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.

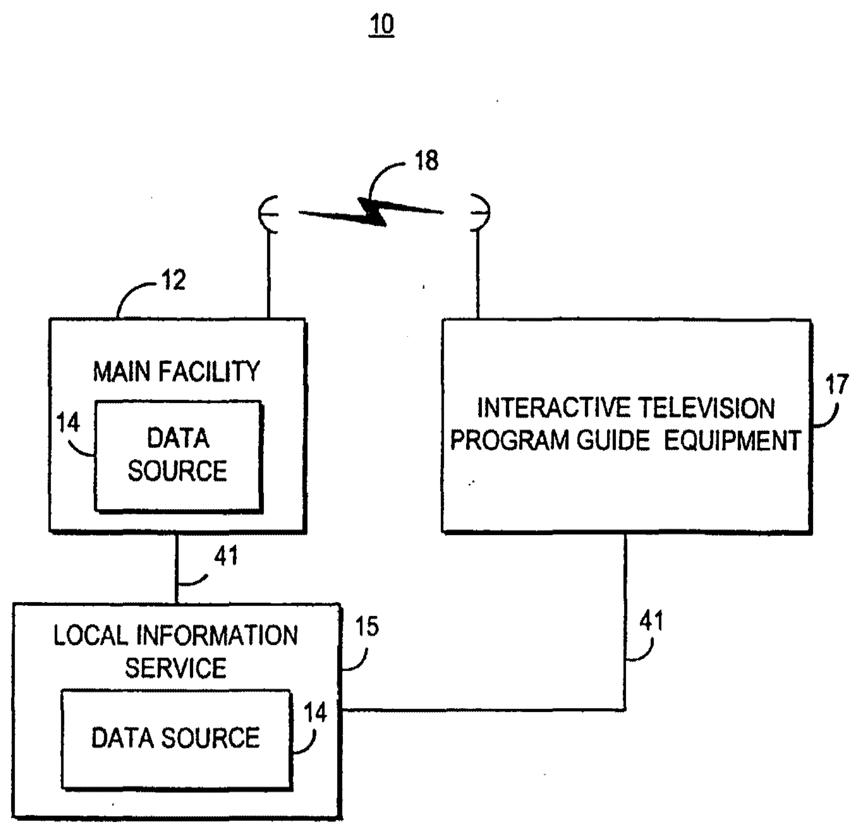

10
18
12
MAIN FACILITY
14
DATA SOURCE
INTERACTIVE TELEVISION PROGRAM GUIDE EQUIPMENT
17
41
LOCAL INFORMATION SERVICE
15
14
DATA SOURCE
41

FIG. 1

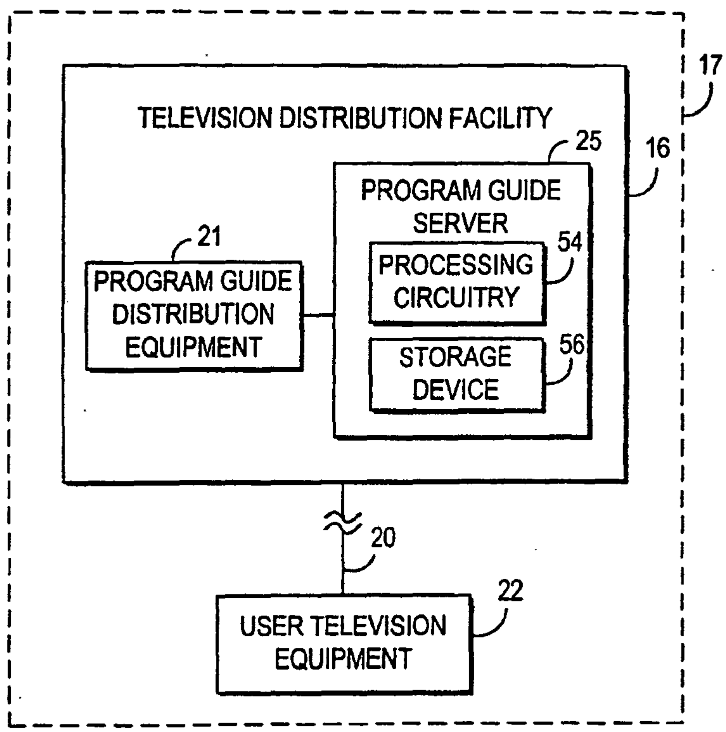

TELEVISION DISTRIBUTION FACILITY
PROGRAM GUIDE SERVER
PROCESSING CIRCUITRY
STORAGE DEVICE
PROGRAM GUIDE DISTRIBUTION EQUIPMENT
USER TELEVISION EQUIPMENT
17
16
25
21
54
56
20
22

FIG. 2a

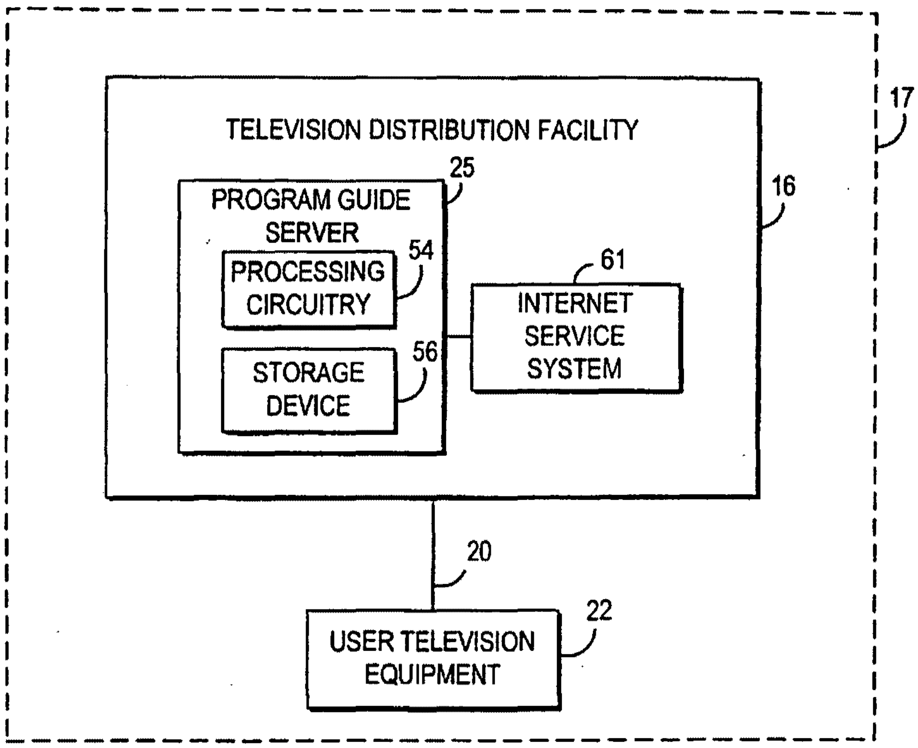

TELEVISION DISTRIBUTION FACILITY
PROGRAM GUIDE SERVER
25
PROCESSING CIRCUITRY
54
STORAGE DEVICE
56
INTERNET SERVICE SYSTEM
61
16
17
20
USER TELEVISION EQUIPMENT
22

FIG. 2b

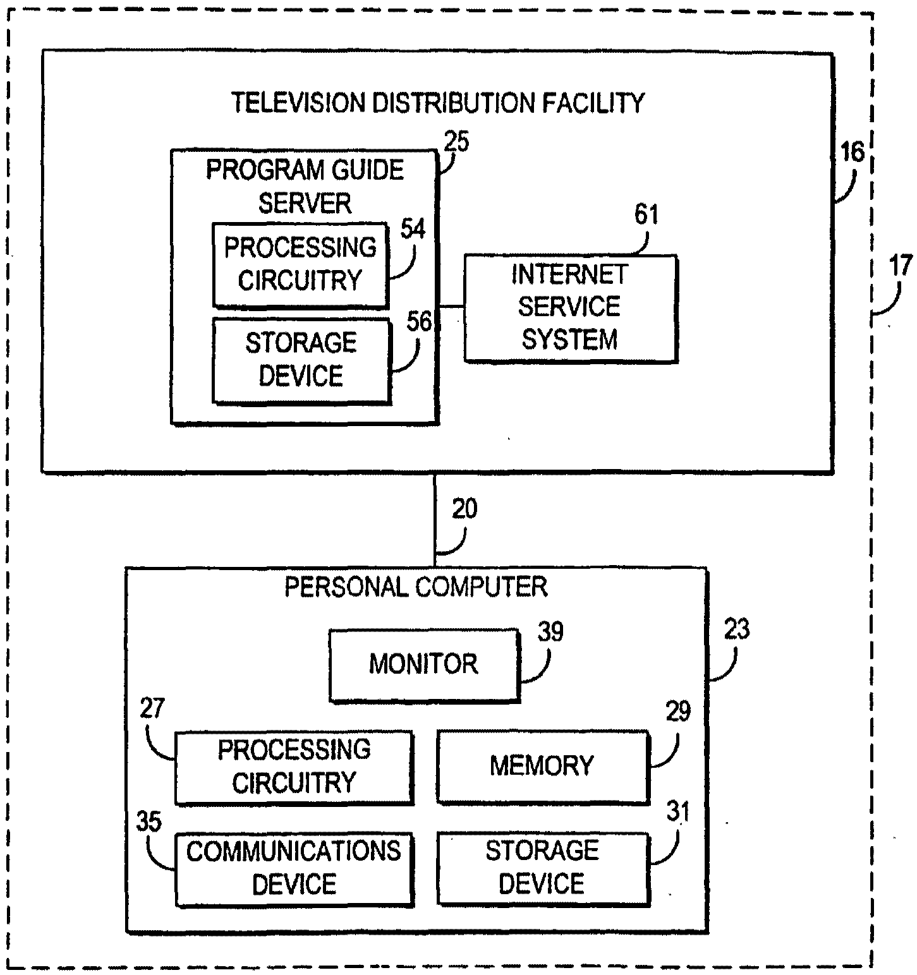

TELEVISION DISTRIBUTION FACILITY
PROGRAM GUIDE SERVER
25
PROCESSING CIRCUITRY
54
STORAGE DEVICE
56
61
INTERNET SERVICE SYSTEM
16
17
20
PERSONAL COMPUTER
MONITOR
39
23
27
PROCESSING CIRCUITRY
MEMORY
29
35
COMMUNICATIONS DEVICE
STORAGE DEVICE
31

FIG. 2c

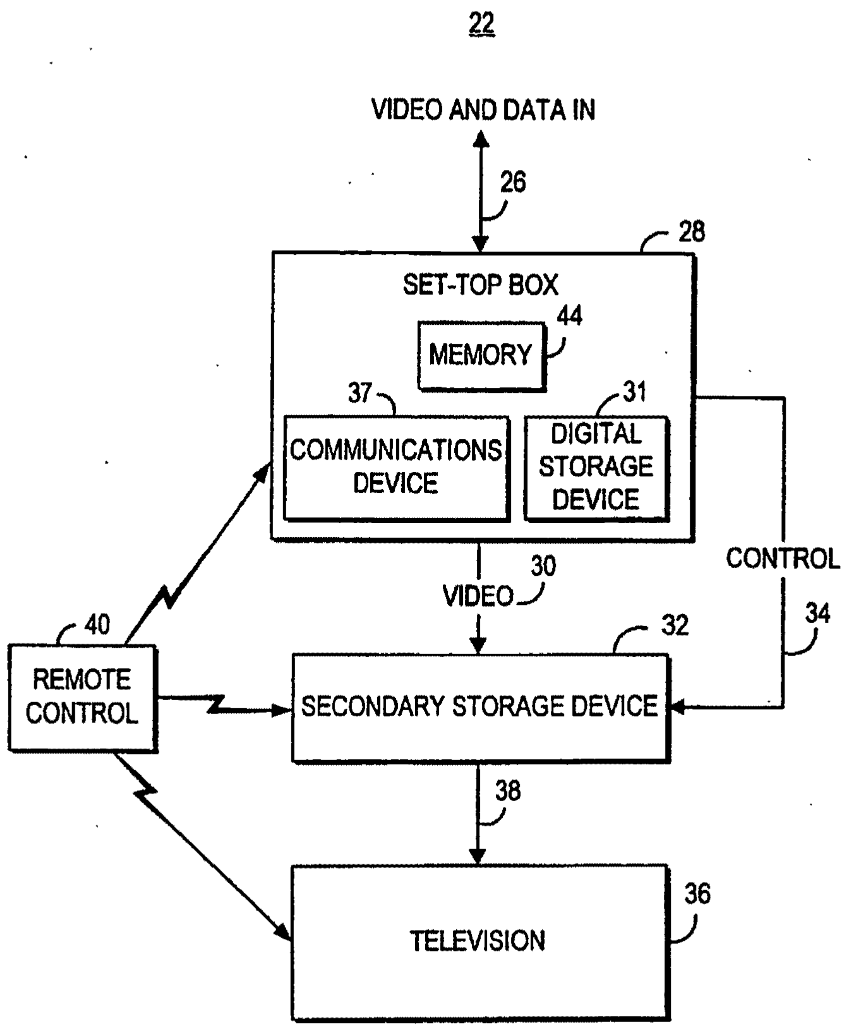

22
VIDEO AND DATA IN
26
28
SET-TOP BOX
MEMORY
44
37
31
COMMUNICATIONS DEVICE
DIGITAL STORAGE DEVICE
30
VIDEO
CONTROL
34
40
32
REMOTE CONTROL
SECONDARY STORAGE DEVICE
38
36
TELEVISION

FIG. 3

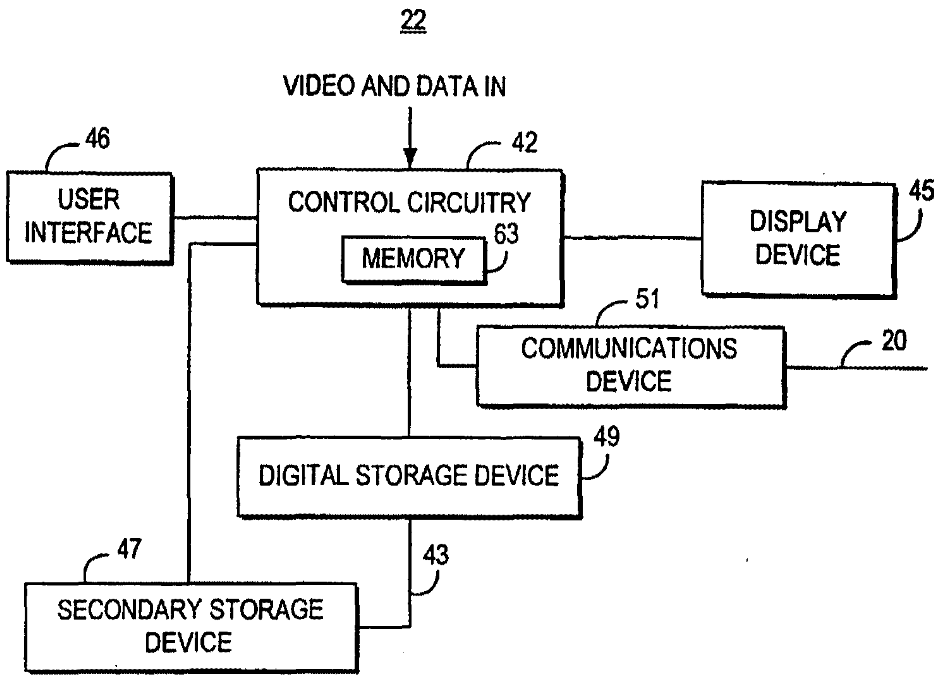

22
VIDEO AND DATA IN
46
USER INTERFACE
42
CONTROL CIRCUITRY
MEMORY
63
DISPLAY DEVICE
45
51
COMMUNICATIONS DEVICE
20
DIGITAL STORAGE DEVICE
49
47
43
SECONDARY STORAGE DEVICE

FIG. 4

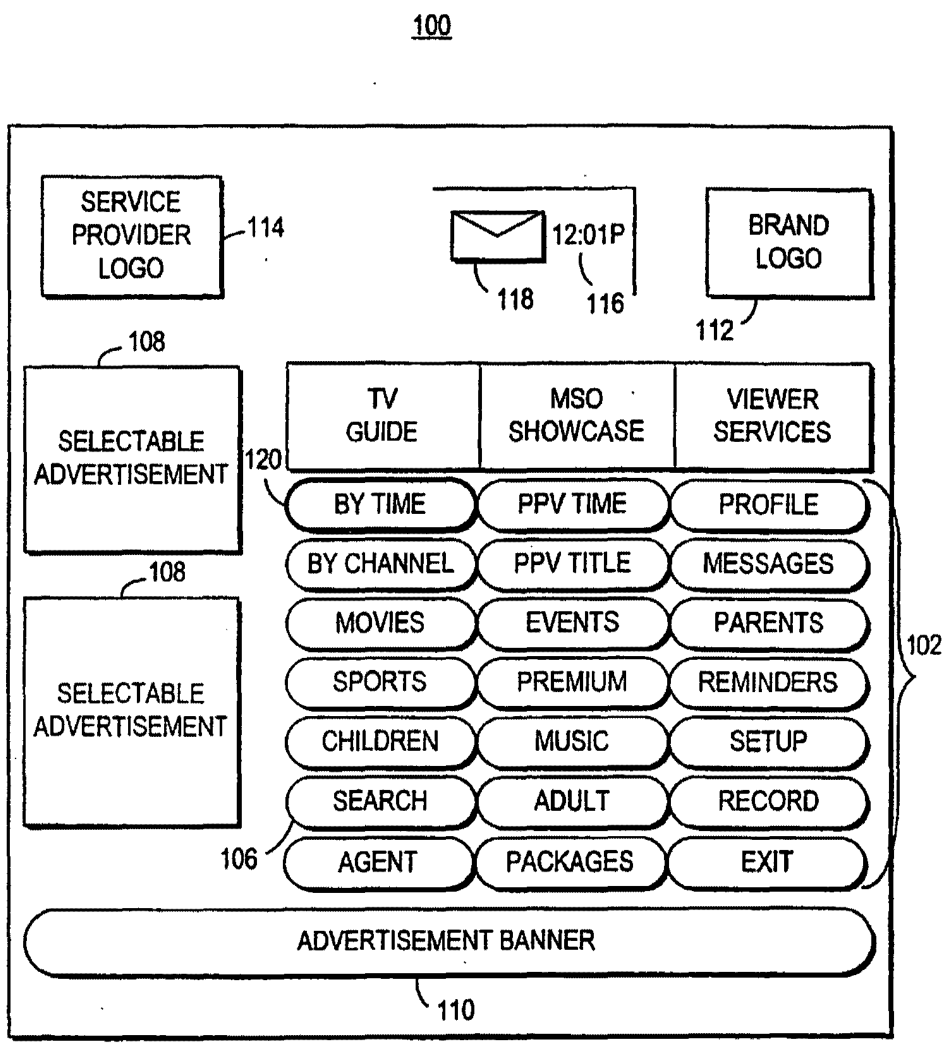

100
SERVICE PROVIDER LOGO
114
12:01P
118
116
BRAND LOGO
112
108
SELECTABLE ADVERTISEMENT
TV GUIDE
MSO SHOWCASE
VIEWER SERVICES
120
BY TIME
PPV TIME
PROFILE
BY CHANNEL
PPV TITLE
MESSAGES
108
MOVIES
EVENTS
PARENTS
102
SELECTABLE ADVERTISEMENT
SPORTS
PREMIUM
REMINDERS
CHILDREN
MUSIC
SETUP
SEARCH
ADULT
RECORD
106
AGENT
PACKAGES
EXIT
ADVERTISEMENT BANNER
110

FIG. 5

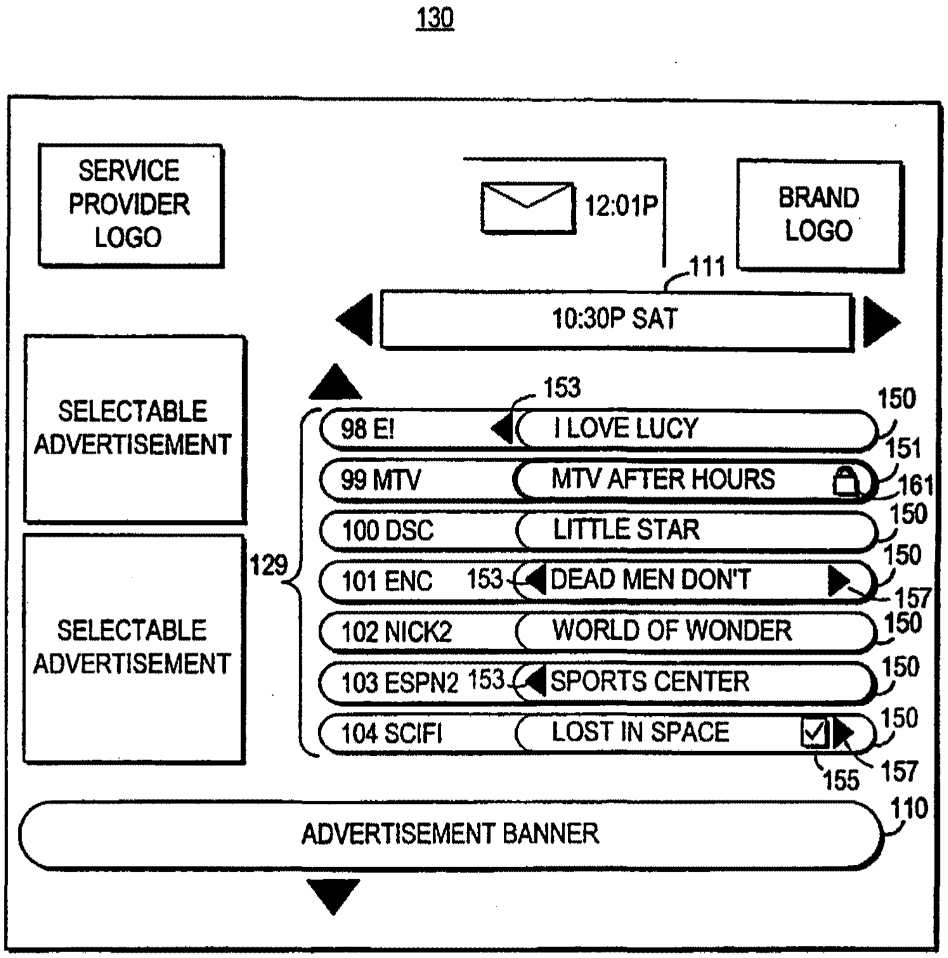

130
SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
111
10:30P SAT
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
129
153
150
98 E!
I LOVE LUCY
151
99 MTV
MTV AFTER HOURS
161
100 DSC
LITTLE STAR
150
150
101 ENC
153
DEAD MEN DON'T
157
102 NICK2
WORLD OF WONDER
150
103 ESPN2
153
SPORTS CENTER
150
104 SCIFI
LOST IN SPACE
150
155
157
ADVERTISEMENT BANNER
110

