AddToCache/2,
        oaa_ClearCache/0,
        oaa_TraceMsg/2,
        oaa_ComTraceMsg/2,
        oaa_Inform/3,
        oaa_Id/1,
        oaa_Name/1
       ]).


%*************************************************************************
%* RCS Header and internal version
%*************************************************************************

% rcs version number
rcsid('$Header: /home/trestle4/OAA/src/V2/prolog/RCS/oaa.pl,v 1.127 1998/12/23
23:14:18 martin Exp cheyer $').

:- op(599,yfx,::).

%*************************************************************************
% Include files
%*************************************************************************
:- use_module(library(basics)).
:- use_module(library(read_sent)).
:- use_module(library(lists)).
:- use_module(library(sets)).
:- use_module(library(strings)).
:- use_module(library(files)).
:- use_module(library(environ)).  % read environment vars
:- use_module(library(ctr)).
:- use_module(library(charsio)).  % for sprintf and with_output_to_chars
:- use_module(library(ask)).      % for ask_oneof
:- use_module(library(samsort)).  % for samsort(Ordered,Raw,Sort)
:- use_module(library(date)).     % for now(Time)

:- use_module(library(tcp), [tcp_now/1, tcp_time_plus/3]).


% IMPORTANT: COM module.  We don't want to hard code the name of the
% file that contains module 'com'.  So, when this file is loaded,
% we first check to see if module 'com' is already present, then
% we check to see if the file containing 'com' has been specified
% on the command line, and if neither of those works, we load the
% default file (./com_tcp).
%
% In the case where the module has already been
% loaded, the following seems like the right thing to do:
%     :- use_module(com, _File, all).
% BUT when compiling, this approach results in "undefined" errors from
% qcon.  Thus, for now, in oaa.pl, we are explicitly using com: with all
% calls to the com module.

:- ( current_predicate(_, com:_) ->
       use_module(com, _File, all)
   | unix(argv(ListOfArgs)), append(_, ['-com', File | _], ListOfArgs) ->
       use_module(File, all)
   | otherwise ->
       use_module(com_tcp, all)
   ).


%*****************************************************************************
% Global variables
%*****************************************************************************
:- dynamic
            oaa_already_loaded/1,   % record if file already loaded
                oaa_solvables/1,        % list of agent capabilities
                oaa_trigger/5,          % a built-in solvable
            oaa_trace/1,            % trace mode: on or off
            oaa_com_trace/1,  % com_trace mode: on or off
            oaa_debug/1,            % debug mode: on or off
            oaa_cache/2,            % cached solutions
            oaa_event_buffer/1,     % buffer of waiting events
            oaa_waiting_for/2,      % used for recursive blocking solve
            oaa_waiting_event/1,    %    problem...
            oaa_timeout/1,          % tcp timeout value (use oaa_SetTimeout)
            oaa_delay_table/5,      % table of delayed solutions
            oaa_delay/2,            % the current goal is delayed
            oaa_data_ref/3,         % bookkeeping for 'data' solvables
            oaa_current_contexts/2, % Solve parameters to be propagated
            oaa_callback/2,         % Record of app-specific callbacks
    % These may appear in setup.pl:
                oaa_host/1,             % for root, my host; otherwise,
                                    % host of my parent
                oaa_port/1.             % ... similarly ...



oaa_LibraryVersion(3.0).

     % solvables shared by all agents
     % Note: all built-in DATA solvables must be declared dynamic to avoid
     % QP warnings and exceptions.
oaa_built_in_solvables([
  % @@DLM: If we do away with TriggerId, we could use param
  % unique_values(true).
  solvable(oaa_trigger(_TriggerId, _Type, _Condition, _Action, _Params),
                       [type(data)], [write(true)])
]).

% We'll always have exactly one oaa_solvables fact.  Note that application
% code should NOT include a declaration or clause for oaa_solvables/1.
oaa_solvables([]).



%*****************************************************************************
% Initialization and connection functions
%*****************************************************************************

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_Register
% purpose: Once a comm link is established, either as a client to a Facilitator
%  or as a server for other agents, oaa_Register will setup and registration
%  information for this agent.
%  inputs:
%   - ConnectionId: the symbolic connection Id (client or server connection)
%   - AgentName: the name of the agent
%   - Solvables: solvable list
% remarks:
%   The following information is stored about the current connection,
%   accessible through com_GetInfo(ConnectionId, Info):
%
%     oaa_name(Name)    : the name of the current agent
%     oaa_id(Id)  : the Id for the agent
%       connection(C)   : system-level communications handle
%                           (e.g., socket number)
%
%   if connecting as client, this is also available:
%     fac_id(Id)  : the Facilitator's Id
%     fac_name(Name)    : the Facilitator's name
%     fac_lang(L) : the Facilitator's language
%     fac_version(V)    : the version of the Facilitator's agent library
%
%   In addition, the following predicates are written to parent Facilitator,
%   or locally if the ConnectionId is a server connection:
%
%     agent_host(Id, Name, Host)
%
%   Solvables are also written using oaa_Declare()
%
%   It is possible for an agent to create both server and client connections:
%   such an agent was classified in OAA 1.0 as an agent of class "node"
%   (as opposed to a pure client "leaf" or pure server "root").
%
% examples:
%   % connecting to a Facilitator
%     MySolvables = [do(something)],
%     com_Connect(parent, ConnectionInfo),
%     oaa_Register(parent, my_agent_name, MySolvables).
%
%   % connecting as a Facilitator
%     MySolvables = [],
%     com_ListenAt(incoming, ConnectionInfo),
%     oaa_Register(incoming, root, MySolvables).
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% For client connecting to Facilitator
oaa_Register(ConnectionId, AgentName, Solvables) :-
      % succeeds only if exists an open client connection for ConnectionId
      %    as created by com_Connect()
      com:com_connection_info(ConnectionId, _Protocol, client, _Info,
connected),

      com:com_AddInfo(ConnectionId, oaa_name(AgentName)),

      % FIXED HACK: default now works thanks to update in com_tcp.pl for
      %    the default mode
      % HACK!!! Why doesn't this work right without it?
      % for some reason, when we send the handshaking info in
      %    default mode (instead of quintus_binary), the facilitator's
      %    tcp_select(VerySmallTimeout, Event) doesn't timeout!!!!
      %    So it keeps hanging until some other event (such as disconnect)
      %    arrives.
      com:com_AddInfo(ConnectionId, format(default)),

      % lookupversion number
      oaa_LibraryVersion(Version),

      %%% handshaking with Facilitator -- exchange information...
      % note: for this first communication, no format is defined for the
      %       connection, so it will be sent using default (ascii) format.
      %       Information coming back from Facilitator will update the
      %       format() field for the connection, improving future
      %       communication.
      com:com_SendData(ConnectionId,
            event(ev_connect([oaa_name(AgentName), agent_language(prolog),
             format(quintus_binary), agent_version(Version)]), [])),

      %% Get the connection acknowledgement:
      %  potential bug: what if selected event is NOT from FacId connection?
      oaa_GetEvent(ConnEvent, _Parms, 0),
      ConnEvent = ev_connected(FacInfoList),
      com:com_AddInfo(ConnectionId, FacInfoList),

      oaa_Id(MyId),

        % write host
        ( environ('HOST', MyHost) ->
           oaa_AddData(agent_host(MyId, AgentName, MyHost), [address(parent)])
        | true),

        % Declare solvables (and post to parent facilitator):
        % Note: OK if Solvables = [].
        oaa_Declare(Solvables, [], [], [if_exists(overwrite)], _).

% For Faciliator serving client agents
oaa_Register(ConnectionId, AgentName, Solvables) :-
```

```
      % succeeds only if exists an open client connection for ConnectionId
      %    as created by com_Connect()
      com:com_connection_info(ConnectionId, _Protocol, server, _Info,
connected),

      AgentId = 0,  % A facilitator's ID is always 0
      com:com_AddInfo(ConnectionId, [oaa_id(AgentId),oaa_name(AgentName)]),

      % The fac. records its own agent_data in the same way as its clients'.
        % Note that we can't call oaa_add_data_local until after the solvables
        % have been declared, and we can't declare solvables until we're
        % open - so we have to bootstrap this assertion:
        oaa_assertz(agent_data(AgentId, open, [], AgentName), AgentId, _),

        % Note: OK if Solvables = [].
        oaa_Declare(Solvables, [], [], [if_exists(overwrite)], _),

        % write host
        ( environ('HOST', MyHost) ->
           oaa_add_data_local(agent_host(AgentId, AgentName, MyHost),[])
        | true).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% name:    oaa_ResolveVariables(+VariableList)
% purpose: Tries to instantiate the arguments by looking in the command
%        line arguments, environment variables, and setup files
%  inputs:
%   - VarList:    A list of lists: the first sublist that completely resolves
%        provides the value for oaa_ResolveVariables.
% remarks:
%    sublists may contain elements in the following format:
%       env(EnvVar, Val)       : looks for "EnvVar" in environment vars
%     env_int(EnvVar, Val)     : Returns value for EnvVar as an integer
%     cmd(CmdVar, Val)   : looks for "CmdVar <Val>" on command line
%     setup(SVar, Val)   : reads SVar from setup file
% example:
%    resolves host and port by searching first commandline, then environment
%    variables, finally reads setup file.
%
%     oaa_ResolveVariables([
%         [cmd('-oaa_host',Host), cmd('-oaa_port', Port)],
%       [env('OAA_HOST', Host), env_int('OAA_PORT', Port)],
%       [setup(oaa_host, Host), setup(oaa_port, Port)]
%     ])
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_ResolveVariables([VarList|_]) :-
      oaa_resolve_variables(VarList), !.
oaa_ResolveVariables([_VarList|Rest]) :-
      oaa_ResolveVariables(Rest).

oaa_resolve_variables([]).

oaa_resolve_variables([env_int(EnvVar, Val)|Rest]) :- !,
      environ(EnvVar, EnvAtom),
      name(EnvAtom, EnvChars),
      number_chars(Val, EnvChars),
      oaa_resolve_variables(Rest).

oaa_resolve_variables([env(EnvVar, Val)|Rest]) :- !,
      environ(EnvVar, Val),
      oaa_resolve_variables(Rest).

oaa_resolve_variables([cmd(CmdVar, Val)|Rest]) :- !,
      % get command line arguments
      unix(argv(ListOfArgs)),
      append(_, [CmdVar, Val|_], ListOfArgs),
      oaa_resolve_variables(Rest).

oaa_resolve_variables([setup(SVar, Val)|Rest]) :- !,
      % read setup file to load all values
      oaa_read_setup_file,
      Pred =.. [SVar, Val],
        on_exception(_, Pred, fail),
      oaa_resolve_variables(Rest).



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_read_setup_file
% purpose: Finds and loads setup file
% remarks:
%    Always succeeds.
%    The search path for 'setup.pl' is as follows:
%      1. Current directory
%      2. Home directory for user
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_read_setup_file :-
      oaa_already_loaded(setup), !.
oaa_read_setup_file :-
      ( absolute_file_name('setup.pl', LocalSetupFile),
        can_open_file(LocalSetupFile, read, fail) ->
          SetupFile = LocalSetupFile
      | absolute_file_name('~/setup.pl', UserSetupFile),
          can_open_file(UserSetupFile, read, fail) ->
             SetupFile = UserSetupFile
      ),

      (ground(SetupFile) ->
         format('Loading OAA setup file:~n  ~w~n', [SetupFile]),
         ( oaa_consult(SetupFile, _) ->
             assert(oaa_already_loaded(setup))
         | otherwise ->
             format('~w: A problem was encountered in loading the setup file~n',
                  ['WARNING'])
         )
      | true).
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_Ready
% purpose: Changes the agent's 'open' status to 'ready', indicating that the
%        agent is now ready to receive messages.
% remarks:
%    if requested, prints 'Ready' to standard out.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_Ready(ShouldPrint) :-

    % replaces 'open' status with 'ready'.
    ((\+ oaa_class(root), oaa_Name(MySymbolicName)) ->
      oaa_PostEvent(ev_ready(MySymbolicName), [])
    | true),

    % if ShouldPrint, print ready
    (on_exception(_,ShouldPrint,fail) ->
       format('Ready.~n', [])
    | true).




%*****************************************************************************
% Classifying and Manipulating ICL expressions
%*****************************************************************************

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    icl_BuiltIn(+Goal).
% purpose: Test whether an expression is an ICL built-in goal.
% remarks:
%     - icl_BuiltIn differs significantly from the Quintus Prolog predicate
%       built_in, in that here we do not include basic constructors such
%       as ',' and ';'.
%     - oaa_Interpret/2 must be defined for every goal for which
%       icl_BuiltIn succeeds.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
icl_BuiltIn((_A = _B)).
icl_BuiltIn((_A == _B)).
icl_BuiltIn((_A \== _B)).
icl_BuiltIn((_A =< _B)).
icl_BuiltIn((_A >= _B)).
icl_BuiltIn((_A < _B)).
icl_BuiltIn((_A > _B)).
icl_BuiltIn(member(_,_)).
icl_BuiltIn(memberchk(_,_)).
icl_BuiltIn(findall(_,_,_)).
icl_BuiltIn(icl_ConsistentParams(_,_)).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    icl_BasicGoal(+Goal).
% purpose: Test whether an expression is an ICL basic (non-compound) goal;
%          that is, just a functor with 0 or more arguments.
% remarks:
%     - Basic goals include built-in's as well as solvables.
%     - This is a syntactic test; that is, we're not checking whether the
%       Goal is a declared solvable.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
icl_BasicGoal(Goal) :-
    var(Goal), !, fail.
icl_BasicGoal(Goal) :-
    is_list(Goal), !, fail.
icl_BasicGoal(Goal) :-
    icl_compound_goal(Goal), !, fail.
icl_BasicGoal(Goal) :-
    icl_BuiltIn(Goal),
    !.
icl_BasicGoal(Goal) :-
    Goal =.. [Functor | _],
    atom(Functor).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    icl_compound_goal(+Goal).
% purpose: Test whether an expression is an ICL compound goal.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
icl_compound_goal(_X:_Y).
icl_compound_goal(_X::_Y).
icl_compound_goal((\+ _P)).
icl_compound_goal((_P -> _Q ; _R)).
icl_compound_goal((_P -> _Q)).
icl_compound_goal((_X, _Y)).
icl_compound_goal((_X ; _Y)).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    icl_GoalComponents(+ICLGoal, -A, -G, -P).
%          icl_GoalComponents(-ICLGoal, +A, +G, +P).
%          icl_GoalComponents(+ICLGoal, +A, +G, +P).
% purpose: Assemble, disassemble, or match against the top-level components
%          of an ICL goal.
% remarks:
%     - The top-level structure of an ICL goal is Address:Goal::Params,
%       with Address and Params BOTH OPTIONAL.  Thus, every ICL goal
%       either explicitly or implicitly includes all three components.
%     - This may be used with any ICL goal, basic or compound.
%     - When P is missing, its value is returned or matched as [].  When A is
%       missing, its value is returned or matched as 'unknown'.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The first 4 clauses handled all cases where the ICL Goal is bound;
% the remainder handle those where it is a var.
icl_GoalComponents(A:G::P, Address, Goal, Params) :-
    \+ var(A), \+ var(G), \+ var(P),
    !,
    Address = A, Goal = G, Params = P.
icl_GoalComponents(A:G, Address, Goal, Params) :-
    \+ var(A), \+ var(G),
    !,
    Address = A, Goal = G, Params = [].
icl_GoalComponents(G::P, Address, Goal, Params) :-
    \+ var(G), \+ var(P),
    !,
    Address = unknown, Goal = G, Params = P.
```

```
icl_GoalComponents(G, Address, Goal, Params) :-
    \+ var(G),
    !,
    Address = unknown, Goal = G, Params = [].
icl_GoalComponents(Goal, unknown, Goal, []) :-
    !.
icl_GoalComponents(Address:Goal, Address, Goal, []) :-
    !.
icl_GoalComponents(Goal::Params, unknown, Goal, Params) :-
    !.
icl_GoalComponents(Address:Goal::Params, Address, Goal, Params) :-
    !.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Permissions and parameter lists
%
% These procedures are used in processing solvables permissions, and
% parameter lists of all kinds (including those used with solvables,
% those contained in events, and those used in calls to various
% library procedures).
%
% All permissions and many parameters have default values.
%
% Permissions and parameters lists have a standard form, as defined by
% the predicates below.  To save bandwidth and promote readability, a
% "perm" or "param" list in standard form OMITS default values.  For
% easier processing (e.g., comparing/merging param lists), boolean
% params in standard form always include a single argument 'true' or
% 'false'.
%
% In definitions of solvables and calls to documented library
% procedures, it's OK to include default params in a Params list, if
% desired.  For boolean params, when the intended value is 'true', it's
% OK just to specify the functor, for example, instead of
% cache(true), it's OK just to include 'cache'.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% icl_standardize_perms(+Perms, +KeepDefaults, -Standardized).

icl_standardize_perms([], _KeepDefaults, []).
icl_standardize_perms([Perm | Perms], KeepDefaults, [SPerm | SPerms]) :-
    icl_perm_standard_form(Perm, SPerm),
    ( KeepDefaults ; (\+ icl_perm_default(SPerm)) ),
    !,
    icl_standardize_perms(Perms, KeepDefaults, SPerms).
icl_standardize_perms([_Perm | Perms], KeepDefaults, SPerms) :-
    icl_standardize_perms(Perms, KeepDefaults, SPerms).

icl_perm_standard_form(Perm, SPerm) :-
    atom(Perm),
    !,
    SPerm =.. [Perm, true].
icl_perm_standard_form(Perm, Perm).

icl_perm_default(call(true)).
icl_perm_default(read(false)).
icl_perm_default(write(false)).

% icl_standardize_params(+Params, +KeepDefaults, -Standardized).
%
% Normally there's no need to keep the default value of a param,
% but there are exceptional situations.  If KeepDefaults is true,
% default values are kept.

icl_standardize_params([], _, []).
icl_standardize_params([Param | Rest], KeepDefaults, AllStandardized) :-
    icl_param_standard_form(Param, FullStandardized),
    ( KeepDefaults ->
      Standardized = FullStandardized
    | otherwise ->
      icl_remove_default_params(FullStandardized, Standardized)
    ),
    icl_standardize_params(Rest, KeepDefaults, RestStandardized),
    append(Standardized, RestStandardized, AllStandardized).

% icl_param_standard_form(+Param, -StandardParams).
%
% Maps from an element of a parameter list to a list of elements
% in standardized form.  The parameter list element can be from
% any context (from a call to Solve, AddTrigger, AddData, etc.).

icl_param_standard_form(reply(false), [reply(none)]) :-
    !.
    % broadcast has been retained, as a synonym for reply(none):
icl_param_standard_form(broadcast, [reply(none)]) :-
    !.
icl_param_standard_form(broadcast(true),  [reply(none)]) :-
    !.
icl_param_standard_form(broadcast(false),  [reply(true)]) :-
    !.
icl_param_standard_form(address(Addr),  [address(SAddr)]) :-
    !,
    icl_standardize_address(Addr, SAddr).
icl_param_standard_form(strategy(query), [parallel_ok(true)]) :-
    !.
icl_param_standard_form(strategy(action),
                  [parallel_ok(false), solution_limit(1)]) :-
    !.
icl_param_standard_form(strategy(inform),
                  [parallel_ok(true), reply(none)]) :-
    !.
icl_param_standard_form(callback(Mod:Proc), [callback(Mod:Proc)]) :-
    !.
icl_param_standard_form(callback(Proc), [callback(user:Proc)]) :-
    !.
icl_param_standard_form(Param,  [SParam]) :-
    atom(Param),
    !,
    SParam =.. [Param, true].
icl_param_standard_form(Param, [Param]).

icl_param_default(from(unknown)).
icl_param_default(priority(5)).
icl_param_default(utility(5)).
icl_param_default(if_exists(append)).
icl_param_default(type(procedure)).
icl_param_default(private(false)).
icl_param_default(single_value(false)).
icl_param_default(unique_values(false)).
icl_param_default(rules_ok(false)).
icl_param_default(bookkeeping(true)).
icl_param_default(persistent(false)).
icl_param_default(at_beginning(false)).
icl_param_default(do_all(false)).
icl_param_default(reflexive(true)).
icl_param_default(parallel_ok(true)).
icl_param_default(reply(true)).
icl_param_default(block(true)).
icl_param_default(cache(false)).
icl_param_default(flush_events(false)).
icl_param_default(recurrence(when)).

icl_remove_default_params([], []).
icl_remove_default_params([Param | Rest], Removed) :-
    icl_param_default(Param),
    !,
    icl_remove_default_params(Rest, Removed).
icl_remove_default_params([Param | Rest], [Param | Removed]) :-
    icl_remove_default_params(Rest, Removed).

% icl_GetParamValue(+Param, +ParamList).
%
% Param must have a functor, but its argument(s) can be either ground
% or variables.  E.g., persistent(X).
%
% To get or test the value of a parameter that has a default, it is
% best to call icl_GetParamValue.  For a parameter that has no default,
% you can use icl_GetParamValue OR memberchk.

icl_GetParamValue(Param, ParamList) :-
    predicate_skeleton(Param, Skel),
    memberchk(Skel, ParamList),
    !,
    Skel = Param.
icl_GetParamValue(Param, _ParamList) :-
    predicate_skeleton(Param, Skel),
    icl_param_default(Skel),
    !,
    Skel = Param.

icl_GetPermValue(Perm, PermList) :-
    predicate_skeleton(Perm, Skel),
    memberchk(Skel, PermList),
    !,
    Skel = Perm.
icl_GetPermValue(Perm, _PermList) :-
    predicate_skeleton(Perm, Skel),
    icl_perm_default(Skel),
    !,
    Skel = Perm.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    icl_ConsistentParams(+Test, +ParamList)
% purpose: Often used in solvable declarations to filter on a certain
%       condition.
% definition:
%         Test a param list: if one or more values are given in a parameter
%       list for parameter ParamName, then ParamValue must be defined as
%       one of the values to succeed.  If ParamValue is NOT defined, then
%       icl_ConsistentParams succeeds.
% example:
%    A natural language parser agent can only handle English definitions:
%
%       convert(nl, icl,Input,Params,Output) :-
%          icl_ConsistentParams(language(english),Params).
%
%    if "language(english)" is defined in parameter list of a solve request,
%       the nl agent will receive the request.
%    if "language(spanish)" is defined in the parameter list, the nl agent
%       WILL NOT receive the request.
%    if no language parameter is specified, the request WILL be sent
%    if "language(X)" is specified, the request WILL be sent to the nl agent
% remarks:
%    - Test may contain either a single predicate or a list of test predicates,
%      in which case icl_ConsistentParams will execute all consistency tests.
%    - Interesting note: icl_ConsistentParams() checks consistency as a
%      relation between the two arguments, so it doesn't matter which argument
%      specifies the test list and which the parameters to test.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

icl_ConsistentParams(_TestList, []) :- !.
icl_ConsistentParams([], _ParamList) :- !.
icl_ConsistentParams([Test|RTest], [P1|RParams]) :- !,
   ParamList = [P1|RParams],
   predicate_skeleton(Test, TestWithVars),
   (memberchk(TestWithVars, ParamList) ->
      memberchk(Test, ParamList)
   | true),
   icl_ConsistentParams(RTest, ParamList).
% either Test or Params is NOT a list
icl_ConsistentParams(Test, Param) :-
      (Test = [_|_] ->
         NewTest = Test
      |  NewTest = [Test]),
      (Param = [_|_] ->
         NewParam = Param
      |  NewParam = [Param]),
      icl_ConsistentParams(NewTest, NewParam).



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Agent identity and addressing
%
% Every agent (including facilitators) has a symbolic name, a full address,
% and a local address (or "local ID").  A full address has the form:
%     addr(tcp(Host,Port))              for a facilitator (if TCP is protocol)
%     addr(tcp(Host,Port), LocalID)     for a client agent.
%
% Even though it doesn't appear in the full address, a facilitator also
% has a local ID, for consistency and convenient reference. The
% local ID of a client agent is assigned to it by its facilitator.
% This, and the facilitator's local ID, are passed to the client at
% connection time.
%
% Full addresses are globally unique, and local addresses are unique with
% respect to a facilitator.  Symbolic names are NOT unique in any sense.
%
% The local ID happens to be an integer, but developers should not rely
% on this.
%
% When specifying addresses, in address/1 params for calls to
% oaa_AddData, oaa_Solve, etc., either names or addresses may be used.
% In addition, for convenience, reserved terms 'self', 'parent', and
% 'facilitator' may also be used.
%
% More precisely, the address parameter may contain any of the following:
% a full address; a local ID (when the addressee is known to be either
% the facilitator or a peer client); a name, enclosed in the name/1 functor;
% 'self'; 'parent'; or 'facilitator'.  ('parent' and 'facilitator are
% synonymous.)
%
% Address parameters are standardized as follows: A full address for the
% local facilitator or a peer client is changed to the local ID; all
% other full addresses are left as is.  Names are left as is.  'self',
% 'parent', and 'facilitator' are changed to the appropriate local ID.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % This can only be used AFTER oaa_SetupCommunication has been called,
    % because of the reliance here on com:com_connection_info/5.
icl_standardize_address(Addr, SAddr) :-
    \+ is_list(Addr),
    !,
    icl_standardize_address([Addr], SAddr).
icl_standardize_address([], []).
icl_standardize_address([Addr | Addrs], [SAddr | SAddrs]) :-
    icl_standardize_addressee(Addr, SAddr),
    !,
    icl_standardize_address(Addrs, SAddrs).
icl_standardize_address([_Addr | Addrs], SAddrs) :-
    icl_standardize_address(Addrs, SAddrs).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

icl_standardize_addressee(addr(Addr), ParentId) :-
    com:com_GetInfo(parent, addr(Addr)),
    com:com_GetInfo(parent, fac_id(ParentId)),
    !.
icl_standardize_addressee(addr(Addr), addr(Addr)) :-
    !.
icl_standardize_addressee(addr(Addr, LID), LID) :-
```

```
    com:com_GetInfo(parent, addr(Addr)),
    !.
icl_standardize_addressee(addr(Addr, LID), LID) :-
    com:com_GetInfo(incoming, addr(Addr)),
    !.
icl_standardize_addressee(addr(Addr, LID), addr(Addr, LID)) :-
    !.
icl_standardize_addressee(name(Name), name(Name)) :-
    !,
    icl_name(Name).
icl_standardize_addressee(Name, name(Name)) :-
    icl_name(Name),
    !,
    format('~w (~w): addressee name, in address/1 param, should be specified
as:~n  name(~w)~n',
         ['WARNING', 'liboaa.pl', Name]).
icl_standardize_addressee(Id, TrueId) :-
    icl_true_id(Id, TrueId),
    !.
icl_standardize_addressee(Whatever, _) :-
    format('~w (~w): Illegal addressee, in address/1 param, discarded:~n  ~w~n',
         ['WARNING', 'liboaa.pl', Whatever]),
    fail.

icl_true_id(self, Me) :-
    !,
    oaa_Id(Me).
icl_true_id(parent, Parent) :-
    !,
    com:com_GetInfo(parent, fac_id(Parent)).
icl_true_id(facilitator, Parent) :-
    !,
    com:com_GetInfo(parent, fac_id(Parent)).
icl_true_id(Id, Id) :-
    icl_id(Id).

icl_id(Num) :-
    integer(Num),
    Num >= 0.

icl_name(self) :-
    !, fail.
icl_name(parent) :-
    !, fail.
icl_name(facilitator) :-
    !, fail.
icl_name(Atom) :-
    atom(Atom).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    icl_ConvertSolvables(+ShorthandSolvables, -StandardSolvables).
%          icl_ConvertSolvables(-ShorthandSolvables, +StandardSolvables).
%
% purpose: Convert between shorthand and standard forms of solvables list.
% remarks:
%     - In the standard form, each element is a term solvable(Goal,
%       Params, Permissions), with Permissions and Params both lists.
%     In the Permissions and Params lists, values appear only when they
%       are OTHER than the default.
%     - In the shorthand form, each element can be solvable/3, as above,
%       or solvable(Goal, Params), or solvable(Goal), or just Goal.
%     - Note that "shorthand" means "anything goes" - so shorthand
%       solvables are a superset of standard solvables.
%     - Permissions (defaults in square brackets):
%         call(T_F) [true], read(T_F) [false], write(T_F) [false]
%     - Params (defaults in square brackets):
%         type(Data_Procedure) [procedure],
%         callback(Functor) [no default]
%         utility(N) [5]
%         synonym(SynonymHead, RealHead) [none]
%         rules_ok(T_F) [false],
%         single_value(T_F) [false],
%       unique_values(T_F) [false],
%         private(T_F) [false]
%         bookkeeping(T_F) [true]
%         persistent(T_F) [false]
%     - Refer to Agent Library Reference Manual for details on Permissions
%       and Params.
%     - (@@DLM) This might be the place to check the validity of solvables,
%       such as using only built-ins in tests.  Also, check for dependencies
%       between solvables; e.g., when persistent(false) is there,
%       bookkeeping(true) must also be there.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
icl_ConvertSolvables(ShorthandSolvables, StandardSolvables) :-
    var(StandardSolvables),
    !,
    icl_standardize_solvables(ShorthandSolvables, StandardSolvables).
icl_ConvertSolvables(ShorthandSolvables, StandardSolvables) :-
    icl_readable_solvables(StandardSolvables, ShorthandSolvables).

    % icl_standardize_solvables(+ShorthandSolvables,
    %                                  -StandardSolvables).
icl_standardize_solvables([], []).
icl_standardize_solvables([Shorthand | RestSH], [Standard | RestStan]) :-
    icl_standardize_solvable(Shorthand, Standard),
    icl_standardize_solvables(RestSH, RestStan).

    % icl_standardize_solvable(+Shorthand, -Standard).
icl_standardize_solvable(solvable((Goal :- Test), Params, Perms), Standard) :-
    !,
    append([test(Test)], Params, NewParams),
    icl_standardize_solvable(solvable(Goal, NewParams, Perms), Standard).
icl_standardize_solvable(solvable((Goal :- Test), Params), Standard) :-
    !,
    icl_standardize_solvable(solvable(Goal, [test(Test) | Params], []),
                       Standard).
icl_standardize_solvable(solvable((Goal :- Test)), Standard) :-
    !,
    icl_standardize_solvable(solvable(Goal, [test(Test)], []), Standard).
icl_standardize_solvable((Goal :- Test), Standard) :-
    !,
    icl_standardize_solvable(solvable(Goal, [test(Test)], []), Standard).
icl_standardize_solvable(solvable(Goal, Params, Perms),
                   solvable(Goal, NewParams, NewPerms)) :-
    !,
    icl_standardize_params(Params, false, NewParams),
    icl_standardize_perms(Perms, false, NewPerms).
icl_standardize_solvable(solvable(Goal, Params),
                   solvable(Goal, NewParams, [])) :-
    !,
    icl_standardize_params(Params, false, NewParams).
icl_standardize_solvable(solvable(Goal), solvable(Goal, [], [])) :- !.
icl_standardize_solvable(Goal, solvable(Goal, [], [])) :- !.

    % icl_readable_solvables(+StandardSolvables,
    %                                  -ShorthandSolvables).
    % This is provided for use in "pretty-printing" solvables, in trace
    % messages, etc.
icl_readable_solvables([], []).
icl_readable_solvables([Standard | RestStan], [Shorthand | RestSh]) :-
    icl_readable_solvable(Standard, Shorthand),
    icl_readable_solvables(RestStan, RestSh).

    % icl_readable_solvable(+Standard, -Shorthand).
icl_readable_solvable(solvable(Goal, [], []), Goal) :- !.
icl_readable_solvable(solvable(Goal, Params, []), solvable(Goal, Params)) :- !.
icl_readable_solvable(solvable(Goal, Params, Perms),
                  solvable(Goal, Params, Perms)) :- !.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    icl_minimally_instantiate_solvables(+ShorthandSolvables,
%                                                -MinimalSolvables).
% purpose: Convert from shorthand (or standard form) to minimally instantiated
%          solvables list.
% remarks: - This is special-purpose. It's used to massage a list of solvables
%            that are to be UNdeclared, to make sure each of them will unify
%            with some existing solvable.  Perms and Params are completely
%            ignored in the unification; only the Goal is relevant.  So each
%            minimally instantiated solvable is simply solvable(Goal, _, _).
%          - Note that "shorthand" means "anything goes" - so shorthand
%            solvables are a superset of standard solvables.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % icl_minimally_instantiate_solvables(+ShorthandSolvables,
    %                                      -Solvables).
icl_minimally_instantiate_solvables([], []).
icl_minimally_instantiate_solvables([Shorthand | RestSH],
                            [Minimal | RestMin]) :-
    icl_minimally_instantiate_solvable(Shorthand, Minimal),
    icl_minimally_instantiate_solvables(RestSH, RestMin).

    % icl_minimally_instantiate_solvable(+Shorthand, -Minimal).
icl_minimally_instantiate_solvable(solvable((Goal :- _Test), Params, Perms),
                           Minimal) :-
    !,
    icl_minimally_instantiate_solvable(solvable(Goal, Params, Perms),
                               Minimal).
icl_minimally_instantiate_solvable(solvable((Goal :- _Test), Params),
                           Minimal) :-
    !,
    icl_minimally_instantiate_solvable(solvable(Goal, Params, []), Minimal).
icl_minimally_instantiate_solvable(solvable((Goal :- _Test)), Minimal) :-
    !,
    icl_minimally_instantiate_solvable(solvable(Goal, [], []), Minimal).
icl_minimally_instantiate_solvable((Goal :- _Test), Minimal) :-
    !,
    icl_minimally_instantiate_solvable(solvable(Goal, [], []), Minimal).
icl_minimally_instantiate_solvable(solvable(Goal, _Params, _Perms),
                   solvable(Goal, _, _)) :-
    !.
icl_minimally_instantiate_solvable(solvable(Goal, _Params),
                   solvable(Goal, _, _)) :-
    !.
icl_minimally_instantiate_solvable(solvable(Goal), solvable(Goal, _, _)) :- !.
icl_minimally_instantiate_solvable(Goal, solvable(Goal, _, _)) :- !.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:   oaa_goal_matches_solvables(+Goal, +Solvables,
%                                      -RealGoal, -MatchedSolvable).
% purpose: Determine whether a call to Goal is handled by the agent with
%          these Solvables.
% arguments:
%     - Goal must be non-compound (basic) to match: no address, no params,
%       no subgoals.
%     - Solvables must be in standard form.
%     - RealGoal is what should actually be called, after taking synonyms
%       into account.
%     - MatchedSolvable is the solvable record corresponding to RealGoal.
% remarks:
%   - A solvable's params may contain a single test, but it can
%       be compound:
%       solvable(g(X), [test((X > 1,X < 10))], [...]).
%       Tests should contain only prolog builtins.
%   -  Any solvable can be a synonym of another solvable (including a
%       synonym of a synonym), but eventually there must be a non-synonym
%       solvable.  Synonyms must be used with care.  If predicate A
%       is synonymed to predicate B, there must be a solvable for clause B,
%       for A to be usable.
%   -  When a predicate A is synonymed to predicate B, all other params
%       and all permissions associated with A are ignored.
%   -  Uses would_unify (and \+ \+) so that any variables in the goal are
%       not bound by the solvable, thereby unnecessarily constraining query
%       I forget why: I think it was because we had some problems
%       matching solutions coming back.  However, this has an unusual
%       side effect: if your solvable is t(6) and your query is t(X),
%       the query arrives at the agent as t(X), not t(6), which might
%       be unexpected.  Look into this more someday...
%   -  However, when Goal is a synonym, variables in the synonym param DO
%      get unified correctly.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_goal_matches_solvables(Goal, Solvables, RealGoal, RealMatched) :-
    oaa_built_in_solvables(BuiltIns),
    append(BuiltIns, Solvables, AllSolvables),
    oaa_goal_in_solvables(Goal, AllSolvables, Matched),
    Matched = solvable(_, Params, _),
    % See if Goal is a synonym predicate
    ( icl_GetParamValue(synonym(Goal, SynGoal), Params) ->
        oaa_goal_matches_solvables(SynGoal, Solvables, RealGoal, RealMatched)
    | otherwise ->
        RealGoal = Goal,
      RealMatched = Matched
    ),
    !.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_goal_in_solvables(+Goal, +Solvables, -MatchedSolvable).
% purpose: Determine whether a call to Goal is handled by the agent with
%          these Solvables.
% purpose: Determine whether Goal appears in Solvables, with
%          appropriate Params and Perms for it to be called.
% arguments:
%     - Goal must be non-compound (basic) to match: no address, no params,
%       no subgoals.
%     - Solvables must be in standard form.
% remarks:
%   - Should not be called directly; only by oaa_goal_matches_solvables.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_goal_in_solvables(Goal, [solvable(G1,Params,Perms) | _Rest],
                  solvable(G1,Params,Perms)) :-
      would_unify(Goal, G1),
        icl_GetParamValue(synonym(Goal, _RealGoal), Params),
      !.
oaa_goal_in_solvables(Goal, [solvable(G1,Params,Perms) | _Rest],
                  solvable(G1,Params,Perms)) :-
      would_unify(Goal, G1),
      icl_GetPermValue(call(true), Perms),
      ( icl_GetParamValue(test(T), Params) ->
          \+ \+ oaa_Interpret((Goal = G1, T), [])
      | otherwise ->
          true
      ),
      !.
oaa_goal_in_solvables(Goal, [_|Rest], Matched) :-
      oaa_goal_in_solvables(Goal, Rest, Matched).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_data_matches_solvables(+Clause, +Solvables, +Perm
%                                      -RealClause, -MatchedSolvable).
% purpose: Determine whether Clause can be read or written by the agent with
%          these Solvables, and return the "real" form of the clause that
%          takes synonyms into account.
% arguments:
%     - Clause must be non-compound (basic) to match: no address, no params,
%       no subClauses.
%     - Solvables must be in standard form.
%     _ Perm is 'read' or 'write'.
%     - RealClause is what should actually be used (asserted, retracted,
%       replaced).
%     - MatchedSolvable is the solvable record corresponding to RealClause.
% remarks:
```

```
%     "Writing" means making an assertion.
%     "Reading" is different than "calling".  "Reading" is retrieving the
%       definition clauses of a predicate (including the bodies, if any).
%       Reading is not currently supported by any library procedures.
%     Any solvable can be a synonym of another solvable (including a
%       synonym of a synonym), but eventually there must be a non-synonym
%       solvable.  Synonyms must be used with care.  If predicate A
%       is synonymed to predicate B, there must be a solvable for clause B,
%       for A to be usable.
%     When a predicate A is synonymed to predicate B, all other params
%       and all permissions associated with A are ignored.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_data_matches_solvables(Clause, Solvables, Perm, RealClause, RealMatched) :-
    oaa_built_in_solvables(BuiltIns),
    append(BuiltIns, Solvables, AllSolvables),
    oaa_data_in_solvables(Clause, AllSolvables, Perm, Matched),
    Matched = solvable(_, Params, _),
    ( Clause = (Head :- Body) ->
      true
    | otherwise ->
      Head = Clause
    ),
    % See if Clause is a synonym predicate
    ( icl_GetParamValue(synonym(Head, SynHead), Params) ->
        ( Clause = (Head :- Body) ->
          SynClause = (SynHead :- Body)
      | otherwise ->
          SynClause = SynHead
      ),
        oaa_data_matches_solvables(SynClause, Solvables, Perm,
                                         RealClause, RealMatched)
    | otherwise ->
        RealClause = Clause,
      RealMatched = Matched
    ),
    !.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_data_in_solvables(+Clause, +Solvables, +Perm, -MatchedSolvable).
% purpose: Determine whether (the Head of) Clause appears in Solvables, with
%          appropriate Params and Perms for it to be read or written.
% arguments:
%     - Clause must be non-compound (basic) to match: no address, no params,
%       no subClauses.
%     - Solvables must be in standard form.
% remarks:
%     - Should not be called directly; only by oaa_data_matches_solvables.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_data_in_solvables(Clause, [solvable(G1,Params,Perms) | _Rest], _Perm,
                  solvable(G1,Params,Perms) ) :-
        ( Clause = (Head :- _Body) ->
            true
      | otherwise ->
          Head = Clause
      ),
      would_unify(Head, G1),
        icl_GetParamValue(synonym(Head, _RealHead), Params),
      % @@DLM: OK, so it's a synonym, but shouldn't we check
      % the permissions and type(data) for the referenced solvable?
      !.
oaa_data_in_solvables(Clause, [solvable(G1,Params,Perms) | _Rest], Perm,
                  solvable(G1,Params,Perms) ) :-
        icl_GetParamValue(type(data), Params),
        ( Clause = (Head :- _Body) ->
            icl_GetParamValue( rules_ok(true), Params)
      | otherwise ->
          Head = Clause
      ),
      would_unify(Head, G1),
      ( Perm == write ->
          icl_GetPermValue(write(true), Perms)
      | otherwise ->
          icl_GetPermValue(call(true), Perms)
      ),
      !.
oaa_data_in_solvables(Clause, [_|Rest], Perm, Matched) :-
      oaa_data_in_solvables(Clause, Rest, Perm, Matched).



%******************************************************************************
% Retrieving and managing events
%******************************************************************************

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_MainLoop
% purpose: The main event loop for the application.
%          Reads an event, executes (interprets) it,
%        checks on_receive triggers for the event,
%        checks any application-dependent triggers,
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

oaa_MainLoop(ShouldPrint) :-

    oaa_Ready(ShouldPrint),

    repeat,
       oaa_GetEvent(Event, Params, 0),
       oaa_ProcessEvent(Event, Params),
    fail.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_ProcessEvent
% purpose: Interprets an incoming event
%    - For a timeout, checks task triggers and calls user's idle procedure
%    - Otherwise, oaa_Interprets the event, checks on_receive comm
%      triggers, and then checks task triggers.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_ProcessEvent(timeout, _Params) :- !,
      oaa_CheckTriggers(task, _, _), !,
      oaa_call_callback(app_idle, _, []).
oaa_ProcessEvent(Event, Params) :-
```

```
        ( oaa_Interpret(Event, Params) -> true | true ),
        oaa_CheckTriggers(task, _, _), !.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_SetTimeout
% purpose: Sets the timeout value used by oaa_GetEvent
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_SetTimeout(NSecs) :-
      % Make sure NSecs is valid number
      number(NSecs),
      (NSecs < 0 ->
         TimeOut = 0
      |  TimeOut = NSecs),

      oaa_TraceMsg('~nSetting event timeout to ''~q''.~n', [TimeOut]),
      on_exception(_,retractall(oaa_timeout(_)), true),
      assert(oaa_timeout(TimeOut)).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_GetEvent
% purpose: Return the next event to execute
% remarks:
%    - if a oaa_timeout(Secs) is set to a positive real number by
%      oaa_SetTimeout, wait Secs for an event.
%      If none arrives in this time, return Event = `timeout'
%    - Reads ALL events available on communication stream, sorts the events
%      according to priority, chooses the next event to execute,
%      and then saves the rest for next time oaa_GetEvent is called.
%    - The communication stream is read every time oaa_GetEvent is called, even
%      if there are already saved events (a new one might have a higher
%      priority!)
%    - If saved events exist, return immediately (timeout not considered).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_GetEvent(Event, Params, LowestPriority) :-
      % see if previously saved events to process
      ( retract(oaa_event_buffer(SavedEvents)) ->
          true
      | otherwise ->
          SavedEvents = []
      ),

      % If at least one event can be found with an appropriate priority
      %    from among the saved events, no timeout needed -- flush tcp
      %    buffer, and read_all available
      (oaa_choose_event(LowestPriority, SavedEvents, _OneEvent, _Remainder) ->
         TimeoutSecs = 0.01
      |
         on_exception(_,oaa_timeout(TimeoutSecs),TimeoutSecs=0)
      |
         TimeoutSecs=0
      ),

      oaa_read_all_events(TimeoutSecs, MoreEvents, FlushPriority),

      % if one of the new events has a flush in it, see if it
      %    flushes any of the saved events
      % note: MoreEvents have already been flushed by FlushPriority
      oaa_flush_events(SavedEvents, FlushPriority, RemainingSavedEvents),

      % These are the events we've read so far and haven't executed yet...
      append(RemainingSavedEvents, MoreEvents, EventList),

      (oaa_sort_and_get_event(EventList, LowestPriority, Event, Params) ->
         % we are able to find an appropriate event from list
         % The event will be returned, so fire triggers on it
           oaa_CheckTriggers(comm, event(Event, Params), receive)
      |
         % no good event found, return timeout
         Event = timeout,
         Params = []
      ),
        % This cut is essential to avoid faulty behavior (DLM):
        !.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_sort_and_get_event
% purpose: Sort raw events by priority, choose the highest priority event
%    or FirstIn if equal priority, extract event data and sender,
%    and store the rest of events
% remarks:
%    The chosen event must be of HIGHER priority than LowestPriority, and
%    oaa_sort_and_get_event can fail if no appropriate event is found
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_sort_and_get_event(EventList, LowestPriority, Event, Params) :-
      samsort(oaa_priority_compare, EventList, SortedList),
      oaa_choose_event(LowestPriority, SortedList, RawEvent, Remainder),
      oaa_extract_event(RawEvent, Event, Params),
      (Remainder = [] ;
       assert(oaa_event_buffer(Remainder))),
      !.

oaa_priority_compare(E1, E2) :-
      oaa_extract_event_param(E1, _, priority(P1)),
      oaa_extract_event_param(E2, _, priority(P2)),
      !, P1 >= P2.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_choose_event
% purpose: Extracts the first event from a list which has a HIGHER priority
%        than the required lowest.  Fails if none found.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_choose_event(LowestPriority, [Event|Remainder], Event, Remainder) :-
        oaa_extract_event_param(Event, _, priority(P)),
        LowestPriority < P,
        !.
oaa_choose_event(LowestPriority, [E|Rest], Event, [E|Rest2]) :-
        oaa_choose_event(LowestPriority, Rest, Event, Rest2).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% name:    oaa_read_all_events
% purpose: Flush the communication event queue, reading ALL available events and
%          returning a list of them, or empty list if none available.
% remarks:
%    - Events are retrieved in raw (unextracted) form.
%    - We check to make sure the event is Validated (security hook)
%      before returning it
%    - We check to see if the event is flushed by a later event.
%      If so, we notify event sender of the flush and we don't return the
%      event.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_read_all_events(TimeOut, Events, FlushPriority) :-
      oaa_select_event(TimeOut, E), !,
      (E == timeout ->
         Events = [],
         FlushPriority = 0    % lowest event priority: don't flush events
      |
         % read one event, so read all the rest
         oaa_read_all_events(0.0001, RestEvents, RestFlushPriority),

         % check if read Event is acceptable (security hook)
         (oaa_ValidateEvent(E,OkEvent) ->
            oaa_ComTraceMsg('~n[COM received]:~n   ~q~n', [OkEvent]),

            % get event's priority
            oaa_extract_event_param(OkEvent, _, priority(P)),

            % if less than some higher priority flush event, discard event
            %    and perhaps notify sender
            (P < RestFlushPriority ->
             % event will be removed,
             oaa_flush_notification(OkEvent),
             FlushPriority = RestFlushPriority,
             Events = RestEvents
            |
             % keep event: not flushed
               Events = [OkEvent|RestEvents],

             % see if this event adds a flush:
             %   if so record new flush priority
               (oaa_event_param(OkEvent, flush_events(true)) ->
                FlushPriority = P
             |  FlushPriority = RestFlushPriority)
            )

         % Not validated, skip event
         | Events = RestEvents)
      ).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_ValidateEvent
% purpose: Check that an incoming lowlevel event should be processed.
%          This is the place to put security checks on events.
%    The default behavior defined by the library can be made more
%    stringent by individual agents using the callback oaa_AppValidateEvent
% remarks:
```

```
%    oaa_ValidateEvent has the right to modify the incoming event,
%    or refuse it altogether by failing.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_ValidateEvent(E,OkEvent) :-
      % if oaa_AppValidateEvent is defined, use it.
      predicate_property(user:oaa_AppValidateProperty(_,_), _),
      !,
      user:oaa_AppValidateProperty(E, OkEvent).
% currently, no security checks are performed
oaa_ValidateEvent(OkEvent,OkEvent).



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_flush_events
% purpose: Flushes any events with a lower priority than the FlushPriority
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_flush_events([], _FlushPriority, []).
oaa_flush_events([Event|RestEvents], FlushPriority, RemainingEvents) :-
      oaa_flush_events(RestEvents, FlushPriority, RestSaved),

      % get event's priority
      oaa_extract_event_param(Event, _, priority(P)),

      % if lower priority than we are flushing, notify and remove
      (P < FlushPriority ->
         oaa_flush_notification(Event),
         RemainingEvents = RestSaved
      |
         RemainingEvents = [Event|RestSaved]
      ).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_flush_notification
% purpose: Given a raw event, grabs its real event and looks up whether
%          a notification should be sent out regarding the event's
%        cancellation due to a flush.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_flush_notification(RawEvent) :-
      oaa_extract_event(RawEvent, Event, _Params),
      (oaa_get_flush_notify(Event, NotifyEvent) ->
         oaa_PostEvent(NotifyEvent, [])
      | true), !.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_get_flush_notify
% purpose: Records a list of events which require a return notification
%        if the event is flushed.
% remarks:
%    currently, only the ev_solve() event returns a message;
%    all other events are flushed without notification
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% @@Additional entries needed here:
oaa_get_flush_notify(ev_solve(ID, Goal, Params),
                 ev_solved(ID, FromMe, Goal, Params, [])) :-
      (icl_GetParamValue(reply(none), Params) ->
         fail
      | oaa_Id(FromMe)).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_select_event
% purpose: If a positive timeout is defined, wait N seconds for an event
%    to arrive
%    Otherwise block-wait until an event arrives.
% remarks: IMPORTANT: Connected/1 gets special handling, because we want
%          the connection ID and oaa ID to be assigned immediately.
%          Otherwise, oaa_translate_incoming_event and oaa_unwrap_event
%          won't always work properly for subsequent events from the
%          new connection (or would have to be more complicated).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_select_event(TimeOut, Event) :-
      com:com_SelectEvent(TimeOut, InEvent),
      ( InEvent = connected(_) ->
          oaa_ProcessEvent(InEvent, []),
          oaa_select_event(TimeOut, Event)
      | otherwise ->
          oaa_translate_incoming_event(InEvent, TranslatedEvent),
          oaa_unwrap_event(TranslatedEvent, _Connection, Event)
      ).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_unwrap_event(+TranslatedEvent, -Connection, -Event).
% arguments: TranslatedEvent: An event from another agent, which has already
%            been translated for version compatibility, if necessary.
%            Event: An event term in our standard internal format, as required
%                by all other library procedures.
%            Connection: The CONNECTION of the immediate agent
%                from which this message came (note that an agent's CONNECTION
%                can be different than its ID).
% purpose: Remove an event term from its communications wrapper (if any),
%          and returns it in our standard internal form:
%          'timeout' OR event(Content, Params).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% timeout is the ONLY event that doesn't get embedded in event/2:
oaa_unwrap_event(timeout, unknown, timeout) :-
    !.
oaa_unwrap_event(term(Connection, event(Content,Params)), ConnectionId,
                event(Content, NewParams)) :-
    !,
    ( com:com_GetInfo(ConnectionId, connection(Connection)) ->
      true
    | otherwise ->
      format(
          '~w: incoming event from an unrecognized connection (~w):~n  ~w~n',
        ['INTERNAL ERROR', Connection, event(Content,Params)]),
        ConnectionId = unknown
    ),
    ( memberchk(from(_), Params) ->
        NewParams = [connection_id(ConnectionId) | Params]
    | Content = ev_connected(InfoList),
      memberchk(fac_id(Id), InfoList) ->
        NewParams = [from(Id), connection_id(ConnectionId) | Params]
    | ConnectionId = parent,
      com:com_GetInfo(ConnectionId, fac_id(Id)) ->
        NewParams = [from(Id), connection_id(ConnectionId) | Params]
    | com:com_GetInfo(ConnectionId, oaa_id(Id)) ->
        NewParams = [from(Id), connection_id(ConnectionId) | Params]
    | otherwise ->
      % With current code, this should never happen.  But I can
      % imagine code changes that might need this (DLM 98/02/18):
        NewParams = [from(unknown), connection_id(ConnectionId) | Params]
    ).

% This handles connected/1, end_of_file/1, wakeup/1:
oaa_unwrap_event(Content, unknown, event(Content, [])).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_translate_incoming_event(+InEvent, -OutEvent).
% purpose: Provides backwards compatibility by calling a hook
%    (user:oaa_event_translation/7) that translates incoming events from agents
of
%    other versions.  Also allows for event differences based on language.
%    The idea is to return an event with both format and contents that
%    are appropriate for the agent receiving the event.
% remarks: user:oaa_event_translation/7 can be hard-coded, loaded at runtime,
%    or whatever.  If it's not present, we return the same event.
%    Note that the translation hook is somewhat limited.  It allows a single
%    event to be translated to another single event, and with essentially
%    no information about context.  This inadequate or awkward for some cases.
%    Those cases are handled using extra clauses of user:oaa_AppDoEvent (in
%    translations.pl).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Special cases.  There's no need to translate these.  And, it could be
% problematical, because we don't yet know the language and version of
% the sender.
oaa_translate_incoming_event(term(Conn, event(Contents, Params)),
                       term(Conn, event(Contents, Params))) :-
      ( Contents = ev_connect(_) ;
        Contents = ev_connected(_) ),
    !.

oaa_translate_incoming_event(term(Connection, InEvent),
                       term(Connection, OutEvent)) :-
    current_predicate(oaa_event_translation,
                  user:oaa_event_translation(_,_,_,_,_,_,_)),
    ( com:com_GetInfo(ConnectionId, connection(Connection)) ->
      true
    | otherwise ->
      true
    ),
        % These assumptions may not always be right, but will
      % nearly always get the desired results.
        % :
    ( ground(ConnectionId),
      com:com_GetInfo(ConnectionId, agent_version(PriorVersion)) ->
        true
    | otherwise ->
        PriorVersion = 2.1
    ),
    ( ground(ConnectionId),
      com:com_GetInfo(ConnectionId, agent_language(PriorLanguage)) ->
        true
    | otherwise ->
        PriorLanguage = c
    ),
    oaa_LibraryVersion(MyVersion),
    ( MyVersion \== PriorVersion ; PriorLanguage \== prolog ),
    user:oaa_event_translation(PriorVersion, PriorLanguage, MyVersion, prolog,
                          Connection, InEvent, OutEvent),
    !.
% This handles timeout/0, connected/1, end_of_file/1, wakeup/1.
% Also passes through any event for which there is no translation.
oaa_translate_incoming_event(Event, Event) :-  !.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_extract_event
% purpose: Extract the content and parameters from an event term.
% remarks: Always succeeds.
%          The content part of the term is often (loosely) called the Event.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

oaa_extract_event(event(Content, Params), Content, Params) :-
    !.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_extract_event_param
% purpose:  Extract the content and a parameter value from an event term.
% remarks: Always succeeds - unless you ask for a param that has no default
%          value.
%          The content part of the term is often (loosely) called the Event.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

oaa_extract_event_param(event(Content, Params), Content, Param) :- !,
        icl_GetParamValue(Param, Params).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_event_param
% purpose:  Extract a parameter from an event term.
% remarks:  This FAILS if the parameter isn't present (unlike
%           oaa_extract_event_param).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

oaa_event_param(event(_Content, Params), Param) :- !,
        memberchk(Param, Params).


%*****************************************************************************
% Interpreting EVENTS
%*****************************************************************************
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_Interpret(+ICLExpression, +Params)
% purpose: Executes an incoming event
% remarks: Implements a simple meta-interpreter for executing complex goals.
%          Agent goals are interpreted by oaa_exec_event().
%
%          The contents of Params will vary depending on context.
%        When oaa_Interpret is called on an incoming event, Params
%        will (usually) include from(Sender).  Calls generated internally
%        may contain from(self).  Additional params may
%        accumulate through recursive calls to oaa_Interpret.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_Interpret(Goal, _) :- var(Goal), !, fail.  % How could this happen?
oaa_Interpret(true, _) :- !.
oaa_Interpret(fail, _) :- !, fail.
oaa_Interpret(false, _) :- !, fail.
oaa_Interpret((\+ P), Params) :- !, \+ oaa_Interpret(P, Params).
oaa_Interpret((P -> Q ; _R), Params) :-
    oaa_Interpret(P, Params), !, oaa_Interpret(Q, Params).
oaa_Interpret((_P -> _Q ; R), Params) :- !, oaa_Interpret(R, Params).
oaa_Interpret((P -> Q), Params) :- !, oaa_Interpret((P -> Q ; fail), Params).
oaa_Interpret((X, Y), Params) :- !,
    oaa_Interpret(X, Params), oaa_Interpret(Y, Params).
oaa_Interpret((X ; Y), Params) :- !,
    (oaa_Interpret(X, Params) ; oaa_Interpret(Y, Params)).
oaa_Interpret(findall(Var, Goal, All), Params) :- !,
    findall(Var, oaa_Interpret(Goal, Params), All).
oaa_Interpret(P, _Params) :- icl_BuiltIn(P), !, call(P).
oaa_Interpret(X, Params) :- oaa_exec_event(X, Params).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_exec_event
% purpose: Defines execution of events built into all agents
% remarks: Goals that can't be handled by oaa_exec_event are passed to the
%          user-declared app_do_event callback, if present.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% turn on trace
oaa_exec_event(ev_trace_on, _) :-
      abolish(oaa_trace/1),
      assert(oaa_trace(on)),
      format('~nTrace on.~n', []), !.

% turn off trace
oaa_exec_event(ev_trace_off, _) :-
      abolish(oaa_trace/1),
      assert(oaa_trace(off)),
      format('~nTrace off.~n', []), !.

% tcp level trace
oaa_exec_event(ev_com_trace_on, _) :-
      abolish(oaa_com_trace/1),
      assert(oaa_com_trace(on)),
      format('~nCOMMUNICATION PROTOCOL trace on.~n', []), !.

% tcp level trace
```

```
oaa_exec_event(ev_com_trace_off, _) :-
      abolish(oaa_com_trace/1),
      assert(oaa_com_trace(off)),
      format('~nCOMMUNICATION PROTOCOL trace off.~n', []), !.


% turn on debug
oaa_exec_event(ev_debug_on, _) :-
      abolish(oaa_debug/1),
      assert(oaa_debug(on)),
      format('~nDebug on.~n', []), !.

% turn off debug
oaa_exec_event(ev_debug_off, _) :-
      abolish(oaa_debug/1),
      assert(oaa_debug(off)),
      format('~nDebug off.~n', []), !.

% Set the timeout value
oaa_exec_event(ev_set_timeout(N), _) :-
      abolish(timeout/1),
      assert(timeout(N)),
      format('~nTimeout set to ~q.~n', [N]), !.

% Notification that some other agent has disconnected.  Currently, this applies
% only to peer client agents, and the arg. will always be a local ID.
oaa_exec_event(ev_agent_disconnected(LID), _) :-
        oaa_remove_data_owned_by(LID).

% quit to UNIX
oaa_exec_event(ev_halt, _) :-
      format('~nDisconnecting...~n', []),
      com:com_Disconnect(parent),
      ( oaa_call_callback(app_done, _, []) ; true ),
      halt.

oaa_exec_event(ev_update(ID, Mode, Clause, Params), EvParams) :-
      oaa_Id(AgentId),
      append(Params, EvParams, AllParams),
      ( Mode = add ->
          Functor = oaa_add_data_local
      | Mode = remove ->
          Functor = oaa_remove_data_local
      | Mode = replace ->
          Functor = oaa_replace_data_local
      ),
      Call =.. [Functor, Clause, AllParams],
      ( call(Call) ->
          Updaters = [AgentId]
      | otherwise ->
          Updaters = []
      ),
      (icl_GetParamValue(reply(none), AllParams) -> true |
           oaa_PostEvent(ev_updated(ID, Mode, Clause, Params, Updaters),
                   [])
      ).
```

```
% add or remove a local trigger
oaa_exec_event(ev_update_trigger(ID, Mode, Type,
                         Condition, Action, TrigParams),
             Params) :-
      oaa_Id(AgentId),
      append(TrigParams, Params, NewParams),
      ( Mode == add ->
          Functor = oaa_add_trigger_local
      | Mode == remove ->
          Functor = oaa_remove_trigger_local
      ),
      Call =.. [Functor, Type, Condition, Action, NewParams],
      ( call(Call) ->
          Updaters = [AgentId]
      | otherwise ->
          Updaters = []
      ),
      ( icl_GetParamValue(reply(none), Params) ->
         true
      | otherwise ->
           oaa_PostEvent(ev_trigger_updated(ID, Mode, Type, Condition,
                                 Action, TrigParams, Updaters),
                   [])
      ),
      ( Mode = add ->
            oaa_Inform(trigger, 'trigger_added(~q,~q,~q,~q)~n',
                   [Type, Condition, Action, NewParams])
      | true
      ).

% When asked to solve a goal, see if you know how to solve
% it, then find all solutions.  Send the solutions to the
% caller.
%
% The various params lists must be used with care.  Searching different
% lists may be appropriate for different params, depending on their
% meanings.  Another consideration is that Solve params and Goal params,
% as returned to the requesting agent, must unify with the original
% lists that came from the requesting agent.

oaa_exec_event(ev_solve(ID, FullGoal, SolveParams), Params) :-
      oaa_class(leaf),
        icl_GoalComponents(FullGoal, _, _, GoalParams),

        % More "local" params take precedence, so they go to the
      % beginning of the list:
      append([SolveParams, Params], InheritedParams),
      append([GoalParams, InheritedParams], AllParams),
      % Assert context:
      findall(context(C), member(context(C), AllParams), Contexts),
      asserta( oaa_current_contexts(ID, Contexts) ),

      oaa_TraceMsg('~n~nAttempting to solve:~n  Goal:~q~n  Params:~q~n',
                   [FullGoal, InheritedParams]),
        findall(FullGoal,
                oaa_solve_local(FullGoal, InheritedParams),
            Solutions),
        oaa_TraceMsg('~nSolutions found for ~q:~n   ~q~n',
                     [FullGoal, Solutions]),

      % If user has requested to delay the solution (oaaDelaySolution)
      % save current userId, Goal and Params in delay table, to be
      % sent back in an ev_solved() msg later (oaaReturnDelayedSolutions).

      (retract(oaa_delay(ID, UserId)) ->
         assert(oaa_delay_table(ID, UserId, FullGoal, SolveParams, AllParams))
      |
         (icl_GetParamValue(reply(none), AllParams) -> true |
            (oaa_Id(FromKS) ; FromKS = unknown), !,
              oaa_PostEvent(ev_solved(ID, FromKS, FullGoal, SolveParams,
                                      Solutions),[])
         )
      ),

      % Retract context:
      retractall( oaa_current_contexts(ID, _) ).


% This is for subgoals (of goals passed in solve events) that have
% Params.  Subgoals with no params will fall through to the next clause.
oaa_exec_event(Goal::GoalParams, Params) :-
        oaa_solve_local(Goal::GoalParams, Params).

% call user events.  Must not have a cut, to return all solutions.
oaa_exec_event(Event, Params) :-
      oaa_turn_on_debug,
      ( oaa_solvables(Solvables) -> true | otherwise -> Solvables = []),
      ( (oaa_goal_matches_solvables(Event, Solvables, Goal, Matched),
         Matched = solvable(_, SolvParams, _),
         (icl_GetParamValue(callback(CB), SolvParams) ;
          oaa_callback(app_do_event, CB)))
      ;
        (oaa_callback(app_do_event, CB),
         Goal = Event)
      ),
      !,
      ( CB = Module:Functor ->
          true
      | otherwise ->
          Module = user,
          Functor = CB
      ),
      Call =.. [Functor, Goal, Params],
      on_exception(E,
         Module:Call,
         ( oaa_TraceMsg('WARNING (agent.pl): Exception raised thru callback
handler (~w):~n  ~q~n',
                       [Functor, E]),
           fail )),
      oaa_turn_off_debug.


% What to do about test(TEST)?
% if test(TEST) is listed in arguments, solve
%    it locally.
passes_tests(Params) :-
      oaa_class(leaf),
      icl_GetParamValue(test(Test), Params),
      !,
      oaa_Solve(Test, [level_limit(0)]).
% With compound goals, we also want to allow tests on the facilitator.
% @@DLM: Is this the best way?
passes_tests(Params) :-
      (oaa_class(root);oaa_class(node)),
      icl_GetParamValue(test(Test), Params),
      !,
      oaa_solve_local(Test, []).
passes_tests(_Params) :-
      true.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_DelaySolution
% purpose: Requests that the current AppDoEvent not return solutions to the
%        current goal until a later time.
%  inputs:
%   - Id: an Id which will be used to later match solutions to request
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_DelaySolution(Id) :-
      oaa_current_contexts(GoalId, _Contexts), !,
      assert(oaa_delay(GoalId, Id)).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_ReturnDelayedSolutions
% purpose: Returns the list of solutions for a delayed request
%  inputs:
%   - Id: an Id referring to a previously saved oaa_DelaySolution
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_ReturnDelayedSolutions(Id, SolutionList) :-
      (retract(oaa_delay_table(GoalId, Id, Goal, SolveParams,AllParams)) ->
         (icl_GetParamValue(reply(none), AllParams) -> true |
            (oaa_Id(FromKS) ; FromKS = unknown), !,
            % make sure all Solutions unify with original goal
            findall(Goal, member(Goal,SolutionList), Solutions),
              oaa_PostEvent(ev_solved(GoalId, FromKS, Goal, SolveParams,
                                Solutions),[])
         )
      | true).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_AddDelayedContextParams
% purpose: When a goal is delayed using oaa_DelaySolution(), incoming context
%        parameters from the original request can not be automatically
%        concatenated to outgoing oaa_Solve requests -- since an agent can
%        manage multiple delayed goals at the same time, liboaa doesn't
%        know the correct context for the outgoing oaa_Solve without explicit
%        direction from the programmer.  Hence, an agent programmer who
%        wants to call oaa_Solve during a delayed goal is expected to
%        use this function to add the saved contexts for the delayed goal to
%        his/her outgoing oaa_Solve parameters.
%  inputs:
%   - Id: an Id which will be used to later match solutions to request
%   - Params: Parameters for solve goal
%   - NewParams: Params augmented by saved contexts.
%  example:
%      oaa_AppDoEvent(goal(_X),_Params) :-  oaa_DelayEvent(a_goal).
%      oaa_AppDoEvent(temp_event(Y),_Params) :-
%           oaa_AddDelayedContextParams(a_goal, [], P),
%           oaa_Solve(sub_goal(Y), P).
%      oaa_AppDoEvent(final_event(S), _Params) :-
%           oaa_ReturnDelayedSolutions(a_goal, [goal(S)]).
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_AddDelayedContextParams(Id, Params, NewParams) :-
      retract(oaa_delay_table(_GoalId, Id, _Goal, _SolveParams, AllParams)),
      findall(context(C), member(context(C), AllParams), Contexts),
      append(Contexts, Params, NewParams).




%*****************************************************************************
% Agent-Facilitator communication
%*****************************************************************************

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_PostEvent
% purpose: Sends a low-level event to another agent
% remarks:
%    Should NOT be used before there's a connection established for
%      the destination (such as when a client sends ev_connect to its
%      facilitator).  In such unusual cases, use com_SendData directly.
%      For application developers, this just means don't call
%      oaa_PostEvent until after you've called oaa_Register.
%    Parameters may include:
%      - priority(P):
%      - address(A): specify address of specific server or client agent
%        A must be an agent ID, not a name.  If caller is a client agent,
%        the only meaningful address is that of the client's facilitator.
%      - from(KS): where the event originally originated
%    IMPORTANT: there may be a different address INSIDE the event;
%      these should not be confused!
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_PostEvent(Contents, Params) :-

      % see if any params of interest
        (memberchk(priority(_P), Params);
       memberchk(from(_Agent), Params) ->
          SendEvent = event(Contents, Params)
      |
          SendEvent = event(Contents, [])
      ),

      % find destination: if none, dest = server
      (memberchk(address(Dest), Params) ->
         true
      |
         Dest = parent
      ),

      icl_true_id(Dest, DestId),
        oaa_translate_outgoing_event(SendEvent, DestId, TransEvent),

      oaa_ComTraceMsg('~n[COM send to ~q]:~n   ~q~n', [Dest, TransEvent]),

      oaa_convert_id_to_comm_id(DestId, CommId),
      % send event to destination
      com:com_SendData(CommId, TransEvent),

      % Use SendEvent here, becuase triggers always contain event/2
      % to unify with.
        oaa_CheckTriggers(comm, SendEvent, send).


oaa_convert_id_to_comm_id(Id, CId) :-
      com:com_GetInfo(CId, fac_id(Id)), !.
oaa_convert_id_to_comm_id(Id, CId) :-
      com:com_GetInfo(CId, oaa_id(Id)), !.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_translate_outgoing_event(+Event, +DestId, -NewEvent).
% purpose: Provides backwards compatibility by calling a hook
%    (user:oaa_event_translation/7) that translates outgoing events to agents of
%    other versions.  Also allows for event differences based on language.
% remarks: user:oaa_event_translation/7 can be hard-coded, loaded at runtime,
%    or whatever.  If it's not present, we return the same event.
%    See also comments for oaa_translate_incoming_event.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Special cases.  There's no need to translate these.  And, it could be
% problematical, because we don't yet know the language and version of
% the receiver.  See comments for oaa_unwrap_event.
oaa_translate_outgoing_event(event(Contents, Params), _DestId,
                       event(Contents, Params)) :-
      ( Contents = ev_connect(_) ;
        Contents = ev_connected(_) ),
    !.
oaa_translate_outgoing_event(event(Content, Params), DestId, TransEvent) :-
    current_predicate(oaa_event_translation,
                  user:oaa_event_translation(_,_,_,_,_,_,_)),
        % These assumptions may not always be right, but will
      % nearly always get the desired results:
    com:com_GetInfo(Connection, oaa_id(DestId)),
    ( com:com_GetInfo(Connection, agent_version(DestVersion)) ->
        true
    | otherwise ->
        DestVersion = 2.1
    ),
    ( com:com_GetInfo(Connection, agent_language(DestLanguage)) ->
        true
    | otherwise ->
        DestLanguage = c
    ),
    oaa_LibraryVersion(MyVersion),
    user:oaa_event_translation(MyVersion, prolog, DestVersion, DestLanguage,
                          Connection, event(Content, Params), TransEvent),
    !.
oaa_translate_outgoing_event(Event, _, Event).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_Version
% purpose: Lookup the language and library version number for an agent
% remarks: The default version (if unspecified) is 1.0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

oaa_Version(AgentId, Language, Version) :-
   icl_true_id(AgentId, TrueId),
   % Asking for my version:
   oaa_Id(TrueId),
   Language = prolog,
   oaa_LibraryVersion(Version),
   !.
oaa_Version(AgentId, Language, Version) :-
   icl_true_id(AgentId, TrueId),
   ( com:com_GetInfo(CommId, oaa_id(TrueId)) ;
     com:com_GetInfo(CommId, fac_id(TrueId)) ),
   ( com:com_GetInfo(CommId, agent_language(Language)) ->
       true
   | otherwise ->
       Language = unknown
   ),
   ( com:com_GetInfo(CommId, agent_version(Version)) ->
       true
   | otherwise ->
       Version = 1.0
   ),
   !.
oaa_Version(AgentId, Language, Version) :-
    (oaa_class(leaf) ; oaa_class(node)),
    icl_true_id(AgentId, TrueId),
        % The use of caching here could be dangerous - unless we install a
        % mechanism for automatic updating of the cache.
    oaa_Solve(agent_version(TrueId, Language, Version),
              [address(parent)]),
   !.
oaa_Version(_, prolog, 1.0).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_CanSolve
% purpose: Asks the Facilitator for a list of agents which could solve a Goal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_CanSolve(Goal,KSList) :-
    oaa_Solve(can_solve(Goal, KSList), [address(parent)]).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_Ping
% purpose: Tests whether a given agent is currently responding to requests.
% inputs:
%   AgentAddr: address of agent to test
%   TimeLimit: Time limit (in seconds) for how long to wait for a response
% outputs:
%   TotalResponseTime for round trip (in seconds)
% remarks: Fails if a ping is not returned in TimeLimit amount of time
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_Ping(AgentAddr, TimeLimit, TotalResponseTime) :-
    ground(AgentAddr),
    number(TimeLimit),
    TimeLimit >= 0,
    tcp_now(Before),
    oaa_Solve(true, [address(AgentAddr), time_limit(TimeLimit)]),
    tcp_now(After),
    tcp_time_plus(Before, TotalResponseTimeMs, After),
    TotalResponseTime is TotalResponseTimeMs / 1000.



%*****************************************************************************
% Declaring Solvables
%*****************************************************************************

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_Declare(+Solvables, +CommonPermissions, +CommonParams, +Params,
%                      -DeclaredSolvables)
% purpose: Declare solvables for a client or facilitator, and inform the
%          parent if appropriate.
% arguments:
%   Solvables: A single solvable or a list of solvables, in shorthand or
%          standard form.
%   CommonPermissions: Permissions to be distributed to each solvable in
%          Solvables.  This is purely for programming convenience. See
%          comments for icl_ConvertSolvables for possible values, and
%          solvables documentation for their meanings.
%   CommonParams: Params to be distributed to each solvable in Solvables.
%          This is purely for programming convenience. See comments for
%          icl_ConvertSolvables for possible values, and solvables
%          documentation for their meanings.
%   Params:
%      address(X): Where the solvable will exist.  X may be either 'self'
%          or 'parent' (or the appropriate local ids).  Default: 'self'.
%      if_exists(OverwriteOrAppend): What to do when declaring solvables
%          for self, and some already exist. Default: append.
%   DeclaredSolvables: Returns a list, in standard form, of all solvables
%      successfully declared.
% remarks:
%    - Any agent can declare solvables for itself. In addition, a client can
%      ask its facilitator to declare solvables.  Client-requested facilitator
%      solvables will automatically acquire permission write(true), and params
%      type(data), rules_ok(false), private(false), and bookkeeping(true).
%    - If called by a leaf or node agent, assumes agent is already registered
%      with a parent facilitator.
%    - Predicates can only be declared once.  Changing an existing
%      predicate definition should be done with oaa_Redeclare.  However,
%      a request to declare a predicate, which is already declared in
%      precisely the same way, succeeds transparently.
%    - @@Future params may include 'num_context_args(N)'.
%    - @@Future solvable params may include 'shared'.
%    - synonym predicates can have their own triggers, but share the clause
%      database with their master table.
%    - views and filters, as provided by the OAA V1 DB agent, are not
%      supported as separate params, but the same functionality is available
%      using other params.
%    - @@Do we want client agents to request declarations on other client
%      agents?
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_Declare(Solvable, InitialCommonPerms, InitialCommonParams,
          InitialParams, DeclaredSolvables) :-
    ( is_list(Solvable) ->
      SolvableList = Solvable
    | otherwise ->
      SolvableList = [Solvable]
    ),
    icl_ConvertSolvables(SolvableList, Solvables),
    icl_standardize_perms(InitialCommonPerms, false, CommonPerms),
    icl_standardize_params(InitialCommonParams, false, CommonParams),
    icl_standardize_params(InitialParams, false, Params),
    oaa_distribute_perms(Solvables, CommonPerms, Solvables1),
    oaa_distribute_params(Solvables1, CommonParams, NewSolvables),
    oaa_declare_aux(add, NewSolvables, Params, DeclaredSolvables).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_DeclareData(+Solvables, +Params, -DeclaredSolvables)
% purpose: Declare data solvables for an agent.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_DeclareData(Solv, Params, DeclaredSolvs) :-
    \+ is_list(Solv),
    !,
    oaa_DeclareData([Solv], Params, DeclaredSolvs).
oaa_DeclareData(Solvs, Params, DeclaredSolvs) :-
    % It's only necessary to specify the non-default perms and params.
    CommonPerms = [write(true)],
    CommonParams = [type(data)],
    oaa_Declare(Solvs, CommonPerms, CommonParams, Params, DeclaredSolvs).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_Undeclare(+Solvables, +Params, -UndeclaredSolvables)
% purpose: Remove solvables from a client or facilitator, and inform the
%          parent if appropriate.
% arguments:
%   Solvables: A single solvable or a list of solvables, in shorthand or
%      standard form.  If a solvable is in standard form, however, ONLY
%       the goal is considered in selecting the solvables to be removed
%      (permissions and parameters are ignored).
%   Params:
%      address(X): Where the solvable exists.  X may be either 'self'
%          or 'parent' (or the appropriate local ids).  Default: 'self'.
%   DeclaredSolvables: Returns a list, in standard form, of all solvables
%      successfully removed.
% remarks:
%    - If called by a leaf or node agent, assumes agent is already registered
%      with a parent facilitator.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_Undeclare(Solvable, InitialParams, UndeclaredSolvables) :-
    ( is_list(Solvable) ->
      SolvableList = Solvable
    | otherwise ->
      SolvableList = [Solvable]
    ),
    icl_minimally_instantiate_solvables(SolvableList, Solvables),
    icl_standardize_params(InitialParams, false, Params),
    oaa_declare_aux(remove, Solvables, Params, UndeclaredSolvables).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_Redeclare(+Solvable, +NewSolvable, +Params)
% purpose: Replace a solvable on a client or facilitator, and inform the
%          parent if appropriate.
% arguments:
%   Solvable: A single solvable, in shorthand or standard form.  If in
%      standard form, however, ONLY the goal is considered in selecting
%      the solvable to be replaced (permissions and parameters are ignored).
%   NewSolvable: A single solvable, in shorthand or standard form.
%   Params:
%      address(X): Where the solvable exists.  X may be either 'self'
%          or 'parent' (or the appropriate local ids).  Default: 'self'.
% remarks:
%    - If called by a leaf or node agent, assumes agent is already registered
%      with a parent facilitator.
%    - FAILS if the operation cannot be completed.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_Redeclare(InitialSolvable, InitialNewSolvable, InitialParams) :-
    icl_minimally_instantiate_solvables([InitialSolvable], [Solvable]),
    icl_ConvertSolvables([InitialNewSolvable], [NewSolvable]),
    icl_standardize_params(InitialParams, false, Params),
    oaa_declare_aux(replace, Solvable, [with(NewSolvable) | Params],
                RedeclaredSolvables),
    RedeclaredSolvables \== [].


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_declare_aux(+Mode, +Solvables, +Params, -DeclaredSolvables)
% purpose: Common code for oaa_Declare, oaa_Undeclare, oaa_Redeclare.
% Mode: add, remove, or replace.
% Solvables: for Mode = add, a list of Solvables in standard form.
%            for Mode = remove, a list of Solvables in "minimally instantiated"
%                form.
%            for Mode = replace, a list containing a single Solvable, in
%                "minimally instantiated" form.
% Params: whatever is appropriate for oaa_Declare, _Undeclare, _Redeclare.
%            Must already be in standard form.
% DeclaredSolvables: A list of all solvables successfully added (or removed
%            or replaced), in standard form.
% remarks:
%   A number of params and perms are required when requesting that a
%   parent declare solvables (see comments for oaa_Declare).  We could ensure
%   their presence here, but it's not essential, because the facilitator will
%   enforce this.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Here, a client is asking the facilitator to add, remove, or replace
% solvables.
oaa_declare_aux(Mode, Solvables, Params, DeclaredSolvables) :-
    com:com_GetInfo(parent, fac_id(ParentId)),
    memberchk(address([ParentId]), Params),
    !,
    % Send the request to the Facilitator
    oaa_PostEvent(ev_post_declare(Mode, Solvables, Params), []),
    oaa_poll_until_event(
        ev_reply_declared(Mode, Solvables, Params, DeclaredSolvables)).

% Leaf, node or root adding, removing or replacing its own solvables:
oaa_declare_aux(Mode, Solvables, Params, DeclaredSolvables) :-
    oaa_Id(Me),
    ( memberchk(address(Addr), Params) ->
        Addr = [Me]
    | true),
    !,

    oaa_declare_local(Mode, Solvables, Params, DeclaredSolvables),

    % If I'm a facilitator, I must also "register" my Solvables with myself.
    % (If I'm a node, this will also register them with my parent.)
    ( (\+ oaa_class(leaf), DeclaredSolvables \== []) ->
        oaa_Name(MyName),
          user:oaa_AppDoEvent(
            ev_register_solvables(Mode, DeclaredSolvables, MyName, Params),
          [from(Me)])
    | true
    ),

    % If I'm a leaf, post public solvables to parent facilitator:
    select_elements(DeclaredSolvables, oaa_public_solvable, PublicSolvables),
    ( (oaa_class(leaf), PublicSolvables \== []) ->
      com:com_GetInfo(parent, oaa_name(MyNameC)),
        oaa_PostEvent(
          ev_register_solvables(Mode, PublicSolvables, MyNameC, Params),
          [])
    | true ).

    % Solvable must be in standard form.
oaa_public_solvable(solvable(_Solvable, Params, _Perms)) :-
    icl_GetParamValue(private(false), Params).

    % Solvable must be in standard form.
oaa_data_solvable(solvable(_Solvable, Params, _Perms)) :-
    icl_GetParamValue(type(data), Params).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:      oaa_declare_local(+Mode, +Solvables, +Params, -DeclaredSolvables)
% purpose:   Declare solvables for an agent.
% Mode:      add, remove, or replace.
% Solvables: The form they're in depends on the mode.  See oaa_declare_aux.
% DeclaredSolvables: Returns those members of Solvables for which
%     the operation was successful (more specifically, those that should
%     be passed up to the parent in ev_register_solvables).  Always returned
%     in STANDARD FORM.
% Also see:  comments for oaa_Declare, oaa_Undeclare, oaa_Redeclare.
% remarks:
%   - This performs the local processing needed by calls to oaa_Declare,
%     and by ev_declare events.
%   - Solvables and Params must already be in standard form.
%
%   @@DLM: Could do more careful testing to be sure the solvables are
%   all valid for the requested operation.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_declare_local(Mode, Solvable, Params, DeclaredSolvables) :-
    \+ is_list(Solvable),
    !,
    oaa_declare_local(Mode, [Solvable], Params, DeclaredSolvables).
oaa_declare_local(add, InitialSolvables, Params, DeclaredSolvables) :-
    ( icl_GetParamValue(if_exists(overwrite), Params) ->
      CurrentSolvables = []
    | oaa_solvables(CurrentSolvables) ->
      true
    | CurrentSolvables = []
    ),
    % This will eliminate those that unify with an already declared solvable.
    % @@DLM: Should do more, though: warnings.
    solvables_to_be_added(InitialSolvables, CurrentSolvables,
                                DeclaredSolvables),

    % Make sure Quintus has the correct properties for each DB solvable.
    select_elements(DeclaredSolvables, oaa_data_solvable, DBSolvables),
    oaa_declare_for_prolog(DBSolvables),

    append(CurrentSolvables, DeclaredSolvables, AllSolvables),
    retractall(oaa_solvables(_)),
    assert(oaa_solvables(AllSolvables)).

oaa_declare_local(remove, Solvables, _Params, RemovedSolvables) :-
    % See which ones are really declared:
    ( oaa_solvables(Current) -> true | Current = [] ),
    solvables_to_be_removed(Solvables, Current, RemovedSolvables),
    % Retract all clauses from data solvables:
    select_elements(RemovedSolvables, oaa_data_solvable, DBSolvables),
    oaa_remove_solvables_data(DBSolvables),
    % Assert the new solvables list:
    retractall(oaa_solvables(_)),
    subtract(Current, RemovedSolvables, New),
    assert(oaa_solvables(New)).

oaa_declare_local(replace, [Solvable], Params, [Solvable]) :-
    memberchk(with(NewSolvable), Params),
    % Make sure Solvable is really declared:
    ( oaa_solvables(Current) -> true | otherwise -> Current = []),
    memberchk(Solvable, Current),
    !,
    % If a data solvable, maybe retract all its clauses:
    ( oaa_data_solvable(Solvable) ->
        oaa_remove_solvables_data([Solvable])
    | true
    ),
    % Assert the new solvables list:
    retractall(oaa_solvables(_)),
    replace_element(Solvable, Current, NewSolvable, New),
    assert(oaa_solvables(New)).
oaa_declare_local(replace, [Solvable], _Params, []) :-
    Solvable = solvable(Goal, _, _),
    format('~w: Ignoring attempt to replace a non-existent solvable:~n  ~w~n',
         ['WARNING', Goal]).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name: oaa_distribute_params(+Solvables, +CommonParams, -NewSolvables).
%       oaa_distribute_perms(+Solvables, +CommonPerms, -NewSolvables).
% purpose: Add CommonParams (CommonPerms) to the Params (Permissions) list of
%       each solvable in Solvables.
% Solvables: a solvables list, in standard form.
% remarks: @@Should warn when a solvables has a param that conflicts with
%          CommonParams.  Also, should have an arg that says which version of
%          of the conflicting param to keep.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_distribute_params([], _CommonParams, []).
oaa_distribute_params([Solvable | Solvables], CommonParams,
                  [NewSolvable | NewSolvables]) :-
    Solvable = solvable(Goal, Params, Perms),
    union(Params, CommonParams, NewParams),
    NewSolvable = solvable(Goal, NewParams, Perms),
    oaa_distribute_params(Solvables, CommonParams, NewSolvables).

oaa_distribute_perms([], _CommonPerms, []).
oaa_distribute_perms([Solvable | Solvables], CommonPerms,
                  [NewSolvable | NewSolvables]) :-
    Solvable = solvable(Goal, Params, Perms),
    union(Perms, CommonPerms, NewPerms),
    NewSolvable = solvable(Goal, Params, NewPerms),
    oaa_distribute_perms(Solvables, CommonPerms, NewSolvables).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name: solvables_to_be_added(+ProposedSolvs, +CurrentSolvs, -SolvsToBeAdded).
% purpose: Checks a list of solvables, to make sure they can legally be
%          declared.
% ProposedSolvs: Must be in STANDARD FORM.
% CurrentSolvs: This agent's current solvables.
% SolvsToBeAdded: A subset of ProposedSolvs.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
solvables_to_be_added([], _Current, []).
solvables_to_be_added([Solvable | Solvables], Current, OKSolvables) :-
    Solvable = solvable(Goal, _, _),
    memberchk(solvable(Goal, _, _), Current),
    !,
    format('~w: Ignoring attempt to declare an already existing solvable:~n
~w~n',
         ['WARNING', Goal]),
    solvables_to_be_added(Solvables, Current, OKSolvables).
solvables_to_be_added([Solvable | Solvables], Current,
                      [Solvable | OKSolvables]) :-
    solvables_to_be_added(Solvables, Current, OKSolvables).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name: solvables_to_be_removed(+ProposedSolvs, +CurrentSolvs,
%                               -SolvsToBeRemoved).
% purpose: Checks a list of solvables, to make sure they can legally be
%          UNdeclared.
% ProposedSolvs: Must be in MINIMALLY INSTANTIATED FORM.
% CurrentSolvs: This agent's current solvables.
% SolvsToBeRemoved: A subset of ProposedSolvs, but returned in standard form,
%   fully instantiated.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
solvables_to_be_removed([], _Current, []).
solvables_to_be_removed([Solvable | Solvables], Current,
                 [Solvable | OKSolvables]) :-
    memberchk(Solvable, Current),
    !,
    solvables_to_be_removed(Solvables, Current, OKSolvables).
solvables_to_be_removed([Solvable | Solvables], Current, OKSolvables) :-
    Solvable = solvable(Goal, _, _),
    format('~w: Ignoring attempt to remove a non-existent solvable:~n  ~w~n',
         ['WARNING', Goal]),
    solvables_to_be_removed(Solvables, Current, OKSolvables).



%*****************************************************************************
% Updating Data Solvables
%*****************************************************************************

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_AddData(+Clause, +Params).
% purpose: Add a new clause for a DATA solvable (locally and/or remotely)
% Params:
%    address(X): a list including 'self', 'parent', and/or the
%      addresses of other client agents.  The default (no address)
%      behavior is the same as with oaa_Solve.
%    reflexive(T_F): Save as with oaa_Solve.  Default: true.
%    at_beginning(T_F): if true, uses asserta instead of assertz.
%      Default: false.
%    single_value(T_F): if true, ALL clauses for this predicate are removed
%      before adding the new clause.
%      Default: false.
%    unique_values(T_F): if true, at most one copy of each value is stored.
%      Default: false.
%    owner(LocalId): if bookkeeping(true) for this solvable, record
%      LocalId as the owner.
%      Default: the agent from which the request originated.
%    get_address(X): Returns a list of addresses (ids) of agents that
%      were sent the request.
%    get_satisfiers(X): Returns a list of addresses (ids) of agents that
%      successfully completed the request.
%    reply({true,none}): When data is being added on
%      a remote agent or agents, this tells whether reply message(s) are
%      desired.
%     block(Mode) : true: Block until the reply arrives.
%                 : false: Don't block.  In
%                         this case, the reply events (ev_reply_updated)
%                         can be handled by the user's app_do_event callback
%                 Default: true.  Note that reply(none) overrides
%                         block(true).
% remarks:
%    - Clause is normally a fact (no body), but with Prolog agents, and
%      with rules_ok(true), it's possible for it to have a body.
%    - Triggers will be examined with the on(add) operation mask
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_AddData(Clause, Params) :-
    oaa_update(add, Clause, Params).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_RemoveData(+Clause, +Params).
% purpose: Remove a clause from a DATA solvable (locally and/or remotely)
% Params:
%    address(X): a list including 'self', 'parent', and/or the
%      addresses of other client agents.  The default (no address)
%      behavior is the same as with oaa_Solve and oaa_AddData.
%    reflexive(T_F): Save as with oaa_Solve.  Default: true.
%    do_all(T_F): If true, removes all predicate values that match the Clause
%      Default: false (removes only the first)
%    get_address(X): Returns a list of addresses (ids) of agents that
%      were sent the request.
%    get_satisfiers(X): Returns a list of addresses (ids) of agents that
%      successfully completed the request.
%    owner(LocalId): if bookkeeping(true) for this solvable, remove only
%      data owned by LocalId.
%      Default: ignore owner in removing data.
%    reply({true,none}): When data is being removed on
%      a remote agent or agents, this tells whether reply message(s) are
%      desired.
%     block(Mode) : true: Block until the reply arrives.
%                 : false: Don't block.  In
%                         this case, the reply events (ev_reply_updated)
%                         can be handled by the user's app_do_event callback
%                 Default: true.  Note that reply(none) overrides
%                         block(true).
% remarks:
%    - Clause is normally a fact (no body), but with Prolog agents, and
%      with rules_ok(true), it's possible for it to have a body.
%    - Triggers will be examined with the 'on_Retract' operation mask.
%    - Not for backtracking.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_RemoveData(Clause, Params) :-
    oaa_update(remove, Clause, Params).


%----------------------------------------------------------------------------
% name:    oaa_ReplaceData(+Clause1, +Clause2, +Params).
% purpose: Change a predicate value to a new one
% Clause1: Must be a clause of a writable data solvable.
% Clause2: Must be a clause of a writable data solvable.
% Params:
%    address(X): a list including 'self', 'parent', and/or the
%      addresses of other client agents.  The default (no address)
%      behavior is the same as with oaa_Solve and oaa_AddData.
%    reflexive(T_F): Save as with oaa_Solve.  Default: true.
%    do_all(T_F): If, true, changes all predicate values that match the
%            Clause1 specification
%            default is 'false': changes only the first
%    at_beginning(T_F): If true, uses asserta instead of assertz
%            default is 'false'
%    owner(LocalId): if bookkeeping(true) for this solvable, record
%      LocalId as the owner of each new data item.  Note: It is not possible
%      to specify the owner of the data to be replaced, just that of the
%      NEW data.
%      Default: the agent from which the request originated.
%    get_address(X): Returns a list of addresses (ids) of agents that
%      were sent the request.
%    get_satisfiers(X): Returns a list of addresses (ids) of agents that
%      successfully completed the request.
%    reply({true,none}): When data is being replaced on
%      a remote agent or agents, this tells whether reply message(s) are
%      desired.
%    block(Mode)   : true: Block until the reply arrives.
%                  : false: Don't block.  In
%                          this case, the reply events (ev_reply_updated)
%                          can be handled by the user's app_do_event callback
%                  Default: true.  Note that reply(none) overrides
%                          block(true).
% remarks:
%    - Clause1 and/or Clause2 may be synonym predicates.
%    - Clause1 and Clause2 are not required to have the same functor.
%    - Clause1 and Clause2 may share variables.
%    - Triggers will be examined with the 'remove' operation mask with Clause1,
%      and the 'add' operation mask with Clause2.
%    - db_replace triggers on the Pred2 argument, not on the Pred1 arg
%    - at_beginning param only used if do_all is false
%----------------------------------------------------------------------------
oaa_ReplaceData(Clause1, Clause2, Params) :-
    oaa_update(replace, Clause1, [with(Clause2) | Params]).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:     oaa_update(+Mode, +Clause, +Params).
% purpose:  Common code for oaa_AddData, oaaRemoveData, and oaa_ReplaceData.
% Mode:     add, remove, or replace.
% Clause, Params: May include whatever is appropriate for oaa_AddData,
%                 oaaRemoveData, or oaa_ReplaceData.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_update(Mode, Clause, InitialParams) :-
   icl_standardize_params(InitialParams, false, Params),
   % Is there a specified address?
   ( memberchk(address(Addr), Params) ->
       true
   | otherwise ->
       Addr = []
```

```
),

% Decide whether or not to update locally:
oaa_Id(Me),
( memberchk(Me, Addr) ->
    delete(Addr, Me, NewAddr),
    replace_element(address(Addr), Params, address(NewAddr), Params1),
    Self = true
| otherwise ->
    NewAddr = Addr,
    Params1 = Params
),
( Addr = [], icl_GetParamValue(reflexive(true), Params1) ->
    % do NOT use remove_element here:
    delete(Params1, reflexive(true), Params2),
    ( oaa_solvables(Solvables) -> true | otherwise -> Solvables = [] ),
    ( oaa_data_matches_solvables(Clause, Solvables, write, _, _) ->
        Self = true
    | otherwise ->
        true
    )
| otherwise ->
    Params2 = Params1
),

% Update locally if appropriate:
( Self == true ->
    Requestees1 = [Me],
    ( Mode == add ->
        Functor = oaa_add_data_local
    | Mode == replace ->
        Functor = oaa_replace_data_local
    | Mode == remove ->
        Functor = oaa_remove_data_local
    ),
    LocalCall =.. [Functor, Clause, Params2],
    ( call(LocalCall) ->
        Updaters1 = [Me]
    |   Updaters1 = [])
| otherwise ->
    Requestees1 = [],
    Updaters1 = []
),

% Update remotely if appropriate:
( oaa_class(leaf), (Addr == [] ; NewAddr \== []) ->
    % Send the ev_post_update event to the Facilitator
    oaa_PostEvent(ev_post_update(Mode, Clause, Params2), []),
    % In the return event, Requestee2s lists all agents to whom
    % the update request was sent; Updaters2 lists those who succeeded.
    ( (icl_GetParamValue(reply(asynchronous), Params) ;
       icl_GetParamValue(reply(none), Params)) ->
        Requestees2 = [],
        Updaters2 = []
    | otherwise ->
        oaa_poll_until_event(
          ev_reply_updated(Mode, Clause, Params2, Requestees2, Updaters2))
       )
   | otherwise ->
       Requestees2 = [],
       Updaters2 = []
   ),
   append(Updaters1, Updaters2, Updaters),
   % Return Updaters if requested:
   ( memberchk(get_satisfiers(Updaters), Params) -> true | true ),
   append(Requestees1, Requestees2, Requestees),
   % Return Requestees if requested:
   ( memberchk(get_address(Requestees), Params) -> true | true ).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:      oaa_add_data_local(+Clause, +Params)
% purpose:   Assert a clause for an agent's solvable.
% arguments: See comments for oaa_AddData.
% remarks:
%   This performs the local processing needed for calls to oaa_AddData, and
%   ev_update(add, ...) requests.
%   Application code should not call oaa_add_data_local directly, but rather
%   oaa_AddData with address(self).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_add_data_local(Clause1, Params) :-
    ( oaa_solvables(Solvables) -> true | otherwise -> Solvables = []),
    oaa_data_matches_solvables(Clause1, Solvables, write, Clause, Matched),
    Matched = solvable(Pred, DeclParams, _Perms),
    ( Clause = (Head :- Body) ->
        true
    | otherwise ->
        Head = Clause,
        Body = true
    ),

    append(Params, DeclParams, AllParams),
    % If there's no callback, leave Callback a var:
    ( memberchk(callback(Callback), AllParams) -> true | true ),

    % if single value, erase all old values
    (icl_GetParamValue(single_value(true), AllParams) ->
        ( \+ icl_GetParamValue(bookkeeping(false), DeclParams) ->
          oaa_retractall((Pred :- _), _OldOwner, Callback)
      | otherwise ->
          retract_all((Pred :- _))
      )
    | true),

    % if unique_values(true), make sure fact not already in database
    ( clause(Head, Body), icl_GetParamValue(unique_values(true), AllParams) ->
        true
    | otherwise ->
        ( \+ icl_GetParamValue(bookkeeping(false), DeclParams) ->
          oaa_data_owner(Params, Owner),
            ( icl_GetParamValue(at_beginning(true), AllParams) ->
                oaa_asserta(Clause, Owner, Callback)
            |
                oaa_assertz(Clause, Owner, Callback)
            )
        | otherwise ->
            ( icl_GetParamValue(at_beginning(true), AllParams) ->
                asserta(Clause)
            |
                assertz(Clause)
            )
        )
    ),
    oaa_CheckTriggers(data, Head, add),
    !.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:      oaa_remove_data_local(+Clause, +Params)
% purpose:   Retract a clause (or all clauses) from an agent's solvable.
% arguments: See comments for oaaRemoveData.
% remarks:
%   This performs the local processing needed for calls to oaaRemoveData, and
%   ev_update(remove, ...) requests.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_remove_data_local(Clause1, Params) :-
    ( oaa_solvables(Solvables) -> true | otherwise -> Solvables = []),
    oaa_data_matches_solvables(Clause1, Solvables, write, Clause, Matched),
    Matched = solvable(_Pred, DeclParams, _Perms),
    ( Clause = (Head :- Body) ->
        true
    | otherwise ->
        Head = Clause,
        Body = true
    ),
    append(Params, DeclParams, AllParams),
    ( memberchk(callback(Callback), AllParams) -> true | true ),

    ( \+ icl_GetParamValue(bookkeeping(false), DeclParams) ->
      ( icl_GetParamValue(owner(Owner), Params) -> true | true ),
        ( icl_GetParamValue(do_all(true), Params) ->
            oaa_retractall(Clause, Owner, Callback)
      | otherwise ->
          oaa_retract(Clause, Owner, Callback)
      )
    | otherwise ->
        ( icl_GetParamValue(do_all(true), Params) ->
            retract_all(Clause)
      | otherwise ->
            retract(Clause)
      )
    ),

    oaa_CheckTriggers(data, Head, remove),
    !.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:      oaa_replace_data_local(+Clause1, +Params)
% purpose:   Replace one or more clauses from an agent's solvable.
% arguments: See comments for oaa_ReplaceData.
% remarks:
%   This performs the local processing needed for calls to oaa_ReplaceData, and
%   ev_update(replace, ...) requests.
%   Clause1 is the thing to be replaced.  The thing to replace it with must
%   be present in Params, as with(Clause2).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_replace_data_local(Clause1In, Params) :-
    memberchk(with(Clause2In), Params),
    ( oaa_solvables(Solvables) -> true | otherwise -> Solvables = []),
    oaa_data_matches_solvables(Clause1In, Solvables, write, Clause1, Matched),
    oaa_data_matches_solvables(Clause2In, Solvables, write, Clause2, _Matched2),
    Matched = solvable(_Pred, DeclParams, _Perms),
    ( Clause1 = (Head :- Body) ->
        true
    | otherwise ->
        Head = Clause1,
        Body = true
    ),

    append(Params, DeclParams, AllParams),
    ( memberchk(callback(Callback), AllParams) -> true | true ),

    % do replace of either one or all occurrences
    ( \+ icl_GetParamValue(bookkeeping(false), DeclParams) ->
        oaa_data_owner(Params, Owner),
        ( icl_GetParamValue(do_all(true), Params) ->
            oaa_replace_all(Clause1, Clause2, Owner, Callback)
        | otherwise ->
            oaa_retract(Clause1, _OldOwner, Callback),
            (icl_GetParamValue(at_beginning(true), AllParams) ->
               oaa_asserta(Clause2, Owner, Callback)
          |  oaa_assertz(Clause2, Owner, Callback)
          )
      )
    | otherwise ->
        ( icl_GetParamValue(do_all(true), Params) ->
            replace_all(Clause1, Clause2)
      | otherwise ->
            retract(Clause1),
            (icl_GetParamValue(at_beginning(true), AllParams) ->
               asserta(Clause2)
          |  assertz(Clause2)
          )
      )
    ),
    oaa_CheckTriggers(data, Clause1, remove),
    oaa_CheckTriggers(data, Clause2, add),
    !.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    retract_all
% purpose: Remove all clauses matching Clause1
% remarks: Always succeeds.  Needed because retractall((func(X) :- Y)) doesn't
%          work.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
retract_all(Clause1) :-
   retract(Clause1),
   fail.
retract_all(_Clause1).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    replace_all
% purpose: Replace all clauses matching Clause1 by Clause2
% remarks: Always succeeds
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
replace_all(Clause1, Clause2) :-
   retract(Clause1),
   assert(Clause2),
   fail.
replace_all(_Clause1, _Clause2).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name: oaa_data_owner(+Params, -Owner)
% purpose: Determine data ownership from the available params
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_data_owner(Params, Owner) :-
    ( memberchk(owner(Owner), Params) ->
        true
    | memberchk(from(Owner), Params) ->
      true
    | oaa_Id(Owner) ->
      true
    | otherwise ->
      Owner = unknown
    ).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name: oaa_Id(MyId)
% purpose: Return the Id of the current agent
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% if connected to a Facilitator, use this Id
oaa_Id(MyId) :-
      com:com_GetInfo(parent, oaa_id(MyId)), !.
% For root, get any id
oaa_Id(MyId) :-
      com:com_GetInfo(ConnectionId, type(server)),
      com:com_GetInfo(ConnectionId, oaa_id(MyId)), !.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name: oaa_Name(MyName)
% purpose: Return the name of the current agent
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% if connected to a Facilitator, use this Id
oaa_Name(MyName) :-
      com:com_GetInfo(parent, oaa_name(MyName)), !.
% For root, get any id
oaa_Name(MyName) :-
      com:com_GetInfo(ConnectionId, type(server)),
      com:com_GetInfo(ConnectionId, oaa_name(MyName)), !.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name: oaa_class(MyClass)
% purpose: Return the class (leaf, node, root) of the current agent
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% if connected to a Facilitator, use this Id
oaa_class(leaf) :-
      com:com_GetInfo(_, type(client)),
      \+ com:com_GetInfo(_, type(server)), !.
oaa_class(node) :-
      com:com_GetInfo(_, type(client)),
      com:com_GetInfo(_, type(server)), !.
oaa_class(root) :-
      com:com_GetInfo(_, type(server)),
      \+ com:com_GetInfo(_, type(client)), !.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name: oaa_asserta(Clause, Owner, SpecifiedCallback)
%       oaa_assertz(Clause, Owner, SpecifiedCallback)
%       oaa_retract(Clause, Owner, SpecifiedCallback)
%       oaa_retractall(Clause, Owner, SpecifiedCallback)
%       oaa_replace_all(Clause1, Clause2, Owner, SpecifiedCallback)
% purpose: Perform data updates with bookkeeping info (in oaa_data_ref/3)
% remarks: These should only be used with data solvables having param
%          bookkeeping(true).
%          There are still a couple limitations related to data callbacks.
%          First, callbacks don't work when bookkeeping(false).
%          Second, oaa_replace_all assumes the same callback is appropriate
%          for both the old and new facts.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_asserta(Clause, Owner, Callback) :-
    asserta(Clause, Ref),
    now(Time),
    assert(oaa_data_ref(Ref, Owner, Time)),
    oaa_call_callback(app_on_data_change, Callback, [add(Clause)]).

oaa_assertz(Clause, Owner, Callback) :-
    assertz(Clause, Ref),
    now(Time),
    assert(oaa_data_ref(Ref, Owner, Time)),
    oaa_call_callback(app_on_data_change, Callback, [add(Clause)]).

oaa_retract(Clause, Owner, Callback) :-
    ( Clause = (Head :- Body) ->
        true
    | otherwise ->
        Head = Clause,
        Body = true
    ),
    clause(Head, Body, Ref),
    ( retract(oaa_data_ref(Ref, Owner, _)) ->
        erase(Ref),
      oaa_call_callback(app_on_data_change, Callback, [remove(Clause)])
    ).

oaa_retractall(Clause, Owner, Callback) :-
    ( Clause = (Head :- Body) ->
        true
    | otherwise ->
```

```
        Head = Clause,
        Body = true
    ),
    clause(Head, Body, Ref),
    ( retract(oaa_data_ref(Ref, Owner, _)) ->
        erase(Ref),
      oaa_call_callback(app_on_data_change, Callback, [remove(Clause)])
    ),
    fail.
oaa_retractall(_Clause, _Owner, _Callback).

oaa_replace_all(Clause1, Clause2, Owner, Callback) :-
   oaa_retract(Clause1, _OldOwner, Callback),
   oaa_assertz(Clause2, Owner, Callback),
   % This would be redundant:
   % oaa_call_callback(app_on_data_change, Callback, [replace(Clause1,
Clause2)]),
   fail.
oaa_replace_all(_Clause1, _Clause2, _Owner, _Callback).



%*****************************************************************************
% Trigger Handling
%*****************************************************************************

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_CheckTriggers
% purpose: Given a trigger type, a mask and an Op (e.g. [send, receive],
%    [add, remove], etc), see if any triggers fire.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_CheckTriggers(Type, Condition, Op) :-
      % for each matching trigger
      oaa_solve_local(
            oaa_trigger(TriggerId, Type, Condition, Action, Params),
          []),

      ( (Type == task, \+ var(Condition)) ->
         % We don't want this to succeed more than once, so use ->
         ( oaa_Interpret(Condition, [from(self)]) -> true )
      | otherwise ->
          true
      ),

      % see if on(Op) has been specified
      (memberchk(on(OpSpecified), Params) ->
         OpMask = OpSpecified
      |  OpMask = _),

      % see if Op is OK
      ( (ground(OpMask), OpMask = [_|_]) ->
          memberchk(Op, OpMask)
      | otherwise ->
            Op = OpMask
      ),

      % test additional conditions
```

```
        (memberchk(test(Test), Params) ->
           % We don't want this to succeed more than once, so use ->
           ( oaa_Interpret(Test, [from(self)]) -> true )
        |  Test = 'true'),

        % check recurrence: remove trigger?
        (remove_element(recurrence(R), Params, NewParams) ->
           (R = whenever ->
              true        % don't remove trigger if 'whenever'
           | integer(R), R > 1 ->
              R2 is R - 1,
              % decrement recurrence count
              oaa_remove_data_local(
                  oaa_trigger(TriggerId, Type, Condition, Action, Params),
                []),
              oaa_add_data_local(
                    oaa_trigger(TriggerId, Type, Condition, Action,
                          [recurrence(R2)|NewParams]),
                 [])
           | oaa_remove_local_trigger_by_id(TriggerId)
           )

        |
           R = when,
           oaa_remove_local_trigger_by_id(TriggerId)
        ),

        oaa_TraceMsg(
         '~n~q trigger fired (~q): ~q AND ~q,~n  Action: ~q~n',
           [Type, Op, Cond, Test, Action]),

        (Type \== comm ->
           oaa_Inform(trigger,
              'trigger_fired(~q,~q,~q,~q)~n',
               [Type, Cond, Action, Params])
        | true),

        % FIRE!!!!
          oaa_fire_trigger(Action),

        % loop back for more triggers
        fail.
oaa_CheckTriggers(_Type, _Cond, _Op).

oaa_fire_trigger(oaa_Solve(Goal, Params)) :-
    !,
    ( memberchk(block(_), Params) ->
      NewParams = Params
    | otherwise ->
      append([block(false)], Params, NewParams)
    ),
    oaa_Solve(Goal, NewParams).
oaa_fire_trigger(oaa_Solve(Goal)) :-
    !,
    oaa_Solve(Goal, [block(false)]).
oaa_fire_trigger(oaa_Interpret(Goal, Params)) :-
    !,
    ( memberchk(from(_), Params) ->
      NewParams = Params
    | otherwise ->
      oaa_Id(Me),
      append([from(Me)], Params, NewParams)
    ),
    oaa_Interpret(Goal, NewParams).
oaa_fire_trigger(oaa_Interpret(Goal)) :-
    !,
    oaa_Id(Me),
    oaa_Interpret(Goal, [from(Me)]).
oaa_fire_trigger(Goal) :-
    oaa_Id(Me),
    oaa_Interpret(Goal, [from(Me)]).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_AddTrigger
% purpose: Adds a trigger according to parameters
% Type     = comm, data, task, time
% Condition= comm:event to match,  data:data to match, task:solvable to call
%           time:@@
% Action   = Can be any of these:
%              oaa_Solve(Goal, Params)
%              oaa_Interpret(Goal, Params)
%              Goal [passed to oaa_Interpret with default params]
% Params   =
%   address(X): a list including 'self', 'parent', and/or the
%     addresses of other client agents.  Default: see below.
%   test(T): additional tests before trigger will fire [@@needs work?]
%   on(OP) : operation check: on(add), on(remove), on(receive), etc.
%   recurrence(R): when, whenever, or integer (# of times to execute)
%   reply({true,none}): When a trigger is being added on
%     a remote agent or agents, this tells whether reply message(s) are
%     desired.
%   block(Mode) : true: Block until the reply arrives.
%               : false: Don't block.  In
%                        this case, the reply events
%                         can be handled by the user's app_do_event callback
%               Default: true.  Note that reply(none) overrides
%                        block(true).
%   get_address(X): Returns a list of addresses (ids) of agents that
%     were sent the request.
%   get_satisfiers(X): Returns a list of addresses (ids) of agents that
%     successfully completed the request.
%
% Default destination for triggers:
%    Data triggers: all agents with solvables matching the Condition
%        field.
%    All other types: the local agent
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_AddTrigger(Type, Condition, Action, InitialParams) :-
   oaa_update_trigger(add, Type, Condition, Action, InitialParams).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_RemoveTrigger
% purpose: Removes a trigger from a local or remote agent
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_RemoveTrigger(Type,Condition,Action,Params) :-
   oaa_update_trigger(remove, Type, Condition, Action, Params).

oaa_update_trigger(Mode, Type, InCondition, Action, InParams) :-
   ( (Type == comm, \+ InCondition = event(_,_)) ->
       Condition = event(InCondition, _)
   | otherwise ->
       Condition = InCondition
   ),
   icl_standardize_params(InParams, false, Params),
   % Is there a specified address?
   ( memberchk(address(Addr), Params) ->
       true
   | otherwise ->
       Addr = []
   ),

   % Decide whether or not to update locally:
   oaa_Id(Me),
   ( Addr \== [], memberchk(Me, Addr) ->
       delete(Addr, Me, NewAddr),
       replace_element(address(Addr), Params, address(NewAddr), Params1),
       Self = true
   | Addr = [], Type == data, icl_GetParamValue(reflexive(true), Params) ->
       % Do NOT use remove_element here:
       delete(Params, reflexive(true), Params1),
       NewAddr = Addr,
       Self = true
   | Addr = [], Type \== data ->
       NewAddr = Addr,
       Params1 = Params,
       Self = true
   | otherwise ->
       NewAddr = Addr,
       Params1 = Params
   ),

   % Update locally if appropriate:
   ( Self == true ->
       Requestees1 = [Me],
       ( Type == add ->
           Functor = oaa_add_trigger_local
       | otherwise ->
           Functor = oaa_remove_trigger_local
       ),
       LocalCall =.. [Functor, Type, Condition, Action, Params1],
       ( call(LocalCall) ->
           Updaters1 = [Me]
       |   Updaters1 = [])
   | otherwise ->
       Requestees1 = [],
       Updaters1 = []
   ),

   % Update remotely if appropriate:
```

```
   ( oaa_class(leaf), ((Addr == [], Type = data) ; NewAddr \== []) ->
       % Send the request event to the Facilitator
       oaa_PostEvent(
         ev_post_trigger_update(Mode,Type,Condition,Action,Params1), []),
       ( (icl_GetParamValue(reply(asynchronous), Params) ;
          icl_GetParamValue(reply(none), Params)) ->
           Requestees2 = [],
           Updaters2 = []
       | otherwise ->
           % In the return event, Requestees lists all agents to whom
           % the update request was sent; Updaters2 lists those who succeeded.
           oaa_poll_until_event(
             ev_reply_trigger_updated(Mode, Type, Condition, Action, Params1,
                                    Requestees2, Updaters2))
       )
   | otherwise ->
       Requestees2 = [],
       Updaters2 = []
   ),
   append(Updaters1, Updaters2, Updaters),
   % Return Updaters if requested:
   ( memberchk(get_satisfiers(Updaters), Params) -> true | true ),
   append(Requestees1, Requestees2, Requestees),
   % Return Requestees if requested:
   ( memberchk(get_address(Requestees), Params) -> true | true ).

oaa_add_trigger_local(Type, Condition, Action, Params) :-
    gensym(trg, TriggerId),
    oaa_add_data_local(
        oaa_trigger(TriggerId, Type, Condition, Action, Params),
      []).

oaa_remove_trigger_local(Type, Condition, Action, Params) :-
    oaa_remove_data_local(
        oaa_trigger(_TriggerId, Type, Condition, Action, Params),
      []).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_remove_local_trigger_by_id
% purpose: Removes a local trigger given its unique identifier
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_remove_local_trigger_by_id(TriggerId) :-
      oaa_remove_data_local(oaa_trigger(TriggerId, _,_,_,_), []),
      !.



%*****************************************************************************
% Requesting Services
%*****************************************************************************

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_Solve
% purpose: Sends work or information requests to distributed agents, brokered
%        by the Facilitator agent
```

```
%
%   The default behavior (paramlist = []) is to act like the Prolog primitive
%   call(Goal), blocking until Goal is finished, and unifying and backtracking
%   over solutions for Goal.
%
%   This behavior may be modified by a parameter list, which may contain:
%
%       cache(T_F)        : cache all solutions locally, and if good solutions
%                      already exist in the cache, use the local values
%                      instead of making a distributed request.
%                            Default: false.
%       level_limit(N)  : highest number of hierarchical levels to climb for
%                            solutions.
%       address(AgentId): send request to specific agent, given its name or Addr
%                   If AgentID is 'self', solves the goal locally
%       reply(Mode)     : true: Reply desired.
%                       : none: No reply desired.
%                       Default: true, except when the call to oaa_Solve
%                         is a trigger action, in which case it is
%                         none.  'none' is used here instead of false,
%                         because we anticipate some additional values.
%       block(Mode)     : true: Block until the reply arrives.
%                       : false: Don't block.  In
%                         this case, the reply events (ev_reply_solved)
%                         can be handled by the user's app_do_event callback
%                       Default: true, except when the call to oaa_Solve
%                         is a trigger action, in which case it is
%                         false.  Note that reply(none) overrides
%                         block(true).
%       solution_limit(N)
%                 : limits the maximum number of solutions found to N
%       time_limit(N)   : Waits a maximum of N seconds before returning
%                      (failure if no solution found in time).
%       context(C)      : Passes a context value through any subsequent
%                   solves.
%       parallel_ok(T_F): if T_F is 'true' (default), multiple agents
%                   that can solve the Goal will attempt to work on it
%                   in parallel.  If 'false', one agent will be selected
%                   at a time to solve the goal, until the maximum
%                   number of requested solutions (see solution_limit) is
%                   found.
%     reflexive(T_F)
%                 : If T_F is `true', the Facilitator will consider the
%                      originating agent when choosing agents to solve a
%                      request.  Default: true.
%     priority(P) :  P ranges from 1 (low priority) to 10 (high priority)
%                            with a default of 5.
%       flush_events(T_F)
%                       : Will flush (dispose of) all events of lower priority
%                   currently queued at the destination agent.  These
%                   events are lost, and will not be executed.
%                   This parameter should be used with caution!!!
%                         Default: false.
%       get_address(X)  : Returns a list of addresses (ids) of agents that
%                         were asked to solve the goal, or one of its subgoals
%       get_satisfiers(X)
%                       : Returns a list of addresses (ids) of agents that
%                         succeeded in solving the goal, or one of its
%                         subgoals.
%
%       strategy(S)     : Shorthand for certain combinations of the above
%                         parameters.  S is one of
%                           query = [parallel_ok(true)]
%                           action = [parallel_ok(false), solution_limit(1)]
%                           inform = [parallel_ok(true), reply(none)]
%
%   Remarks: Note that certain combinations of parameters are inconsistent,
%   and are handled as follows:
%       reply(none) overrides block(true)
%       reply(none) overrides parallel_ok(false)
%
%   All of the above parameters may be used in the "global" parameter
%   list (the second argument to oaa_Solve), when Goal is non-compound.
%   Most can be used in the global list with compound goals also.
%   Some of these parameters can also be used in the NESTED parameter
%   lists of compound goals.  Uses of these parameters with compound
%   goals are documented elsewhere.  When that documentation exists,
%   this will go there:
%   With many compound goals, however, the get_satisfier/1 parameter isn't
%   really meaningful.  Thus, with compound goals, it is often best to use
%   this parameter in a nested parameter list.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_Solve(Goal, InitialParams) :-
      % Trace message
      oaa_TraceMsg('~n~nStarting oaa_Solve request:~n   ~q [~q]...~n',
         [Goal,Params]),

      icl_standardize_params(InitialParams, false, Params),
      % Check for inappropriate params
        ( icl_GetParamValue(cache(true), Params), icl_compound_goal(Goal) ->
          format('~w: ~w  (~w)~n  Goal: ~w~n',
               ['WARNING', 'Ignoring ''cache'' parameter',
                'cannot be used with compound goal', Goal]),
            Compound = true
      | otherwise ->
          Compound = false
      ),

      % Add context to params
      ( oaa_current_contexts(_, Contexts) ->
          append(Contexts, Params, NewParams)
      | otherwise ->
          NewParams = Params
      ),

      % check cache
      (icl_GetParamValue(cache(true), NewParams), \+ Compound,
       on_exception(_, oaa_InCache(Goal, Solutions), fail) ->
           oaa_TraceMsg('~n~nSolutions found in cache:~n   ~q.~n',
              [Solutions])
      |
          % Should I solve this only locally?
        (oaa_Id(Me),
          memberchk(address(Me), Params) ->
            findall(Goal, oaa_solve_local(Goal, NewParams), Solutions)

        |
          % send request to Facilitator
          oaa_cont_solve(Goal, NewParams, Solutions),

          % print appropriate trace message
          (icl_GetParamValue(reply(none), NewParams) ->
             oaa_TraceMsg('~n~nMessage broadcast.~n', [])
          |
             oaa_TraceMsg('~n~nSolutions returned:~n   ~q.~n',
               [Solutions])
          ),

          % cache returned solutions if necessary
          ((icl_GetParamValue(cache(true), NewParams), Solutions \== []) ->
             oaa_AddToCache(Goal, Solutions),
             oaa_TraceMsg('Solutions cached.~n',[])
          | true)
        )

      ),!,

      % backtrack over all solutions
      member(Goal, Solutions).

oaa_solve_local(FullGoal, Params) :-
    % Validate the goal:
    icl_GoalComponents(FullGoal, _, Goal1, GoalParams),
    ( oaa_solvables(Solvables) -> true | otherwise -> Solvables = []),
    ( icl_compound_goal(Goal1) ;
      icl_BuiltIn(Goal1) ;
      oaa_goal_matches_solvables(Goal1, Solvables, Goal, Matched) ),
    !,

    % More "local" params take precedence, so they go to the
    % beginning of the list:
    append([GoalParams, Params], AllParams),

    % We don't want tests to be performed repeatedly with compound goals,
    % so we remove them after testing.
    ( passes_tests(AllParams) ->
        delete(AllParams, test(_), NewParams),
      ( ( \+ var(Matched), Matched = solvable(_, SolvParams, _),
          icl_GetParamValue(type(data), SolvParams) ) ->
            ( memberchk(solution_limit(N), AllParams) ->
                call_n(N, Goal)
            | otherwise ->
                call(Goal)
            )
      | otherwise ->
          ( memberchk(solution_limit(N), AllParams) ->
              call_n(N, oaa_Interpret(Goal, NewParams))
          | otherwise ->
              oaa_Interpret(Goal, NewParams)
          )
      )
    | otherwise ->
        oaa_TraceMsg('~nDoesn''t pass test in: ~q~n', [AllParams]),
      fail
    ).

oaa_solve_local(FullGoal, _Params) :-
        format('~nError: do not know how to solve: ~q~n', [FullGoal]), fail.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_cont_solve
% purpose: Post request for solutions, and if appropriate, poll until
%          results are returned.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_cont_solve(Goal, GlobalParams, Solutions) :-
      % Send the ev_post_solve event to the Facilitator
      oaa_PostEvent(ev_post_solve(Goal, GlobalParams),[]),

      % Compound goals may also contain relevant params
        icl_GoalComponents(Goal, _, _, Params),
      append(Params, GlobalParams, AllParams),

      % If delayed reply or no reply OK, succeed immediately
      ( ( icl_GetParamValue(reply(false), AllParams) ;
          icl_GetParamValue(reply(none), AllParams) ;
          icl_GetParamValue(block(false), AllParams) ) ->
         Solutions = [Goal],
         Requestees = [],
         Solvers = []
      |
         % otherwise wait for solutions to return

         icl_GetParamValue(priority(P), AllParams),
         oaa_poll_until_event(ev_reply_solved(Requestees, Solvers, Goal,
SolvedParams, Solutions),
                            P),
        % The facilitator is responsible for making SolvedParams
        % unifiable with GlobalParams.  This msg is to keep facilitator
        % writers honest.
        ( GlobalParams = SolvedParams ->
            true
        | otherwise ->
            format('~w: ~w ~w~n  ~w: ~w~n',
                 ['WARNING:', 'Params in solved event don''t unify',
                  'with original params', 'SolvedParams', SolvedParams])
        )
      ),
        % Return Solvers if requested:
      ( memberchk(get_satisfiers(Solvers), GlobalParams) -> true | true ),
        % Return Requestees if requested:
      ( memberchk(get_address(Requestees), GlobalParams) -> true | true ).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_Solve/1
% purpose: Convenience function:  oaa_Solve with default parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
oaa_Solve(Goal) :- oaa_Solve(Goal, []).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_InCache
% purpose: Retrieve solutions from the cache if the goal we are
%          asking for is properly contained in the cache (check subsumption)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_InCache(Goal, Solutions):-
    oaa_cache(SomeGoal, _),
    subsumes_chk(SomeGoal, Goal),
    !,
    findall(Solution, oaa_cache(Goal, Solution), Solutions).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_AddToCache
% purpose: Add each solution to goal one at a time
%        so we can retrieve solutions later using findall
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_AddToCache(Goal, Solutions) :-
        member(Solution, Solutions),
         \+ oaa_cache(Goal, Solution),
        assert(oaa_cache(Goal, Solution)),
        fail.
oaa_AddToCache(_Goal, _Solutions).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_ClearCache
% purpose: Clear the cache
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_ClearCache :-
    retractall(oaa_cache(_,_)).



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_poll_until_event
% purpose: Block until requested event arrives in oaa_GetEvent
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

oaa_poll_until_event(Event) :-
    icl_param_default(priority(P)),
    oaa_poll_until_event(Event,P).

oaa_poll_until_event(Event,Priority) :-
      oaa_poll_until_all_events([Event],Priority).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_poll_until_all_events
% purpose: Block until all requested events arrive
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% no more events: we're done!
```

```
oaa_poll_until_all_events([], _Priority) :- !.

%% @@Adam - you were apparently working on this; I corrected a syntax
%% error or two, but otherwise left it alone.  - Dave
oaa_poll_until_all_events(EventList, Priority) :-
      % If we have a waiting_event, grab it
      %   see problem description in (oaa_is_waiting_for)
      (oaa_grab_waiting_event(EventList, Event) ;
       oaa_GetEvent(Event, Params, 0)),

      % if timeout returned, check triggers and call user:oaa_AppIdle
      %    then fail (continue with next clause)
      (Event = timeout ->
         oaa_CheckTriggers(task, _, _),
         oaa_call_callback(app_idle, _, []),
         fail
      |
         oaa_cont_poll_until_all_events(EventList, Event, Params, Priority)
      ), !.

% if oaa_GetEvent fails (e.g. timeout), just continue waiting
oaa_poll_until_all_events(EventList, Priority) :-
      oaa_poll_until_all_events(EventList, Priority).

oaa_cont_poll_until_all_events(EventList, Event, _Params, Priority) :-
      remove_element(Event, EventList, NewEventList), !,
      oaa_poll_until_all_events(NewEventList, Priority).
oaa_cont_poll_until_all_events(EventList, Event, Params, Priority) :-
      % if the new event is a ev_reply_solved() message for which we
      % are waiting at a higher recursive level, save this for
      % a later time, until we pop back out to the correct level.
      (oaa_is_waiting_for(Event) ->
         assert(oaa_waiting_event(Event))
      |
         % record what events we are waiting for on this processing level
         gensym(wait, WaitId),
         assert(oaa_waiting_for(WaitId, EventList)),

         (oaa_ProcessEvent(Event, Params) | true), !,

         % level over, remove waiting statement
         retract(oaa_waiting_for(WaitId, EventList))
      ),
      oaa_poll_until_all_events(EventList, Priority).

%*****************************************************************************
% Callbacks
%*****************************************************************************


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_RegisterCallback
% purpose: Declare what procedures should be used for callbacks.  These
%          are application-defined procedures called by library code.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

oaa_RegisterCallback(CallbackID, CallbackProc) :-
```

```
    ( CallbackProc = Module:Proc ->
      true
    | otherwise ->
      Module = user,
      Proc = CallbackProc
    ),
    retractall( oaa_callback(CallbackID, _) ),
    assert( oaa_callback(CallbackID, Module:Proc) ).

oaa_call_callback(CallbackID, SpecifiedCB, Args) :-
    ( ground(SpecifiedCB) ->
      SpecifiedCB = Module:Functor
    | otherwise ->
      oaa_callback(CallbackID, Module:Functor)
    ),
    !,
    Call =.. [Functor | Args],
    on_exception(E,
             Module:Call,
             ( oaa_TraceMsg('WARNING (oaa.pl): Exception raised thru callback
handler (~w):~n  ~q~n',
                          [Module:Functor, E]),
               fail )
            ).
oaa_call_callback(_CallbackID, _SpecifiedCB, _Args).

%******************************************************************************
% Debugging
%******************************************************************************

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_TraceMsg
% purpose: If trace mode is on, display message and arguments
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_TraceMsg(FormatString, Args) :-
      (oaa_trace(on) ->
          format(FormatString, Args)
%         oaa_Inform(trace_info, FormatString, Args)
          ;
          true).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_ComTraceMsg
% purpose: If com trace mode is on, display message and arguments
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_ComTraceMsg(FormatString, Args) :-
      (oaa_com_trace(on) ->
          format(FormatString, Args)
%         oaa_Inform(trace_info, FormatString, Args)
          ;
          true).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_turn_on_debug
% purpose: start debugging if debug mode is on
% remarks:
```

```
%    Use predicate_property and call so as to avoid errors in
%    building and running a Quintus runtime system.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_turn_on_debug :-
      (oaa_debug(on) ->
          ( predicate_property(user:trace, built_in) ->
              call(user:trace)
          | true )
        | true).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_turn_off_debug
% purpose: stop debugging if debug mode is on
% remarks:
%    Use predicate_property and call so as to avoid errors in
%    building and running a Quintus runtime system.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_turn_off_debug :-
      (oaa_debug(on) ->
          ( predicate_property(user:nodebug, built_in) ->
              call(user:nodebug)
          | true )
        | true).


%*****************************************************************************
% User Interface
%*****************************************************************************

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_Inform
% purpose: sends a typed message to interested agents
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_Inform(TypeInfo, FormatString, Args) :-
          oaa_TraceMsg(FormatString, Args),
          (oaa_class(leaf) ->
             sprintf(Result, FormatString, Args),
             oaa_Solve(inform(TypeInfo, Result), [strategy(inform)])
            |
             true
          ), !.




%*****************************************************************************
% Connection primitives
%*****************************************************************************

%%% BUG/HACK!!!!!
% tcp_send/1 is not currently defined (new version of quintus)
% so these predicates should fail.  This means we can't have
% multilevel facilitators.
% However, if we fix it by the tcp_send/2 version (commented out),
% killing the agent doesn't shut down both connections and the
% facilitator server doesn't register the agent as disconnected.
% This must be fixed, but I don't have time now...

% Ask the root agent for the address of facilitator FacName.
% Either FacId or FacName may be bound.
% IMPORTANT: This assumes the root agent is the only connection when
% this is called.
% @@Not happy with the use of a Connection number in the address param here.
% Can an address be a connection number as well as an id or name???  [No.]

% get_address(FacId, FacName, Port, Host):-
%         tcp_connected(RootConnection),
%     oaa_Solve(agent_location(FacId, FacName, Port, Host),
%             [address(RootConnection)]).


%% succeed if FacName has not been registered with the root agent.
%%    otherwise, ask user to enter a different name for FacName

% check_name_duplication(MyName, NewMyName) :-
%         tcp_send(ev_check_agent_name(MyName)),
%         oaa_select_event(0, X),
%         oaa_extract_event(X, Result, _), %% 'UNIQUE'
%         (Result == 'UNIQUE' -> NewMyName = MyName
%                 ;
%         format('Name is duplicated~n',[]),
%         format('The following are registered ~n ~q ~n',[Result]),
%         format('Input agent name again:',[]),
%         read(NewMyName)).

% report_address_to_root(MyName, NewAddress):-
%     tcp_send(register_port_number(MyName, NewAddress)).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% routines to fix bug:
%   blocking solve1
%      incoming event generates blocking solve2
%      solution to solve1 thrown away!!!
%      solutions to solve2
%   stuck  waiting for solve1 forever
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % name:    oaa_is_waiting_for
   % purpose: Check to see if the current event is something we are waiting
   %          for on a higher recursive level
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   oaa_is_waiting_for(Event) :-
        oaa_waiting_for(_Id, EventList),
        memberchk(Event, EventList).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % name:    oaa_grab_waiting_event
   % purpose: If one of the delayed events is in the EventList that we are
   %     waiting for, return this event and remove from delayed list
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   oaa_grab_waiting_event(EventList, Event) :-
```

```
        oaa_waiting_event(Event),
        memberchk(Event, EventList),
        !,
        retract(oaa_waiting_event(Event)).


%*****************************************************************************
% OAA Utilities
%*****************************************************************************

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_remove_solvables_data(Solvables).
% purpose: For each data solvable, remove all clauses belonging to it.
% remarks: - Solvables must be in standard form, and should include only
%            data solvables.
%          - Permissions are ignored.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_remove_solvables_data([]).
oaa_remove_solvables_data([Solvable | Solvables]) :-
    Solvable = solvable(Goal, Params, _Perms),
    icl_GetParamValue(type(data), Params),
    \+ memberchk(synonym(_, _), Params),
    !,
    % This should have already been done, but to be safe:
    (clause(Goal, _, _) -> true | true),
    predicate_skeleton(Goal, Skeleton),
    ( oaa_remove_data_local(Skeleton, [do_all(true)]) ->
        true
    | otherwise ->
        format('~w: Problem in removing all data for solvable: ~w~n',
             ['! ERROR', Goal])
    ),
    oaa_remove_solvables_data(Solvables).
oaa_remove_solvables_data([_Solvable | Solvables]) :-
    oaa_remove_solvables_data(Solvables).

oaa_remove_data_owned_by(Id) :-
    ( oaa_solvables(Solvables) -> true | otherwise -> Solvables = []),
    oaa_built_in_solvables(BuiltIns),
    append(BuiltIns, Solvables, AllSolvables),
    oaa_remove_data_owned_by(AllSolvables, Id).

oaa_remove_data_owned_by([], _Id).
oaa_remove_data_owned_by([Solvable | Solvables], Id) :-
    Solvable = solvable(Goal, Params, _Perms),
    icl_GetParamValue(type(data), Params),
    \+ icl_GetParamValue(persistent(true), Params),
    \+ icl_GetParamValue(synonym(_, _), Params),
    !,
    % This should have already been done, but to be safe:
    (clause(Goal, _, _) -> true | true),
    predicate_skeleton(Goal, Skeleton),
    ( oaa_remove_data_local(Skeleton, [owner(Id), do_all(true)]) ->
        true
    | otherwise ->
        format('~w: Problem in removing data owned by ~w for solvable:~n  ~w~n',
             ['! ERROR', Id, Goal])
    ),
    oaa_remove_data_owned_by(Solvables, Id).
oaa_remove_data_owned_by([_Solvable | Solvables], Id) :-
    oaa_remove_data_owned_by(Solvables, Id).


%*****************************************************************************
% General Utilities
%*****************************************************************************

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_consult(+FilePath, -AbsFileName).
% purpose:
% remarks: We don't use Quintus' builtin consult, because it's too picky
%          about associating predicates with files.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_consult(FilePath, AbsFileName) :-
    absolute_file_name(FilePath, AbsFileName),
    can_open_file(AbsFileName, read, fail),
    open(AbsFileName, read, Stream),
    load_clauses(Stream),
    close(Stream).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    load_clauses(+Stream).
% purpose:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load_clauses(Stream) :-
    repeat,
    read_term(Stream, [], Term),
    ( Term = ':-'(_Body) ->
      true
    | Term = end_of_file ->
      true
    | otherwise ->
        load_clause(Term)
    ),
    ( at_end_of_file(Stream) ->
      !
    | otherwise ->
      fail
    ).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    load_clause(+Term).
% purpose:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load_clause(Term) :-
    assert( Term ).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    oaa_declare_for_prolog(Solvables).
% purpose: For each solvable, make sure it's known to Prolog as a dynamic
%          predicate.  This will prevent exceptions and warnings from
%          calls and retracts before there have been any asserts.
% remarks: Solvables must be in standard form, and should include only
%          data solvables.
%          This is probably Quintus-specific.
%          We are assuming that none of these predicates are known to
%          Prolog as compiled predicates.  Would be better to check for this.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
oaa_declare_for_prolog([]).
oaa_declare_for_prolog([solvable(Pred, _, _) | Rest]) :-
    copy_term(Pred, PredCopy),
    ( clause(PredCopy, _Body) -> true | true ),
    oaa_declare_for_prolog(Rest).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    predicate_skeleton(+Goal, +Skeleton).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
predicate_skeleton(Goal, Skeleton) :-
    functor(Goal, Functor, Arity),
    functor(Skeleton, Functor, Arity).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    sprintf
% purpose: C-like command formats a string + args into an atom
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sprintf(AtomResult, FormatStr, Args) :-
   with_output_to_chars(format(FormatStr, Args), Chars),
   name(AtomResult, Chars).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    memberchk_nobind
% purpose: like memberchk, but doesn't bind variables in Elt when doing test.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
memberchk_nobind(Elt, [H|_]) :-
      would_unify(Elt, H), !.
memberchk_nobind(Elt, [_|T]) :-
      memberchk_nobind(Elt, T).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    would_unify
% purpose: succeeds if X and Y WOULD unify, but doesn't actually do the
%          unification (no variables are bound by test)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
would_unify(X,Y) :- \+ \+ X = Y.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    remove_element
% purpose: Removes the element X from a list
% remarks: Fails if X is not an element in the list
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
remove_element(X, [X|Rest], Rest) :- !.
remove_element(X, [Y|Rest], [Y|Rest2]) :- remove_element(X, Rest, Rest2).
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    replace_element(Elt, List, New, NewList)
% purpose: Replaces the element Elt, if present in List, with the element New
% remarks: If there are multiple occurrences of Elt, only replaces the first
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
replace_element(Elt, [Elt|Rest], New, [New|Rest]) :- !.
replace_element(Elt, [Y|Rest], New, [Y|Rest2]) :-
    replace_element(Elt, Rest, New, Rest2).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    select_elements(List, Selector, NewList)
% purpose: Selects all List elements for which Selector(element) succeeds.
% remarks: If there are multiple occurrences of Elt, only replaces the first
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
select_elements([], _Selector, []).
select_elements([Element | Elements], Selector, [Element | Selected]) :-
    Test =.. [Selector, Element],
    call( Test ),
    !,
    select_elements(Elements, Selector, Selected).
select_elements([_Element | Elements], Selector, Selected) :-
    select_elements(Elements, Selector, Selected).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% name:    call_n(+N, +Goal)
% purpose: Call Goal with a limit on the number of solutions generated.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

call_n(1, Goal) :-
    call(Goal),
    !.
call_n(N, Goal) :-
    % Remember the counter's value in case anyone else is using it.
    ctr_is(12, CtrOrig),
    call_n_aux(N, Goal, CtrOrig).

call_n_aux(N, Goal, CtrOrig) :-
    N > 1,
    ctr_set(12, 1),
    call(Goal),
    ctr_inc(12, 1, M),
    ( M =< N ->
        true
    | otherwise ->
        ctr_set(12, CtrOrig),
      !,
      fail
    ).
    % This clause is for when the Goal fails before M > N:
call_n_aux(_N, _Goal, CtrOrig) :-
    ctr_set(12, CtrOrig),
    !,
    fail.
```

```
% findall with a limit on the number of solutions generated.
findNSolutions(0, _Var, _Predicate, []).
findNSolutions(1, Var, Predicate, [Var]) :-
        call(Predicate), !.
findNSolutions(1, _Var, _Predicate, []).
findNSolutions(N, Var, Predicate, Solutions) :-
    N > 1,
    % Save the counter's value in case anyone else is using it.
    ctr_is(12, CtrOrig),
    ctr_set(12, 1),
    findall(Var,
            (Predicate, ctr_inc(12, 1, M),
             (M >= N -> ! | otherwise -> true)),
           Solutions),
    ctr_set(12, CtrOrig).

% ================================================================
% No longer used: replaced or obsolete
% ================================================================

% initialize all data flags
% oaa_init_flags :-
%     % set appropriate prolog flags
%     prolog_flag(fileerrors,_,on),
%     prolog_flag(syntax_errors,_,error),
%     % Let's use retractall so as to avoid unknown exceptions when tracing:
%     retractall(oaa_cache(_,_)),
%     retractall(oaa_already_loaded(_)),
%     assert(oaa_trace(off)),
%     assert(oaa_debug(off)),
%     assert(oaa_com_trace(off)),
%     tcp_trace(_,off).
```

APPENDIX A.V

Source code file named translations.pl.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   File      : translations.pl
%   Primary Authors  : David Martin, Adam Cheyer
%   Purpose : Provides translations for backward compatibility with OAA 1.0
%
%   ----------------------------------------------------------------------------
%   Unpublished-rights reserved under the copyright laws of the United States.
%
%
%   Unpublished Copyright (c) 1998, SRI International.
%   "Open Agent Architecture" and "OAA" are Trademarks of SRI International.
%   ----------------------------------------------------------------------------

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This file is loaded by facilitator code, and thus no
% module imports are needed here.

% Currently, we support a 3.0 facilitator with a mix of 3.0 and/or pre-3.0
% clients.
% A pre-3.0 facilitator with a 3.0 client is NOT supported, and probably
% never will be.

:- multifile oaa_AppDoEvent/2.

% At present we only support the case where the facilitator is 3.0, and
% the client is pre-3.0.

    % Here we can ignore the languages.
oaa_event_translation(2.0, L1, 3.0, L2, Connection, Event1, Event2) :-
    oaa_event_translation(2.1, L1, 3.0, L2, Connection, Event1, Event2).
oaa_event_translation(2.1, _L1, 3.0, _L2, _Connection, Event1, Event2) :-
    ( Event1 = event(From, Contents1, Priority) ->
      Params2 = [from(From), priority(Priority)]
    | Event1 = event(From, Contents1) ->
      Params2 = [from(From)]
    | Event1 = Contents1 ->
      Params2 = []
    ),
    ( ev_trans_21_30(Contents1, Contents2) ->
      true
    | otherwise ->
      Contents2 = Contents1
    ),
    Event2 = event(Contents2, Params2).

    % Here we can ignore the languages.
oaa_event_translation(3.0, L1, 2.0, L2, Connection, Event1, Event2) :-
    oaa_event_translation(3.0, L1, 2.1, L2, Connection, Event1, Event2).
oaa_event_translation(3.0, _L1, 2.1, _L2, _Connection, Event1, Event2) :-
    Event1 = event(Contents1, Params1),
    ( ev_trans_30_21(Contents1, Params1, Contents2) ->
      true
    | otherwise ->
      Contents1 = Contents2
    ),
    ( memberchk(from(KS), Params1) ->
      Event2 = event(KS, Contents2)
    | otherwise ->
      Event2 = Contents2
    ),
    !.
    % Anything not specified explicitly stays the same:
oaa_event_translation(3.0, _L1, 2.1, _L2, _Connection, E1, E1).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The following could go to or from the facilitator.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ev_trans_21_30(trace_on, ev_trace_on).
ev_trans_21_30(trace_off, ev_trace_off).
ev_trans_21_30(tcp_trace_on, ev_com_trace_on).
ev_trans_21_30(tcp_trace_off, ev_com_trace_off).
ev_trans_21_30(debug_on, ev_debug_on).
ev_trans_21_30(debug_off, ev_debug_off).
ev_trans_21_30(set_timeout(N), ev_set_timeout(N)).
ev_trans_21_30(halt, ev_halt).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The following are sent only from (pre-3.0) client to facilitator.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ev_trans_21_30(post_event(Event), ev_post_event(NewEvent)) :-
    ev_trans_21_30(Event, NewEvent).
ev_trans_21_30(post_event(To, Event), ev_post_event(To, NewEvent)) :-
    ev_trans_21_30(Event, NewEvent).

ev_trans_21_30(post_query(Goal, Params),
             ev_post_solve(Goal, [reflexive(false) | NewParams])) :-
    params_trans_21_30(Params, NewParams).

% This is the message from a facilitator to its parent facilitator;
% will probably evolve:
% ev_trans_21_30(register_solvable_goals(AGL), register_solvable_goals(AGL)).
% NO, we don't want to translate this.  The old form is still handled
% by the new facilitator:
% ev_trans_21_30(register_solvable_goals(GoalList, KSName),
%                          ev_register_solvables(add, GoalList, KSName,
%                                    [if_exists(overwrite)])).

ev_trans_21_30(solved(GoalId, FromKS, Goal, SolveParams, Solutions),
       ev_solved(GoalId, FromKS, Goal, SolveParams, Solutions)).

/* post_trigger/4: retained for backwards compatibility */
ev_trans_21_30(post_trigger(test, Type, Cond, Action), NewEvent) :-
    ev_trans_21_30(post_trigger(test, Type, unused, unused, Cond, Action),
               NewEvent).

/* post_trigger/4: retained for backwards compatibility */
ev_trans_21_30(post_trigger(data, Type, Cond, Action), NewEvent) :-
    ev_trans_21_30(post_trigger(data, Type,
                        [on_write, on_write_replace, on_replace],
                        Cond, true, Action), NewEvent).
```

```
/* post_trigger/4: retained for backwards compatibility */
ev_trans_21_30(post_trigger(event, Type, Cond, Action), NewEvent) :-
    ev_trans_21_30(post_trigger(event, Type, [on_receive], Cond, true, Action),
               NewEvent).

ev_trans_21_30(post_trigger(Kind,Recur,OpMask,Template,Test,Action),
             ev_post_trigger_update(add,Mode,Condition,NewAction,Params)) :-
    ( Kind == test -> Mode = task
    | Kind == event -> Mode = comm
    | Kind == alarm -> Mode = time
    | otherwise -> Mode = Kind ),
    ( Recur == whenever ->
      Recurrence = [recurrence(whenever)]
    | otherwise ->
      Recurrence = [recurrence(when)]
    ),
    template_trans_21_30(Kind, Template, Condition),
    ( var(Test) -> TestParam = [] | otherwise -> TestParam = [test(Test)] ),
    ( Mode == data, ev_trans_21_30(Action, NewAction) -> true
    | otherwise -> NewAction = Action ),
    opmask_trans_21_30(OpMask, OpParam),
    ( Mode == data ->
      oaa_Id(FacId),
      Addr = [address(FacId)]
    | otherwise ->
      Addr = []
    ),
    append([Addr, [reply(none),reflexive(false)],
          Recurrence, TestParam, OpParam], Params).
ev_trans_21_30(post_trigger(KS, Kind,Recur,OpMask,Template,Test,Action),
             ev_post_trigger_update(add,Type,Condition,NewAction,Params)) :-
    ( Kind == test -> Type = task
    | Kind == event -> Type = comm
    | Kind == alarm -> Type = time
    | otherwise -> Type = Kind),
    ( Recur == whenever ->
      Recurrence = recurrence(whenever)
    | otherwise ->
      Recurrence = recurrence(when)
    ),
    template_trans_21_30(Kind, Template, Condition),
    ( var(Test) -> TestParam = [] | otherwise -> TestParam = [test(Test)] ),
    oaa_Id(FacId),
    ( KS == FacId, ev_trans_21_30(Action, NewAction) -> true
    | otherwise -> NewAction = Action ),
    opmask_trans_21_30(OpMask, OpParam),
    append([[address(KS), reply(none), reflexive(false)],
          Recurrence, TestParam, OpParam],
         Params).

params_trans_21_30([], []).
params_trans_21_30([Param | Params], [NewParam | NewParams]) :-
    ( param_trans_21_30(Param, NewParam) ->
      true
    | otherwise ->
      NewParam = Param
    ),
    params_trans_21_30(Params, NewParams).

param_trans_21_30(cache, cache(true)).
param_trans_21_30(solution_limit(N), solution_limit(N)).
param_trans_21_30(reflexive, reflexive(true)).
param_trans_21_30(address(A), address(NewA)) :-
    ( is_list(A) -> NewA = A | otherwise -> NewA = [A] ).
param_trans_21_30(broadcast, reply(none)).
param_trans_21_30(asynchronous, reply(asynchronous)).
% @@DLM: is this handled?:
param_trans_21_30(test(T), test(T)).
param_trans_21_30(level_limit(N), level_limit(N)).
param_trans_21_30(time_limit(N), time_limit(N)).
% @@DLM: NOT HANDLED!:
param_trans_21_30(and_parallel, and_parallel).
param_trans_21_30(or_parallel, or_parallel).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The following could go to or from the facilitator.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ev_trans_30_21(ev_trace_on, _EvParams, trace_on).
ev_trans_30_21(ev_trace_off, _EvParams, trace_off).
ev_trans_30_21(ev_com_trace_on, _EvParams, tcp_trace_on).
ev_trans_30_21(ev_com_trace_off, _EvParams, tcp_trace_off).
ev_trans_30_21(ev_debug_on, _EvParams, debug_on).
ev_trans_30_21(ev_debug_off, _EvParams, debug_off).
ev_trans_30_21(ev_set_timeout(N), _EvParams, set_timeout(N)).
ev_trans_30_21(ev_halt, _EvParams, halt).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The following are sent only from facilitator to client.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ev_trans_30_21(
             ev_solve(ID, Goal, NewParams),
            _EventParams,
             solve(ID, Goal, Params)) :-
    params_trans_30_21(Params, NewParams).

ev_trans_30_21(ev_reply_solved(_, Solved, Goal, SolveParams, Solutions),
             _EventParams,
             solved(FromKS, Goal, SolveParams, Solutions)) :-
    ( Solved = [FromKS] ->
      true
    | otherwise ->
      FromKS = Solved
    ).

    % OBSOLETE: forget these:
% ev_trans_30_21(add_trigger(data, Type, Cond, Action),
% ev_trans_30_21(add_trigger(event, Type, Cond, Action)
% ev_trans_30_21(add_trigger(test, Type, Cond, Action)
% @@DLM: Don't think this is needed:
% ev_trans_30_21(inform_ui(TypeInfo, Result), ))

ev_trans_30_21(
    ev_update_trigger(_ID, add, Type, Condition, Action, TrigParams),
    _EventParams,
    add_trigger(Kind, Recur, OpMask, Template, Test, Action) ) :-
    ( Type = task -> Kind == test
    | Type = comm-> Kind == event
    | Type = time-> Kind == alarm
    | otherwise -> Type = Kind ),
    ( memberchk(recurrence(whenever), TrigParams) ->
      Recur = whenever
    | otherwise ->
      Recur = when
    ),
    Template = Condition,
    ( memberchk(test(Test), TrigParams) -> true | otherwise -> Test = _ ),
    ( memberchk(on(OpParam), TrigParams) ->
      true
    | otherwise ->
      OpParam = _
    ),
    opmask_trans_30_21(OpParam, OpMask),
    ( memberchk(test(Test), TrigParams) -> true | true ).

params_trans_30_21([], []).
params_trans_30_21([Param | Params], [NewParam | NewParams]) :-
    ( param_trans_30_21(Param, NewParam) ->
      true
    | otherwise ->
      NewParam = Param
    ),
    params_trans_30_21(Params, NewParams).

param_trans_30_21(cache(true), cache).
param_trans_30_21(solution_limit(N), solution_limit(N)).
param_trans_30_21(reflexive(true), reflexive).
% @@DLM: double-check this:
param_trans_30_21(address(A), address(A)).
param_trans_30_21(reply(none), broadcast).
param_trans_30_21(reply(asynchronous), asynchronous).
% @@DLM: is this handled?:
param_trans_30_21(test(T), test(T)).
param_trans_30_21(level_limit(N), level_limit(N)).
param_trans_30_21(time_limit(N), time_limit(N)).
% @@DLM: NOT HANDLED!:
param_trans_30_21(and_parallel, and_parallel).
param_trans_30_21(or_parallel, or_parallel).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The following are sent only from a pre-3.0 facilitator to a client.
% Backwards compatibility not currently supported.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% ev_trans_21_30(solved(FromKS, Goal, SolveParams, Solutions),
%        ev_reply_solved([FromKS], Solvers, Goal, SolveParams, Solutions)) :-
%     ( Solutions == [] ->
%     Solvers = []
%     | otherwise ->
%     Solvers = [FromKS]
%     ),
%     ( memberchk(get_address(FromKS), SolveParams) ->
%     true
%     | otherwise ->
%     FromKS = unknown
%     ).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Auxiliary procedures.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Returns either a Singleton list or an empty list.
opmask_trans_21_30(OpMask, []) :-
    var(OpMask),
    !.
opmask_trans_21_30(OpMask, OpParam) :-
    \+ is_list(OpMask),
    !,
    opmask_trans_21_30([OpMask], OpParam).
opmask_trans_21_30([],  []).
opmask_trans_21_30([Elt | Rest], [EltTrans | RestTrans]) :-
    opmask_elt_trans_21_30(Elt, EltTrans),
    !,
    opmask_trans_21_30(Rest, RestTrans).
opmask_trans_21_30([_Elt | Rest], RestTrans) :-
    !,
    opmask_trans_21_30(Rest, RestTrans).
opmask_elt_trans_21_30(on_send, on(send)).
opmask_elt_trans_21_30(on_receive, on(receive)).
opmask_elt_trans_21_30(on_write, on(add)).
opmask_elt_trans_21_30(on_retract, on(remove)).
opmask_elt_trans_21_30(on_replace, on(replace)).
% This one probably doesn't have a precise translation:
opmask_elt_trans_21_30(on_write_replace, on(replace)).

opmask_trans_30_21(OpMask, OpMask) :-
    var(OpMask),
    !.
opmask_trans_30_21(OpMask, OpParam) :-
    \+ is_list(OpMask),
    !,
    opmask_trans_30_21([OpMask], OpParam).
opmask_trans_30_21([],  []).
opmask_trans_30_21([Elt | Rest], [EltTrans | RestTrans]) :-
    opmask_elt_trans_30_21(Elt, EltTrans),
    !,
    opmask_trans_30_21(Rest, RestTrans).
opmask_trans_30_21([_Elt | Rest], RestTrans) :-
    !,
    opmask_trans_30_21(Rest, RestTrans).
opmask_elt_trans_30_21(on(send), on_send).
opmask_elt_trans_30_21(on(receive), on_receive).
opmask_elt_trans_30_21(on(add), on_write).
opmask_elt_trans_30_21(on(remove), on_retract).
opmask_elt_trans_30_21(on(replace), on_replace).
% This one probably doesn't have a precise translation:
```

```
opmask_elt_trans_30_21(on(replace), on_write_replace).

template_trans_21_30(data,
                 data(ksdata, [AgentId,Status,Solvables,Name]),
                 agent_data(AgentId,Status,Solvables,Name)) :-
    !.
template_trans_21_30(data, Template, Template) :-
    !.
template_trans_21_30(event, Template, Condition) :-
    !,
    ev_trans_21_30(Template, Condition).
template_trans_21_30(_, Template, Template).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Event handlers for selected pre-3.0 events.
%
% In these cases, this approach is easier than providing an event
% translation.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


oaa_AppDoEvent(register_solvable_goals(GoalList), Params) :-
        memberchk( connection_id(Connection), Params),
      % This hack inherited from b.pl:
      oaa_AppDoEvent(register_solvable_goals(GoalList, Connection),
                   Params).

oaa_AppDoEvent(register_solvable_goals(GoalList, Name), Params) :-
        memberchk( connection_id(Connection), Params),
      update_connected(Connection, [oaa_name(Name)]),
      icl_ConvertSolvables(GoalList, Solvables),
      oaa_AppDoEvent(ev_register_solvables(add,Solvables,Name,[if_exists(overwri
te)]),
                   Params).

oaa_AppDoEvent(can_solve(Goal), EvParams) :-
    memberchk(from(KS), EvParams),
    findall(SomeKS, choose_ks_for_goal(KS, Goal, _, [], SomeKS, _), AgentList),
    oaa_PostEvent(return_can_solve(Goal, AgentList), [address(KS)]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% BB events
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

oaa_AppDoEvent(write_bb(ksdata, [Id,Status,Solvables,Name]),
             EvParams) :-
    !,
    ( var(Solvables) ->
      % (Surely this never happens.)
        oaa:oaa_add_data_local(agent_data(Id,Status,Solvables,Name), [from(Id)])
    | otherwise ->
        icl_ConvertSolvables(Solvables, FormalSolvables),
        oaa_AppDoEvent(ev_register_solvables(add,FormalSolvables,Name,
                                            [if_exists(overwrite)]),
                        [from(Id) | EvParams])
    ).
oaa_AppDoEvent(write_bb(oaa_version, V), EvParams) :-
```

```
    !,
    memberchk(from(Id), EvParams),
    % oaa:oaa_add_data_local(data(oaa_Version, V), [from(Id)]),
    com_GetInfo(ConnectionId, oaa_id(Id)),
    com_AddInfo(ConnectionId, agent_version(V)).
oaa_AppDoEvent(write_bb(language, Language), EvParams) :-
    !,
    memberchk(from(Id), EvParams),
    com_GetInfo(ConnectionId, oaa_id(Id)),
    com_AddInfo(ConnectionId, agent_language(Language)).
oaa_AppDoEvent(write_bb(kshost, Host), EvParams) :-
    !,
    memberchk(from(Id), EvParams),
    oaa:oaa_solve_local(agent_data(Id, _, _, Name), []),
    oaa:oaa_add_data_local(agent_host(Id, Name, Host),
                       [from(Id) | EvParams]).
oaa_AppDoEvent(write_bb(Item, Data), EvParams) :-
    !,
    memberchk(from(Id), EvParams),
    oaa:oaa_add_data_local(data(Item, Data), [from(Id)]).

oaa_AppDoEvent(write_once_bb(Item, Data), EvParams) :-
    (Item = ksdata ; Item = oaa_version ; Item = language ; Item = kshost),
    !,
    oaa_AppDoEvent(write_bb(Item, Data), [single_value(true) | EvParams]).
oaa_AppDoEvent(write_once_bb(Item, Data), EvParams) :-
    !,
    memberchk(from(Id), EvParams),
    oaa:oaa_add_data_local(data(Item, Data), [from(Id), single_value(true)]).

oaa_AppDoEvent(write_replace_bb(Item, Data), EvParams) :-
    (Item = ksdata ; Item = oaa_version ; Item = language ; Item = kshost),
    !,
    oaa_AppDoEvent(write_bb(Item, Data), [unique_values(true) | EvParams]).
oaa_AppDoEvent(write_replace_bb(Item, Data), EvParams) :-
    !,
    memberchk(from(Id), EvParams),
    oaa:oaa_add_data_local(data(Item, Data), [from(Id), unique_values(true)]).

oaa_AppDoEvent(replace_bb(ksdata, [A,open,C,Name], [A,ready,C,Name]),
             EvParams) :-
    !,
    oaa_AppDoEvent(ev_ready(Name), EvParams).
oaa_AppDoEvent(replace_bb(ksdata, [Id,Status,Solvables,Name],
                            [NewId,NewStatus,NewSolvables,NewName]),
             EvParams) :-
    !,
    ( var(NewSolvables) ->
        oaa:oaa_replace_data_local(agent_data(Id,Status,Solvables,Name),
            [from(Id), with(agent_data(NewId,NewStatus,NewSolvables,NewName))])
    | otherwise ->
        icl_ConvertSolvables(NewSolvables, FormalSolvables),
        oaa_AppDoEvent(ev_register_solvables(add,FormalSolvables,NewName,
                                           [if_exists(overwrite)]),
                       [from(NewId) | EvParams])
    ).
oaa_AppDoEvent(replace_bb(Item, OldData, NewData), EvParams) :-
```

```
    !,
    memberchk(from(Id), EvParams),
    oaa:oaa_replace_data_local(data(Item, OldData),
                     [from(Id), with(data(Item, NewData))]).

% @@DLM: May need more special-purpose clauses starting here:
oaa_AppDoEvent(retract_bb(Item, Data), EvParams) :-
    !,
    memberchk(from(Id), EvParams),
    oaa:oaa_remove_data_local(data(Item, Data), [from(Id)]).

oaa_AppDoEvent(read_bb(ksdata,[AgentId,Status,Solvables,Name]), EvParams) :-
    !,
    memberchk(from(Id), EvParams),
    findall(read_bb(ksdata, [AgentId,Status,Solvables,Name]),
          oaa:oaa_solve_local(agent_data(AgentId,Status,Solvables,Name), []),
          Solutions),
    oaa_simplify_ksdata(Solutions, Simplified),
    oaa_PostEvent(return_read_bb(Simplified), [address(Id)]).
oaa_AppDoEvent(read_bb(KS,kshost,Host), EvParams) :-
    !,
    memberchk(from(Id), EvParams),
    findall(read_bb(KS, kshost, Host),
          oaa:oaa_solve_local(agent_host(KS,_,Host), []),
          Solutions),
    oaa_PostEvent(return_read_bb(Solutions), [address(Id)]).
oaa_AppDoEvent(read_bb(oaa_version,V), EvParams) :-
    !,
    memberchk(from(Id), EvParams),
    % Not sure if this works (but this clause is probably never called):
    findall(read_bb(oaa_version, V),
            ( com_GetInfo(ConnectionId, oaa_id(_)),
            com_GetInfo(ConnectionId, agent_version(V)) ),
          Solutions),
    oaa_PostEvent(return_read_bb(Solutions), [address(Id)]).

oaa_AppDoEvent(read_bb(KS,oaa_version,V), EvParams) :-
    !,
    memberchk(from(Id), EvParams),
    findall(read_bb(KS, oaa_version, V),
            ( com_GetInfo(ConnectionId, oaa_id(KS)),
            com_GetInfo(ConnectionId, agent_version(V)) ),
          Solutions),
    oaa_PostEvent(return_read_bb(Solutions), [address(Id)]).
oaa_AppDoEvent(read_bb(Item,Data), EvParams) :-
    !,
    memberchk(from(Id), EvParams),
    findall(read_bb(Item, Data),
          oaa:oaa_solve_local(data(Item, Data), []),
          Solutions),
    oaa_PostEvent(return_read_bb(Solutions), [address(Id)]).
    % @@The owner parameter isn't implemented yet for solve!
oaa_AppDoEvent(read_bb(_KS, Item,Data), EvParams) :-
    !,
    memberchk(from(Id), EvParams),
    findall(read_bb(Item, Data),
          oaa:oaa_solve_local(data(Item, Data), []),
          Solutions),
    oaa_PostEvent(return_read_bb(Solutions), [address(Id)]).

oaa_simplify_ksdata([], []).
oaa_simplify_ksdata([KSData | Rest], [Simplified | RestSimp]) :-
    KSData = read_bb(ksdata, [A, B, Solvables, D]),
    icl_ConvertSolvables(SimplifiedSolvables, Solvables),
    Simplified = read_bb(ksdata, [A, B, SimplifiedSolvables, D]),
    oaa_simplify_ksdata(Rest, RestSimp).
```

IN THE CLAIMS:

See A1

1. A computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of:

registering a description of each active client agent's functional capabilities, using an expandable, platform-independent, inter-agent language;

receiving a request for service as a base goal in the inter-agent language, in the form of an arbitrarily complex goal expression; and

dynamically interpreting the goal expression, said act of interpreting further comprising:

generating one or more sub-goals using the inter-agent language; and

dispatching each of the sub-goals to a selected client agent for performance, based on a match between the sub-goal being dispatched and the registered functional capabilities of the selected client agent.