FIG. 6

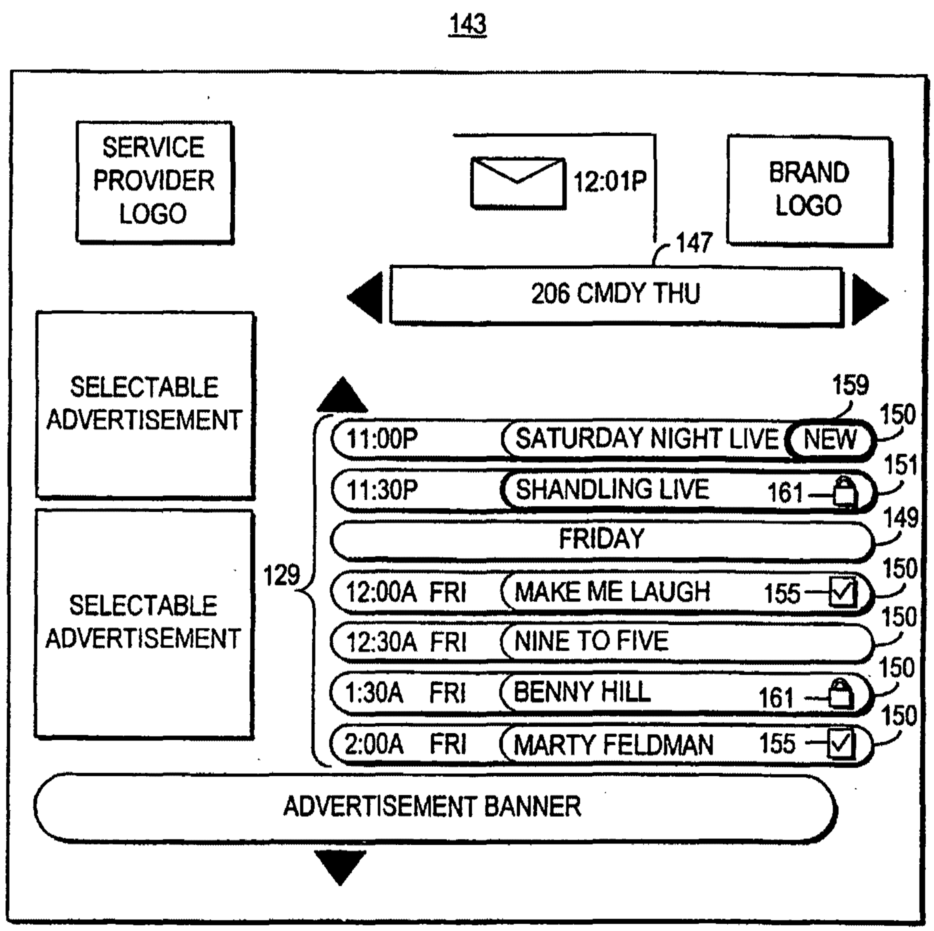

143
SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
147
206 CMDY THU
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
159
150
11:00P
SATURDAY NIGHT LIVE
NEW
151
11:30P
SHANDLING LIVE
161
149
FRIDAY
129
150
12:00A FRI
MAKE ME LAUGH
155
150
12:30A FRI
NINE TO FIVE
150
1:30A FRI
BENNY HILL
161
150
2:00A FRI
MARTY FELDMAN
155
ADVERTISEMENT BANNER

FIG. 7

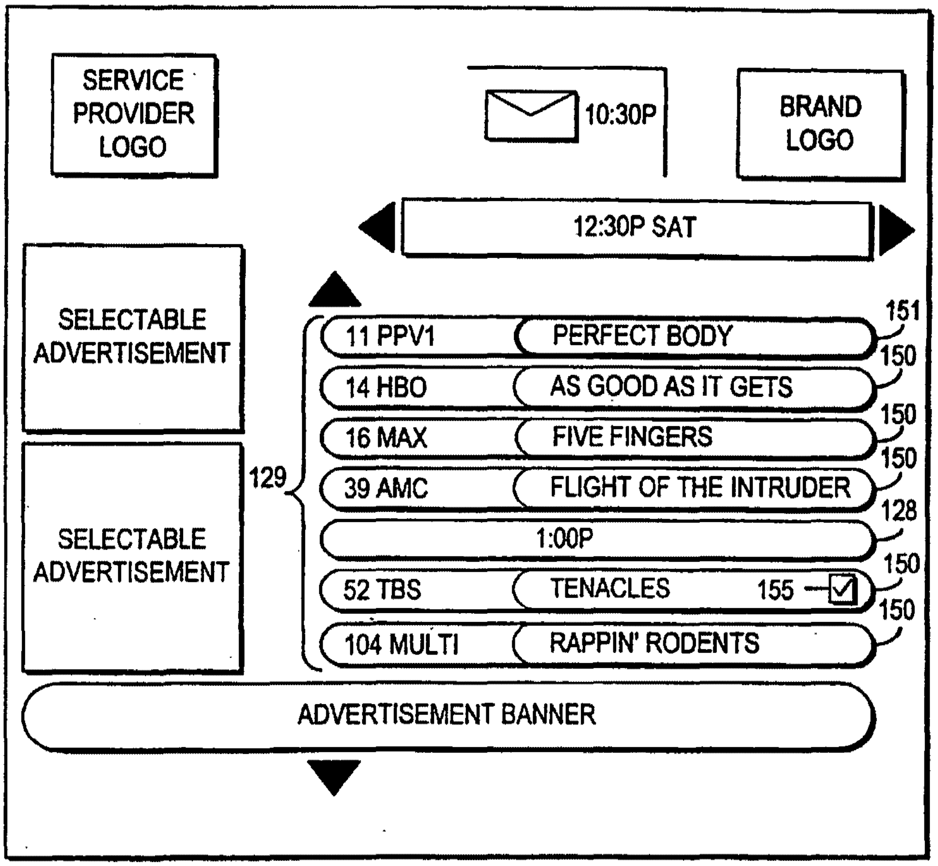

SERVICE PROVIDER LOGO
10:30P
BRAND LOGO
12:30P SAT
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
129
11 PPV1
PERFECT BODY
151
150
14 HBO
AS GOOD AS IT GETS
150
16 MAX
FIVE FINGERS
150
39 AMC
FLIGHT OF THE INTRUDER
128
1:00P
150
52 TBS
TENACLES
155
150
104 MULTI
RAPPIN' RODENTS
ADVERTISEMENT BANNER

FIG. 8a

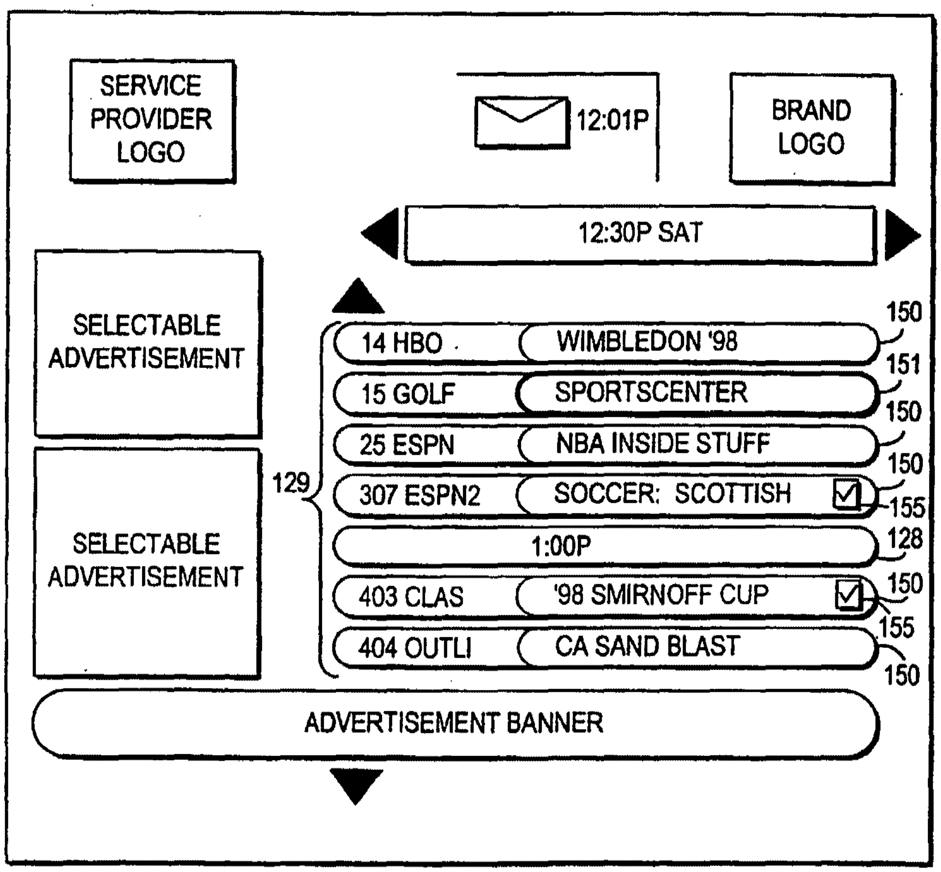

SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
12:30P SAT
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
129
14 HBO
WIMBLEDON '98
150
151
15 GOLF
SPORTSCENTER
150
25 ESPN
NBA INSIDE STUFF
150
307 ESPN2
SOCCER: SCOTTISH
155
1:00P
128
403 CLAS
'98 SMIRNOFF CUP
150
155
404 OUTLI
CA SAND BLAST
150
ADVERTISEMENT BANNER

FIG. 8b

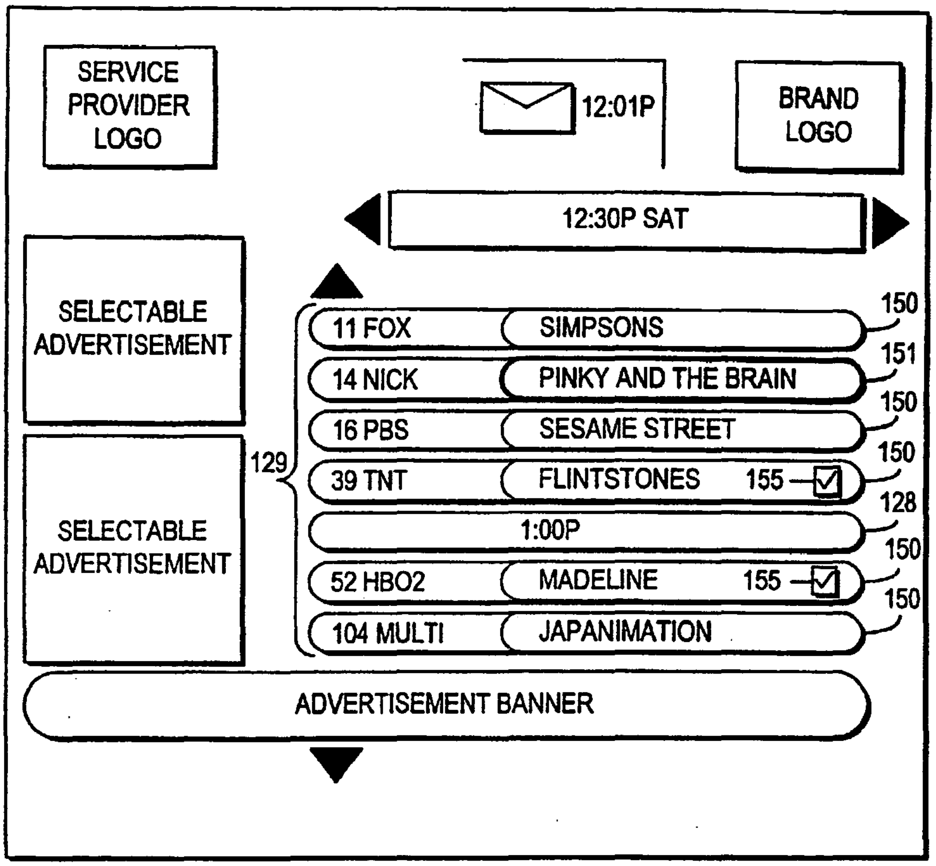

SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
12:30P SAT
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
129
11 FOX
SIMPSONS
150
14 NICK
PINKY AND THE BRAIN
151
16 PBS
SESAME STREET
150
39 TNT
FLINTSTONES
155
150
1:00P
128
52 HBO2
MADELINE
155
150
104 MULTI
JAPANIMATION
150
ADVERTISEMENT BANNER

FIG. 8c

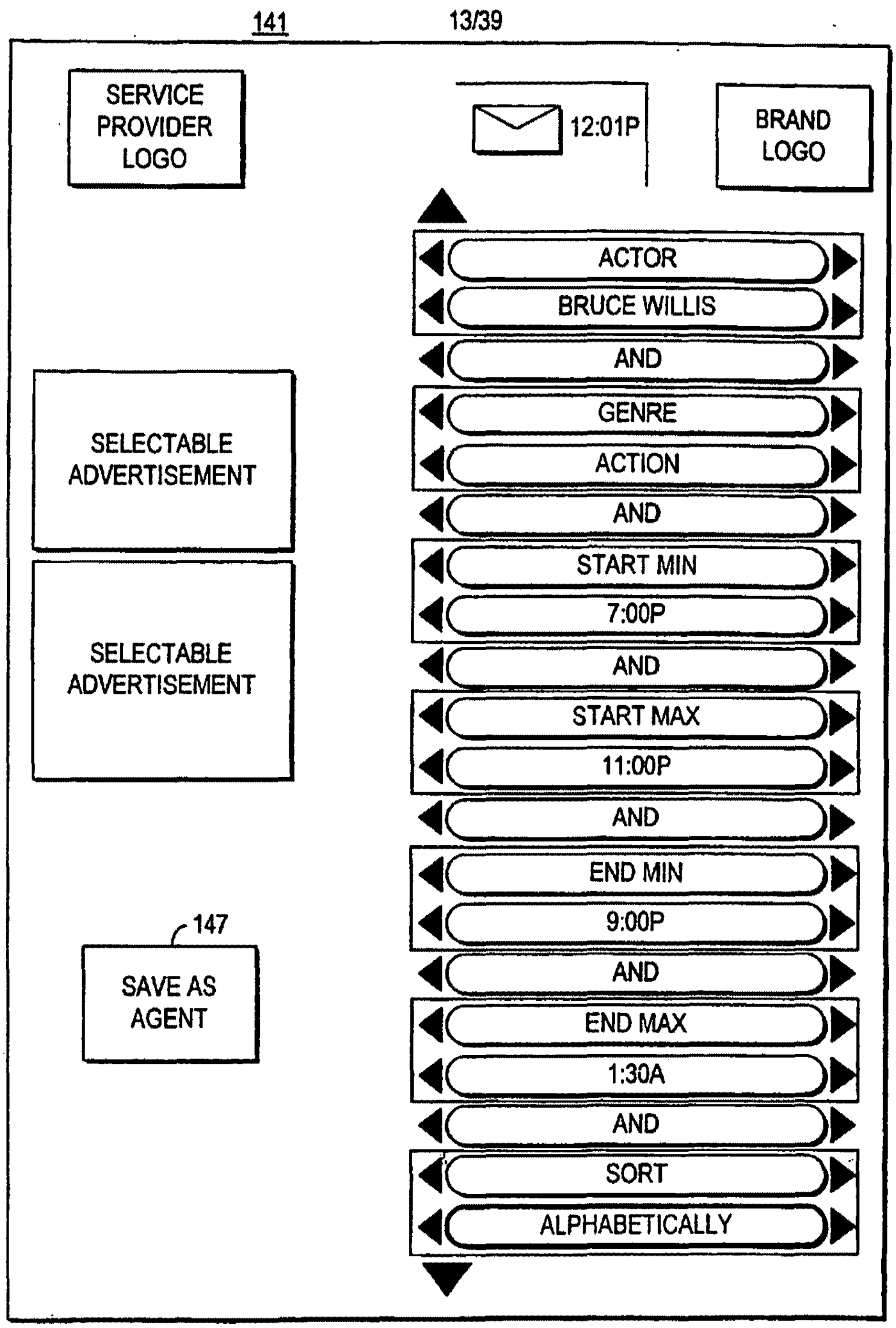

141
SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
ACTOR
BRUCE WILLIS
AND
GENRE
ACTION
AND
START MIN
7:00P
AND
START MAX
11:00P
AND
END MIN
9:00P
AND
END MAX
1:30A
AND
SORT
ALPHABETICALLY
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
147
SAVE AS AGENT

FIG. 9a

149

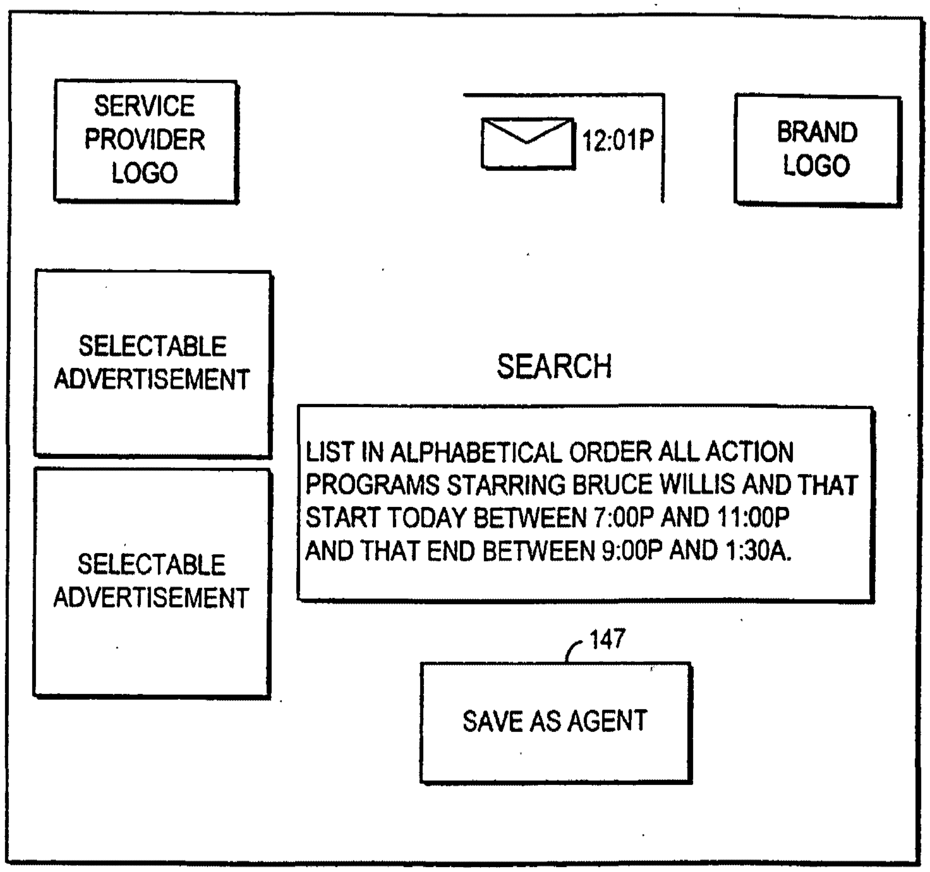

SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
SEARCH
LIST IN ALPHABETICAL ORDER ALL ACTION PROGRAMS STARRING BRUCE WILLIS AND THAT START TODAY BETWEEN 7:00P AND 11:00P AND THAT END BETWEEN 9:00P AND 1:30A.
147
SAVE AS AGENT