2. A computer-implemented method as recited in claim 1, further including the following acts of:

receiving a new request for service as a base goal using the inter-agent language, in the form of another arbitrarily complex goal expression, from at least one of the selected client agents in response to the sub-goal dispatched to said agent; and

recursively applying the last step of claim 1 in order to perform the new request for service.

3. A computer implemented method as recited in claim 2 wherein the act of registering a specific agent further includes:

invoking the specific agent in order to activate the specific agent;

instantiating an instance of the specific agent; and

transmitting the new agent profile from the specific agent to the facilitator agent in response to the instantiation of the specific agent.

Sub B1

4. A computer implemented method as recited in claim 1 further including the act of deactivating a specific client agent no longer available to provide services by deleting the registration of the specific client agent.

5. A computer implemented method as recited in claim 1 further comprising the act of providing an agent registry data structure.

6. A computer implemented method as recited in claim 5 wherein the agent registry data structure includes at least one symbolic name for each active agent.

7. A computer implemented method as recited in claim 5 wherein the agent registry data structure includes at least one data declaration for each active agent.

8. A computer implemented method as recited in claim 5 wherein the agent registry data structure includes at least one trigger declaration for one active agent.

9. A computer implemented method as recited in claim 5 wherein the agent registry data structure includes at least one task declaration, and process characteristics for each active agent.

10. A computer implemented method as recited in claim 5 wherein the agent registry data structure includes at least one process characteristic for each active agent.

11. A computer implemented method as recited in claim 1 further comprising the act of establishing communication between the plurality of distributed agents.

12. A computer implemented method as recited in claim 1 further comprising the acts of:

receiving a request for service in a second language differing from the inter-agent language;

selecting a registered agent capable of converting the second language into the inter-agent language; and

forwarding the request for service in a second language to the registered agent capable of converting the second language into the inter-agent language, implicitly requesting that such a conversion be performed and the results returned.

13. A computer implemented method as recited in claim 12 wherein the request includes a natural language query, and the registered agent capable of converting the second language into the inter-agent language service is a natural language agent.

14. A computer implemented method as recited in claim 13 wherein the natural language query was generated by a user interface agent.

15. A computer implemented method as recited in claim 1, wherein the base goal requires setting a trigger having conditional functionality and consequential functionality.

16. A computer implemented method as recited in claim 15 wherein the trigger is an outgoing communications trigger, the computer implemented method further including the acts of:

monitoring all outgoing communication events in order to determine whether a specific outgoing communication event has occurred; and

in response to the occurrence of the specific outgoing communication event, performing the particular action defined by the trigger.

17. A computer implemented method as recited in claim 15 wherein the trigger is an incoming communications trigger, the computer implemented method further including the acts of:

monitoring all incoming communication events in order to determine whether a specific incoming communication event has occurred; and

in response to the occurrence of a specific incoming communication event satisfying the trigger conditional functionality, performing the particular consequential functionality defined by the trigger.

18. A computer implemented method as recited in claim 15 wherein the trigger is a data trigger, the computer implemented method further including the acts of:

monitoring a state of a data repository; and

in response to a particular state event satisfying the trigger conditional functionality, performing the particular consequential functionality defined by the trigger.

19. A computer implemented method as recited in claim 15 wherein the trigger is a time trigger, the computer implemented method further including the acts of:

monitoring for the occurrence of a particular time condition; and

in response to the occurrence of a particular time condition satisfying the trigger conditional functionality, performing the particular consequential functionality defined by the trigger.

20. A computer implemented method as recited in claim 15 wherein the trigger is installed and executed within the facilitator agent.

21. A computer implemented method as recited in claim 15 wherein the trigger is installed and executed within a first service-providing agent.

22. A computer implemented method as recited in claim 15 wherein the conditional functionality of the trigger is installed on a facilitator agent.

23. A computer implemented method as recited in claim 22 wherein the consequential functionality is installed on a specific service-providing agent other than a facilitator agent.

24. A computer implemented method as recited in claim 15 wherein the conditional functionality of the trigger is installed on a specific service-providing agent other than a facilitator agent.

25. A computer implemented method as recited in claim 15 wherein the consequential functionality of the trigger is installed on a facilitator agent.

26. A computer implemented method as recited in claim 1 wherein the base goal is a compound goal having sub-goals separated by operators.

27. A computer implemented method as recited in claim 26 wherein the type of available operators includes a conjunction operator, a disjunction operator, and a conditional execution operator.

28. A computer implemented method as recited in claim 27 wherein the type of available operators further includes a parallel disjunction operator that indicates that disjunct goals are to be performed by different agents.

29. A computer program stored on a computer readable medium, the computer program executable to facilitate cooperative task completion within a distributed computing environment, the distributed computing environment including a plurality of autonomous electronic agents, the distributed computing environment supporting an Interagent Communication Language, the computer program comprising computer executable instructions for:

providing an agent registry that declares capabilities of service-providing electronic agents currently active within the distributed computing environment;

interpreting a service request in order to determine a base goal that may be a compound, arbitrarily complex base goal, the service request adhering to an Interagent Communication Language (ICL), the act of interpreting including the sub-acts of:

determining any task completion advice provided by the base goal, and

determining any task completion constraints provided by the base goal;

constructing a base goal satisfaction plan including the sub-acts of:

determining whether the requested service is available,

determining sub-goals required in completing the base goal,

selecting service-providing electronic agents from the agent registry suitable for performing the determined sub-goals, and

ordering a delegation of sub-goal requests to best complete the requested service; and

implementing the base goal satisfaction plan.

30. A computer program as recited in claim 29 wherein the computer executable instruction for providing an agent registry includes the following computer executable instructions for registering a specific service-providing electronic agent into the agent registry:

establishing a bi-directional communications link between the specific agent and a facilitator agent controlling the agent registry;

providing a new agent profile to the facilitator agent, the new agent profile defining publicly available capabilities of the specific agent; and

registering the specific agent together with the new agent profile within the agent registry, thereby making available to the facilitator agent the capabilities of the specific agent.

31. A computer program as recited in claim 30 wherein the computer executable instruction for registering a specific agent further includes:

invoking the specific agent in order to activate the specific agent;

instantiating an instance of the specific agent; and

transmitting the new agent profile from the specific agent to the facilitator agent in response to the instantiation of the specific agent.

32. A computer program as recited in claim 29 wherein the computer executable instruction for providing an agent registry includes a computer executable instruction for removing a specific service-providing electronic agent from the registry upon determining that the specific agent is no longer available to provide services.

33. A computer program as recited in claim 29 wherein the provided agent registry includes a symbolic name, a unique address, data declarations, trigger declarations, task declarations, and process characteristics for each active agent.

34. A computer program as recited in claim 29 further including computer executable instructions for receiving the service request via a communications link established with a client.

35. A computer program as recited in claim 29 wherein the computer executable instruction for providing a service request includes instructions for:

receiving a non-ICL format service request;

selecting an active agent capable of converting the non-ICL formal service request into an ICL format service request;

forwarding the non-ICL format service request to the active agent capable of converting the non-ICL format service request, together with a request that such conversion be performed; and

receiving an ICL format service request corresponding to the non-ICL format service request.

36. A computer program as recited in claim 35 wherein the non-ICL format service request includes a natural language query, and the active agent capable of converting the non-ICL formal service request into an ICL format service request is a natural language agent.

37. A computer program as recited in claim 36 wherein the natural language query is generated by a user interface agent.

38. A computer program as recited in claim 29, the computer program further including computer executable instructions for implementing a base goal that requires setting a trigger having conditional and consequential functionality.

39. A computer program as recited in claim 38 wherein the trigger is an outgoing communications trigger, the computer program further including computer executable instructions for:

monitoring all outgoing communication events in order to determine whether a specific outgoing communication event has occurred; and

in response to the occurrence of the specific outgoing communication event, performing the particular action defined by the trigger.

40. A computer program as recited in claim 38 wherein the trigger is an incoming communications trigger, the computer program further including computer executable instructions for:

monitoring all incoming communication events in order to determine whether a specific incoming communication event has occurred; and

in response to the occurrence of the specific incoming communication event, performing the particular action defined by the trigger.

41. A computer program as recited in claim 38 wherein the trigger is a data trigger, the computer program further including computer executable instructions for:

monitoring a state of a data repository; and

in response to a particular state event, performing the particular action defined by the trigger.

42. A computer program as recited in claim 38 wherein the trigger is a time trigger, the computer program further including computer executable instructions for:

monitoring for the occurrence of a particular time condition; and

in response to the occurrence of the particular time condition, performing the particular action defined by the trigger.

43. A computer program as recited in claim 38 further including computer executable instructions for installing and executing the trigger within the facilitator agent.

44. A computer program as recited in claim 38 further including computer executable instructions for installing and executing the trigger within a first service-providing agent.

45. A computer program as recited in claim 29 further including computer executable instructions for interpreting compound goals having sub-goals separated by operators.

46. A computer program as recited in claim 45 wherein the type of available operators includes a conjunction operator, a disjunction operator, and a conditional execution operator.

47. A computer program as recited in claim 46 wherein the type of available operators further includes a parallel disjunction operator that indicates that disjunct goals are to be performed by different agents.

Sub a2

~~48.~~ An Interagent Communication Language (ICL) providing a basis for facilitated cooperative task completion within a distributed computing environment having a facilitator agent and a plurality of autonomous service-providing electronic agents, the ICL enabling agents to perform queries of other agents, exchange information with other agents, set triggers within other agents, an ICL syntax supporting compound goal expressions such that goals within a single request provided according to the ICL syntax may be coupled by a conjunctive operator, a disjunctive operator, a conditional execution operator, and a parallel disjunctive operator parallel disjunctive operator that indicates that disjunct goals are to be performed by different agents.

49. An ICL as recited in claim 48, wherein the ICL is computer platform independent.

50. An ICL as recited in claim 48 wherein the ICL is independent of computer programming languages which the plurality of agents are programmed in.

51. An ICL as recited in claim 48 wherein the ICL syntax supports explicit task completion constraints within goal expressions.

52. An ICL as recited in claim 51 wherein possible types of task completion constraints include use of specific agent constraints and response time constraints.

53. An ICL as recited in claim 51 wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

54. An ICL as recited in claim 48 wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

55. An ICL as recited in claim 48 wherein each autonomous service-providing electronic agent defines and publishes a set of capability declarations or solvables, expressed in ICL, that describes services provided by such electronic agent.

56. An ICL as recited in claim 55 wherein an electronic agent's solvables define an interface for the electronic agent.

57. An ICL as recited in claim 56 wherein the facilitator agent maintains an agent registry making available a plurality of electronic agent interfaces.

58. An ICL as recited in claim 57 wherein the possible types of solvables includes procedure solvables, a procedure solvable operable to implement a procedure such as a test or an action.

59. An ICL as recited in claim 58 wherein the possible types of solvables further includes data solvables, a data solvable operable to provide access to a collection of data.

60. An ICL as recited in claim 58 wherein the possible types of solvables includes data solvables, a data solvable operable to provide access to a collection of data.

~~61.~~ A facilitator agent arranged to coordinate cooperative task completion within a distributed computing environment having a plurality of autonomous service-providing electronic agents, the facilitator agent comprising:

an agent registry that declares capabilities of service-providing electronic agents currently active within the distributed computing environment; and

a facilitating engine operable to parse a service request in order to interpret a compound goal set forth therein, the compound goal including both local and global constraints and control parameters, the service request formed according to an Interagent Communication Language (ICL), the facilitating engine further operable to construct a goal satisfaction plan specifying the coordination of a suitable delegation of sub-goal requests to complete the requested service satisfying both the local and global constraints and control parameters.

62. A facilitator agent as recited in claim 61, wherein the facilitating engine is capable of modifying the goal satisfaction plan during execution, the modifying initiated by events such as new agent declarations within the agent registry, decisions made by remote agents, and information provided to the facilitating engine by remote agents.

63. A facilitator agent as recited in claim 61 wherein the agent registry includes a symbolic name, a unique address, data declarations, trigger declarations, task declarations, and process characteristics for each active agent.

64. A facilitator agent as recited in claim 61 wherein the facilitating engine is operable to install a trigger mechanism requesting that a certain action be taken when a certain set of conditions are met.

65. A facilitator agent as recited in claim 64 wherein the trigger mechanism is a communication trigger that monitors communication events and performs the certain action when a certain communication event occurs.

66. A facilitator agent as recited in claim 64 wherein the trigger mechanism is a data trigger that monitors a state of a data repository and performs the certain action when a certain data state is obtained.

67. A facilitator agent as recited in claim 66 wherein the data repository is local to the facilitator agent.

68. A facilitator agent as recited in claim 66 wherein the data repository is remote from the facilitator agent.

69. A facilitator agent as recited in claim 64 wherein the trigger mechanism is a task trigger having a set of conditions.

70. A facilitator agent as recited in claim 61, the facilitator agent further including a global database accessible to at least one of the service-providing electronic agents.

71. A software-based, flexible computer architecture for communication and cooperation among distributed electronic agents, the architecture contemplating a distributed computing system comprising:

a plurality of service-providing electronic agents; and

a facilitator agent in bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including:

an agent registry that declares capabilities of service-providing electronic agents currently active within the distributed computing environment;

a facilitating engine operable to parse a service request in order to interpret an arbitrarily complex goal set forth therein, the facilitating engine further operable to construct a goal satisfaction plan including the coordination of a suitable delegation of sub-goal requests to best complete the requested service.

72. A computer architecture as recited in claim 71, wherein the basis for the computer architect is an Interagent Communication Language (ICL) enabling agents to perform queries of other agents, exchange information with other agents, and set triggers within other agents, the ICL further defined by an ICL syntax supporting compound goal expressions such that goals within a single request provided according to the ICL syntax may be coupled by a conjunctive operator, a disjunctive operator, a conditional execution operator, and a parallel disjunctive operator parallel disjunctive operator that indicates that disjunct goals are to be performed by different agents.

73. A computer architecture as recited in claim 72, wherein the ICL is computer platform independent.

74. A computer architecture as recited in claim 73 wherein the ICL is independent of computer programming languages in which the plurality of agents are programmed.

75. A computer architecture as recited in claim 73 wherein the ICL syntax supports explicit task completion constraints within goal expressions.

76. A computer architecture as recited in claim 75 wherein possible types of task completion constraints include use of specific agent constraints and response time constraints.

77. A computer architecture as recited in claim 75 wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

78. A computer architecture as recited in claim 73 wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

79. A computer architecture as recited in claim 73 wherein each autonomous service-providing electronic agent defines and publishes a set of capability declarations or solvables, expressed in ICL, that describes services provided by such electronic agent.

80. A computer architecture as recited in claim 79 wherein an electronic agent's solvables define an interface for the electronic agent.

81. A computer architecture as recited in claim 80 wherein the possible types of solvables includes procedure solvables, a procedure solvable operable to implement a procedure such as a test or an action.

~~82. A computer architecture as recited in claim 81 wherein the possible types of solvables further includes data solvables, a data solvable operable to provide access to a collection of data.~~

~~83. A computer architecture as recited in claim 82 wherein the possible types of solvables includes a data solvable operable to provide access to modify a collection of data.~~

~~84. A computer architecture as recited in claim 71 wherein the planning component of the facilitating engine are distributed across at least two computer processes.~~

87

~~85. A computer architecture as recited in claim 71 wherein the execution component of the facilitating engine is distributed across at least two computer processes.~~

~~86. A data wave carrier providing a transport mechanism for information communication in a distributed computing environment having at least one facilitator agent and at least one active client agent, the data wave carrier comprising a signal representation of an inter-agent language description of an active client agent's functional capabilities.~~

~~87. A data wave carrier as recited in claim 85, the data wave carrier further comprising a signal representation of a request for service in the inter-agent language from a first agent to a second agent.~~

~~88. A data wave carrier as recited in claim 85, the data wave carrier further comprising a signal representation of a goal dispatched to an agent for performance from a facilitator agent.~~

~~89. A data wave carrier as recited in claim 88 wherein a later state of the data wave carrier comprises a signal representation of a response to the dispatched goal including results and/or a status report from the agent for performance to the facilitator agent.~~

Software-Based Architecture for Communication and Cooperation Among Distributed Electronic Agents

ABSTRACT

A highly flexible, software-based architecture is disclosed for constructing distributed systems. The architecture supports cooperative task completion by flexible, dynamic configurations of autonomous electronic agents. Communication and cooperation between agents are brokered by one or more facilitators, which are responsible for matching requests, from users and agents, with descriptions of the capabilities of other agents. It is not generally required that a user or agent know the identities, locations, or number of other agents involved in satisfying a request, and relatively minimal effort is involved in incorporating new agents and "wrapping" legacy applications. Extreme flexibility is achieved through an architecture organized around the declaration of capabilities by service-providing agents, the construction of arbitrarily complex goals by users and service-requesting agents, and the role of facilitators in delegating and coordinating the satisfaction of these goals, subject to advice and constraints that may accompany them. Additional mechanisms and features include facilities for creating and maintaining shared repositories of data; the use of triggers to instantiate commitments within and between agents; agent-based provision of multi-modal user interfaces, including natural language; and built-in support for including the user as a privileged member of the agent community. Specialized embodiments providing enhanced scalability are also described.

DECLARATION AND POWER OF ATTORNEY
FOR ORIGINAL U.S. PATENT APPLICATION

Attorney's Docket No. SRI1P016

As a below-named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe that I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled: SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS, the specification of which is attached hereto.

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, CFR § 1.56.

A/C DXM

And I hereby appoint the law firm of Hickman ~~& Martine~~ Stephens & Coleman, LLP, including **Paul L. Hickman (Reg. No. 28, 516); L. Keith Stephens (Reg. No. 32,632); Brian R. Coleman (Reg. No. 39,145); Dawn L. Palmer (Reg. No. 41,238); Jerray Wei (Reg. No. 43,247); and Ian L. Cartier (Reg. No. 38,406)** as my principal attorneys to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith:

Send Correspondence To:

Brian R. Coleman
HICKMAN STEPHENS & COLEMAN, LLP
P.O. BOX 52037
Palo Alto, California 94303-0746

Direct Telephone Calls To: **Brian R. Coleman at telephone number (650) 470-7430**

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

Typewritten Full Name of Sole or First Inventor: Adam J. Cheyer
Citizenship: USA
Inventor's signature: Adam J. Cheyer
Date of Signature: 1/5/99
Residence: (City) Palo Alto
(State/Country) CA
Post Office Address: 757 Cereza Drive Palo Alto CA 94306

Typewritten Full Name of Second Inventor: David L. Martin
Citizenship: USA
Inventor's signature: David L. Martin
Date of Signature: 1/5/99
Residence: (City) Santa Clara
(State/Country) CA
Post Office Address: 167 CRONIN DR. Santa Clara, CA 95051

16

PATENT APPLICATION FEE DETERMINATION RECORD
Effective November 10, 1998

Application or Docket Number
09/225,198

CLAIMS AS FILED - PART I

	(Column 1)		(Column 2)	SMALL ENTITY TYPE □		OR	OTHER THAN SMALL ENTITY	
FOR	NUMBER FILED		NUMBER EXTRA	RATE	FEE		RATE	FEE
BASIC FEE					380.00	OR		760.00
TOTAL CLAIMS	89	minus 20=	* 69	X$ 9=		OR	X$18=	1242
INDEPENDENT CLAIMS	6	minus 3 =	* 3	X39=		OR	X78=	234
MULTIPLE DEPENDENT CLAIM PRESENT				+130=		OR	+260=	
				TOTAL		OR	TOTAL	2236

* If the difference in column 1 is less than zero, enter "0" in column 2

CLAIMS AS AMENDED - PART II

111

AMENDMENT A		(Column 1)		(Column 2)	(Column 3)	SMALL ENTITY		OR	OTHER THAN SMALL ENTITY	
		CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE	ADDITIONAL FEE		RATE	ADDITIONAL FEE
	Total	* 89	Minus	** 20	= 69	X$ 9=		OR	69 X$18=	1242
	Independent	* 6	Minus	*** 3	= 3	X39=		OR	3 X78=	234.00
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					+130=		OR	+260=	
						TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

AMENDMENT B		(Column 1)		(Column 2)	(Column 3)					
		CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE	ADDITIONAL FEE		RATE	ADDITIONAL FEE
	Total	*	Minus	**	=	X$ 9=		OR	X$18=	
	Independent	*	Minus	***	=	X39=		OR	X78=	
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					+130=		OR	+260=	
						TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

AMENDMENT C		(Column 1)		(Column 2)	(Column 3)					
		CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE	ADDITIONAL FEE		RATE	ADDITIONAL FEE
	Total	*	Minus	**	=	X$ 9=		OR	X$18=	
	Independent	*	Minus	***	=	X39=		OR	X78=	
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					+130=		OR	+260=	
						TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If th "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20."
*** If th "Highest Number Previously Paid F r" IN THIS SPACE is less than 3, enter "3."
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in th appropriate box in column 1.

Best Available Copy

PATENT APPLICATION SERIAL NO. ______________________

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
FEE RECORD SHEET

```
01/19/1999 HVILLARI 00000027 500384   09225198

01 FC:101          760.00 CH
02 FC:102          234.00 CH
03 FC:103         1242.00 CH
```

ARTIFACT SHEET

Enter artifact number below. Artifact number is application number + artifact type code (see list below) + sequential letter (A, B, C ...). The first artifact folder for an artifact type receives the letter A, the second B, etc.. Examples: 59123456PA, 59123456PB, 59123456ZA, 59123456ZB

09225198PA

Indicate quantity of a single type of artifact received but not scanned. Create individual artifact folder/box and artifact number for each Artifact Type.

☑ CD(s) containing:

computer program listing ☑
Doc Code: Computer Artifact Type Code: P

pages of specification
and/or sequence listing ☐
and/or table
Doc Code: Artifact Artifact Type Code: S

content unspecified or combined ☐
Doc Code: Artifact Artifact Type Code: U

☐ Stapled Set(s) Color Documents or B/W Photographs
Doc Code: Artifact Artifact Type Code: C

☐ Microfilm(s)
Doc Code: Artifact Artifact Type Code: F

☐ Video tape(s)
Doc Code: Artifact Artifact Type Code: V

☐ Model(s)
Doc Code: Artifact Artifact Type Code: M

☐ Bound Document(s)
Doc Code: Artifact Artifact Type Code: B

☐ Confidential Information Disclosure Statement or Other Documents marked Proprietary, Trade Secrets, Subject to Protective Order, Material Submitted under MPEP 724.02, etc.
Doc Code: Artifact Artifact Type Code X

☐ Other, description: ______________________
Doc Code: Artifact Artifact Type Code: Z

March 8, 2004

ARTIFACT SHEET

Enter artifact number below. Artifact number is application number + artifact type code (see list below) + sequential letter (A, B, C ...). The first artifact folder for an artifact type receives the letter A, the second B, etc.. Examples: 59123456PA, 59123456PB, 59123456ZA, 59123456ZB

09 225 198 PB

Indicate quantity of a single type of artifact received but not scanned. Create individual artifact folder/box and artifact number for each Artifact Type.

☑ CD(s) containing:
computer program listing ☑
Doc Code: Computer Artifact Type Code: P
pages of specification
and/or sequence listing
and/or table □
Doc Code: Artifact Artifact Type Code: S
content unspecified or combined □
Doc Code: Artifact Artifact Type Code: U

□ Stapled Set(s) Color Documents or B/W Photographs
Doc Code: Artifact Artifact Type Code: C

□ Microfilm(s)
Doc Code: Artifact Artifact Type Code: F

□ Video tape(s)
Doc Code: Artifact Artifact Type Code: V

□ Model(s)
Doc Code: Artifact Artifact Type Code: M

□ Bound Document(s)
Doc Code: Artifact Artifact Type Code: B

□ Confidential Information Disclosure Statement or Other Documents marked Proprietary, Trade Secrets, Subject to Protective Order, Material Submitted under MPEP 724.02, etc.
Doc Code: Artifact Artifact Type Code X

□ Other, description: ______________________
Doc Code: Artifact Artifact Type Code: Z

March 8, 2004

C-82755

PATENT #2 RS

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the application of:)
Cheyer et al.)
Application No.: 09/225,198)
Filed: January 5, 1999)
For: SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS)

Group: 2755
Examiner: Unassigned
Atty. Docket No.: SRI1P016
Date: May 11, 1999

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, DC 20231 on May 11, 1999

Signed: Jay Vasudevan
Jay Vasudevan

INFORMATION DISCLOSURE STATEMENT UNDER 37 CFR §§1.56 AND 1.97(c)

Assistant Commissioner for Patents
Washington, DC 20231

Dear Sir:

The references listed in the attached PTO Form 1449, copies of which are attached, may be material to examination of the above-identified patent application. Applicants submit these references in compliance with their duty of disclosure pursuant to 37 CFR §§1.56 and 1.97. The Examiner is requested to make these references of official record in this application.

Reference No. R on Page 4 of PTO form 1449 contains documents downloaded from a web site owned by Dejima, Inc. at http://www.dejima.com on April 29, 1999 and March 18, 1999. The applicant makes no representation that this web site has not changed between the dates of downloading or that this web site will not change in the future.

This Information Disclosure Statement is not to be construed as a representation that a search has been made, that additional information material to the examination of this application does not exist, or that these references indeed constitute prior art.

This Information Disclosure Statement is believed to be filed before the mailing date of a first Office Action on the merits. Accordingly, it is believed that no fees are due in connection with the filing of this Information Disclosure Statement. However, if it is determined that any fees are due, the Commissioner is hereby authorized to charge such fees to Deposit Account 50-0384 (Order No. SRI1P016).

Respectfully submitted,
HICKMAN STEPHENS & COLEMAN, LLP

Brian R. Coleman
Reg. No. 39,145

P.O. Box 52037
Palo Alto, CA 94303-0746
Telephone: (650) 470-7430

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/225,198	01/05/1999	ADAM J. CHEYER	SRI1P016	2756

25696 7590 07/17/2002

OPPENHEIMER WOLFF & DONNELLY
P. O. BOX 10356
PALO ALTO, CA 94303

EXAMINER
BULLOCK JR, LEWIS ALEXANDER

ART UNIT	PAPER NUMBER
2151	

DATE MAILED: 07/17/2002

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 07-01)

Office Action Summary	**Application No.** 09/225,198	**Applicant(s)** CHEYER ET AL.	
	Examiner Lewis A. Bullock, Jr.	**Art Unit** 2151	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)□ Responsive to communication(s) filed on _____ .

2a)□ This action is **FINAL**. 2b)☒ This action is non-final.

3)□ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-89 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)□ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-89 is/are rejected.

7)□ Claim(s) _____ is/are objected to.

8)□ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)□ The specification is objected to by the Examiner.

10)□ The drawing(s) filed on _____ is/are: a)□ accepted or b)□ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)□ The proposed drawing correction filed on _____ is: a)□ approved b)□ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)□ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)□ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)□ All b)□ Some * c)□ None of:

1.□ Certified copies of the priority documents have been received.

2.□ Certified copies of the priority documents have been received in Application No. _____ .

3.□ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)□ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) □ The translation of the foreign language provisional application has been received.

15)□ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☒ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO-1449) Paper No(s) 2 .

4) □ Interview Summary (PTO-413) Paper No(s). _____ .

5) □ Notice of Informal Patent Application (PTO-152)

6) □ Other:

DETAILED ACTION

Claim Rejections - 35 USC § 112

1. Claim 2 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Applicant claims the recursively applying the last step of claim 1, however the Examiner cannot determine which step applicant is referring to. Applicant is either referring to the dynamically interpreting step and its substep or the dispatching step of the dynamically interpreting step. Clarification is requested.

2. Claim 3 recites the limitation "from the specific agent to the facilitator agent" in lines 5-6. There is insufficient antecedent basis for this limitation in the claim. There is no mention of the facilitator agent anywhere in the parent claims. In review of the specification the examiner finds the facilitator agent performs the steps of claim 1, however, claim 1 does not detail the facilitator agent as performing the steps. The examiner request Applicant to amend claim 1 to detail that the facilitator agent performs the functionality.

3. Claims 84 and 85 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Claims 84 and 85 recite the planning and execution components, however neither component has antecedent basis in the parent claim 71. Correction is requested.

4. Claims 87 and 88 recite the limitation "A data wave carrier as recited in claim 85" in line 1. There is insufficient antecedent basis for this limitation in the claim. Claims 87 and 88 should be dependent on claim 86 not claim 85 and are further examined as such.

Claim Rejections - 35 USC § 102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

6. Claims 1, 2, 5-11, 15-28, 48-89 are rejected under 35 U.S.C. 102(a) as being anticipated by “Building Distributed Software Systems with the Open Agent Architecture” by MARTIN.

As to claim 1, MARTIN teaches a computer-implemented method for communication and cooperative task completion among a plurality of distributed agents (application agent / meta agent / user interface agent), comprising the acts of: registering a description of each client agent’s functional capabilities (capabilities specifications), using a platform independent inter-agent language (ICL); receiving a request for service as a base goal (goals created by requesters of service) in the inter-agent language, in the form of an arbitrarily complex goal expression; and dynamically interpreting the goal expression (goals) (via facilitator) comprising: generating one or more sub-goals using the inter-agent language; and dispatching each of the sub-goals to a selected client agent (service providers) for performance, based on a match between the sub-goal being dispatched and the registered functional capabilities of the selected client agent (pg. 7, Mechanisms of Cooperation; pg. 12-14, Requesting Services; Refining Service Requests, and Facilitation).

As to claim 2, MARTIN teaches receiving a new request (subgoal) for service as a base goal from at least one of the selected client agents in response to the sub-goal and recursively applying the dynamically interpreting (pg. 13, Refining Service Requests).

As to claims 5-10, MARTIN teaches providing an agent registry data structure that can comprise of symbolic names, data declarations, trigger declarations, and task and process characteristics (pg. 13-14, Facilitation; pg. 7, "In processing a request...it can use ICL to request services of other agents, set triggers, and read or write shared data on the facilitator...").

As to claim 11, MARTIN teaches establishing communication between distributed agents (pg. 6, The facilitator is a specialized server agent that is responsible for coordinating agent communications and cooperative problem-solving.").

As to claims 15-25, MARTIN teaches the base goal requires setting a trigger having conditional functionality and consequential functionality which can be stored on the facilitator agent and/or the service providing agent (pgs. 16-17, Autonomous Monitoring Using Triggers).

As to claims 26-28, MARTIN teaches the base goal is a compound goal having sub-goals separated by operators, i.e. conjuction operator, disjunction operator, conditional operator, and a parallel operator (pg. 12-13, Compound goals).

As to claim 48, MARTIN teaches an Inter-agent Communication Language (ICL) providing a basis for facilitated cooperative task completion within a distributed computing environment having a facilitator agent (facilitator) and a plurality of electronic agents (service providing agents / service requesting agents), the ICL enabling agents to perform queries of other agents, exchange information with other agents, set triggers within other agents (pgs. 4-7, Overview of OAA System Structure, Mechanisms of Cooperation; pg. 8, "OAA agents employ ICL to perform queries, execute actions, exchange information, set triggers, and manipulate data in the agent community."), an ICL syntax supporting compound goal expressions such that goals within a single request provided according to the ICL syntax may be coupled by a conjunctive operator, a disjunctive operator, a conditional execution operator, and a parallel operator that indicates that goals are to be performed by different agents (pg. 12, Compound goals).

As to claim 49 and 50, MARTIN teaches the ICL is platform and language independent (pg. 8, "OAA's Inter-agent Communication Language (ICL) is the interface, communication, and task coordination language shared by all agents, regardless of what platform they run on or what computer language they are programmed in.").

As to claims 51-54, MARTIN teaches the ICL supports task completion constraints within goal expressions (pg. 9, "A number of important declarations...we consider each of these elements.").

As to claims 55-60, MARTIN teaches each electronic agent defines and publishes a set of capability declarations or solvables that describe services and an interface to the electronic agent (pg. 9, "A number of important declarations...we consider each of these elements.").

As to claims 61 and 62, reference is made to an agent that performs the method of claim 1 above and is therefore met by the rejection of claim 1 above. However, claim 61 further details an agent register and the construction of a goal satisfaction plan. MARTIN teaches an agent register (knowledge base) (pg. 13-14, Facilitation); and the construction of a goal satisfaction plan (pg. 13, "When a facilitator receives a compound goal, its job is to construct a goal satisfaction plan and oversee its satisfaction in the most appropriate, efficient manner that is consistent with the specified advice.").

As to claim 63, refer to claim 5 for rejection.

As to claim 64-69, refer to claims 15-25 for rejection.

As to claim 70, MARTIN teaches the agent registry (knowledge base) is a database accessible to all electronic agents (via the facilitator) (pg. 13-14, Facilitation).

As to claim 71, reference is made to an architecture that encompasses the agent of claim 61 above, and is therefore met by the rejection of claim 61 above. However claim 71, further details the facilitator agent in bi-directional communication with the electronic agents. MARTIN teaches the facilitator agent in bi-directional communication with the electronic agents (fig 1).

As to claim 72, refer to claim 48 for rejection.

As to claims 73 and 74, refer to claims 49 and 50 for rejection.

As to claims 75-78, refer to claims 51-54 for rejection.

As to claims 79-83, refer to claims 54-60 for rejection.

As to claims 84 and 85, MARTIN teaches the facilitating engine is distributed across at least two processes (pg. 6, "Larger systems can be assembled from multiple facilitator/client groups…").

As to claim 86, MARTIN teaches a data wave carrier (system) providing a transport mechanism (layer of conversational protocol / communication functions) for information communication in a distributed computing environment having at least one facilitator agent (facilitator) and at least one client agent (application agent / user interface agent), the carrier comprising a signal representation of an inter-agent language description of a client agent's functional capabilities (registering by the service provider agents) (pg. 6-9).

As to claim 87, MARTIN teaches a signal representation of a request for service in the inter-agent language from a first agent to a second agent (request for service from an service requesting agent to the facilitator) (pg. 12, Requesting Services).

As to claim 88, MARTIN teaches a signal representation of a goal dispatched to an agent for performance from a facilitator agent (pg. 13-14, Facilitation).

As to claim 89, MARTIN teaches a signal representation of a response to the dispatched goal including results and/or a status report from the agent for performance to the facilitator agent (pg. 13-14, Facilitation).

7. Claims 1, 2, 5-11, and 15-25 are rejected under 35 U.S.C. 102(b) as being anticipated by "Development Tools for the Open Agent Architecture" by MARTIN.

As to claim 1, MARTIN teaches a computer-implemented method for communication and cooperative task completion among a plurality of distributed agents (sub-agents / agents), comprising the acts of: registering a description of each client agent's functional capabilities, using a platform independent inter-agent language (pg. 5, Each facilator records the published capabilities of their subagents..."); receiving a request as a base goal in the inter-agent language (ICL form), in the form of an arbitrarily complex goal expression; and dynamically interpreting the goal expression comprising: generating one or more sub-goals using the inter-agent language; and dispatching each of the sub-goals to a selected client agent for performance ("pg. 5, "...and when requests arrive (expressed in the Inter-agent Communication Language, described below), the facilitator is responsible for breaking them down and for distributing sub-requests to the appropriate agents; "For example, every agent can...and request solutions for a set of goals,...").

As to claim 2, MARTIN teaches receiving a new request for service as a base goal from at least one of the selected client agents in response to the sub-goal and recursively applying the dynamically interpreting (pg. 5, "An agent satisfying a request may require supporting information, and the OAA provides numerous means of requesting data from other agents or from the user.").

As to claims 5-10, MARTIN teaches providing an agent registry data structure that can comprise of symbolic names, data declarations, trigger declarations, and task and process characteristics (pg. 5, "For example, every agent can install local or remote triggers on data..").

As to claim 11, MARTIN teaches establishing communication between distributed agents (pg. 5, ...the facilitator is responsible for breaking them down and for distributing sub-requests to the appropriate agent.").

As to claims 15-25, MARTIN teaches the base goal requires setting a trigger having conditional functionality and consequential functionality which can be stored on the facilitator agent and/or the service providing agent (pg. 5, "For example, every agent can install local or remote triggers on data..").

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

9. Claims 3, 29-34, and 38-47 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Building Distributed Software Systems with the Open Agent Architecture" by MARTIN.

As to claim 3, MARTIN teaches the act of registering and transmitting the new agent profile from the specific agent to the facilitator agent (pg. 7, "When invoked, a client agent makes a connection to a facilitator...an agent informs its parent facilitator of the services it is capable of providing."). It would be obvious that an agent that is initially created is instantiated in memory before it is registered.

As to claim 29, MARTIN teaches a method to facilitate cooperative task completion within a distributed computing environment supporting an Inter-agent Communication Language among a plurality of electronic agents (fig 1) comprising: providing an agent registry (knowledge base) as disclosed (pg. 13-14, Facilitation); interpreting a service request in order to determine a base goal (compound goal) comprising: determining any task completion advice provided by the base goal, and determining any task completion constraints provided by the base goal (pg. 14, "It may also use strategies or advice specified by the requester.."); constructing a base goal satisfaction plan (pg. 13, "When a facilitator receives a compound goal, its job is to construct a goal satisfaction plan and oversee its satisfaction in the most appropriate, efficient manner that is consistent with the specified advice.") comprising: determining whether the requested service is available, determining sub-goals required in completing the base goal (delegation), selecting suitable service-providing electronic agents for performing the sub-goals, and ordering a delegation of sub-goal requests to complete the requested service; and implementing the base goal satisfaction plan (pg. 13-14, Facilitation). However, MARTIN does not explicitly mention that the method is operable in a computer program product. It would be obvious to one skilled in the art to generate program code that would entail the method of Martin and thereby obvious that the method can be entailed in a computer program product.

As to claims 30 and 31, MARTIN teaches registering a specific agent (service provider agents) into the agent registry comprising: establishing a bi-directional communications link between the specific agent and a facilitator agent (facilitator) controlling the agent registry; providing a new agent profile to the facilitator agent; and registering the specific agent with the profile thereby making the capabilities available to the facilitator agent (pgs. 9-10, Providing Services; pg. 7, Mechanisms of Cooperation).

As to claim 32, refer to claim 3 for rejection.

As to claim 33, refer to claim 5 for rejection.

As to claim 34, refer to claim 11 for rejection.

As to claims 38-44, refer to claims 15-25 for rejection.

As to claims 45-47, refer to claims 26-28 for rejection.

10. Claims 4, 12-14 and 35-37 is rejected under 35 U.S.C. 103(a) as being unpatentable over "Building Distributed Software Systems with the Open Agent Architecture" by MARTIN1 in view of "Information Brokering in an Agent Architecture" by MARTIN2.

As to claim 4, MARTIN1 substantially discloses the invention above. However, MARTIN1 does not explicitly mention the cited limitation. MARTIN2 teaches deactivating a client agent no longer available to provide services by deleting the registration (pg. 9, Source agents that need to go offline...so that it can unregister the source and retract its schema mapping rules."). Therefore it would be obvious to combine the teachings of MARTIN1 with the teachings of MARTIN2 in order to provide transparent access to a plurality of independent agents (abstract).

As to claims 12-14, MARTIN1 substantially discloses the invention above. However, MARTIN1 does not explicitly mention the cited limitation. MARTIN2 teaches receiving a request for service in a second language (source shema); selecting a registered agent capable of converting the second language into the inter-agent language (broker schema); and forwarding the request for service in a second language to the registered agent for conversion to be performed and the results returned (pg. 12-13, Queries Expressed in a Source Schema). Refer to claim 4 for the motivation to combine.

As to claims 35-37, refer to claims 12-14 for rejection.

11. Claims 3, 29-34, 38-47, 61-71, and 84-89 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Developing Tools for the Open Agent Architecture" by MARTIN.

As to claim 3, MARTIN teaches the act of registering and transmitting the new agent profile from the specific agent to the facilitator agent (pg. 5, "Every agent participating in an OAA-based system defines and publishes a set of capabilities specifications, expressed in the ICL, describing the services that it provides."). It would be obvious that an agent that is initially created is instantiated in memory before it is registered.

As to claim 29, MARTIN teaches a method to facilitate cooperative task completion within a distributed computing environment supporting an Inter-agent Communication Language among a plurality of electronic agents (sub-agents / agents) comprising: providing an agent registry as disclosed (facilitator storage of published sub-agents capabilities); interpreting a service request in order to determine a base goal (via facilitator) constructing a base goal satisfaction plan comprising: determining whether the requested service is available, determining sub-goals required in completing the base goal (determine solutions for a set of goals) selecting suitable service-providing electronic agents for performing the sub-goals, and ordering a delegation of sub-goal requests to complete the requested service; and implementing the base goal satisfaction plan (pg. 5, "The facilitator is responsible for breaking them down and for distributing sub-requests to the appropriate agents."). However, MARTIN does not explicitly mention that the method is operable in a computer program product or the sending of advice or constraints. It would be obvious that since an agent can request solutions for a goal to be satisfied under a variety of different control strategies (pg. 5) that the control strategies are the advice and/or constraints. It would also be obvious to one skilled in the art to generate program code that would entail the method of Martin and thereby obvious that the method can be entailed in a computer program product.

As to claims 30 and 31, MARTIN teaches registering a specific agent (agent) into the agent registry (list of agents capabilities) comprising: establishing a bi-directional communications link between the specific agent and a facilitator agent controlling the agent registry; providing a new agent profile to the facilitator agent; and registering the specific agent with the profile thereby making the capabilities available to the facilitator agent (pg. 5, "Each facilitator records the published capabilities of their subagents..."; "Every agent participating in an OAA-based system...describing the services that it provides.").

As to claim 32, refer to claim 3 for rejection.

As to claim 33, refer to claim 5 for rejection.

As to claim 34, refer to claim 11 for rejection.

As to claims 38-44, refer to claims 15-25 for rejection.

As to claims 45-47, refer to claims 26-28 for rejection.

As to claim 61 and 62, reference is made to an agent that performs the method of claim 1 above and is therefore met by the rejection of claim 1 above. However, claim 61 further details an agent register and the construction of a goal satisfaction plan. MARTIN teaches every agent participating in an OAA-based system defines and publishes a set of capabilities describing the services that it provides and that the facilitator records these published capabilities (pg. 5). Therefore, there is an agent register of the capabilities of each agent. MARTIN also teaches an agent can request solutions for a set of goals to be satisfied under a variety of different control strategies. It would be obvious that since solutions are determined based on the goals and control strategies that a goal satisfaction plan is created.

As to claim 63, refer to claim 5 for rejection.

As to claim 64-69, refer to claims 15-25 for rejection.

As to claim 70, MARTIN teaches the agent registry (agent library / list of agent capabilities) is a database accessible to all electronic agents (pg. 5, A collection of agents satisfies requests from users, or other agents...one or more facilitators."; "An agent satisfying a request may require supporting information...requesting data from other agents or from the user.").

As to claim 71, reference is made to an architecture that encompasses the agent of claim 61 above, and is therefore met by the rejection of claim 61 above. However claim 71, further details the facilitator agent in bi-directional communication with the electronic agents. MARTIN teaches the facilitator can distribute request to the agents and the agents can request information via the facilitator (pg. 5), therefore it would be obvious that the facilitator and agents are in bi-directional communication.

As to claims 84 and 85, MARTIN teaches the facilitating engine is distributed across at least two processes (pg. 5, "Facilitators can, in turn, be connected as clients of other facilitators.").

As to claim 86, MARTIN teaches system for information communication in a distributed computing environment having at least one facilitator agent (facilitator) and at least one client agent (sub-agent / agents), the carrier comprising a signal representation of an inter-agent language description (ICL registration of capabilities) of a client agent's functional capabilities (pg. 5, "Each facilitator records the published capabilities of their subagents.."). It would be obvious that the system has a data wave carrier and a transport mechanism for network communication.

As to claim 87, MARTIN teaches a signal representation of a request for service in the inter-agent language from a first agent (client agent sending a query) to a second agent (facilitator) (pg. 5).

As to claim 88, MARTIN teaches a signal representation of a goal dispatched to an agent for performance from a facilitator agent (every agent can request solutions for a set of goals / facilitator is responsible for breaking them down and for distributing sub-requests to the appropriate agent) (pg. 5).

As to claim 89, It is well known in the art to one skilled in the art that an agent can send back a response after processing the request.

12. Claims 4, 12-14, 26-28, 35-37, 48-60, 72-83 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Development Tools for the Open Agent Architecture" by MARTIN1 in view of "Information Brokering in an Agent Architecture" by MARTIN2.

As to claim 4, MARTIN1 substantially discloses the invention above. However, MARTIN1 does not explicitly mention the cited limitation. MARTIN2 teaches deactivating a client agent no longer available to provide services by deleting the registration (pg. 9, Source agents that need to go offline...so that it can unregister the source and retract its schema mapping rules."). Therefore it would be obvious to combine the teachings of MARTIN1 with the teachings of MARTIN2 in order to provide transparent access to a plurality of independent agents (abstract).

As to claims 12-14, MARTIN1 substantially discloses the invention above. However, MARTIN1 does not explicitly mention the cited limitation. MARTIN2 teaches receiving a request for service in a second language (source schema); selecting a registered agent capable of converting the second language into the inter-agent language (broker schema); and forwarding the request for service in a second language to the registered agent for conversion to be performed and the results returned (pg. 12-13, Queries Expressed in a Source Schema). Refer to claim 4 for the motivation to combine.

As to claims 26-28, MARTIN1 substantially discloses the invention above. However, MARTIN1 does not explicitly mention the cited limitation. MARTIN2 teaches the base goal is a compound goal having sub-goals (pg. 8, "Queries submitted to the Broker are expression...and backtracking in expressing and processing queries."). It would be obvious that since the base goal (query) is broken down and distributed to as sub-requests to the appropriate agents or solutions are requested for a set of goals as disclosed in MARTIN1 that the base goal as a compound goal is broken down based on operators disclosing where it can be broken down. Refer to claim 4 for the motivation to combine.

As to claims 35-37, refer to claims 12-14 for rejection.

As to claim 48, MARTIN1 teaches an Inter-agent Communication Language (ICL) providing a basis for facilitated cooperative task completion within a distributed computing environment having a facilitator agent (facilitator) and a plurality of electronic agents (sub-agents / agents), the ICL enabling agents to perform queries of other agents, exchange information with other agents, set triggers within other agents (pg. 5, Agents share a common communication language...and may run on any network linked platform."). However, MARTIN1 does not teach the ICL supporting compound goal expressions. MARTIN2 teaches the query is a base goal stored in as a compound goal having sub-goals (pg. 8, "Queries submitted to the Broker are expression...and backtracking in expressing and processing queries."). It would be obvious that since the base goal (query) is broken down and distributed to as sub-requests to the appropriate agents or solutions are requested for a set of goals as disclosed in MARTIN1 that the base goal as a compound goal is broken down based on operators disclosing where it can be broken down. Refer to claim 4 for the motivation to combine.

As to claim 49 and 50, MARTIN1 teaches the ICL is platform and language independent (pg. 5, "The OAA's Inter-agent Communication Language...they are programmed in.").

As to claims 51-54, MARTIN1 teaches the ICL supports task completion constraints (triggers) within goal expressions (pg. 5).

As to claims 54-60, MARTIN1 teaches each electronic agent defines and publishes a set of capability declarations or solvables that describe services and an interface to the electronic agent (pg. 5, "Every agent participating in an OAA-based system defines and publishes...we refer to these capabilities specifications as solvables.").

As to claim 72, refer to claim 48 for rejection.

As to claims 73 and 74, refer to claims 49 and 50 for rejection.

As to claims 75-78, refer to claims 51-54 for rejection.

As to claims 79-83, refer to claims 54-60 for rejection.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Lewis A. Bullock, Jr. whose telephone number is (703) 305-0439. The examiner can normally be reached on Monday-Friday, 8:30 am - 5:00 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Alvin E. Oberley can be reached on (703) 305-9716. The fax phone numbers for the organization where this application or proceeding is assigned are (703) 746-7239 for regular communications and (703) 746-7238 for After Final communications.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-0286.

ST. JOHN COURTENAY III
PRIMARY EXAMINER

***
July 11, 2002

Form PTO 948 (Rev. 8-98) U.S. DEPARTMENT OF COMMERCE - Patent and Trademark Office Application No. 09/225,198

NOTICE OF DRAFTSPERSON'S PATENT DRAWING REVIEW

The drawing(s) filed (insert date) 01/05/99 are:

A. ☑ approved by the Draftsperson under 37 CFR 1.84 or 1.152.

B. □ objected to by the Draftsperson under 37 CFR 1.84 or 1.152 for the reasons indicated below. The Examiner will require submission of new, corrected drawings when necessary. Corrected drawing must be sumitted according to the instructions on the back of this notice.

1. DRAWINGS. 37 CFR 1.84(a): Acceptable categories of drawings: Black ink. Color.
   ____ Color drawings are not acceptable until petiton is granted. Fig(s) ________
   ____ Pencil and non black ink not permitted. Fig(s) ________
2. PHOTOGRAPHS. 37 CFR 1.84 (b)
   ____ 1 full-tone set is required. Fig(s) ________
   ____ Photographs not properly mounted (must use brystol board or photographic double-weight paper). Fig(s) ________
   ____ Poor quality (half-tone). Fig(s) ________
3. TYPE OF PAPER. 37 CFR 1.84(e)
   ____ Paper not flexible, strong, white, and durable. Fig(s) ________
   ____ Erasures, alterations, overwritings, interlineations, folds, copy machine marks not accepted. Fig(s) ________
   ____ Mylar, velum paper is not acceptable (too thin). Fig(s) ________
4. SIZE OF PAPER. 37 CFR 1.84(f): Acceptable sizes:
   ____ 21.0 cm by 29.7 cm (DIN size A4)
   ____ 21.6 cm by 27.9 cm (8 1/2 x 11 inches)
   ____ All drawing sheets not the same size. Sheet(s) ________
   ____ Drawings sheets not an acceptable size. Fig(s) ________
5. MARGINS. 37 CFR 1.84(g): Acceptable margins:

   Top 2.5 cm Left 2.5cm Right 1.5 cm Bottom 1.0 cm
   SIZE: A4 Size
   Top 2.5 cm Left 2.5 cm Right 1.5 cm Bottom 1.0 cm
   SIZE: 8 1/2 x 11
   Margins not acceptable. Fig(s) ________
   ________ Top (T) ________ Left (L)
   ________ Right (R) ________ Bottom (B)
6. VIEWS. 37 CFR 1.84(h)
   REMINDER: Specification may require revision to correspond to drawing changes.
   Partial views. 37 CFR 1.84(h)(2)
   ____ Brackets needed to show figure as one entity. Fig(s) ________
   ____ Views not labeled separately or properly. Fig(s) ________
   ____ Enlarged view not labeled separetely or properly. Fig(s) ________
7. SECTIONAL VIEWS. 37 CFR 1.84 (h)(3)
   ____ Hatching not indicated for sectional portions of an object. Fig(s) ________
   ____ Sectional designation should be noted with Arabic or Roman numbers. Fig(s) ________
8. ARRANGEMENT OF VIEWS. 37 CFR 1.84(i)
   ____ Words do not appear on a horizontal, left-to-right fashion when page is either upright or turned so that the top becomes the right side, except for graphs. Fig(s) ________
9. SCALE. 37 CFR 1.84(k)
   ____ Scale not large enough to show mechanism without crowding when drawing is reduced in size to two-thirds in reproduction. Fig(s) ________
10. CHARACTER OF LINES, NUMBERS, & LETTERS. 37 CFR 1.84(i)
    ____ Lines, numbers & letters not uniformly thick and well defined, clean, durable, and black (poor line quality). Fig(s) ________
11. SHADING. 37 CFR 1.84(m)
    ____ Solid black areas pale. Fig(s) ________
    ____ Solid black shading not permitted. Fig(s) ________
    ____ Shade lines, pale, rough and blurred. Fig(s) ________
12. NUMBERS, LETTERS, & REFERENCE CHARACTERS. 37 CFR 1.84(p)
    ____ Numbers and reference characters not plain and legible. Fig(s) ________
    ____ Figure legends are poor. Fig(s) ________
    ____ Numbers and reference characters not oriented in the same direction as the view. 37 CFR 1.84(p)(1) Fig(s) ________
    ____ English alphabet not used. 37 CFR 1.84(p)(2) Figs ________
    ____ Numbers, letters and reference characters must be at least .32 cm (1/8 inch) in height. 37 CFR 1.84(p)(3) Fig(s) ________
13. LEAD LINES. 37 CFR 1.84(q)
    ____ Lead lines cross each other. Fig(s) ________
    ____ Lead lines missing. Fig(s) ________
14. NUMBERING OF SHEETS OF DRAWINGS. 37 CFR 1.84(t)
    ____ Sheets not numbered consecutively, and in Arabic numerals beginning with number 1. Sheet(s) ________
15. NUMBERING OF VIEWS. 37 CFR 1.84(u)
    ____ Views not numbered consecutively, and in Arabic numerals, beginning with number 1. Fig(s) ________
16. CORRECTIONS. 37 CFR 1.84(w)
    ____ Corrections not made from prior PTO-948 dated ________
17. DESIGN DRAWINGS. 37 CFR 1.152
    ____ Surface shading shown not appropriate. Fig(s) ________
    ____ Solid black shading not used for color contrast. Fig(s) ________

COMMENTS

REVIEWER LAM DATE 02/18/99 TELEPHONE NO. ________

ATTACHMENT TO PAPER NO. 3

Notice of References Cited	Application/Control No. 09/225,198	Applicant(s)/Patent Under Reexamination CHEYER ET AL.	
	Examiner Lewis A. Bullock, Jr.	Art Unit 2151	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-6,338,081	01-2002	Furusawa et al.	709/202
	B	US-5,960,404	09-1999	Chaar et al.	705/11
	C	US-6,216,173	04-2001	Jones et al.	135/77
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	Cheyer, Adam. "Mechanisms of Cooperation." October 19, 1998.
	V	DeVoe, Deborah. "SRI distributed agents promise flexibility." InfoWorld. December 30 1996.
	W	Sycara, Katia et al. "Distributed Intelligent Agents." IEEE. December 1996.
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Form 1449 (Modified) **Information Disclosure Statement By Applicant** (Use Several Sheets if Necessary)	Atty Docket No. SRI1P016 Applicant: Cheyer et al. Filing Date: January 5. 1999	Serial No.: 09/225,198 Group 2755

U.S. Patent Documents

Examiner Initial	No.	Patent No.	Date	Patentee	Class	Sub-class	Filing Date
	A						
	B						
	C						
	D						
	E						
	F						
	G						
	H						
	I						
	J						
	K						

Foreign Patent or Published Foreign Patent Application

Examiner Initial	No.	Document No.	Publication Date	Country or Patent Office	Class	Sub-class	Translation Yes	Translation No
	L							
	M							
	N							
	O							
	P							

Other Documents

Examiner Initial	No.	Author, Title, Date, Place (e.g. Journal) of Publication
LAB	R	MORAN, Douglas B. and CHEYER, Adam J., "Intelligent Agent-based User Interfaces", Article Intelligence center, SRI International
LAB	S	MARTIN, David L., CHEYER, Adam J. and MORAN, Douglas B., "Building Distributed Software Systems with the Open Agent Architecture"
LAB	T	COHEN, Philip R. and CHEYER, Adam, SRI International, WANG, Michelle, Stanford University, BAEG, Soon Cheol, ETRI, "An Open Agent Architecture"

Examiner Lewis A Bullock Jr	Date Considered 7/11/02

Examiner: Initial citation considered. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Form 1449 (Modified) **Information Disclosure Statement By Applicant** (Use Several Sheets if Necessary)	Atty Docket No. SRI1P016 Applicant: Cheyer et al. Filing Date: January 5. 1999	Serial No.: 09/225,198 Group 2755

U.S. Patent Documents

Examiner Initial	No.	Patent No.	Date	Patentee	Class	Sub-class	Filing Date
	A						
	B						
	C						
	D						
	E						
	F						
	G						
	H						
	I						
	J						
	K						

Foreign Patent or Published Foreign Patent Application

Examiner Initial	No.	Document No.	Publication Date	Country or Patent Office	Class	Sub-class	Translation	
							Yes	No
	L							
	M							
	N							
	O							
	P							

Other Documents

Examiner Initial	No.	Author, Title, Date, Place (e.g. Journal) of Publication
	R	JULIA, Luc E. and CHEYER, Adam J., SRI International "Cooperative Agents and Recognition Systems (CARS) for Drivers and Passengers",
	S	MORAN, Douglas B., CHEYER, Adam J., JULIA, Luc E., MARTIN, David L., SRI International, and PARK, Sangkyu, Electronics and Telecommunications Research Institute, "Multimodal User Interfaces in the Open Agent Architecture",
	T	CHEYER, Adam and LULIA, Luc, SRI International "Multimodal Maps: An Agent-based Approach",

Examiner	Date Considered 7/11/02

Examiner: Initial citation considered. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Form 1449 (Modified) **Information Disclosure Statement By Applicant** (Use Several Sheets if Necessary)	Atty Docket No. SRI1P016 Applicant: Cheyer et al. Filing Date: January 5. 1999	Serial No.: 09/225,198 Group 2755

U.S. Patent Documents

Examiner Initial	No.	Patent No.	Date	Patentee	Class	Sub-class	Filing Date
	A						
	B						
	C						
	D						
	E						
	F						
	G						
	H						
	I						
	J						
	K						

Foreign Patent or Published Foreign Patent Application

Examiner Initial	No.	Document No.	Publication Date	Country or Patent Office	Class	Sub-class	Translation Yes	Translation No
	L							
	M							
	N							
	O							
	P							

Other Documents

Examiner Initial	No.	Author, Title, Date, Place (e.g. Journal) of Publication
LAB	R	CUTKOSKY, Mark R., ENGELMORE, Robert S., FIKES, Richard E., GENESERETH, Michael R., GRUBER, Thomas R., Stanford University, MARK, William, Lockheed Palo Alto Research Labs, TENENBAUM, Jay M., WEBER, Jay C., Enterprise Integration Technologies, "An Experiment in Integrating Concurrent Engineering Systems",
LAB	S	MARTIN, David L., CHEYER, Adam, SRI International, LEE, Gowang-Lo, ETRI, "Development Tools for the Open Agent Architecture", The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM96), London, April 1996
LAB	T	CHEYER, Adam, MARTIN, David and MORAN, Douglas, "The Open Agent architecture™", SRI International, AI Center

Examiner Lewis A. Bullock Jr	Date Considered 7/11/02

Examiner: Initial citation considered. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Form 1449 (Modified) **Information Disclosure Statement By Applicant** (Use Several Sheets if Necessary)	Atty Docket No. SRI1P016 Applicant: Cheyer et al. Filing Date: January 5. 1999	Serial No.: 09/225,198 Group 2755

U.S. Patent Documents

Examiner Initial	No.	Patent No.	Date	Patentee	Class	Sub-class	Filing Date
	A						
	B						
	C						
	D						
	E						
	F						
	G						
	H						
	I						
	J						
	K						

Foreign Patent or Published Foreign Patent Application

Examiner Initial	No.	Document No.	Publication Date	Country or Patent Office	Class	Sub-class	Translation Yes	Translation No
	L							
	M							
	N							
	O							
	P							

Other Documents

Examiner Initial	No.	Author, Title, Date, Place (e.g. Journal) of Publication
fap	R	Dejima, Inc., http://www.dejima.com/
fap	S	COHEN, Philip R, CHEYER, Adam, WANG, Michelle, Stanford University, BAEG, Soon Cheol ETRI; "An Open Agent Architecture," AAAI Spring Symposium, pp1-8, March 1994
fap	T	MARTIN, David; OOHAMA, Hiroki; MORAN, Douglas; CHEYER, Adam; "Information Brokering in an Agent Architecture," Proceeding of the 2nd International Conference on Practical Application of Intelligent Agents & Multi-Agent Technology, London, April 1997

Examiner Lewis A Bullock Jr	Date Considered 7/11/02

Examiner: Initial citation considered. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

~~2755~~ 2151

Attorney Docket No. 59501-8016.US01

I hereby certify that this correspondence is being deposited with the U.S. Postal Service with sufficient postage as First Class Mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C., 20231, on:

Date: August 6, 2002 By: Jamie Hughes
Jamie L. Hughes

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF:
Cheyer
APPLICATION NO.: 09/225,198
FILED: 01/05/1999
FOR: **SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS**

EXAMINER: UNKNOWN
ART UNIT: 2755

Information Disclosure Statement After First Office Action but Before Final Action or Notice of Allowance – 37 CFR 1.97(c)

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

1. Timing of Submission

The information transmitted herewith is being filed *after* three months of the filing date of this application or after the mailing date of the first Office action on the merits, whichever occurred last, but *before* the mailing date of either a final action under 37 CFR 1.113 or a Notice of Allowance under 37 CFR 1.311, whichever occurs first. The references listed on the enclosed Form PTO/SB/08A may be material to the examination of this application; the Examiner is requested to make them of record in the application.

2. Cited Information

☒ Copies of the following references are enclosed:

☒ All cited references

3. Effect of Information Disclosure Statement (37 CFR 1.97(h))

This Information Disclosure Statement is not to be construed as a representation that: (i) a search has been made; (ii) additional information material to the examination of this application does not exist; (iii) the information, protocols, results and the like reported by third parties are accurate or enabling; or (iv) the cited information is, or is considered to be, material to patentability. In addition, applicant does not admit that any enclosed item of information constitutes prior art to the subject invention and specifically reserves the right to demonstrate that any such reference is not prior art.

4. Fee Payment (37 CFR 1.97(c)) or Certification (37 CFR 1.97(e))

☒ Applicant elects to pay the fee under 37 CFR 1.17(p) $180.00.

□ Check enclosed for $.

☒ Please charge the above fee(s) to Deposit Account No. 50-2207 this paper is provided in triplicate.

Respectfully submitted,
Perkins Coie LLP

Date: 6 Aug 2002

Brian R. Coleman
Registration No. 39,145

Correspondence Address:
Customer No. 22918
Perkins Coie LLP
P.O. Box 2168
Menlo Park, California 94026
(650) 838-4300

2. Cited Information

☒ Copies of the following references are enclosed:

☒ All cited references

3. Effect of Information Disclosure Statement (37 CFR 1.97(h))

This Information Disclosure Statement is not to be construed as a representation that: (i) a search has been made; (ii) additional information material to the examination of this application does not exist; (iii) the information, protocols, results and the like reported by third parties are accurate or enabling; or (iv) the cited information is, or is considered to be, material to patentability. In addition, applicant does not admit that any enclosed item of information constitutes prior art to the subject invention and specifically reserves the right to demonstrate that any such reference is not prior art.

4. Fee Payment (37 CFR 1.97(c)) or Certification (37 CFR 1.97(e))

☒ Applicant elects to pay the fee under 37 CFR 1.17(p) $180.00.

☐ Check enclosed for $.

☒ Please charge the above fee(s) to Deposit Account No. 50-2207 this paper is provided in triplicate.

Respectfully submitted,
Perkins Coie LLP

Date: 6 Aug 2002

Brian R. Coleman
Registration No. 39,145

Correspondence Address:
Customer No. 22918
Perkins Coie LLP
P.O. Box 2168
Menlo Park, California 94026
(650) 838-4300

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification: H04N 7/16	A1	(11) International Publication Number: (43) International Publication Date:	WO 00/11869 02 March 2000 (02.03.2000)

(21) International Application Number: PCT/US99/19051

(22) International Filing Date: 20 August 1999 (20.08.1999)

(30) Priority Data:

60/097,538	21 August 1998 (21.08.1998)	US
not furnished	30 July 1999 (30.07.1999)	US

(60) Parent Application or Grant
UNITED VIDEO PROPERTIES, INC. [/]; (). ELLIS, Michael, D. [/]; (). LEMMONS, Thomas, R. [/]; (). THOMAS, William, L. [/]; (). TREYZ, G., Victor ; ().

Published

(54) Title: CLIENT-SERVER ELECTRONIC PROGRAM GUIDE
(54) Titre: GUIDE DE PROGRAMMES ELECTRONIQUE CLIENT-SERVEUR

(57) Abstract

A client-server interactive television program guide system is provided. An interactive television program guide client is implemented on user television equipment. The interactive television program guide provides users with an opportunity to define expressions that are processed by the program guide server. The program guide server may provide program guide data, schedules reminders, schedules program recordings, and parentally locks programs based on the expressions. Users' viewing histories may be tracked. The program guide server may analyze the viewing histories and generates viewing recommendations, targets advertising, and collects program ratings information based on the viewing histories.

(57) Abrégé

L'invention concerne un système de guide de programmes de télévision interactif entre un client et un serveur. Un client de guide de programmes de télévision interactif est mis en application sur l'installation télévisuelle d'un utilisateur. Ce guide de programmes permet aux utilisateurs de définir des expressions traitées par le serveur de guide de programmes. Ce serveur peut produire des données de guide de programmes, des rappels de programmation, des enregistrements de programmes et, de même, verrouille des programmes en fonction des expressions. Il est possible de rechercher l'historique de visualisation des utilisateurs. Le serveur de guide de programmes peut analyser les historiques de visualisation et générer des recommandations de visualisation, des publicités ciblées et recueillir des informations d'évaluation de programmes en fonction de ces historiques de visualisation.

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 :
H04N 7/16

A1

(11) International Publication Number: **WO 00/11869**

(43) International Publication Date: 2 March 2000 (02.03.00)

(21) International Application Number: PCT/US99/19051

(22) International Filing Date: 20 August 1999 (20.08.99)

(30) Priority Data:

60/097,538	21 August 1998 (21.08.98)	US
not furnished	13 August 1999 (13.08.99)	US

(71) Applicant: UNITED VIDEO PROPERTIES, INC. [US/US]; 7140 South Lewis Avenue, Tulsa, OK 74136 (US).

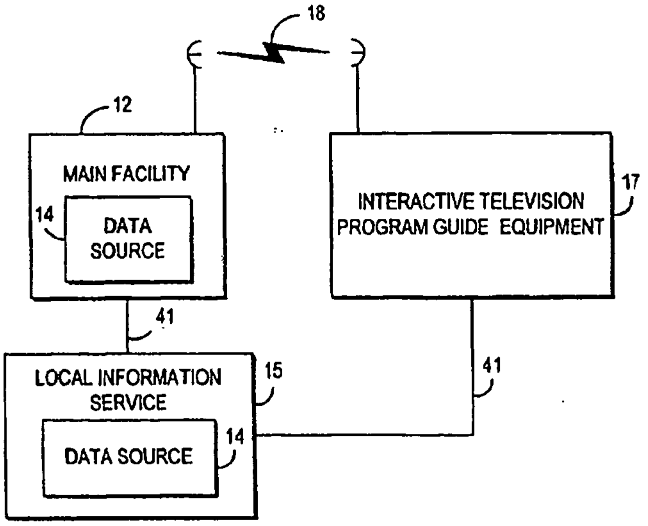
(72) Inventors: ELLIS, Michael, D.; 1300 Kingwood Place, Boulder, CO 80304 (US). LEMMONS, Thomas, R.; Route 2, Box 1178, Sand Springs, OK 74063 (US). THOMAS, William, L.; 11611 South 70th East Avenue, Bixby, OK 74008 (US).

(74) Agents: TREYZ, G., Victor et al.; Fish & Neave, 1251 Avenue of the Americas, New York, NY 10020 (US).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: CLIENT-SERVER ELECTRONIC PROGRAM GUIDE


18
12
MAIN FACILITY
14
DATA SOURCE
INTERACTIVE TELEVISION PROGRAM GUIDE EQUIPMENT
17
41
LOCAL INFORMATION SERVICE
15
DATA SOURCE
14
41

(57) Abstract

A client-server interactive television program guide system is provided. An interactive television program guide client is implemented on user television equipment. The interactive television program guide provides users with an opportunity to define expressions that are processed by the program guide server. The program guide server may provide program guide data, schedules reminders, schedules program recordings, and parentally locks programs based on the expressions. Users' viewing histories may be tracked. The program guide server may analyze the viewing histories and generates viewing recommendations, targets advertising, and collects program ratings information based on the viewing histories.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

Description

CLIENT-SERVER ELECTRONIC PROGRAM GUIDE

Background of the Invention

This invention relates to interactive television program guide systems, and more particularly, to interactive television program guide systems based on client-server arrangements.

Cable, satellite, and broadcast television systems provide viewers with a large number of television channels. Users have traditionally consulted printed television program schedules to determine the programs being broadcast at a particular time. More recently, interactive television program guides have been developed that allow television program information to be displayed on a user's television. Interactive television program guides, which are typically implemented on set-top boxes, allow users to navigate through television program listings using a remote control. In a typical program guide, various groups of television program listings are displayed in predefined or user-selected categories. Program listings are typically displayed in a grid or table. On-line program guides have been proposed that require users to navigate the Internet to access program listings.