FIG. 9b

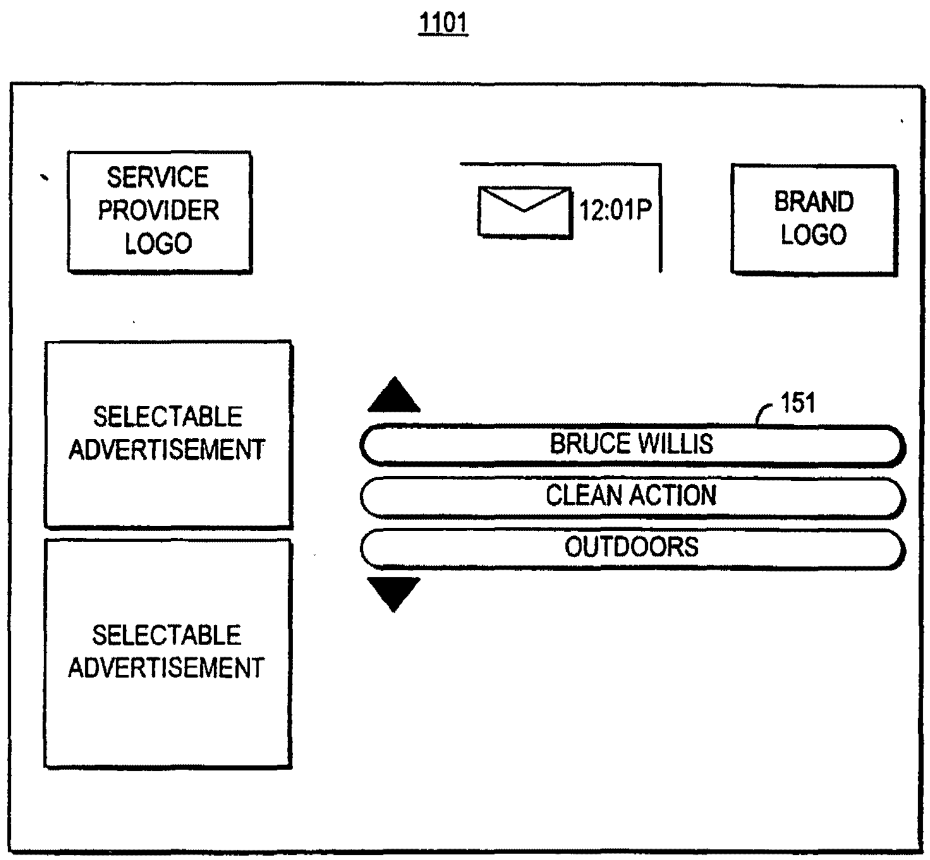

1101
SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
SELECTABLE ADVERTISEMENT
151
BRUCE WILLIS
CLEAN ACTION
OUTDOORS
SELECTABLE ADVERTISEMENT

FIG. 10

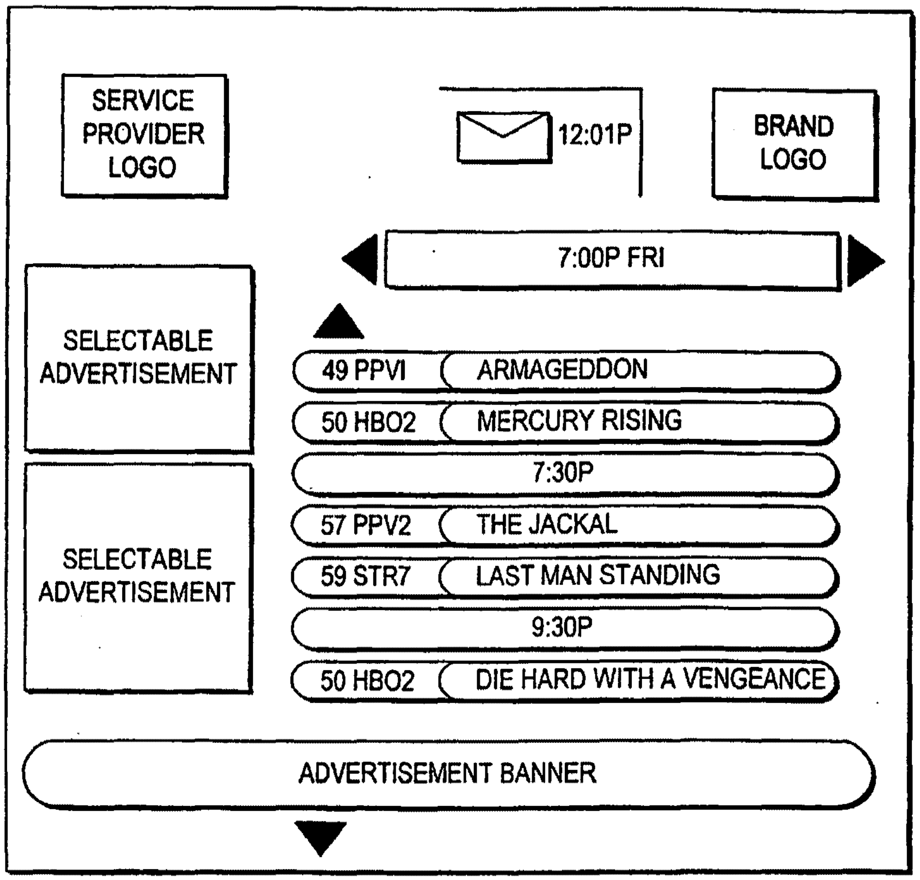

SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
7:00P FRI
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
49 PPVI
ARMAGEDDON
50 HBO2
MERCURY RISING
7:30P
57 PPV2
THE JACKAL
59 STR7
LAST MAN STANDING
9:30P
50 HBO2
DIE HARD WITH A VENGEANCE
ADVERTISEMENT BANNER

FIG. 11

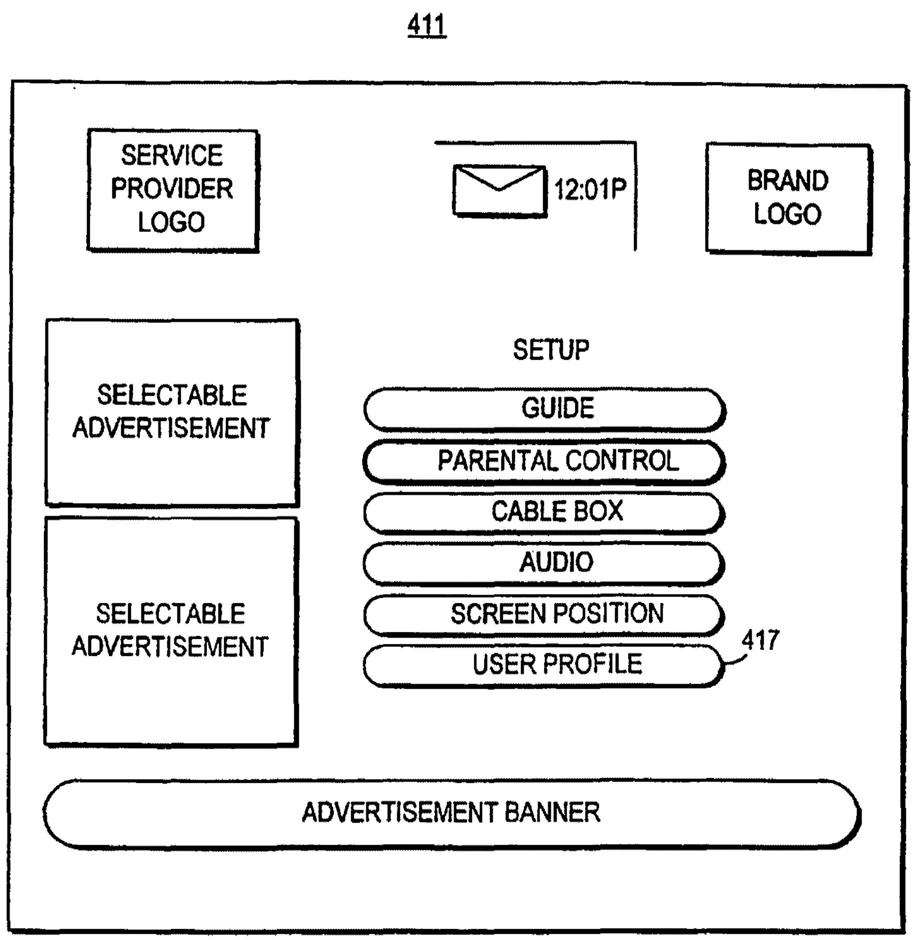

411
SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
SELECTABLE ADVERTISEMENT
SETUP
GUIDE
PARENTAL CONTROL
CABLE BOX
AUDIO
SCREEN POSITION
USER PROFILE
417
SELECTABLE ADVERTISEMENT
ADVERTISEMENT BANNER

FIG. 12

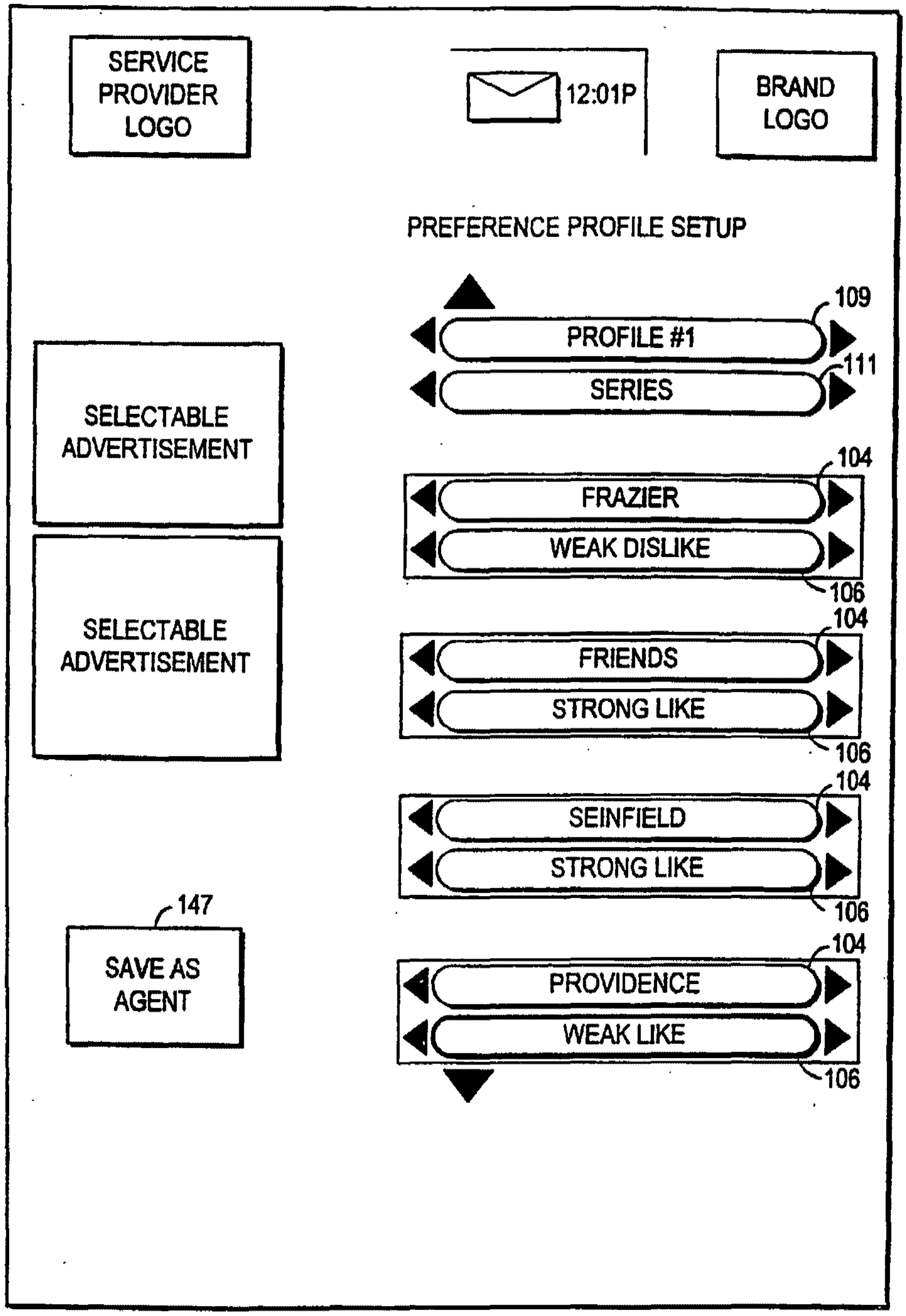


FIG. 13a

SUBSTITUTE SHEET (RULE 26)

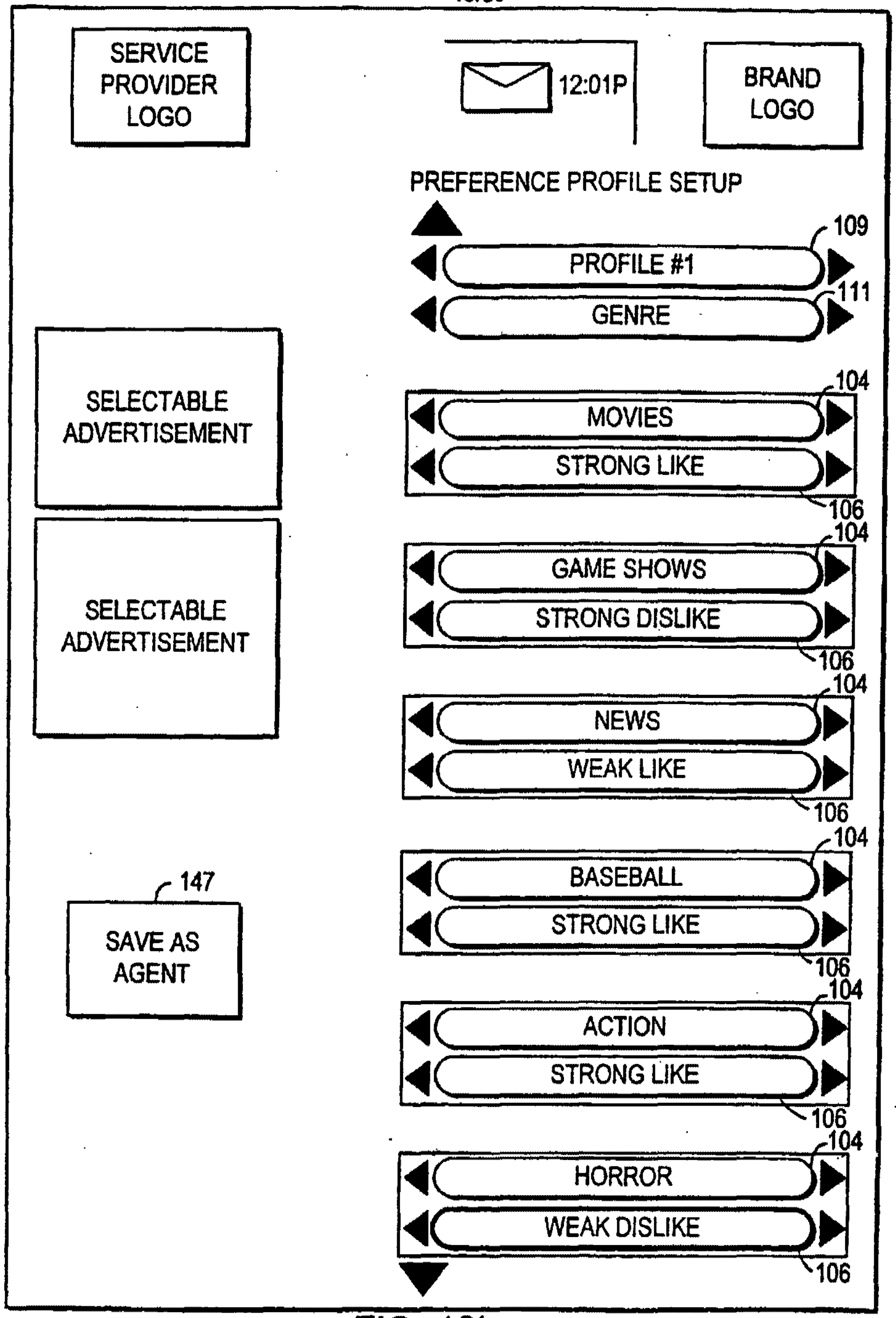

SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
PREFERENCE PROFILE SETUP
109
PROFILE #1
111
GENRE
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
104
MOVIES
STRONG LIKE
106
GAME SHOWS
STRONG DISLIKE
NEWS
WEAK LIKE
147
SAVE AS AGENT
BASEBALL
STRONG LIKE
ACTION
STRONG LIKE
HORROR
WEAK DISLIKE

FIG. 13b

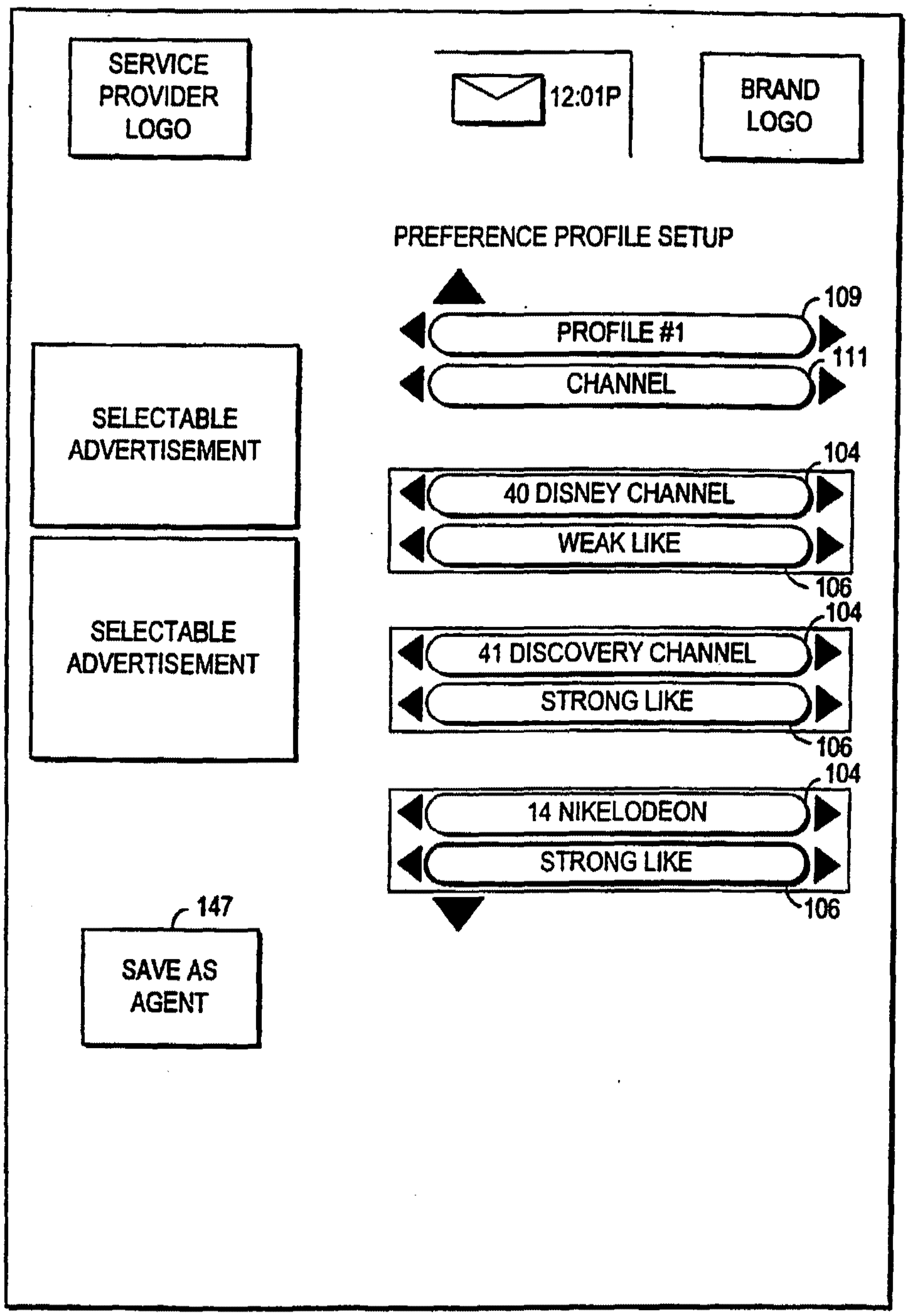


FIG. 13c

SUBSTITUTE SHEET (RULE 26)

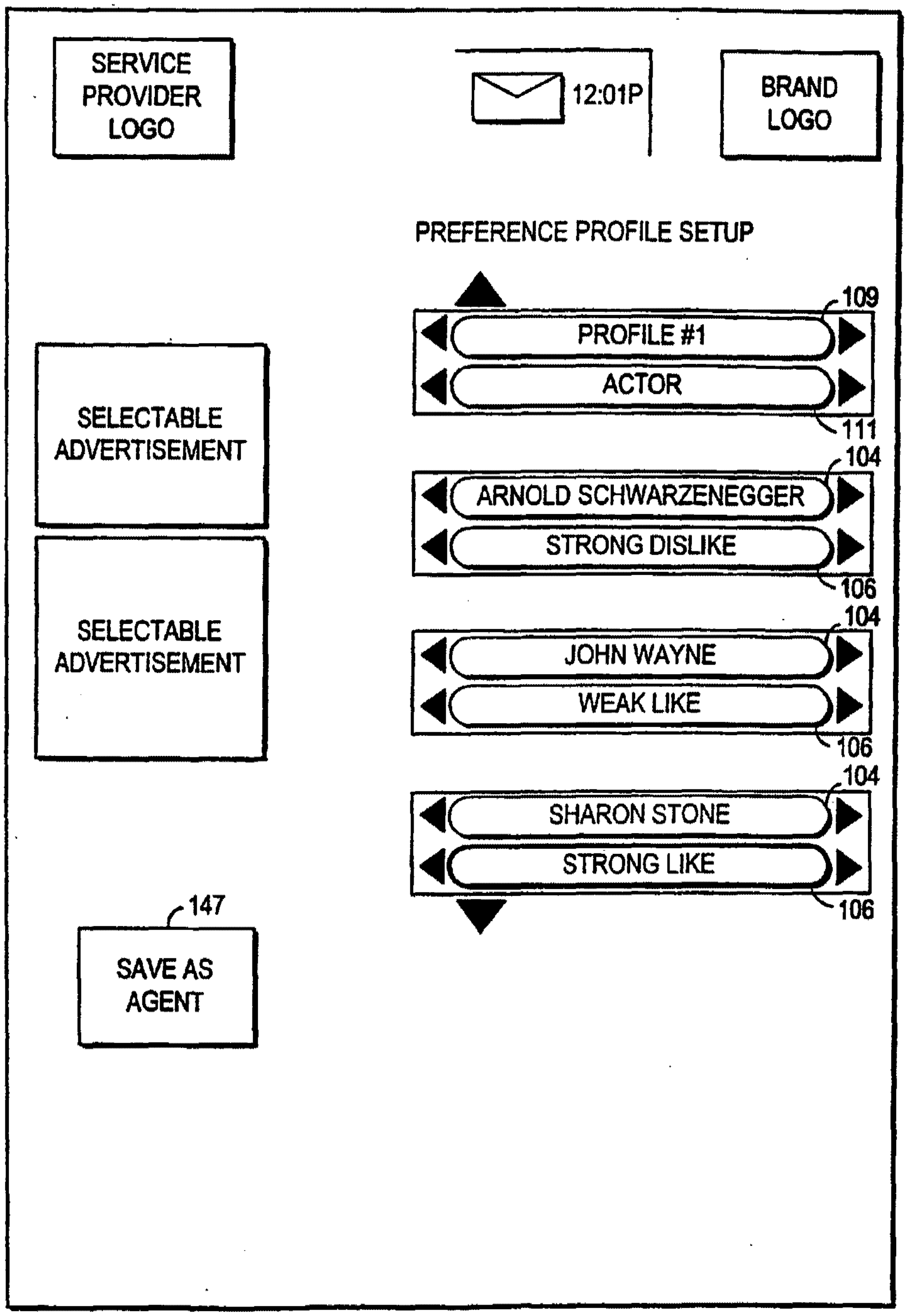


FIG. 13d

SUBSTITUTE SHEET (RULE 26)

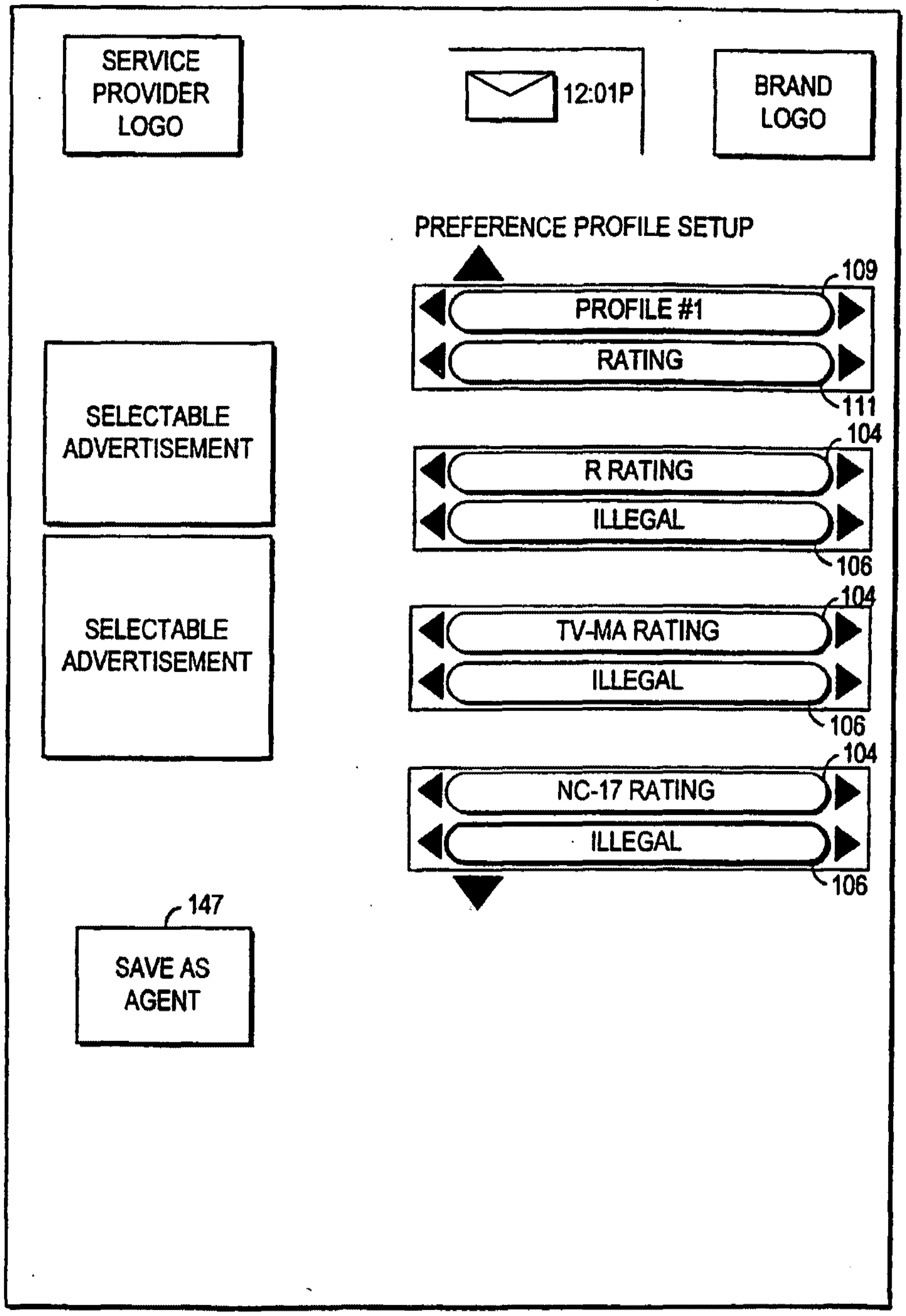


FIG. 13e

SUBSTITUTE SHEET (RULE 26)

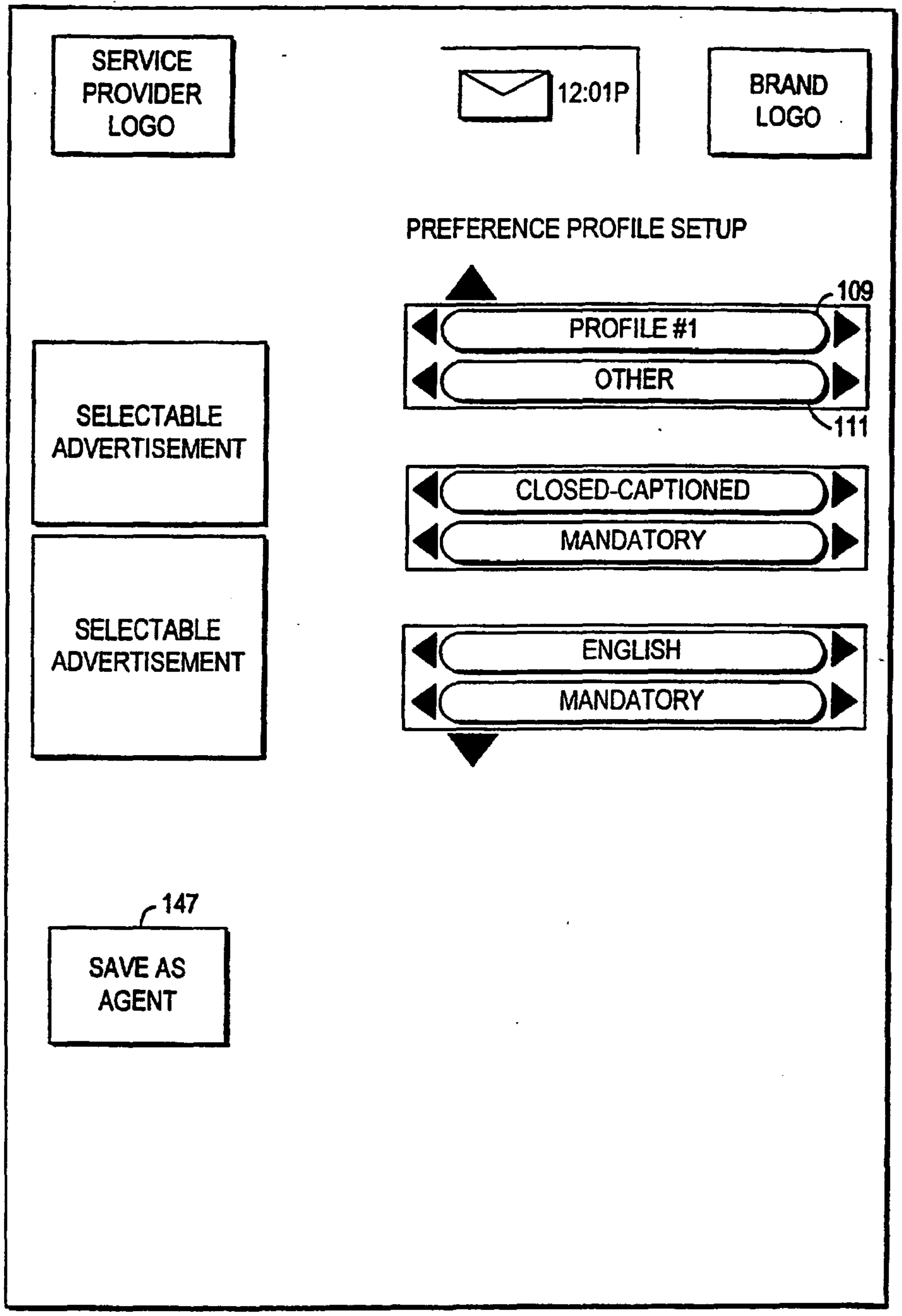

SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
PREFERENCE PROFILE SETUP
109
PROFILE #1
OTHER
111
SELECTABLE ADVERTISEMENT
CLOSED-CAPTIONED
MANDATORY
SELECTABLE ADVERTISEMENT
ENGLISH
MANDATORY
147
SAVE AS AGENT

FIG. 13f

130

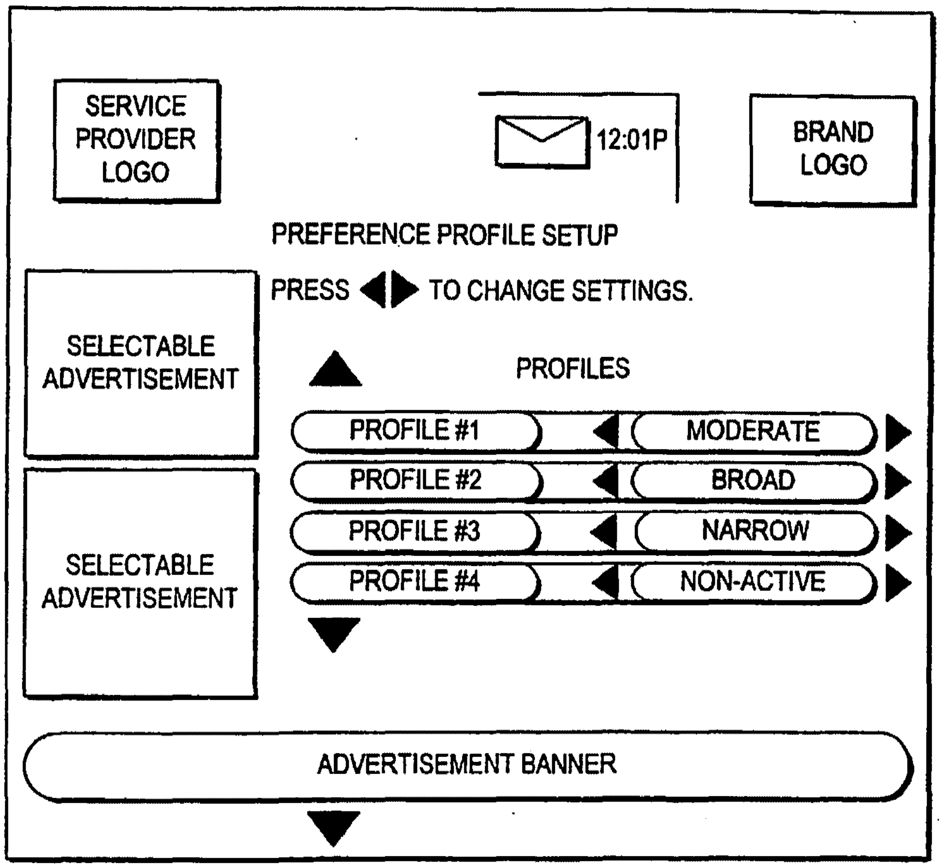

SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
PREFERENCE PROFILE SETUP
PRESS TO CHANGE SETTINGS.
SELECTABLE ADVERTISEMENT
PROFILES
PROFILE #1
MODERATE
PROFILE #2
BROAD
PROFILE #3
NARROW
PROFILE #4
NON-ACTIVE
SELECTABLE ADVERTISEMENT
ADVERTISEMENT BANNER

FIG. 14

SUBSTITUTE SHEET (RULE 26)

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

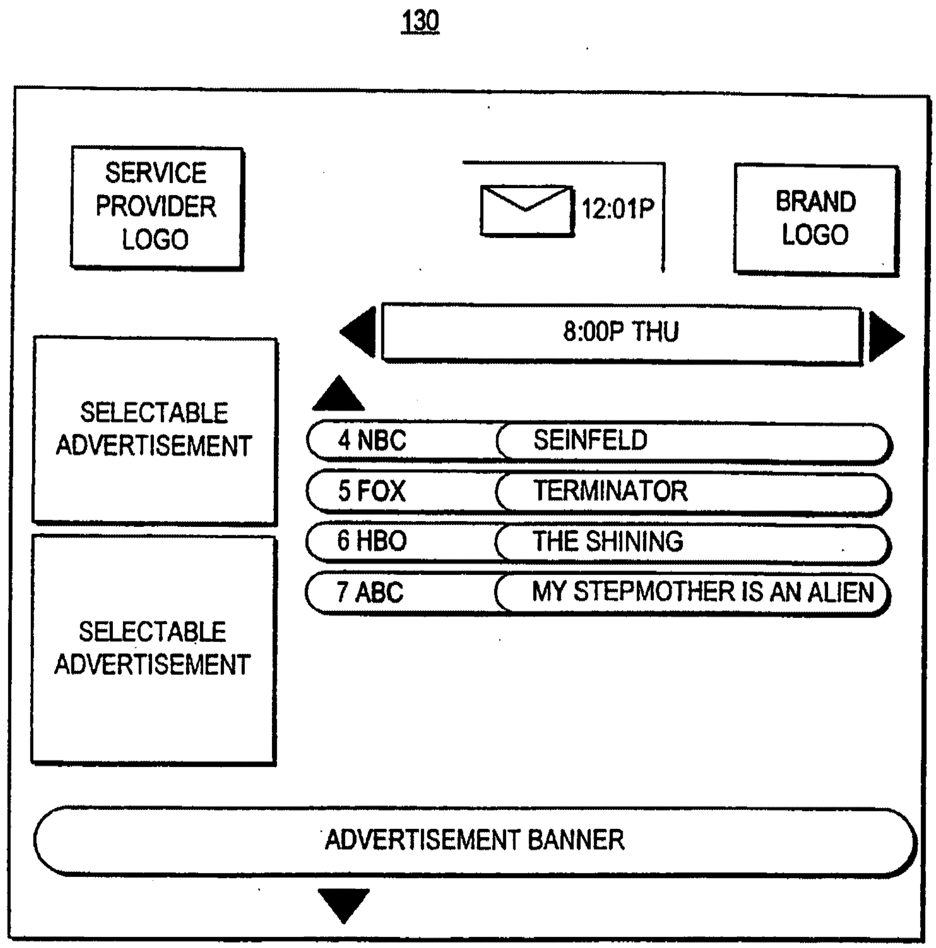

130
SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
8:00P THU
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
4 NBC
SEINFELD
5 FOX
TERMINATOR
6 HBO
THE SHINING
7 ABC
MY STEPMOTHER IS AN ALIEN
ADVERTISEMENT BANNER

FIG. 16a

130

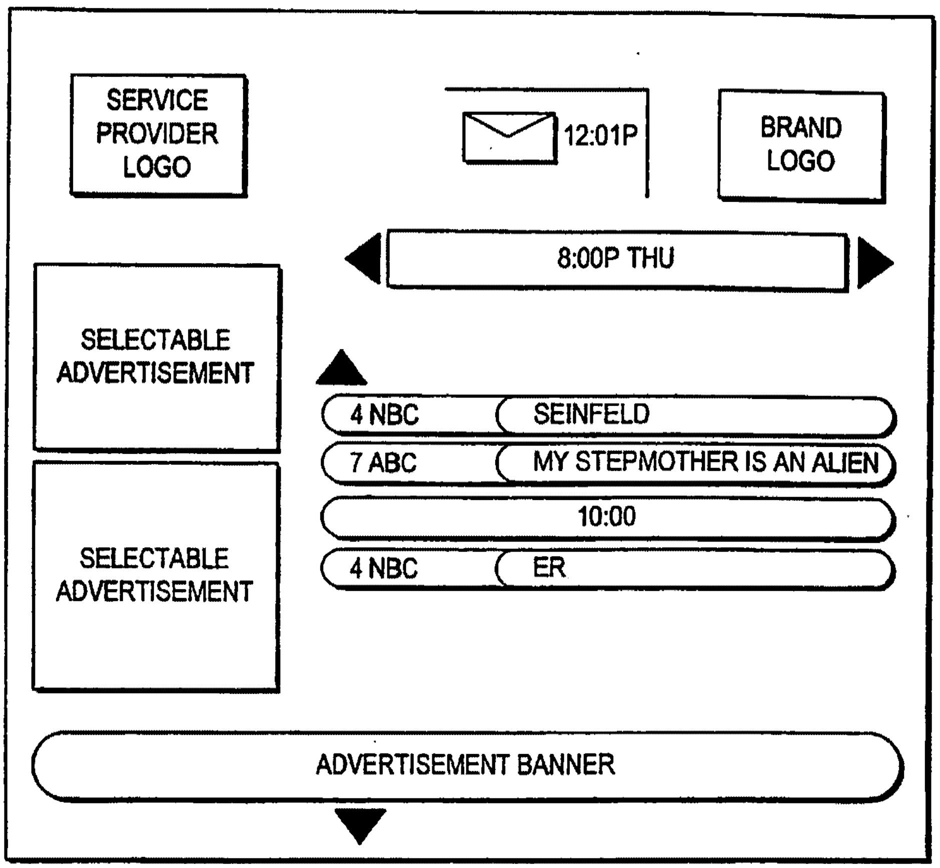

SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
8:00P THU
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
4 NBC
SEINFELD
7 ABC
MY STEPMOTHER IS AN ALIEN
10:00
4 NBC
ER
ADVERTISEMENT BANNER

FIG. 16b

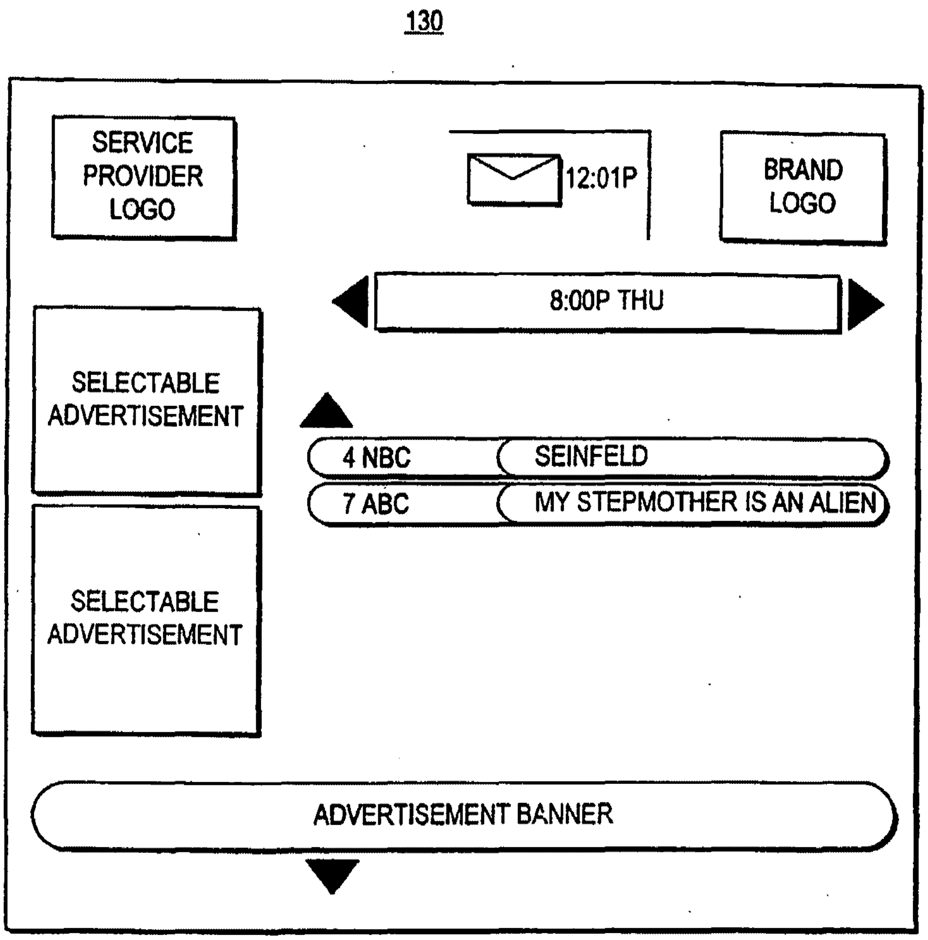

130
SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
8:00P THU
SELECTABLE ADVERTISEMENT
4 NBC
SEINFELD
7 ABC
MY STEPMOTHER IS AN ALIEN
SELECTABLE ADVERTISEMENT
ADVERTISEMENT BANNER