Client-server based program guides have been proposed in which program listings are stored on a server at a cable system headend. The server provides the program listings to program guide clients implemented on the set-top boxes of a number of users associated with each headend. As users navigate within a program listings grid, the server provides program listings to the client for display. Such systems, may be limited in their functionality due to their limited use of the resources of the server.

It is therefore an object of the present invention to provide an interactive televison program guide system in which server resources are used to provide enhanced program guide features not provided by conventional set-top-box-based or client-server-based program guides.

Summary of the Invention

This and other objects of the present invention are accomplished in accordance with the principles of the present invention by providing a client-server based interactive television program guide system in which a main facility (e.g., a satellite uplink facility or a facility that feeds such an uplink facility) provides data from one or more data sources to a number of television distribution facilities such as cable system headends, broadcast distribution facilities, satellite television distribution facilities, or other suitable distribution facilities. Some of the data sources may be located at different facilities and have their data provided to the main facility for localization and distribution or may provide their data to the television distribution facilities directly. The data provided to the television distribution facilities includes television programming data (e.g., titles, channels, content information, rating information, program identifiers, series identifiers, or any other information associated with television programming), and other program guide data for additional services other than television program listings (e.g., weather information, associated Internet web links, computer software, etc.). The main facility (and other sources) may provide the program guide data to the television distribution facilities via a satellite link, a telephone network link, a cable or fiber optic link, a microwave link, an Internet link, a combination of such links, or any other suitable communications link.

Each television distribution facility has a program guide server. If desired, program guide servers may also be located at cable system network nodes or other facilities separate from the television distribution facilities or other distribution facilities. Each program guide server stores the program guide data provided by the main facility and provides access to the program guide data to program guide clients implemented on the user television equipment of a number of users associated with each television distribution facility. The program guide servers may also store user data, such as user preference profiles, parental control settings, record and reminder settings, viewing history, and other suitable data.

Providing program guide data with a program guide server and storing user data on the server may provide users with opportunities to perform various functions that may enhance the users' television viewing experience. Users may, for example, set user preference profiles or other favorites that are stored by the program guide server and used by the server to customize the program guide viewing experience for the user. The program guide server may filter program guide data based on the user preference profiles. Only data that is of interest to the user may then be provided to the guide client, thereby tending to minimize the memory requirements of the user's television equipment and lessen the bandwidth requirements of the local distribution network.

A client-server based architecture may also provide users with the ability to search and sort through program related information in ways that might not otherwise be possible due to the limited processing and storage capabilities of the users' television equipment. If desired, users may be provided with access to program guide data without requiring them to navigate the Internet. Users may, for example, define sophisticated boolean or natural language expressions having one or more criteria for searching through and sorting program guide data, scheduling reminders, automatically recording programs and parentally controlling programs. The criteria may also be derived by the program guide server or program guide client from user profiles or by monitoring usage of the program guide. The criteria may be stored on the program guide server. Users may be provided with an opportunity to access, modify, or delete the expressions.

The program guide server may also track the users' viewing histories to provide a user-customized program guide experience. Programs or series of episodes users have watched may be identified and used by the program guide, for example, to inform users when there are showings in the series that the users have not watched. The program guide may, for example, provide viewing recommendations based on a user's viewing history and, if appropriate, on user preference profiles or other criteria stored by the program guide server. The program guide may also target advertisements toward users based on the viewing histories or criteria, and may track the viewing of programs to generate viewership ratings.

Further features of the invention, its nature and various advantages will be more apparent from the accompanying drawings and the following detailed description of the preferred embodiments.

Brief Description of the Drawings

FIG. 1 is a schematic block diagram of an illustrative system in accordance with the present invention.

FIGS. 2a, 2b, and 2c show illustrative arrangements for the interactive program guide equipment of FIG. 1 in accordance with the principles of the present invention.

FIG. 3 is an illustrative schematic block diagram of a user television equipment of FIGS. 2a and 2b in accordance with the principles of the present invention.

FIG. 4 is a generalized schematic block diagram of portions of the illustrative user television equipment of FIG. 3 in accordance with the principles of the present invention.

FIG. 5 is an illustrative main menu screen in accordance with the principles of the present invention.

FIG. 6 is an illustrative program listings by time screen in accordance with the principles of the present invention.

FIG. 7 is an illustrative program listings by channel screen in accordance with the principles of the present invention.

FIGS. 8a-8c are illustrative program listings by category screens in accordance with the principles of the present invention.

FIG. 9a is an illustrative boolean type criteria screen in accordance with the principles of the present invention.

FIG. 9b is an illustrative natural language criteria screen in accordance with the principles of the present invention.

FIG. 10 shows an illustrative agents screen in accordance with the principles of the present invention.

FIG. 11 is an illustrative program listings screen in which program listings found according to the illustrative expressions of FIGS. 9a and 9b are displayed in accordance with the principles of the present invention.

FIG. 12 shows an illustrative setup screen in accordance with the principles of the present invention.

FIGS. 13a-13f show illustrative preference profile screens in accordance with the principles of the present invention.

FIG. 14 shows an illustrative profile activation screen in accordance with the principles of the present invention.

FIG. 15 shows a table containing an illustrative list of programs that might be available to a user after defining the preference profiles of FIGS. 13a-13f in accordance with the principles of the present invention.

FIGS. 16a-16c are illustrative program listings screens that may be displayed according to the preference profiles of FIGS. 13a-13f in accordance with the principles of the present invention.

FIGS. 17a and 17b show illustrative criteria screens in accordance with the principles of the present invention.

FIGS. 18 and 19 show illustrative program reminder lists generated according to the expressions of FIGS. 17a and 17b in accordance with the principles of the present invention.

FIGS. 20a and 20b show an illustrative viewer recommendation overlay, in accordance with the principles of the present invention.

FIG. 20c shows an illustrative additional information screen in accordance with the principles of the present invention.

FIG. 21 is a flowchart of illustrative steps involved in providing users with an opportunity to define preference profiles and access program guide data according to the preference profiles in accordance with the principles of the present invention.

FIG. 22 is a flowchart of illustrative steps involved in providing users with an opportunity to search program guide data, other information, and videos in accordance with the principles of the present invention.

FIG. 23 is a flowchart of illustrative steps involved in processing and using expressions in accordance with the principles of the present invention.

FIG. 24 is a flowchart of illustrative steps involved in tracking and using viewing histories in accordance with the principles of the present invention.

Detailed Description of the Preferred Embodiments

An illustrative system 10 in accordance with the present invention is shown in FIG. 1. Main facility 12 may provide program guide data from data source 14 to interactive television program guide equipment 17 via communications link 18. There may be multiple program guide data sources in main facility 12 but only one has been shown to avoid over-complicating the drawing. If desired, program guide data sources may be located at facilities separate from main facility 12 such as at local information services 15, and may have their data provided to main facility 12 for localization and distribution. Data sources 14 may be any suitable computer or computer-based system for obtaining data (e.g., manually from an operator, electronically via a computer network or other connection, or via storage media) and placing the data into electronic form for distribution by main facility 12. Link 18 may be a satellite link, a telephone network link, a cable or fiber optic link, a microwave link, an Internet link, a combination of such links, or any other suitable communications link. Video signals may also be transmitted over link 18 if desired.

Local information service 15 may be any suitable facility for obtaining data particular to a localized region and providing the data to main facility 12 or interactive television program guide equipment 17 over communications links 41. Local information service 15 may be, for example, a local weather station that measures weather data, a local newspaper that obtains local high school and college sporting information, or any other suitable provider of information. Local information service 15 may be a local business with a computer for providing main facility 12 with, for example, local ski reports, fishing conditions, menus, etc., or any other suitable provider of information. Link 41 may be a satellite link, a telephone network link, a cable or fiber optic link, a microwave link, an Internet link, a combination of such links, or any other suitable communications link. Additional data sources 14 may be located at other facilities for providing main facility 12 with non-localized data (e.g., non-localized program guide data) over link 41.

The program guide data transmitted by main facility 12 to interactive television program guide equipment 17 may include television programming data (e.g., program identifiers, times, channels, titles, descriptions, series identifiers, etc.) and other data for services other than television program listings (e.g., help text, pay-per-view information, weather information, sports information, music channel information, associated Internet web links, associated software, etc.). There are preferably numerous pieces or installations of interactive television program guide equipment 17, although only one is shown in FIG. 1 to avoid over-complicating the drawing.

Program guide data may be transmitted by main facility 12 to interactive television program guide equipment 17 using any suitable approach. Data files may, for example, be encapsulated as objects and transmitted using a suitable Internet based addressing scheme and protocol stack (e.g., a stack which uses the user datagram protocol (UDP) and Internet protocol (IP)). Systems in which program guide data is transmitted from a main facility to television distribution facilities are described, for example, in Gollahon et al. U.S. patent application Serial No. 09/332,624, filed June 11, 1999 (Attorney Docket No. UV-106), which is hereby incorporated by reference herein in its entirety.

A client-server based interactive television program guide is implemented on interactive television program guide equipment 17. Three illustrative arrangements for interactive television program guide equipment 17 are shown in FIGS. 2a-2c. FIG. 2a shows an illustrative arrangement for interactive television program guide equipment 17 in which a program guide server obtains program guide data directly from main facility 12. FIG. 2b shows an illustrative arrangement for interactive television program guide equipment 17 in which a program guide server obtains program guide data from main facility 12 or some other facility (e.g., local information service 15) via the Internet. In either of these approaches, users may be provided with opportunities to access program guide data without having to navigate the Internet, if desired. As shown in FIGS. 2a and 2b, interactive program guide television equipment 17 may include television distribution facility 16 and user television equipment 22.

Television distribution facility 16 may have program guide distribution equipment 21 and program guide server 25. Distribution equipment 21 is equipment suitable for providing program guide data from program guide server 25 to user television equipment 22 over communications path 20. Distribution equipment 21 may include, for example, suitable transmission hardware for distributing program guide data on a television channel sideband, in the vertical blanking interval of a television channel, using an in-band digital signal, using an out-of-band digital signal, over a dedicated computer network or Internet link, or by any other data transmission technique suitable for the type of communications path 20. Analog or digital video signals (e.g., television programs) may also be distributed by distribution equipment 21 to user television equipment 22 over communications paths 20 on multiple analog or digital television channels. Alternatively, videos may be distributed to user television equipment 22 from some other suitable distribution facility, such as a cable system headend, a broadcast distribution facility, a satellite television distribution facility, or any other suitable type of television distribution facility.

Communications paths 20 may be any communications paths suitable for distributing program guide data. Communications paths 20 may include, for example, a satellite link, a telephone network link, a cable or fiber optic link, a microwave link, an Internet link, a data-over-cable service interface specification (DOCSIS) link, a combination of such links, or any other suitable communications link. Communications paths 20 preferably have sufficient bandwidth to allow television distribution facility 16 or another distribution facility to distribute television programming to user television equipment 22. There are typically multiple pieces of user television equipment 22 and multiple associated communications paths 20, although only one piece of user television equipment 22 and communications path 20 are shown in FIGS. 2a and 2b to avoid over-complicating the drawings. If desired, television programming and program guide data may be provided over separate communications paths.

Program guide server 25 may be based on any suitable combination of server software and hardware. Program guide server 25 may retrieve program guide data or video files from storage device 56 in response to program guide data or video requests generated by an interactive television program guide client implemented on user television equipment 22. As shown in FIGS. 2a and 2b, program guide server 25 may include processing circuitry 54 and storage device 56. Processing circuitry 54 may include any suitable processor, such as a microprocessor or group of microprocessors, and other processing circuitry such as caching circuitry, video decoding circuitry, direct memory access (DMA) circuitry, input/output (I/O) circuitry, etc.

Storage device 56 may be a memory or other storage device, such as random access memory (RAM), flash memory, a hard disk drive, etc., that is suitable for storing the program guide data transmitted to television distribution facility 16 by main facility 12. User data, such as user preference profiles, preferences, parental control settings, record and reminder settings, viewing histories, and other suitable data may also be stored on storage device 56 by program guide server 25. Program guide data and user data may be stored on storage device 56 in any suitable format (e.g., a Structured Query Language (SQL) database). If desired, storage 56 may also store video files for playing back on demand.

Processing circuitry 54 may process requests for program guide data by searching the program guide data stored on storage device 56 for the requested data, retrieving the data, and providing the retrieved data to distribution equipment 21 for distribution to user television equipment 22. Processing circuitry 54 may also process storage requests generated by the program guide client that direct program guide server 25 to store user data. Alternatively, program guide server 25 may distribute program guide data to and receive user data from user television equipment 22 directly. If communications paths 20 include an Internet link, DOCSIS link, or other high speed computer network link (e.g., 10BaseT, 100BaseT, 10BaseF, T1, T3, etc.), for example, processing circuitry 54 may include circuitry suitable for transmitting program guide and user data and receiving program guide data and storage requests over such a link.

Program guide server 25 may communicate with user television equipment 22 using any suitable communications protocol. For example, program guide server 25 may use a communications protocol stack that includes transmission control protocol (TCP) and Internet protocol (IP) layers, sequenced packet exchange (SPX) and internetwork packet exchange (IPX) layers, Appletalk transaction protocol (ATP) and datagram delivery protocol (DDP) layers, DOCSIS, or any other suitable protocol or combination of protocols. User television equipment 22 may also include suitable hardware for communicating with program guide server 25 over communications paths 20 (e.g., Ethernet cards, modems (digital, analog, or cable), etc.)

The program guide client on user television equipment 22 may retrieve program guide data from and store user data on program guide server 25 using any suitable client-server based approach. The program guide may, for example, pass SQL requests as messages to program guide server 25. In another suitable approach, the program guide may invoke remote procedures that reside on program guide server 25 using one or more remote procedure calls. Program guide server 25 may execute SQL statements for such invoked remote procedures. In still another suitable approach, client objects executed by the program guide may communicate with server objects executed by program guide server 25 using, for example, an object request broker (ORB). This may involve using, for example, Microsoft's Distributed Component Object Model (DCOM) approach. As used herein, "record requests" and "storage requests" are intended to encompass any of these types of inter-process or inter-object communications, or any other suitable type of inter-process or inter-object communication.

FIG. 2b shows an illustrative arrangement for interactive television program guide equipment 17 in which program guide server 25 obtains program guide data via the Internet. The program guide data obtained by program guide server 25 may be provided by main facility 12 or from some other source (e.g., local information service 15) and made available on the Internet. Internet service system 61 may use any suitable combination of hardware and software capable of providing program guide data from the Internet to program guide server 25 using an Internet based approach (e.g., using the HyperText Transfer Protocol (HTTP), File Transfer Protocol (FTP), etc.). FIG. 2b shows Internet service system 61 as being encompassed by television distribution facility 16. If desired, Internet service system 61 may be located at a facility that is separate from television distribution facility 16. Internet service system 61 may, for example, be located at main facility 12 or at some other Internet node suitable for providing program guide data from the Internet to program guide server 25. The functionality of Internet service system 61 and program guide server 25 may be integrated into one system if desired.

Another suitable arrangement for interactive television program guide equipment 17 is shown in FIG. 2c. Interactive television program guide equipment 17 may include, for example, television distribution facility 16 having program guide server 25 and Internet service system 61. A program guide client application may run on personal computer 23. The client may access program guide server 25 via Internet service system 61 and communications path 20. Personal computer 23 may include processing circuitry 27, memory 29, storage device 31, communications device 35, and monitor 39.

Processing circuitry 27 may include any suitable processor, such as a microprocessor or group of microprocessors, and other processing circuitry such as caching circuitry, direct memory access (DMA) circuitry, input/output (I/O) circuitry, etc. Processing circuitry 27 may also include suitable circuitry for displaying television programming. Personal computer 23 may include, for example, a PC/TV card. Memory 29 may be any suitable memory, such as random access memory (RAM) or read only memory (ROM), that is suitable for storing the computer instructions and data. Storage device 31 may be any suitable storage device, such as a hard disk, floppy disk drive, flash RAM card, recordable CD-ROM drive, or any other suitable storage device. Communications device 35 may be any suitable communications device, such as a conventional analog modem or cable modem.

An illustrative arrangement for user television equipment 22 of FIGS. 2a and 2b is shown in FIG. 3. User television equipment 22 of FIG. 3 receives analog video or a digital video stream and data, program guide data, or any suitable combination thereof, from television distribution facility 16 (FIG. 1) at input 26. During normal television viewing, a user tunes set-top box 28 to a desired television channel. The signal for that television channel is then provided at video output 30. The signal supplied at output 30 is typically either a radio-frequency (RF) signal on a predefined channel (e.g., channel 3 or 4), or a analog demodulated video signal, but may also be a digital signal provided to television 36 on an appropriate digital bus (e.g., a bus using the Institute of Electrical and Electronics Engineers (IEEE) 1394 standard, (not shown)). The video signal at output 30 is received by optional secondary storage device 32.

The interactive television program guide client may run on set-top box 28, on television 36 (if television 36 has suitable processing circuitry and memory), on a suitable analog or digital receiver connected to television 36, or on digital storage device 31 if digital storage device 31 has suitable processing circuitry and memory. The interactive television program guide client may also run cooperatively on a suitable combination of these devices. Interactive television application systems in which a cooperative interactive television program guide application runs on multiple devices are described, for example, in Ellis U.S. patent application Serial No. 09/186,598, filed November 5, 1998, which is hereby incorporated by reference herein in its entirety.

Secondary storage device 32 can be any suitable type of analog or digital program storage device or player (e.g., a videocassette recorder, a digital versatile disc (DVD) player, etc.). Program recording and other features may be controlled by set-top box 28 using control path 34. If secondary storage device 32 is a videocassette recorder, for example, a typical control path 34 involves the use of an infrared transmitter coupled to the infrared receiver in the videocassette recorder that normally accepts commands from a remote control such as remote control 40. Remote control 40 may be used to control set-top box 28, secondary storage device 32, and television 36.

If desired, a user may record programs, program guide data, or a combination thereof in digital form on optional digital storage device 31. Digital storage device 31 may be a writeable optical storage device (such as a DVD player capable of handling recordable DVD discs), a magnetic storage device (such as a disk drive or digital tape), or any other digital storage device. Interactive television program guide systems that have digital storage devices are described, for example, in Hassell et al. U.S. patent application Serial No. 09/157,256, filed September 17, 1998, which is hereby incorporated by reference herein in its entirety.

Digital storage device 31 can be contained in set-top box 28 or it can be an external device connected to set-top box 28 via an output port and appropriate interface. Digital storage device 31 may, for example, be contained in local media server 29. If necessary, processing circuitry in set-top box 28 formats the received video, audio and data signals into a digital file format. Preferably, the file format is an open file format such as the Moving Picture Experts Group (MPEG) MPEG-2 standard or the Moving Joint Photographic Experts Group (MJPEG) standard. The resulting data is streamed to digital storage device 31 via an appropriate bus (e.g., a bus using the Institute Electrical and Electronics Engineers (IEEE) 1394 standard), and is stored on digital storage device 31. In another suitable approach, an MPEG-2 data stream or series of files may be received from distribution equipment 21 and stored.

Television 36 receives video signals from secondary storage device 32 via communications path 38. The video signals on communications path 38 may either be generated by secondary storage device 32 when playing back a prerecorded storage medium (e.g., a videocassette or a recordable digital video disc), by digital storage device 31 when playing back a prerecorded digital medium, may be passed through from set-top box 28, may be provided directly to television 36 from set-top box 28 if secondary storage device 32 is not included in user television equipment 22, or may be received directly by television 36. During normal television viewing, the video signals provided to television 36 correspond to the desired channel to which a user has tuned with set-top box 28. Video signals may also be provided to television 36 by set-top box 28 when set-top box 28 is used to play back information stored on digital storage device 31.

Set-top box 28 may have communications device 37 for communicating with program guide server 25 over communications path 20. Communications device 37 may be a modem (e.g., any suitable analog or digital standard, cellular, or cable modem), network interface card (e.g., an Ethernet card, Token ring card, etc.), a combination of such devices, or any other suitable communications device. Television 36 may also have such a suitable communications device if desired.

Set-top box 28 may have memory 44. Memory 44 may be any memory or other storage device, such as a random access memory (RAM), read only memory (ROM), flash memory, a hard disk drive, a combination of such devices, etc., that is suitable for storing program guide client instructions and program guide data for use by the program guide client.

A more generalized embodiment of user television equipment 22 of FIG. 3 is shown in FIG. 4. As shown in FIG. 4, program guide data from television distribution facility 16 (FIG. 1) and programming are received by control circuitry 42 of user television equipment 22. The functions of control circuitry 42 may be provided using the set-top box arrangement of FIGS. 2a and 2b. Alternatively, these functions may be integrated into an advanced television receiver, personal computer television (PC/TV) such as shown in FIG. 2c, or any other suitable arrangement. If desired, a combination of such arrangements may be used.

User television equipment 22 may also have secondary storage device 47 and digital storage device 49 for recording programming. Secondary storage device 47 can be any suitable type of analog or digital program storage device (e.g., a videocassette recorder, a digital versatile disc (DVD), etc.). Program recording and other features may be controlled by control circuitry 42. Digital storage device 49 may be, for example, a writeable optical storage device (such as a DVD player capable of handling recordable DVD discs), a magnetic storage device (such as a disk drive or digital tape), or any other digital storage device.

User television equipment 22 may also have memory 63. Memory 63 may be any memory or other storage device, such as a random access memory (RAM), read only memory (ROM), flash memory, a hard disk drive, a combination of such devices, etc., that is suitable for storing program guide client instructions and program guide data for use by control circuitry 42.

User television equipment 22 of FIG. 4 may also have communications device 51 for supporting communications between the program guide client and program guide server 25 and via communications path 20. Communications device 51 may be a modem (e.g., any suitable analog or digital standard, cellular, or cable modem), network interface card (e.g., an Ethernet card, Token ring card, etc.), a combination of such devices, or any other suitable communications device.

A user controls the operation of user television equipment 22 with user interface 46. User interface 46 may be a pointing device, wireless remote control, keyboard, touch-pad, voice recognition system, or any other suitable user input device. To watch television, a user instructs control circuitry 42 to display a desired television channel on display device 45. To access the functions of the program guide, a user instructs the program guide implemented on interactive television program guide equipment 17 to generate a main menu or other desired program guide display screen for display on display device 45. If desired, the program guide client running on user television equipment 22 may provide users with access to program guide features without requiring them to navigate the Internet.

The program guide may provide users with an opportunity to access program guide features through a main menu. A main menu screen, such as illustrative main menu screen 100 of FIG. 5, may include menu 102 of selectable program guide features 106. If desired, program guide features 106 may be organized according to feature type. In menu 102, for example, program guide features 106 have been organized into three columns. The column labeled "TV GUIDE" is for listings related features, the column labeled "MSO SHOWCASE" is for multiple system operator (MSO) related features, and the column labeled "VIEWER SERVICES" is for viewer related features. The interactive television program guide may generate a display screen for a particular program guide feature when a user selects that feature from menu 102.

Main menu screen 100 may include one or more selectable advertisements 108. Selectable advertisements 108 may, for example, include text and graphics advertising pay-per-view programs or other programs or products. When a user selects a selectable advertisement 108, the program guide may display information (e.g., pay-per-view information) or take other actions related to the content of the advertisement. Pure text advertisements may be presented, if desired, as illustrated by selectable advertisement banner 110.

Main menu screen 100 may also include other screen elements. The brand of the program guide product may be indicated, for example, using a product brand logo graphic such as product brand logo graphic 112. The identity of the television service provider may be presented, for example, using a service provider logo graphic such as service provider logo graphic 114. The current time may be displayed in clock display region 116. In addition, a suitable indicator such as indicator graphic 118 may be used to indicate to a user that mail from a cable operator is waiting for a user if the program guide supports messaging functions.

The interactive television program guide may provide a user with an opportunity to view television program listings. A user may indicate a desire to view program listings by, for example, positioning highlight region 120 over a desired program guide feature 106. Alternatively, the program guide may present program listings when a user presses a suitable key (e.g., a "guide" key) on remote control 40. When a user indicates a desire to view television program listings, the program guide client requests listings from program guide server 25 and generates an appropriate program listings screen for display on display device 45 (FIG. 4). Program listings screens may be overlaid on a program being viewed by a user or overlaid on a portion of the program in a "browse" mode. Program listings screens are described, for example, in Knudson et al. U.S. patent application Serial No. 09/357,941, filed July 16, 1999 (Attorney Docket No. UV-114), which is hereby incorporated by reference herein in its entirety.

A program listings screen may contain one or more groups or lists of program listings organized according to one or more organization criteria (e.g., by time, by channel, by program category, etc.). The program guide may, for example, provide a user with an opportunity to view listings by time, by channel, according to a number of categories (e.g., movies, sports, children, etc.), or may allow a user to search for a listing by title. Program listings may be displayed using any suitable list, table, grid, or other suitable display arrangement. If desired, program listings screens may include selectable advertisements, product brand logo graphics, service provider brand graphics, clocks, or any other suitable indicator or graphic.

A user may indicate a desire to view program listings by time, channel, or category by, for example, selecting a selectable feature 106 from menu 102. In response, the program guide client may issue one or more requests to program guide server 25 for listings in the selected category if such listings are not already cached in memory 63 (FIG. 4). Program guide server 25 may retrieve program guide data stored on storage device 56, on another server, or from Internet service system 61, and provide the data to the program guide client via program guide distribution equipment 21.

The program guide client may display program listings in a suitable program listings screen on user television equipment 22. FIG. 6 illustrates the display of program listings by time. Program listings screen 130 of FIG. 6 may include highlight region 151, which highlights the current program listing 150. A user may position highlight region 151 by entering appropriate commands with user interface 46. For example, if user interface 46 has a keypad, a user can position highlight region 151 using "up" and "down" arrow keys on remote control 40. A user may select a listing by, for example, pressing on the "OK" or "info" key on remote control 40. Alternatively, a touch sensitive screen, trackball, voice recognition device, or other suitable device may be used to move highlight region 151 or to select program listings without the use of highlight region 151. In still another approach, a user may speak a television program listing into a voice request recognition system. These methods of selecting program listings are merely illustrative. Any other suitable approach for selecting program listings may be used if desired.

A user may view additional listings for the time slot indicated in timebar 111 by, for example, pressing an "up" or "down" arrow, or a "page up" or "page down" key on remote control 40. The user may also see listings for the next 24 hour period, or the last 24 hour period, by pressing a "day forward" or "day backward" key on remote control 40, respectively. If there are no listings starting exactly 24 hours in the indicated direction, the program guide may pick programs starting at either closer or further than 24 hours away. If desired, the program guide may require a user to scroll through advertisement banner 110. A user may view program listings for other time slots by, for example, pressing "right" and "left" arrows on remote control 40.

FIG. 7 illustrates the display of program listings by channel. A user may scroll up and down to view program listings for additional time slots, and may scroll left and right to view program listings for other channels. If desired, the day for which program listings are displayed may be included in display area 147 with the channel number as shown.

The program guide may provide users with an opportunity to view program listings sorted by category. A user may, for example, press a special category key on remote control 40 (e.g., "movies", "sports", "children", etc.), select a selectable category feature from main menu screen 100 (FIG. 5), or may indicate a desire to view program listings by category using any other suitable approach. FIG. 8a is an illustrative program listings screen in which program listings for movies are displayed. FIG. 8b is an illustrative program listings screen in which program listings for sports-related programming are displayed. FIG. 8c is an illustrative program listings screen in which program listings for children's programs are displayed.

In program listings display screens such as those shown in FIGS. 7a and 8a-8c for example, program listings within lists 129 may be divided into predefined time slots, such as into 30 minute time slots. Between each time slot, separator 128 may be displayed to indicate to a user that a user has scrolled or paged program listings from one time slot to the next. In FIG. 7 for example, a user is scrolling from program listings in the 11:30 PM to the 12:00 AM time slot. This is indicated by the display of the name of the next week day. In FIGS. 8a-8c, for example, a user is scrolling from program listings in the 12:30 PM time slot to program listings in the 1:00 PM time slot. If desired, separators 128 may be displayed only for those timeslots for which there are listings. When the user scrolls within listings, highlight region 151 may skip separator 128. FIGS. 6, 7, and 8a-8c also illustrate how the program guide may display an advertisement banner so that a user is required to scroll past the banner to access additional program listings.

The program listings screens of FIGS. 6, 7, 8a, 8b, and 8c have also been shown as including various other screen elements. Program listings display screens may include, for example, selectable advertisements, advertisement banners, brand logos, service provider logos, clocks, message indicators, or any other suitable screen element. The program guide may provide users with access to selectable advertisements in response to, for example, a user pressing left arrows to move highlight region 151 to highlight a selectable advertisement. In the illustrative program listings screens of FIGS. 6, 8a, 8b, and 8c, the program guide may also adjust the time displayed in timebar 123 as the user scrolls or pages through program listings to reflect the time of the program listing at the top of the list.

The program guide client may provide a user with an opportunity to define sophisticated boolean or natural language expressions of one or more criteria. Such criteria may include, for example, attribute type and attribute information that is provided by program guide server 25. The user defined expressions may be stored by program guide server 25 for searching through and sorting program guide data, scheduling reminders, automatically recording programs, and parentally controlling programs. Criteria may also be derived by the program guide server or program guide client from user profiles or by monitoring usage of the program guide or advertising. Program guide server 25 may also use expressions to obtain other types of information or programs. Program guide server 25 may obtain, for example, video-on-demand programs, web site links, games, chat group links, merchandise information, or any other suitable information or programming from data sources 14 located at main facility 12 or other facilities. The program guide client may provide users with an opportunity to access, modify, or delete the expressions if desired.

A user may indicate a desire to search program guide data by, for example, selecting selectable Search feature 106 of main menu 102 (FIG. 5). In response, the program guide client may display a criteria screen, such as illustrative criteria screen 141 and 149 of FIGS. 9a and 9b. The program guide client may display criteria screen 141 of FIG. 9a to provide a user with an opportunity to define a boolean expression. The user may construct a boolean expression by selecting criteria such as attribute types, attributes, logical operators, and sorting criteria. User selectable criteria may also include what program guide server 25 searches for such as, for example, program listings, program information, web sites, video-on-demand videos, software, or any other suitable program guide data, other information, or videos.

Users may define expressions by, for example, arrowing up or down between criteria, arrowing left or right to choose an attribute, attribute type or logical operator, and pressing a suitable key to indicate that the user is finished (e.g., an "OK" key). In the example of FIG. 9a, the user has constructed a boolean expression for all action programs that have the actor Bruce Willis, that start between 7:00P and 11:00P, and that end between 9:00P and 1:30A on the current day. FIG. 9a has not been shown as including criteria for selecting what program guide server 25 searches for to avoid over-complicating the drawing.

The program guide client may display criteria screen 149 of FIG. 9b to provide a user with an opportunity to construct a natural language expression. The user may enter a natural language phrase, such as "List in alphabetical order all action programs starring Bruce Willis and that start today between 7:00P and 11:00P and that end between 9:00P and 1:30A" using user interface 46 (FIG. 4).

The program guide client may submit the user defined boolean expression or the natural language expression to program guide server 25 for processing. Program guide server 25 may process the expression, and provide the resulting program guide data (e.g., program listings, program information, software, Internet links, etc.) or video programs to the program guide client for display. FIG. 11 shows an illustrative program listings screen that may be displayed by the program guide client in response to the expressions defined in FIGS. 9a and 9b.

Users may also indicate a desire to have program guide server 25 automatically process expressions by, for example, saving defined expressions as agents. A user may indicate a desire to save an expression as an agent by, for example, selecting Save As Agent selectable feature 147 of FIGS. 9a and 9b after defining a boolean or natural language expression. The program guide client may automatically highlight Save As Agent selectable feature 147 when a user indicates that the user is finished defining an expression (e.g., by pressing an "OK" key). If desired the program guide client may provide the user with an opportunity to name the agent.

Users may access saved expressions or agents by, for example, selecting selectable Agent feature 106 of main menu 102. In response, the program guide client may display a list of saved expressions or agents. An illustrative agents screen 1101 is shown in FIG. 10. A user may indicate a desire to view program listings by, for example, positioning highlight region 151 over the desired expression and pressing an "OK" key on remote control 40. In response to a user indicating a desire to access an expression, the program guide client may submit the user defined expression to program guide server 25 for processing. Program guide server may process the expression, and provide program listings to the program guide client for display in a program listings screen. For example, if a user saved the boolean expression of FIG. 9a, named it "Bruce Willis", and then indicated a desire to access listings for the expression the program guide client may display the listings screen of FIG. 10.

In still another approach, the program guide client may provide the expression to program guide server 25 in response to the user saving the expression as an agent. Program guide server 25 may store the expression and monitor the data stored on storage device 56 for program guide listings, program information, other information, software, videos, etc., that match the expression. Program guide server 25 may also query other sources for program guide data and videos that match the expression via, for example, the Internet. Program guide server 25 may obtain the program guide data, other information or videos from storage device 56 or other sources and provide them to the program guide client when the user indicates a desire to access the agent. Alternatively, program guide server 25 may provide the program guide data, other information, or videos to the program guide client automatically when the user accesses a feature of the program guide that would display such information. In still another suitable approach, program guide server 25 may provide, for example, program identifiers and air times to the program guide client for use in generating program reminders that indicate found programs.

The program guide may also provide users with an opportunity to define user preferences that allow users to customize their program guide experience. Systems in which interactive television program guides provide users with opportunities to define user preference profiles are described, for example, in Ellis et al. U.S. patent application Serial No. 09/034,934, filed March 4, 1998 (Attorney Docket No. UV-43), which is hereby incorporated by reference herein in its entirety. Users may indicate a desire to set up user preference profiles, for example, by selecting a selectable Setup feature 106 from main menu 102 of FIG. 5. When a user selects a selectable Setup feature 106 from main menu 102, the program guide client may display a setup screen, such as illustrative setup screen 411 of FIG. 12.

Setup screen 411 may provide a user with an opportunity to set up various guide features, set parental control features, set features of set-top box 28 (FIG. 3), set audio features, set the screen position, set user preference profiles, or to set up any other feature or suitable combination of features. The user may indicate a desire to set up a user preference profile by, for example, selecting User Profile feature 417. When the user indicates a desire to set up a user preference profile, the program guide client may display a user preference profile setup screen, such as the preference profile setup screens shown in FIGS. 13a-13f. This method of defining user profiles is only illustrative, as any suitable method may be used.

In practice, there may be multiple users associated with each user television equipment 22. The program guide may provide users with the ability to set up multiple user preference profiles. Users may switch between user preference profiles by, for example, selecting preference profile selector 109 and arrowing right or left to select the desired user preference profile. In FIGS. 13a-13f, for example, the user has selected Preference profile #1, which may correspond to a particular user.

User preference profiles may include criteria such as preference attributes 104 and preference levels 106. Preference attributes 104 may be organized by type. Attribute types and attributes may be programmed into the program guide client, or may be retrieved by the program guide client from program guide server 25. In the former approach, the available attribute types and attributes may remain static until the program guide client is updated. In the latter approach, the available attribute types and attributes may be dynamic. Suitable attribute types and attributes may be provided at any time by main facility 12 or television distribution facility 16. Each time a user indicates a desire to set up a user preference profile, the program guide client may query program guide server 25 for the available attribute types and attributes. When a user indicates a desire to set up a user preference profile in either approach, the program guide client may query program guide server 25 for the user preference profiles associated with that program guide client.

FIGS. 13a-13f show six illustrative views of preference profile setup screens in which the user has selected attribute types by, for example, selecting attribute selector 111 and arrowing right or left until a desired preference attribute type is displayed. For example, FIGS. 13a-13f illustrate how the program guide may provide a user with an opportunity to set preference levels for series, genres, channels, actors and actresses, ratings, and other types of preference attributes, respectively. The user may select preference attributes by, for example, arrowing down after selecting an attribute type. The user may then arrow right or left until a desired attribute is displayed. After the desired preference attribute is displayed, the user may, for example, arrow down to set a preference level for the attribute. The user may then, for example, arrow right or left to select a suitable preference level.

Preference levels that may be used to indicate the user's interest or disinterest in a given preference attribute include strong like, weak like, strong dislike, weak dislike, mandatory (appropriate, e.g., for closed-captioning for a deaf person), illegal (appropriate, e.g., for R-rated programs for a child) and don't care (neutral). After the user indicates that he or she is finished defining a profile (e.g., by pressing an "OK" key or remote control 40), the program guide client may provide the preference profile data to program guide server 25 for use in providing program guide data. The user may arrow down again to select additional criteria, or arrow up to edit criteria that has already been selected. The user may delete an attribute by, for example, setting its preference level to "don't care."

The user may activate or deactivate one or more defined preference profiles by, for example, selecting selectable Profile feature 106 from main menu 102 of FIG. 5. The program guide client may respond by, for example, querying program guide server 25 for any defined preference profiles, providing the user with a list of preference profiles, and providing the user with an opportunity to activate or deactivate one or more preference profiles as shown in FIG. 14. A user may deactivate a preference profile by, for example, setting the profile to non-active. A user may set a preference profile as active to varying degrees. For example, a user may set a profile as active by setting the profile to "wide", "moderate", or "narrow" scope.

The program guide client may also indicate to program guide server 25 which profiles are activated or deactivated. The program guide server may use, for example, the attributes of one or more user preference profiles as additional criteria when retrieving data in response to data requests from the program guide client. If multiple preference profiles are used simultaneously, program guide server 25 may reconcile any conflicts using any suitable approach. Interactive television program guide systems that resolve conflicts among multiple active user preference profiles are described, for example, in above-mentioned Ellis et al. U.S. patent application Serial No. 09/034,934, filed March 4, 1998.

FIG. 15 is a table containing an illustrative list of programs that might be available to a user. The results that appear under the columns labeled "narrow scope", "moderate scope", and "wide scope", show which programs satisfy the preference attributes and preference levels of, for example, Profile #1 as illustratively defined in FIGS. 13a-13f. In practice, a listings screen generated based on a profile that is set to widest scope may typically include a larger number of program listings depending on the mandatory attributes set by the user.

When the user activates Profile #1 and sets it to the widest scope, program guide server 25 may provide program guide data for programs that have all mandatory attributes and no illegal attributes. For example, Seinfeld, The Shining, ER, Terminator, and My Stepmother is an Alien are included in the widest preference scope because they have the only mandatory attribute that is specified in Profile #1 -- closed-captioning (as set in FIG. 13f). In addition, they have no preference attributes with a preference level of illegal (R rating, TV-MA rating, or NC-17 rating (as set in FIG. 13e). The Night at the Opera is not included because it does not have a mandatory attribute (closed-captioning). Dante's Peak is not included because it has a illegal rating (R). An illustrative program listings screen that may be displayed by the program guide client with such limited data is shown in FIG. 16a (ER has not been listed because, presumably, it would be in a different time block).

When the user activates Profile #1 and sets it to the moderate scope, program guide server 25 may provide program guide data for programs that have no preference attributes with an associated preference level of disliked, that have all mandatory attributes, and that have no illegal attributes. The Shining is not included because horrors have a preference level of "weak dislike" (as set in FIG. 13b). Dante's Peak is not included because it has an R-rating, which has an attribute level of illegal (as set in FIG. 13e). Night at the Opera is not included because it is not closed-captioned, which is a mandatory attribute (as set in FIG. 13f). The Terminator, for example is not within the moderate scope of Profile #1 because the preference attribute of horror in Profile #1 has an associated preference level of "weak dislike" and the preference attribute of Schwarzenegger (an actor in the program Terminator) has an associated preference level of "strong dislike" (as set in FIGS. 13b and 13d, respectively). Seinfeld and ER are included because they do not have any disliked attributes.

When faced with two different preference levels associated with the same program, the program guide uses the stronger of the two. My Stepmother is an Alien is included, for example, because it has a "strong like" preference attribute that outweighs the "weak dislike". An illustrative program listings screen that may be displayed by the program guide client with such limited program guide data is shown in FIG. 16b. In practice, a listings screen generated based on a profile that is set to moderate scope may typically include a larger number of program listings depending on the mandatory attributes set by the user.

When the user activates Profile #1 and sets it to the narrow preference scope, program guide server 25 may provide program guide data for all liked programs that are not more disliked and that have all mandatory attributes and no illegal attributes. The Shining is not included because it has a weakly disliked attribute, horror. Terminator is not included because it has a strongly disliked attribute, Arnold Schwarzenegger. My Stepmother is an Alien is included because the strongly liked attribute of comedy has priority over the weakly disliked attribute of horror. Dante's Peak is not included because it has a rating of R. Night at the Opera is not included because it is not closed-captioned. ER is not within the narrow scope because it does not have any liked attributes. It is at best, neutral. An illustrative program listings screen that may be displayed by the program guide client with such limited program guide data is shown in FIG. 16c.

The program guide may also provide users with an opportunity to schedule reminders using boolean or natural language expressions having one or more criteria. If desired, program guide server 25 may schedule reminders based on user preference profiles and agents. Reminders may be scheduled for individual programs or series of programs. Systems in which reminders are set for series of programs are described, for example, in Knudson et al. U.S. patent application Serial No. 09/330,792, filed June 11, 1999 (Attorney Docket No. UV-56), which is hereby incorporated by reference herein in its entirety.

A user may indicate a desire to schedule a reminder by, for example, selecting a selectable Reminders feature 106 from main menu 100 of FIG. 5. In response, the program guide may display a criteria screen. Illustrative criteria screens 161 and 169 are shown in FIGS. 17a and 17b. The program guide client may display criteria screen 161 of FIG. 17a to provide a user with an opportunity to set reminders according to a boolean type expression. The user may construct a boolean expression by selecting criteria such as attribute types, attributes, and logical operators. The user may make such selections, for example, using any suitable combination of right, left, up, or down arrow key sequences to sequence through the attribute types, attributes and logical operators. In the example of FIG. 17a, the user has defined a boolean expression to schedule reminders for comedies that star Gary Shandling and that have a rating less than R. In the example of FIG. 17b, the user has defined a similar natural language expression.

The program guide client may submit the user defined boolean or natural language expression to program guide server 25 for processing. Program guide server 25 may process the expression and schedule reminders for all of the programs that meet the expression. Program reminders may be scheduled using any suitable approach. In one suitable approach, program guide server 25 may store program identifiers and air times and send messages to the program guide client at an appropriate time before a program starts. In another suitable approach, program guide server 25 may process an expression and provide program identifiers and air times to the program guide client. The program guide client may, for example, maintain a list of program identifiers and display program reminders at an appropriate time before the programs start.

The program guide may remind a user that a program is airing at the time a program airs. In an alternative approach, the program guide may remind a user at some predetermined period of time before the program airs that a program is going to air. FIGS. 18 and 19 show illustrative program reminder lists 171. In FIG. 18, reminder list 171 is overlaid on top of the currently display television program to provide a user with the opportunity to view a reminder while still viewing a portion of the television program that a user is watching. In FIG. 19, reminder list 171 is shown overlaid on top of a program listings display screen. The program guide may provide a user with an opportunity to scroll through reminder list 171 by, for example, using remote control arrow keys. The program guide may hide the reminder list when, for example, a user selects hide reminder feature 172. The guide may also display reminder list 171 if, for example, the user presses an "OK" key at any time while watching TV.

The program guide may also provide users with an opportunity to schedule programs for recording by secondary storage device 47 or digital storage device 49 (FIG. 4) using boolean or natural language expressions. If desired, program guide server 25 may schedule programs for recording based on user preference profiles or agents. Programs may also be scheduled for recording by program guide server 25. Program guide systems in which programs are recorded by a remote server are described, for example, in Ellis et al. U.S. patent application Serial No. 09/332,244, filed June 11, 1999 (Attorney Docket No. UV-84), which is hereby incorporated by reference herein in its entirety.

A user may indicate a desire to schedule a program for recording by, for example, selecting a selectable Record feature 106 from main menu 102 of FIG. 5. In response, the program guide may display a criteria screen, such as illustrative criteria screens 161 and 169 of FIGS. 17a and 17b. The program guide client may display criteria screen 161 of FIG. 17a to provide a user with an opportunity to schedule a program for recording according to a boolean type expression. The user may construct a boolean expression by selecting criteria such as attribute types, attributes, and logical operators. The user may make such selections, for example, using any suitable combination of right, left, up, or down arrow key sequences to sequence through the attribute types, attributes and logical operators. In the example of FIG. 17a, the user has defined a boolean expression to schedule for recording comedies that star Gary Shandling and that have a rating less than R. In the example of FIG. 17b, the user has defined a similar natural language expression with similar criteria.

The program guide client may submit the user defined boolean or natural language expression to program guide server 25 for processing. Program guide server 25 may process the expression and schedule all of the programs that meet the expression for recording. Recording by program guide server 25 may be performed, for example, as described in above-mentioned Ellis et al. U.S. patent application Serial No. 09/332,244, filed June 11, 1999 (Attorney Docket No. UV-84). In another suitable approach, program guide server 25 may process the expression and provide program identifiers and air times to the program guide client. The program guide client may, for example, maintain a list of program identifiers and program air times and may instruct optional secondary storage device 47 or digital storage device 49 to record the programs.

The program guide may also provide users with an opportunity to parentally control titles, programs, or channels using boolean or natural language expressions. If desired, program guide server 25 may parentally control programs based on user preference profiles. A user may indicate a desire to parentally control titles, programs, or channels by, for example, selecting a selectable Parents feature 106 from main menu 102 of FIG. 5. In response, the program guide may display a criteria screen, such as illustrative criteria screens 161 and 169 of FIGS. 17a and 17b. The program guide client may display criteria screen 161 of FIG. 17a to provide a user with an opportunity to control programs, for example, according to a boolean type expression. The user may construct a boolean type expression by selecting criteria such as attribute types, attributes, and logical operators. The user may make such selections, for example, using any suitable combination of right, left, up, or down arrow key sequences to sequence through the attribute types, attributes and logical operators. In the example of FIG. 17a, the user has defined a boolean expression to lock out comedies that star Gary Shandling and that have a rating less than R. In the example of FIG. 17b, the user has defined a similar natural language expression with similar criteria.

The program guide client may submit the user defined boolean or natural language expression to program guide server 25 for processing. Program guide server 25 may process the expression, determine all of the programs that meet the expression, and indicate the programs that are locked to the program guide client when providing program listings to the program guide client using a suitable indicator (e.g., "locked" tag contained in the listings information). The program guide client may, for example, indicate that a program is locked by displaying lock indicator 161 when displaying locked listings in a listing screen, as shown, for example, in FIG. 7. By placing the processing and storage burdens of locking programs on program guide server 25 instead of user television equipment 22, more titles may be locked than would otherwise because of the limited processing and storage resources of user television equipment 22. If desired, titles, programs, or channels may also be locked using conventional parental control techniques. Program guide systems that provide users with an opportunity to parentally control titles, programs, or channels are described, for example, in above-mentioned Knudson et al. U.S. patent application Serial No. 09/357,941 filed July 16, 1999 (Attorney Docket No. UV-114).

Program guide server 25 may also record the viewing histories of users on storage device 56. Viewing histories may be created using any suitable approach. The program guide client may, for example, keep track of all of the programs that a user watches for longer than a predefined time, and record the household that the guide client is running in, the current active preference profile or profiles, the program (or its identifier), and how long the user watched the program. The program guide client may also track when users order pay-per-view programs, record programs, and schedule reminders for programs, and may also provide this information to program guide server 25 as part of the viewing histories. Other types of information may also be included in the viewing histories. User defined expressions, for example, may be stored by program guide server 25 to track what types of programs users search for. In addition, user demographic values may be calculated by program guide server 25 and used to more accurately target advertisements or recommend programs. Systems in which user demographic values are calculated are described, for example, in Knudson et al. U.S. patent application Serial No. 09/139,777, filed August 25, 1998 (Attorney Docket NO. UV-58), which is hereby incorporated by reference herein in its entirety.

The program guide client may provide the viewing history information to program guide server 25 continuously (e.g., each time the program guide client determines that a user has watched a program for the predefined time), periodically, in response to polls or requests from program guide server 25, or with any other suitable frequency. If desired, the program guide client may also monitor advertisement usage, such as what selectable advertisements users have selected. Program guide systems in which user viewing activities and advertisement usage are tracked are described, for example, in Thomas et al. U.S. patent application Serial No. 09/139,798, filed August 25, 1998 (Attorney Docket No. UV-57), which is hereby incorporated by reference herein in its entirety.

The program guide may process user profiles along with the viewer histories to present a more customized viewing experience to the user. The program guide may, for example, identify which programs or series episodes users have watched. Program guide server 25 may, for example, identify episodes that users have not yet watched and may indicate such episodes to the program guide client when the program guide client requests program listings. The program guide client in turn may indicate that a program is new to a household by, for example, displaying a suitable icon or changing the display characteristics of a listing (e.g., changing its color). FIG. 7 shows, for example, the display of New indicator 159 in list 129 to indicate to a user that the user has not seen a particular episode of Saturday Night Live. Program guide server 25 may also calculate ratings, such as Nielsen ratings, based on the viewing histories and provide such information to interested parties.

The program guide may also use the viewing history and user preferences to target the user with advertisements. Program guide systems in which users are targeted with advertisements are described, for example, in Knudson et al. U.S. patent application Serial No. 09/034,939, filed March 4, 1998 (Attorney Docket No. UV-42), which is hereby incorporated by reference herein in its entirety. Targeted advertisements may contain text, graphics, or video. Targeted advertisements may also be active objects containing various user-selectable options. For example, a targeted advertisement may allow the user to request that additional information on a product be mailed to the user's home, may allow the user to purchase a product, or may allow the user to view additional information on a product using the program guide. Targeted advertisements may be displayed in any suitable program guide display screen. The program guide client may, for example, display targeted advertisements in criteria or profile screens based on a displayed criteria, profile, or agent. Selectable advertisements 108 and advertisement banner 110, for example, may be targeted advertisements.

The program guide may make personalized viewing recommendations based on the viewing histories, preference profiles, or any suitable combination thereof. Program guide server 25 may, for example, construct relational database expressions from the viewing histories that define expressions for the program categories and ratings for programs that users have watched, scheduled reminders for, searched for, or ordered the most. Program guide server 25 may then apply user preference profile criteria to the programs, and generate personal viewing recommendations. In still another suitable approach, program guide server 25 or the program guide client may filter viewing recommendations that are generated by main facility 12 or television distribution facility 16 based on similar expressions, profiles, viewing histories, etc.

Assume, for the purpose of illustration, that a user has run the expression illustrated in FIGS. 9a and 9b, and has set the user profiles of FIGS. 13a-13f, program guide server 25 may determine that the movie Armageddon meets the criteria of the expression that was run, and also meets the criteria of the current user profile. Armageddon is a movie (strong like), an action (strong like), and does not have an illegal rating (it is rated PG-13). Program guide server 25 may indicate the movie Armageddon (or its identifier) and its air time to the program guide client and indicate to the client (e.g., using a second identifier) that a viewer recommendation for the movie is to be displayed. The program guide client may display a viewer recommendation overlay, such as overlay 2111 shown in FIGS. 20a and 20b, over a program the user is watching or over a program guide display screen, respectively. The user may press a suitable key on remote control 40 (e.g., an "info" key) to access additional information for a recommended program. An illustrative additional information screen is shown in FIG. 20c. Additional program information screens are described, for example, in above-mentioned Knudson et al. U.S. patent Application Serial No. 09/357,941 filed July 16, 1999 (Attorney Docket No. UV-114). The program guide client may tune user television equipment 22 to the channel on which a recommended viewing is aired when, for example, a user selects "Yes". If desired, recommendations may include a suitable graphic, such as a graphic indicating the recommended program.

FIGS. 21-24 show flowcharts of illustrative steps involved in performing various aspects of the present invention. The steps shown in FIGS. 21-24 are only illustrative, and may be performed in any suitable order.

FIG. 21 shows a flowchart of illustrative steps involved in storing preference profiles on program guide server 25. If desired, the steps shown may be performed in a client-server interactive program guide system in which users are not required to navigate the Internet. At step 2000, the program guide client running on user television equipment 22 provides a user with an opportunity to define a preference profile. The preference profile may include user selected or defined levels of desirability of various program characteristics, such as genre and rating. Users may define preference profiles by, for example, selecting a profile (step 2002) and selecting criteria (step 2004) such as attribute types (step 2006) and attributes (step 2008). Preference profiles may, for example, be created as database files (e.g., SQL files) containing suitable database expressions that are provided to program guide server 25. Program guide server 25 may store the preference profiles at step 2012.

Program guide data is provided from program guide server 25 to the program guide client and is displayed by the program guide client at steps 2020 and 2030, respectively. Program guide server 25 or the program guide client may use preference profiles to filter out undesirable program guide data. This may be accomplished using any suitable approach. Program guide server 25 may, for example, only provide program listings information or other program guide data that meets the preference profile or profiles to the program guide client (step 2025). Alternatively, program guide server 25 may provide program guide data, other information, or videos to the program guide client and the program guide client may filter the data, other information, or videos by displaying only those elements that meet the preference profile or profiles (step 2035).

Program guide server 25 may perform additional functions based on preference profiles if desired. Program guide server 25 may, for example, lock programs according to preference profiles (step

2040), automatically record programs according to preference profiles (step 2050), schedule reminders based on preference profiles (step 2060), or target advertising based on preference profiles (step 2070). If desired, program guide [illegible] viewing recommendations based on preference profiles at step 2080. Step 2080 may also include filtering viewing recommendations based on preference profiles provided by main facility 12 or television distribution facility 16 (step 2085).

FIG. 22 is a flowchart of illustrative steps involved in providing users with an opportunity to search program guide data in accordance with the principles of the present invention. If desired, the steps shown may be performed in a client-server interactive program guide system in which users are not required to navigate the Internet. At step 2100, the program guide client provides a user with an opportunity to define an expression, such as a boolean or natural language expression. This may include, for example, providing a user with an opportunity to select attribute types, attributes, and logical operators (steps 2102, 2104, and 2106, respectively). The user may also be provided with an opportunity to save the expression as an agent (step 2110). The program guide client provides the expression to program guide server 25 for processing at step 2120. The program guide client may for example, provide a boolean or natural language expression in a text file. Alternatively, the program guide client may construct suitable database expressions and provide the expressions to program guide server 25 as one or more suitable database files (e.g., as SQL files).

If the user indicated a desire to save an expression as an agent at step 2110, program guide server 25 may save the expression as an agent at step 2130. Otherwise, program guide server 25 may process the expression (step 2140) using any suitable approach. This may depend on how the expression was provided by the program guide client. If boolean or natural language expressions were provided as text files, for example, program guide server 25 may parse the expressions and construct a suitable database expression. Alternatively, database expressions may have been provided by the program guide client. In either approach, program guide server 25 may search its database or databases at other facilities for program guide data (e.g., program listings, additional program information, etc.), other information (e.g., software, Internet links, etc.), or videos (e.g., video-on-demand videos) and may provide the results to the program guide client at step 2150. At step 2160 the program guide client may display the results on user television equipment 22.

If the user indicated a desire to save the expression as an agent at step 2110. Program guide server 25 may save the expression as an agent using any suitable approach. Agents may be maintained, for example, in a database that program guide server 25 monitors periodically. If desired, the agent may be forwarded to other servers at other facilities, thereby providing a user with the ability to monitor multiple databases for program guide data, other information, or videos. Agents may be run automatically (e.g., databases may be queried) on one or more servers at step 2145. Step 2145 may be performed periodically, each time a database is updated, or with any other suitable frequency. Program guide server 25 may provide its results and the results of other servers (if desired) to the program guide client at step 2155. The program guide client may display the results at 2165. The results may be displayed, for example, in the form of reminders for which reminder information was provided at step 2155.

FIG. 23 shows a flowchart of illustrative steps involved in processing and using expressions on program guide server 25 in accordance with the principles of the present invention. If desired, the steps shown may be performed in a client-server interactive program guide system in which users are not required to navigate the Internet. The program guide client provides users with an opportunity to define an expression (e.g., boolean or natural language expressions) at step 2100. This may include, for example, providing a user with an opportunity to select attribute types, attributes and logical operators (steps 2102, 2104, and 2106, respectively). The program guide client provides the expression to program guide server 25 for processing at step 2210 as any suitable type of file. The program guide client may for example, provide a boolean or natural language expression in a text file. Alternatively, the program guide client may construct suitable database expressions and provide the expressions to program guide server 25 as one or more suitable database files (e.g., as SQL files).

Program guide server 25 may process the expression (step 2220) using any suitable approach depending on how the expression was provided to program guide server 25 from the program guide client. If boolean or natural language expressions were provided as text files, for example, program guide server 25 may parse the expressions and construct a suitable database expression. Alternatively, database expressions may have been provided to program guide server 25 from the program guide client. In either approach, program guide server 25 may search its database or databases at other facilities and may provide the results to the program guide client or use the results to perform any suitable program guide function.

Reminders may be scheduled based on the results of the search (step 2230). Program guide server 25 may, for example, store reminder information (e.g., program identifiers and air times) at step 2235 and send messages to the program guide client at an appropriate time before a program starts. In another suitable approach, program guide server 25 may process an expression and provide program identifiers and air times to the program guide client. The program guide client may, for example, maintain a list of program identifiers and display program reminders at an appropriate time before the programs start.

Programs may also be automatically recorded by program guide server 25 or user television equipment 22 based on the results of the expression (step 2240). Program guide server 25 may, for example, provide program identifiers and air times to the program guide client. The program guide client may, for example, maintain a list of program identifiers and program air times and may instruct optional secondary storage device 47 or digital storage device 49 to record the programs at the appropriate time.

Programs may be parentally locked based on the expression results (step 2250). Program guide server 25 may, for example, store parental control information (e.g., program identifiers in a database, table, or list of programs to be locked) at step 2260. Program guide server 25 may indicate to the program guide client that programs are locked when providing program listings to the program guide client. Alternatively, program guide server 25 may indicate to the program guide client the programs that were found as a result of the expression. The program guide client may lock the programs locally using any suitable approach. The program guide client may, for example, indicate that a program is locked by displaying lock indicator 161 when displaying locked listings in a listing screen, as shown, for example, in FIG. 7.

FIG. 24 shows a flowchart of illustrative steps involved in tracking and using viewing histories in accordance with the principles of the present invention. If desired, the steps shown may be performed in a client-server interactive program guide system in which users are not required to navigate the Internet. Viewing histories are tracked at step 2300. This may include tracking programs that users watch (step 2310), tracking reminders scheduled by a user with program guide server 25 or using conventional techniques (step 2320), tracking pay-per-view programs that the user orders (step 2330), advertisement usage (step 2335), track recorded programs (step 2337), track any other suitable user activity, or any suitable combination thereof. The program guide client may provide the viewing history information to program guide server 25 continuously (i.e., each time the program guide client determines that a user has watched a program for the predefined time), periodically, in response to polls or requests from program guide server 25, or with any other suitable frequency.

The viewing history tracked in steps 2310-2335 may be stored on program guide server 25 at step 2340. If desired, user-defined expressions that are processed by program guide server 25 may also be stored on program guide server 25 (step 2345). User demographic values may be calculated by program guide server 25 at step 2347. The viewing history and its expressions and user demographic values may be used by program guide server 25 to perform any suitable function. Program guide server 25 may, for example, collect program rating information (step 2350), or target advertising (step 2360).

Program guide server 25 may search its or another server's database for programs that are consistent with the viewing history (step 2370). If desired, program guide server 25 may find programs that are also consistent with preference profiles stored by program guide server 25 (step 2375). Program guide server may perform any suitable function using the results of the search. Program guide server 25 may, for example, identify episodes of programs that are new to a user (step 2380), or provide viewing recommendations in the form of, for example, reminders or recommendations for non-program items (e.g., software, Internet links, etc.) (step 2390).

The foregoing is merely illustrative of the principles of this invention and various modifications can be made by those skilled in the art without departing from the scope and spirit of the invention.

Claims

What is claimed is:

1. A method for use in a client-server interactive television program guide system comprising:

providing a user with an opportunity to define user preferences using an interactive television program guide client that is implemented on user television equipment, without requiring the user to navigate the Internet;

providing the user preferences to a program guide server; and

providing program guide data to the program guide client according to the user preferences.

2. The method defined in claim 1 further comprising:

generating a viewing recommendation based on the user preferences with the program guide server; and

displaying the user preferences with the interactive television program guide client on the user television equipment.

3. The method defined in claim 1 wherein providing a user with an opportunity to define user preferences comprises providing a user with an opportunity to designate a preference level for a plurality of preference attributes.

4. The method defined in claim 1 further comprising providing software to the program guide client according to the user preferences.

5. The system defined in claim 1 further comprising providing Internet links to the program guide client according to the user preferences.

6. A method for use in a client-server interactive television program guide system for scheduling reminders according to user defined expressions, comprising:

providing a user with an opportunity to define an expression with an interactive television program guide client implemented on user television equipment without requiring the user to navigate the Internet;

storing the expression on a program guide server;

processing the expression with the program guide server to find programs that satisfy the expression; and

scheduling with the program guide server reminders for programs that satisfy the expression.

7. The method defined in claim 6 wherein scheduling with the program guide server reminders for programs that satisfy the expression comprises providing at least one message from the program guide server to the program guide client before each of the programs that satisfy the expression begin.

8. The method defined in claim 6 wherein scheduling with the program guide server reminders for programs that satisfy the expression comprises providing program identifiers for each of the programs

that satisfy the expression from the program guide server to the program guide client.

9. A method for use in a client-server interactive television program guide system for scheduling programs for recording according to user defined expressions, comprising:

providing a user with an opportunity to define an expression with an interactive television program guide client implemented on user television equipment without requiring the user to navigate the Internet;

storing the expression on a program guide server;

processing the expression with the program guide server to find programs that satisfy the expression; and

scheduling with the program guide server the programs that satisfy the expression for recording.

10. The method defined in claim 9 wherein scheduling with the program guide server the programs that satisfy the expression for recording comprises scheduling with the program guide server the programs that satisfy the expression for recording by the user television equipment.

11. The method defined in claim 9 wherein scheduling with the program guide server the programs that satisfy the expression for recording comprises scheduling with the program guide server the programs that satisfy the expression for recording by the program guide server.

12. A method for use in a client-server interactive television program guide system for parentally controlling programs according to user defined expressions, comprising:

providing a user with an opportunity to define an expression with an interactive television program guide client implemented on user television equipment without requiring the user to navigate the Internet;

storing the expression on a program guide server;

processing the expression with the program guide server to find programs that satisfy the expression; and

locking with the program guide server programs that satisfy the expression.

13. The method defined in claim 12 wherein locking with the program guide server programs that satisfy the expression comprises indicating to the program guide client that the programs that satisfy the expression are locked.

14. A method for use in a client-server interactive television program guide system for tracking a user's viewing history, comprising:

tracking a user's viewing history;

storing the user's viewing history on a program guide server;

finding programs with the program guide server that are consistent with the user's viewing history; and

indicating on user television equipment the programs found by the program guide server that are consistent with the user's viewing history and that the user has not watched, with an interactive television program guide client implemented on the user television equipment.

15. The method defined in claim 14 wherein storing the user's viewing history comprises storing a user defined expression with the program guide server.

16. The method defined in claim 14 wherein storing the user's viewing history comprises calculating user demographic values with the program guide server.

17. The method defined in claim 14 further comprising:

providing a user with an opportunity to define a user preference profile with the interactive television program guide client implemented on user television equipment;

storing the user preference profile on a program guide server; and

finding programs with the program guide server that are consistent with the user preference profile, wherein:

indicating on user television equipment the programs found by the program guide server that are consistent with the user's viewing history and that the user has not watched comprises indicating on user television equipment the programs found by the program guide server that are consistent with the user's viewing history and the user preference profile and that the user has not watched.

18. The method defined in claim 14 further comprising:

targeting advertising with the program guide server based on the user's viewing history; and

displaying the advertising with the interactive television program guide client on the user television equipment.

19. The method defined in claim 14 further comprising collecting program ratings information with the program guide server based on the user's viewing history.

20. A client-server interactive television program guide system comprising:

means for providing a user with an opportunity to define user preferences using an interactive television program guide client that is implemented on user television equipment, without requiring the user to navigate the Internet;

means for providing the user preferences to a program guide server; and

means for providing program guide data from the program guide server to the program guide client according to the user preferences.

21. The system defined in claim 20 further comprising:

means for generating a viewing recommendation based on the user preferences with the program guide server; and

means for displaying the user preferences with the interactive television program guide client on the user television equipment.

22. The system defined in claim 20 wherein the means for providing a user with an opportunity to

define user preferences comprises means for providing a user with an opportunity to designate a preference level for a plurality of preference attributes.

23. The system defined in claim 20 further comprising means for providing software from the program guide server to the program guide client according to the user preferences.

24. The system defined in claim 20 further comprising means for providing Internet links from the program guide server to the program guide client according to the user preferences.

25. A client-server interactive television program guide system for scheduling reminders according to user defined expressions, comprising:

means for providing a user with an opportunity to define an expression with an interactive television program guide client implemented on user television equipment, without requiring the user to navigate the Internet;

means for processing the expression with a program guide server to find programs that satisfy the expression; and

means for scheduling with the program guide server reminders for programs that satisfy the expression.

26. The system defined in claim 25 wherein the means for scheduling with the program guide server reminders for programs that satisfy the expression comprises means for providing at least one message from the program guide server to the program guide client

before each of the programs that satisfy the expression begin.

27. The system defined in claim 25 wherein the means for scheduling with the program guide server reminders for programs that satisfy the expression comprises means for providing program identifiers for each of the programs that satisfy the expression from the program guide server to the program guide client.

28. A client-server interactive television program guide system for scheduling programs for recording according to user defined expressions, comprising:

means for providing a user with an opportunity to define an expression with an interactive television program guide client implemented on user television equipment, without requiring the user to navigate the Internet;

means for processing the expression with a program guide server to find programs that satisfy the expression; and

means for scheduling with the program guide server the programs that satisfy the expression for recording.

29. The system defined in claim 28 wherein the means for scheduling with the program guide server the programs that satisfy the expression for recording comprises means for scheduling with the program guide server the programs that satisfy the expression for recording by the user television equipment.

30. The system defined in claim 28 wherein the means for scheduling with the program guide server

the programs that satisfy the expression for recording comprises means for scheduling with the program guide server the programs that satisfy the expression for recording by the program guide server.

31. A client-server interactive television program guide system for parentally controlling programs according to user defined expressions, comprising:

means for providing a user with an opportunity to define an expression with an interactive television program guide client implemented on user television equipment, without requiring the user to navigate the Internet;

means for processing the expression with a program guide server to find programs that satisfy the expression; and

means for locking with the program guide server programs that satisfy the expression.

32. The system defined in claim 31 wherein the means for locking with the program guide server programs that satisfy the expression comprises means for indicating to the program guide client that the programs that satisfy the expression are locked.

33. A client-server interactive television program guide system for tracking a user's viewing history, comprising:

means for tracking a user's viewing history with a program guide server;

means for indicating on user television equipment programs that are consistent with the user's viewing history and that the user has not watched, with an interactive television program guide client implemented on the user television equipment.

34. The system defined in claim 33 wherein the means for tracking the user's viewing history comprises means for storing a user defined expression with the program guide server.

35. The system defined in claim 33 wherein the means for tracking the user's viewing history comprises means for calculating user demographic values with the program guide server.

36. The system defined in claim 33 further comprising:

means for providing a user with an opportunity to define a user preference profile with the interactive television program guide client implemented on user television equipment; and

means for finding programs with the program guide server that are consistent with the user preference profile, wherein:

the means for indicating on user television equipment the programs found by the program guide server that are consistent with the user's viewing history and that the user has not watched comprises means for indicating on user television equipment the programs found by the program guide server that are consistent with the user's viewing history and the user preference profile and that the user has not watched.

37. The system defined in claim 36 further comprising:

means for targeting advertising with the program guide server based on the user's viewing history; and

means for displaying the advertising with the interactive television program guide client on the user television equipment.

38. The system defined in claim 36 further comprising means for collecting program ratings information with the program guide server based on the user's viewing history.

39. A client-server interactive television program guide system comprising:

a program guide server;

user television equipment on which an interactive television program guide client is implemented, wherein the interactive television program guide client is programmed to provide a user with an opportunity to define user preferences without requiring the user to navigate the Internet; and

a communications path over which the user preferences are provided by the interactive television program guide client to the program guide server.

40. The system defined in claim 39 wherein:

the program guide server is programmed to generate a viewing recommendation based on the user preferences; and

the interactive television program guide client is further programmed to display the viewing recommendation on the user television equipment.

41. The system defined in claim 39 wherein the interactive television program guide client is further programmed to provide a user with an opportunity to designate a preference level for a plurality of preference attributes.

42. The system defined in claim 39 wherein the program guide server is programmed to provide software to the interactive television program guide client according to the user preferences.

43. The system defined in claim 39 wherein the program guide server is programmed to provide Internet links to the interactive television program guide client according to the user preferences.

44. A client-server interactive television program guide system for scheduling reminders according to user defined expressions, comprising:

user television equipment on which an interactive television program guide client is implemented, wherein the program guide client is programmed to provide a user with an opportunity to define an expression without requiring the user to navigate the Internet;

a communications path over which the expression is provided by the interactive television program guide client to a program guide server, wherein the program guide server is programmed to find programs that satisfy the expression and schedule reminders for programs that satisfy the expression.

45. The system defined in claim 44 wherein scheduling with the program guide server reminders for programs that satisfy the expression comprises

providing at least one message from the program guide server to the program guide client before each of the programs that satisfy the expression begin.

46. The system defined in claim 44 wherein the program guide server is further programmed to provide program identifiers for each of the programs that satisfy the expression to the interactive television program guide client over the communications path.

47. A client-server interactive television program guide system for scheduling programs for recording according to user defined expressions, comprising:

user television equipment on which an interactive television program guide client is implemented, wherein the interactive television program guide client is programmed to provide a user with an opportunity to define an expression without requiring the user to navigate the Internet;

a communications path over which the expression is provided by the interactive television program guide client to a program guide server, wherein the program guide server is programmed to find programs that satisfy the expression and schedule the programs that satisfy the expression for recording.

48. The system defined in claim 47 wherein:

the user television equipment comprises a storage device; and

the program guide server is further programmed to schedule the programs that satisfy the expression for recording by the storage device.

49. The system defined in claim 47 wherein the program guide server comprises a storage device on which the programs that satisfy the expression are stored.

50. A client-server interactive television program guide system for parentally controlling programs according to user defined expressions, comprising:

user television equipment on which an interactive television program guide client is implemented, wherein the interactive television program guide client is programmed to provide a user with an opportunity to define an expression without requiring the user to navigate the Internet;

a communications path over which the interactive television program guide client provides the expression to a program guide server, wherein the program guide server is programmed to find programs that satisfy the expression and lock programs that satisfy the expression.

51. The system defined in claim 50 wherein the program guide server is programmed to indicate to the interactive television program guide client the locked programs over the communications path; and

the interactive television program guide client is further programmed to indicate to the user the locked programs with the user television equipment.

52. A client-server interactive television program guide system for tracking a user's viewing history, comprising:

user television equipment on which an interactive television program guide client is implemented, wherein the interactive television program guide client is programmed to provide viewing history information to a program guide server over a communications path, wherein:

the program guide server is programmed to find programs based on the viewing history information and to indicate the programs to the interactive television program guide client over the communications path; and

the interactive television program guide client is further programmed to indicate on the user television equipment a subset of the programs wherein the subset of the programs are programs that the user has not watched.

53. The system defined in claim 52 wherein the program guide server is further programmed to calculate user demographic values based on the viewing history information.

54. The system defined in claim 52 wherein:

the interactive television program guide client is further programmed to provide user preference information to the program guide server over the communications path; and

the program guide server is further programmed to obtain programs based on the user preference information and to indicate the programs to the interactive television program guide client.

55. The system defined in claim 54 wherein:

the program guide server is programmed to target advertisements based on the user preference information and to provide the advertisements to the interactive television program guide client over the communications path; and

the interactive television program guide client is further programmed to display the advertisements on the user television equipment.

56. The system defined in claim 54 wherein the program guide server is further programmed to collect program ratings information based on the viewing history information.