FIG. 16c

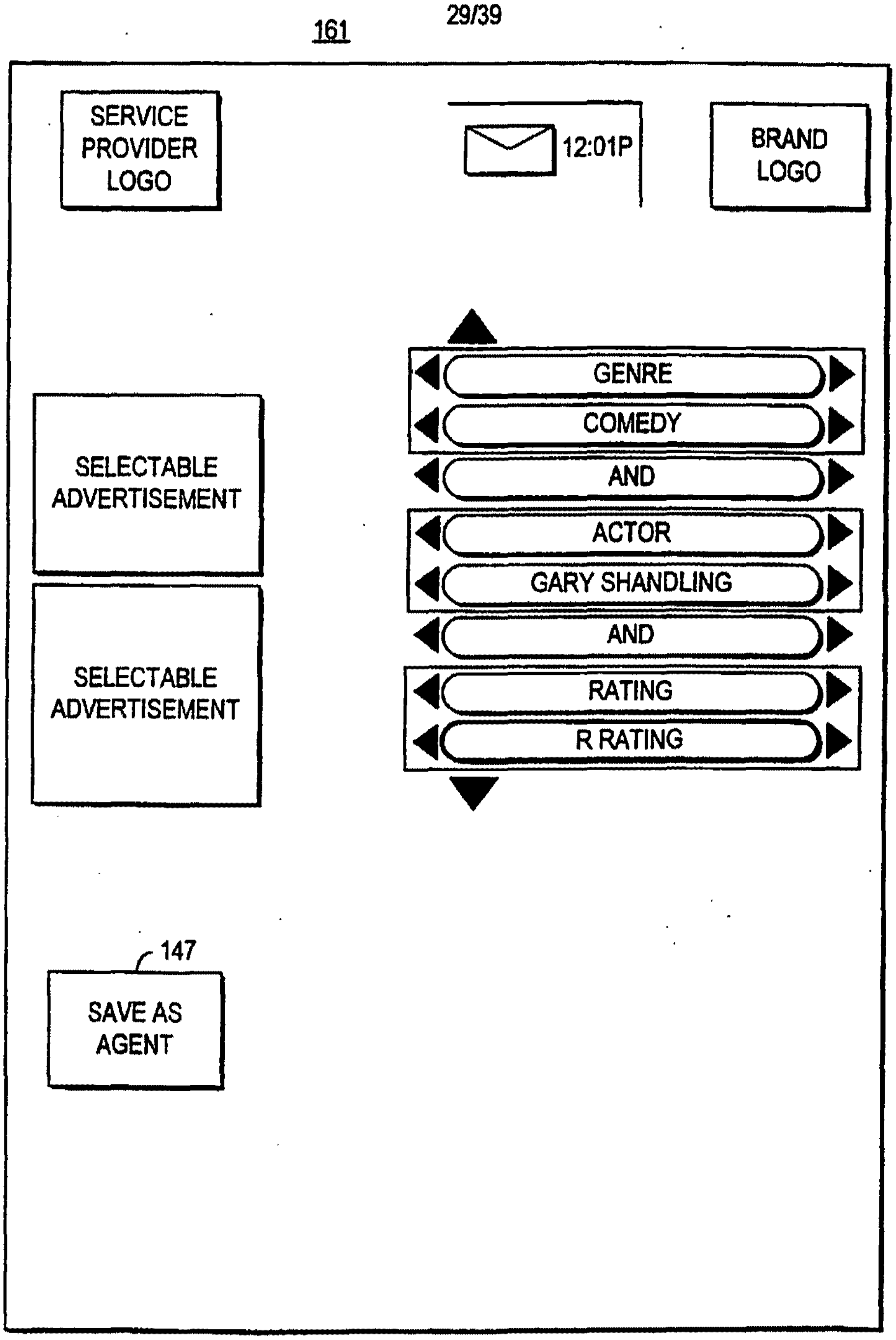

161
SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
GENRE
COMEDY
AND
ACTOR
GARY SHANDLING
AND
RATING
R RATING
147
SAVE AS AGENT

FIG. 17a

169

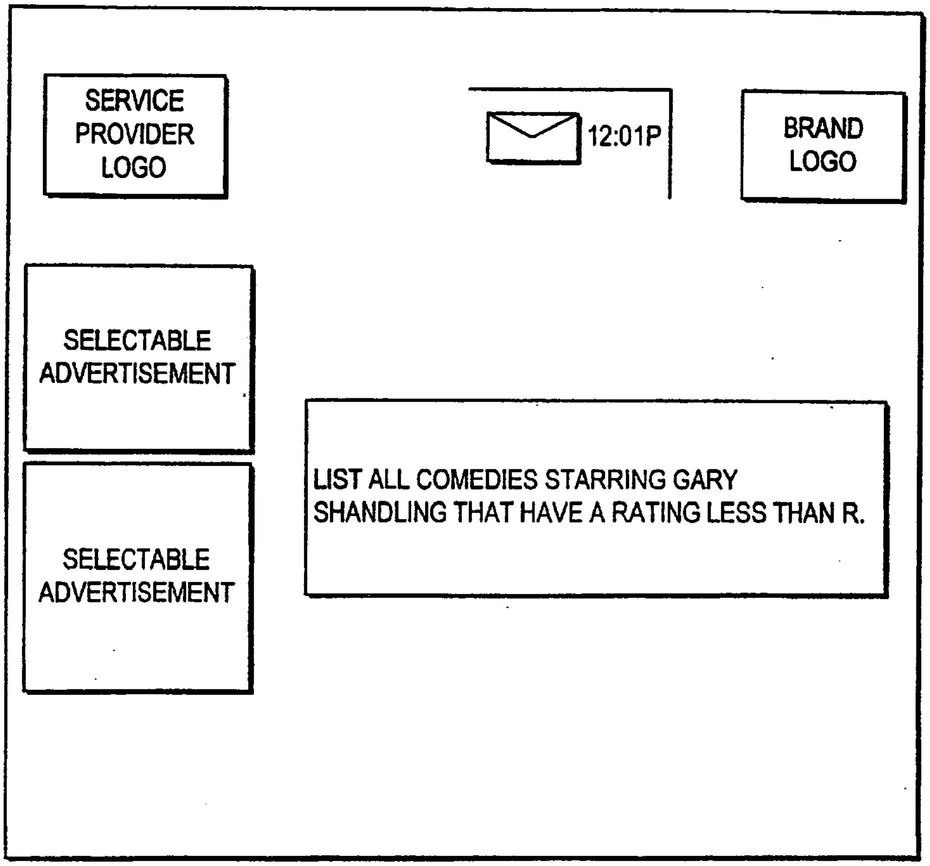

SERVICE PROVIDER LOGO
12:01P
BRAND LOGO
SELECTABLE ADVERTISEMENT
SELECTABLE ADVERTISEMENT
LIST ALL COMEDIES STARRING GARY SHANDLING THAT HAVE A RATING LESS THAN R.

FIG. 17b

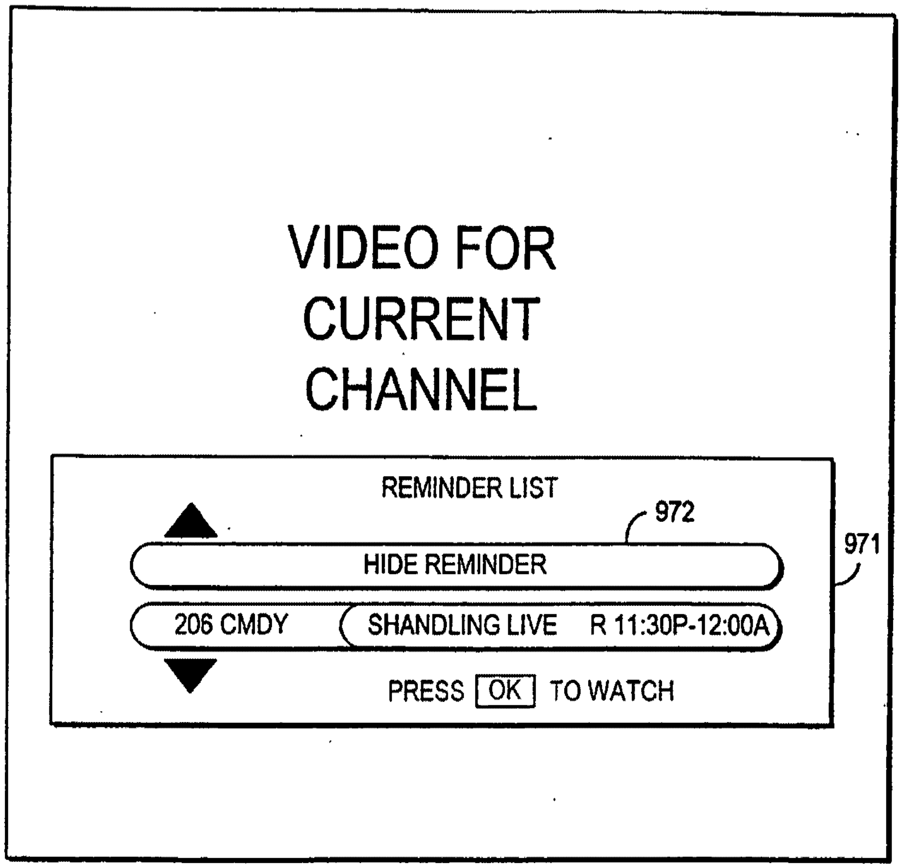

VIDEO FOR
CURRENT
CHANNEL
REMINDER LIST
972
971
HIDE REMINDER
206 CMDY
SHANDLING LIVE
R 11:30P-12:00A
PRESS OK TO WATCH

FIG. 18

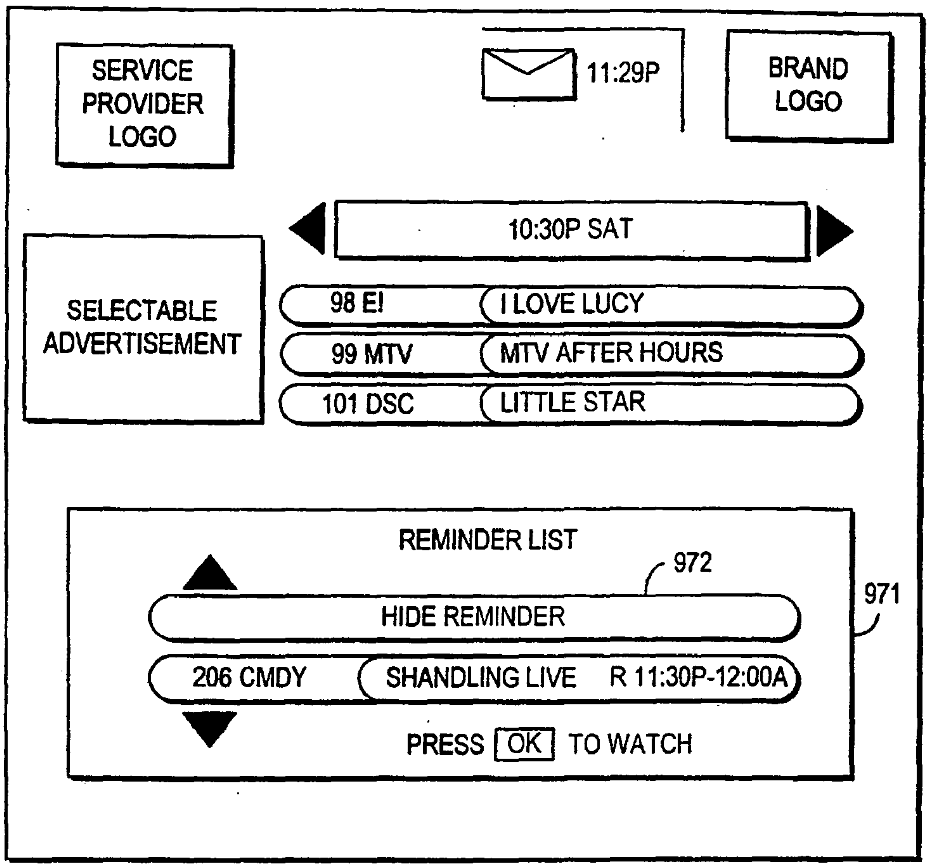

SERVICE PROVIDER LOGO
11:29P
BRAND LOGO
10:30P SAT
SELECTABLE ADVERTISEMENT
98 E! I LOVE LUCY
99 MTV MTV AFTER HOURS
101 DSC LITTLE STAR
REMINDER LIST
972
971
HIDE REMINDER
206 CMDY SHANDLING LIVE R 11:30P-12:00A
PRESS OK TO WATCH

FIG. 19

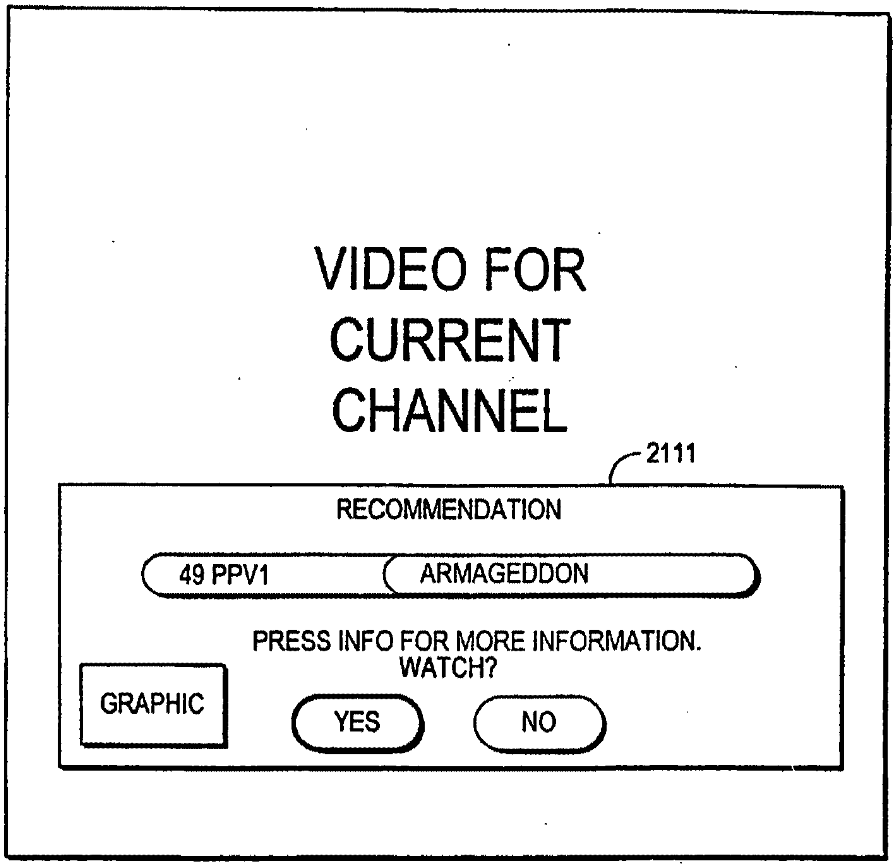

VIDEO FOR
CURRENT
CHANNEL
2111
RECOMMENDATION
49 PPV1
ARMAGEDDON
PRESS INFO FOR MORE INFORMATION.
WATCH?
GRAPHIC
YES
NO

FIG. 20a

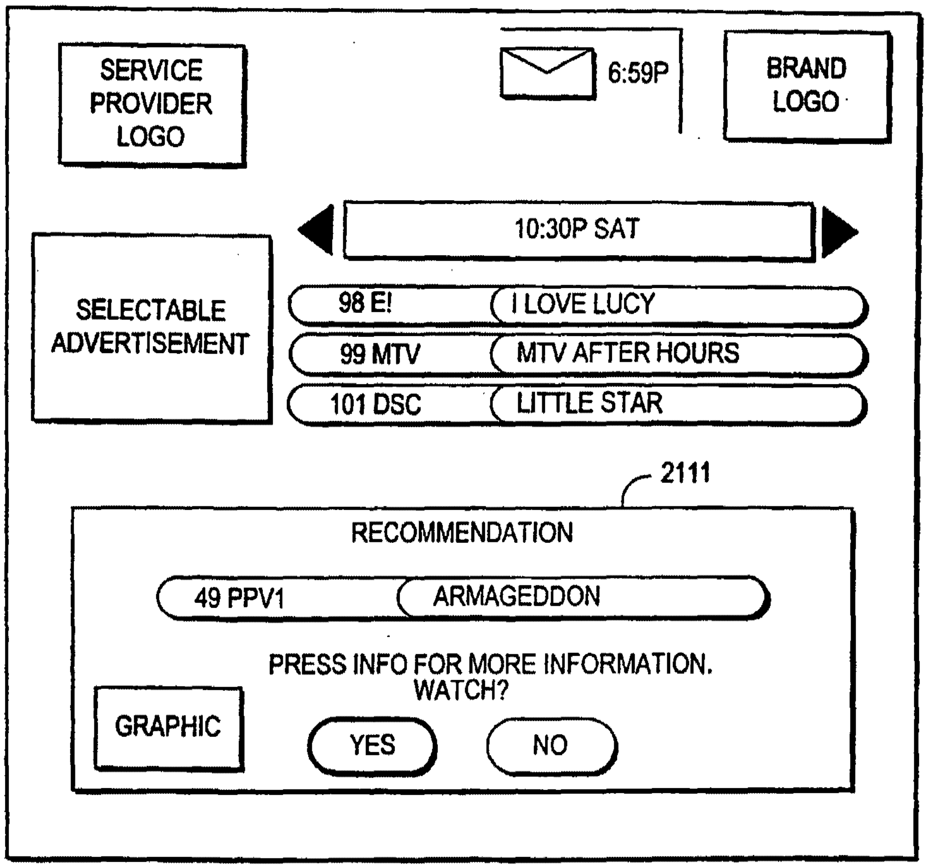

SERVICE PROVIDER LOGO
6:59P
BRAND LOGO
10:30P SAT
SELECTABLE ADVERTISEMENT
98 E!
I LOVE LUCY
99 MTV
MTV AFTER HOURS
101 DSC
LITTLE STAR
2111
RECOMMENDATION
49 PPV1
ARMAGEDDON
PRESS INFO FOR MORE INFORMATION.
WATCH?
GRAPHIC
YES
NO

FIG. 20b

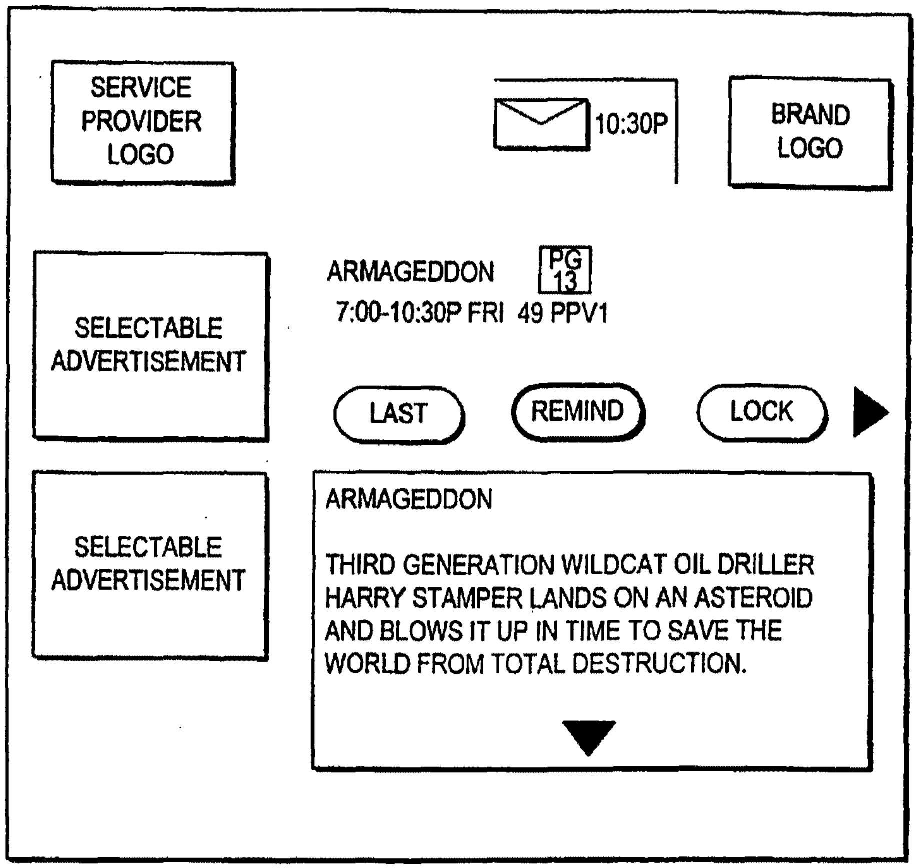

SERVICE
PROVIDER
LOGO
10:30P
BRAND
LOGO
SELECTABLE
ADVERTISEMENT
ARMAGEDDON
PG
13
7:00-10:30P FRI 49 PPV1
LAST
REMIND
LOCK
SELECTABLE
ADVERTISEMENT
ARMAGEDDON
THIRD GENERATION WILDCAT OIL DRILLER
HARRY STAMPER LANDS ON AN ASTEROID
AND BLOWS IT UP IN TIME TO SAVE THE
WORLD FROM TOTAL DESTRUCTION.

FIG. 20c

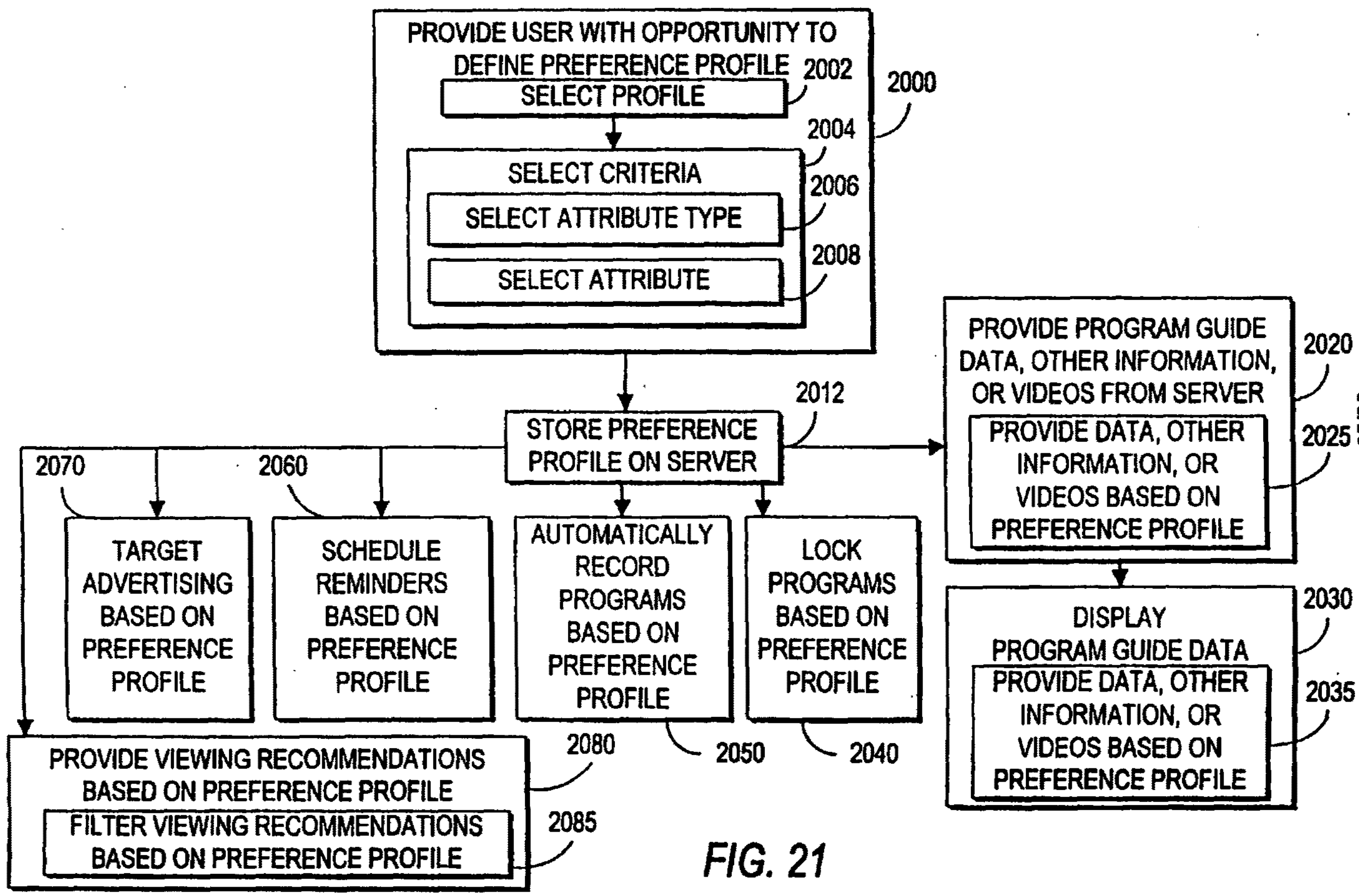


FIG. 21

SUBSTITUTE SHEET (RULE 26)

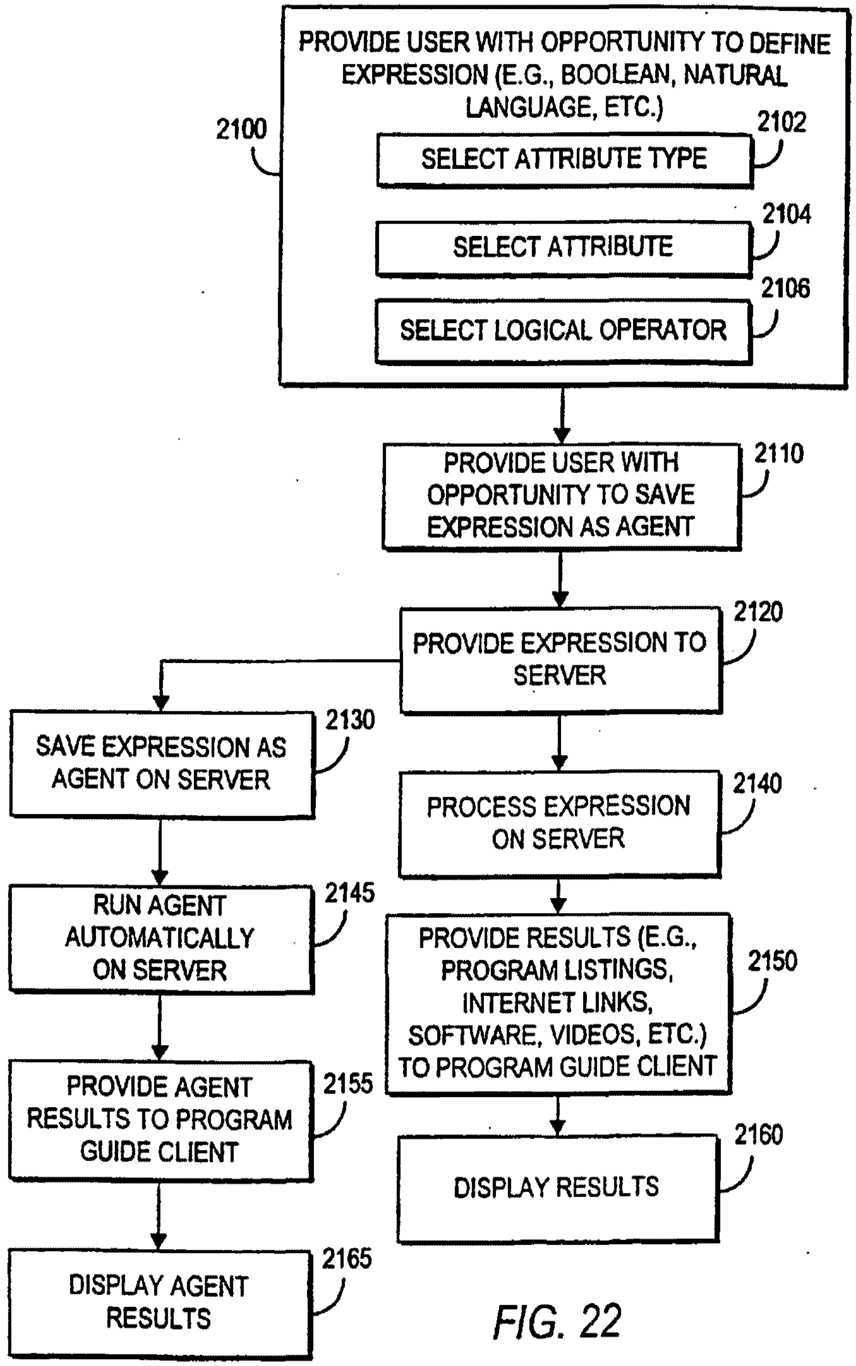

PROVIDE USER WITH OPPORTUNITY TO DEFINE EXPRESSION (E.G., BOOLEAN, NATURAL LANGUAGE, ETC.)
2100
2102
SELECT ATTRIBUTE TYPE
2104
SELECT ATTRIBUTE
2106
SELECT LOGICAL OPERATOR
PROVIDE USER WITH OPPORTUNITY TO SAVE EXPRESSION AS AGENT
2110
PROVIDE EXPRESSION TO SERVER
2120
SAVE EXPRESSION AS AGENT ON SERVER
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

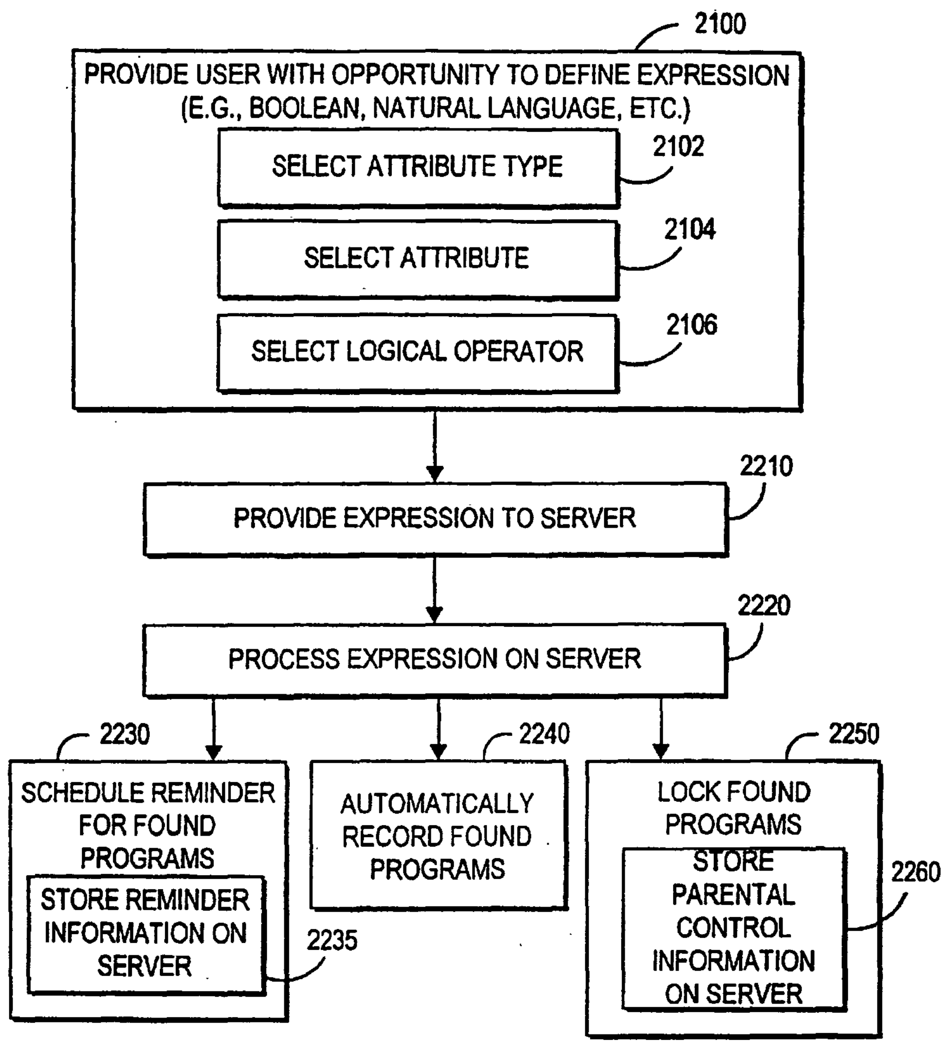

2100
PROVIDE USER WITH OPPORTUNITY TO DEFINE EXPRESSION (E.G., BOOLEAN, NATURAL LANGUAGE, ETC.)
SELECT ATTRIBUTE TYPE
2102
SELECT ATTRIBUTE
2104
SELECT LOGICAL OPERATOR
2106
PROVIDE EXPRESSION TO SERVER
2210
PROCESS EXPRESSION ON SERVER
2220
2230
SCHEDULE REMINDER FOR FOUND PROGRAMS
STORE REMINDER INFORMATION ON SERVER
2235
2240
AUTOMATICALLY RECORD FOUND PROGRAMS
2250
LOCK FOUND PROGRAMS
STORE PARENTAL CONTROL INFORMATION ON SERVER
2260

FIG. 23

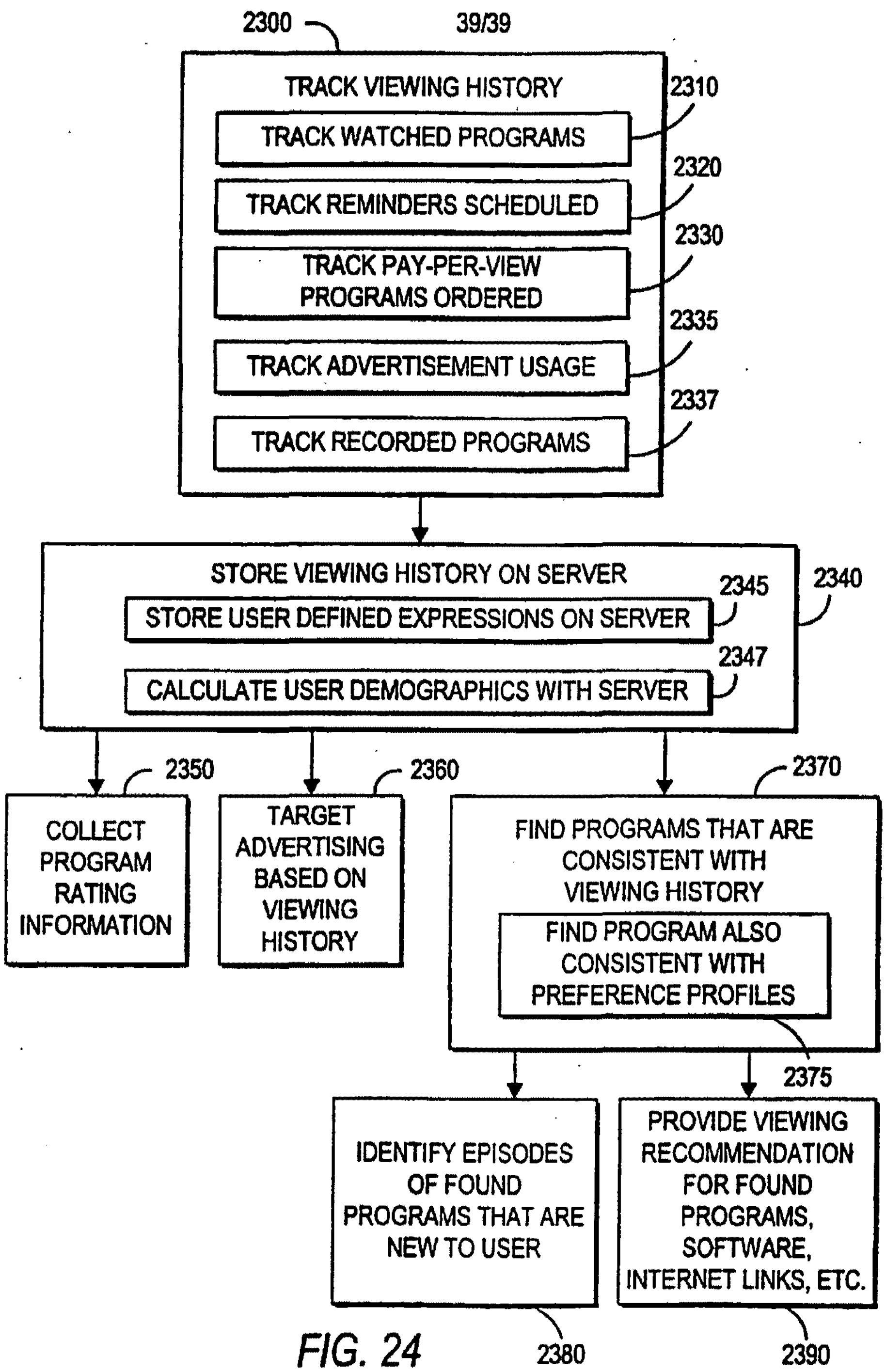

2300
TRACK VIEWING HISTORY
TRACK WATCHED PROGRAMS
2310
TRACK REMINDERS SCHEDULED
2320
TRACK PAY-PER-VIEW PROGRAMS ORDERED
2330
TRACK ADVERTISEMENT USAGE
2335
TRACK RECORDED PROGRAMS
2337
STORE VIEWING HISTORY ON SERVER
2340
STORE USER DEFINED EXPRESSIONS ON SERVER
2345
CALCULATE USER DEMOGRAPHICS WITH 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

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

[X] Further documents are listed in the continuation of box C.

[X] 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	Date of mailing of the international search report
18 November 1999	24/11/1999

Name and mailing address of the ISA	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	Van der Zaal, R

1

Form PCT/ISA/210 (second sheet) (July 1992)

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
F../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	11-05-1998
			EP	0932979 A	04-08-1999

Form PCT/ISA/210 (patent family annex) (July 1992)

(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.

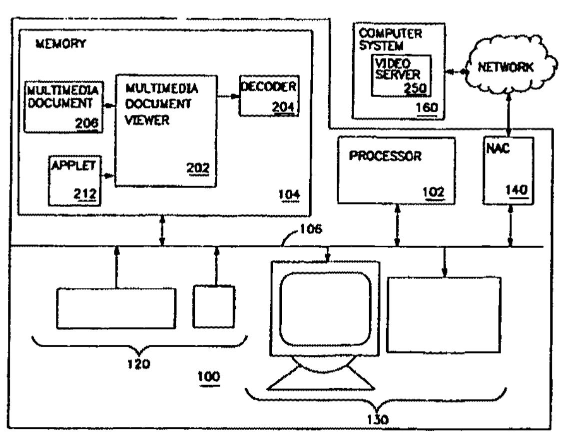


FIG. 1

EP 0 803 826 A2

Printed by Jouve, 75001 PARIS (FR)

Description

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;

}
```

<u>MsmConnect</u>

```
/*
 * @(#)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");
        }
    }
```

<u>**smcrm**</u>

```
/*
 * @(#)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.