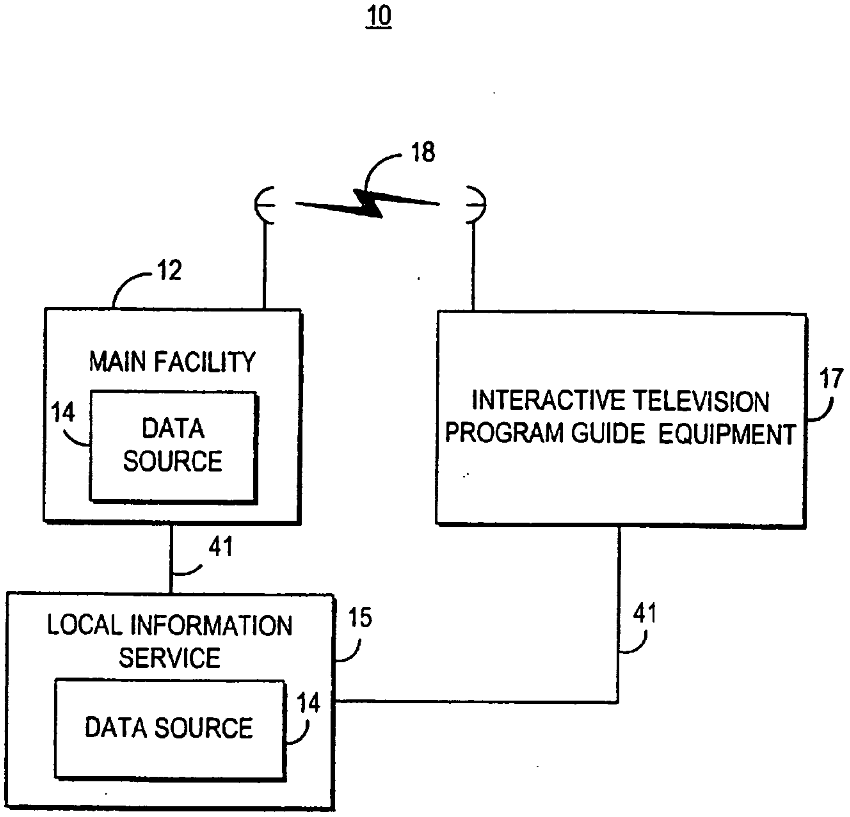

10
18
12
MAIN FACILITY
14
DATA SOURCE
INTERACTIVE TELEVISION PROGRAM GUIDE EQUIPMENT
17
41
LOCAL INFORMATION SERVICE
15
41
DATA SOURCE
14

FIG. 1

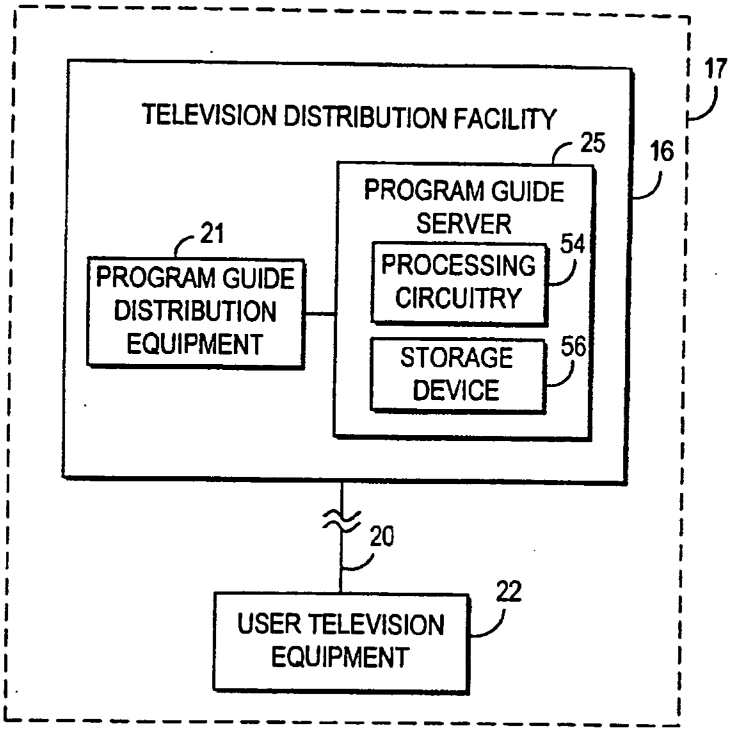

TELEVISION DISTRIBUTION FACILITY
25
PROGRAM GUIDE SERVER
21
PROGRAM GUIDE DISTRIBUTION EQUIPMENT
PROCESSING CIRCUITRY
54
STORAGE DEVICE
56
16
17
20
22
USER TELEVISION EQUIPMENT

FIG. 2a

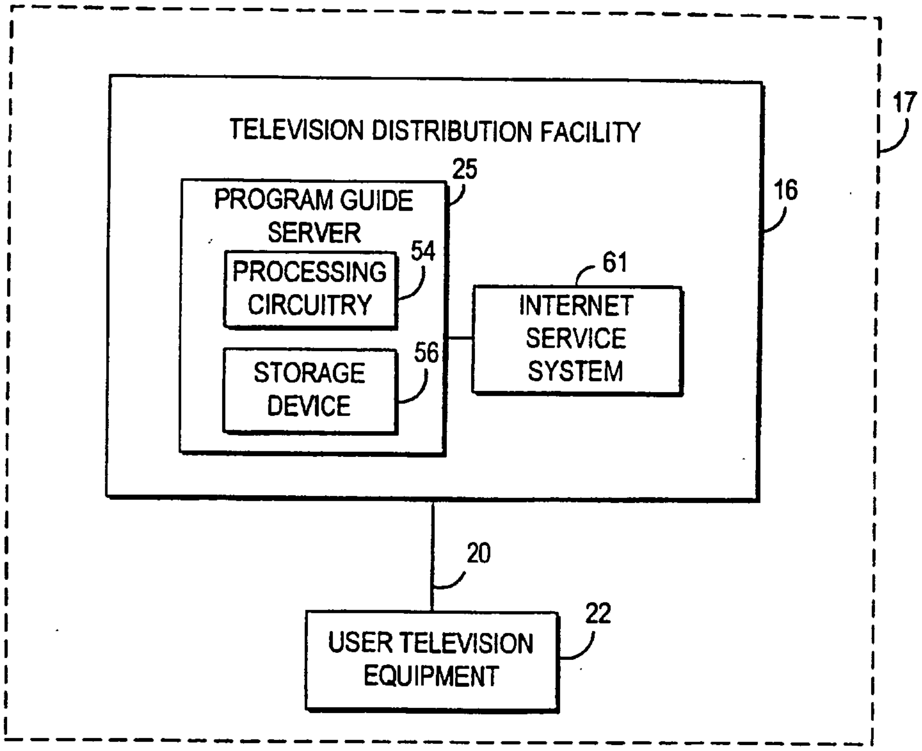

TELEVISION DISTRIBUTION FACILITY
PROGRAM GUIDE SERVER
25
PROCESSING CIRCUITRY
54
STORAGE DEVICE
56
INTERNET SERVICE SYSTEM
61
16
17
20
USER TELEVISION EQUIPMENT
22

FIG. 2b

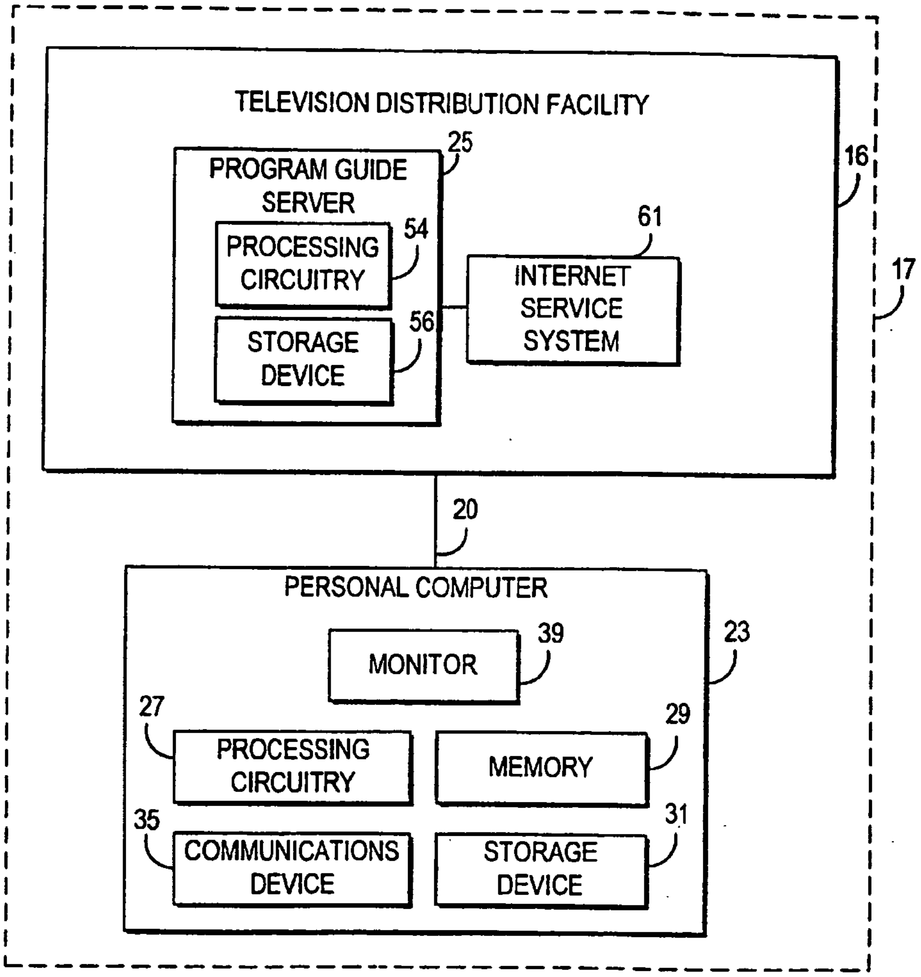

TELEVISION DISTRIBUTION FACILITY
PROGRAM GUIDE SERVER
25
PROCESSING CIRCUITRY
54
STORAGE DEVICE
56
61
INTERNET SERVICE SYSTEM
16
17
20
PERSONAL COMPUTER
MONITOR
39
23
27
PROCESSING CIRCUITRY
MEMORY
29
35
COMMUNICATIONS DEVICE
STORAGE DEVICE
31

FIG. 2c

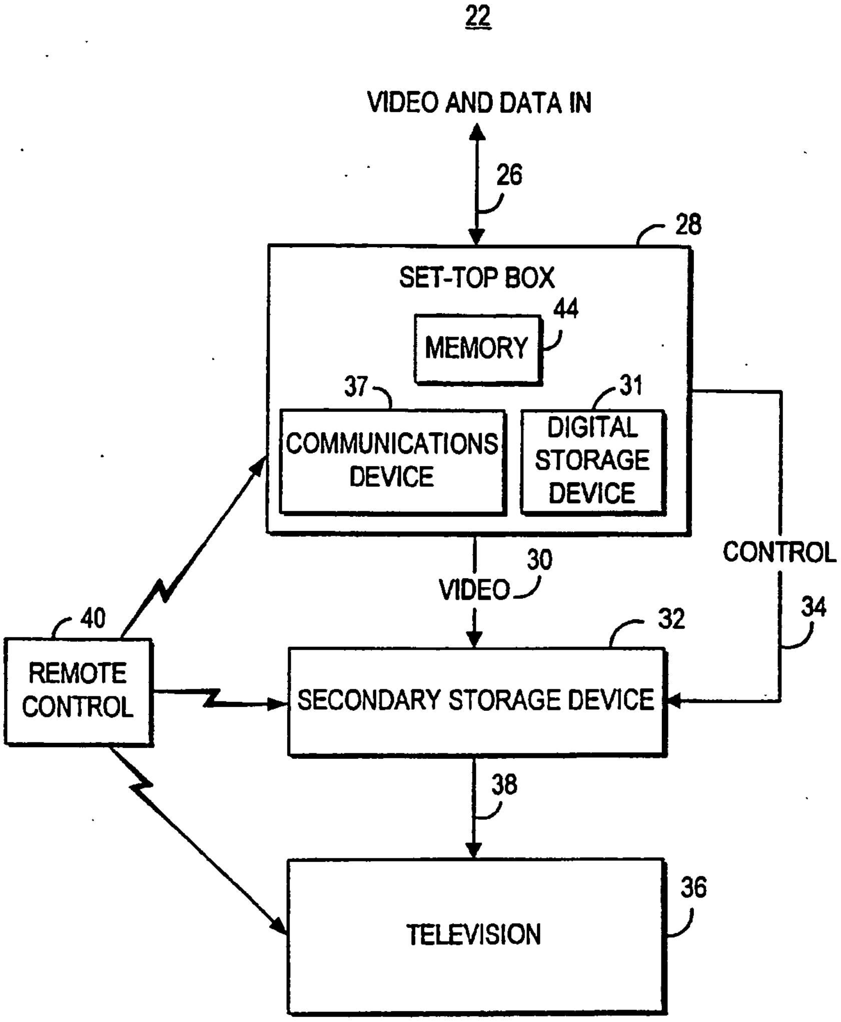

22
VIDEO AND DATA IN
26
28
SET-TOP BOX
44
MEMORY
37
31
COMMUNICATIONS DEVICE
DIGITAL STORAGE DEVICE
30
VIDEO
CONTROL
34
40
32
REMOTE CONTROL
SECONDARY STORAGE DEVICE
38
36
TELEVISION

FIG. 3

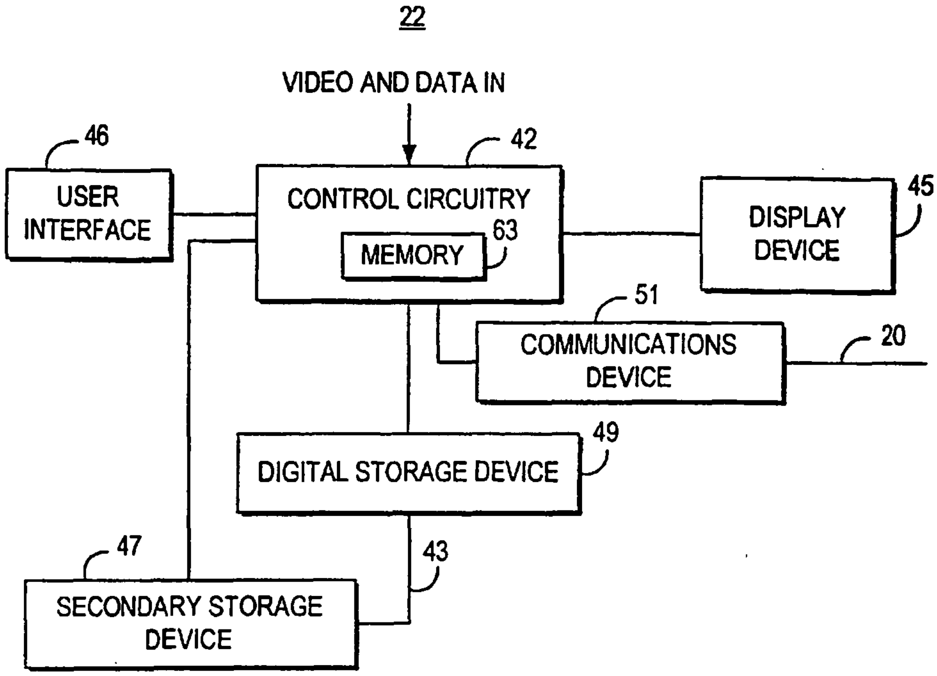

22
VIDEO AND DATA IN
46
USER INTERFACE
42
CONTROL CIRCUITRY
MEMORY
63
45
DISPLAY DEVICE
51
COMMUNICATIONS DEVICE
20
DIGITAL STORAGE DEVICE
49
47
43
SECONDARY STORAGE DEVICE

FIG. 4

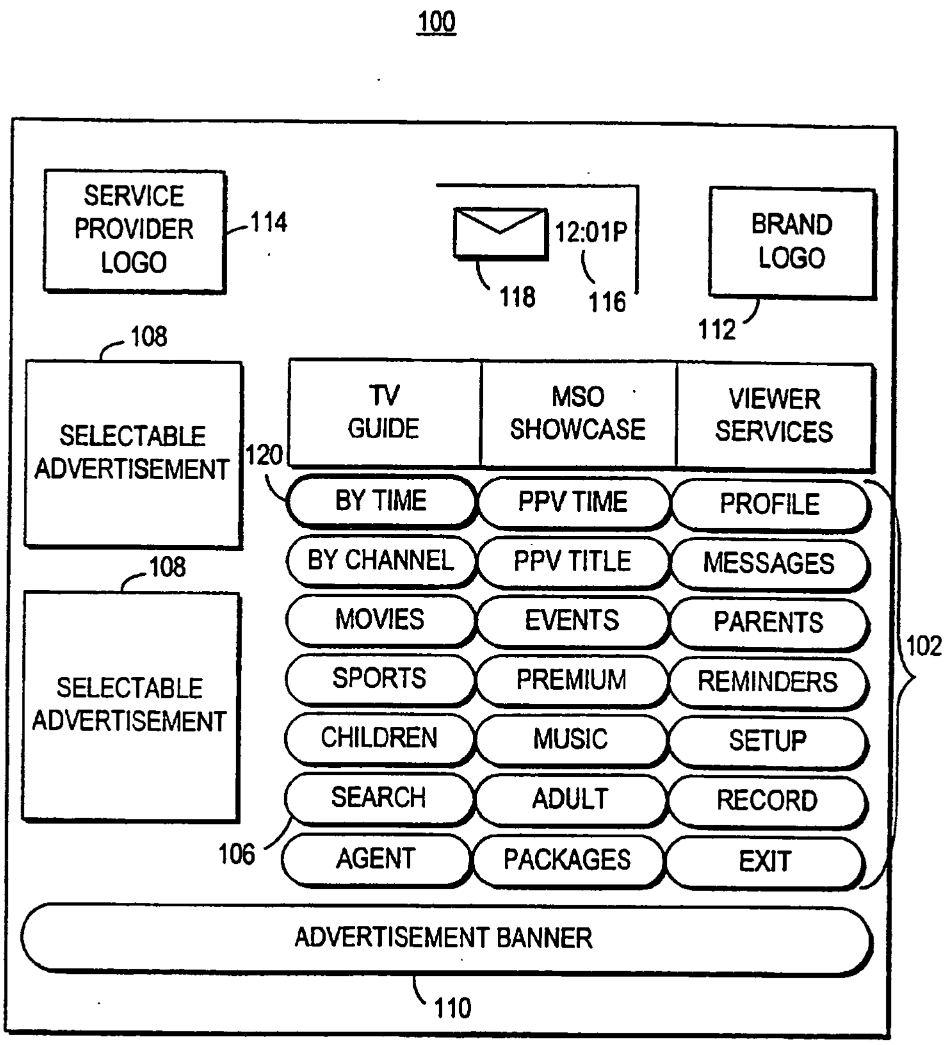

100
SERVICE PROVIDER LOGO
114
12:01P
118
116
BRAND LOGO
112
108
SELECTABLE ADVERTISEMENT
108
SELECTABLE ADVERTISEMENT
TV GUIDE
MSO SHOWCASE
VIEWER SERVICES
120
BY TIME
PPV TIME
PROFILE
BY CHANNEL
PPV TITLE
MESSAGES
MOVIES
EVENTS
PARENTS
SPORTS
PREMIUM
REMINDERS
CHILDREN
MUSIC
SETUP
SEARCH
ADULT
RECORD
106
AGENT
PACKAGES
EXIT
102
ADVERTISEMENT BANNER
110

FIG. 5

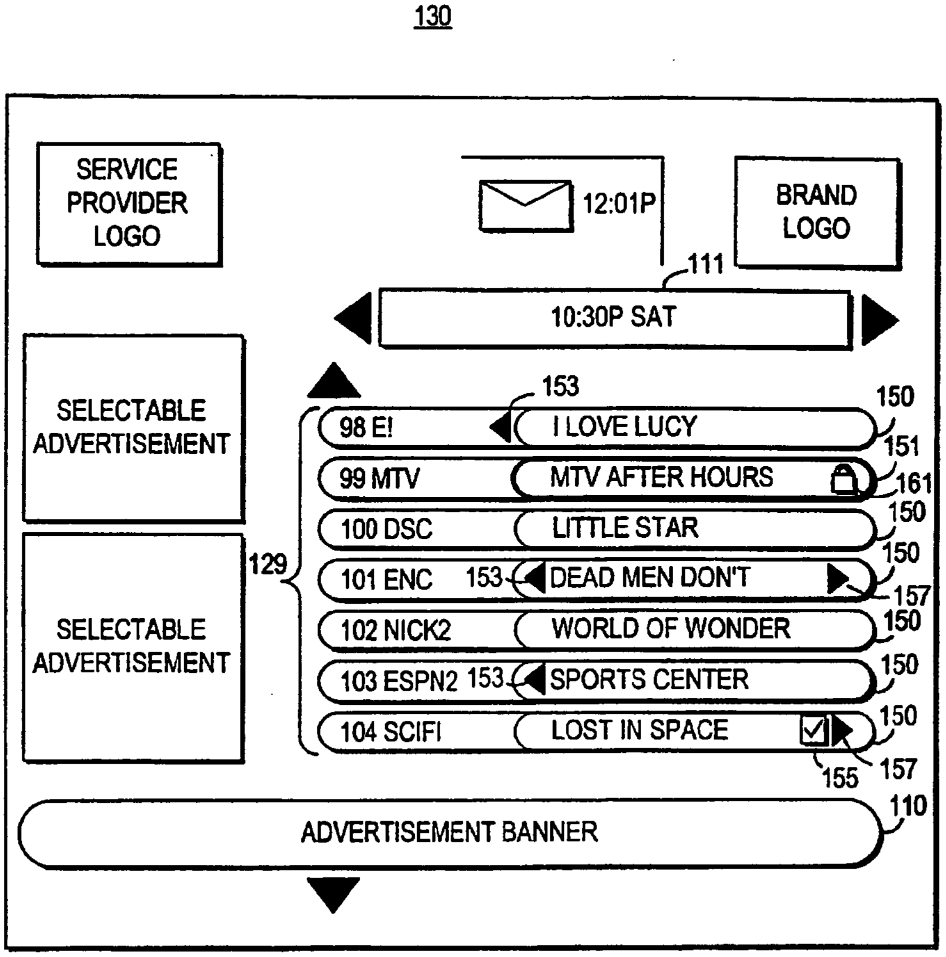

130
SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
111
10:30P SAT
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
129
153
150
98 E!
I LOVE LUCY
151
99 MTV
MTV AFTER HOURS
161
150
100 DSC
LITTLE STAR
150
101 ENC
153
DEAD MEN DON'T
157
150
102 NICK2
WORLD OF WONDER
150
103 ESPN2
153
SPORTS CENTER
150
104 SCIFI
LOST IN SPACE
155
157
110
ADVERTISEMENT BANNER

FIG. 6

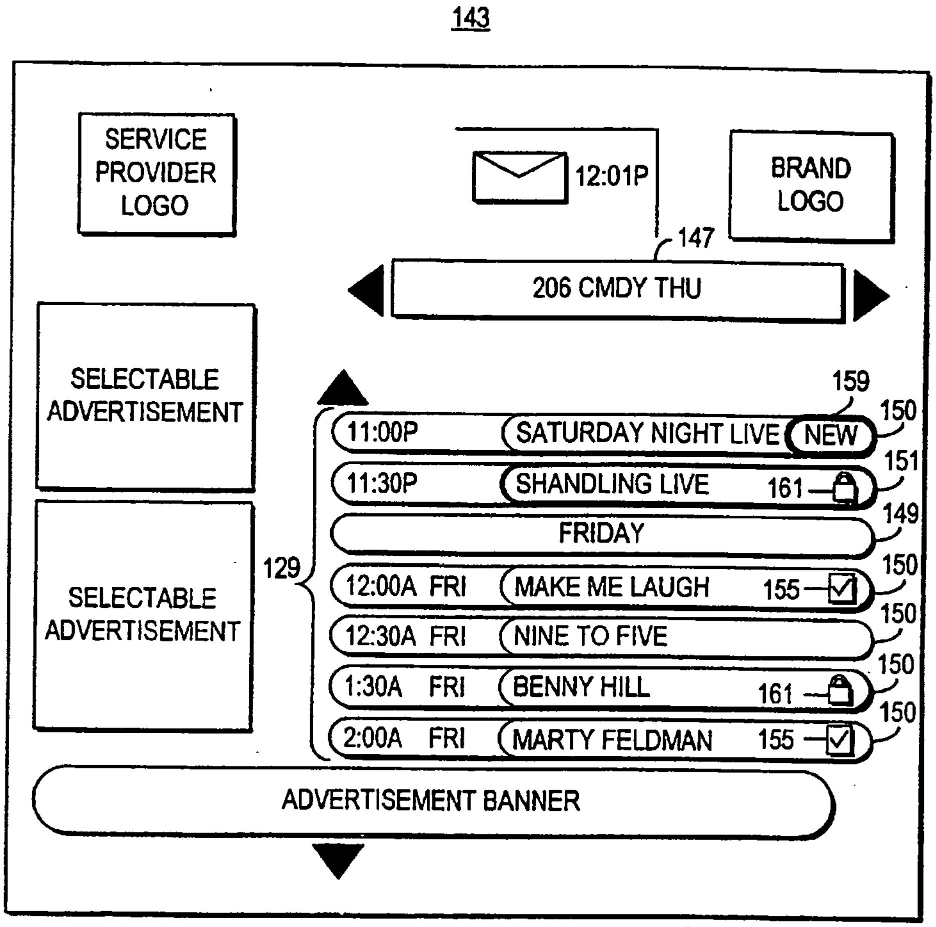

143
SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
147
206 CMDY THU
SELECTABLE ADVERTISEMENT
159
150
11:00P
SATURDAY NIGHT LIVE
NEW
151
11:30P
SHANDLING LIVE
161
149
FRIDAY
129
SELECTABLE ADVERTISEMENT
150
12:00A FRI
MAKE ME LAUGH
155
150
12:30A FRI
NINE TO FIVE
150
1:30A FRI
BENNY HILL
161
150
2:00A FRI
MARTY FELDMAN
155
ADVERTISEMENT BANNER

FIG. 7

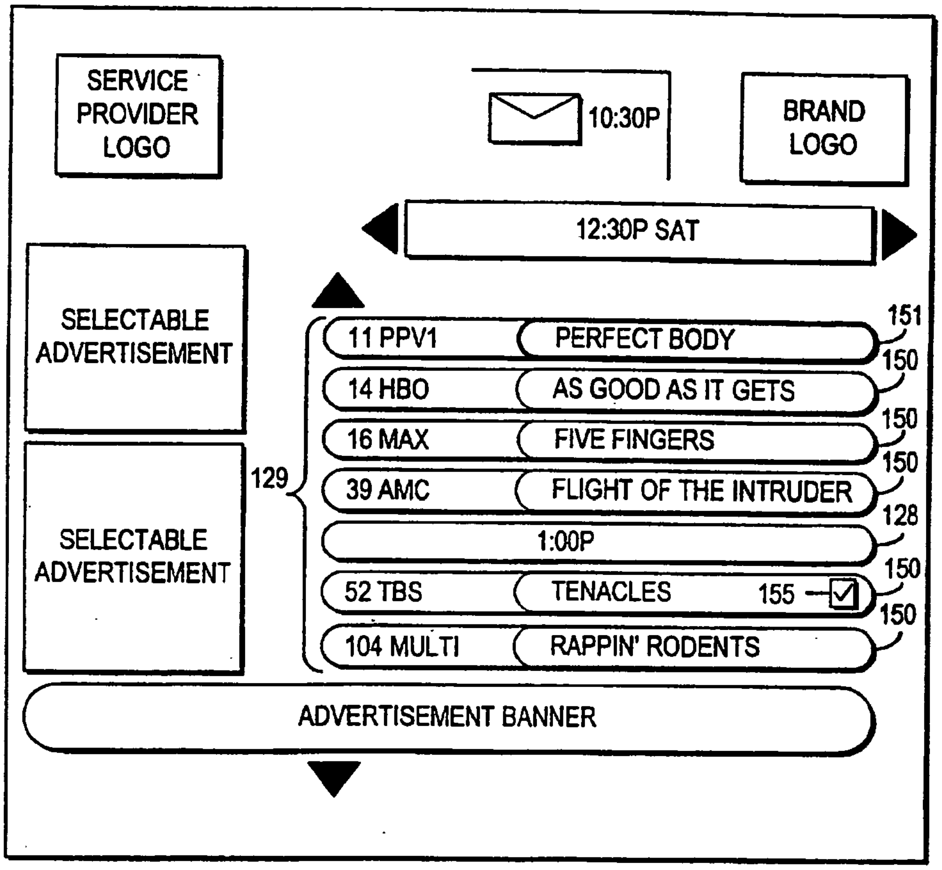

SERVICE PROVIDER LOGO
10:30P
BRAND LOGO
12:30P SAT
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
129
11 PPV1
PERFECT BODY
151
14 HBO
AS GOOD AS IT GETS
150
16 MAX
FIVE FINGERS
150
39 AMC
FLIGHT OF THE INTRUDER
150
1:00P
128
52 TBS
TENACLES
155
150
104 MULTI
RAPPIN' RODENTS
150
ADVERTISEMENT BANNER

FIG. 8a

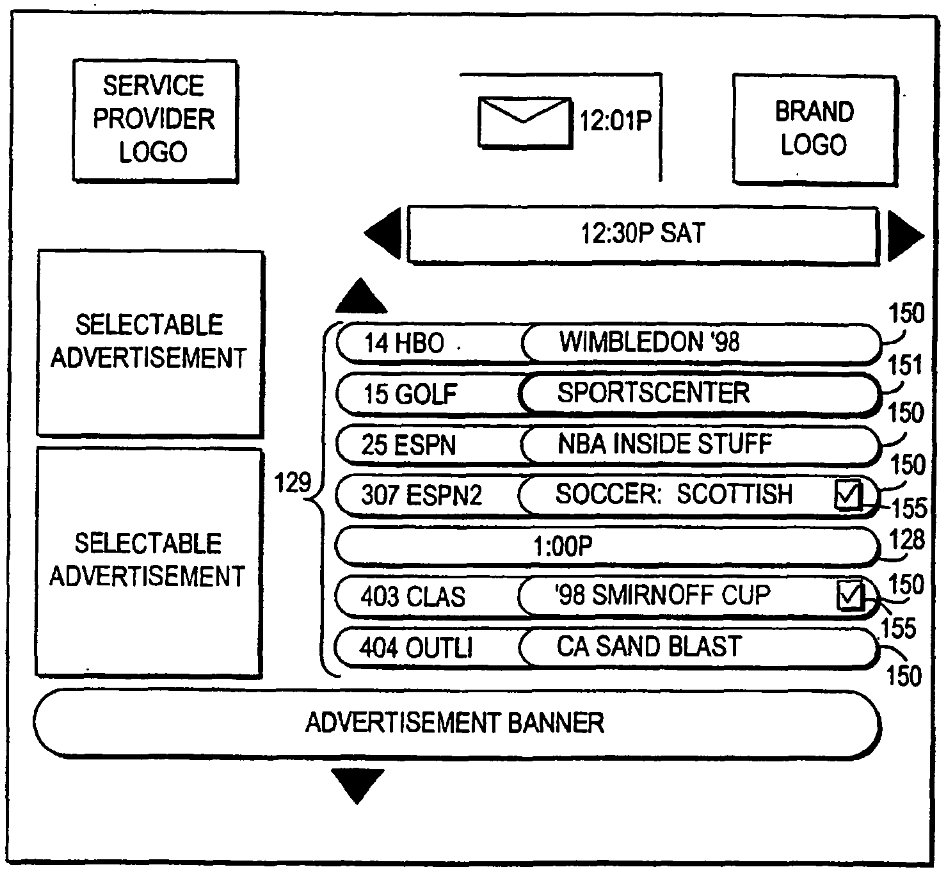

SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
12:30P SAT
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
129
14 HBO
WIMBLEDON '98
150
15 GOLF
SPORTSCENTER
151
25 ESPN
NBA INSIDE STUFF
150
307 ESPN2
SOCCER: SCOTTISH ☑
150
155
1:00P
128
403 CLAS
'98 SMIRNOFF CUP ☑
150
155
404 OUTLI
CA SAND BLAST
150
ADVERTISEMENT BANNER

FIG. 8b

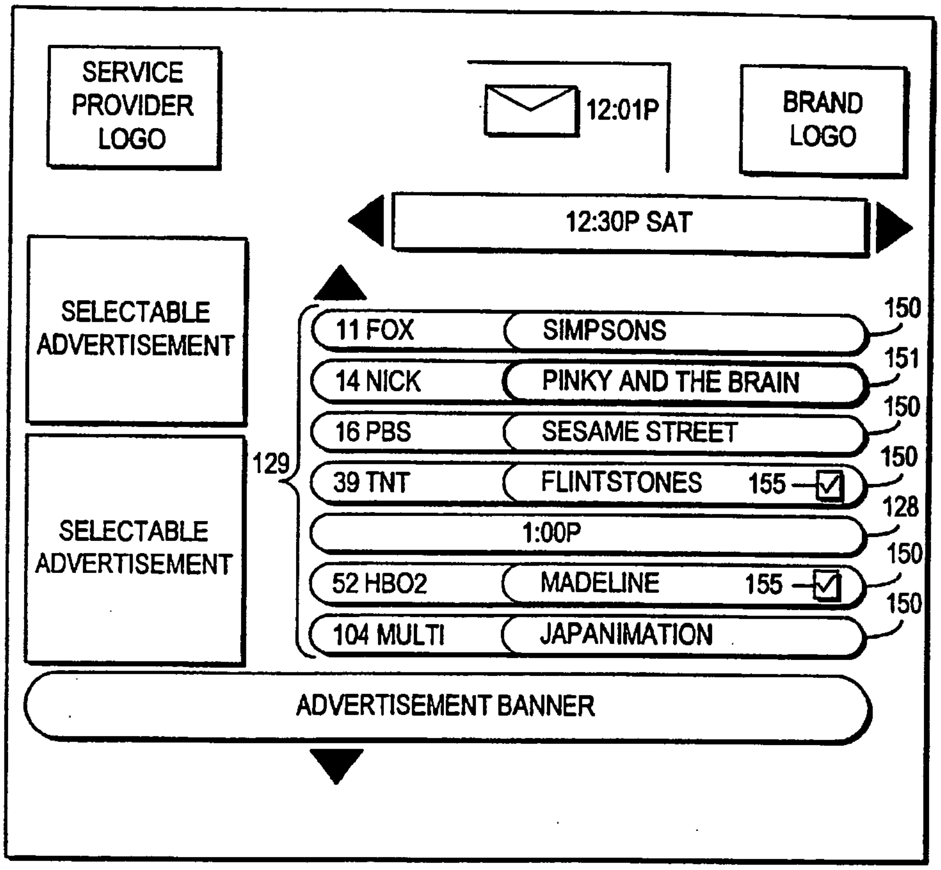

SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
12:30P SAT
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
129
11 FOX
SIMPSONS
150
14 NICK
PINKY AND THE BRAIN
151
16 PBS
SESAME STREET
150
39 TNT
FLINTSTONES
155
150
1:00P
128
52 HBO2
MADELINE
155
150
104 MULTI
JAPANIMATION
150
ADVERTISEMENT BANNER

FIG. 8c

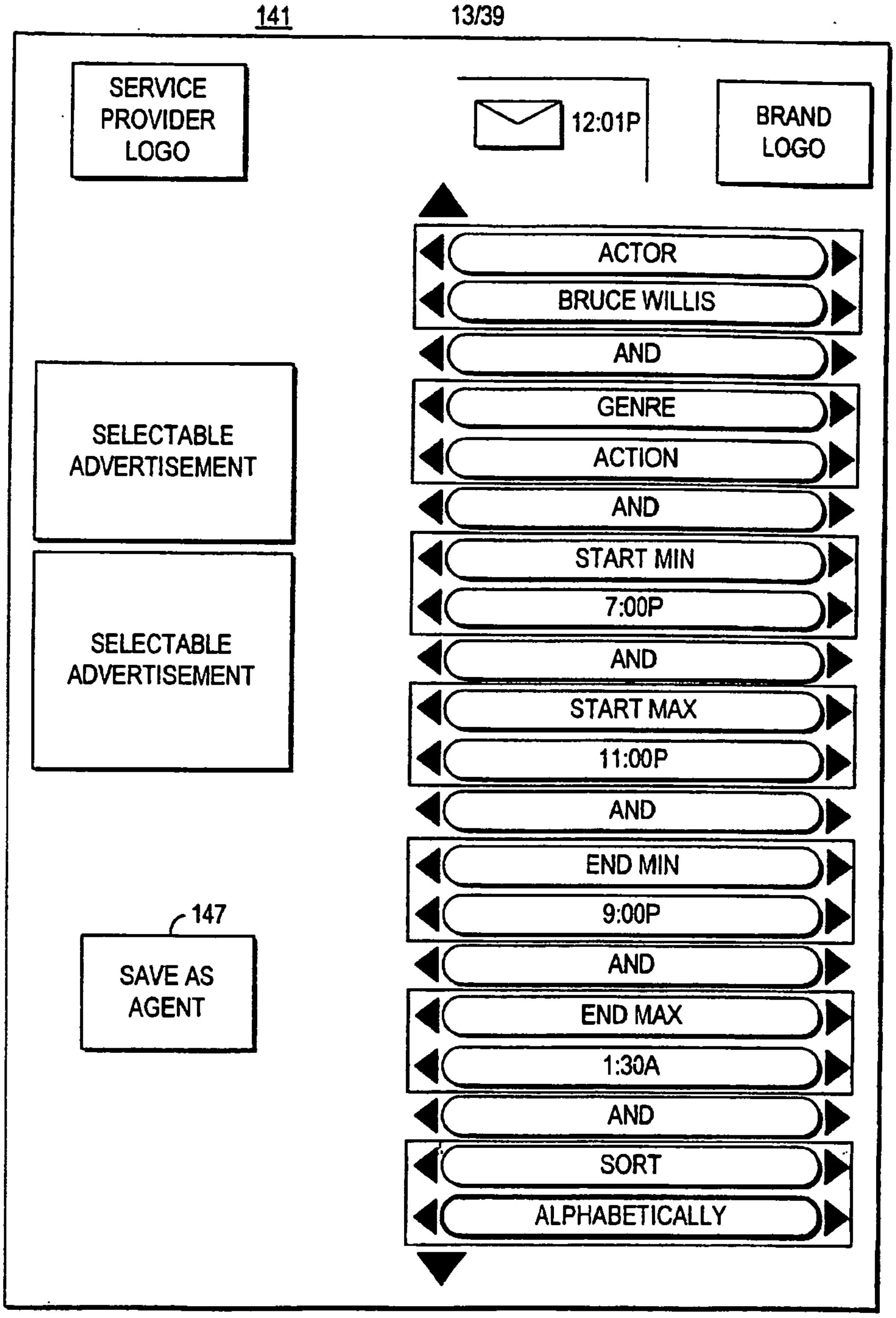
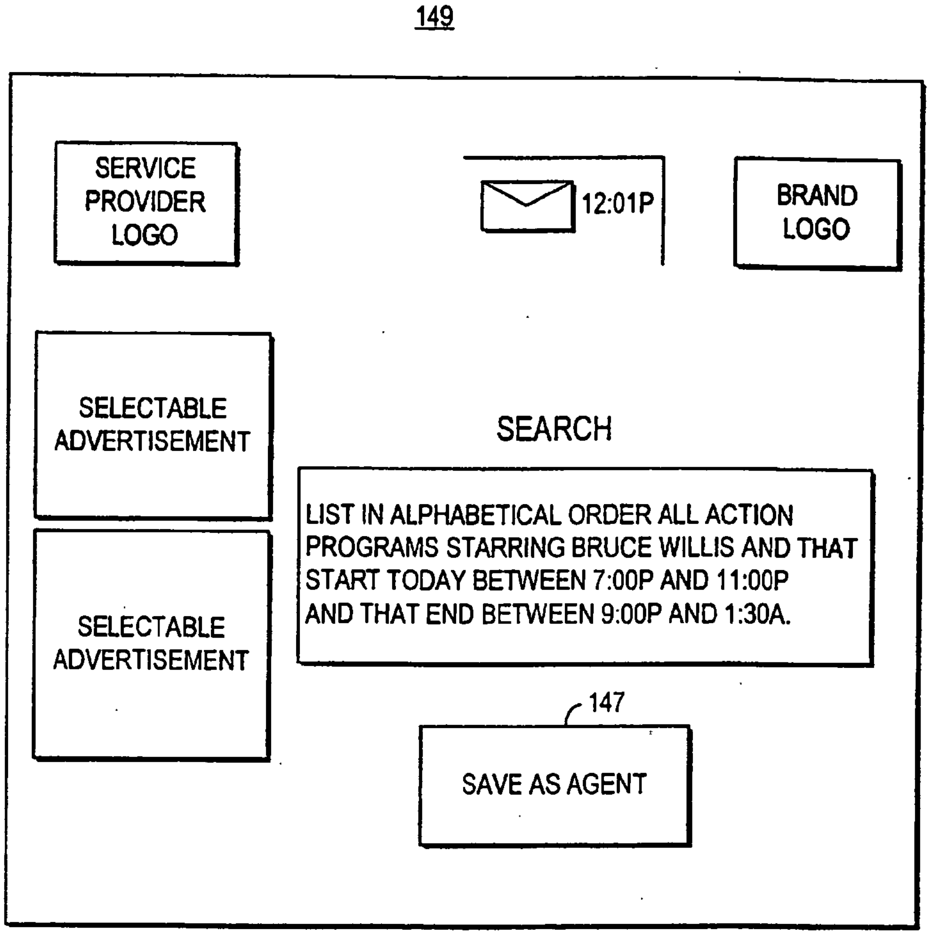

141
SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
ACTOR
BRUCE WILLIS
AND
GENRE
ACTION
AND
START MIN
7:00P
AND
START MAX
11:00P
AND
END MIN
9:00P
AND
END MAX
1:30A
AND
SORT
ALPHABETICALLY
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
147
SAVE AS AGENT

FIG. 9a



LIST IN ALPHABETICAL ORDER ALL ACTION PROGRAMS STARRING BRUCE WILLIS AND THAT START TODAY BETWEEN 7:00P AND 11:00P AND THAT END BETWEEN 9:00P AND 1:30A.

FIG. 9b

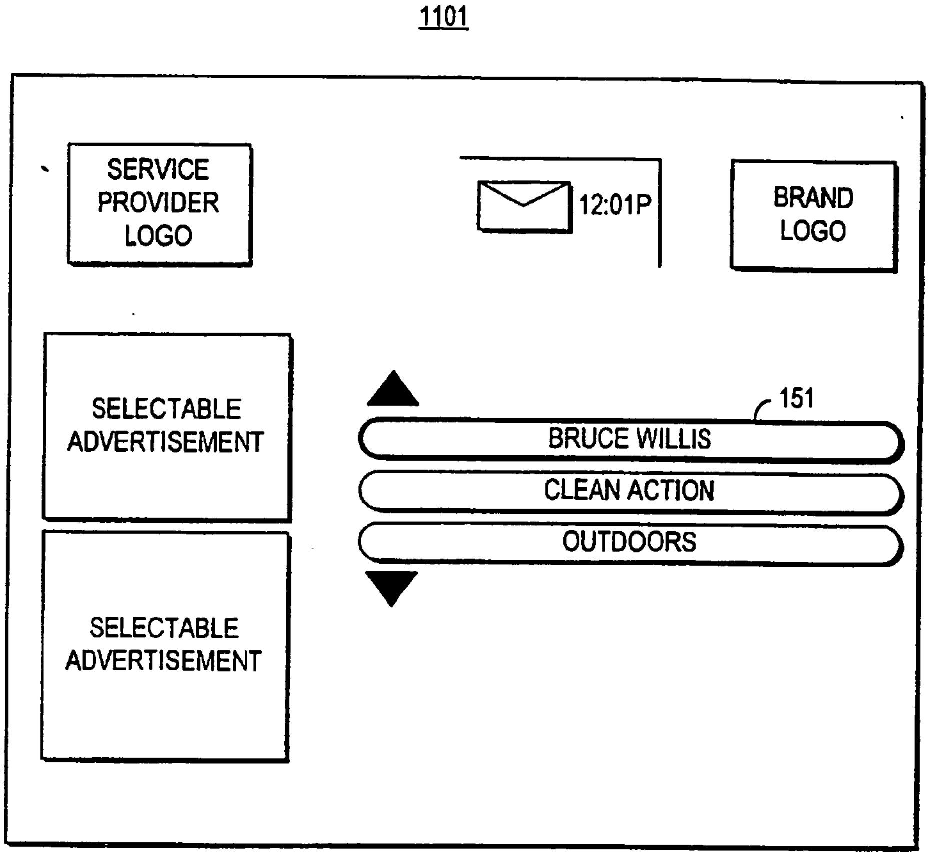

1101
SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
151
BRUCE WILLIS
CLEAN ACTION
OUTDOORS

FIG. 10

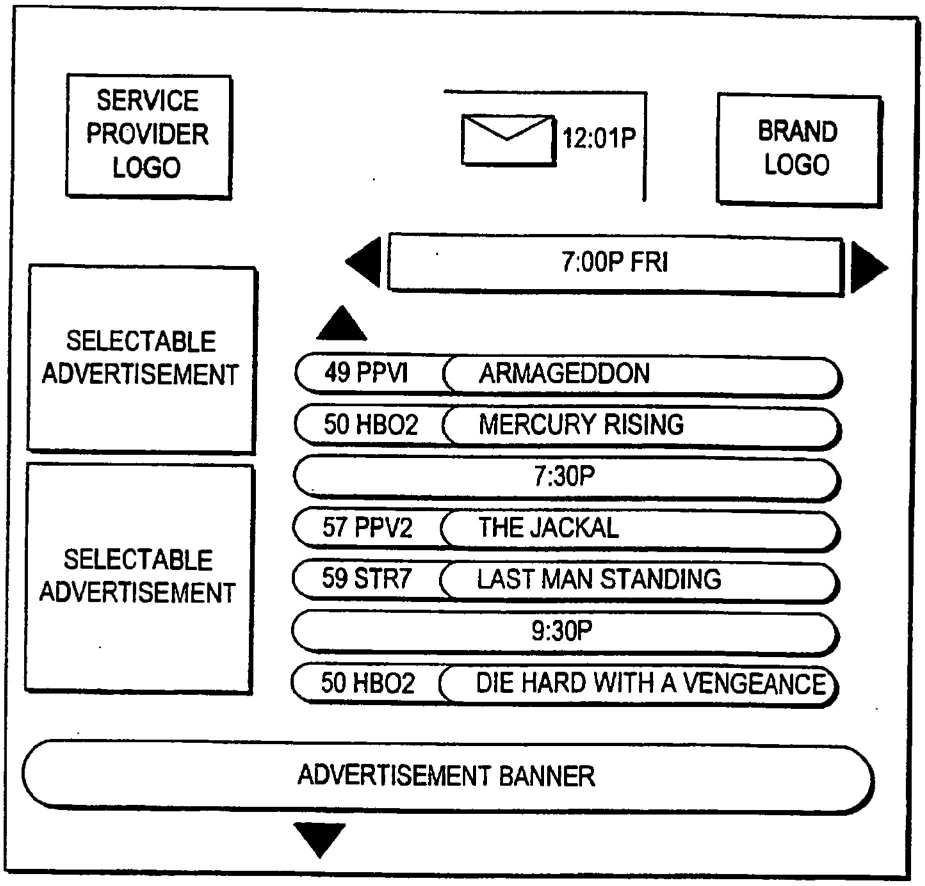

SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
7:00P FRI
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
49 PPVI
ARMAGEDDON
50 HBO2
MERCURY RISING
7:30P
57 PPV2
THE JACKAL
59 STR7
LAST MAN STANDING
9:30P
50 HBO2
DIE HARD WITH A VENGEANCE
ADVERTISEMENT BANNER

FIG. 11

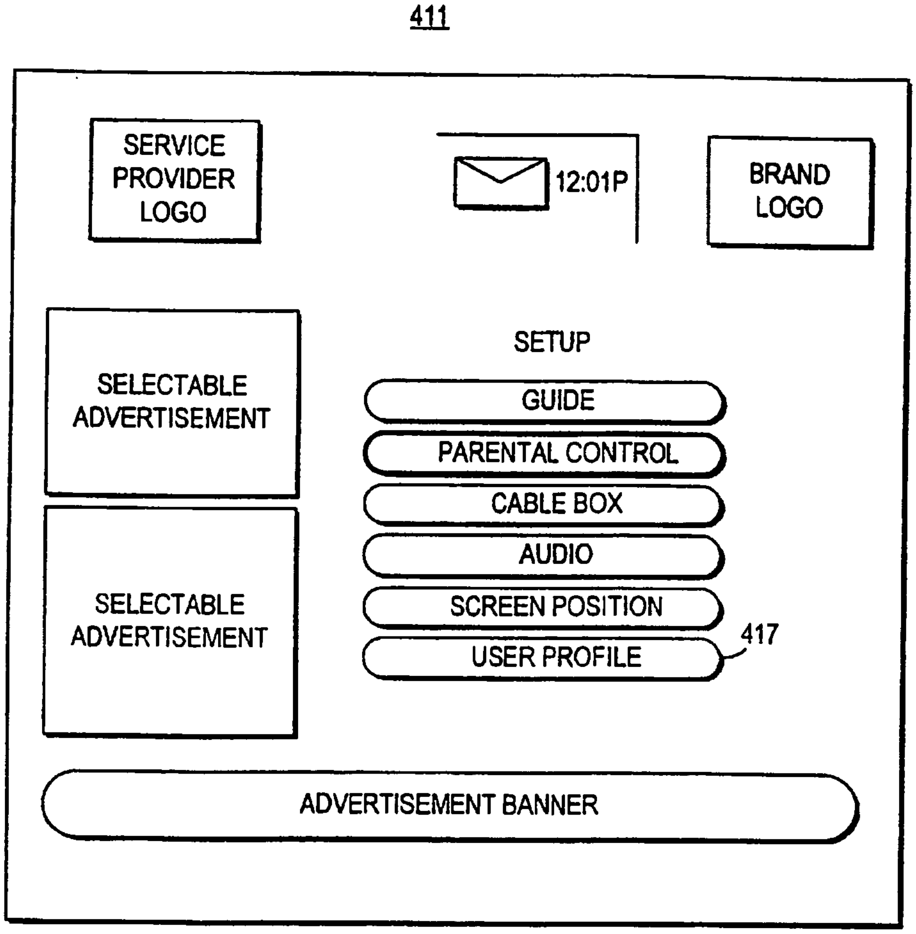

411
SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
SELECTABLE ADVERTISEMENT
SETUP
GUIDE
PARENTAL CONTROL
CABLE BOX
AUDIO
SCREEN POSITION
USER PROFILE
417
SELECTABLE ADVERTISEMENT
ADVERTISEMENT BANNER

FIG. 12

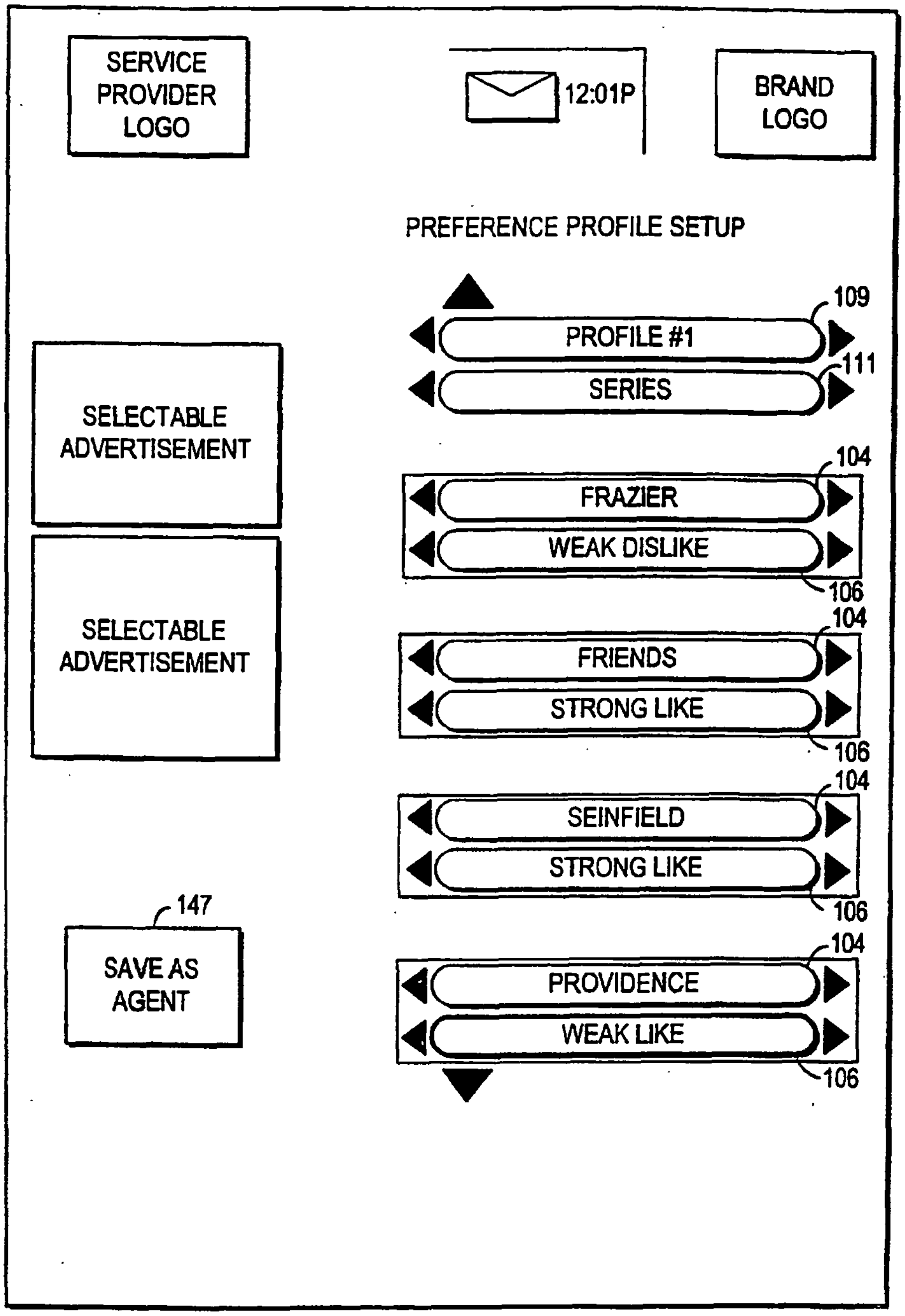

SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
PREFERENCE PROFILE SETUP
109
PROFILE #1
111
SERIES
SELECTABLE ADVERTISEMENT
104
FRAZIER
WEAK DISLIKE
106
SELECTABLE ADVERTISEMENT
104
FRIENDS
STRONG LIKE
106
104
SEINFIELD
STRONG LIKE
106
147
SAVE AS AGENT
104
PROVIDENCE
WEAK LIKE
106

FIG. 13a

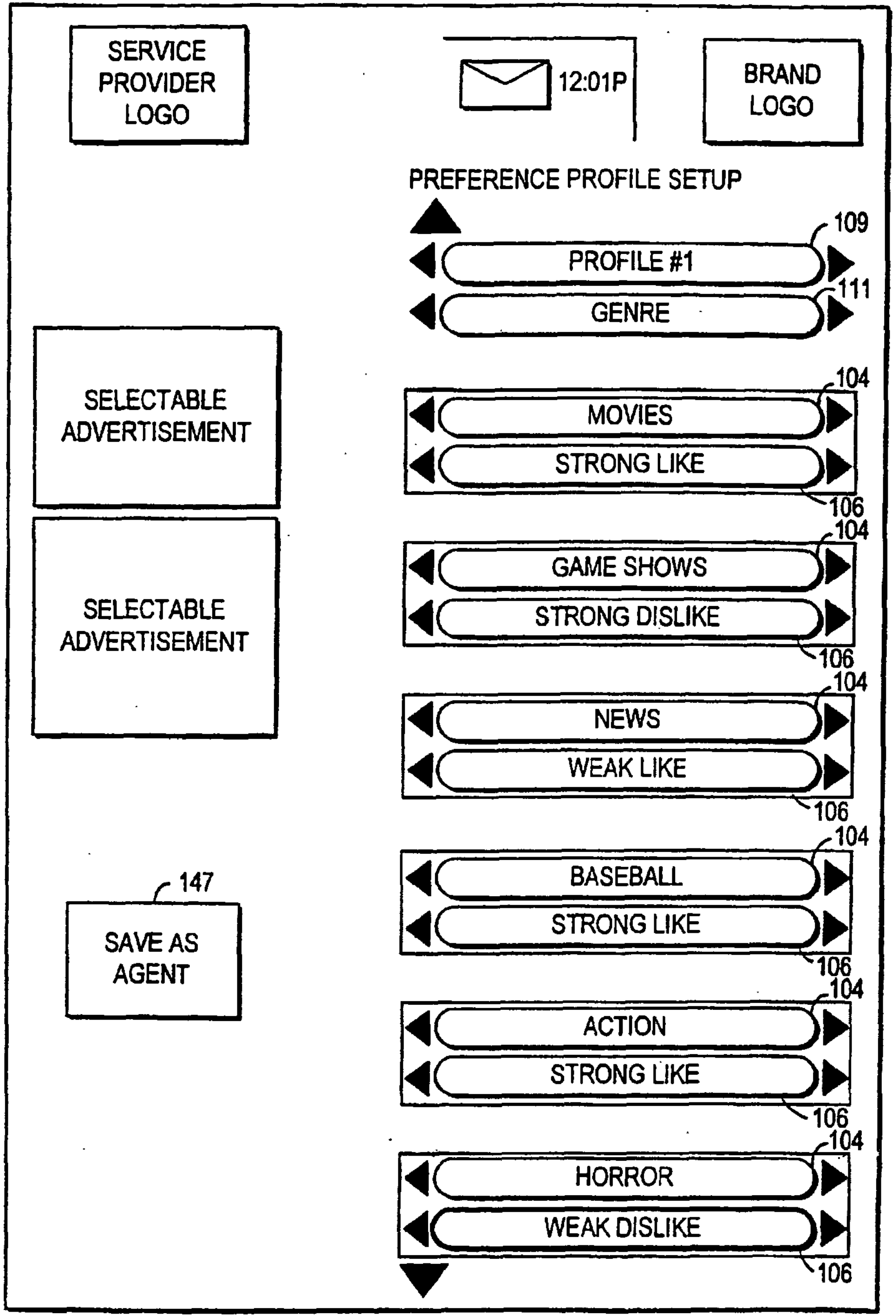

SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
PREFERENCE PROFILE SETUP
109
PROFILE #1
111
GENRE
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
104
MOVIES
STRONG LIKE
106
104
GAME SHOWS
STRONG DISLIKE
106
104
NEWS
WEAK LIKE
106
147
SAVE AS AGENT
104
BASEBALL
STRONG LIKE
106
104
ACTION
STRONG LIKE
106
104
HORROR
WEAK DISLIKE
106

FIG. 13b

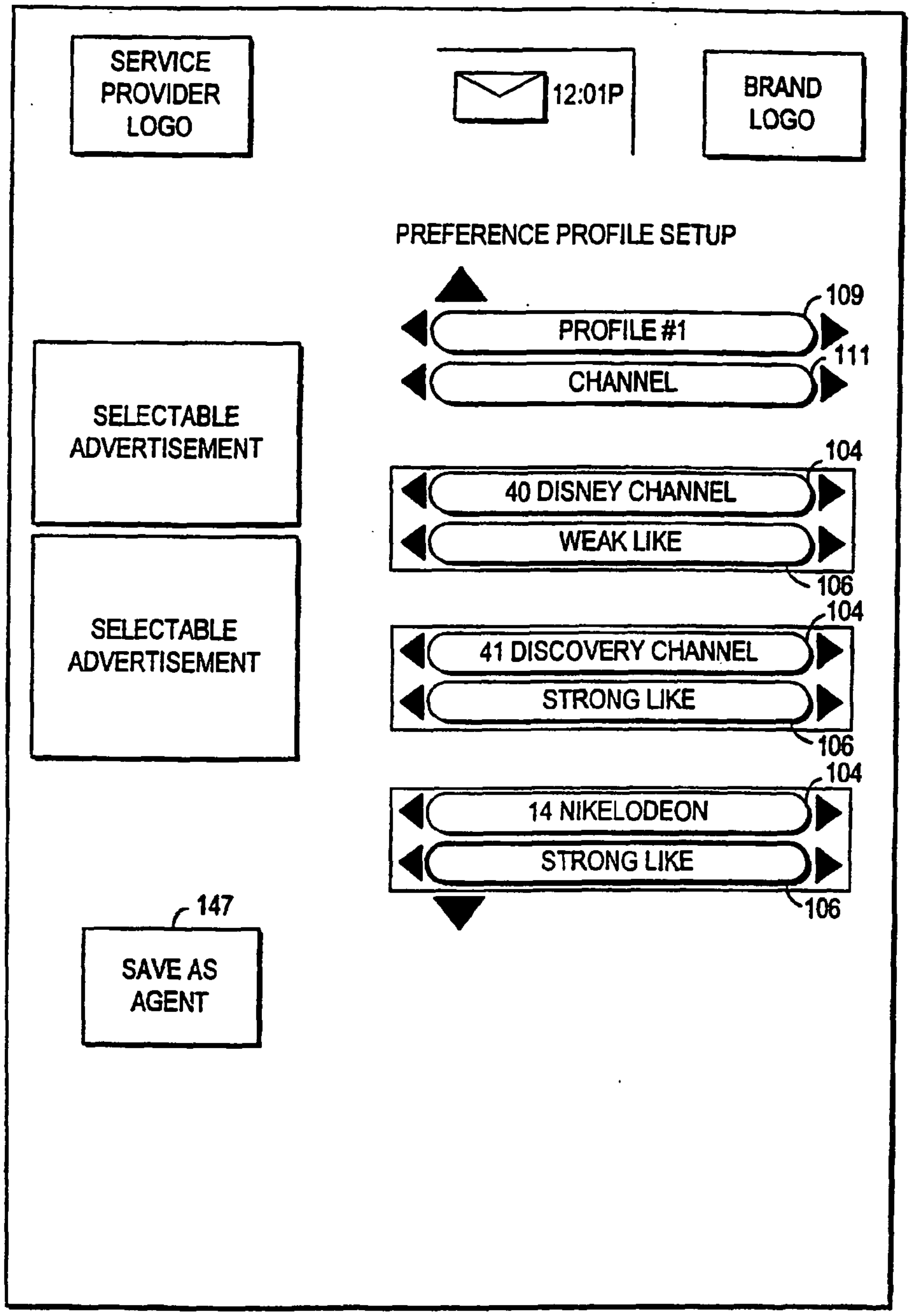

SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
PREFERENCE PROFILE SETUP
109
PROFILE #1
111
CHANNEL
SELECTABLE ADVERTISEMENT
104
40 DISNEY CHANNEL
WEAK LIKE
106
SELECTABLE ADVERTISEMENT
104
41 DISCOVERY CHANNEL
STRONG LIKE
106
104
14 NIKELODEON
STRONG LIKE
106
147
SAVE AS AGENT

FIG. 13c

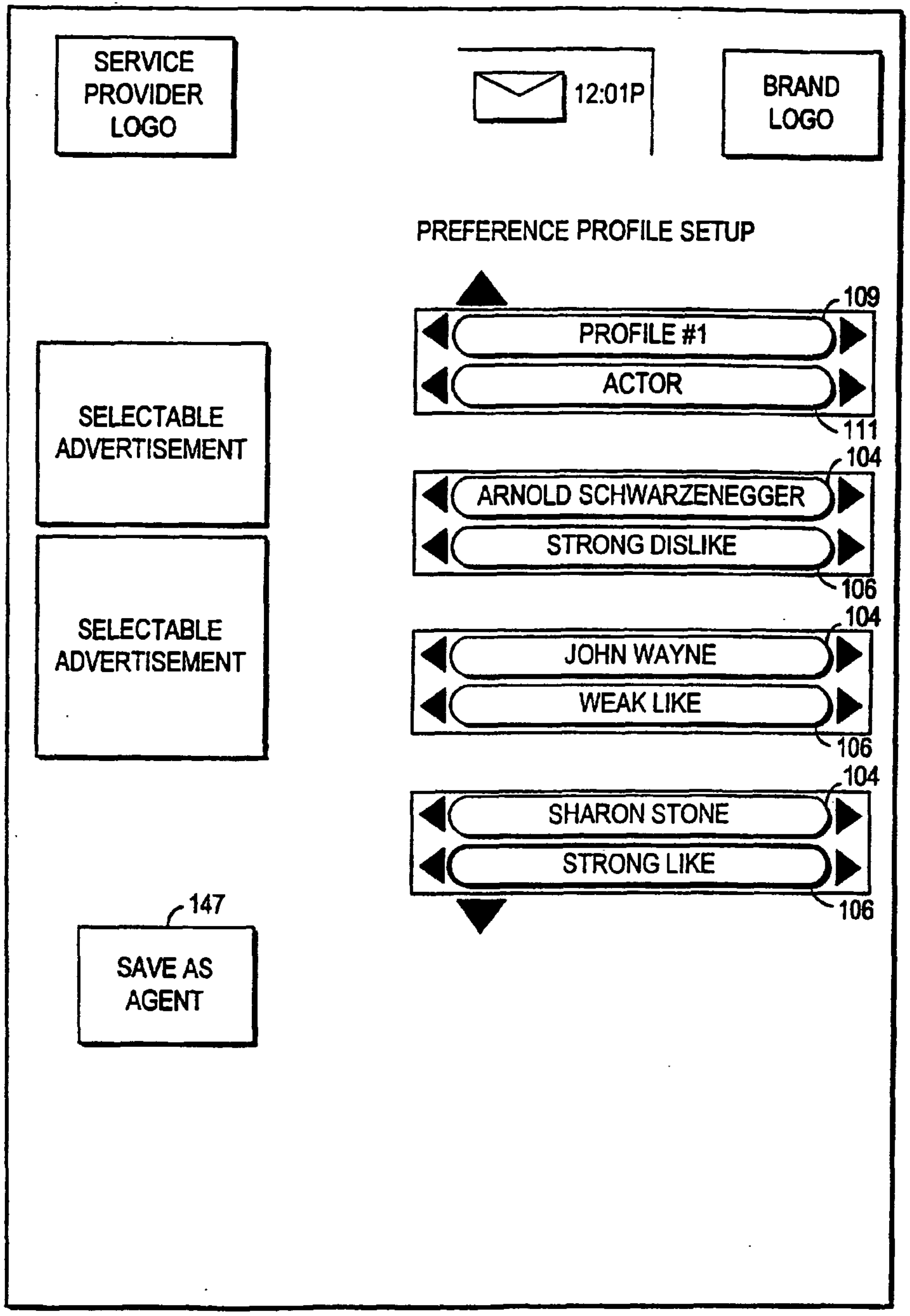

SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
PREFERENCE PROFILE SETUP
109
PROFILE #1
ACTOR
111
SELECTABLE ADVERTISEMENT
104
ARNOLD SCHWARZENEGGER
STRONG DISLIKE
106
SELECTABLE ADVERTISEMENT
104
JOHN WAYNE
WEAK LIKE
106
104
SHARON STONE
STRONG LIKE
106
147
SAVE AS AGENT

FIG. 13d

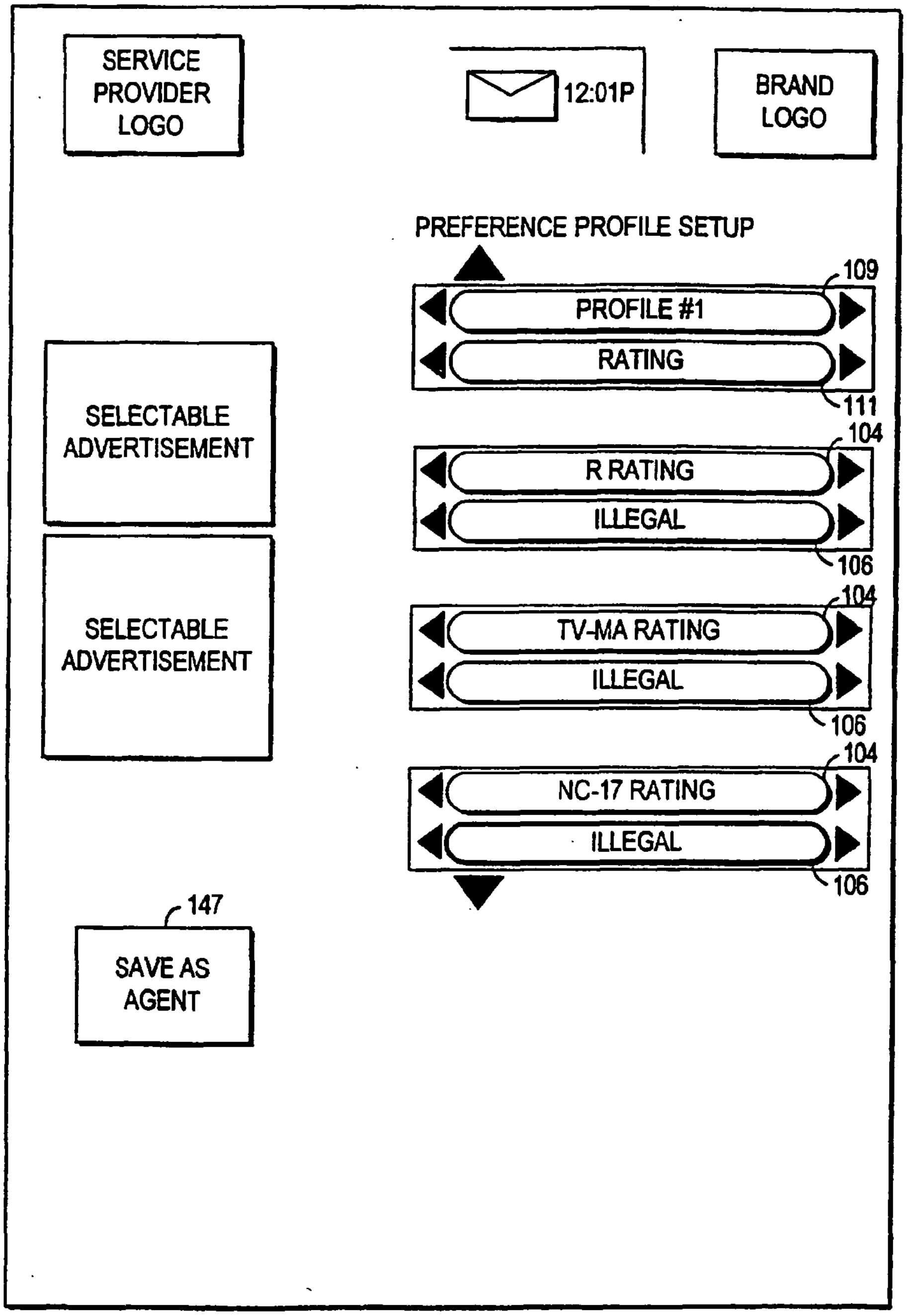

SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
PREFERENCE PROFILE SETUP
109
PROFILE #1
RATING
111
SELECTABLE ADVERTISEMENT
104
R RATING
ILLEGAL
106
SELECTABLE ADVERTISEMENT
104
TV-MA RATING
ILLEGAL
106
104
NC-17 RATING
ILLEGAL
106
147
SAVE AS AGENT

FIG. 13e

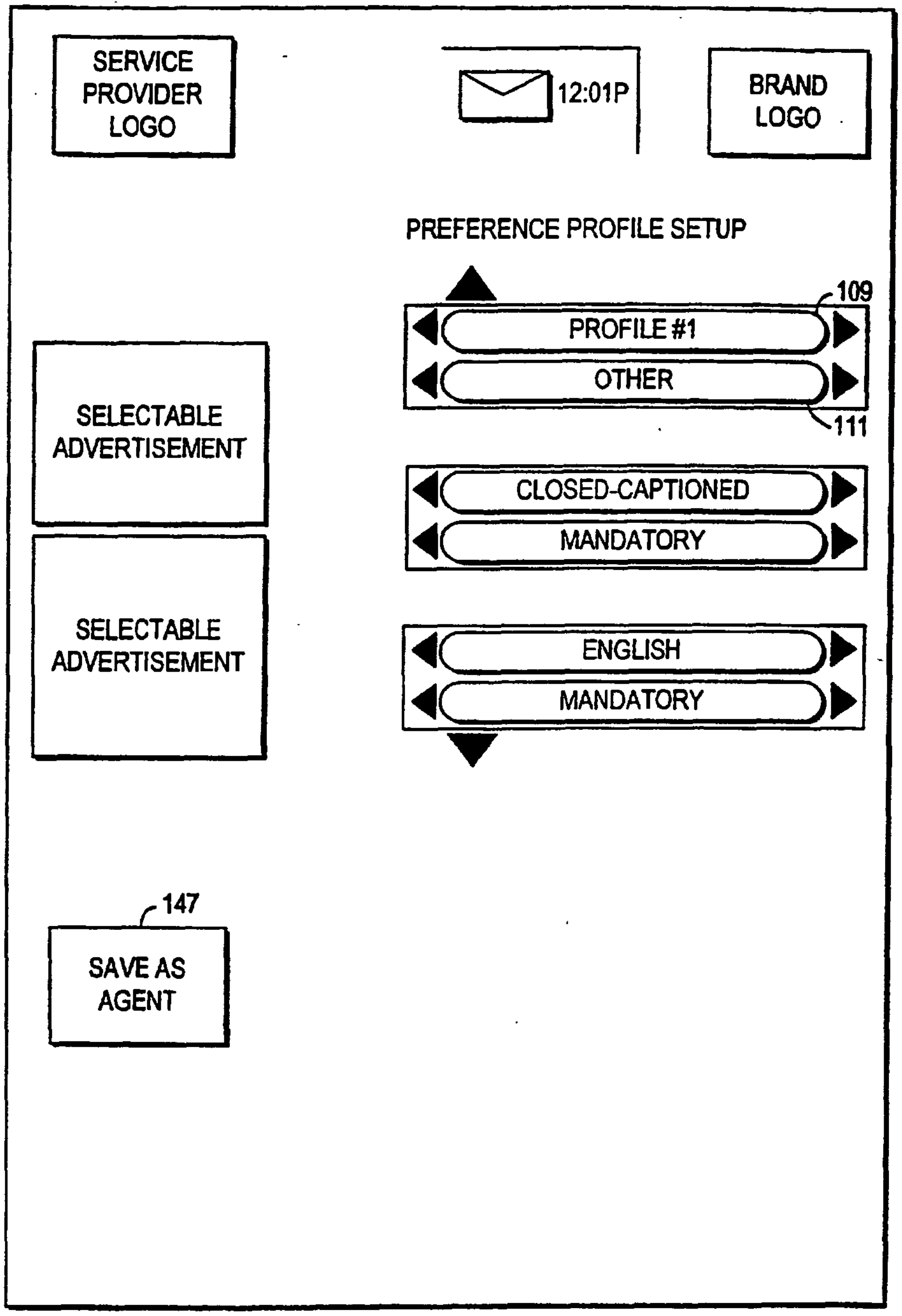
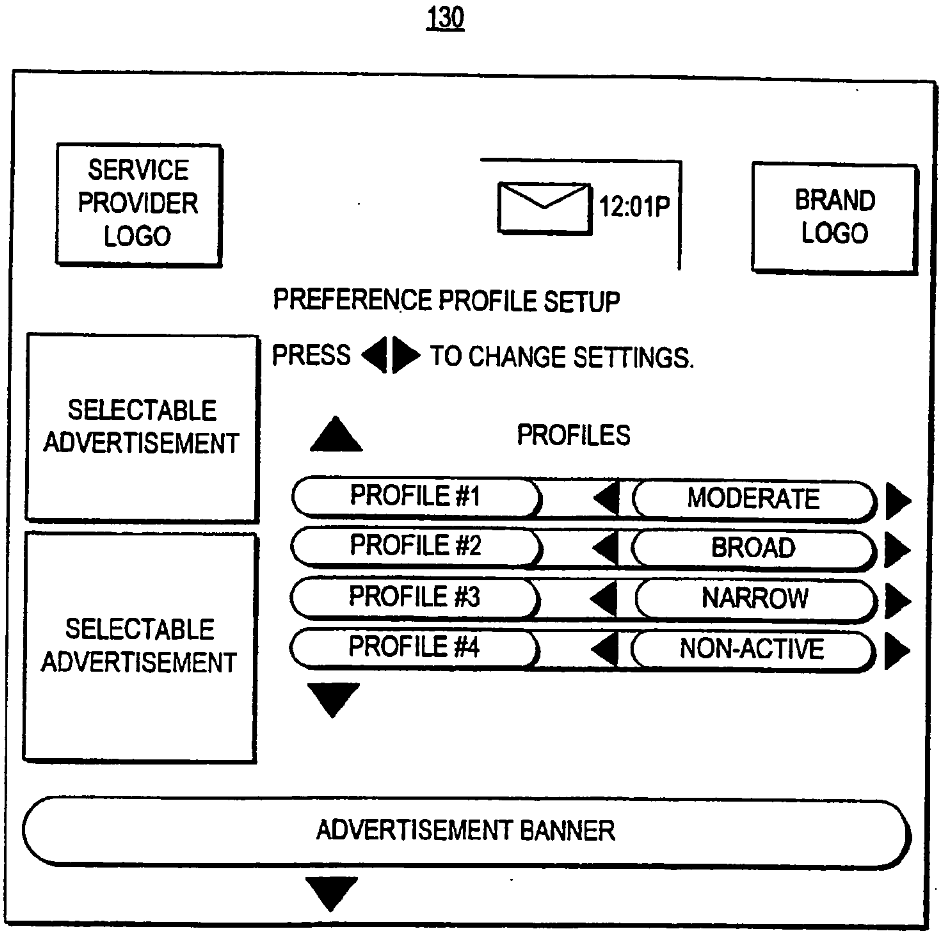

SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
PREFERENCE PROFILE SETUP
109
PROFILE #1
OTHER
111
SELECTABLE ADVERTISEMENT
CLOSED-CAPTIONED
MANDATORY
SELECTABLE ADVERTISEMENT
ENGLISH
MANDATORY
147
SAVE AS AGENT

FIG. 13f



PREFERENCE PROFILE SETUP

PRESS ◀▶ TO CHANGE SETTINGS.

▲ PROFILES

PROFILE #1 ◀ MODERATE ▶

PROFILE #2 ◀ BROAD ▶

PROFILE #3 ◀ NARROW ▶

PROFILE #4 ◀ NON-ACTIVE ▶

▼

FIG. 14

NARROW SCOPE	MODERATE SCOPE	WIDE SCOPE	TITLE	GENRE	CC	RATING	MANDATORY+ NOT ILLEGAL	HIGHEST LEVEL
Y	Y	Y	SEINFELD	COMEDY	Y	TV-PG	Y	SL
N	N	Y	THE SHINING	HORROR	Y	PG-13	Y	WD
N	N	N	DANTE'S PEAK	COMEDY	Y	R	N	SL
N	N	N	NIGHT AT THE OPERA	COMEDY	N	G	N	SL
N	Y	Y	ER	DRAMA	Y	TV-PG	Y	NEUTRAL
N	N	Y	TERMINATOR	ACTION HORROR	Y	PG-13	Y	SD
N	Y	Y	MY STEPMOTHER IS AN ALIEN	COMEDY HORROR	Y	PG-13	Y	SL+WD

FIG. 15

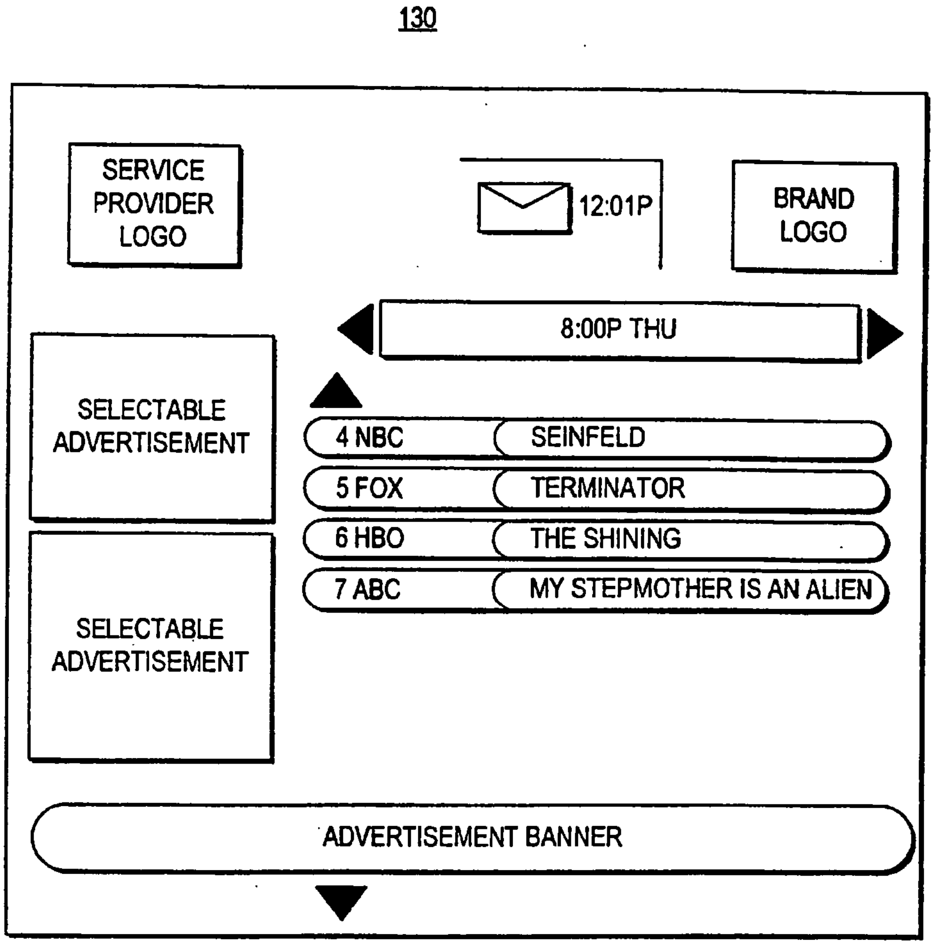

130
SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
8:00P THU
SELECTABLE ADVERTISEMENT
4 NBC
SEINFELD
5 FOX
TERMINATOR
6 HBO
THE SHINING
7 ABC
MY STEPMOTHER IS AN ALIEN
SELECTABLE ADVERTISEMENT
ADVERTISEMENT BANNER

FIG. 16a

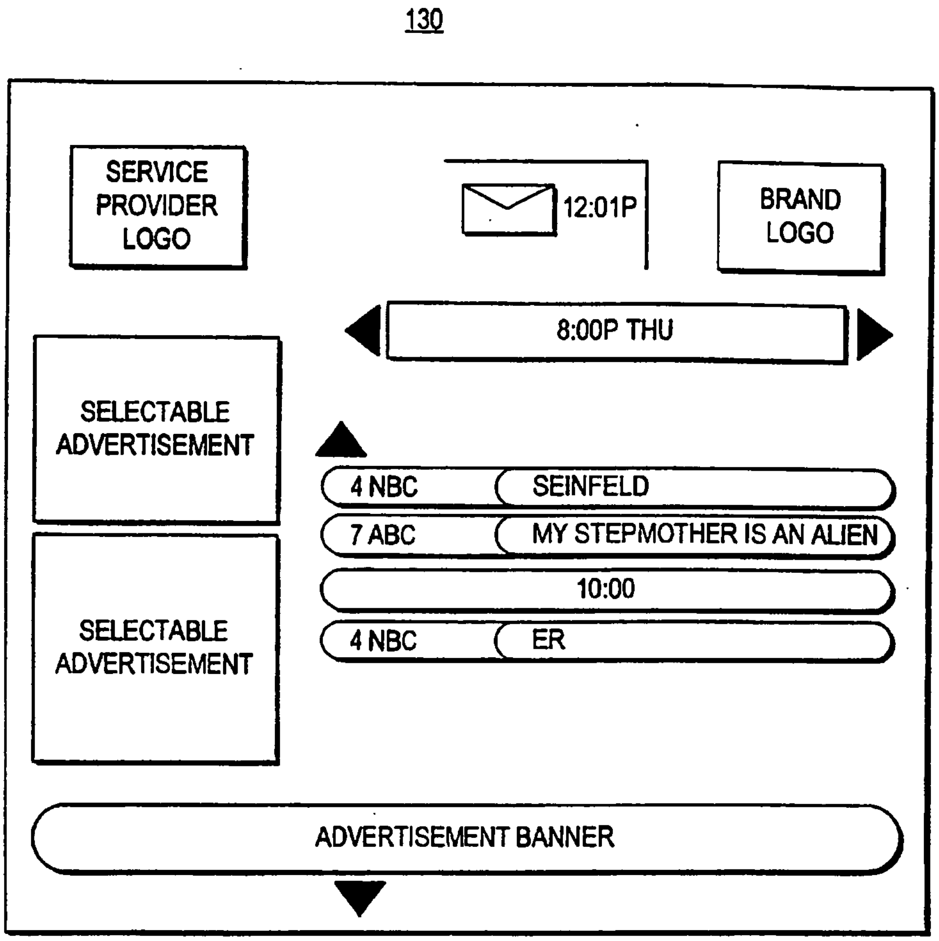

130
SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
8:00P THU
SELECTABLE ADVERTISEMENT
4 NBC
SEINFELD
7 ABC
MY STEPMOTHER IS AN ALIEN
10:00
SELECTABLE ADVERTISEMENT
4 NBC
ER
ADVERTISEMENT BANNER

FIG. 16b

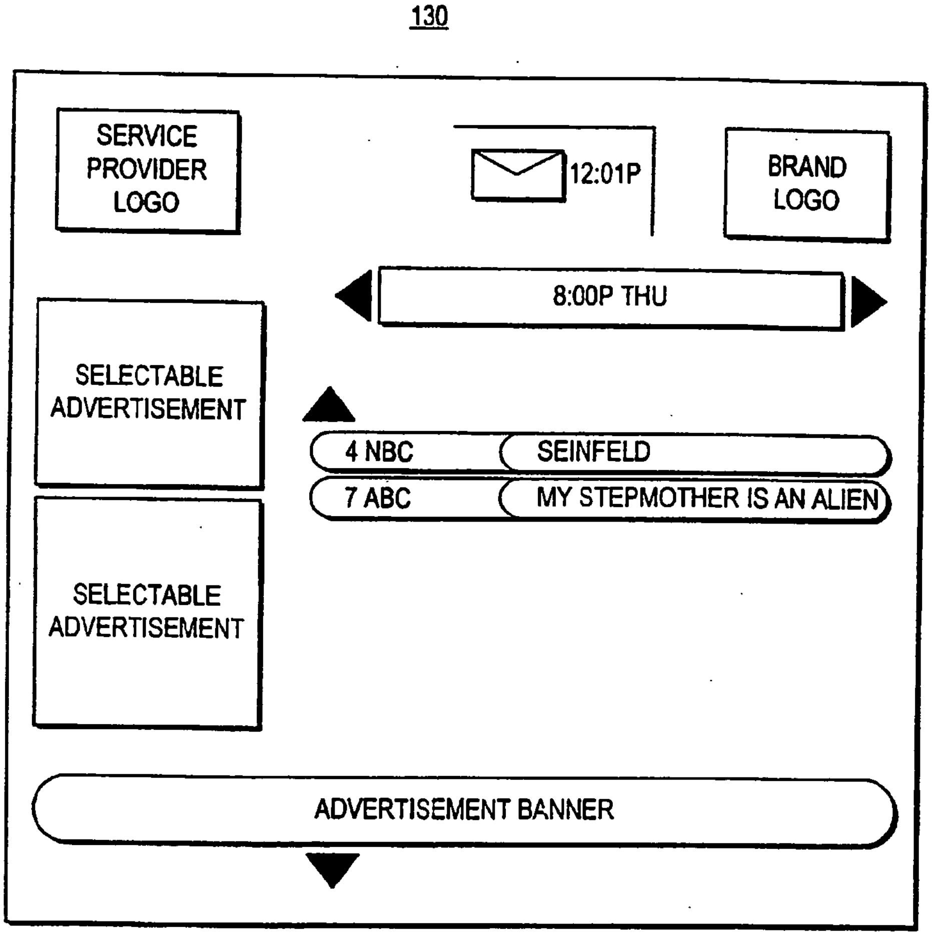

130
SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
8:00P THU
SELECTABLE ADVERTISEMENT
4 NBC
SEINFELD
7 ABC
MY STEPMOTHER IS AN ALIEN
SELECTABLE ADVERTISEMENT
ADVERTISEMENT BANNER

FIG. 16c

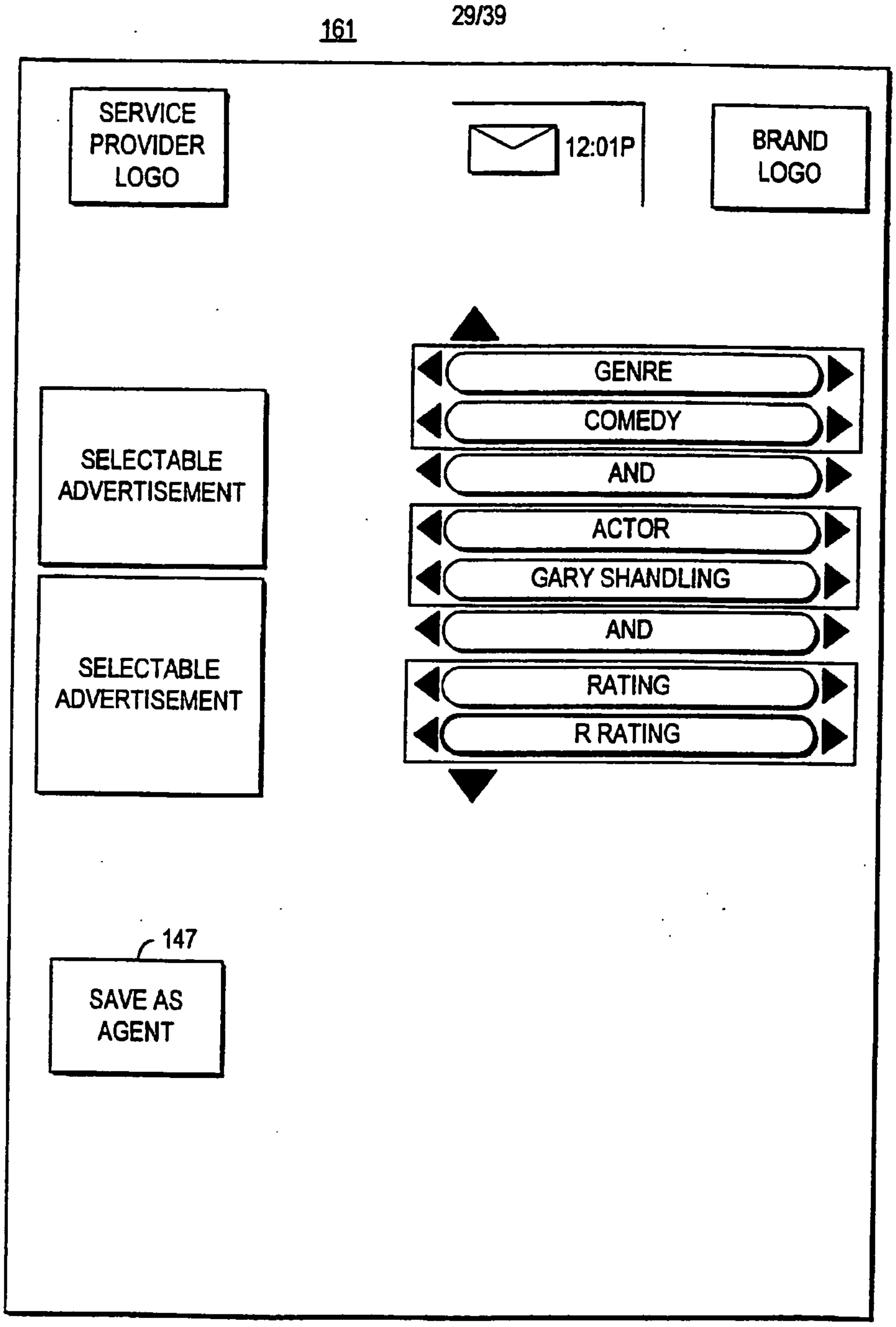
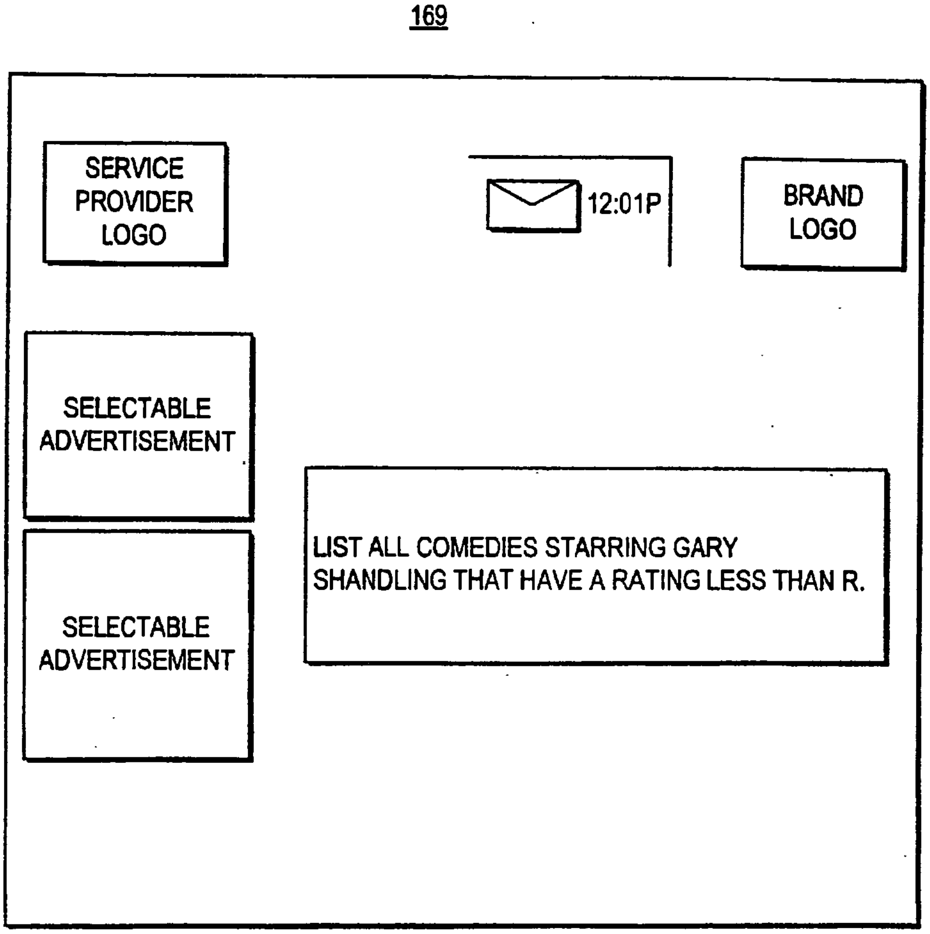

161
SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
GENRE
COMEDY
AND
ACTOR
GARY SHANDLING
AND
RATING
R RATING
147
SAVE AS AGENT

FIG. 17a



LIST ALL COMEDIES STARRING GARY SHANDLING THAT HAVE A RATING LESS THAN R.

FIG. 17b

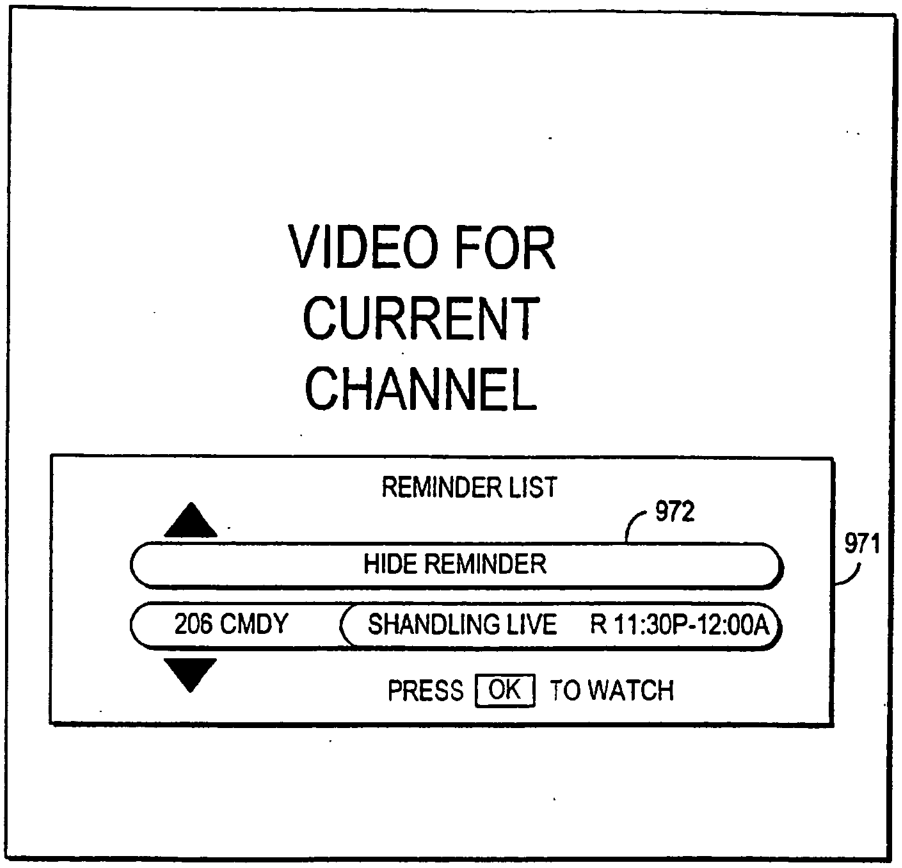

VIDEO FOR
CURRENT
CHANNEL
REMINDER LIST
972
971
HIDE REMINDER
206 CMDY
SHANDLING LIVE
R 11:30P-12:00A
PRESS OK TO WATCH

FIG. 18

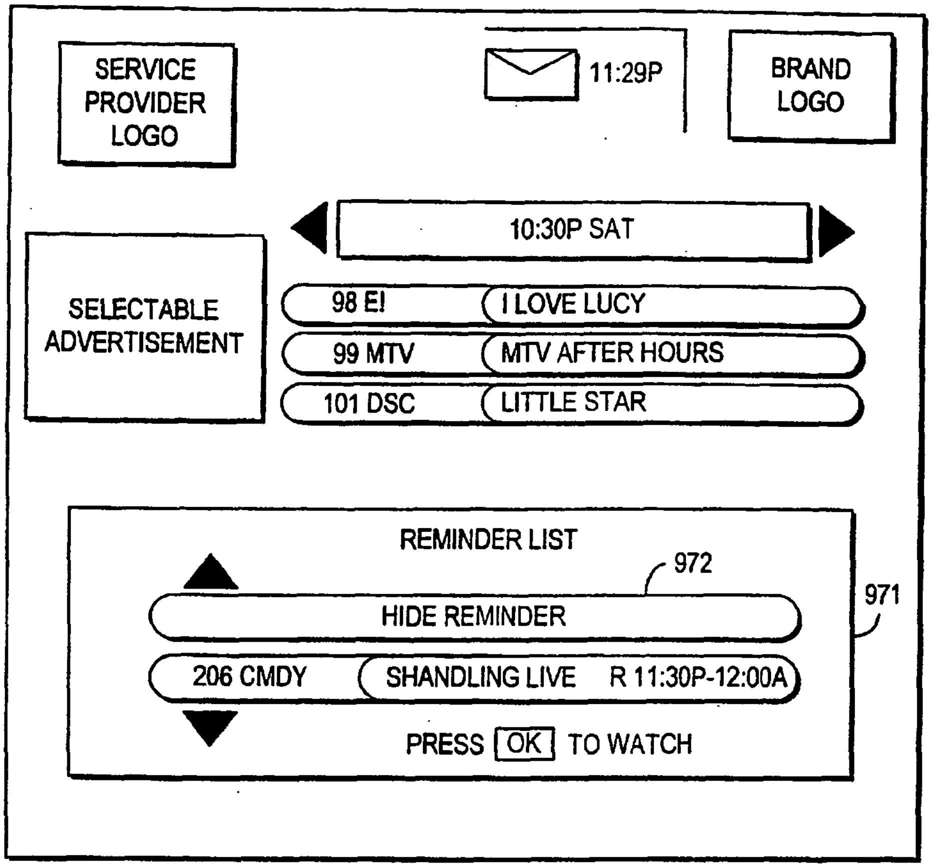

SERVICE PROVIDER LOGO
11:29P
BRAND LOGO
10:30P SAT
SELECTABLE ADVERTISEMENT
98 E!
I LOVE LUCY
99 MTV
MTV AFTER HOURS
101 DSC
LITTLE STAR
REMINDER LIST
972
971
HIDE REMINDER
206 CMDY
SHANDLING LIVE R 11:30P-12:00A
PRESS OK TO WATCH

FIG. 19

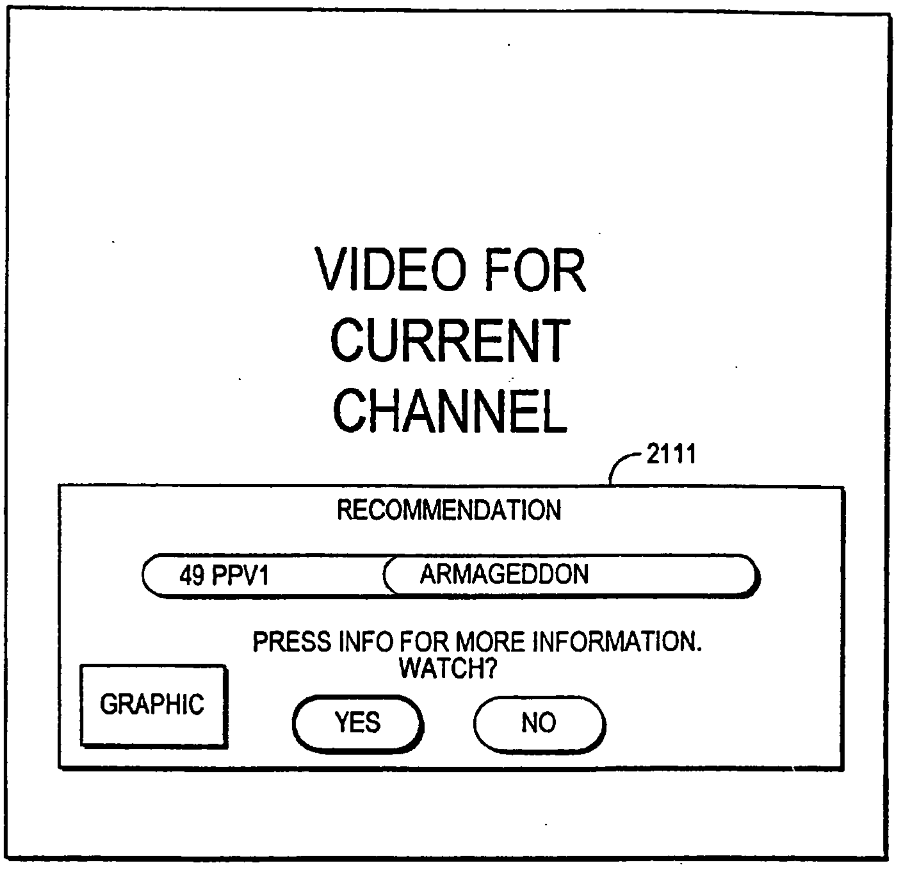

VIDEO FOR
CURRENT
CHANNEL
2111
RECOMMENDATION
49 PPV1
ARMAGEDDON
PRESS INFO FOR MORE INFORMATION.
WATCH?
GRAPHIC
YES
NO

FIG. 20a

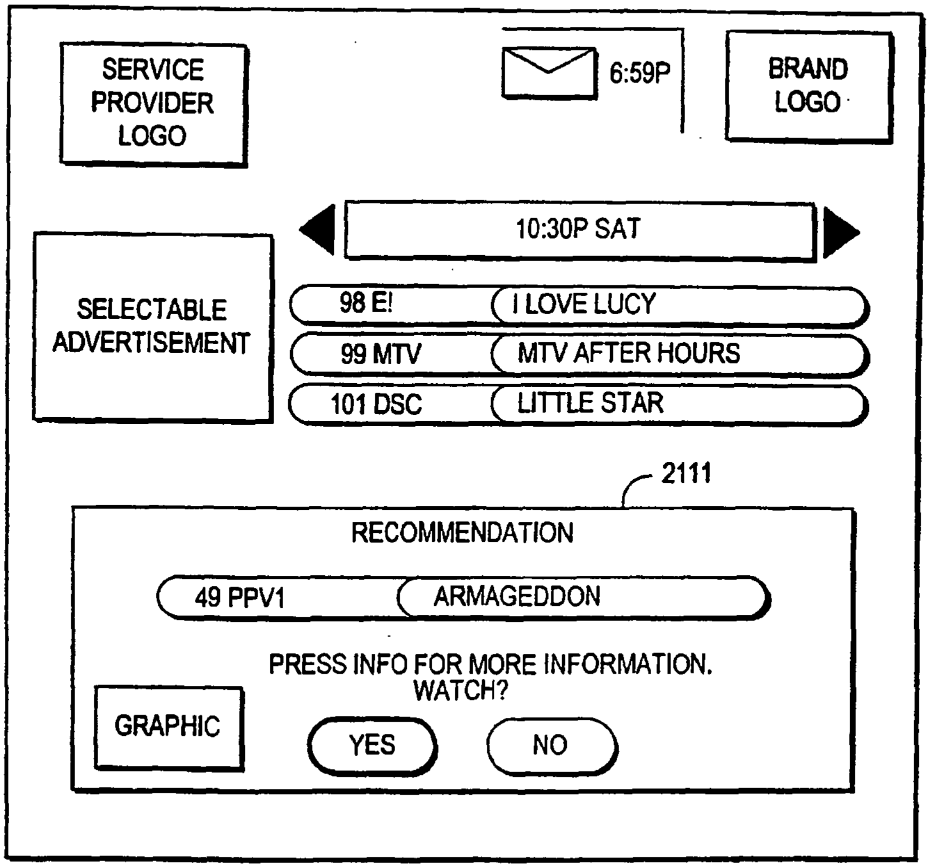
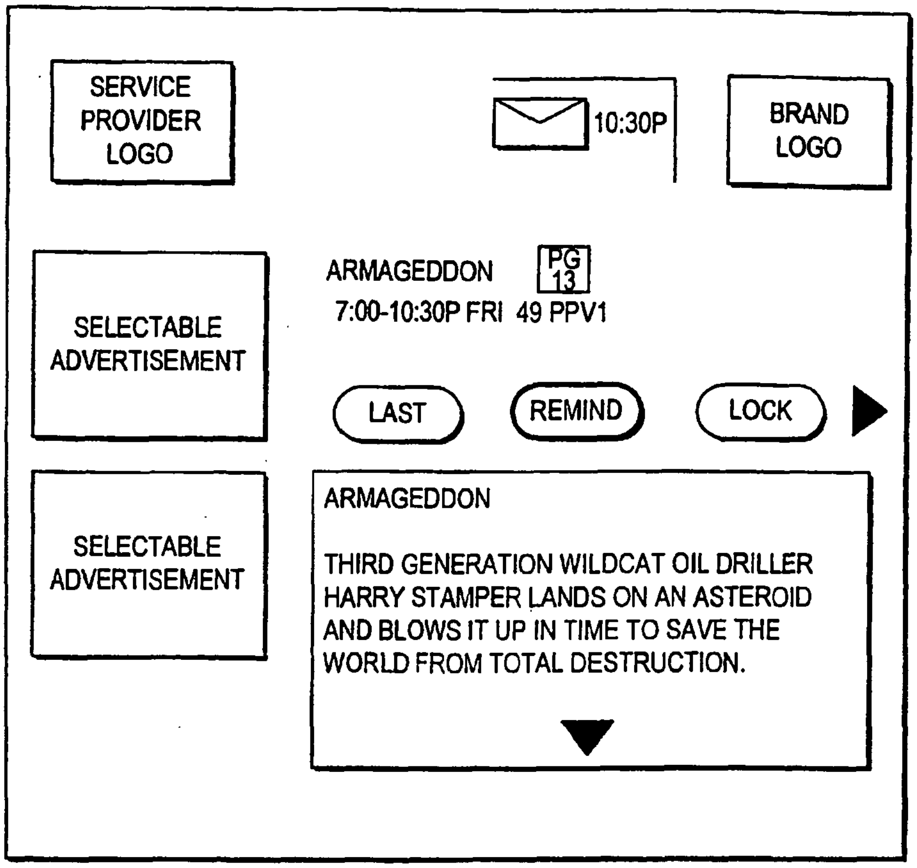

SERVICE PROVIDER LOGO
6:59P
BRAND LOGO
10:30P SAT
SELECTABLE ADVERTISEMENT
98 E!
I LOVE LUCY
99 MTV
MTV AFTER HOURS
101 DSC
LITTLE STAR
2111
RECOMMENDATION
49 PPV1
ARMAGEDDON
PRESS INFO FOR MORE INFORMATION.
WATCH?
GRAPHIC
YES
NO

FIG. 20b



THIRD GENERATION WILDCAT OIL DRILLER HARRY STAMPER LANDS ON AN ASTEROID AND BLOWS IT UP IN TIME TO SAVE THE WORLD FROM TOTAL DESTRUCTION.

FIG. 20c

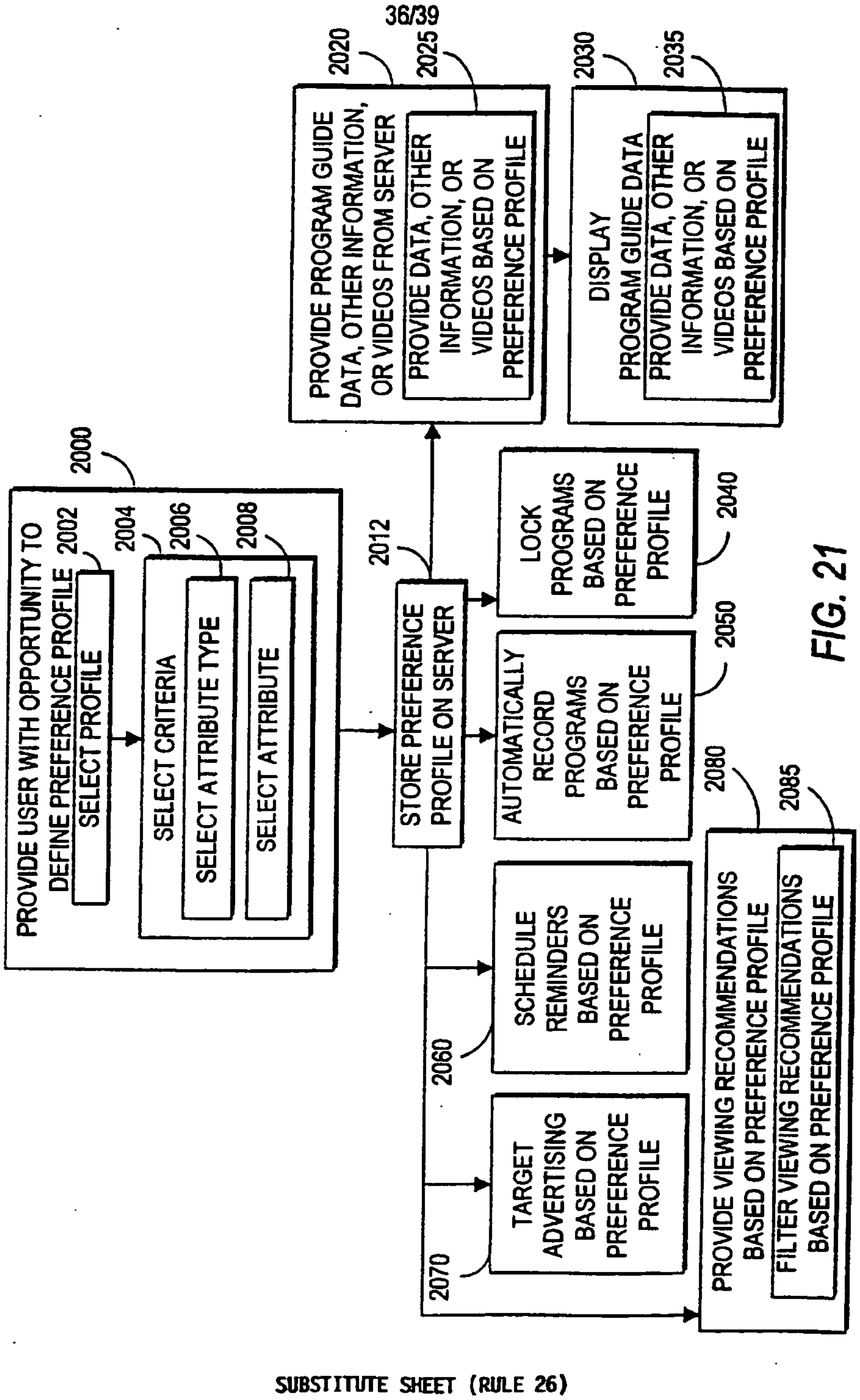

2000
PROVIDE USER WITH OPPORTUNITY TO DEFINE PREFERENCE PROFILE
2002
SELECT PROFILE
2004
SELECT CRITERIA
2006
SELECT ATTRIBUTE TYPE
2008
SELECT ATTRIBUTE
2012
STORE PREFERENCE PROFILE ON SERVER
2020
PROVIDE PROGRAM GUIDE DATA, OTHER INFORMATION, OR VIDEOS FROM SERVER
2025
PROVIDE DATA, OTHER INFORMATION, OR VIDEOS BASED ON PREFERENCE PROFILE
2030
DISPLAY PROGRAM GUIDE DATA
2035
PROVIDE DATA, OTHER INFORMATION, OR VIDEOS BASED ON PREFERENCE PROFILE
LOCK PROGRAMS BASED ON PREFERENCE PROFILE
2040
AUTOMATICALLY RECORD PROGRAMS BASED ON PREFERENCE PROFILE
2050
SCHEDULE REMINDERS BASED ON PREFERENCE PROFILE
2060
TARGET ADVERTISING BASED ON PREFERENCE PROFILE
2070
PROVIDE VIEWING RECOMMENDATIONS BASED ON PREFERENCE PROFILE
2080
FILTER VIEWING RECOMMENDATIONS BASED ON PREFERENCE PROFILE
2085

FIG. 21

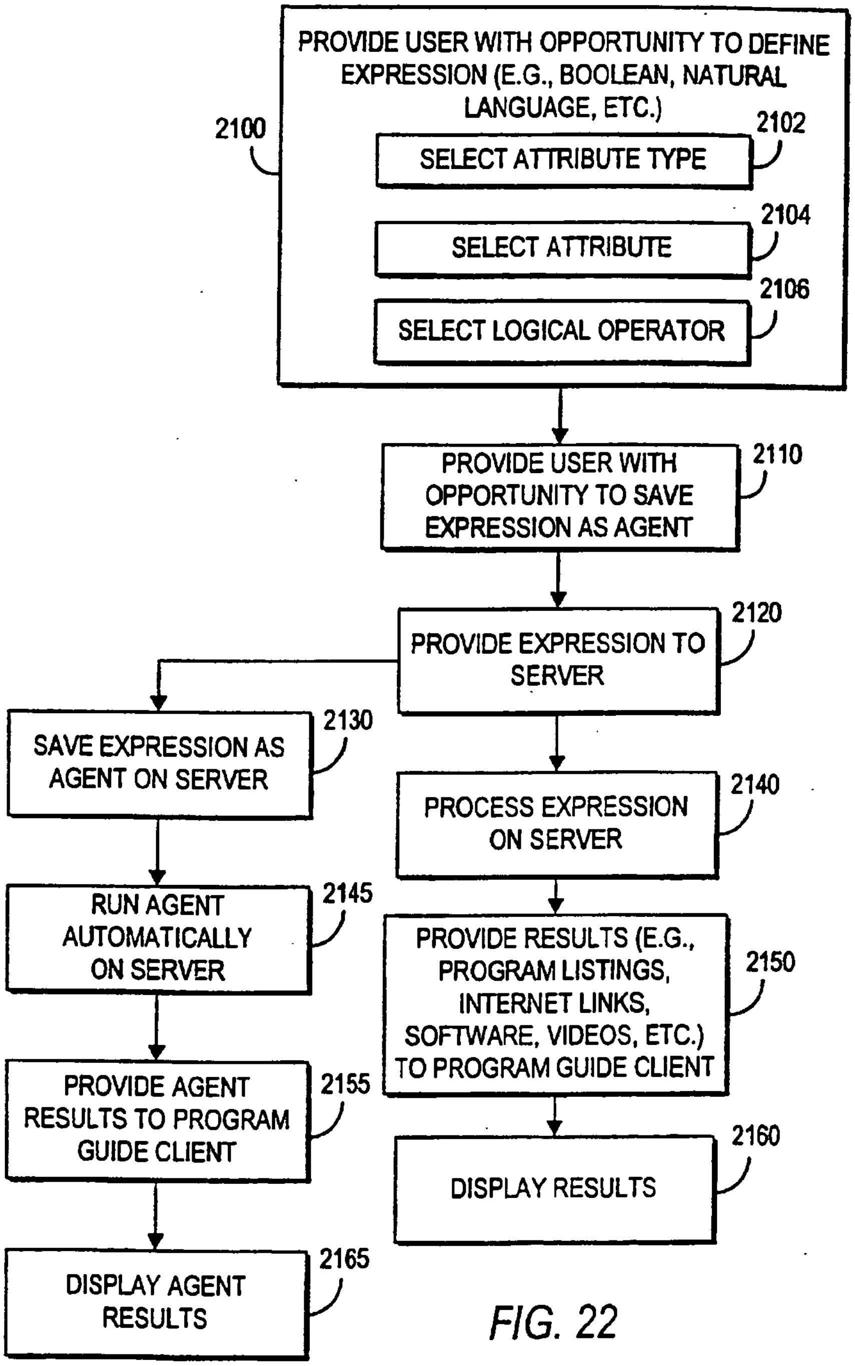

PROVIDE USER WITH OPPORTUNITY TO DEFINE EXPRESSION (E.G., BOOLEAN, NATURAL LANGUAGE, ETC.)
2100
SELECT ATTRIBUTE TYPE
2102
SELECT ATTRIBUTE
2104
SELECT LOGICAL OPERATOR
2106
PROVIDE USER WITH OPPORTUNITY TO SAVE EXPRESSION AS AGENT
2110
PROVIDE EXPRESSION TO SERVER
2120
SAVE EXPRESSION AS AGENT ON SERVER
2130
PROCESS EXPRESSION ON SERVER
2140
RUN AGENT AUTOMATICALLY ON SERVER
2145
PROVIDE RESULTS (E.G., PROGRAM LISTINGS, INTERNET LINKS, SOFTWARE, VIDEOS, ETC.) TO PROGRAM GUIDE CLIENT
2150
PROVIDE AGENT RESULTS TO PROGRAM GUIDE CLIENT
2155
DISPLAY RESULTS
2160
DISPLAY AGENT RESULTS
2165

FIG. 22

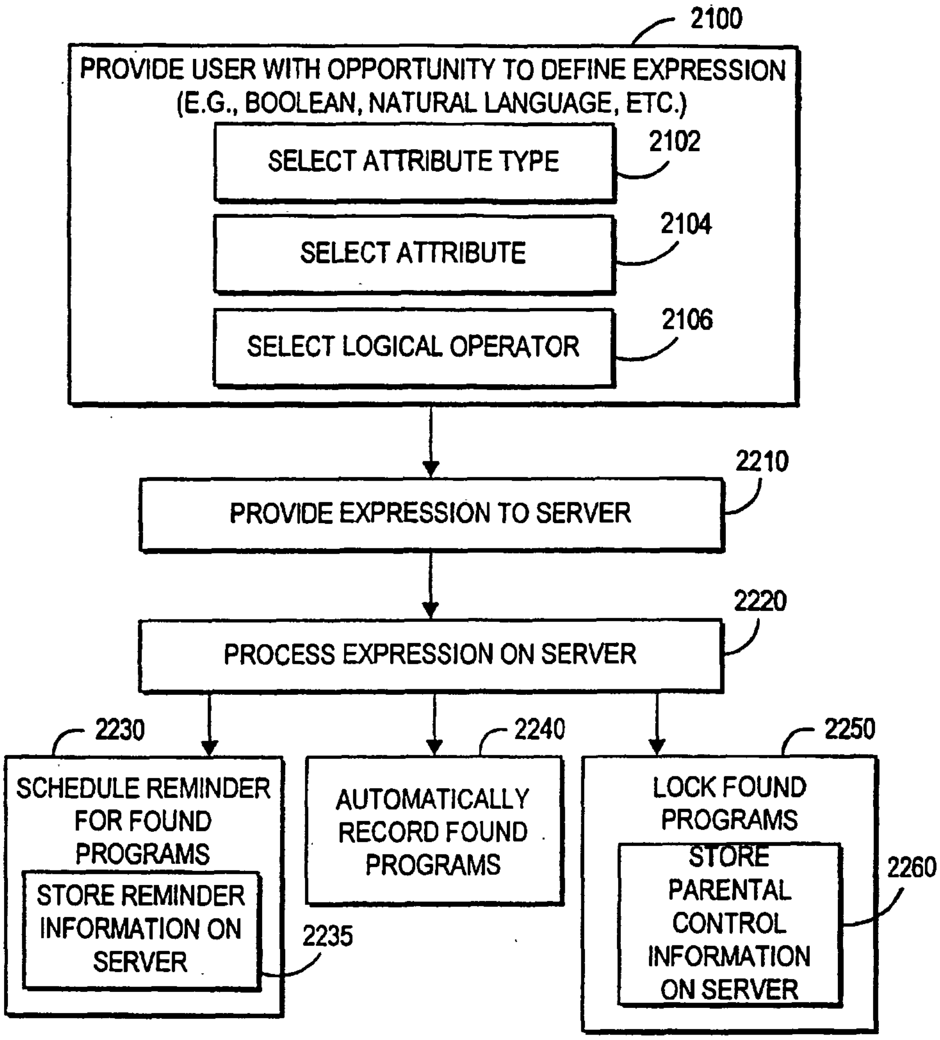

2100
PROVIDE USER WITH OPPORTUNITY TO DEFINE EXPRESSION (E.G., BOOLEAN, NATURAL LANGUAGE, ETC.)
SELECT ATTRIBUTE TYPE
2102
SELECT ATTRIBUTE
2104
SELECT LOGICAL OPERATOR
2106
PROVIDE EXPRESSION TO SERVER
2210
PROCESS EXPRESSION ON SERVER
2220
2230
SCHEDULE REMINDER FOR FOUND PROGRAMS
STORE REMINDER INFORMATION ON SERVER
2235
2240
AUTOMATICALLY RECORD FOUND PROGRAMS
2250
LOCK FOUND PROGRAMS
STORE PARENTAL CONTROL INFORMATION ON SERVER
2260

FIG. 23

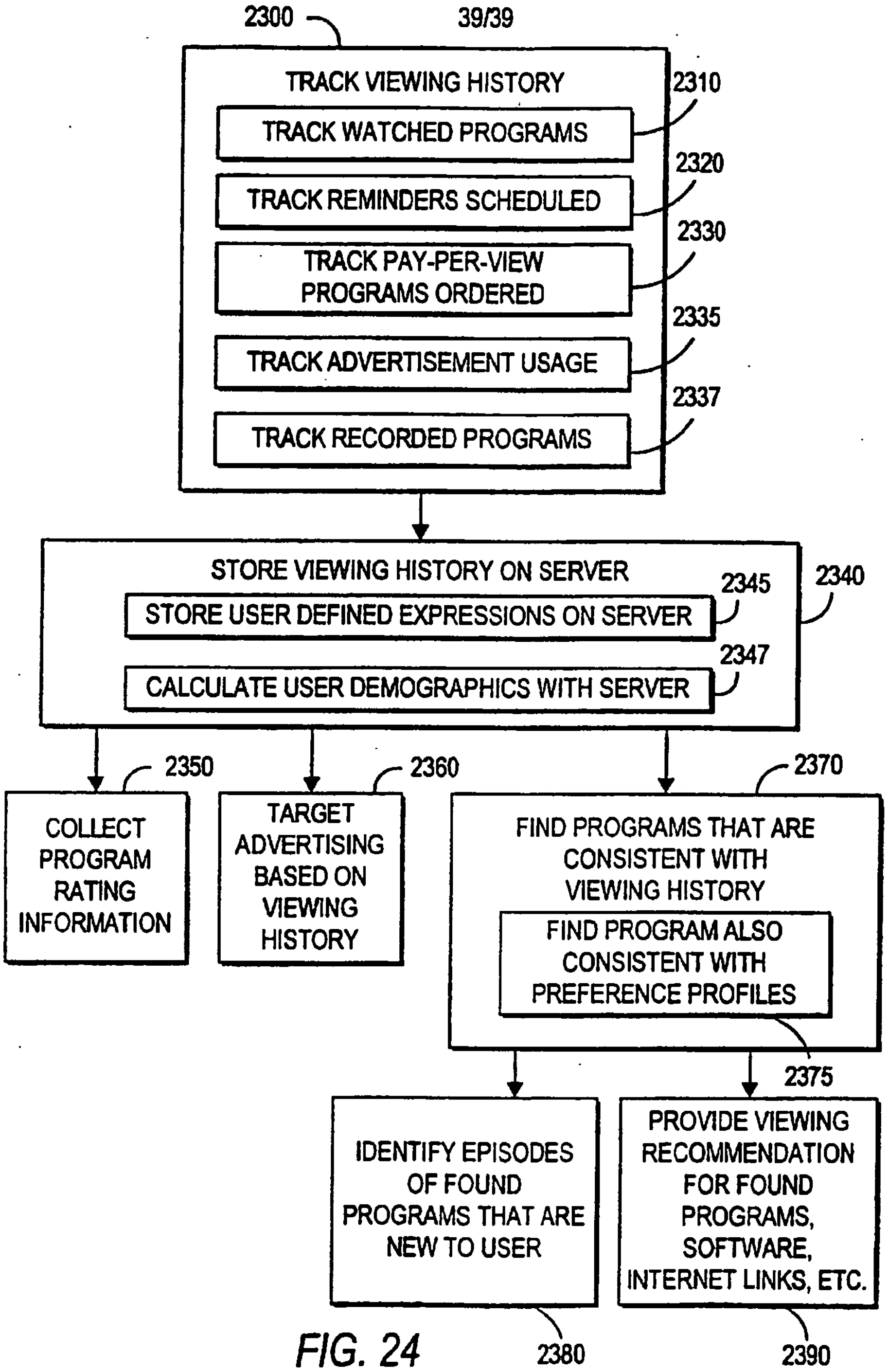

2300
TRACK VIEWING HISTORY
TRACK WATCHED PROGRAMS
2310
TRACK REMINDERS SCHEDULED
2320
TRACK PAY-PER-VIEW PROGRAMS ORDERED
2330
TRACK ADVERTISEMENT USAGE
2335
TRACK RECORDED PROGRAMS
2337
STORE VIEWING HISTORY ON SERVER
2340
STORE USER DEFINED EXPRESSIONS ON SERVER
2345
CALCULATE USER DEMOGRAPHICS WITH SERVER
2347
COLLECT PROGRAM RATING INFORMATION
2350
TARGET ADVERTISING BASED ON VIEWING HISTORY
2360
FIND PROGRAMS THAT ARE CONSISTENT WITH VIEWING HISTORY
2370
FIND PROGRAM ALSO CONSISTENT WITH PREFERENCE PROFILES
2375
IDENTIFY EPISODES OF FOUND PROGRAMS THAT ARE NEW TO USER
2380
PROVIDE VIEWING RECOMMENDATION FOR FOUND PROGRAMS, SOFTWARE, INTERNET LINKS, ETC.
2390

FIG. 24

SUBSTITUTE SHEET (RULE 26)

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 99/19051

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04N7/16

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 94 14284 A (DISCOVERY COMMUNICAT INC) 23 June 1994 (1994-06-23) page 11, line 16 -page 13, line 30 page 15, line 22 -page 18, line 12 page 19, line 21 -page 21, line 10 page 32, line 11 -page 38, line 12 page 45, line 1 -page 46, line 3 page 59, line 11 -page 61, line 14 page 67, line 18 -page 70, line 32 figures 1-14 --- -/--	1-4, 6-11, 14-23, 25-30, 33-42, 44-49, 52-56

☒ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search 18 November 1999	Date of mailing of the international search report 24/11/1999
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Van der Zaal, R

1

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 99/19051

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
X	WO 96 41478 A (TV GUIDE ON SCREEN) 19 December 1996 (1996-12-19) page 12, line 32 -page 15, line 24 page 16, line 18 -page 35, line 19 page 36, line 12 -page 39, line 11 figures 1-58 ---	1-13, 20-32, 39-51
A	WO 98 17064 A (GEMSTAR DEVELOPMENT CORPORATION) 23 April 1998 (1998-04-23) page 5, line 6 -page 7, line 11 -----	5,24,43

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US 99/19051

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9641478 A		PL 323914 A	27-04-1998
WO 9817064 A	23-04-1998	AU 4823197 A EP 0932979 A	11-05-1998 04-08-1999

(19) Europäisch Patentamt
European Pat nt Office
Office européen des br v t

(11) EP 0 803 826 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
29.10.1997 Bulletin 1997/44

(51) Int Cl.6: G06F 17/30, H04N 7/173

(21) Application number: 97302676.8

(22) Date of filing: 18.04.1997

(84) Designated Contracting States:
DE FR GB IT NL SE

(30) Priority: 22.04.1996 US 636118

(71) Applicant: SUN MICROSYSTEMS, INC.
Mountain View, CA 94043 (US)

(72) Inventors:
- Lindblad, Christopher
Stanford, California 94309 (US)
- Cachat, Stephan E.
Mountain View, California 94041 (US)

(74) Representative: W.P. Thompson & Co.
Coopers Building,
Church Street
Liverpool L1 3AB (GB)

(54) Video on demand applet method and apparatus for inclusion of motion video in multimedia documents

(57) The present specification describes a computer process which requests streams of motion video titles and decodes and displays the motion video signals of the stream for display in a computer display device is constructed in the form of an applet 212 of a multimedia document viewer 202 such as a World Wide Web browser. Accordingly, a designer of multimedia documents such as HTML pages can easily incorporate motion video titles into such HTML pages by specifying a few parameters of a desired title or a desired portion of a title to be requested from a video server 250. The applet 212 builds bit stream control signals from the specification of the title or the portion of the title. The bit stream control signals request transmission of the title or the portion of the title from a bit stream server such as a video server 250 and are in a form appropriate for processing by the bit stream server. The applet 212 transmits the bit stream control signals to the bit stream server 250 to thereby request that the bit stream server 250 initiate transmission of a bit stream representing the requested title or the requested portion of the title. The applet 212 also builds decoder control signals from the specification of the title or the portion of the title. The decoder control signals direct a bit stream decoder 204 to receive the requested bit stream from the bit stream server 250 and to decode a motion video signal from the bit stream. The applet 212 transmits the decoder control signals to the decoder 204 to cause the decoder 204 to receive the bit stream and to decode the motion video signal from the bit stream.

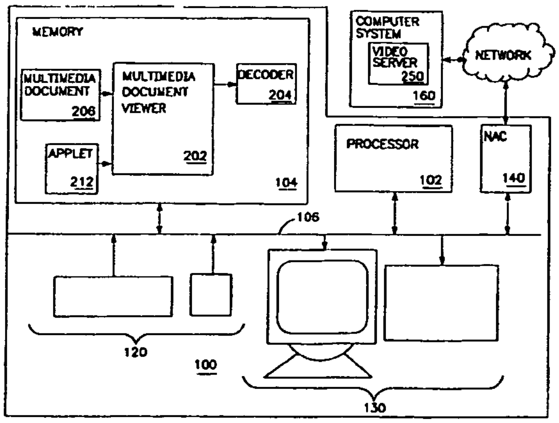


FIG. 1

EP 0 803 826 A2

Printed by Jouve, 75001 PARIS (FR)

Description

A portion of the disclosure of this patent document contains material which is subject to copyright protection. Th copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it app ars in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

FIELD OF THE INVENTION

The present invention relates to computer graphical display of motion video and, in particular, to a method and apparatus for facilitating inclusion of motion video in multimedia computer displays.

BACKGROUND OF THE INVENTION

Video servers, including networked video servers, transmit "bit streams" to a video client. Such bit streams, which are sometimes referred to as "streams," generally represent video and/or audio signals which represent titles in a library of multimedia sources. Examples of titles of such a library typically include recordings of motion pictures. In general, a video server receives from a video client a request for a particular title and transmits a stream of the particular title to the video client. An example of a video client is a set top box which is generally known and which decodes the stream received from the video server and transmits the decoded signal to a connected television. The requesting of a particular title, receiving the stream of the particular title, and decoding the stream for display on a television are collectively and generally referred to as video on demand.

Examples of such video on demand servers are described in U.S. Patent Application Serial Number 08/572,639, filed December 14, 1995 by Kallol Mandal and Steven Kleiman and entitled "Method and Apparatus for Delivering Simultaneous Constant Bit Rate Compressed Video Streams at Arbitrary Bit Rates with Constrained Drift and Jitter" (hereinafter the '639 Application) and in U.S. Patent Application Serial Number 08/572,648, filed December 14, 1995 by Kallol Mandal and Steven Kleiman and entitled "Method and Apparatus for Distributing Network Bandwidth on a Video Server for Transmission of Bit Streams Across Multiple Network Interfaces Connected to a Single Internet Protocol (IP) Network" (hereinafter the '648 Application). Both the '639 Application and the '648 Application are incorporated herein in their entirety by reference.

The popularity of the Internet global network is growing extremely rapidly, and perhaps the most popular protocol of the Internet is the Hyper Text Transfer Protocol (HTTP) of the World Wide Web. According to the HTTP protocol of the World Wide Web, documents, which are generally referred to as "pages," incorporate text, graphical images, sound, and motion video which, when viewed, form a multimedia presentation to user. Such pages are typically viewed using a World Wide Web browser, which is a computer process capable of retrieving HTTP pages and presenting the contents of such pages to a user of a computer system through output devices such as a computer video display device and a computer audio circuit coupled to one or more audio speakers. An example of a World Wide Web browser is the Netscape browser available from Netscape Communications Corporation of Mountain View, California.

To display motion video, conventional browsers typically (i) transfer to the computer system in which the browser executes an entire data file which includes data representing a title and (ii) subsequently initiate execution of a player computer process which displays the title to the user on a computer display device. The player computer process is separate from the browser and therefore displays the motion video of the title outside of the page displayed by the browser. In addition, transferring the entire data file prior to displaying the motion video of the title delays substantially the display of the motion video since such data files are typically quite large, e.g., typically 1.8 gigabytes of data to represent a two-hour, VHS-quality motion picture.

Currently, no browser is capable of seamlessly integrating motion video streams into a page of the World Wide Web.

SUMMARY OF THE INVENTION

In accordance with the present invention, a computer process which requests streams of motion video titles and decodes and displays the motion video signals of the stream for display in a computer display device is constructed in the form of an applet of a multimedia document viewer such as a World Wide Web browser. Accordingly, a designer of multimedia documents such as HTML pages can easily incorporate motion video titles into such HTML pages by specifying a few parameters of a desired title or a desired portion of a title to be requested from a video server. The specification of the parameters is in the general form of a well-known parameter specification format dictated by the particular interface of the computer instruction language in which the applet is written.

The applet builds bit stream control signals from the specification of the title or the portion of the title. The bit stream control signals request transmission of the title or the portion of the title from a bit stream server such as a video server and ar in a form appropriate for processing by th bit stream server. The applet transmits the bit stream control signals to the bit stream server to thereby request that th bit stream server initiate transmission of a bit stream r presenting the requested title or the requested portion of th title.

The applet also builds decoder control signals from the specification of the title or the portion of the title. The decoder control signals direct a bit stream decoder to receive the requested bit stream from the bit stream server and to decode a motion video signal from the bit stream. Th applet transmits the decoder control signals to the decoder to cause the decoder to receive the bit stream and to decode the motion video signal from the bit stream.

By using an applet of a multimedia document viewer to request and control receipt by a decoder of a motion video bit stream and to control decoding of the motion video bit stream by the decoder, a designer of a multimedia document can easily and conveniently include motion video images in multimedia documents. In addition, since the applet transmits bit stream control signals to a video server, the motion video signals which can be incorporated into a multimedia document are any such motion video signals stored in such a video server. Such video servers will likely include a large number and wide variety of motion video signals, thereby providing a wealth of motion video content for inclusion in multimedia documents.

The present invention will now be further described, by way of example, with reference to the accompanying drawings, in which:-

Figure 1 is a block diagram of a computer system which is connected to a video server through a network and which includes a multimedia document viewer which in turn processes an applet to include motion video images in a representation of a multimedia document in accordance with the presenting invention.

Figure 2 is a block diagram showing the multimedia document viewer, applet, and video server of Figure 1 in greater detail.

Figure 3 is a block diagram of an applet tag of Figure 2 in greater detail.

Figure 4 is a block diagram of the applet of Figure 2 in greater detail.

DETAILED DESCRIPTION

In accordance with the present invention, a multimedia document 206 (Figure 2) includes an applet 214 which causes a multimedia document viewer 202 to execute an applet 212. Execution of applet 212 requests transmission of a bit stream of a particular title from a video server 250 and controls receipt and decoding of the bit stream by a decoder 204. Decoder 204, in response to control signals received from applet 212, decodes the received bit stream to produce a motion video image and displays the motion video image as an integral part of the representation of multimedia document 206. To include a motion video image as an integral part of a multimedia document, a designer of the multimedia document simply includes in the multimedia document an applet tag, e.g., applet tag 214, which specifies (i) applet 212, (ii) video servoer 250 as the source of a bit stream, and (iii) the particular bit stream to request from video server 250. A brief description of the operating environment of multimedia document viewer 202 and applet 212 facilitates appreciation ofthe present invention.

Figure 1 is a block diagram of a computer system 100 which is generally of the architecture of most computer systems available today. Computer system 100 includes a processor 102 which fetches computer instructions from a memory 104 through a bus 106 and executes those computer instructions. In executing computer instructions fetched from memory 104, processor 102 can retrieve data from or write data to memory 104, display information on one or more computer display devices 130, or receive command signals from one or more user-input devices 120. Processor 102 can be, for example, any of the SPARC processors available from Sun Microsystems, Inc. of Mountain View, California. Memory 104 can include any type of computer memory including, without limitation, randomly accessible memory (RAM), read-only memory (ROM), and storage devices which include magnetic and optical storage media such as magnetic or optical disks. Computer 100 can be, for example, any of the SPARCstation workstation computer systems available from Sun Microsystems, Inc. of Mountain View, California.

Sun, Sun Microsystems, the Sun Logo, Java and Hot Java are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All SPARC trademarks are used under license and are trademarks of SPARC International, Inc. in the United States and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

Computer display devices 130 can include generally any computer display device such as a printer, a cathode ray tube (CRT), light-emitting diode (LED) display, or a liquid crystal display (LCD). User input devices 120 can include generally any user input device such as a keyboard, a keypad, an electronic mouse, a trackball, a digitizing tablet, thumbwheels, a light-sensitive pen, a touch-sensitive pad, or voice-recognition circuitry.

Computer system 100 also includes network access circuitry 140 which is coupled to processor 102 and memory 104 through bus 106 and which is coupled to a network 150. In accordance with control signals received from processor 102 through bus 106, network access circuitry 140 coordinates transfer of data through network 150 between network access circuitry 140 and similar network access circuitry (not shown) in computer 100B or other computer systems coupled to comput r syst m 100 through network 150. Th transfer of data through network 150 is conventional. Since a video stream r pres nting a VHS-quality motion picture encoded in MPEG-1 format has a bit rat of approximately 1.5 Mbit/second to 2 Mbit/second, a useful minimum threshold is that network access circuitry 140 is capable of r - ceiving data at a rate of at least 2 Mbit/second. Higher quality motion video images hav bit rates as high as 8 Mbit/second or high r. Ther fore, in one mbodiment, network acc ss circuitry 140 is capable of receiving data at a rate of at least 8 Mbit/second. Network access circuitry 140 can be generally any circuitry which is used to transfer data between a computer system and network such as computer system 100 and network 150 and can be, for example, an Ethernet controller chip.

A number of computer processes execute in processor 102 from memory 104, including a multimedia document viewer 202 and a decoder 204. Multimedia document viewer 202 is a computer process which reads a multimedia document 206 and displays the multimedia information specified in multimedia document 206 in one or more of computer display devices 130. In one embodiment, multimedia document 206 is a document in HTML format and multimedia document viewer 202 is an HTML viewer such as the Netscape World Wide Web browser available from Netscape Communications Corporation of Mountain View, California. Multimedia document viewer 202 and multimedia document 206 are shown in greater detail in Figure 2.

Multimedia document viewer 202 retrieves data and tags from a multimedia document such as multimedia document 206. A tag is data which is not itself substantive content of a multimedia document but instead provides format information and can include specification of substantive content which is to be included in the multimedia document and which is located in memory 104 outside of multimedia document 206. For example, a tag can specify a file stored in memory 104 as containing a graphical image which is to be included as substantive content of multimedia document 206. The data and tags of multimedia document 206 collectively define the composition, including substantive content and formatting, of multimedia document 206; and multimedia document viewer 202 displays such substantive content in one or more of computer display devices 130 (Figure 1) in accordance with the data and tags of multimedia document 206. In one embodiment, multimedia document 206 is an HTML document, and the data and tags of multimedia document 206 comport with the HTML language. Multimedia document 206 includes an applet tag 214 (Figure 2) which specifies an applet 212 and a number of operational characteristics of applet 212 as described more completely below.

Multimedia document viewer 202 includes an applet interpreter 210 which retrieves from applet 212 computer instructions and translates such computer instructions into computer instructions of a form appropriate for execution by processor 102 (Figure 1) and submits the translated computer instructions to processor 102 for execution. In one embodiment, applet interpreter 210 (Figure 2) translates and submits for execution a single computer instruction of applet 212 prior to translation and submission for execution of a subsequent computer instruction of applet 212. Applet interpreter 210 can be, for example, the Java applet interpreter or the Hot Java World Wide Web browser available from Sun Microsystems, Inc. and, in such an embodiment, applet 212 comports with the Java computer instruction language interpreted by the Java applet interpreter. As described more completely below, applet 212 is a novel applet which, when executed by processor 102 (Figure 1) through applet interpreter 210 (Figure 2), requests a title from a video server 250 and causes the received bit stream representing the requested title to be decoded in a decoder 204 and displayed in a computer display device as an integral part of a multimedia display of multimedia document 206.

In executing the computer instructions of applet 212, applet interpreter 210 transmits, through network 150 (Figure 1), control signals to an applications programming interface (API) 252 (Figure 2) of a video server 250 which executes within a computer system 160 (Figure 1). Illustrative examples of video server 250 of computer system 160 are described in the '639 and '648 Applications. API 252 (Figure 2) of video server 250 implements a remote procedure calling (RPC) protocol in which API 252 controls video server 250 in response to control signals received by API 252. For example, in response to control signals which request a title and which are transmitted to API 252 by applet interpreter 210, API 252 causes a bit pump 254 of video server 250 to initiate transmission through network 150 (Figure 1) to decoder 204 (Figure 2) of a bit stream representing the requested title. In addition, API 252 can transmit to applet interpreter 210 status information regarding a title stored within video server 250 or regarding a bit stream transmitted by bit pump 254 in response to control signals requesting such status information.

Decoder 204 is a computer process executing within processor 102 (Figure 1) from memory 104. Decoder 204 receives data representing a motion video display encoded in a particular format. In one embodiment, decoder 204 is the MPEG Expert (MPX) decoder available from Applied Vision and decodes motion video signals according to the MPEG-1 encoding format. Applet interpreter 210 transmits to decoder 204 control signals which control the decoding by decoder 204 of the bit stream received from bit pump 254 of video server 250. Specifically, applet interpreter 210 transmits to decoder 204 control signals directing decoder 204 to start or stop decoding the bit stream received from bit pump 254 or specifying characteristics of the bit stream received from bit pump 254 such as the bit rate, encoding format, and the coordinates of a particular location within one or more of computer display devices 130 (Figure 1) in which to display the decoded motion video images. In addition, applet 212 determines which communications port through network access circuitry 140 (Figure 1) the bit stream is to be received and transmits to decoder 204 (Figure 2) control signals identifying the selected communications port. Applet 212 can therefore determine which communications ports are used by other applications and can avoid conflicts resulting from access of decoder 204 of a communications port by selecting a communications port which is not used by another computer process of comput r system 100 (Figure 1).

Applet tag 214 is shown in greater detail in Figure 3. Applet tag 214 includes a number of fields which collectively define a bit stream to be received and decoded for display by decoder 204 (Figure 2). A field is a collection of data which collectively define a item of information. Applet tag 214 includes (i) an applet identifier field 302, (ii) a width field 304, (iii) a height field 306, (iv) a server identifier field 308, and (v) an encoding format field 310. Applet tag 214 can also include any of the following optional fields: (vi) a title field 312, (vii) an image field 314, (viii) a play/pause field 316, (ix) a start field 318, and (x) a duration field 320.

Applet identifier field 302 specifies applet 212 as the applet to be retrieved and executed by applet interpreter 210. Width field 304 and height field 306 specify the width and height, respectively, in display coordinate space of a computer display device, i.e., specify the size of the viewport in which the decoded motion video image is displayed. Server identifier field 308 specifies video server 250 (Figure 2) as the source of the desired bit stream. Encoding format field 310 (Figure 3) specifies the particular encoding format, e.g., MPEG1SYS encoding format, ofthe bit stream received by decoder 204 (Figure 2). Title field 312 (Figure 3) specifies the particular title to be retrieved from server 250 (Figure 2). Alternatively, title field 312 can specify the address of a multicast bit stream.

Image field 314 (Figure 3), if included, specifies a still video image to be displayed in the space specified by width field 304 and height field 306 if the title specified by title field 312 is unavailable. Play/pause field 316, if included, specifies whether the motion video image received from video server 250 (Figure 2) is initially in a play state or in a paused state. Start field 318 (Figure 3), if included, specifies an offset into the title of a portion of the title, i.e., the point within the title at which the bit stream should begin. For example, start field 318 can specify that the requested bit stream begin at 3 minutes and 10 seconds into the title. Duration field 320, if included specifies the duration of a desired portion of the title. For example, duration field 320 can specify that a 30-minute portion of the title is requested. In one embodiment, start field 318 and duration field 320 are specified in terms of an integer number of nanoseconds.

Thus, by specifying the few fields described above and shown in Figure 3, a designer of multimedia document 206 can include as an integral part of multimedia document 206 a motion video image retrieved from video server 250. The following is an illustrative example of applet tag 214 in HTML format.

```
<applet code="SunMediaCenterPlayer.class" width=704 height=520>
<param name=port value="1973">
<param name=format value="MPEG1SYS">
<param name=host value="sqas-6">
<param name=img value="/images/bkgx.gif">
</applet>
```

Applet 212 (Figure 2) includes computer instructions which, when executed, request a title from video server 250 and control decoding and display of the decoded motion video signals by decoder 204 and is shown in greater detail in Figure 4. The computer instructions of applet 212 are organized into various levels, each of which defines a respective component of the behavior of applet 212. Applet 212 includes a player level 402, an API level 404, a decoder level 406, and a detailed decoder level 408.

Player level 402 includes computer instructions which, when executed, implement a graphical user interface in which a user can control the bit stream received by video server 250 (Figure 2) and the display of the decoded motion video signals of the bit stream by physical manipulation of one or more of user input devices 120 (Figure 1). In one embodiment, the computer instructions of player level 402 (Figure 4), when executed, cause graphical and/or textual representation of control mechanisms to be displayed in one or more of computer display devices 130 (Figure 1). Such control mechanisms are known and conventional and include, without limitation, virtual buttons, pull-down menus, virtual radio buttons, virtual check boxes, and sliding scroll bars. In a conventional manner, a user activates one or more of such control mechanisms by physical manipulation of one or more of user input devices 120 (Figure 1) and such physical manipulation results in receipt by player level 402 (Figure 4) of applet 212 of signals and/or data representing such activation.

API level 404 includes computer instructions which, when executed, implement the RPC protocol of API 252 (Figure 2) of video server 250 and invoke RPC calls to API 252 to control the bit stream transmitted by bit pump 254 in accordance with interaction of a user with the graphical user interface implemented by player level 402 (Figure 4).

Decoder level 406 and detailed decoder level 408 collectively control operation of decoder 204 (Figure 2), generally controlling the decoding of the bit stream received from video server 250 by decoder 204 and the display in a computer display device of the decoded motion video image. Decoder level 406 includes computer instructions and data structures which are not specific to any particular decoder, while detailed decoder level 408 includes computer instructions and data structures which are specific to decoder 204. It is generally preferred that detailed decoder level 408 is as small and simple as possible such that the majority of computer instructions of decoder levels 406 and 408 ar included in decoder level 406. Accordingly, adapting applet 212 (Figure 2) to operat in conjunction with a decoder oth r than decoder 204 requir s modification of only detailed decoder level 408 and, therefor , as littl modification as possible.

App ndix A is a computer sourc code listing of a preferred embodiment of applet 212. Th modules of Appendix A ar written in the Java applet computer instruction language developed by Sun Microsystems, Inc. of Mountain View, California. The computer instructions of the Java applet computer instruction language are object-oriented, and each of the modules of Appendix A represents a respective class of objects. Player level 402 (Figure 4), in this embodiment, includes classes SunMediaCenterPlayer, Player, and PositionSlider as defined in the computer source code listing of Appendix A. API level 404, in this embodiment, includes classes MsmPlayer, MsmSession, MsmAccessRight, MsmPersistence, MsmPlaylist, MsmToString, MsmItem, MsmTitleItem, MsmDeadAirItem, MsmException, XdrBlock, and PortMapper as defined in the computer source code listing of Appendix A. Decoder level 406, in this embodiment, includes classes Decoder and DecoderImpl as defined in the computer source code listing of Appendix A. Detailed decoder level 408, in this embodiment, includes class MpxDecoderImpl as defined in the computer source code listing of Appendix A.

In the preferred embodiment of the present invention defined by Appendix A, a module "loop" includes computer instructions of the C computer instruction language and defines a loop computer process which executes independently of multimedia document viewer 202 (Figure 2). The loop computer process cooperates with multimedia document viewer 202 and decoder 204 to request and receive from video server 250 bit streams representing multicast motion video signals.

The above description is illustrative only and is not limiting. The present invention is therefore defined solely and completely by the appended claims together with their full scope of equivalents.