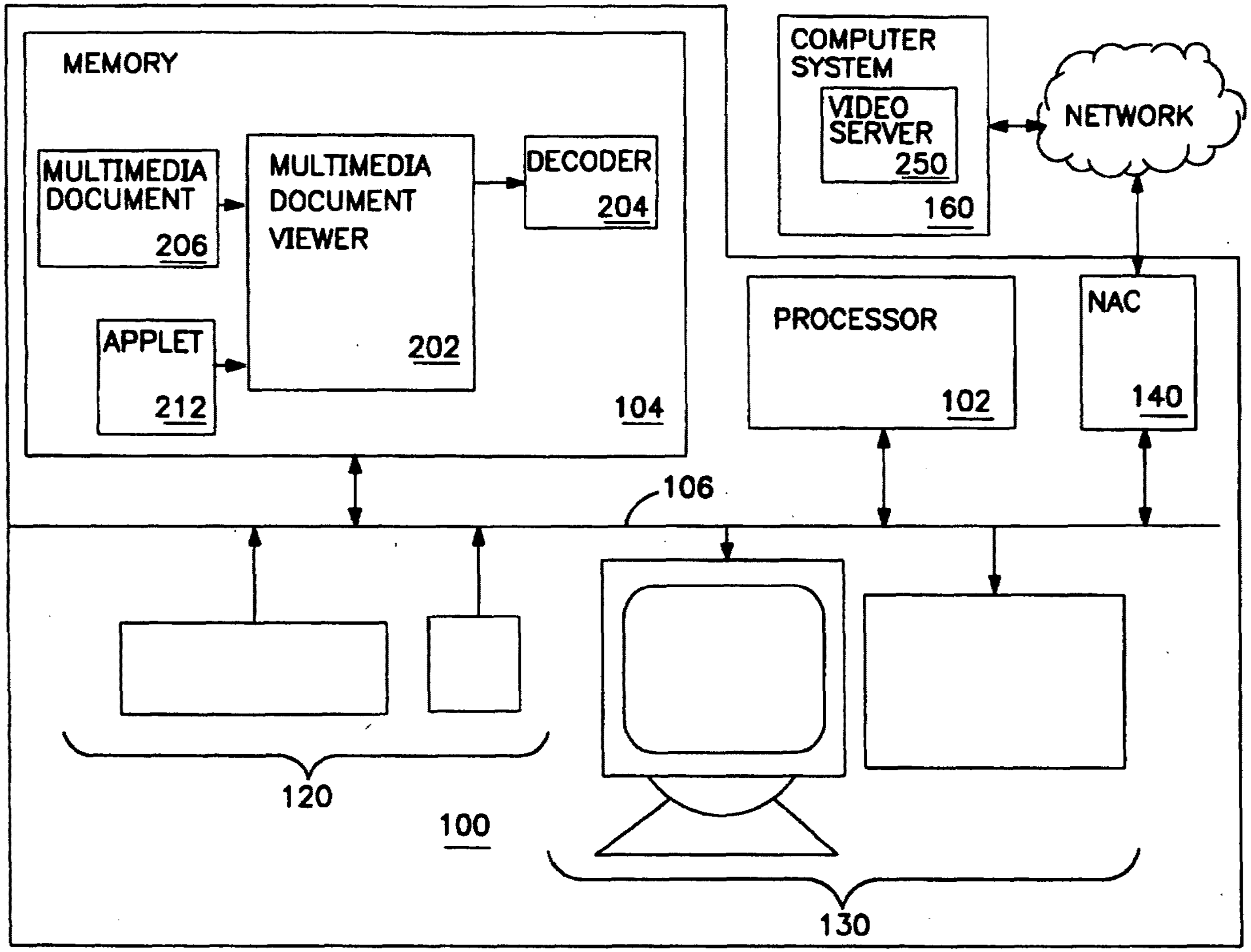

MEMORY
MULTIMEDIA DOCUMENT
206
MULTIMEDIA DOCUMENT VIEWER
202
DECODER
204
APPLET
212
104
COMPUTER SYSTEM
VIDEO SERVER
250
160
NETWORK
PROCESSOR
102
NAC
140
106
120
100
130

FIG. 1

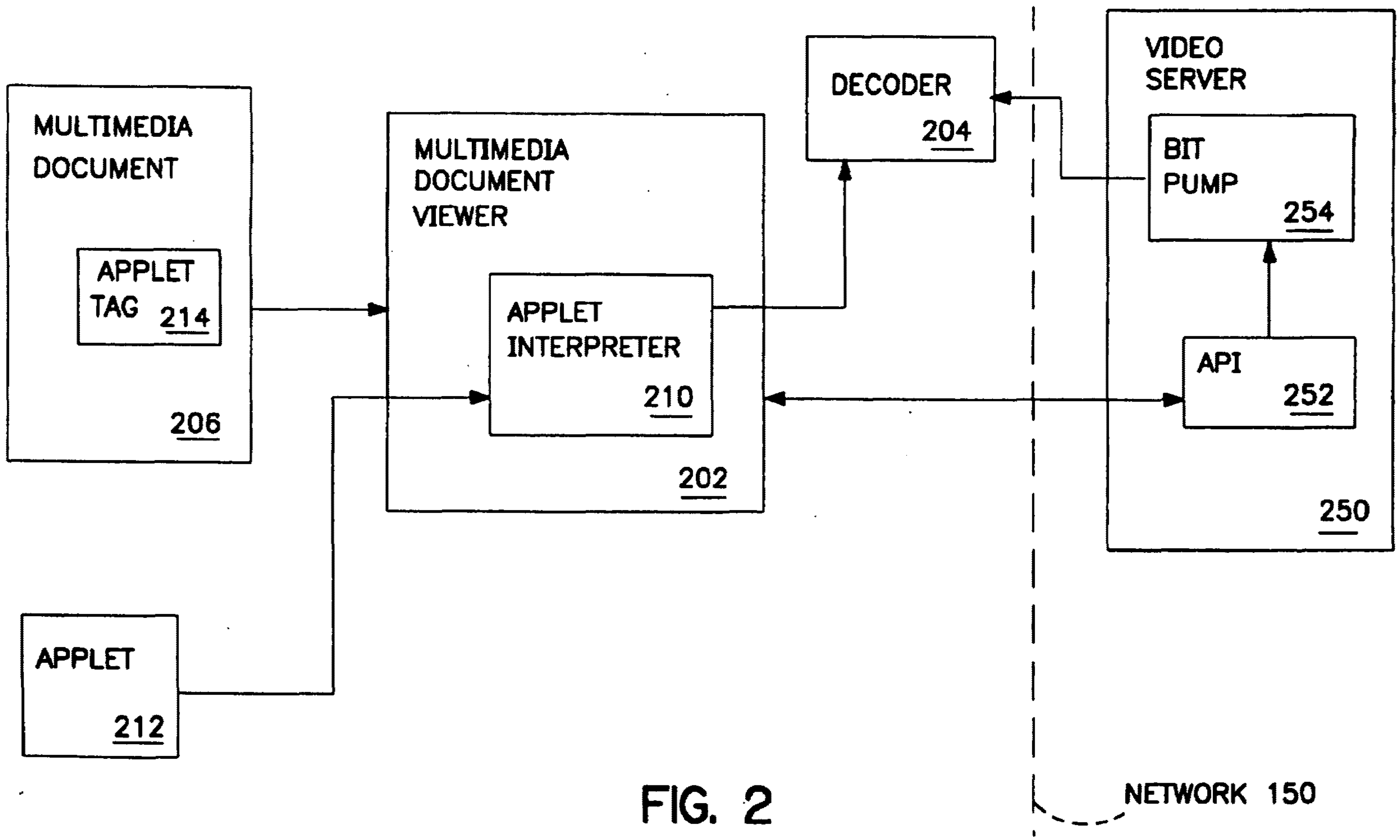
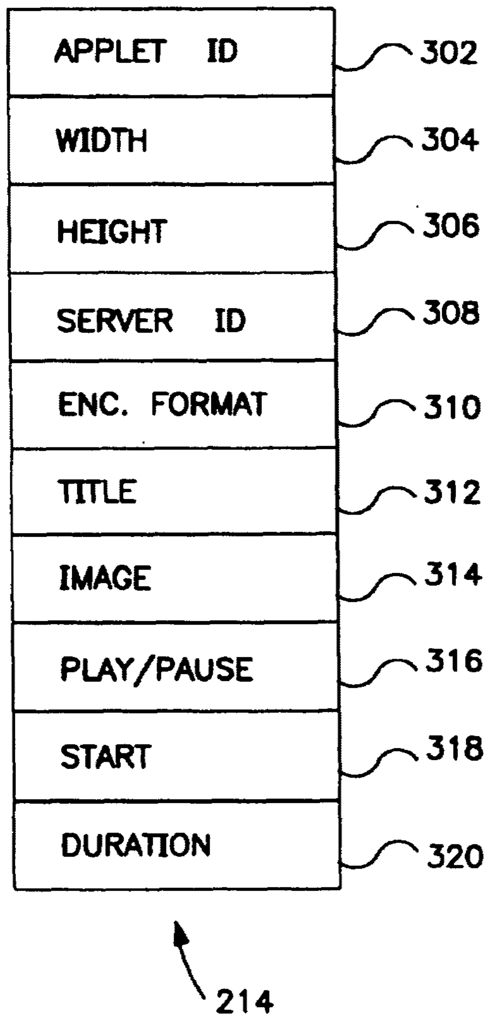

MULTIMEDIA DOCUMENT
APPLET TAG 214
206
MULTIMEDIA DOCUMENT VIEWER
APPLET INTERPRETER
210
202
DECODER
204
VIDEO SERVER
BIT PUMP
254
API
252
250
APPLET
212
NETWORK 150

FIG. 2

104

EP 0 803 826 A2

FIG. 3



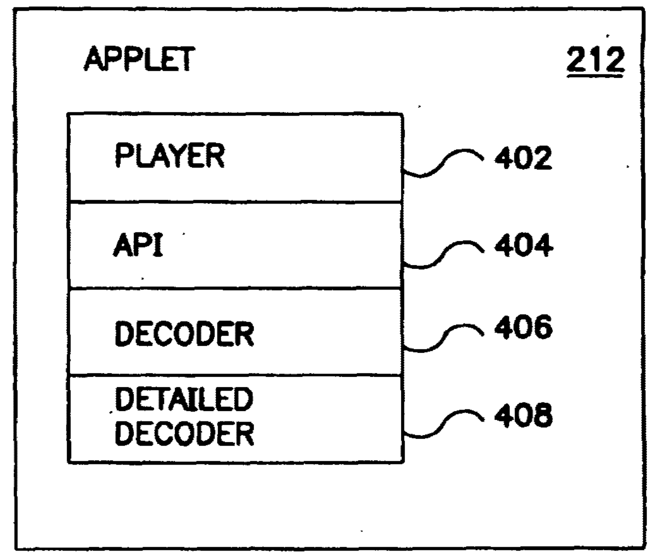


FIG. 4

OIPE JC69
NOV 2 5 2002
PATENT & TRADEMARK OFFICE

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**

RECEIVED
NOV 2 7 2002
Technology Center 2100

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

Date: November 18, 2002

Carina M. Tan
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

OIPE NOV 2 5 2002 PATENT & TRADEMARK OFFICE

#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

RECEIVED
NOV 2 7 2002
Technology Center 2100

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**."

11/26/2002 NMOHAMM1 00000101 09225198
01 FC:2251 55.00 OP

AMENDED CLAIMS IN CLEAN FORM

IN THE CLAIMS:

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;

Sub B1

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

Date: November 18, 2002
(Monday)

Carina M. Tan

Carina M. Tan
Registration No. 45,769

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 <u>as corresponding registered functional capabilities,</u> 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 <u>arbitrarily complex</u> goal expression, said act of interpreting further comprising:

generating one or more sub-goals [using] <u>expressed in</u> the inter-agent language; [and]

<u>constructing a goal satisfaction plan that includes said one or more sub-goals; and</u>

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 <u>dynamically interpreting the arbitrarily complex goal expression</u> [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] <u>an</u> 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<u>, 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,</u> 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] <u>86</u>, the data wave carrier further comprising a <u>corresponding</u> signal representation of [request] <u>said one or more requests</u> 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] <u>86</u>, 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

1

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-

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,

11/15/02
Date

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.

PTO-90C (Rev. 07-01)

8 M

Office Action Summary	Application No.	Applicant(s)
	09/225,198	CHEYER ET AL.
	Examiner	**Art Unit**
	Lewis A. Bullock, Jr.	2126

-- 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 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 expressions (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.

As to claim 64-69, refer to claims 15-25 for rejection.

As to claim 70, MARTIN1 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 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, MARTIN2 teaches that facilitator engines (broker agents) are distributed across at least two computer processes (multiple broker agents in an architecture) (pg 7, pg. 16) wherein each stores a planning component (schema mapping rules) (pg. 8). It would be obvious that since the broker performs the delegation that it also has an execution component and therefore each broker agent has an execution component.

As to claim 87, MARTIN1 teaches a 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). 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.

As to claim 88, MARTIN1 teaches a 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). 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.

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. 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.

Response to Arguments

4. Applicant's arguments with respect to claims 1-89 have been considered but are moot in view of the new ground(s) of 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.

ALVIN OBERLEY
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

lab
February 21, 2003

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 2126	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-5,802,396	09-1998	Gray, Thomas A.	710/20
	B	US-5,638,494	06-1997	Pinard et al.	709/202
	C	US-			
	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	
	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.

OIPE JC109 AUG 1 3 2002 PATENT & TRADEMARK OFFICE

INFORMATION DISCLOSURE STATEMENT BY APPLICANT
Form PTO-1449 (Modified)
(Use several sheets if necessary)

Sheet	1	of	2

COMPLETE IF KNOWN	
Application Number	09/225,798
Confirmation Number	
Filing Date	January 5, 1999
First Named Inventor	Cheyer
Group Art Unit	2755
Examiner Name	Unassigned
Attorney Docket No.	59501-8016.US01

RECEIVED AUG 1 5 2002 Technology Center 2100

U.S. PATENT DOCUMENTS

Examiner Initials	Cite No.	U.S. Patent or Application NUMBER	Kind Code (if known)	Name of Patentee or Inventor of Cited Document	Date of Publication or Filing Date of Cited Document	Pages, Columns, Lines, Where Relevant Figures Appear
fab	1	5,197,005		Schwartz et al.	3/23/93	
fab	2	5,386,556		Hedin et al.	1/31/95	
fab	3	5,434,777		Luciw	7/18/95	
fab	4	5,519,608		Kupiec	5/21/96	
fab	5	5,608,624		Luciw	3/4/97	
fab	6	5,721,938		Stuckey	2/24/98	
fab	7	5,729,659		Potter	3/17/98	
fab	8	5,748,974		Johnson	5/5/98	
fab	9	5,774,859		Houser et al.	6/30/98	
fab	10	5,794,050		Dahlgren et al.	8/11/98	

FOREIGN PATENT DOCUMENTS

Examiner Initial	Cite No.	Office	NUMBER	Kind Code (if known)	Name of Patentee or Applicant of Cited Document	Date of Publication or Filing Date of Cited Document	Pages, Columns, Lines, Where Relevant Figures Appear	T
fab	11	WO	00/11869		Ellis et al.	3/2/00		
fab	12	EP	0 803 826 A2		Lindblad et al.	10/29/97		

OTHER PRIOR ART-NON PATENT LITERATURE DOCUMENTS

Examiner Initials	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume issue number(s), publisher, city and/or country where published.	T
fab	13	Dowding, John et al., "Gemini: A Natural Language System For Spoken-Language Understanding", SRI International	
fab	14	http://www.ai.sri.com/~oaa/infowiz.html, "InfoWiz: An Animated Voice Interactive Information System, May 8, 2000	
fab	15	Dowding, John, "Interleaving Syntax and Semantics in an Efficient Bottom-up Parser", SRI International	
fab	16	Moore, Robert et al., "Combining Linguistic and Statistical Knowledge Sources in a Natural-Language Processing for ATIS", SRI International	

EXAMINER	DATE CONSIDERED
Louis A. Bullock Jr	2/21/03

*EXAMINER: Initial if reference considered, whether or not criteria is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to application(s).

OIPE JC109 AUG 13 2002 PATENT & TRADEMARK OFFICE

RECEIVED AUG 15 2002 Technology Center 2100

INFORMATION DISCLOSURE STATEMENT BY APPLICANT Form PTO-1449 (Modified) (Use several sheets if necessary)	COMPLETE IF KNOWN	
	Application Number	09/225,198
	Confirmation Number	
	Filing Date	January 5, 1999
	First Named Inventor	Cheyer
	Group Art Unit	2755
	Examiner Name	Unassigned
Sheet 2 of 2	Attorney Docket No.	59501-8016.US01

U.S. PATENT DOCUMENTS

Examiner Initials	Cite No.	U.S. Patent or Application NUMBER	Kind Code (if known)	Name of Patentee or Inventor of Cited Document	Date of Publication or Filing Date of Cited Document	Pages, Columns, Lines, Where Relevant Figures Appear
fab	17	5,802,526		Fawcett et al.	9/1/98	
fab	18	6,192,338		Haszto et al.	2/2001	
fab	19	6,173,279		Levin et al.	1/2001	
fab	20	5,805,775		Eberman et al.	9/8/98	
fab	21	5,855,002		Armstrong	12/29/98	
fab	22	5,890,123		Brown et al.	3/30/99	
fab	23	5,963,940		Liddy et al.	10/5/99	
fab	24	6,003,072		Gerritsen et al	12/14/99	
fab	25	6,012,030		French-St. George et al.	1/4/00	
fab	26	6,026,388		Liddy et al.	2/15/00	
fab	27	6,080,202		Strickland et al.	6/27/00	
fab	28	6,021,427		Spagna et al.	1/1/00	
fab	29	6,338,081		Furusawa et al.		
fab	30	6,144,989		Hodjat et al.		
fab	31	6,226,666		Chang et al.		

OTHER PRIOR ART-NON PATENT LITERATURE DOCUMENTS

Examiner Initials	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume issue number(s), publisher, city and/or country where published.	T
fab	32	Stent, Amanda et al., "The CommandTalk Spoken Dialog System", SRI International	
fab	33	Moore, Robert et al., "CommandTalk: A Spoken-Language Interface for Battlefield Simulations:, October 23, 1997, SRI International	
fab	34	Dowding, John et al., "Interpreting Language in Context in CommandTalk", February 5, 1999, SRI International	

EXAMINER	DATE CONSIDERED
Lewis A. Bullock Jr	2/21/03

*EXAMINER: Initial if reference considered, whether or not criteria is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to application(s).

BY022180

2126

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF:
ADAM CHEYER ET AL.
APPLICATION NO.: 09/225,198
FILING DATE: JANUARY 5, 1999
FOR: **SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS**

ATTORNEY DOCKET NO.: 59501.8016.US01
EXAMINER: LEWIS ALEXANDER BULLOCK JR.
ART UNIT: 2126

#9

Change of Address

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

Effective immediately, please direct all further communications in the above-identified patent application to the following address:

Brian R. Coleman
Patent Attorney
Perkins Coie LLP
P. O. Box 2168
Menlo Park, CA 95026-2168

Respectfully submitted,
Perkins Coie LLP

Date: April 24 2003

Brian R. Coleman
Registration No. 39,145

Correspondence Address:
Customer No. 22918
Perkins Coie LLP
P. O. Box 2168
Menlo Park, California 94026-2168
(650) 838-4300

pall

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
P.O. Box 1450
Alexandria, Virginia 22313-1450
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

22918 7590 06/03/2003

PERKINS COIE LLP
P.O. BOX 2168
MENLO PARK, CA 94026

EXAMINER	
BULLOCK JR, LEWIS ALEXANDER	
ART UNIT	**PAPER NUMBER**
2126	10

DATE MAILED: 06/03/2003

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

PTO-90C (Rev. 07-01)

PRG

Interview Summary	Application N .	Applicant(s)	
	09/225,198	CHEYER ET AL.	
	Examiner	**Art Unit**	
	Lewis A. Bullock, Jr.	2126	

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

(1) *Lewis A. Bullock, Jr.*. (3)_____.

(2) *Corina Tan*. (4)_____.

Date of Interview: *2/29/03*.

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: *Claim 1*.

Identification of prior art discussed: *Martin*.

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 Continuation Sheet*.

(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.)

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.

Lewis A. Bullock Jr.

Examiner's signature, if required

Summary of Record of Interview Requirements

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. 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 t 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.

Continuation of Substance of Interview including description of the general nature of what was agreed to if an agreement was reached, or any other comments: Applicant proposed amending the claims such that the goal satisfaction plan entails the facilitating engine using "reasoning that includes one or more of domain-independent coordination strategies, domain-specific reasoning, and application-specific reasoning comprising rules and learning algorithms. Applicant argues this is quite different then the query execution plan as detailed in Martin. The examiner will consider the amendments in view of the prior art of record in responding in the subsequent action. The interview concluded.

JUN 0 6 2003

2151
CD-Rom

Attorney Docket No. 59501-8016.US01

CERTIFICATE OF MAILING (37 CFR 1.8(a))

I hereby certify that this paper (along with any referred to as being attached or enclosed) is being deposited with the U.S. Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Date: June 3, 2003

Sharyl Brown

Sharyl Brown

RECEIVED
JUN 1 6 2003
Technology Center 2100

Applicants: CHEYER et al.
Application No.: 09/225,198
Filed: January 5, 1999
Examiner: L. A. Bullock, Jr.
Group Art Unit 2151
For: **SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS**

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL FOR AMENDMENT AND RESPONSE AND

COMPUTER PROGRAM LISTING APPENDIX SUBMITTED ON COMPACT DISC

Sir:

1. Transmitted herewith are the following:

- ☒ Amendment and Response
- ☒ Copy 1 and Copy 2 of Compact Disc both containing the identical contents of Appendix A as filed with the patent application on January 5, 1999.
- ☒ Amended first page of Specification
- ☒ IDS, 1449 and 3 references

2. Machine format is ISO-9660 file system:

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

3. Fee Authorization

Applicants believe that there is no fee due, however, the Commissioner is authorized to charge any underpayment of fees to Deposit Account No. 50-2207. This paper is submitted in duplicate.

Respectfully submitted,
Perkins Coie LLP

Date: June 3, 2003

Carina M. Tan

Carina M. Tan
Registration No. 45,769

Correspondence Address:
Customer No. 22918
Perkins Coie LLP
P. O. Box 2168
Menlo Park, California 94026-2168
(650) 838-4300

Please forward to Group Art Unit 2/5?

Amended Compact Discs

EXAMINER NOTE: THIS PAPER IS AN INTERNAL WORKSHEET ONLY. DO NOT ENCLOSE WITH ANY COMMUNICATION TO THE APPLICANT. ITS PURPOSE IS ONLY THAT OF AN AID IN HIGHLIGHTING A PARTICULAR PROBLEM IN A COMPACT DISC.

THE ATTACHED CD (COPY 1) HAS BEEN REVIEWED BY OIPE FOR COMPLIANCE WITH 37 CFR 1.52(E). ***Please match this CD with the application listed below.***

Date:
Serial No./Control No. 09 225 198
Reviewed By: Q Williams Phone: 305 3027

- [x] The compact discs are readable and acceptable.
- [ ] Copy 1 and Copy 2 of the compact discs are not the same.
- [ ] The compact discs are unreadable.
- [ ] The files on the compact discs are not in ASCII.
- [ ] The compact discs contain at least one virus.
- [ ] Other

12 B/ RWB 9-28-03

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: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450,

on June 3, 2003 by Sharyl Brown

Sharyl Brown

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 March 3, 2003, the shortened statutory period for which runs until June 3, 2003.

IN THE SPECIFICATION

Done

Enclosed is substitute Page 1 of the specification which has been amended to identify the compact disk and lists the file names, size, and creation date of each file.

IN THE CLAIMS

Please amend Claims 1, 29, 61, 71 and 86. The claim amendments are submitted in "revised amendment format" as described in *AMENDMENTS IN A REVISED FORMAT NOW PERMITTED*, signed January 31, 2003, and published in *Official Gazette* on February 25, 2003.

B1

CLAIM AMENDMENTS

1. (Currently 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; 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 ~~that includes said one or more sub-goals; and~~ , 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. (Previously 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. (Previously Amended) A computer-implemented method as recited in claim 2 wherein the act of registering a specific agent further includes:

B1

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.