APPENDIX A

SunMediaCenterPlayer

```
/*
 * @(#)SunMediaCenterPlayer.java
 *
 * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
 *
 * version     1.0
 * author Christopher Lindblad
 *                  - &
 */

import java.applet.*;
import java.awt.*;
import java.net.*;
import java.io.*;
import COM.Sun.isg.smcjc.*;

public class SunMediaCenterPlayer extends Applet {
    private Player player;
    private TextArea reporter;
    private Thread thread;

    public SunMediaCenterPlayer() {
     setLayout(new BorderLayout());
     player = new Player();
     add("Center", player);
    }

    public synchronized void init() {
     if (reporter != null && reporter.getParent() == this) {
         remove(reporter);
         reporter.setText("");
         validate();
     }
     try {
         int port=getParameterInt("port",-1);
         int vc=getParameterInt("vc",-1);
            if (vc!=-1){
             player.init(
             getParameterRequired("host"),
             getParameterRequired("title"),
                getParameterLong("start", 0L),
                getParameterLong("duration", 0L),
                getParameterString("loop",
       "false").equalsIgnoreCase("true"),
                getParameterString("cmd", "play"),
                getParameterImage("img", null),
                     vc,"",
                     getParameterURL("CC"),
                getParameterRequired("interface"));
               }else{
                 if (port==-1){
                player.init(
                getParameterRequired("host"),
                getParameterRequired("title"),
                getParameterLong("start", 0L),
                getParameterLong("duration", 0L),
                getParameterString("loop",
       "false").equalsIgnoreCase("true"),
                getParameterString("cmd", "play"),
                getParameterImage("img", null),
                     port,"",
                     getParameterURL("CC"),null);
                 }else{
                player.init(
                getParameterRequired("host"),
                "none",0L,0L,false,"play",
                getParameterImage("img", null),
                     port,
                getParameterRequired("format"),
                     getParameterURL("CC"),null);
                 }
               }
         } catch (IOException e) {
             report(e, "parsing Sun MediaCenter player parameters");
         }
        }

        public synchronized void start() {
         try player.start(); catch (IOException e)
             report(e, "starting a Sun MediaCenter player");
        }

        public synchronized void stop() {
         try player.stop(); catch (IOException e)
             report(e, "stopping a Sun MediaCenter player");
        }
```

```
    private String getParameterRequired(String key) throws
IOException {
     String val = getParameter(key);
     if (val != null) return val;
     throw new IOException("missing required parameter " + key);
    }

    private int getParameterIntRequired(String key) throws
IOException {
     String val = getParameter(key);
     if (val != null)
           try return Integer.parseInt(val); catch
(NumberFormatException e)
              throw new IOException(
                  "parameter " + key + " is not a valid int: " +
val);
;
     throw new IOException("missing required parameter " + key);
    }

    private URL getParameterURL(String key) {
        URL res=null;
     String val = getParameter(key);
     if (val == null) return null;
        try res=new URL(val);
          catch (MalformedURLException e) try res=new
URL(getDocumentBase(),val);
            catch (MalformedURLException f)
System.out.println("MalformedURLException");
        return res;
    }

    private String getParameterString(String key, String dflt) {
     String val = getParameter(key);
     if (val == null) return dflt;
     return val;
    }

    private int getParameterInt(String key, int dflt) throws
IOException {
     String val = getParameter(key);
     if (val == null) return dflt;
     try return Integer.parseInt(val); catch
(NumberFormatException e)
        throw new IOException(
         "parameter " + key + " is not a valid int: " + val);
    }
    private long getParameterLong(String key, long dflt) throws
IOException {
     String val = getParameter(key);
     if (val == null) return dflt;
     try return Long.parseLong(val); catch (NumberFormatException
e)
         throw new IOException(
          "parameter " + key + " is not a valid long: " + val);
    }

    private Image getParameterImage(String key, Image dflt) {
     String val = getParameter(key);
     if (val == null) return dflt;
     return getImage(getDocumentBase(), val);
    }

    private synchronized void report(Exception e, String doing) {
     ByteArrayOutputStream os = new ByteArrayOutputStream();
     PrintStream ps = new PrintStream(os);
     ps.print("An error occurred while ");
     ps.print(doing);
     ps.println(":");
     e.printStackTrace(ps);
     if (reporter == null) {
         reporter = new TextArea("");
         reporter.setEditable(false);
     }
     reporter.appendText(os.toString());
     if (reporter.getParent() != this) {
         add("North", reporter);
         validate();
     }
    }

}
```

Player

```
/*
 * @(#)Player.java
 *
 * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
 *
 * version     1.1sc
 * author Christopher Lindblad    ( Msm API & Mpx API )
 * author Stephane CACHAT         (Closed Caption & Multicasting)
 *
 */

package COM.Sun.isg.smcjc;

import java.applet.*;
import java.awt.*;
import java.io.*;
import java.net.*;

public class Player extends Panel implements Runnable {
    private long playDuration;
    private long startOffset;
    private long seekPosition;
    private long tellPosition;
    private double tellPositiond;
    private MsmPlayer player;
    private String host;
    private String titleName;
    private String msg;
    private String format;
    private Image img;
    private Thread thread;
    private Panel controlLine;
    private Panel controlButtons;
    private TextArea reporter;
    private Decoder decoder;
    private PositionSlider positionSlider;
    private Button[] buttons;
    private int cmd = 999;
    private int initialCmd;
    private int port;
    private boolean loop;
    private boolean Msm;
    private URL CC;
    private List CCt;
    private int CCz=0;
    private String[] CCb=new String[1024];
    private Double[] CCi=new Double[1024];
    private int CCl=0;
    private int CCo=0;
    private int CCm=0;
    private boolean playing = false;
    private TextField CCs;
    private String ATM;

    public Player() {
     setLayout(new BorderLayout());
     decoder = new Decoder();
     add("Center", decoder);
    }

    public synchronized void init(
     String host, String titleName,
     long startOffset, long playDuration, boolean loop,
     String cmd, Image img,int port,String format,URL CC,String
ATM)
     throws IOException {
            URLConnection uc;
            Double d;
            String str;
            int i=0;
            int j=0;

            this.port=port;
            if ((port!=-1)&&(ATM==null)){
                Msm=false;
            }else{
                Msm=true;
             this.initialCmd = parseCmd(cmd);
            }
            this.CC=CC;
            this.ATM=ATM;
         this.host = host;
         this.titleName = titleName;
         this.startOffset = startOffset;
         this.playDuration = playDuration;
         this.loop = loop;
         this.img = img;
         this.format = format;
            if (CC!=null){
              CCt= new List();
              CCt.minimumSize(6);
```

```
               CCt.preferredSize(6);
               uc= CC.openConnection();
               DataInputStream in=new
DataInputStream(uc.getInputStream());
               str="-";
            CCb[i]=new String("*");
               CCi[i]=new Double(0.0);
               i++;
               while (in.available()>0){
                 str=in.readLine();
                 while
((str.trim().length()==0)&&(in.available()>0)) str=in.readLine();
                 if (str!=null){
                   j=str.trim().indexOf(' ');
                   if (j>0){
               CCb[i]=new String(str.substring(j+1)).trim();
                     CCt.addItem(CCb[i]);
                     if (CCb[i]==null) CCb[i]="*";
                    CCi[i]=new Double(str.substring(0,j).trim());
                     i++;
                }
                 }
               }
               CCm=i-1;
               in.close();
             }
     }

    public synchronized void start() throws IOException {
     if (reporter != null && reporter.getParent() == this) {
         remove(reporter);
         reporter.setText("");
         validate();
     }
     if (thread == null) {
         cmd = initialCmd;
         thread = new Thread(this);
         thread.start();
     }
    }

    public synchronized void stop() throws IOException {
     if (thread != null) {
         thread = null;
         notify();
     }
    }
    public synchronized boolean action(Event evt, Object arg) {
     if (buttons != null && evt.target instanceof Button) {
          Button b = (Button)evt.target;
          for (int i = 0; i < buttons.length; i++) {
           if (b == buttons[i]) cmd = i;
          }
          notify();
     };
     if (CC != null && evt.target ==CCt) {
          seekPosition = (long)(new
Double(CCi[CCt.getSelectedIndex()].doubleValue()*10).intValue())*
100000000;
          cmd = SEEK;
          notify();
         };
     if (CC != null && evt.target==CCs) {
             if (CCl<CCm){
               CCz=CCl+1;
             }else{
               CCz=0;
             };

while((CCz!=CCl)&&(CCb[CCz].indexOf(CCs.getText())<0))  {
               CCz++;
               if (CCz>CCm) CCz=0;
             }
             if (CCb[CCz].indexOf(CCs.getText())>=0){
               CCt.select(CCz);
               CCt.makeVisible(CCz+1);
           seekPosition = (long)(new
Double(CCi[CCt.getSelectedIndex()].doubleValue()*10).intValue())*
100000000;
           cmd = SEEK;
           notify();
             }
         }
     return true;
    }

    private void setConnect(MsmConnect connect) throws
IOException {
     try {
         player.setConnect(connect);
     } catch (MsmException e) {
         /* Try it with destTiAddr in beta 0.5 syntax. */
System.out.println("DesTiAddr="+connect.destTiAddr);
         InputStream is = new
```

```
StringBufferInputStream(connect.destTiAddr);
          StreamTokenizer st = new StreamTokenizer(is);
          String host;
          int udpport;
             if(ATM==null){
            if (st.nextToken() == StreamTokenizer.TT_WORD &&
           st.sval.equals("host") &&
           st.nextToken() == '=' &&
           st.nextToken() == StreamTokenizer.TT_WORD &&
           (host = st.sval) != null &&
           st.nextToken() == ',' &&
           st.nextToken() == StreamTokenizer.TT_WORD &&
           st.sval.equals("udpport") &&
           st.nextToken() == '=' &&
           st.nextToken() == StreamTokenizer.TT_NUMBER &&
           (udpport = (int)st.nval) != 0) {
           connect.destTiAddr = "be0,"+host+","+udpport;
           player.setConnect(connect);
            } else {
           throw e;
            }
            }else{
            throw e;
            }
     }
    }

    public synchronized void run() {
     Thread currentThread = Thread.currentThread();
     MsmSession session = null;
     MsmTitle title = null;
        MsmItem[] items = null;
        int speed=0;


     if (Msm){
            controlButtons = new Panel();
         controlButtons.setLayout(new FlowLayout());
         controlButtons.add(cmds[PAUSE], new
Button(labels[PAUSE]));
         controlLine = new Panel();
         controlLine.setLayout(new BorderLayout());
         controlLine.add("East", controlButtons);
         positionSlider = new PositionSlider(this);
         controlLine.add("Center", positionSlider);
         add("South", controlLine);
            if (CC!=null){
               Panel CCp=new Panel();
               CCp.setLayout(new BorderLayout());
               Panel CCq=new Panel();
               CCq.setLayout(new BorderLayout());

               CCs= new TextField(15);
               CCs.isEditable();
            CCq.add("South", CCs);
               Label l=new Label("Search");
            CCq.add("Center", l);
               CCp.add("East",CCq);
               CCp.add("Center",CCt);
            controlLine.add("North",CCp);
            }
        }
     try {
         if (Msm){
                items = new MsmItem[1];
             session = new MsmSession(host);
             title = session.getTitleStatus(titleName);
             if (playDuration == 0L) playDuration =
title.totalPlayDuration;
                format=title.format;
            }
         decoder.init(format, img,host,port,ATM);
         if (Msm){
                titleInit(title);
             player = new MsmPlayer(session, info(),
MsmPlayer.TIME_MAXTIME);
             player.setPersistence(new MsmPersistence(
              MsmPersistence.TYPE_NONE,
              MsmPlayer.TIME_MAXTIME));
             items[0] = new MsmTitleItem(
              titleName, playDuration, startOffset, playDuration,
              playDuration, false, true, title.maxBitRate);
             player.setPlaylist(new MsmPlaylist(
              MsmPlayer.TIME_CURRENT, loop, 0,
MsmPlayer.TIME_MAXTIME,
              items, 0, 0));
             setConnect(new MsmConnect(
              decoder.destTiAddr(), decoder.encap(),
title.maxBitRate));
             playing = false;
             speed = MsmPlayer.SPEED_FORWARD;
            }else{
             invalidate();
             validate();
            }
         while (currentThread == thread) {
          switch (cmd) {
          case NOP: {
              if (Msm) {
                        MsmPlayStatus status =
player.getPlayStatus();
                  if (tellPosition != status.currentPosition) {
                   tellPosition = status.currentPosition;
                   positionSlider.repaint();
                  }

tellPositiond=(tellPosition/1000000000)+3.0;
                        if (CC!=null){
                          CCo=CCl;
                          while
((CCi[CCl+1].doubleValue()<tellPositiond)&&(CCl+1<CCm))  CCl++;
                          while
((CCi[CCl].doubleValue()>tellPositiond)&&(CCl>0)) CCl--;
                          if (CCo!=CCl) {
                             CCt.select(CCl-1);
                             CCt.makeVisible(CCl);
                          }
                        }
                  player.setPersistence(new MsmPersistence(
                   MsmPersistence.TYPE_NONE,
                   status.currentDate+60*1000000000L));
              }
                    break;
          }
          case PAUSE: {
              decoder.pause();
              if (Msm) player.pause(MsmPlayer.TIME_CURRENT);
              decoder.flush();
              playing = false;
              decoder.play();
              break;
          }
          case GOTO_START: {
              tellPosition = 0L;
              if (Msm) positionSlider.repaint();
              decoder.stop();
              if (Msm) player.play(MsmPlayer.SPEED_FORWARD,
                   0L,
                   0L,
                   MsmPlayer.TIME_CURRENT);
              decoder.flush();
```

```
    break;
}
case GOTO_END: {
    tellPosition = playDuration;
    if (Msm) positionSlider.repaint();
    decoder.stop();
    if (Msm) player.play(MsmPlayer.SPEED_REVERSE,
          playDuration,
          0L,
          MsmPlayer.TIME_CURRENT);
    decoder.flush();
    break;
}
case SEEK: {
    tellPosition = seekPosition;
    if (Msm) positionSlider.repaint();
    if (playing) {
     decoder.flush();
     if (Msm) player.play(speed,
              seekPosition,
              MsmPlayer.TIME_MAXTIME,
              MsmPlayer.TIME_CURRENT);
    } else {
     long duration = SEEKDURATION;
     long position = seekPosition-duration;
     if (position < 0L) {
         duration += position;
         position -= position;
     }
     decoder.play();
     decoder.flush();
     if (Msm) player.play(MsmPlayer.SPEED_FORWARD,
              position,
              duration,
              MsmPlayer.TIME_CURRENT);
    }
    break;
}
default: {
    decoder.play();
    decoder.flush();
    if (Msm){
              speed = cmd;
        player.play(speed,
          MsmPlayer.TIME_CURRENT,
          MsmPlayer.TIME_MAXTIME,
          MsmPlayer.TIME_CURRENT);
```

```
                  playing = true;
                        if (CC!=null)
                          if (CCo!=CCl) {
                             CCt.select(CCl-1);
                             CCt.makeVisible(CCl);
                          }
                    }
          }
          }
          cmd = NOP;
          try wait(100); catch (InterruptedException e);
         }
     } catch (Exception e) {
         report(e, "communicating with a Sun MediaCenter
server");
     } finally {
         try {
          try decoder.stop(); catch (Exception e)
              report(e, "stopping a video decoder");
                if (Msm){
              if (player != null) {
                  try player.delete(); catch (Exception e)
                   report(e, "deleting a Sun MediaCenter
player");
                  player = null;
                    }
          }
         } finally {
                if(Msm){
              if (session != null) {
                  try session.close(); catch (Exception e)
                   report(e, "closing a Sun MediaCenter
connection");
              }
                }
         }
     }
    }

    /*
     * Callback from the PositionSlider.
     * Unsynchronized to avoid deadlock.
     * @return value between 0 and 1 indicating where in the file
we are.
     */
    public double tell() {
     if (playDuration == 0L) return 0.0D;
```

```
 return (double)tellPosition / (double)playDuration;
}

/*
 * Callback from the PositionSlider.
 * Seek to a relative position in a file.
 * @param position Value between 0 and 1
 * indicating where in the file to go.
 */
public synchronized void seek(double position) {
 if (playDuration == 0) return;
 seekPosition = (long)(position*playDuration);
 cmd = SEEK;
 notify();
}

private String info() throws UnknownHostException {
    String hostName =
InetAddress.getLocalHost().getHostName();
    String javaVersion = System.getProperty("java.version");
    String javaVendor = System.getProperty("java.vendor");
    String osArch = System.getProperty("os.arch");
    String osName = System.getProperty("os.name");
    String osVersion = System.getProperty("os.version");
    return hostName
        + " Java " + javaVersion + " (" + javaVendor + ")"
        + " (" + osArch + " " + osName + " " + osVersion +
")";
}

private void addButton(int i) {
 buttons[i] = new Button(labels[i]);
 controlButtons.add(cmds[i], buttons[i]);
}

/**
 * Initialize for a title.
 * @param title The title to play.
 */
private void titleInit(MsmTitle title) throws IOException {
 controlButtons.removeAll();
 buttons = new Button[labels.length];
 for (int i = MsmPlayer.SPEED_SLOWEST_FORWARD;
      i <= MsmPlayer.SPEED_SCENE_FORWARD;
      i++) {
     if (title.speedScale[i] != 0) {
      addButton(GOTO_START);
```

```
                    break;
                   }
              }
              for (int i = MsmPlayer.SPEED_SCENE_REVERSE;
                   i <= MsmPlayer.SPEED_SLOWEST_REVERSE;
                   i++) {
                  if (title.speedScale[i] != 0) addButton(i);
              }
              addButton(PAUSE);
              for (int i = MsmPlayer.SPEED_SLOWEST_FORWARD;
                   i <= MsmPlayer.SPEED_SCENE_FORWARD;
                   i++) {
                  if (title.speedScale[i] != 0) addButton(i);
              }
              for (int i = MsmPlayer.SPEED_SCENE_REVERSE;
                   i <= MsmPlayer.SPEED_SLOWEST_REVERSE;
                   i++) {
                  if (title.speedScale[i] != 0) {
                   addButton(GOTO_END);
                   break;
                  }
              }
              /* recompute layout */
              controlLine.invalidate();
              invalidate();
              validate();
              /* resize if we need to */
              Component c = getParent();
              while (c != null) {
                  if (c instanceof Applet) {
                   Dimension ps = c.preferredSize();
                   Rectangle b = c.bounds();
                   if (ps.width != b.width || ps.height != b.height) {
                       // This wedges Netscape Navigator 2.0
                       // c.resize(ps.width, ps.height);
                   }
                   break;
                  }
              }
             }

             private void report(Exception e, String doing) {
              ByteArrayOutputStream os = new ByteArrayOutputStream();
              PrintStream ps = new PrintStream(os);
              ps.print("An error occurred while ");
              ps.print(doing);
              ps.println(":");
```

```
 e.printStackTrace(ps);
 if (reporter == null) {
     reporter = new TextArea("");
     reporter.setEditable(false);
 }
 reporter.appendText(os.toString());
 if (reporter.getParent() != this) {
     add("North", reporter);
     validate();
 }
}

private int parseCmd(String cmd) throws IOException {
 for (int i = 0; i < cmds.length; i++) {
     if (cmd.equalsIgnoreCase(cmds[i])) return i;
 }
 throw new IOException("Not a valid Player command: "+cmd);
}

private static final long SEEKDURATION = 4000000000L;

private static final int PAUSE = 16;
private static final int GOTO_START = 17;
private static final int GOTO_END = 18;
private static final int SEEK = 19;
private static final int NOP = 20;

private static final String[] labels = {
 "|<<<<",       // MsmPlayer.SPEED_SCENE_REVERSE
 "<<<<",             // MsmPlayer.SPEED_FASTEST_REVERSE
 "<<<",              // MsmPlayer.SPEED_FASTER_REVERSE
 "<<",               // MsmPlayer.SPEED_FAST_REVERSE
 "<",           // MsmPlayer.SPEED_REVERSE
 "|<",               // MsmPlayer.SPEED_SLOW_REVERSE
 "||<",              // MsmPlayer.SPEED_SLOWER_REVERSE
 "|||<",             // MsmPlayer.SPEED_SLOWEST_REVERSE
 ">|||",             // MsmPlayer.SPEED_SLOWEST_FORWARD
 ">||",              // MsmPlayer.SPEED_SLOWER_FORWARD
 ">|",               // MsmPlayer.SPEED_SLOW_FORWARD
 ">",           // MsmPlayer.SPEED_FORWARD
 ">>",               // MsmPlayer.SPEED_FAST_FORWARD
 ">>>",              // MsmPlayer.SPEED_FASTER_FORWARD
 ">>>>",             // MsmPlayer.SPEED_FASTEST_FORWARD
 ">>>>|",       // MsmPlayer.SPEED_SCENE_FORWARD
 "||",               // PAUSE
 "||<<<<",     // GOTO_START
 ">>>>||",      // GOTO_END
```

```
      "",              // SEEK
      "",              // NOP
     };

     private static final String[] cmds = {
      "scene_reverse",     // MsmPlayer.SPEED_SCENE_REVERSE
      "fastest_reverse",   // MsmPlayer.SPEED_FASTEST_REVERSE
      "faster_reverse",    // MsmPlayer.SPEED_FASTER_REVERSE
      "fast_reverse",          // MsmPlayer.SPEED_FAST_REVERSE
      "reverse",           // MsmPlayer.SPEED_REVERSE
      "slow_reverse",          // MsmPlayer.SPEED_SLOW_REVERSE
      "slower_reverse",    // MsmPlayer.SPEED_SLOWER_REVERSE
      "slowest_reverse",   // MsmPlayer.SPEED_SLOWEST_REVERSE
      "slowest_forward",   // MsmPlayer.SPEED_SLOWEST_FORWARD
      "slower_forward",    // MsmPlayer.SPEED_SLOWER_FORWARD
      "slow_forward",          // MsmPlayer.SPEED_SLOW_FORWARD
      "play",              // MsmPlayer.SPEED_FORWARD
      "fast_forward",          // MsmPlayer.SPEED_FAST_FORWARD
      "faster_forward",    // MsmPlayer.SPEED_FASTER_FORWARD
      "fastest_forward",       // MsmPlayer.SPEED_FASTEST_FORWARD
      "scene_forward",     // MsmPlayer.SPEED_SCENE_FORWARD
      "pause",        // PAUSE
      "goto_start",        // GOTO_START
         "goto_end",       // GOTO_END
         "seek",               // SEEK
         "nop",            // NOP
     };

 }
```

P sitionSlider

```
/*
 * @(#)PositionSlider.java
 *
 * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
 *
 * version     1.0
 * author Christopher Lindblad
 *
 */

package COM.Sun.isg.smcjc;

import java.awt.*;
import java.io.*;

class PositionSlider extends Canvas {
    private Player player;
    private int hgap;
    private int vgap;
    private int wid;

    public PositionSlider(Player player) {
     this(player, 5, 5, 6);
    }

    public PositionSlider(Player player, int hgap, int vgap, int
wid) {
     this.player = player;
     this.hgap = hgap;
     this.vgap = vgap;
     this.wid = wid;
    }

    public void update(Graphics g) {
     paint(g);
    }

    public synchronized void paint(Graphics g) {
     Rectangle r = bounds();
     int position = (int)((r.width-hgap*2)*player.tell())+hgap;
     g.setColor(getBackground());
     g.fillRect(0, 0, r.width, vgap*2);
     g.fillRect(0, r.height-vgap*2, r.width, vgap*2);
     g.fillRect(0, vgap*2, r.width-hgap*2, r.height-vgap*2);
     g.fillRect(r.width-hgap, vgap*2, r.width, r.height-vgap*2);
     g.fill3DRect(hgap, vgap*2, r.width-hgap*2, r.height-vgap*4,
false);
     g.fill3DRect(position-2, vgap, wid, r.height-vgap*2, true);
    }

    private synchronized void seek(int x) {
     Rectangle r = bounds();
     double position = ((double)(x-hgap)) /
((double)(r.width-hgap*2));
     if (position < 0.0D) position = 0.0D;
     if (position > 1.0D) position = 1.0D;
     player.seek(position);
    }

    public boolean mouseDown(Event e, int x, int y) {
     seek(x);
     return true;
    }

    public boolean mouseDrag(Event e, int x, int y) {
     seek(x);
     return true;
    }

}
```

MsmPlayer

```
/*
 * @(#)MsmPlayer.java
 *
 * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
 *
 * version     1.0
 * author Christopher Lindblad
 *
 */

package COM.Sun.isg.smcjc;

import java.io.*;

/**
 * Media Stream Manager Client API
 *
 * MSM allows for the creation of "players".  A player is a
persistent entity
 * that provides for the scheduled delivery of isochronous data
to a
 * particular destination.  To accomplish this task, a player
maintains a
 * playlist of titles, the state of a "playhead" which traverses
this
 * playlist, and an access list controlling who can perform
various functions
 * on the player.
 *
 * MSM, when supplied with titles that have been prepared for
presentation at
 * multiple presentation rates, manages the position index
lookups and stream
 * switching necessary for "trick play".
 *
 * Associated with a player is a "playhead" that maintains a
destination for
 * the isochronous data (possibly different than the controlling
client) and a
 * "playPosition" which travels along the playlist at the
selected
 * presentation rate and delivers isochronous data as scheduled
to the
 * destination.  The position, presentation rate, and
```

```
presentation direction
 * of the playhead can be controlled via play(), pause(), and
resume(). The
 * initiation of play can be synchronized with "wall clock time"
via play();
 * presentation will then stay synchronized with wall-clock time
as long as
 * presentation rate and direction are Normal-Rate,
Forward-Direction.
 *
 * Latency from invocation of the play() request until actual
start of stream
 * may be reduced by "pre-rolling" with a play() request that has
zero
 * duration. This may also be used to set a current playlist
position without
 * actually starting play.
 *
 * MSM manages concurrent updates to a playlist by returning a
modification
 * timestamp with playlist status. The modification timestamp
indicates the
 * time of the last modification of the playlist. When a client
wishes to
 * update a playlist, the client will first obtain status
containing a
 * modification timestamp to understand the current state of the
playlist.
 * Based on this status, the client then determines the
appropriate updates
 * and passes those updates along with the modification timestamp
of the
 * status on which the updates were based to msm. If msm finds
that the
 * modification timestamp has not changed, implying that the
clients updates
 * are based on currently valid playlist state, the playlist
update will
 * succeed. If the modification timestamp indicates that the
playlist has
 * been modified since this client obtained status, the update
will be
 * rejected. In this case, the client should reobtain status,
reaccess the
 * update, and then if appropriate resubmit the update with the
modification
 * timestamp of the new status. There is a designated timestamp
```

```
that forces
 * playlist modifications, this may be used if some external
method of
 * concurrency control is preferred.
 *
 * MsmPlaylist may be edit while play is in progress.  Normally,
changes to the
 * playlist will not take effect until the current item in play
completes.  A
 * playlist modification can be forced to take effect immediately
by calling
 * resume().  resume() should be called with the speed argument
being the
 * current (or desired new speed) and the startPosition argument
being
 * TIME_CURRENT. If the contents of the playlist at the current
position of
 * the playhead have not been modified, this call will not
disturb the
 * outgoing data stream.
 *
 * MSM optionally maintains players persistently across server
outages. When
 * this option is selected, a successful return from a player
request
 * indicates that the player modifications have been made
persistently.
 * Persistent players may optionally restart play on state
recovery, play may
 * be restarted at the last played position or at the position
that the
 * position that play would be add had no outage occurred.
 *
 * Access to read and modify players is controlled by access
control lists
 * associated with the players.  These may be modified by
 * msmPlayerSetAccess().
 *
 * Access rights are "Read", "Control", and "Admin".  Read rights
all state to
 * be seen.  Control rights allow "trick-play" operations to be
controlled.
 * Admin rights allow creation of players, and connection,
access, and
 * persistence attributes of players to be set.  Access rights
are associated
 * with "agents" (eg users) appropriate for the authorization
```

```
mechanism
 * selected.  The reserved agent name "*" represents ALL agents,
those
 * granting a right to "*", grants the right to all agents.
 *
 */
public class MsmPlayer {
    private MsmSession session;
    private byte[] handle;

    /**
     * Creates a player.  The player is initialized
non-persistent.
     * @param session A server session.
     * @param info Saved, but uninterpreted by server. May be
null.
     *   Used to describe the player for administrative purposes.
     * @param terminateDate Date at which player should be
auto-deleted.
     *   If TIME_MAXTIME, the player will never be auto-deleted,
it must
     *   be deleted via delete.
     * @exception IOException If an error has occurred.
     */
    public MsmPlayer(MsmSession session, String info, long
terminateDate)
     throws IOException {
         this.session = session;
         XdrBlock call = session.newCall(PLAYER_CREATE);
         call.xdroutString(info);
         call.xdroutMsmTime(terminateDate);
         XdrBlock reply = session.rpc(call);
         handle = reply.xdrinBytes(HANDLELEN);
         reply.done();
     }

    MsmPlayer(MsmSession session, XdrBlock xdr) {
     this.session = session;
     handle = xdr.xdrinBytes(HANDLELEN);
    }

    void xdrout(XdrBlock xdr) {
     xdr.xdroutBytes(handle,HANDLELEN);
    }

    public MsmSession getSession() {
     return session;
    }

    public byte[] getHandle() {
     return handle;
    }

    /**
     * Opens an existing player.
     * @param session A server session.
     * @param handle An opaque handle to the player.
     */
    public MsmPlayer(MsmSession session, byte[] handle) {
     this.session = session;
     this.handle = handle;
    }

    /**
     * Deletes the player.  In progress play of the player is
stopped.
     * @exception IOException If an error has occurred.
     */
    public void delete() throws IOException {
     XdrBlock call = session.newCall(PLAYER_DELETE);
     this.xdrout(call);
     session.rpc(call).done();
    }

    /**
     * Modifies access control list for player.
     * @param rights The access modifications.
     * @exception IOException If an error has occurred.
     */
    public void setAccess(MsmAccessRight[] rights) throws
IOException {
     XdrBlock call = session.newCall(PLAYER_SETACCESS);
     this.xdrout(call);
     call.xdroutInt(rights.length);
     for (int i = 0; i < rights.length; i++)
rights[i].xdrout(call);
     session.rpc(call).done();
    }

    /**
     * Get access control list for player.
     * @return The access modifications.
     * @exception IOException If an error has occurred.
     */
    public MsmAccessRight[] getAccess() throws IOException {
     XdrBlock call = session.newCall(PLAYER_GETACCESS);
     this.xdrout(call);
     XdrBlock reply = session.rpc(call);
     MsmAccessRight[] result = new
MsmAccessRight[reply.xdrinInt()];
     for (int i = 0; i < result.length; i++) {
         result[i] = new MsmAccessRight(reply);
     }
     reply.done();
     return result;
    }

    /**
     * Sets persistence for player.
     * @param prstp A MsmPersistence containing the persistence
to be set.
     * @exception IOException If an error has occurred.
     */
    public void setPersistence(MsmPersistence prst) throws
IOException {
     XdrBlock call = session.newCall(PLAYER_SETPERSISTENCE);
     this.xdrout(call);
     prst.xdrout(call);
     session.rpc(call).done();
    }

    /**
     * Get persistence information for player.
     * @exception IOException If an error has occurred.
     */
    public MsmPersistence getPersistence() throws IOException {
     XdrBlock call = session.newCall(PLAYER_GETPERSISTENCE);
     this.xdrout(call);
     XdrBlock reply = session.rpc(call);
     MsmPersistence result = new MsmPersistence(reply);
     reply.done();
     return result;
    }

    /**
     * Replaces a portion of the playlist for this player. The
portion to be
     * replaced and the new titles to inserted are indicated via
MsmPlaylist
     * struct pointed to by playlistp.
     * @param playlist A MsmPlaylist that indicates the period on
```

```
the playlist
      *   to be (re)scheduled and the new titles to place within
that period.
      * @exception IOException If an error has occurred.
      */
     public void setPlaylist(MsmPlaylist playlist) throws
IOException {
      XdrBlock call = session.newCall(PLAYER_SETPLAYLIST);
      this.xdrout(call);
      playlist.xdrout(call);
      session.rpc(call).done();
     }

     /**
      * Obtains a portion of the playlist for this player.
      * @param startPosition The position within the playlist at
which to start
      *     returning status.
      * @param playlistDuration The number of milliseseconds of
the playlist for
      *   which to return status.
      * @exception IOException If an error has occurred.
      */
     public MsmPlaylist getPlaylist(long startPosition, long
playlistDuration)
      throws IOException {
          XdrBlock call = session.newCall(PLAYER_GETPLAYLIST);
          this.xdrout(call);
          call.xdroutMsmTime(startPosition);
          call.xdroutMsmTime(playlistDuration);
          XdrBlock reply = session.rpc(call);
          MsmPlaylist result = new MsmPlaylist(reply);
          reply.done();
          return result;
      }

     /**
      * Obtains the playlist for this player.
      * @exception IOException If an error has occurred.
      */
     public MsmPlaylist getPlaylist() throws IOException {
      return getPlaylist(TIME_ZERO, TIME_MAXTIME);
     }

     /**
      * MsmConnects a player to the specified destination address.
      * An error is return if play is in progress at the time of a
```

```
setConnect().
     * @param connect A MsmConnect instance containing a
transport-independent
     *   address string for the destination of Media Server data
controlled
     *   by this player.  A connectp of NULL disconnects the
player from the
     *   current destination.
     * @exception IOException If an error has occurred.
     */
    public void setConnect(MsmConnect connect) throws IOException
{
     XdrBlock call = session.newCall(PLAYER_SETCONNECT);
     this.xdrout(call);
     connect.xdrout(call);
     session.rpc(call).done();
    }

    /**
     * Get current connection for player.
     * @exception IOException If an error has occurred.
     */
    public MsmConnect getConnect() throws IOException {
     XdrBlock call = session.newCall(PLAYER_GETCONNECT);
     this.xdrout(call);
     XdrBlock reply = session.rpc(call);
     MsmConnect result = new MsmConnect(reply);
     reply.done();
     return result;
    }

    /**
     * Schedules play to commence at startDate.  Play
     * will begin at playlist startPosition and continue for
playDuration NPT
     * seconds or until paused.  An error is returned if the
player is not
     * connected.
     * Only one play() command can be pending, a second play()
overrides any
     * pending play().
     * @param speed The speed at which to play.
     * @param startPosition The position within the playlist at
which to begin
     *   play.  TIME_CURRENT means the current play position.
     * @param playDuration The duration of play.
     *   TIME_MAXTIME indicates "forever".
     * @param startDate The wall-clock time of day at which to
begin play.
     *   A value of TIME_CURRENT means start play immediately.
     * @exception IOException If an error has occurred.
     */
    public void play(
     int speed, long startPosition, long playDuration, long
startDate)
     throws IOException {
         XdrBlock call = session.newCall(PLAYER_PLAY);
         this.xdrout(call);
         call.xdroutInt(speed);
         call.xdroutMsmTime(startPosition);
         call.xdroutMsmTime(playDuration);
         call.xdroutMsmTime(startDate);
         session.rpc(call).done();
     }

    /**
     * Pauses play on the player.
     * Only one pause() command can be pending, a second pause()
     * overrides any pending pause().
     * @param pausePosition The position within the playlist at
which to pause
     *   playing.  If current play position is later than
pausePosition
     *   (taking into account the direction of play), play pauses
immediately.
     *   A value of TIME_CURRENT means stop immediately.
     * @return The time at which play actually paused.
     * @exception IOException If an error has occurred.
     */
    public long pause(long pausePosition) throws Exception {
     XdrBlock call = session.newCall(PLAYER_PAUSE);
     this.xdrout(call);
     call.xdroutMsmTime(pausePosition);
     XdrBlock reply = session.rpc(call);
     long result = reply.xdrinMsmTime();
     reply.done();
     return result;
    }

    /**
     * Resumes playing.  Play will continue until paused
     * or the end of the playlist (looped playlists play
forever).
     * @param speed The speed at which to resume play.
     * @param startPosition The position within the playlist at
which to
     *   resume play.  TIME_CURRENT means the current play
position.
     * @exception IOException If an error has occurred.
     */
    public void resume(int speed, long startPosition) throws
IOException {
     XdrBlock call = session.newCall(PLAYER_RESUME);
     this.xdrout(call);
     call.xdroutInt(speed);
     call.xdroutMsmTime(startPosition);
     session.rpc(call).done();
    }

    /**
     * Get play state for a player.
     * @return  A MsmPlayStatus instance.
     * @exception IOException If an error has occurred.
     */
    public MsmPlayStatus getPlayStatus() throws IOException {
     XdrBlock call = session.newCall(PLAYER_GETPLAYSTATUS);
     this.xdrout(call);
     XdrBlock reply = session.rpc(call);
     MsmPlayStatus result = new MsmPlayStatus(reply);
     reply.done();
     return result;
    }

    public String toString() {
     return MsmToString.playerToString(this);
    }

    private static final int HANDLELEN = 12;

    public static final long TIME_BADTIME =                  -1L;
    public static final long TIME_CURRENT =                  -2L;
    public static final long TIME_ZERO    =                   0L;
    public static final long TIME_MAXTIME = 2147483647999999999L;
    public static final long TIME_MINTIME =                   1L;

    public static final int SPEED_SCENE_REVERSE = 0;
    public static final int SPEED_FASTEST_REVERSE = 1;
    public static final int SPEED_FASTER_REVERSE = 2;
    public static final int SPEED_FAST_REVERSE = 3;
    public static final int SPEED_REVERSE = 4;
    public static final int SPEED_SLOW_REVERSE = 5;
    public static final int SPEED_SLOWER_REVERSE = 6;
    public static final int SPEED_SLOWEST_REVERSE = 7;
    public static final int SPEED_SLOWEST_FORWARD = 8;
    public static final int SPEED_SLOWER_FORWARD = 9;
    public static final int SPEED_SLOW_FORWARD = 10;
    public static final int SPEED_FORWARD = 11;
    public static final int SPEED_FAST_FORWARD = 12;
    public static final int SPEED_FASTER_FORWARD = 13;
    public static final int SPEED_FASTEST_FORWARD = 14;
    public static final int SPEED_SCENE_FORWARD = 15;

    private static final int PROG = 0x206d736d;
    private static final int VERS = 1;

    private static final int SERVER_AUTHTYPE       =  1;
    private static final int PLAYER_CREATE         =  2;
    private static final int PLAYER_DELETE         =  3;
    private static final int PLAYER_LIST           =  4;
    private static final int PLAYER_SETACCESS      =  5;
    private static final int PLAYER_GETACCESS      =  6;
    private static final int PLAYER_SETPERSISTENCE =  7;
    private static final int PLAYER_GETPERSISTENCE =  8;
    private static final int PLAYER_SETPLAYLIST    =  9;
    private static final int PLAYER_GETPLAYLIST    = 10;
    private static final int PLAYER_SETCONNECT     = 11;
    private static final int PLAYER_GETCONNECT     = 12;
    private static final int PLAYER_PLAY           = 13;
    private static final int PLAYER_PAUSE          = 14;
    private static final int PLAYER_RESUME         = 15;
    private static final int PLAYER_GETPLAYSTATUS  = 16;
    private static final int TITLE_GETSTATUS       = 17;
}
```

MsmSession

```
/*
 * @(#)MsmSession.java
 *
 * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
 *
 * version     1.0
 * author Christopher Lindblad
 *
 */

package COM.Sun.isg.smcjc;

import java.io.*;
import java.net.*;
import java.util.*;

/**
 * Media Stream Manager Client API
 *
 * The Media Stream Manager (msm) API provides an RPC interface
for managing
 * the scheduling and play of isochronous media streams.
 */
public class MsmSession {
    private String serverHostName;
    private Socket socket;
    private InputStream is;
    private OutputStream os;
    private int prog;
    private int vers;

    /**
     * Create a RPC session for the named server.
     * @param serverHostName The host name of a MSM server.
     * @exception IOException If an error has occurred.
     */
    public MsmSession(String serverHostName) throws IOException {
     this.serverHostName = serverHostName;
     socket = new Socket(serverHostName, pmapGetPort());
     is = new BufferedInputStream(socket.getInputStream());
     os = new BufferedOutputStream(socket.getOutputStream());
    }

    private int pmapGetPort() throws IOException {
     PortMapper pmap = null;
     try {
         pmap = new PortMapper(serverHostName);
         int port;
         prog = 100236;
         vers = 1;
         port = pmap.getPort(prog, vers, PortMapper.IPPROTO_TCP);
         if (port != 0) return port;
         prog = 0x206d736d;
         vers = 1;
         port = pmap.getPort(prog, vers, PortMapper.IPPROTO_TCP);
         if (port != 0) return port;
     } finally {
         if (pmap != null) pmap.close();
     }
     throw new MsmException("no msm server on "+serverHostName);
    }

    /**
     * Closes a session with an MSM server.
     * @exception MsmException If an error has occurred.
     */
    public void close() throws IOException {
     socket.close();
    }

    /**
     * All players on this server.
     * @return an array of all players.
     * @exception IOException If an error has occurred.
     */
    public MsmPlayer[] players() throws IOException {
     XdrBlock reply = rpc(newCall(PLAYER_LIST));
     MsmPlayer[] result = new MsmPlayer[reply.xdrinInt()];
     for (int i = 0; i < result.length; i++) {
         result[i] = new MsmPlayer(this, reply);
     }
     reply.done();
     return result;
    }

    /**
     * Obtains status about titles.
     * @param titleName The name of the title on which to obtain
status.
     * @return the status of the title.
     * @exception IOException If an error has occurred.
     */
    public MsmTitle getTitleStatus(String titleName) throws
IOException {
     XdrBlock call = newCall(TITLE_GETSTATUS);
     call.xdroutString(titleName);
     XdrBlock reply = rpc(call);
        MsmTitle result = new MsmTitle(reply);
     reply.done();
     return result;
    }

    /**
     * Returns the server host name.
     */
    public String getServerHostName() {
     return serverHostName;
    }

    XdrBlock newCall(int proc) {
     return new XdrBlock(prog, vers, proc);
    }

    synchronized XdrBlock rpc(XdrBlock call) throws IOException {
     call.send(os);
     XdrBlock reply = new XdrBlock(is);
     try {
         reply.xdrinReplyHeader(call.callXid());
     } catch (IOException e) {
         throw new MsmException(call.callProc(), e.getMessage());
     }
     int err = reply.xdrinInt();
     if (err != 0) throw new MsmException(call.callProc(), err);
     return reply;
    }

    public String toString() {
     return MsmToString.sessionToString(this);
    }

    private static final int SERVER_AUTHTYPE       =  1;
    private static final int PLAYER_CREATE         =  2;
    private static final int PLAYER_DELETE         =  3;
    private static final int PLAYER_LIST           =  4;
    private static final int PLAYER_SETACCESS      =  5;
    private static final int PLAYER_GETACCESS      =  6;
    private static final int PLAYER_SETPERSISTENCE =  7;
    private static final int PLAYER_GETPERSISTENCE =  8;
    private static final int PLAYER_SETPLAYLIST    =  9;
    private static final int PLAYER_GETPLAYLIST    = 10;
    private static final int PLAYER_SETCONNECT     = 11;
    private static final int PLAYER_GETCONNECT     = 12;
    private static final int PLAYER_PLAY           = 13;
    private static final int PLAYER_PAUSE          = 14;
    private static final int PLAYER_RESUME         = 15;
    private static final int PLAYER_GETPLAYSTATUS  = 16;
    private static final int TITLE_GETSTATUS       = 17;

}
```

MsmAccessRight

```
/*
 * @(#)MsmAccessRight.java
 *
 * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
 *
 * version     1.0
 * author Christopher Lindblad
 *
 */

package COM.Sun.isg.smcjc;

/**
 * Access types, operations on access lists, and rights and
 * lists of access rights.
 * Access types (read, admin, control) are the access catagories
 * defined by the MSM server (see MSM doc for each request to
 * determine the access catagory of that request).  Access op's
 * are the operations that can be made to alter access rights of
 * a particular user.  An access right is the pairing of access
 * catagories with a particular user.  An access list is a
collection
 * of access rights for multiple users.
 */
public class MsmAccessRight {
    public String name;
    public int access;
    public int op;

    public MsmAccessRight(String name, int access, int op) {
     this.name = name;
     this.access = access;
     this.op = op;
    }

    MsmAccessRight(XdrBlock xdr) {
     name = xdr.xdrinString();
     access = xdr.xdrinInt();
     op = xdr.xdrinInt();
    }

    void xdrout(XdrBlock xdr) {
     xdr.xdroutString(name);
     xdr.xdroutInt(access);
     xdr.xdroutInt(op);
    }

    public String toString() {
     return MsmToString.accessRightToString(this);
    }

    public static final int ACCESS_NONE = 0;
    public static final int ACCESS_ADMIN = 1;
    public static final int ACCESS_READ = 2;
    public static final int ACCESS_CONTROL = 4;
    public static final int ACCESS_ALL = 7;

    public static final int OP_ADD = 0;
    public static final int OP_REMOVE = 1;
}
```

MsmPersistence

```
/*
 * @(#)MsmPersistence.java
 *
 * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
 *
 * version     1.0
 * author Christopher Lindblad
 *
 */

package COM.Sun.isg.smcjc;

/**
 * MsmPersistence information
 */
public class MsmPersistence {
    /**
     * Indicates the date at which the player should be
     * automatically deleted.  On terminateDate, play if in
progress, will
     * be stopped and the player deleted.  A terminateDate of
MSMTIME_MAXTIME
     * indicates the player should never be automatically
deleted.
     */
    public long terminateDate;

    public int type;

    public MsmPersistence(int type, long terminateDate) {
     this.type = type;
     this.terminateDate = terminateDate;
    }

    MsmPersistence(XdrBlock xdr) {
     type = xdr.xdrinInt();
     terminateDate = xdr.xdrinMsmTime();
    }

    void xdrout(XdrBlock xdr) {
     xdr.xdroutInt(type);
     xdr.xdroutMsmTime(terminateDate);
    }
```

```
        public String toString() {
         return MsmToString.persistenceToString(this);
        }

        /**
         * No persistence across server outage.
         */
        public static final int TYPE_NONE = 0;
        /**
         * Only public static state is preserved, play not is not
    restarted.
         */
        public static final int TYPE_PLAYLIST = 1;
        /**
         * Play is restarted after outage at last known playPosition.
         */
        public static final int TYPE_PLAYPOSITION = 2;
        /**
         * Play is restarted after outage as appropriate for current
    date.
         */
        public static final int TYPE_PLAYCURDATE = 3;
    }
```

MsmPlaylist

```
 /*
  * @(#)MsmPlaylist.java
  *
  * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
  *
  * version     1.0
  * author Christopher Lindblad
  *
  */

package COM.Sun.isg.smcjc;

 /**
  * MsmPlaylist positions are measured in seconds and nanoseconds,
titles on a
  * playlist may be scheduled to start at any non-negative
position. (In some
  * cases it may be convenient to base playlists positions at 0;
in other
  * cases it may be better to base them with the OS representation
of
  * time-of-day.) The playlist maintains a contiguous sequence of
titles and
  * "dead air". A schedule may be edited by replacing any
contiguous
  * sub-sequence of the schedule with another sequence.  It is
also possible
  * to change the starting position of the scheduled list of
titles.  Because
  * of mfs "admission delays", title start times may slip; msm
optionally
  * allows a title to be padded with dead air that can absorb the
slip, or on
  * a slip the same title or a later title can be marked to be
truncated or a
  * later title may be "joined-in-progress" to absorb the slip and
maintain
  * schedule correspondence with clock time.
  */
public class MsmPlaylist {
      /**
       * On Get, the current modification status stamp.  On Put,
modstamp on
       * which mods are based, if modification status has changed.
```

```
Mods are
     * aborted unless modstamp == MsmPlayer.TIME_CURRENT, in
which case mods
     * are always done.
     */
    public long modstamp;

    /**
     * On Get, the starting playlist position for the returned
playlist items
     * on Put, the playlist position where items are to be
replaced.
     */
    public long editStartPosition;

    /**
     * On Get, the total duration of the items returned.  On Put,
the duration
     * of the existing playlist that is to be replaced with new
items.
     *
     * NOTE: On Put, edit range specified by editStartPosition
for length
     * editDuration must lie entirely within existing playlist.
Use
     * MsmPlayer.getPlaylist() to get listStartPosition and
listDuration to
     * determine playlist bounds.
     */
    public long editDuration;

    /**
     * On Get, the startPosition for the entire playlist.  On
Put, the new
     * startPosition for the playlist after edits.
     */
    public long listStartPosition;

    /**
     * On Get, the duration of the entire list.  On Put, ignored.
     */
    public long listDuration;

    public MsmItem[] items;

    /**
     * On Get, the current loop state of the playlist.  On Put,
if TRUE, the
     * playlist wraps from end->start, start-end.
     */
    public boolean isLoop;

    public MsmPlaylist(long modstamp, boolean isLoop, long
editStartPosition,
                 long editDuration, MsmItem[] items,
                 long listStartPosition, long listDuration) {
     this.modstamp = modstamp;
     this.isLoop = isLoop;
     this.editStartPosition = editStartPosition;
     this.editDuration = editDuration;
     this.items = items;
     this.listStartPosition = listStartPosition;
     this.listDuration = listDuration;
    }

    MsmPlaylist(XdrBlock xdr) {
     modstamp = xdr.xdrinMsmTime();
     isLoop = xdr.xdrinBoolean();
     editStartPosition = xdr.xdrinMsmTime();
     editDuration = xdr.xdrinMsmTime();
     items = new MsmItem[xdr.xdrinInt()];
     for (int i = 0; i < items.length; i++) {
         int itemType = xdr.xdrinInt();
         switch (itemType) {
         case TITLE:
          items[i] = new MsmTitleItem(xdr);
          break;
         case DEADAIR:
          items[i] = new MsmDeadAirItem(xdr);
          break;
         }
     }
     listStartPosition = xdr.xdrinMsmTime();
     listDuration = xdr.xdrinMsmTime();
    }

    void xdrout(XdrBlock xdr) {
     xdr.xdroutMsmTime(modstamp);
     xdr.xdroutBoolean(isLoop);
     xdr.xdroutMsmTime(editStartPosition);
     xdr.xdroutMsmTime(editDuration);
     xdr.xdroutInt(items.length);
     for (int i = 0; i < items.length; i++) {
         if (items[i] instanceof MsmTitleItem) {
          xdr.xdroutInt(TITLE);
          ((MsmTitleItem)items[i]).xdrout(xdr);
         } else {
          xdr.xdroutInt(DEADAIR);
          ((MsmDeadAirItem)items[i]).xdrout(xdr);
         }
     }
     xdr.xdroutMsmTime(listStartPosition);
     xdr.xdroutMsmTime(listDuration);
    }

    public String toString() {
     return MsmToString.playlistToString(this);
    }

    private static final int TITLE   = 0;
    private static final int DEADAIR = 1;

}
```

MsmConnect

```
/*
 * @(#)MsmConnect.java
 *
 * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
 *
 * version     1.0
 * author Christopher Lindblad
 *
 */

package COM.Sun.isg.smcjc;

/**
 * Connection paramaters.
 * These parameters are passed directly to mfs_str_open().
 */

public class MsmConnect {
    /**
     * The transport independent address.
     **/
    public String destTiAddr;

    /**
     * The packet encapsulation specifier (eg. MPEG Transport, *
DSS, etc).
     */
    public String encap;

    /**
     * The bits/second network bandwidth to request.
     */
    public int rate;

    public MsmConnect(String destTiAddr, String encap, int rate)
{
     this.destTiAddr = destTiAddr;
     this.encap = encap;
     this.rate = rate;
    }

    MsmConnect(XdrBlock xdr) {
     destTiAddr = xdr.xdrinString();
     encap = xdr.xdrinString();
     rate = xdr.xdrinInt();
    }

    void xdrout(XdrBlock xdr) {
     xdr.xdroutString(destTiAddr);
     xdr.xdroutString(encap);
     xdr.xdroutInt(rate);
    }

    public String toString() {
     return MsmToString.connectToString(this);
    }

}
```

MsmPlayStatus

```
/*
 * @(#)MsmPlayStatus.java
 *
 * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
 *
 * version     1.0
 * author Christopher Lindblad
 *
 */

package COM.Sun.isg.smcjc;

/**
 * MsmPlayStatus indicates the current state of the player.
 * STATE_WAIT indicates that a play command has been given, but
 * that startDate has not arrived.
 */
public class MsmPlayStatus {
    public long pausePosition;
    public long currentDate;
    public long currentPosition;
    public String info;
    public int currentState;
    public int currentSpeed;
    public boolean pausePending;

    MsmPlayStatus(XdrBlock xdr) {
     info = xdr.xdrinString();
     pausePending = xdr.xdrinBoolean();
     pausePosition = xdr.xdrinMsmTime();
     currentState = xdr.xdrinInt();
     currentSpeed = xdr.xdrinInt();
     currentDate = xdr.xdrinMsmTime();
     currentPosition = xdr.xdrinMsmTime();
    }

    public String toString() {
     return MsmToString.playStatusToString(this);
    }

    public static final int STATE_STOP = 0;
    public static final int STATE_WAIT = 1;
    public static final int STATE_PLAY = 2;

    }
```

MsmToString

```
 /*
  * @(#)MsmToString.java
  *
  * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
  *
  * version     1.0
  * author Christopher Lindblad
  *
  */

 package COM.Sun.isg.smcjc;

 import java.util.*;

 class MsmToString {
     static String sessionToString(MsmSession se) {
      return "MsmSession"
          + "[serverHostName=" + se.getServerHostName()
          + "]";
     }

     static String playerToString(MsmPlayer pl) {
      byte[] h = pl.getHandle();
      StringBuffer sb = new StringBuffer(h.length*2);
      for (int i = 0; i < h.length; i++) {
          byte b = h[i];
          sb.append(Character.forDigit((b >> 4) & 0xf, 16));
          sb.append(Character.forDigit( b       & 0xf, 16));
      }
      return "MsmPlayer"
          + "[serverHostName=" +
pl.getSession().getServerHostName()
          + " handle=" + sb.toString()
          + "]";
     }

     private static final String[] rights =
 {"admin","read","control"};

     private static final String[] ops = {"add","remove"};

     static String accessRightToString(MsmAccessRight ar) {
      StringBuffer sb = new StringBuffer();
      for (int i = 0; i < rights.length; i++) {
         if ((ar.access & (1 << i)) != 0) {
          if (sb.length() > 0) sb.append("|");
          sb.append(rights[i]);
         }
     }
     if (sb.length() == 0) sb.append("none");
     String op;
     if (ar.op >= 0 && ar.op < ops.length) op = ops[ar.op];
     else op = String.valueOf(ar.op);
     return "MsmAccessRight"
         + "[name=" + ar.name
            + " access=" + sb.toString()
            + " op=" + op
         + "]";
    }

    static String connectToString(MsmConnect co) {
     return "MsmConnect"
         + "[destTiAddr=\"" + co.destTiAddr +"\""
         + " encap=\"" + co.encap +"\""
            + " rate=" + co.rate
         + "]";
    }

    static String deadAirItemToString(MsmDeadAirItem dai) {
     return "MsmDeadAirItem"
         + "[itemDuration=" + dai.itemDuration
            + " joinInDuration=" + dai.joinInDuration
         + "]";
    }

    private static final String[] types = {
     "none","playlist","playposition","playcurdate"};

    static String persistenceToString(MsmPersistence pe) {
     String type;
     if (pe.type >= 0 && pe.type < types.length) type =
types[pe.type];
     else type = String.valueOf(pe.type);
     return "MsmPersistence"
         + "[type=" + type
            + "
terminateDate=\""+dateToString(pe.terminateDate)+"\""
         + "]";
    }

    static String dateToString(long date) {
     if (date == MsmPlayer.TIME_MAXTIME) return "never";
     else return new Date(date/1000000L).toString();
    }

    private static final String[] states =
{"stop","wait","play"};

    private static final String[] speeds = {
     "scene_reverse","fastest_reverse","faster_reverse","fast_rev
erse",
     "reverse","slow_reverse","slower_reverse","slowest_reverse",
     "slowest_forward","slower_forward","slow_forward","forward",
     "fast_forward","faster_forward","fastest_forward","scene_for
ward"};

    static String playStatusToString(MsmPlayStatus ps) {
     String state;
     if (ps.currentState >= 0 && ps.currentState < states.length)
{
         state = states[ps.currentState];
     } else state = String.valueOf(ps.currentState);
     String speed;
     if (ps.currentSpeed >= 0 && ps.currentSpeed < speeds.length)
{
         speed = speeds[ps.currentSpeed];
     } else speed = String.valueOf(ps.currentSpeed);
     return "MsmPlayStatus"
         + "[info=\"" + ps.info +"\""
            + " pausePending=" + ps.pausePending
            + " pausePosition=" + ps.pausePosition
            + " currentState=" + state
            + " currentSpeed=" + speed
            + " currentDate=\"" + dateToString(ps.currentDate) +
"\""
            + " currentPosition=" + ps.currentPosition
         + "]";
    }

    static String playlistToString(MsmPlaylist pl) {
     StringBuffer sb = new StringBuffer();
     if (pl.items != null) {
         for (int i = 0; i < pl.items.length; i++) {
          if (i != 0) sb.append(",");
          sb.append(pl.items[i].toString());
         }
     }
     return "MsmPlaylist"
```

```
         + "[modstamp=\"" + dateToString(pl.modstamp) + "\""
         + " isLoop=" + pl.isLoop
         + " editStartPosition=" + pl.editStartPosition
         + " editDuration=" + pl.editDuration
         + " items=[" + sb.toString() + "]"
         + " listStartPosition=" + pl.listStartPosition
         + " listDuration=" + pl.listDuration
         + "]";
    }

    static String titleToString(MsmTitle ti) {
     StringBuffer sb = new StringBuffer();
     if (ti.speedScale != null) {
         for (int i = 0; i < ti.speedScale.length; i++) {
          if (i != 0) sb.append(",");
          sb.append(ti.speedScale[i]);
         }
     }
     return "MsmTitle"
         + "[name=\"" + ti.name + "\""
         + " speedScale=[" + sb.toString() + "]"
         + " maxBitRate=" + ti.maxBitRate
         + " totalPlayDuration=" + ti.totalPlayDuration
         + " format=\"" + ti.format + "\""
         + "]";
    }

    static String titleItemToString(MsmTitleItem ti) {
     return "MsmTitleItem"
         + "[titleName=\"" + ti.titleName + "\""
            + " itemDuration=" + ti.itemDuration
            + " startOffset=" + ti.startOffset
            + " playDuration=" + ti.playDuration
            + " joinInDuration=" + ti.joinInDuration
         + " isTimeLocked=" + ti.isTimeLocked
         + " playClosestSpeed=" + ti.playClosestSpeed
            + " maxBitRate=" + ti.maxBitRate
         + "]";
    }

}
```

MsmItem

```
/*
 * @(#)MsmItem.java
 *
 * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
 *
 * version     1.0
 * author Christopher Lindblad
 *
 */

package COM.Sun.isg.smcjc;

public abstract class MsmItem {
    /**
     * The number of milliseconds allocated to this item.
     */
    public long itemDuration;

    /**
     * Time of initial play that may be sacrificed to absorb
previous schedule
     * slips.   Silently limited to itemDuration.   If
TIME_CURRENT,
     * itemDuration is used.
     */
    public long joinInDuration;
}
```

MsmTitleItem

```
/*
 * @(#)MsmTitleItem.java
 *
 * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
 *
 * version     1.0
 * author Christopher Lindblad
 *
 */

package COM.Sun.isg.smcjc;

/*
 * A playlist title item.
 */
public class MsmTitleItem extends MsmItem {
    /**
     * The number of milliseconds into title where play should
begin.  It is
     * illegal for this to be greater than the total play time of
the title.
     */
    public long startOffset;

    /**
     * The number of milliseconds of title to play within this
item.
     * Values less than itemDuration allow some pad for absorbing
admission
     * delays (and the play truncation that would occur), but
should admission
     * delay be zero, dead air would occur for the remainder of
the item. It
     * is illegal for playDuration to be greater than
itemDuration or for
     * playDuration + startOffset to be greater than the total
play time of
     * the title. If TIME_CURRENT, the min of itemDuration and
total play time
     * minus startOffset is used.
     */
    public long playDuration;

    /**
```

```
     * The file pathname for title.
     */
    public String titleName;

    /**
     * Ignored on MsmPlayer.setPlaylist.  Returns max bit rate of
title on
     * MsmPlayer.getPlaylist.
     */
    public int maxBitRate;

    /**
     * If true, terminate play after itemDuration seconds (even
if admission
     * delays have caused schedule to slip and title has not
completed).  If
     * false, always play itemDuration seconds of title, allow
schedule to
     * slip if necessary.
     */
    public boolean isTimeLocked;

    /**
     * If true, plays closest available speed in same direction
if requested
     * speed is not available.  Search for closest is proceeds
towards normal
     * presentation rate.  Play is skipped if normal presentation
rate in
     * direction is not available.  If false, play of title is
skipped if
     * appropriate speed is not available.
     */
    public boolean playClosestSpeed;

    public MsmTitleItem(String titleName, long itemDuration, long
startOffset,
                long playDuration, long joinInDuration,
                boolean isTimeLocked, boolean playClosestSpeed,
                int maxBitRate) {
      this.titleName = titleName;
      this.itemDuration = itemDuration;
      this.startOffset = startOffset;
      this.playDuration = playDuration;
      this.joinInDuration = joinInDuration;
      this.isTimeLocked = isTimeLocked;
      this.playClosestSpeed = playClosestSpeed;
```

```
  this.maxBitRate = maxBitRate;
 }

 MsmTitleItem(XdrBlock xdr) {
  titleName = xdr.xdrinString();
  itemDuration = xdr.xdrinMsmTime();
  startOffset = xdr.xdrinMsmTime();
  playDuration = xdr.xdrinMsmTime();
  joinInDuration = xdr.xdrinMsmTime();
  isTimeLocked = xdr.xdrinBoolean();
  playClosestSpeed = xdr.xdrinBoolean();
  maxBitRate = xdr.xdrinInt();
 }

 void xdrout(XdrBlock xdr) {
  xdr.xdroutString(titleName);
  xdr.xdroutMsmTime(itemDuration);
  xdr.xdroutMsmTime(startOffset);
  xdr.xdroutMsmTime(playDuration);
  xdr.xdroutMsmTime(joinInDuration);
  xdr.xdroutBoolean(isTimeLocked);
  xdr.xdroutBoolean(playClosestSpeed);
  xdr.xdroutInt(maxBitRate);
 }

 public String toString() {
  return MsmToString.titleItemToString(this);
 }

}
```

MsmDeadAirItem

```
/*
 * @(#)MsmDeadAirItem.java
 *
 * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
 *
 * version     1.0
 * author Christopher Lindblad
 *
 */

package COM.Sun.isg.smcjc;

public class MsmDeadAirItem extends MsmItem {
    public MsmDeadAirItem(long itemDuration, long joinInDuration)
{
     this.itemDuration = itemDuration;
     this.joinInDuration = joinInDuration;
    }

    MsmDeadAirItem(XdrBlock xdr) {
     itemDuration = xdr.xdrinMsmTime();
     joinInDuration = xdr.xdrinMsmTime();
    }

    void xdrout(XdrBlock xdr) {
     xdr.xdroutMsmTime(itemDuration);
     xdr.xdroutMsmTime(joinInDuration);
    }

    public String toString() {
     return MsmToString.deadAirItemToString(this);
    }

}
```

MsmException

```
/*
 * @(#)MsmException.java
 *
 * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
 *
 * version     1.0
 * author Christopher Lindblad
 *
 */

package COM.Sun.isg.smcjc;

import java.io.*;

/**
 * Signals that an Media Stream Manager exception has occurred.
 */
public class MsmException extends IOException {
    /**
     * Constructs an MsmException with no detail message.
     * A detail message is a String that describes this
particular exception.
     */
    MsmException() {
     super();
    }

    /**
     * Constructs an MsmException with the specified detail
message.
     * A detail message is a String that describes this
particular exception.
     * @param s the detail message
     */
    MsmException(String s) {
     super(s);
    }

    MsmException(int proc, String msg) {
     super(((proc >= 0 && proc < procNames.length) ?
           procNames[proc] : Integer.toString(proc))
          + ": " +
          msg);
    }
```

```
MsmException(int proc, int err) {
 super(((proc >= 0 && proc < procNames.length) ?
        procNames[proc] : Integer.toString(proc))
       + ": " +
       ((err >= 0 && err < errNames.length) ?
        errNames[err] : Integer.toString(err)));
}

private static final String[] procNames = {
 "null",
 "server authtype",
 "player create",
 "player delete",
 "player list",
 "player access set",
 "player access get",
 "player persistence set",
 "player persistence get",
 "player playlist set",
 "player playlist get",
 "player connect set",
 "player connect get",
 "player play",
 "player pause",
 "player resume",
 "player play status",
 "title status",
};

private static final String[] errNames = {
 "success",              /*  0 */
 "failed",               /*  1 */
 "badarg",               /*  2 */
 "no mem",               /*  3 */
 "no netname",           /*  4 */
 "des auth failed",      /*  5 */
 "kerb auth failed",     /*  6 */
 "no such player",       /*  7 */
 "old modstamp",         /*  8 */
 "item overlap",         /*  9 */
 "bad speed",            /* 10 */
 "bad start date",       /* 11 */
 "not connected",        /* 12 */
 "bad pause position",   /* 13 */
 "play active",          /* 14 */
 "bad file name",    /* 15 */
 "bad mfs file",         /* 16 */
```

```
     "bad file type",    /* 17 */
     "info too long",    /* 18 */
     "auth failed",      /* 19 */
     "bad position",          /* 20 */
     "kerberos unsupported",  /* 21 */
     "bad credentials",  /* 22 */
     "insufficient authorization", /* 23 */
     "bad access op",    /* 24 */
     "bad access type",  /* 25 */
     "bad persist type", /* 26 */
     "bad time arg",          /* 27 */
     "bad start position",    /* 28 */
     "bad duration",          /* 29 */
     "bad start offset", /* 30 */
     "bad edit start pos",    /* 31 */
     "bad edit duration",     /* 32 */
     "bad list start pos",    /* 33 */
     "bad item duration",     /* 34 */
     "bad join in duration",  /* 35 */
     "bad play duration",     /* 36 */
     "bad item type",    /* 37 */
     "bad title type",   /* 38 */
     "no such file",          /* 39 */
     "bad lut file",          /* 40 */
     "bad mfs fs",       /* 41 */
     "toc syntax",       /* 42 */
     "toc eof",          /* 43 */
     "toc bad char",          /* 44 */
     "no normal speed",  /* 45 */
     "dup speeds",       /* 46 */
     "bad file len",          /* 47 */
     "toc incomplete",   /* 48 */
     "toc can't map",    /* 49 */
     "toc bad filesize", /* 50 */
     "toc bad index",    /* 51 */
     "too low connect rate",  /* 52 */
    };

}
```

XdrBlock

```
/*
 * @(#)XdrBlock.java
 *
 * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
 *
 * version     1.0
 * author Christopher Lindblad
 *
 */

package COM.Sun.isg.smcjc;

import java.io.*;
import java.net.*;

/**
 * Used to manipulate ONC RPC calls and replies.
 */
class XdrBlock {
    byte[] buf;
    int ptr;

    /*
     * Create a new empty block.
     * @param size The size of the block.
     */
    public XdrBlock(int size) {
     buf = new byte[size];
    }

    /*
     * Create a new empty block.
     */
    public XdrBlock() {
     this(256);
    }

    /*
     * Create a new block and initialize it with a call header.
     * @param prog The RPC program number.
     * @param vers The RPC version number.
     * @param proc The RPC procedure number.
     * @return The xid generated.
     */
```

```
public XdrBlock(int prog, int vers, int proc) {
 this();
 xdroutCallHeader(prog, vers, proc);
}

/**
 * Create a new block and receive it from an InputStream.
 * @param is The InputStream from which to receive the block.
 * @exception IOException If an IO error has occurred.
 */
public XdrBlock(InputStream is) throws IOException {
 synchronized (is) {
     int hdr;
     do {
      hdr  = readByte(is) << 24;
      hdr |= readByte(is) << 16;
      hdr |= readByte(is) <<  8;
      hdr |= readByte(is)      ;
      int start;
      int count = hdr & 0x7fffffff;
      if (buf == null) {
          start = 0;
          buf = new byte[count];
      } else {
          start = buf.length;
          byte[] tmp = new byte[start + count];
          System.arraycopy(buf, 0, tmp, 0, start);
          buf = tmp;
      }
      while (count > 0) {
          int done = is.read(buf, start, count);
          if (done < 0) throw new IOException("end of file");
          start += done;
          count -= done;
      }
     } while ((hdr & 0x80000000) == 0);
 }
}

private int readByte(InputStream is) throws IOException {
 int result = is.read();
 if (result < 0) throw new IOException("end of file");
 return result;
}

/**
 * Send the block to an output stream.
```

```
     * @param is The OutputStream ro which to send the block.
     * @exception IOException If an IO error has occurred.
     */
    public synchronized void send(OutputStream os) throws
IOException {
     int hdr = ptr | 0x80000000;
     synchronized (os) {
         os.write((hdr >> 24) & 0xff);
         os.write((hdr >> 16) & 0xff);
         os.write((hdr >>  8) & 0xff);
         os.write((hdr      ) & 0xff);
         os.write(buf, 0, ptr);
         if (os instanceof BufferedOutputStream) {
          ((BufferedOutputStream)os).flush();
         }
     }
    }

    /**
     * Input a fixed-length array of bytes from the block.
     * @param len The lenght of the array.
     * @return The byte array.
     */
    public synchronized byte[] xdrinBytes(int len) {
     byte[] result = new byte[len];
     System.arraycopy(buf, ptr, result, 0, len);
     ptr = (ptr + len + 3) & -4;
     return result;
    }

    /**
     * Input a variable-length array of bytes from the block.
     * @return The byte array.
     */
    public synchronized byte[] xdrinBytes() {
     return xdrinBytes(xdrinInt());
    }

    /**
     * Input an int from the block.
     * @return The int.
     */
    public synchronized int xdrinInt() {
     int result;
     result  = (buf[ptr    ] & 0xff) << 24;
     result |= (buf[ptr + 1] & 0xff) << 16;
     result |= (buf[ptr + 2] & 0xff) <<  8;
```

```
  result |= (buf[ptr + 3] & 0xff);
  ptr += 4;
  return result;
 }

 /**
  * Input an boolean from the block.
  * @return The boolean.
  */
 public boolean xdrinBoolean() {
  return xdrinInt() != 0;
 }

 /**
  * Input a String from the block.
  * @return The String.
  */
 public String xdrinString() {
  return new String(xdrinBytes(), 0);
 }

 /**
  * Input a Media Stream Manager Time value
  */
 public synchronized long xdrinMsmTime() {
  long sec = xdrinInt();
  long nsec = xdrinInt();
  if (sec == nsec && sec < 0) return sec;
  return sec*1000000000L + nsec;
 }

 /**
  * Output a fixed-length array of bytes to the block.
  * @param val The array to output.
  * @param len The length of the array to output.
  */
 public synchronized void xdroutBytes(byte[] val, int len) {
  int nxt = (ptr + len + 3) & -4;
  if (nxt > buf.length) grow(nxt);
  System.arraycopy(val, 0, buf, ptr, len);
  ptr = nxt;
 }

 /**
  * Output a variable-length array of bytes to the block.
  * @param val The array to output.
  */
```

```
public synchronized void xdroutBytes(byte[] val) {
 int len = val.length;
 xdroutInt(len);
 xdroutBytes(val, len);
}

/**
 * Output an int to the block.
 * @param val The int to output.
 */
public synchronized void xdroutInt(int val) {
 int nxt = ptr + 4;
 if (nxt > buf.length) grow(nxt);
 buf[ptr    ] = (byte)((val >> 24) & 0xff);
 buf[ptr + 1] = (byte)((val >> 16) & 0xff);
 buf[ptr + 2] = (byte)((val >>  8) & 0xff);
 buf[ptr + 3] = (byte)((val      ) & 0xff);
 ptr = nxt;
}

/**
 * Output an boolean to the block.
 * @param val The boolean to output.
 */
public void xdroutBoolean(boolean val) {
 xdroutInt(val? 1:0);
}

/**
 * Output a String to the block.
 * @param val The String to output.
 */
public void xdroutString(String val) {
 int len = val.length();
 byte[] tmp = new byte[len];
 val.getBytes(0, len, tmp, 0);
 xdroutBytes(tmp);
}

/**
 * Output a Media Stream Manager Time value
 * @param val The time to output.
 */
public synchronized void xdroutMsmTime(long val) {
 if (val < 0) {
     xdroutInt((int)val);
     xdroutInt((int)val);
```

```
        } else {
            xdroutInt((int)(val/1000000000L));
            xdroutInt((int)(val%1000000000L));
        }
       }

       private void grow(int needed) {
        int len = buf.length*2;
        while (len < needed) len *= 2;
        byte[] tmp = new byte[len];
        System.arraycopy(buf, 0, tmp, 0, buf.length);
        buf = tmp;
       }

       /**
        * Output a RPC Call header to the block.
        * @param prog The RPC program number.
        * @param vers The RPC version number.
        * @param proc The RPC procedure number.
        */
       public synchronized void xdroutCallHeader(int prog, int vers,
   int proc) {
        xdroutInt(genXid());
        xdroutInt(CALL);
        xdroutInt(RPCVERS);
        xdroutInt(prog);
        xdroutInt(vers);
        xdroutInt(proc);
        xdroutInt(AUTH_UNIX);
        xdroutBytes(cred());
        xdroutInt(AUTH_NULL);
        xdroutBytes(verf());
       }

       public synchronized int callXid() {
        int tmp = ptr;
        ptr = 0;
        int result = xdrinInt();
        ptr = tmp;
        return result;
       }

       public synchronized int callProc() {
        int tmp = ptr;
        ptr = 20;
        int result = xdrinInt();
        ptr = tmp;
```

```
     return result;
    }

    private static int lastXid = 0;

    private synchronized static int genXid() {
     if (lastXid != 0) lastXid += 1;
     else lastXid = (int)(Math.random() * 2147483648.0D);
     return lastXid;
    }

    private static byte[] lastCred;

    private synchronized static byte[] cred() {
     if (lastCred == null) {
        XdrBlock xdr = new XdrBlock();
        xdr.xdroutInt((int)(System.currentTimeMillis()/1000L));
        String host;
        try host = InetAddress.getLocalHost().getHostName();
        catch (UnknownHostException e) host = "???";
        xdr.xdroutString(host);
        int uid;
        try uid =
Integer.parseInt(System.getProperty("user.uid"));
        catch (NumberFormatException e) uid = 0;
        xdr.xdroutInt(uid);
        int gid;
        try gid =
Integer.parseInt(System.getProperty("user.gid"));
        catch (NumberFormatException e) gid = 0;
        xdr.xdroutInt(gid);
        xdr.xdroutInt(0);    // no gids
        lastCred = new byte[xdr.ptr];
        System.arraycopy(xdr.buf, 0, lastCred, 0, xdr.ptr);
     }
     return lastCred;
    }

    private static byte[] lastVerf;

    private synchronized static byte[] verf() {
     if (lastVerf == null) {
        lastVerf = new byte[0];
     }
     return lastVerf;
    }
    /**
     * Input a RPC reply header from the block.
     * @param xid The expected xid.
     * @exception IOException If an error has occurred.
     */
    public synchronized void xdrinReplyHeader(int xid) throws
IOException {
     int replyXid = xdrinInt();
     if (replyXid != xid) {
         throw new IOException(
          "rpc xid mismatch: " +
          "expected " + xid + " but got " + replyXid);
     }
     int msgType = xdrinInt();
     if (msgType != REPLY) {
         throw new IOException(
          "rpc msg type mismatch: " +
          " expected " + REPLY + " but got " + msgType);
     }
     int replyStat = xdrinInt();
     switch (replyStat) {
     case MSG_ACCEPTED:
         int verfType = xdrinInt();
         byte[] verf = xdrinBytes();
         int acceptStat = xdrinInt();
         switch (acceptStat) {
            case SUCCESS:
          return;
         case PROG_UNAVAIL:
          throw new IOException(
              "rpc accepted: " +
              "remote hasn't exported program");
         case PROG_MISMATCH:
          int low = xdrinInt();
          int high = xdrinInt();
          throw new IOException(
              "rpc accepted: " +
              "version mismatch low=" + low + " high=" + high);
         case PROC_UNAVAIL:
          throw new IOException(
              "rpc accepted: " +
              "program can't support procedure");
         case GARBAGE_ARGS:
          throw new IOException(
              "rpc accepted: " +
              "procedure can't decode params");
         default:
```

```
         throw new IOException(
             "rpc accepted: " +
             "unknown status: " + acceptStat);
         }
    case MSG_DENIED:
        int rejectStat = xdrinInt();
        switch (rejectStat) {
        case RPC_MISMATCH:
         int low = xdrinInt();
         int high = xdrinInt();
         throw new IOException(
             "rpc rejected: " +
             "version mismatch low=" + low + " high=" + high);
        case AUTH_ERROR:
         int authStat = xdrinInt();
         switch (authStat) {
         case AUTH_BADCRED:
             throw new IOException(
              "rpc rejected: " +
              "remote can't authenticate caller: " +
              "bad credentials (seal broken)");
         case AUTH_REJECTEDCRED:
             throw new IOException(
              "rpc rejected: " +
              "remote can't authenticate caller: " +
              "client must begin new session");
         case AUTH_BADVERF:
             throw new IOException(
              "rpc rejected: " +
              "remote can't authenticate caller: " +
              "bad verifier (seal broken)");
         case AUTH_REJECTEDVERF:
             throw new IOException(
              "rpc rejected: " +
              "remote can't authenticate caller: " +
              "verifier expired or replayed");
         case AUTH_TOOWEAK:
             throw new IOException(
              "rpc rejected: " +
              "remote can't authenticate caller: " +
              "rejected for security reasons");
         default:
             throw new IOException(
              "rpc rejected: " +
              "remote can't authenticate caller: " +
              "unknown status: " + authStat);
         }
```

```
          default:
           throw new IOException(
               "rpc rejected: " +
               "unknown status: " + rejectStat);
          }
      default:
          throw new IOException("unknown rpc reply status: " +
replyStat);
      }
     }

     /*
      * Blow up if ptr hasn't reached the end of the block.
      */
     public void done() throws IOException {
      if (ptr != buf.length) {
          throw new IOException(
           (buf.length-ptr) + " extra bytes of data remaining in
reply");
      }
     }

     /*
      * Provisions for authentication of caller to service and
vice-versa are
      * provided as a part of the RPC protocol.  The call message
has two
      * authentication fields, the credentials and verifier.  The
reply
      * message has one authentication field, the response
verifier.  The RPC
      * protocol specification defines all three fields to be the
following
      * opaque type (in the eXternal Data Representation (XDR)
language [9]):
      */
    private static final int AUTH_NULL      = 0;
    private static final int AUTH_UNIX      = 1;
    private static final int AUTH_SHORT     = 2;
    private static final int AUTH_DES       = 3;

    /*
     * RPC Message protocol version 2
     */
    private static final int RPCVERS = 2;
    private static final int CALL    = 0;
    private static final int REPLY   = 1;
    /*
     * A reply to a call message can take on two forms: The
message was
     * either accepted or rejected.
     */
    private static final int MSG_ACCEPTED = 0;
    private static final int MSG_DENIED   = 1;

    /*
     * Given that a call message was accepted, the following is
the status
     * of an attempt to call a remote procedure.
     */
    private static final int SUCCESS       = 0;
    private static final int PROG_UNAVAIL  = 1;
    private static final int PROG_MISMATCH = 2;
    private static final int PROC_UNAVAIL  = 3;
    private static final int GARBAGE_ARGS  = 4;

    /*
     * Reasons why a call message was rejected:
     */
    private static final int RPC_MISMATCH = 0;
    private static final int AUTH_ERROR = 1;

    /*
     * Why authentication failed:
     */
    private static final int AUTH_BADCRED      = 1;
    private static final int AUTH_REJECTEDCRED = 2;
    private static final int AUTH_BADVERF      = 3;
    private static final int AUTH_REJECTEDVERF = 4;
    private static final int AUTH_TOOWEAK      = 5;

}
```

PortMapper

```
/*
 * @(#)PortMapper.java
 *
 * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
 *
 * version     1.0
 * author Christopher Lindblad
 *
 */

package COM.Sun.isg.smcjc;

import java.io.*;
import java.net.*;

/**
 * Interface to the ONC port mapper.
 */
class PortMapper {
    private Socket socket;
    private InputStream is;
    private OutputStream os;

    /**
     * Create a port mapper client.
     * @param host The server for which we want to know the port
mappings.
     * @exception IOException If there is an error.
     */
    public PortMapper(String host) throws IOException {
     socket = new Socket(host, PMAP_PORT);
     is = new BufferedInputStream(socket.getInputStream());
     os = new BufferedOutputStream(socket.getOutputStream());
    }

    /**
     * Get the port number for a particular ONC service.
     * @param prog The RPC program number.
     * @param vers The RPC version number.
     * @param prot Either IPPROTO_TCP or IPPROTO_UDP.
     * @return The port number for the service.
     * @exception IOException If there is an error.
     */
    public synchronized int getPort(int prog, int vers, int prot)
```

```
        throws IOException {
            XdrBlock call = new XdrBlock();
            call.xdroutCallHeader(PMAP_PROG, PMAP_VERS,
    PMAPPROC_GETPORT);
            call.xdroutInt(prog);
            call.xdroutInt(vers);
            call.xdroutInt(prot);
            call.xdroutInt(0);
            call.send(os);
            XdrBlock reply = new XdrBlock(is);
            reply.xdrinReplyHeader(call.callXid());
            int result = reply.xdrinInt();
            reply.done();
            return result;
        }

       /**
        * Closes the port mapper.
        */
       public synchronized void close() throws IOException {
        socket.close();
       }

       static final int IPPROTO_TCP = 6;
       static final int IPPROTO_UDP = 17;

       private static final int PMAP_PROG = 100000;
       private static final int PMAP_VERS = 2;
       private static final int PMAP_PORT = 111;

       private static final int PMAPPROC_NULL    = 0;
       private static final int PMAPPROC_SET     = 1;
       private static final int PMAPPROC_UNSET   = 2;
       private static final int PMAPPROC_GETPORT = 3;
       private static final int PMAPPROC_DUMP    = 4;
       private static final int PMAPPROC_CALLIT  = 5;

   }
```

Decoder

```
/*
 * @(#)Decoder.java
 *
 * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
 *
 * version     1.0
 * author Christopher Lindblad
 *
 */

package COM.Sun.isg.smcjc;

import java.awt.*;
import java.io.*;

public class Decoder extends Panel {
    private DecoderImpl impl;

    public Decoder() {
     setLayout(new BorderLayout());
    }

    public synchronized void init(String format, Image img,String
host,int port,String ATM)
      throws IOException {
           try {
            Class implClass = Class.forName(implClassName(format));
            if (impl == null || impl.getClass() != implClass) {
                removeAll();
                impl = (DecoderImpl)implClass.newInstance();
                add("Center", impl);
            }
            impl.init(format, img, host, port,ATM);
           } catch (ClassNotFoundException e) {
            throw new IOException(e.toString());
           } catch (IllegalAccessException e) {
            throw new IOException(e.toString());
           } catch (InstantiationException e) {
            throw new IOException(e.toString());
           }
      }

    public synchronized void paint(Graphics g) {
      if (impl != null) super.paint(g);
```

```
       else {
           Rectangle b = bounds();
           g.setColor(getBackground());
           g.fill3DRect(0, 0, b.width, b.height, true);
       }
      }

      public synchronized void stop() throws IOException {
       if (impl != null) impl.stop();
      }

      public synchronized void pause() throws IOException {
       if (impl != null) impl.pause();
      }

      public synchronized void play() throws IOException {
       if (impl != null) impl.play();
      }

      public synchronized void flush() throws IOException {
       if (impl != null) impl.flush();
      }

      public synchronized String destTiAddr() throws IOException {
       if (impl != null) return impl.destTiAddr();
       return "";
      }

      public synchronized String encap() throws IOException {
       if (impl != null) return impl.encap();
       return "";
      }

      /**
       * A hacky implementation factory
       */
      private static String implClassName(String format) throws
IOException {
       String osArch = System.getProperty("os.arch", "?os.arch");
       String osName = System.getProperty("os.name", "?os.name");
       String osVersion = System.getProperty("os.version",
"?os.version");
       String spec = format + " " + osArch + " " + osName + " " +
osVersion;
       if (format.equals("MPEG1SYS")) {
           if (osName.equals("Solaris") || osName.equals("SunOS"))
{
```

```
            if (osArch.equals("sparc")) {
                return "COM.Sun.isg.smcjc.MpxDecoderImpl";
            }
          }
      }
      throw new IOException("no decoder for " + spec);
    }
}
```

DecoderImpl

```
/*
 * @(#)DecoderImpl.java
 *
 * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
 *
 * version     1.0
 * author Christopher Lindblad
 *
 */

package COM.Sun.isg.smcjc;

import java.awt.*;
import java.io.*;

abstract class DecoderImpl extends Canvas {
    public abstract void init(String format, Image img, String
host, int port,String ATM) throws IOException;
    public abstract void stop() throws IOException;
    public abstract void pause() throws IOException;
    public abstract void play() throws IOException;
    public abstract void flush() throws IOException;
    public abstract String destTiAddr() throws IOException;
    public abstract String encap() throws IOException;
}
```

MpxDecoderImpl

```
/*
 * @(#)MpxDecoderImpl.java
 *
 * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
 *
 * version     1.0
 * author Christopher Lindblad
 *
 */

package COM.Sun.isg.smcjc;

import java.applet.*;
import java.io.*;
import java.awt.*;
import java.net.*;

class MpxDecoderImpl extends DecoderImpl implements Runnable {
    private String format;
    private String host;
    private int port;
    private int port0;
    private Image img;
    private long fadeTimeMillis;
    private DatagramSocket ctrlSckt;
    private Thread thread;
    private DatagramPacket ctrlPckt;
    private File logFile;
    private float luminance = 1.0F;
    private int dataPort;
    private int scale = 1;
    private int state=STOP;
    private boolean multi=false;
    private boolean ATM=false;
    private String ATMs=null;

    public MpxDecoderImpl() {
     super();
    }

    public synchronized void init(String format, Image img,
String host, int port,String ATMs)
     throws IOException {
         this.format = format;
```

```
          this.img = img;
          ATM=(ATMs!=null);
             this.port=port;
             this.host=host;
             if ((port==-1)&&(!ATM)){
              dataPort = genLocalPort();
             }else{
              dataPort = port;
              port0= genLocalPort();
                 multi=!ATM;
                 if (ATM) this.ATMs = ATMs;
             }
          ctrlPckt = new DatagramPacket(
           new
byte[128],128,InetAddress.getLocalHost(),genLocalPort());
          ctrlWord(0, 0x00000001); // sync
          ctrlWord(1, 0x00000002); // sync
          ctrlWord(2, 0x00000003); // sync
          ctrlWord(3, 0x00000004); // sync
          ctrlWord(4, 0xaaaa0001); // version = 1
          ctrlWord(5, 0xbbbb0001); // channel = 1
          ctrlWord(6, 0x00000000); // sequence = 0
          ctrlWord(7,     0xcccc0000); // flags = 0
          ctrlWord(8,     0xdddd0001); // type = 1
     }

    public Dimension minimumSize() {
     return new Dimension(WIDTH, HEIGHT);
    }

    public synchronized Dimension preferredSize() {
     Dimension dim = new Dimension(WIDTH*scale, HEIGHT*scale);
     return dim;
    }

    public synchronized void layout() {
     Rectangle b = bounds();
     double xscale = (double)b.width/(double)WIDTH;
     double yscale = (double)b.height/(double)HEIGHT;
     int scale = (int)((xscale + yscale) / 2.0 + 0.25);
     if (scale < 1) scale = 1;
     if (scale > 3) scale = 3;
     if (scale != this.scale) {
         this.scale = scale;
         if (state == PAUSE || state == PLAY) updateVideoMode();
     }
    }
    public synchronized void paint(Graphics g) {
     Dimension ps = preferredSize();
     g.setColor(getBackground());
     g.fill3DRect(0, 0, ps.width, ps.height, true);
     if (img != null) g.drawImage(img, 0, 0, ps.width, ps.height,
this);
    }

    public synchronized void stop() throws IOException {
     if (state == PAUSE || state == PLAY) {

            if (multi||ATM){
              StringBuffer sc= new StringBuffer();
              sc.append("kloop ");
System.out.println(sc.toString());
              String[] cmdarray0= new String[3];
              cmdarray0[0] = "/bin/sh";
              cmdarray0[1] = "-c";
              cmdarray0[2] = sc.toString();
              try Runtime.getRuntime().exec(cmdarray0);
              catch (SecurityException e)
System.out.println("Exec="+exec(cmdarray0[2]));
            }
         ctrlWord(9,     MCMD_EXIT);
         ctrlSckt.send(ctrlPckt);
         ctrlSckt.close();
         ctrlSckt = null;
         state = STOP;
         try {
          if (logFile.length() == 0) logFile.delete();
         } catch (SecurityException e) {
          String cmd = "/bin/rm -f "+logFile.getPath();
          try Runtime.getRuntime().exec(cmd);
          catch (SecurityException f) exec(cmd);
         }
     }
    }

    public synchronized void pause() throws IOException {
     if (state == PLAY) {
         ctrlWord(9,      MCMD_PLAYCTR); // identifier
         ctrlWord(10, PC_PAUSE); // action
         ctrlWord(11, Float.floatToIntBits(1.0F)); // speed
         ctrlSckt.send(ctrlPckt);
         state = PAUSE;
     }
    }

    public synchronized void play() throws IOException {
     if (state == PAUSE) {
         ctrlWord(9,      MCMD_PLAYCTR); // identifier
         ctrlWord(10, PC_PLAY);     // action
         ctrlWord(11, Float.floatToIntBits(1.0F)); // speed
         ctrlSckt.send(ctrlPckt);
         state = PLAY;
     } else if (state == STOP) {
         StringBuffer sb = new StringBuffer();
         sb.append("exec mpx");
         if (!multi) {
               if (!ATM){
                 sb.append(" -fn udp,lp,");
              sb.append(dataPort);
               }else{
                 sb.append(" -fn udp,lp,");
              sb.append(port0);
               }
            }else{
               sb.append(" -fn udp,lp,");
            sb.append(port0);
            }
         sb.append(" -xn udp,lp,");
         sb.append(ctrlPckt.getPort());
         sb.append(" -u 2");
         sb.append(" -v ");
         int depth = getColorModel().getPixelSize();
         if (depth == 1) {
          sb.append("mono");
         } else {
          sb.append("col");
          sb.append(depth);
          if (depth == 24 && scale > 1) sb.append("B");
         }
         sb.append(",");
         sb.append(scale);
         sb.append(" -w ");
         sb.append(windowId());
         sb.append(" </dev/null");
         sb.append(" >");
System.out.println(sb.toString());
         logFile = new
File("/tmp/mpx."+System.currentTimeMillis());
         sb.append(logFile.getPath());
         sb.append(" 2>&1");
         String[] cmdarray = new String[3];
         cmdarray[0] = "/bin/sh";
         cmdarray[1] = "-c";
         cmdarray[2] = sb.toString();
         try Runtime.getRuntime().exec(cmdarray);
         catch (SecurityException e) exec(cmdarray[2]);
         ctrlSckt = new DatagramSocket();
         state = PLAY;
            if(ATM){
              StringBuffer sc= new StringBuffer();
              sc.append("loop a ");
              sc.append(dataPort+" ");
              sc.append(port0+" >sasa &");
System.out.println(sc.toString());
           String[] cmdarray0= new String[3];
           cmdarray0[0] = "/bin/sh";
           cmdarray0[1] = "-c";
           cmdarray0[2] = sc.toString();
           try Runtime.getRuntime().exec(cmdarray0);
           catch (SecurityException e)
System.out.println("Exec="+exec(cmdarray0[2]));
            }else if (multi) {
              StringBuffer sc= new StringBuffer();
              sc.append("loop m ");
              sc.append(host+" ");
              sc.append(dataPort+" ");
              sc.append(port0+" &");
System.out.println(sc.toString());
           String[] cmdarray0= new String[3];
           cmdarray0[0] = "/bin/sh";
           cmdarray0[1] = "-c";
           cmdarray0[2] = sc.toString();
           try Runtime.getRuntime().exec(cmdarray0);
           catch (SecurityException e)
System.out.println("Exec="+exec(cmdarray0[2]));
            }
     }
    }

    public synchronized void flush() {
     if (thread == null) {
         thread = new Thread(this);
         thread.start();
     }
     fadeTimeMillis = System.currentTimeMillis() + 4000;
    }
    public synchronized String destTiAddr() throws
UnknownHostException {
        String phost;
     //return "be0,"+phost+","+dataPort;
     if (ATM){
          return "port=" + ATMs + ",vc=" + dataPort;
        }else {
       phost = InetAddress.getLocalHost().getHostName();
          return "host=" + phost + ",udpport=" + dataPort;
        }
    }

    public String encap() {
     return "MPEG1SYS";
    }

    private void ctrlWord(int idx, int val) {
     byte[] buf = ctrlPckt.getData();
     buf[idx*4    ] = (byte)((val >> 24) & 0xff);
     buf[idx*4 + 1] = (byte)((val >> 16) & 0xff);
     buf[idx*4 + 2] = (byte)((val >>  8) & 0xff);
     buf[idx*4 + 3] = (byte)((val      ) & 0xff);
    }

    private void updateVideoMode() {
     ctrlWord(9,  MCMD_PRESCTR); // identifier
     ctrlWord(10, PCTR_VMD|PCTR_LUM); // which
     int depth = getColorModel().getPixelSize();
     int col = (depth==1)? 0 : (depth==24&&scale>1) ? VDM_COLB :
VDM_COL;
     ctrlWord(11, (col<<8)|scale); // video mode
     ctrlWord(12, 0);             // audio mode
     ctrlWord(13, 0);             // audio_volume
     ctrlWord(14, Float.floatToIntBits(luminance)); // luminance
     ctrlWord(15, 0);             // saturation
     ctrlWord(16, 0);             // gamma
     try ctrlSckt.send(ctrlPckt); catch (IOException e);
    }

    public synchronized void run() {
     Thread currentThread = Thread.currentThread();
     try {
         while (currentThread==thread && (state==PAUSE ||
state==PLAY)) {
          long currentTimeMillis = System.currentTimeMillis();
          float last = luminance;
          if (fadeTimeMillis < currentTimeMillis) {
```

```
          if (luminance < 1.0F) luminance += 0.125F;
      } else {
          if (luminance > 0.0F) luminance -= 0.125F;
      }
      if (luminance != last) updateVideoMode();
      if (luminance >= 1.0F) return;
      try wait(125); catch (InterruptedException e);
     }
  } finally {
     if (thread == currentThread) thread = null;
  }
}

private int genLocalPort() throws IOException {
 DatagramSocket sckt = new DatagramSocket();
 int port = sckt.getLocalPort();
 sckt.close();
 return port;
}

private native int windowId();

private native int exec(String cmd);

protected void finalize() {
 try stop(); catch (IOException e);
}

private static final int WIDTH  = 352;
private static final int HEIGHT = 240;

private static final int STOP  = 0;
private static final int PLAY  = 1;
private static final int PAUSE = 2;

/* command identifiers */
private static final int MCMD_NULL       = 0;
private static final int MCMD_EXIT       = 1;
private static final int MCMD_OPENSRC    = 2;
private static final int MCMD_CLOSESRC   = 3;
private static final int MCMD_REENTER    = 4;
private static final int MCMD_PLAYCTR    = 5;
private static final int MCMD_PRESCTR    = 6;
private static final int MCMD_STREAM     = 7;
private static final int MCMD_SENDSTAT   = 8;
private static final int MCMD_STATUS     = 9;
private static final int MCMD_ACK        = 10;
```

```
/* command flags */
private static final int MCFL_SNDACK     = (1<<0);
private static final int MCFL_ORGMPX     = (1<<2);

/* command parameter values: */

/* source_type  :   MCMD_OPENSRC */
private static final int MSC_FNAME       = 1;
private static final int MSC_FDSCP       = 4;

/* flags   :   MCMD_REENTER */
private static final int MRE_FOFS        = (1<<0);
private static final int MRE_ASOPEN      = (1<<2);
private static final int MRE_STRMS       = (1<<3);
private static final int MRE_SEEKVSEQ    = (1<<4);

/* data_type    :   MCMD_OPENSRC, MCMD_REENTER */
private static final int BSTRM_11172     = (1<<0);
private static final int BSTRM_VSEQ      = (1<<1);
private static final int BSTRM_ASEQ      = (1<<2);

/* action       :   MCMD_PLAYCTR */
private static final int PC_PLAY         = (1<<0);
private static final int PC_FWDSPEED     = (1<<1);
private static final int PC_FWDSTEP      = (1<<2);
private static final int PC_PAUSE        = (1<<3);

/* which        :   MCMD_PRESCTR */
private static final int PCTR_VMD        = (1<<0);
private static final int PCTR_AMD        = (1<<1);
private static final int PCTR_AVOL       = (1<<2);
private static final int PCTR_LUM        = (1<<3);
private static final int PCTR_SAT        = (1<<4);
private static final int PCTR_GAM        = (1<<5);

/* video_mode   :   MCMD_PRESCTR
 * 0xvvzz
 *  vv : VDM_COL, VDM_COLB
 *  zz : zoom [1-3]
 */
private static final int VDM_COL         = 1;
private static final int VDM_COLB        = 2;

/* audio_mode   :   MCMD_PRESCTR
 *
 *    cccqqq
```

```
     *   ccc: channel listening selection
     *    Sxx :. 1/0 -> Selection/ No Selection
     *    101 : Left
     *    110 : Right
     *    111 : Left & Right
     *   qqq: audio playback quality selection
     *    Sxx·: 1/0 -> Selection/ No Selection
     *    100 : High
     *    101 : Medium
     *    110 : Low
     */

    /* stream        :    MCMD_STREAM, MCMD_OPENSRC, MCMD_REENTER

     *   vvvvvvvv.aaaaaaaa
     *   aaaaaaaa:
     *     a7: 1-> ignore stream identifier part (bits a5-a0).
     *     a6: audio stream subscription 0/ON, 1/OFF
     *     a5: 1->auto subscribe to first encountered audio
stream,
     *       (a4-a0 = 00000).
     *   a4-a0: subscribe to a particular audio stream [0-31]
     *
     *   vvvvvvvv:
     *     v7: 1-> ignore stream identifier part, bits v5-v0
     *     v6: video stream subscription 0/ON, 1/OFF
     *     v5: 1->auto subscribe to first encountered video
stream,
     *       (v4-v0 = 00000).
     *     v4: 0
     *   v3-v0: subscribe to particular video stream [0-15]
     *
     */

    private static final int STRM_IGNOREID   = 0x80;
    private static final int STRM_SBCOFF     = 0x40;
    private static final int STRM_AUTOSBC    = 0x20;

    static {
     try System.loadLibrary("javampx"); catch
(UnsatisfiedLinkError e)
         System.load("/opt/SUNWsmcjc/lib/libjavampx.so");
    }
}
```

smcrm

```
/*
 * @(#)smcrm.java
 *
 * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
 *
 * version     1.0
 * author Christopher Lindblad
 *
 */

package COM.Sun.isg.smcjc;

public class smcrm {
    private static byte[] parseHandle(String s) {
     int len = s.length()/2;
     byte[] h = new byte[len];
     for (int i = 0; i < len; i++) {
         h[i] = (byte) Integer.parseInt(s.substring(i*2,
(i+1)*2), 16);
     }
     return h;
    }
    public static void main (String args[]) throws Exception {
     MsmSession session = null;
     MsmPlayer player;
     if (args.length != 2) {
         System.err.println("usage: smcrm <serverName>
<playerHandle>");
         return;
     }
     try {
         session = new MsmSession(args[0]);
         player = new MsmPlayer(session, parseHandle(args[1]));
         player.delete();
     } catch (Exception e) {
         System.err.println("smcrm: " + e);
     } finally {
         if (session != null) {
          try session.close(); catch (Exception e)
              System.err.println("smcrm: " + e);
         }
     }
    }
}
```

smcstat

```
/*
 * @(#)smcstat.java
 *
 * Copyright 1995 Sun Microsystems, Inc.  All Rights Reserved.
 *
 * version     1.0
 * author Christopher Lindblad
 *
 */

package COM.Sun.isg.smcjc;

public class smcstat {
    public static void main (String args[]) throws Exception {
     MsmSession session = null;
     MsmPlayer[] players;
     if (args.length != 1) {
         System.err.println("usage: smcstat <serverName>");
         return;
     }
     try {
         session = new MsmSession(args[0]);
         players = session.players();
         System.out.println(session);
         for (int i = 0; i < players.length; i++) {
          MsmPlayer player = players[i];
          MsmPersistence persistence = player.getPersistence();
          MsmConnect connect = player.getConnect();
          MsmPlayStatus status = player.getPlayStatus();
          MsmAccessRight[] rights = player.getAccess();
          MsmPlaylist playlist = player.getPlaylist();
          System.out.println(player);
          System.out.println(persistence);
          System.out.println(connect);
          System.out.println(status);
          for (int j = 0; j < rights.length; j++) {
              System.out.println(rights[j]);
          }
          System.out.println(playlist);
          for (int j = 0; j < playlist.items.length; j++) {
              if (playlist.items[j] instanceof MsmTitleItem) {
               MsmTitleItem ti = (MsmTitleItem)playlist.items[j];
               System.out.println(
                   session.getTitleStatus(ti.titleName));
```

```
                    }
                }
              }
          } catch (Exception e) {
              System.err.println("smcstat: " + e);
          } finally {
              if (session != null) {
               try session.close(); catch (Exception e)
                   System.err.println("smcstat: " + e);
              }
          }
        }
    }
```

LOOP

```
/*
 * @(#)loop.c
 *
 * Copyright 1996 Sun Microsystems, Inc.  All Rights Reserved.
 *
 * version      1.0
 * author       Stephane CACHAT
 *
 */

#include <stdio.h>
#include <stdlib.h>

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <netdb.h>
#include <signal.h>
#include <errno.h>
#include <fcntl.h>
#include <assert.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <time.h>
#include <thread.h>
#include <sys/errno.h>
#include <sys/stropts.h>
#include <fcntl.h>
#include <atm/atmioctl.h>

#ifdef TRUE
#undef TRUE
#endif

#ifdef FALSE
#undef FALSE
#endif

#define FALSE 0
#define TRUE 1
```

```
#define BUF 1024*8

  /***********************************************
   *** Global variables                        ***
   ***********************************************/

/* Parameters */

char servername[256];
char * progName;
char *opt;
int port;
int port0;

/* Socket */

struct sockaddr_in adds;
int skt;
struct sockaddr_in addr;
struct sockaddr_in addx;
struct hostent * hp;
int len;

/* buffer */

char * buffer=NULL;

/* Multicast */

struct ip_mreq mreq;
char * host;

/* Thread */

thread_t Tpump;
int okdone=0;
int flag=1;

/* ATM */
int safd;
int ppa;
char ctlbuf[0x100];

#define vc port

  /***********************************************
   *** Receive&transmit info Multicast         ***
  ************************************************/

void * pumpM(void * result){
  while (flag) {                                    /*main loop*/
    len=recvfrom(skt,buffer,BUF,0,NULL,0);
    if (len) {
    sendto(skt,buffer,len,0,(struct sockaddr *)
&(addx),sizeof(addx));
    }
  }
  flag=1;
}

  /************************************************
   *** Receive&transmit info ATM                ***
   ************************************************/

void * pumpA(void * result){
  struct strbuf   ctl;
  struct strbuf   data;
  int             flags;
fprintf(stderr,"pumpA\n");
  ctl.buf = (char *) ctlbuf;
  ctl.maxlen = 0x100;
  ctl.len = 0;
  data.buf = (char *) buffer;
  data.maxlen = BUF;
  data.len = 0;
  flags = 0;
  while (flag) {                                    /*main loop*/
    if (getmsg(safd, &ctl, &data, &flags) < 0) {
      fprintf(stderr,"getmsg failed, errno=%d\n", errno);
      perror("");
      return;
    }
    len=data.len;
fprintf(stderr,"len=%d\n",len);
    if (len) {
    sendto(skt,buffer+4,len-4,0,(struct sockaddr *)
&(addx),sizeof(addx));
    }
  }
  flag=1;
}

  /************************************************
   *** Collecting arguments                     ***
```

```
    *************************************************/

void print_usage_and_exit (char* a){
    if (strlen(a)) fprintf(stderr,a);
    fprintf(stderr,"\n%s  redirect multicast or atm data stream
to lo0\n",progName);
    fprintf(stderr,"Usage\n");
    fprintf(stderr,"%s m <Multicast address> <in port> <out
port>\n",progName);
    fprintf(stderr,"%s a <VC> <out port>\n",progName);
    (void)exit(0);
}

static void collectArgs(int argc,char **argv){
    int i;
    int j=0;
    FILE * f;
    progName=*argv++;
    if (!*argv) print_usage_and_exit("");
    opt=*argv++;
    if (*opt=='a') {
      if (!*argv) print_usage_and_exit("");
      port=atoi(*argv++);
      if (!*argv) print_usage_and_exit("");
      port0=atoi(*argv++);
      if (port<=0) print_usage_and_exit("");
      if (*argv) print_usage_and_exit("");
      f=fopen("./loop.conf","r");
      if (!f){
        fprintf(stderr,"Can't open loop.conf");
        exit(-1);
      }
      host= (char*) malloc(256);
      fscanf(f,"%s",host);
      fclose(f);
    }else if (*opt=='m') {
      if (!*argv) print_usage_and_exit("");
      host=*argv++;
      if (!*argv) print_usage_and_exit("");
      port=atoi(*argv++);
      if (!*argv) print_usage_and_exit("");
      port0=atoi(*argv++);
      if (port<=0) print_usage_and_exit("");
      if (*argv) print_usage_and_exit("");
    } else print_usage_and_exit("");
}
```

```
  /***********************************************
   *** Getting server IP adress               ***
   ***********************************************/

void getaddr(){
  int udpport;
  unsigned long inaddr;
  struct hostent * hp;
  char n[256];
  int i;

  if (gethostname(servername,256)==-1)
print_usage_and_exit("error while getting hostname");
  if ((inaddr=inet_addr(servername))!=-1){
    adds.sin_addr.s_addr=inaddr;
  }else{
    hp=gethostbyname(servername);
    if (hp!=NULL){
      adds.sin_addr.s_addr=((struct in_addr*)
hp->h_addr)->s_addr;
      adds.sin_port = htons(udpport);
    }
  }
  if ((inaddr=inet_addr(host))!=-1){/*hostname*/
    mreq.imr_multiaddr.s_addr=inaddr;
  }else{
    hp=gethostbyname(host);
    if (hp!=NULL){
      mreq.imr_multiaddr.s_addr=((struct in_addr*)
hp->h_addr)->s_addr;
    }else{
      fprintf(stderr,"Multicast connect failed\n");
    }
  }
 /* mreq.imr_interface.s_addr=INADDR_ANY; */
  gethostname(n,256);
  hp=gethostbyname(n);
  if (hp!=NULL){
      mreq.imr_interface.s_addr=((struct in_addr*)
hp->h_addr)->s_addr;
      addx.sin_addr.s_addr=((struct in_addr*)
hp->h_addr)->s_addr;
      addx.sin_port = htons(port0);
  }else{
      fprintf(stderr,"Multicast connect failed\n");
  }
}
```

```
    /**************************************************
     *** Socket setting Multicast                   ***
     **************************************************/
void goM(){
    getaddr();
    skt=socket(AF_INET,SOCK_DGRAM,0);
    if (skt==0) {
      perror("Create socket");
      exit(EXIT_FAILURE);
    }
    addr.sin_family = AF_INET;
    addr.sin_addr.s_addr = INADDR_ANY;
    addr.sin_port = htons(port);
    bind(skt,(void *)&addr,sizeof(addr));
    if( setsockopt(skt, IPPROTO_IP, IP_ADD_MEMBERSHIP,(char*)&mreq,
sizeof(struct ip_mreq) ) == -1 ){
      fprintf(stderr,"Can't join multicast membership");
      exit(0);
    }
    if (fcntl(skt,F_SETFL,O_NDELAY)==-1){
      fprintf(stderr,"set socket options nb");
      exit(EXIT_FAILURE);
    }

    if (thr_create(0,0,pumpM,0,0,&Tpump)) perror("Can't create
Dispatcher");
}

    /**************************************************
     *** ATM interface setting                      ***
     **************************************************/
void goA(){
    int udpport;
    unsigned long inaddr;
    struct hostent * hp;
    char n[256];

    char interface[10];
    memset(interface, 0, sizeof (interface));
    strcpy(interface, host);
    ppa = interface[strlen(interface) - 1] - '0';
    if ((safd = sa_open(interface)) < 0) {
      fprintf(stderr,"open failed, errno=%d\n", errno);
      perror("open");
      exit(-1);
    }
fprintf(stderr,"ready to attach\n");
```

```
    sa_attach(safd, ppa, -1);
  fprintf(stderr,"attached\n");
    if (sa_add_vpci(safd, vc, NULL_ENCAP, BIG_BUF_TYPE) < 0) {
      fprintf(stderr,"sa_add_vpci failed, errno=%d\n", errno);
      exit(-1);
    }
    sa_setraw(safd);

    gethostname(n,256);
    hp=gethostbyname(n);
    if (hp!=NULL){
        addx.sin_addr.s_addr=((struct in_addr*)
 hp->h_addr)->s_addr;
        addx.sin_port = htons(port0);
    }else{
        fprintf(stderr,"lo0 connect failed\n");
    }
    skt=socket(AF_INET,SOCK_DGRAM,0);
    if (skt==0) {
      perror("Create socket");
      exit(EXIT_FAILURE);
    }
    addr.sin_family = AF_INET;
    addr.sin_addr.s_addr = INADDR_ANY;
    addr.sin_port = htons(port0);
    bind(skt,(void *)&addr,sizeof(addr));
    if (fcntl(skt,F_SETFL,O_NDELAY)==-1){
      fprintf(stderr,"set socket options nb");
      exit(EXIT_FAILURE);
    }

   if (thr_create(0,0,pumpA,0,0,&Tpump)) perror("Can't create
Dispatcher");
}

   /***********************************************
    *** Cleaning ATM                             ***
    ***********************************************/

void doneA(int arg){
 fprintf(stderr,"loop killed by signal %d\n",arg);
 if (!okdone){okdone=1;
  flag=0;
  while (!flag) {
    sleep(1);
  }
  fprintf(stderr,"dispatcher killed\n");
  if (sa_delete_vpci(safd, vc) < 0) {
    fprintf(stderr,"sa_delete_vpci failed, errno=%d\n", errno);
  };
fprintf(stderr,"ready to detach\n");
  sa_detach(safd, -1);
fprintf(stderr,"detached\n");
  sa_close(safd);
  close(skt);
  printf("socket closed\n");
  if (buffer) free(buffer);
  printf("Buffer free\n");
  exit(0);
}}

  /**************************************************
   *** Cleaning Multicast                          ***
   **************************************************/

void doneM(int arg){
 if (!okdone){okdone=1;
  if (setsockopt(skt,IPPROTO_IP, IP_DROP_MEMBERSHIP,(char *)
&mreq,sizeof(mreq))==-1){
    fprintf(stderr,"Can't drop multicast membership");
    exit(0);
  }
  printf("Multicast membership dropped\n");

  flag=0;
  while (!flag) {
    sleep(1);
  }
  printf("dispatcher killed\n");

  close(skt);
  printf("socket closed\n");
  if (buffer) free(buffer);
  printf("Buffer free\n");
  exit(0);
}}

  /**************************************************
   *** Main                                        ***
   **************************************************/

int main(int argc, char** argv)
{
  int i;
  buffer=(char*) malloc(BUF);
  collectArgs(argc,argv);
 if (*opt=='m'){
    printf("host=%s, port=%d, port0=%d\n",host,port,port0);
    signal(SIGQUIT,doneM);
    signal(SIGINT,doneM);
    signal(SIGUSR1,doneM);
    signal(SIGUSR2,doneM);

    printf("go M\n");
    goM();
 }else if (*opt=='a'){
    printf("interface=%s, vc=%d,port0=%d\n",host,vc,port0);
    signal(SIGQUIT,doneA);
    signal(SIGINT,doneA);
    signal(SIGUSR1,doneA);
    signal(SIGUSR2,doneA);

    printf("go A\n");
    goA();
 }

 printf("loop\n");

 while(1) sleep(60);
}
```

Claims

1. A method for processing in a computer which includes a memory a bit stream received from a bit stream server which is operatively coupled to the computer through a network, the method comprising:

   retrieving from a multimedia document stored in the memory a specification of a title;
   building from the specification of the title bit stream control signals which request a bit stream representing the title and which are in a form appropriate for processing by the bit stream server;
   transmitting the bit stream control signals to the bit stream server to thereby request from the bit stream server a bit stream representing the title;
   building from the specification of the title decoder control signals which direct a decoder to receive the bit stream from the bit stream server and which are in a form appropriate for processing by the decoder; and
   transmitting the decoder control signals to the decoder to thereby cause the decoder to receive and decode the bit stream.

2. An applet, capable of executing within a computer system, for requesting and controlling decoding of a bit stream specified in a multimedia document stored in a memory of the computer system, the applet comprising:

   an API module (i) which is configured to build from a specification of the bit stream in the multimedia document bit stream control signals which request transmission of the bit stream from a bit stream server and which are in a form appropriate for processing by the bit stream server and (ii) which is configured to transmit the bit stream control signals to the bit stream server to thereby request from the bit stream server a bit stream representing the title; and
   a decoder module (i) which is operatively coupled to the API module; (ii) which is configur d to build from the specification of the bit stream in the multimedia document decoder control signals which direct a decoder to

receiv the bit stream from th bit stream server and which are in a form appropriat for processing by the decoder; and (iii) which is configured to transmit the decoder control signals to the decoder to thereby cause the decoder to receive and decod the bit stream.

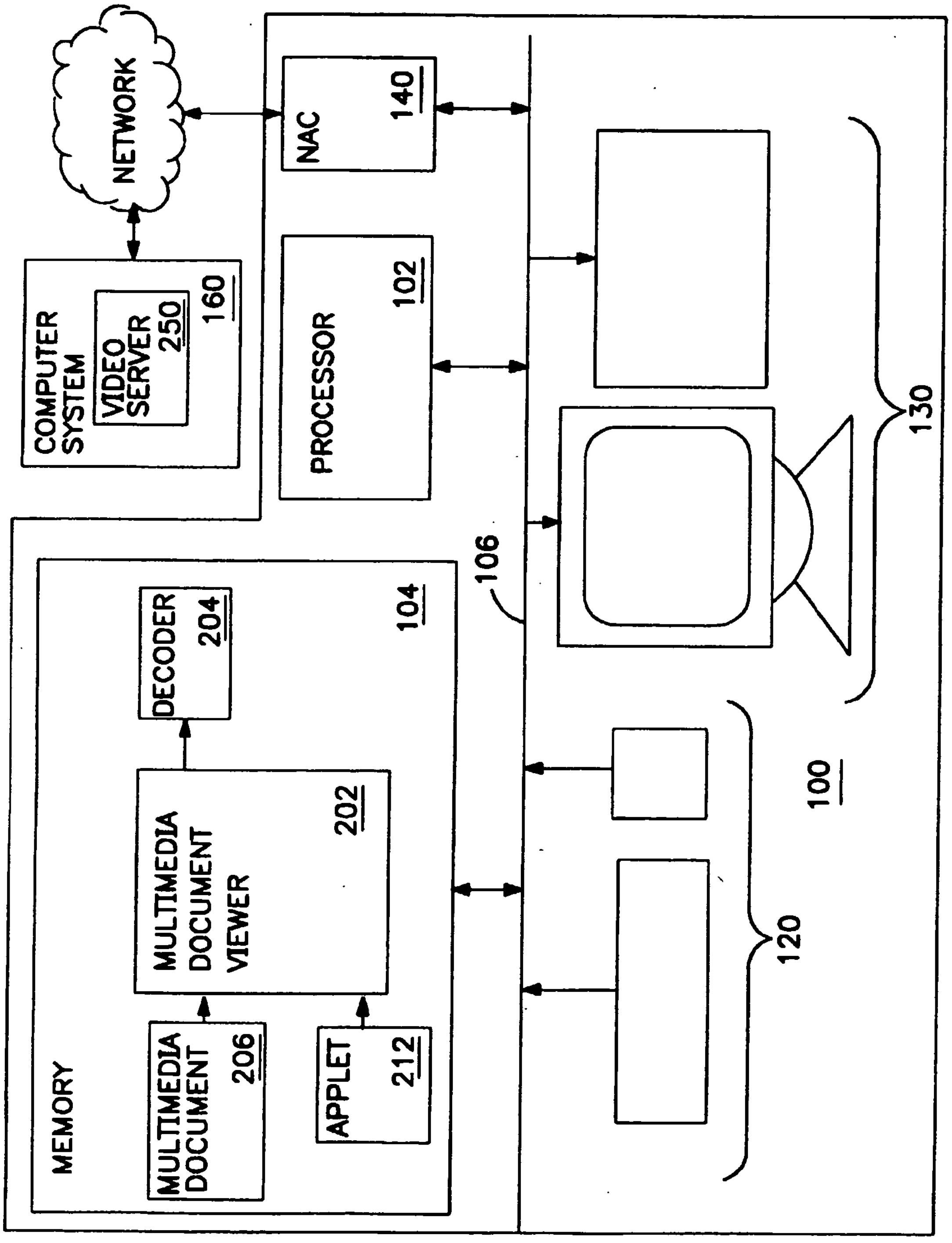

NETWORK
COMPUTER SYSTEM
VIDEO SERVER 250
160
NAC
140
PROCESSOR
102
MEMORY
MULTIMEDIA DOCUMENT 206
APPLET 212
MULTIMEDIA DOCUMENT VIEWER
202
DECODER 204
104
106
130
120
100

FIG. 1

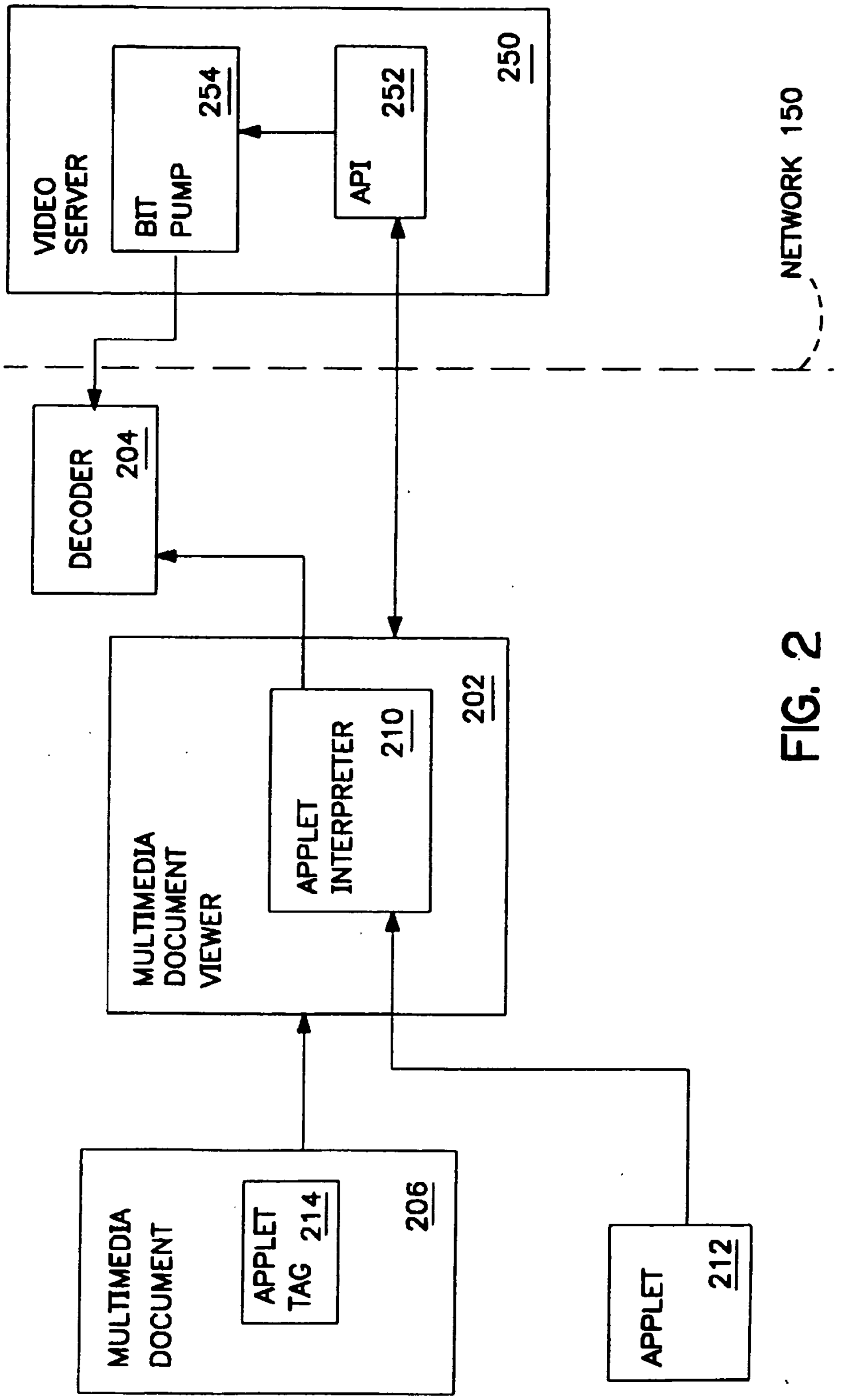

VIDEO SERVER
BIT PUMP
254
API
252
250
NETWORK 150
DECODER
204
MULTIMEDIA DOCUMENT VIEWER
APPLET INTERPRETER
210
202
MULTIMEDIA DOCUMENT
APPLET TAG 214
206
APPLET
212

FIG. 2

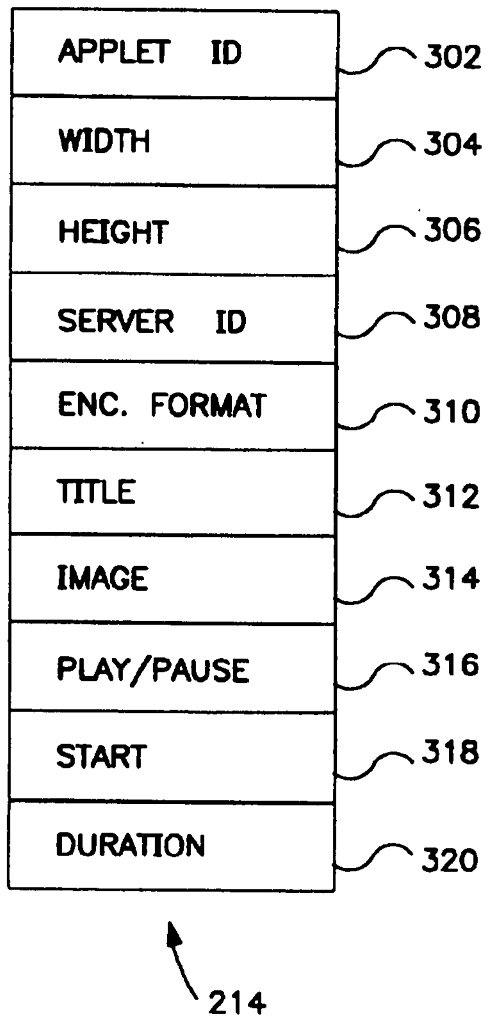

APPLET ID
302
WIDTH
304
HEIGHT
306
SERVER ID
308
ENC. FORMAT
310
TITLE
312
IMAGE
314
PLAY/PAUSE
316
START
318
DURATION
320
214

FIG. 3

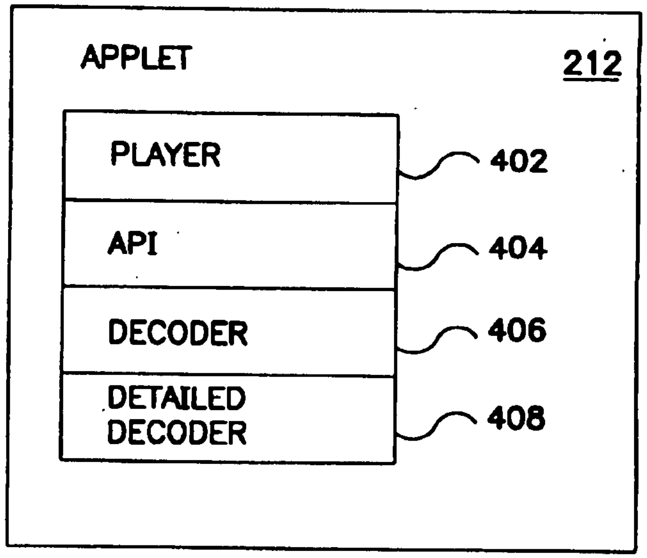

APPLET
212
PLAYER
402
API
404
DECODER
406
DETAILED DECODER
408

FIG. 4

2151

Attorney Docket No. 59501-8016.US01

I hereby certify that this correspondence is being deposited with the U.S. Postal Service with sufficient postage as First Class Mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C., 20231, on:

Date: November 18, 2002

By: Carina M. Tan
Carina M. Tan

Applicant: *CHEYER et al.*
Application No.: 09/225,198
Examiner L. A. Bullock, Jr.
Art Unit: 2151
Filed: January 5, 1999
For: **SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS**

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

1. Transmitted herewith are the following:

☒ Amendment and Response, with Version with Markings to Show Changes Made
☒ Declaration of Adam Cheyer
☒ Declaration of David L. Martin
☒ Applicants request one month extension of time

2. Entity Status

☒ Small Entity Status (37 CFR 1.9 and 1.27) has been established by a previously submitted Small Entity Statement.

3. Provisional Fee Authorization

Check No. 1123 the amount of $55.00 is enclosed for the one month extension of time. Please charge any underpayment in fees for timely filing of this transmittal and enclosures to Deposit Account No. 50-2207.

Respectfully submitted,
Perkins Coie LLP

Carina M. Tan

Date: November 18, 2002

Carina M. Tan
Registration No. 45,769

Corr spondence Address:
Customer No. 22918
Perkins Coie LLP
P.O. Box 2168
Menlo Park, CA 94
(650) 838-4300

#5/Amd A
T. McBeth Brown
12/10/02

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, DC 20231

on November 18, 2002 by Carina M. Tan

Carina M. Tan

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	Atty Dkt. No. 59501-8016.US01
CHEYER et al.	Group Art Unit No.: 2151
Serial No.: 09/225,198	Examiner: L. A. Bullock, Jr.
Filed on: January 5, 1999	

For: SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS

Commissioner of Patents
Washington, D.C. 20231

AMENDMENT AND RESPONSE

Sir:

This is in response to the Office Action mailed July 17, 2002, the shortened statutory period for which runs until October 17, 2002.

IN THE CLAIMS

Please amend Claims 1-3, 48, 84-88. A set of "clean" claims have been provided herein. Further, a set of claims having markings that show the changes that are made in this amendment is attached herewith. The attached pages are captioned "**Version of claims with markings to show changes made**."

AMENDED CLAIMS IN CLEAN FORM

IN THE CLAIMS:

Sub B1

1. (Once amended) A computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of:

registering a description of each active client agent's functional capabilities as corresponding registered functional capabilities, using an expandable, platform-independent, inter-agent language;

receiving a request for service as a base goal in the inter-agent language, in the form of an arbitrarily complex goal expression;

dynamically interpreting the arbitrarily complex goal expression, said act of interpreting further comprising:

generating one or more sub-goals expressed in the inter-agent language;

constructing a goal satisfaction plan that includes said one or more sub-goals; and

dispatching each of the sub-goals to a selected client agent for performance, based on a match between the sub-goal being dispatched and the registered functional capabilities of the selected client agent.

2. (Once amended) A computer-implemented method as recited in claim 1, further including the following acts of:

receiving a new request for service as a base goal using the inter-agent language, in the form of another arbitrarily complex goal expression, from at least one of the selected client agents in response to the sub-goal dispatched to said agent; and

recursively applying the step of dynamically interpreting the arbitrarily complex goal expression in order to perform the new request for service.

3. (Once amended) A computer-implemented method as recited in claim 2 wherein the act of registering a specific agent further includes:

invoking the specific agent in order to activate the specific agent;

instantiating an instance of the specific agent; and transmitting the new agent profile from the specific agent to a facilitator agent in response to the instantiation of the specific agent.

Sub B1

48. (Once amended) An Interagent Communication Language (ICL) providing a basis for facilitated cooperative task completion within a distributed computing environment having a facilitator agent and a plurality of autonomous service-providing electronic agents, wherein:

the ICL having one or more features from a set of features comprising:

enabling agents to perform queries of other agents;

enabling agents to exchange information with other agents; and

enabling agents to_set triggers within other agents; and

the ICL having a syntax supporting compound goal expressions wherein said compound goal expressions are such that goals within a single request provided according to the ICL syntax may be coupled by one or more operators from a set of operators comprising:

a conjunctive operator;

a conditional execution operator; and

a parallel disjunctive operator that indicates that disjunct goals are to be performed by different agents.

84. (Once amended) A computer architecture as recited in claim 71 wherein a planning component of the facilitating engine are distributed across at least two computer processes.

85. (Once amended) A computer architecture as recited in claim 71 wherein an execution component of the facilitating engine is distributed across at least two computer processes.

86. (Once amended) A data wave carrier providing a transport mechanism for information communication in a distributed computing environment having at least one facilitator agent and at least one active client agent, wherein said at least one facilitator agent is operable to construct a goal satisfaction plan for satisfying one or more requests for service from said at least one active client agent, the data wave carrier comprising a signal

representation of an inter-agent language description of an active client agent's functional capabilities.

87. (Once amended) A data wave carrier as recited in claim 86, the data wave carrier further comprising a corresponding signal representation of said one or more requests for service in the inter-agent language from a first agent to a second agent.

88. (Once amended) A data wave carrier as recited in claim 86, the data wave carrier further comprising a signal representation of a goal dispatched to an agent for performance from a facilitator agent.

REMARKS

The Examiner is thanked for the performance of a thorough search. By this amendment, Claims 1-3, 48, and 84-88 have been amended. No claims have been cancelled or added. Hence, Claims 1-89 are pending in the Application. It is respectfully submitted that the amendments to the claims as indicated herein do not add any new matter to this Application. Furthermore, amendments made to the claims as indicated herein have been made to improve readability and clarity of the claims.

SUMMARY OF REJECTIONS/OBJECTIONS

In the Office Action, Claim 2 is rejected under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 3 recites the limitation "from the specific agent to the facilitator agent" and is rejected under 35 U.S.C. § 112, second paragraph for lacking sufficient antecedent basis for this limitation in the claim.

Claims 84 and 85 are rejected under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claims 87 and 88 recite the limitation "A data wave carrier as recited in claim 85" and are rejected under 35 U.S.C. § 112, second paragraph for lacking sufficient antecedent basis for this limitation in the claim.

Claims 1, 2, 5-11, 15-28, 48-89 are rejected under 35 U.S.C. § 102(b) as being anticipated by "Building Distributed Software Systems With The Open Agent Architecture" by Martin et al.

Claims 1, 2, 5-11, and 15-25 are rejected under 35 U.S.C. 102(b) as being anticipated by "Development Tools for the Open Agent Architecture" by Martin et al.

Claims 3, 29-34, and 38-47 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Building Distributed Software Systems with the Open Agent Architecture" by Martin.

Claims 4, 12-14 and 35-37 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Building Distributed Software Systems with the Open Agent Architecture" by Martin 1 in view of "Information Brokering in an Agent Architecture" by Martin 2.

Claims 3, 29-34, 38-47, 61-71 and 84-89 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Developing Tools for the Open Agent Architecture" by Martin et al.

Claims 4, 12-14, 26-28, 35-37, 48-60, 72-83 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Development Tools for the Open Agent Architecture" by Martin 1 in view of "Information Brokering in an Agent Architecture" by Martin 2.

REJECTIONS UNDER 35 U.S.C. § 112

CLAIMS 2, 3, 84, 85, 87, and 88

In the Office Action, Claims 2, 3, 84, 85, 87, and 88 are rejected under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claims 2, 3, 84, 85, 87, and 88 are amended according to the suggestions of the Examiner. Thus, the amendments to the claims as indicated herein have been made in view of the Office Action's rejection under 35 U.S.C. § 112, second paragraph and to improve clarity of the claims.

AFFIDAVITS OF DAVID MARTIN AND ADAM CHEYER UNDER 37 CFR §1.132

Submitted herewith is a declaration under 37 CFR §1.132 by David Martin. In his declaration, David Martin avers that: 1) David Martin, Adam Cheyer and Douglas Moran are the co-authors of the reference, "Building Distributed Software Systems with the Open Agent

Architecture", 2) David Martin and Adam Cheyer are the only inventors of the subject application, 3) the reference, "Building Distributed Software Systems with the Open Agent Architecture" was published in March 1988, which is less than one year from the filing date of January 5, 1999.

Also, submitted herewith is a declaration under 37 CFR §1.132 by Adam Cheyer. In his declaration, Adam Cheyer avers that: 1) David Martin, Adam Cheyer and Douglas Moran are the co-authors of the reference, "Building Distributed Software Systems with the Open Agent Architecture", 2) David Martin and Adam Cheyer are the only inventors of the subject application, 3) the reference, "Building Distributed Software Systems with the Open Agent Architecture" was published in March 1988, which is less than one year from the filing date of January 5, 1999.

In accordance with MPEP 716.10, David Martin's declaration and Adam Cheyer's declaration render the reference, "Building Distributed Software Systems with the Open Agent Architecture" as inapplicable prior art.

REJECTIONS UNDER 35 U.S.C. § 102(b) and § 103(a)

CLAIM 1

Claim 1, as amended, recites in part:

> "receiving a request for service as a base goal in the inter-agent language, in the form of an **arbitrarily complex goal expression**;
> dynamically interpreting the arbitrarily complex goal expression, said act of interpreting further comprising:
> generating one or more sub-goals expressed in the inter-agent language;
> **constructing a goal satisfaction plan that includes said one or more sub-goals**;
> dispatching each of the sub-goals to a selected client agent for performance, based on a match between the sub-goal being dispatched and **the registered functional capabilities** of the selected client agent."

The novel method recited in Claim 1 requires "**constructing a goal satisfaction plan that includes said one or more sub-goals**." None of the cited references disclose, suggest or render obvious the limitation of "constructing a goal satisfaction plan that includes said one or more sub-goals." For example, Claim 1 requires constructing a goal satisfaction plan that includes said one or more sub-goals whenever the sub-goals cannot be generated by a simple decomposition of the "arbitrarily complex goal expression" in Claim 1. In other words, "**a goal satisfaction plan**" is needed to satisfy the "arbitrarily complex goal expression" in Claim 1 whenever there is no direct match between the components of arbitrarily complex goal expression and the "**registered functional capabilities**" of the client agents.

Since, none of the cited references disclose, suggest or render obvious the limitations of Claim 1 including the limitation of "constructing a goal satisfaction plan that includes said one or more sub-goals", Claim 1 is allowable over the art of record. It is respectfully submitted that Claim 1 be held in condition for allowance.

CLAIMS 2-28

Claims 2-28 are either directly or indirectly dependent upon independent Claim 1, and include all the features of Claim 1. Therefore, Claims 2-28 are allowable for at least the reasons provided herein with respect to Claim 1. Furthermore, it is respectfully submitted that Claims 2-28 recite additional features that independently render Claims 2-28 patentable over the art of record. Thus, it is respectfully submitted that Claims 2-28 be held in condition for allowance.

CLAIMS 29, 61, 71 and 86

Claims 29, 61, 71 and 86, each contain the limitation requiring the "construction of a goal satisfaction plan".

Claim 29, recites in part, the limitations of:

"**constructing** a base **goal satisfaction plan** including the sub-acts of:
determining whether the requested service is available,
determining sub-goals required in completing the base goal,
selecting service-providing electronic agents from the agent registry suitable for performing the determined sub-goals;"

Claim 61, recites in part, the limitations of:

"the facilitating engine further operable to **construct a goal satisfaction plan** specifying the coordination of a suitable delegation of sub-goal requests to complete the requested service satisfying both the local and global constraints and control parameters."

Claim 71, recites in part, the limitations of:

"the facilitating engine further operable to **construct a goal satisfaction plan** including the coordination of a suitable delegation of sub-goal requests to best complete the requested service."

Claim 86, recites in part, the limitations of:

"wherein said at least one facilitator agent is operable to **construct a goal satisfaction plan** for satisfying one or more requests for service from said at least one active client agent,"

Thus, Claims 29, 61, 71 and 86 contain limitations that are similar to those described herein with respect to Claim 1. Therefore, based on the reasons stated herein, it is respectfully submitted that Claims 29, 61, 71 and 86, are allowable over the art of record for at least the reasons provided herein with respect to Claim 1. Furthermore, it is respectfully submitted that Claims 29, 61, 71 and 86 recite additional features that independently render Claims 29, 61, 71 and 86 patentable over the art of record. Therefore, it is respectfully submitted that Claims 29, 61, 71 and 86 be held in condition for allowance.

CLAIMS 30-47, 62-70, 72-85, 87-89

Claims 30-47, 62-70, 72-85, 87-89 are either directly or indirectly dependent upon independent Claims 29, 61, 71 and 86, respectively. Therefore, Claims 30-47, 62-70, 72-85, 87-89 are allowable for at least the reasons provided herein with respect to Claims 29, 61, 71, 86 and 1. Furthermore, it is respectfully submitted that Claims 30-47, 62-70, 72-85, 87-89 recite additional features that independently render Claims 30-47, 62-70, 72-85, 87-89 patentable over the art of record. Thus, it is respectfully submitted that Claims 30-47, 62-70, 72-85, 87-89 be held in condition for allowance.

CLAIM 48

Claim 48, as amended, recites in part:

> "the ICL having a syntax supporting compound goal expressions wherein said_compound goal expressions are such that **goals within a single request** provided according to the ICL syntax may **be coupled by one or more operators from a set of operators** comprising:
> **a conjunctive operator;**
> **a conditional execution operator**; and
> **a parallel disjunctive operator** that indicates that disjunct goals are to be performed by different agents."

The novel method recited in Claim48 requires that "**goals within a single request**" are "**coupled by one or more operators from a set of operators**". In Claim 48, the set of operators comprise, **a conjunctive operator, a conditional execution operator,** and **a parallel disjunctive operator.**

None of the cited references disclose, suggest or render obvious the requirement that the "**goals within a single request**" be "coupled by one or more operators from a set of operators", such as **a conjunctive operator, a conditional execution operator,** and **a parallel disjunctive operator**. Claim 48 is allowable over the art of record. Thus, it is respectfully submitted that Claim 48 be held in condition for allowance.

CLAIMS 49-60

Claims 49-60 are either directly or indirectly dependent upon independent Claim 48, and include all the features of Claim 48. Therefore, Claims 49-60 are allowable for at least the reasons provided herein with respect to Claim 48. Furthermore, it is respectfully submitted that Claims 49-60 recite additional features that independently render Claims 49-60 patentable over the art of record. Thus, it is respectfully submitted that Claims 49-60 be held in condition for allowance.

CONCLUSION

For the reasons set forth above, it is respectfully submitted that all of the pending claims are now in condition for allowance. Therefore, the issuance of a formal Notice of Allowance is believed next in order, and that action is most earnestly solicited.

If in the opinion of the Examiner a telephone conference would expedite the prosecution of the subject application, the Examiner is encouraged to call the undersigned at (650) 838-4311.

The Commissioner is authorized to charge any fees due to Applicants' Deposit Account No. 50-2207.

Respectfully submitted,
Perkins Coie LLP

Carina M. Tan

Carina M. Tan
Registration No. 45,769

Date: November 18, 2002
(Monday)

Correspondence Address:

Customer No. 22918
Perkins Coie LLP
P. O. Box 2168
Menlo Park, California 94026
(650) 838-4300

VERSION OF CLAIMS WITH MARKINGS TO SHOW CHANGES MADE

1. (Once amended) A computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of:

registering a description of each active client agent's functional capabilities as corresponding registered functional capabilities, using an expandable, platform-independent, inter-agent language;

receiving a request for service as a base goal in the inter-agent language, in the form of an arbitrarily complex goal expression;

dynamically interpreting the arbitrarily complex goal expression, said act of interpreting further comprising:

generating one or more sub-goals [using] expressed in the inter-agent language; [and]

constructing a goal satisfaction plan that includes said one or more sub-goals; and

dispatching each of the sub-goals to a selected client agent for performance, based on a match between the sub-goal being dispatched and the registered functional capabilities of the selected client agent.

2. (Once amended) A computer-implemented method as recited in claim 1, further including the following acts of:

receiving a new request for service as a base goal using the inter-agent language, in the form of another arbitrarily complex goal expression, from at least one of the selected client agents in response to the sub-goal dispatched to said agent; and

recursively applying the [last] step of dynamically interpreting the arbitrarily complex goal expression [claim 1] in order to perform the new request for service.

3. (Once amended) A computer-implemented method as recited in claim 2 wherein the act of registering a specific agent further includes:

invoking the specific agent in order to activate the specific agent;

instantiating an instance of the specific agent; and

transmitting the new agent profile from the specific agent to [the] a facilitator agent in response to the instantiation of the specific agent.

48. (Once amended) An Interagent Communication Language (ICL) providing a basis for facilitated cooperative task completion within a distributed computing environment having a facilitator agent and a plurality of autonomous service-providing electronic agents, wherein:

the ICL having one or more features from a set of features comprising:

enabling agents to perform queries of other agents[,] ;

enabling agents to exchange information with other agents[,] ; and

enabling agents to set triggers within other agents[,] ; and

[in] the ICL having a syntax supporting compound goal expressions wherein said compound goal expressions are such that goals within a single request provided according to the ICL syntax may be coupled by one or more operators from a set of operators comprising:

a conjunctive operator[,] ;

a conditional execution operator[,] ; and

a parallel disjunctive operator [parallel disjunctive operator] that indicates that disjunct goals are to be performed by different agents.

84. (Once amended) A computer architecture as recited in claim 71 wherein [the] a planning component of the facilitating engine is distributed across at least two computer processes.

85. (Once amended) A computer architecture as recited in claim 71 wherein [the] an execution component of the facilitating engine is distributed across at least two computer processes.

86. (Once amended) A data wave carrier providing a transport mechanism for information communication in a distributed computing environment having at least one facilitator agent and at least one active client agent, wherein said at least one facilitator agent is operable to construct a goal satisfaction plan for satisfying one or more requests for service from said at least one active client agent, the data wave carrier comprising a signal representation of an inter-agent language description of an active client agent's functional capabilities.

87. (Once amended) A data wave carrier as recited in claim [85] 86, the data wave carrier further comprising a corresponding signal representation of [request] said one or more requests for service in the inter-agent language from a first agent to a second agent.

88. (Once amended) A data wave carrier as recited in claim [85] 86, the data wave carrier further comprising a signal representation of a goal dispatched to an agent for performance from a facilitator agent.

Serial No. 09/225.198

I hereby certify that this correspondence is being deposited with the U.S. Postal Service with sufficient postage as First Class Mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C., 20231, on:

Date: November 18, 2002 By: Carina M. Tan

DOCKET NO.: 59501-8016.US01

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF:

Cheyer *et al.*

SERIAL NO.: 09/225,198

FILED: 01/05/99

FOR: SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONICS AGENTS

EXAMINER: Bullock Jr., L.

ART UNIT: 2151

DECLARATION UNDER 37 C.F.R. §1.132

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

I, David L. Martin, declare and affirm as follows:

1. I am a co-inventor, along with Adam J. Cheyer, of the subject matter described and claimed in U.S. Patent Application Serial No. 09/225,198, filed January 05, 1999, entitled SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONICS AGENTS.

2. I am co-author of an article published in March, 1998, entitled "Building Distributed Software Systems with the Open Agent Architecture." The article included as co-authors, Adam J. Cheyer and Douglas B. Moran. Thus, the article was published less than one year from the filing date of the instant application.

3. I and Adam J. Cheyer are the inventors of the subject matter, which is claimed in claims 1-

86 in the instant application.

4. Douglas B. Moran is not a co-inventor of the subject matter described in the subject matter disclosed and claimed in the instant application.

I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the Unites States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Respectfully submitted,

11/14/2002
Date

David L. Martin
David L. Martin

#7

Serial No. 09/225,198

I hereby certify that this correspondence is being deposited with the U.S. Postal Service with sufficient postage as First Class Mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231, on:

Date November 18, 2002

By: Corina M. Tan

DOCKET NO.: 59501-8016.US01

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF:
Cheyer *et al.*
SERIAL NO.: 09/225,198
FILED: 01/05/99
FOR: SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONICS AGENTS

EXAMINER: Bullock Jr., L.
ART UNIT: 2151

DECLARATION UNDER 37 C.F.R. §1.132

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

I, Adam J Cheyer, declare and affirm as follows:

1. I am a co-inventor, along with David L. Martin, of the subject matter described and claimed in U.S. Patent Application Serial No. 09/225,198, filed January 05, 1999, entitled SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONICS AGENTS.

2. I am co-author of an article published in March, 1998, entitled "Building Distributed Software Systems with the Open Agent Architecture." The article included as co-authors, David L. Martin and Douglas B. Moran. Thus, the article was published less than one year from the filing date of the instant application.

3. I and David L. Martin are the inventors of the subject matter, which is claimed in claims 1-

Serial No. 09/225,198

86 in the instant application.

4. Douglas B. Moran is not a co-inventor of the subject matter described in the subject matter disclosec and claimed in the instant application.

I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the Unites States Code and that such willful false statments may jeopardize the validity of the application or any patent issued thereon.

Respectfully submitted,

11/15/02
Date

Adam J. Cheyer
Adam J. Cheyer

3 M

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/225,198	01/05/1999	ADAM J. CHEYER	SRI1P016	2756

25696 7590 03/03/2003

OPPENHEIMER WOLFF & DONNELLY
P. O. BOX 10356
PALO ALTO, CA 94303

EXAMINER
BULLOCK JR, LEWIS ALEXANDER 8

ART UNIT	PAPER NUMBER
2126	

DATE MAILED: 03/03/2003

Please find below and/or attached an Office communication concerning this application or proceeding.

Offic Action Summary	**Applicati n No.** 09/225,198	**Applicant(s)** CHEYER ET AL.	
	Examiner Lewis A. Bullock, Jr.	**Art Unit** 2126	

-- The MAILING DATE of this communicati n appears n the cover sheet with the correspondenc address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 25 November 2002 .

2a)□ This action is **FINAL**. 2b)☒ This action is non-final.

3)□ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-89 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)□ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-89 is/are rejected.

7)□ Claim(s) _____ is/are objected to.

8)□ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)□ The specification is objected to by the Examiner.

10)□ The drawing(s) filed on _____ is/are: a)□ accepted or b)□ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)□ The proposed drawing correction filed on _____ is: a)□ approved b)□ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)□ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)□ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)□ All b)□ Some * c)□ None of:

1.□ Certified copies of the priority documents have been received.

2.□ Certified copies of the priority documents have been received in Application No. _____ .

3.□ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)□ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) □ The translation of the foreign language provisional application has been received.

15)□ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) □ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO-1449) Paper No(s) 4 .

4) □ Interview Summary (PTO-413) Paper No(s). _____ .

5) □ Notice of Informal Patent Application (PTO-152)

6) □ Other: .

DETAILED ACTION

Compact Disc Submission

1. The description portion of this application contains a computer program listing consisting of more than three hundred (300) lines. In accordance with 37 CFR 1.96(c), a computer program listing printout of more than three hundred lines must be submitted as a computer program listing appendix on compact disc conforming to the standards set forth in 37 CFR 1.96(c)(2) and must be appropriately referenced in the specification (see 37 CFR 1.77(b)(4)). Accordingly, applicant is required to cancel the computer program listing appearing in the specification on pages Appendix A.I, file a computer program listing appendix on compact disc in compliance with 37 CFR 1.96(c) and insert an appropriate reference to the newly added computer program listing appendix on compact disc at the beginning of the specification.

Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1-89 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Development Tools for the Open Agent Architecture" by MARTIN1 in view of "Information Brokering in an Agent Architecture" by MARTIN2.

As to claim 1, MARTIN1 teaches a computer-implemented method for communication and cooperative task completion among a plurality of distributed agents (sub-agents / agents), comprising the acts of: registering a description of each client agent's functional capabilities, using a platform independent inter-agent language (pg. 5, Each facilitator records the published capabilities of their subagents..."); receiving a request as a base goal in the inter-agent language (ICL form), in the form of an arbitrarily complex goal expression (request) (pg. 5, "...and when requests arrive.."); and dynamically interpreting the complex goal expression (request) comprising: generating one or more sub-goals (sub-request) expressed in the inter-agent language (ICL) (pg. 5, ...the facilitator is responsible for breaking them down and for distributing subrequest.."); and dispatching each of the sub-goals (sub-request) to a selected client agent (agent) for performance ("pg. 5, "...and when requests arrive (expressed in the Inter-agent Communication Language, described below), the facilitator is responsible for breaking them down and for distributing sub-requests to the appropriate agents; "For example, every agent can...and request solutions for a set of goals,..."). It would be inherent that since the functionalities of an agent are registered with the facilitator that they are stored registered functional capabilities of that agent and that the request is a complex goal since the facilitator can be requested to provide solutions for a set of goals (pg. 5). However, MARTIN1 does not teach the step of constructing a goal satisfaction plan.

MARTIN2 teaches an agent architecture for request communication comprising the step of constructing a goal satisfaction plan (query execution plan) that includes one or more sub-goals (sub-queries) and dispatching each sub-goal (sub-queries) to a selected agent (source) for performance based on a match between the capabilities of the agent and the sub-goal ("for each chunk, rewrite it as a disjunction of translated sub-queries where each disjunct is the translation of the sub-query for one of the source s that can handle that chunk.") (pg. 11-12, Query Processing). Therefore, it would be obvious to one skilled in the art to combine the teachings of MARTIN1 with the teachings of MARTIN2 in order to facilitate query processing (pg. 11).

As to claim 29, MARTIN1 teaches a method to facilitate cooperative task completion within a distributed computing environment supporting an Inter-agent Communication Language among a plurality of electronic agents (sub-agents / agents) comprising: providing an agent registry as disclosed (facilitator storage of published sub-agents capabilities); interpreting a service request in order to determine a base goal (via facilitator); determining whether the requested service is available, determining sub-goals required in completing the base goal (determine solutions for a set of goals) selecting suitable service-providing electronic agents for performing the sub-goals, and ordering a delegation of sub-goal requests to complete the requested service (pg. 5, "The facilitator is responsible for breaking them down and for distributing sub-requests to the appropriate agents."). However, MARTIN1 does not explicitly mention that the method is operable in a computer program product or the sending of advice or constraints. It would be obvious that since an agent can request solutions for a goal to be satisfied under a variety of different control strategies (pg. 5) that the control strategies are the advice and constraints. It would also be obvious to one skilled in the art to generate program code that would entail the method of MARTIN1 and thereby obvious that the method can be entailed in a computer program product. However, MARTIN1 does not teach the step of constructing a base goal satisfaction plan.

MARTIN2 teaches an agent architecture for request communication comprising the step of constructing a goal satisfaction plan (query execution plan) comprising: determining whether the service is available (determine what set of sources provides solutions for that predicate), determining sub-goals required in completing the base goal (determine which are the largest sub-queries that can be treated as chunks and which sources can handle each chunk); selecting service-providing agents ("which sources can handle each chunk), and ordering a delegation of sub-gal request to best complete the requested service ("for each chunk, rewrite it as a disjunction of translated sub-queries...each translated subquery is labeled with the name of the source by which it is to be solved."); and implementing the base goal satisfaction plan ("The plan is then interpreted according to Prolog semantics.") (pg. 11-12, Query Processing). It would be obvious that since an agent can request solutions for a goal to be satisfied under a variety of different control strategies (pg. 5) that the control strategies are the advice and/or constraints. It would also be obvious to one skilled in the art to generate program code that would entail the method of MARTIN2 and thereby obvious that the method can be entailed in a computer program product. Refer to claim 1 for the motivation to combine.

As to claim 48, MARTIN1 teaches an Inter-agent Communication Language (ICL) providing a basis for facilitated cooperative task completion within a distributed computing environment having a facilitator agent (facilitator) and a plurality of electronic agents (sub-agents / agents), the ICL having a feature for allowing the enabling agents (client / agent) to perform queries of other agents (pg. 5, Agents share a common communication language...and may run on any network linked platform."). However, MARTIN1 does not teach the ICL supporting compound goal expressions.

MARTIN2 teaches the query is a base goal stored in as a compound goal having sub-goals (pg. 8, "Queries submitted to the Broker are expression...and backtracking in expressing and processing queries.") and the ICL having expression which may be coupled by a conjunctive operator (pg. 10, "Although the body of the broker predicate rule is characterized as a conjunction of predicates."). It would be obvious that since the base goal (query) is broken down and distributed to as sub-requests to the appropriate agents or solutions are requested for a set of goals as disclosed in MARTIN1 that the base goal as a compound goal is broken down based on operators disclosing where it can be broken down. Refer to claim 1 for the motivation to combine.

As to claim 61, MARTIN1 teaches a facilitator agent (facilitator) arranged to coordinate task completion (process coordination) within a distributed computing environment having a plurality of electronic agents (agents / clients), comprising: an agent registry (storage of records of published capabilities of their subagents) that declares capabilities of service-providing electronic agents (subagents) currently active within the distributed computing environment and that request have constraints and parameters (control strategies) (pg. 5, The Open Agent Architecture). However, MARTIN1 does not teach the facilitating engine.

MARTIN2 teaches a facilitator agent (facilitator) having a facilitating engine (broker agent) (pg. 7, "...the Information Broker agent, working in close cooperation with the OAA facilitotor.") operable to parse a service request in order to interpret a compound goal (pg. 7, "The Broker accepts request (queries) from..."; "The Broker delegates, translates, and relays the appropriate sub-queries to the available source agents.."; pg. 8, "Each query is syntactically the same as a Prolog goal, usually a compound goal."), the compound goal including constraints and parameters (built-in predicates) (pg. 11, "..ICL built-in predicates (including arithmetic comparisons) are included with chuncks to be solved by sources."), the service request formed according to an ICL (pg. 11), the engine further operable to construct a goal satisfaction plan (query execution plan) specifying the coordination of a suitable delegation of sub-goal (sub-queries) requests to complete the requested service satisfying the constraints and parameters (pg. 11, Query Processing). Refer to claim 1 for the motivation to combine.

As to claim 71, reference is made to an architecture that encompasses the agent of claim 61 above, and is therefore met by the rejection of claim 61 above. However claim 71, further details the facilitator agent in bi-directional communication with the electronic agents. MARTIN1 teaches the facilitator can distribute request to the agents and the agents can request information via the facilitator (pg. 5), therefore it would be obvious that the facilitator and agents are in bi-directional communication.

As to claim 86, MARTIN1 teaches a method for information communication in a distributed computing environment having at least one facilitator agent (facilitator) and at least one client agent (sub-agent / agents), comprising storing a representation of an inter-agent language description (ICL registration of capabilities) of a client agent's functional capabilities (pg. 5, "Each facilitator records the published capabilities of their subagents.."). However, MARTIN1 does not explicitly mention that the method is operable in a data wave carrier. It would be obvious and well known in the art that one skilled in the art would generate program code on a data wave carrier that would entail the method of MARTIN1 and thereby obvious that the method can be entailed in a data wave carrier. However, MARTIN1 does not teach the facilitator agent is operable to construct a goal satisfaction plan.

MARTIN2 teaches an agent system for information communication wherein a facilitation agent (broker agent) is operable to construct a goal satisfaction plan (query execution plan) for satisfying one or more request (query) for service from the at least one active client agent (source) (pg. 11-12, Query Processing). Refer to claim 1 for the motivation to combine.

As to claim 2, MARTIN1 teaches receiving a new request for service as a base goal from at least one of the selected client agents in response to the sub-goal and recursively applying the dynamically interpreting step (pg. 5, "An agent satisfying a request may require supporting information, and the OAA provides numerous means of requesting data from other agents or from the user.").

As to claim 3, MARTIN1 teaches the act of registering and transmitting the new agent profile from the specific agent to the facilitator agent (pg. 5, "Every agent participating in an OAA-based system defines and publishes a set of capabilities specifications, expressed in the ICL, describing the services that it provides."). It would be obvious that an agent that is initially created is instantiated in memory before it is registered.

As to claim 4, MARTIN2 teaches deactivating a client agent no longer available to provide services by deleting the registration (pg. 9, Source agents that need to go offline...so that it can unregister the source and retract its schema mapping rules.").

As to claims 5-10, MARTIN1 teaches providing an agent registry data structure that can comprise of symbolic names, data declarations, trigger declarations, and task and process characteristics (pg. 5, "For example, every agent can install local or remote triggers on data...").

As to claim 11, MARTIN1 teaches establishing communication between distributed agents (pg. 5, ...the facilitator is responsible for breaking them down and for distributing sub-requests to the appropriate agent.").

As to claims 12-14, MARTIN2 teaches receiving a request for service in a second language (source schema); selecting a registered agent capable of converting the second language into the inter-agent language (broker schema); and forwarding the request for service in a second language to the registered agent for conversion to be performed and the results returned (pg. 12-13, Queries Expressed in a Source Schema).

As to claims 15-25, MARTIN1 teaches the base goal requires setting a trigger having conditional functionality and consequential functionality which can be stored on the facilitator agent and/or the service providing agent (pg. 5, "For example, every agent can install local or remote triggers on data...").

As to claims 26-28, MARTIN2 teaches the base goal is a compound goal having sub-goals (pg. 8, "Queries submitted to the Broker are expression...and backtracking in expressing and processing queries."). It would be obvious that since the base goal (query) is broken down and distributed to as sub-requests to the appropriate agents or solutions are requested for a set of goals as disclosed in MARTIN1 that the base goal as a compound goal is broken down based on operators disclosing where it can be broken down.

As to claims 30 and 31, MARTIN1 teaches registering a specific agent (agent) into the agent registry (list of agents capabilities) comprising: establishing a bi-directional communications link between the specific agent and a facilitator agent controlling the agent registry; providing a new agent profile to the facilitator agent; and registering the specific agent with the profile thereby making the capabilities available to the facilitator agent (pg. 5, "Each facilitator records the published capabilities of their subagents..."; "Every agent participating in an OAA-based system...describing the services that it provides.").

As to claim 32, refer to claim 3 for rejection.

As to claim 33, refer to claim 5 for rejection.

As to claim 34, refer to claim 11 for rejection.

As to claims 35-37, refer to claims 12-14 for rejection.

As to claims 38-44, refer to claims 15-25 for rejection.

As to claims 45-47, refer to claims 26-28 for rejection.

As to claim 49 and 50, MARTIN1 teaches the ICL is platform and language independent (pg. 5, "The OAA's Inter-agent Communication Language...they are programmed in.").

As to claims 51-54, MARTIN1 teaches the ICL supports task completion constraints (triggers) within goal exprèssions (pg. 5).

As to claims 55-60, MARTIN1 teaches each electronic agent defines and publishes a set of capability declarations or solvables that describe services and an interface to the electronic agent (pg. 5, "Every agent participating in an OAA-based system defines and publishes...we refer to these capabilities specifications as solvables.").

As to claim 62, MARTIN2 teaches the facilitating engine (broker agent) is able to receive events such as online and offline agents (pg. 8-9, The Broker agent). It would be obvious that the plan is modified if a particular agent goes offline since that agent is no longer available.

As to claim 63, refer to claim 5 for rejection.