B1

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.

B1

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.

B1

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-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. (Currently Amended) 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 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 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.

B1

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 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.

B1

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.

B1

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 parallel disjunction operator that indicates that distinct goals are to be performed by different agents.

48. (Currently 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 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.

B1

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 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.

B1

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. (Currently Amended) 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 <u>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</u> ~~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.

31

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. (Currently Amended) 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 <u>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</u>.

B1

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.

B1

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. (Previously 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. (Previously 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. (Currently 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 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.

B1

87. (Previously 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. (Previously 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.

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.

REMARKS

The Examiner is thanked for the performance of a thorough search. By this amendment, Claims 1, 29, 61, 71 and 86 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. Applicants enclose a CD-ROM labeled as Copy 1 and an identical copy of the CD-ROM labeled as Copy 2 containing the identical contents of Appendix A as filed with the patent application on January 5, 1999. Also enclosed is substitute Page 1 of the specification which has been amended to identify the compact disc and list the file names, size, and creation date of each file.

SUMMARY OF REJECTIONS/OBJECTIONS

In the Office Action, Claims 1-89 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Developing Tools for the Open Agent Architecture" by Martin1 in view of "Information Brokering in an Agent Architecture" by Martin2.

REJECTIONS UNDER 35 U.S.C. § 103(a)

<u>CLAIMS 1, 29, 61, 71 and 86</u>

Claim 1 recites, in part, the features:

"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;**"

Claim 1 has been amended to clarify that the facilitating engine uses sophisticated reasoning when delegating sub-goal requests to best complete the requested service request. The facilitating engine's use of reasoning is supported by the specification on page 10, lines 15 – 18. Amended Claim 1 requires that the facilitating engine use "reasoning that includes one or more of domain-independent coordination strategies, domain-specific reasoning, and application-specific reasoning comprising rules and learning algorithms.

For purposes of explanation, assume that the facilitator receives a request such as, "Make Coffee". The facilitator's facilitating engine uses reasoning to generate the following goal satisfaction plan:

Sub-goal request A: Roast coffee beans
Sub-goal request B: Grind coffee beans
Sub-goal request C: Boil water, etc.

The facilitating engine is able to use reasoning to generate a plan to accomplish the base goal, "Make Coffee". The reasoning includes "one or more of domain-independent coordination strategies, domain-specific reasoning, and application-specific reasoning comprising rules and learning algorithms." For example, the facilitating engine uses its domain-specific reasoning based on domain-specific knowledge of symbols and axioms of the domain. In the above example, the facilitating engine uses its knowledge about domain symbols and axioms such as "coffee", "roast", and "beans" in order to generate a goal satisfaction plan by reasoning that making coffee entails roasting coffee beans, grinding coffee beans and boiling water, etc. Also, the coffee beans need to be roasted before the coffee beans can be ground and that only after the coffee beans are ground should water be boiled.

Further, the facilitating engine is able to use reasoning to delegate the sub-goals to service providing agents in such a way as "to best complete the requested service request." For example, assume that several agents are able to roast coffee. The facilitating engine is able to use reasoning to delegate the sub-goal task of roasting coffee to the service-providing agent that can roast beans in the least amount of time because the facilitating engine has reasoned that the least amount of time taken to make coffee is the best way to accomplish the base goal of making coffee.

Similarly, to use an example taken directly from the specification (see page 21, starting at line 29 to page 22, line 1-4), the facilitating engine accomplished the request "Remind Bob about lunch" by reasoning that all available message transfer agents (e.g., fax, phone, mail, pager) are to be enabled to **compete** for the opportunity to carry out the request. In other words, the base goal is carried out not by merely parsing the request into sub-goals **based on the syntax** of the request. Rather, the facilitating engine used reasoning to decide upon using **competing** message transfer agents to reminding Bob of lunch, in lieu of delegating the task to just one message transfer agent.

In contrast, *Martin's* "Development Tools for the Open Agent Architecture" (*Martin1*) and *Martin's* "Information Brokering in An Agent Architecture" fail to teach the goal satisfaction plan that entails the type of reasoning described above as performed by the facilitator agent. As mentioned by the Examiner in the Office Action, *Martin's* "Development Tools for the Open Agent Architecture" does not teach the act of constructing a goal satisfaction plan.

As for *Martin's* "Information Brokering in An Agent Architecture" (*Martin2*), it merely discloses query processing and a query execution plan which is NOT the same as a goal execution plan. Thus, *Martin2* is merely describing a method for information retrieval rather than fulfillment of a service request. Moreover, query execution plans are well-known in database systems. In database systems, query statements are made in query languages such as SQL. SQL statements are fulfilled according to a query execution plan based on the manner in which information is stored in the database. In contrast, the goal satisfaction plan is a plan that entails reasoning in its construction, rather than being based on the manner in which information is stored in a database.

Further, *Martin2* merely teaches that the queries are systematically broken based on syntax of the queries without any kind of reasoning for forming a goal satisfaction plan such as that of the "Make Coffee" example above. In *Martin2*, on page 11, *Martin2* teaches the construction of a query execution plan by analysis of "each predicate in the query" and the rewriting of the query for dispatch to information sources based on "a disjunction of translated subqueries. Therefore in *Martin2*, each request made of information sources **must have appeared syntactically** (albeit with language translation) **in the original query**.

Neither *Martin1* nor *Martin2*, either alone or in combination, disclose, teach, suggest or make obvious the novel features of claim 1. Thus, Claim 1 is allowable.

Claims 29, 61, 71 and 86, each contain similar features regarding the use "reasoning that includes one or more of domain-independent coordination strategies, domain-specific reasoning, and application-specific reasoning comprising rules and learning algorithms. Thus, Claims 26, 61, 71 and 86 are allowable for at least the reasons provided herein in respect to Claim 1.

<u>CLAIMS 2-28, 30-47, 62-70, 72-85 and 87-89</u>

Claims 2-28 are either directly or indirectly dependent upon Claim 1 and include all the limitations of Claim 1 and therefore are allowable for at least the reasons provided herein in respect to Claim 1.

Claims 30-47 are either directly or indirectly dependent upon Claim 29 and include all the limitations of Claim 29 and therefore are allowable for at least the reasons provided herein in respect to Claim 29.

Claims 62-70 are either directly or indirectly dependent upon Claim 61 and include all the limitations of Claim 61 and therefore are allowable for at least the reasons provided herein in respect to Claim 61.

Claims 72-85 are either directly or indirectly dependent upon Claim 71 and include all the limitations of Claim 71 and therefore are allowable for at least the reasons provided herein in respect to Claim 71

Claims 87-89 are either directly or indirectly dependent upon Claim 86 and include all the limitations of Claim 86 and therefore are allowable for at least the reasons provided herein in respect to Claim 86.

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 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 Claim 48 as amended requires that "**goals within a single request**" are "**coupled by one or more operators from a set of operators**". In amended Claim 48, the set of operators comprise, **a conditional execution operator**, and **a parallel disjunctive operator**.

In the Office Action, the Examiner states that "the ICL having expression which may be coupled by a conjunctive operator". The claim has therefore been amended to clarify the applicant's invention. It is to be noted that *Martin2* does not suggest or mention **conditional execution operator**, and **a parallel disjunctive operators**.

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 conditional execution operator** (such as "if" and "when", allowing for particular actions to be predicated on the state, or outcomes of earlier actions), and **a parallel disjunctive operator** (allowing for alternative actions to be performed at the same time, if resources allow, and a first-to-respond strategy may be used in their competition to perform the goal at hand). 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

Date: June 3, 2003

Carina M. Tan
Registration No. 45,769

Correspondence Address:

Customer No. 22918
Perkins Coie LLP
P. O. Box 2168
Menlo Park, California 94026
(650) 838-4300

Marked-up version

Software-Based Architecture for Communication and Cooperation Among Distributed Electronic Agents

By:

Adam J. Cheyer and David L. Martin

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

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*

JUN 0 6 2003 P.E JC48 PATENT & TRADEMARK OFFICE

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: Commissioner for Patents, P.O. Box 1450, Alexandria, VA, 22313-1450, on:

Date: June 3, 2003 By: Sharyl Brown
Sharyl Brown

#11
D. Bond
6/19/03

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE 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

EXAMINER: L. A. BULLOCK, JR.

ART UNIT: 2151

RECEIVED
JUN 0 9 2003
Technology Center 2100

Supplemental Information Disclosure Statement After First Office Action but Before Final Action or Notice of Allowance – 37 CFR 1.97(c)

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

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
☐ References marked by asterisks
☐ The following:

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 submits that no fee is due in light of the following certification under 37 CFR 1.97(e) (check only one):

☐ In accordance with 37 CFR 1.97(e)(1), the undersigned hereby states that each item of information submitted herewith was cited in a communication from a foreign patent office in a counterpart foreign application not more than three months prior to this filing of this statement; or

☒ In accordance with 37 CFR 1.97(e)(2), the undersigned hereby states that no item of information submitted herewith was cited in a communication from a foreign patent office in a counterpart foreign application, or, to the knowledge of the person signing the certification after making reasonable inquiry, was known to any individual designated in 37 CFR 1.56(c), more than three months prior to the filing of this statement.

☒ Please charge any underpayment for timely filing of this paper to Deposit Account No. 50-2207.

5. Patent Term Adjustment (37 CFR 1.704(d))

☐ The undersigned states that each item of information submitted herewith was cited in a communication from a foreign patent office in a counterpart application and that this communication was not received by any individual designated in 37 C.F.R. §1.56(c) more than thirty days prior to the filing of this statement. 37 C.F.R. §1.704(d).

Respectfully submitted,
Perkins Coie LLP

Date: 6/3/03

Carina M. Tan
Registration No. 45,769

Correspondence Address:
Customer No. 22918
Perkins Coie LLP
P.O. Box 2168
Menlo Park, California 94026
(650) 838-4300

Software-Based Architecture for Communication and Cooperation Among Distributed Electronic Agents

RECEIVED
JUN 1 6 2003
Technology Center 2100

By:
Adam J. Cheyer and David L. Martin

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

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*

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
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

22918 7590 11/28/2003

PERKINS COIE LLP
P.O. BOX 2168
MENLO PARK, CA 94026

EXAMINER	
BULLOCK JR, LEWIS ALEXANDER	
ART UNIT	**PAPER NUMBER**
2126	17

DATE MAILED: 11/28/2003

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

PTO-90C (Rev. 10/03)

Office Action Summary	Application N .	Applicant(s)
	09/225,198	CHEYER ET AL.
	Examin r	Art Unit
	Lewis A. Bullock, Jr.	2126

-- The MAILING DATE of this communication appears on the cover sheet with th 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 03 June 2003.

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) 1-86 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) 1-86 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).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. §§ 119 and 120

12)☐ 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.

13)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application) since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

a) ☐ The translation of the foreign language provisional application has been received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121 since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

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) 11.
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, 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. Applicant must include the Appendix A.V, source code file named translations.pl. with the other appendices on a compact disc.

* Applicant is also requested to delete the Brief Description of the Appendices on page 8, line 23 – page 9, line 3, since the amendment to page 1 is made.

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 1-3, 5-11, 15-25, 29-34, 38-44, 61-71, and 86-89 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 KISS (US 6,484,155).

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.

KISS teaches an agent architecture for communicating and cooperation among distributed electronic agents (user agents / meta agents / and knowledge agents), wherein a facilitator agent (meta agent) is operable for generating / constructing a goal satisfaction plan (dynamic "solution plan") associated with the base goal (query) wherein the goal satisfaction plan includes a suitable delegation of sub-goal requests (sub-plans / tasks) to best complete the requested service request-by using domain-independent or domain –specific reasoning (col. 5, lines 14-45; col. 8, lines 21 – col. 9, line 26; col. 10, lines 10-38; col. 2, lines 50-67). Therefore, it would be obvious to combine the teachings of MARTIN1 with the teachings of KISS in order that inference be distributed and cooperative over a distributed environment (col. 3, lines 47 – col. 4, line 17).

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."). It would be inherent 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 determined for the base goal. 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.

KISS teaches an agent architecture for communicating and cooperation among distributed electronic agents (user agents / meta agents / and knowledge agents), wherein a facilitator agent (meta agent) is operable for generating / constructing a goal satisfaction plan (dynamic "solution plan") associated with the base goal (query) wherein the goal satisfaction plan includes a suitable delegation of sub-goal requests (sub-plans / tasks) to best complete the requested service request-by using domain-independent or domain –specific reasoning (col. 5, lines 14-45; col. 8, lines 21 – col. 9, line 26; col. 10, lines 10-38; col. 2, lines 50-67). Therefore, it would be obvious to combine the teachings of MARTIN1 with the teachings of KISS in order that inference be distributed and cooperative over a distributed environment (col. 3, lines 47 – col. 4, line 17).

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) according to an Interagent Communication language, 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 constructs a goal satisfaction plan.

KISS teaches an agent architecture for communicating and cooperation among distributed electronic agents (user agents / meta agents / and knowledge agents), wherein a facilitator agent (meta agent) has a facilitating engine operable to parse a service request (query) in order to interpret a compound goal (goal statement), wherein the compound goal includes local and global constraints and parameters (col. 5, lines 33 – 64; col. 8, line 32 – col. 9, line 37) and the engine further operable for generating / constructing a goal satisfaction plan (dynamic "solution plan") associated with the base goal (query) wherein the goal satisfaction plan includes a suitable delegation of sub-goal requests (sub-plans / tasks) to best complete the requested service request-by using domain-independent or domain –specific reasoning (col. 5, lines 14-45; col. 8, lines 21 – col. 9, line 26; col. 10, lines 10-38; col. 2, lines 50-67). Therefore, it would be obvious to combine the teachings of MARTIN1 with the teachings of KISS in order that inference be distributed and cooperative over a distributed environment (col. 3, lines 47 – col. 4, line 17).

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.

KISS teaches an agent architecture for communicating and cooperation among distributed electronic agents (user agents / meta agents / and knowledge agents), wherein a facilitator agent (meta agent) is operable for generating / constructing a goal satisfaction plan (dynamic "solution plan") associated with the base goal (query) wherein the goal satisfaction plan includes a suitable delegation of sub-goal requests (sub-plans / tasks) to best complete the requested service request-by using domain-independent or domain –specific reasoning (col. 5, lines 14-45; col. 8, lines 21 – col. 9, line 26; col. 10, lines 10-38; col. 2, lines 50-67). Therefore, it would be obvious to combine the teachings of MARTIN1 with the teachings of KISS in order that inference be distributed and cooperative over a distributed environment (col. 3, lines 47 – col. 4, line 17).

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 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 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 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 38-44, refer to claims 15-25 for rejection.

As to claim 62, KISS teaches 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 (col. 5, line 20-64).

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, MARTIN1 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 87, MARTIN1 teaches a 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). 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 KISS and thereby obvious that the method can be entailed in a data wave carrier.

As to claim 88, MARTIN1 teaches a 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). 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 KISS and thereby obvious that the method can be entailed in a data wave carrier.

As to claim 89, KISS teaches a response to the dispatched goal including results from the agent for performance to the facilitator agent (col. 5, line 65 – col. 6, line 28). 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 KISS and thereby obvious that the method can be entailed in a data wave carrier.

4. Claims 4, 12-14, 26-28, 35-37, 45-47, and 72-85 are rejected under 35 U.S.C. 103(a) as being unpatentable over MARTIN1 in view of KISS as applied to claim 1 above, and further in view of "Information Brokering in an Agent Architecture" by MARTIN2.

As to claim 4, MARTIN1 and KISS substantially disclose the invention. However, neither reference teaches the cited deactivating. 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 KISS and MARTIN2 in order to facilitate the transparent delegation, translation, and relaying of the appropriate subqueries to the available source agents (pg. 7-8; pg. 1).

As to claims 12-14, MARTIN1 and KISS substantially disclose the invention. However, neither reference teaches the cited receiving. 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 teaches the base goal or request is expressed in the Interagent Communication Language and is broken down such that subrequests are distributed to the appropriate agents (pg. 5). However, combination does not teach that operators including a conjunction operator or a parallel disjunction operator separate the base goal.

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 and disjunction operator (pg. 10, "Although the body of the broker predicate rule is characterized as a conjunction of predicates….Disjunction, negation…"). 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 is a compound goal and 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 claims 45-47, refer to claims 26-28 for rejection.

As to claim 72, 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, exchange information, and set triggers with other agents (pg. 5, Agents share a common communication language...and may run on any network linked platform."; pg. 5, "The Open Agent Architecture"). It is inherent that since triggers are used in order for a message to be sent to an agent, that the trigger is a conditional execution operator. However, neither MARTIN1 nor KISS teach the ICL supporting compound goal expressions from a disjunction operation.

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 parallel disjunctive operation or conditional execution operation or conjunctive operator (pg. 10, "Disjunction, negation (that is, Prolog-style negation as failure), and a few other control operators are also allowed."). 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 73 and 74, 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 75-78, MARTIN1 teaches the ICL supports task completion constraints (triggers) within goal expressions (pg. 5).

As to claims 79-83, 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 to be stored by the facilitator agent in a registry (pg. 5, "Every agent participating in an OAA-based system defines and publishes...we refer to these capabilities specifications as solvables.").

As to claims 84 and 85, MARTIN1 and KISS substantially disclose the invention. However, neither reference teaches the cited distribution. MARTIN2 teaches that facilitator engines (broker agents) are distributed across at least two computer processes (multiple broker agents in an architecture) (pg 7, pg. 16) wherein each stores a planning component (schema mapping rules) (pg. 8). It would be obvious that since the broker performs the delegation that it also has an execution component and therefore each broker agent has an execution component. Refer to claim 4 for the motivation to combine.

5. Claims 48-60 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 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, exchange information, and set triggers with other agents (pg. 5, Agents share a common communication language…and may run on any network linked platform."; pg. 5, "The Open Agent Architecture"). It is inherent that since triggers are used in order for a message to be sent to an agent, that the trigger is a conditional execution operator. However, MARTIN1 does not teach the ICL supporting compound goal expressions from a disjunction operation.

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 parallel disjunctive operation or conditional execution operation (pg. 10, "Disjunction, negation (that is, Prolog-style negation as failure), and a few other control operators are also allowed."). 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 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 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 to be stored by the facilitator agent in a registry (pg. 5, "Every agent participating in an OAA-based system defines and publishes...we refer to these capabilities specifications as solvables.").

Response to Arguments

6. Applicant's arguments with respect to claims 1-86 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

7. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). 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 date of this final action.

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, John A Follansbee can be reached on (703) 305-8498. The fax phone number for the organization where this application or proceeding is assigned is (703) 746-7239.

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.

lab

JOHN FOLLANSBEE
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

Notice of Referenc s Cited	Application/Control No. 09/225,198	Applicant(s)/Patent Under Reexamination CHEYER ET AL.	
	Examiner Lewis A. Bullock, Jr.	Art Unit 2126	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-6,484,155	11-2002	Kiss et al.	706/46
	B	US-6,212,649	04-2001	Yalowitz et al.	714/31
	C	US-			
	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	Moran et al. "Multimodal User Interfaces in the Open Agent Architecture." Proceedings of the International Conference on Intelligent User Interfaces. 6-9/1997.
	V	Martin, David et al. "The Open Agent Architecture: A Framework for Buidling Distributed Software Systems." October 19, 1998.
	W	Wilkins, David et al. "Multiagent Planning Architecture." SRI International. December 8, 1997.
	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.

OIPE JC48
JUN 0 6 2003
PATENT & TRADEMARK OFFICE

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

Form PTO-1449 (Modified)
(Use several sheets if necessary)

Sheet 1 of 1

MPLETE IF KNOWN	
Application Number	09/225,198
Filed	January 5, 1999
First Named Inventor	Cheyer
Group Art Unit	2151
Examiner Name	L. A. Bullock, Jr.
Atty Dkt No.	59501-8016.US01

U.S. PATENT DOCUMENTS

Examiner Initials	Cite No.	U.S. Patent or Application NUMBER	Kind Code (if known)	Name of Patentee or Inventor of Cited Document	Date of Publication or Filing Date of Cited Document	Pages, Columns, Lines, Where Relevant Figures Appear
LAB		6,047,053		Miner et al.	04/04/00	
LAB		6,256,771	B1	O'Neil et al.	07/03/01	
LAB		6,411,684	B1	Cohn et al.	06/25/02	
						RECEIVED
						JUN 0 9 2003
						Technology Center 2100

FOREIGN PATENT DOCUMENTS

Examiner Initial	Cite No.	Foreign Patent or Application Office	NUMBER	Kind Code (if known)	Name of Patentee or Applicant of Cited Document	Date of Publication or Filing Date of Cited Document	Pages, Columns, Lines, Where Relevant Figures Appear	T

OTHER PRIOR ART-NON PATENT LITERATURE DOCUMENTS

Examiner Initials	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume issue number(s), publisher, city and/or country where published.	T

EXAMINER	DATE CONSIDERED
Lee A. Bullock Jr	11/20/03

*EXAMINER: Initial if reference considered, whether or not criteria is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to application(s).

Sㇴ

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
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

22918 7590 03/17/2004

PERKINS COIE LLP
P.O. BOX 2168
MENLO PARK, CA 94026

EXAMINER	
BULLOCK JR, LEWIS ALEXANDER	
ART UNIT	**PAPER NUMBER**
2126	14

DATE MAILED: 03/17/2004

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

PTO-90C (Rev. 10/03)

PR4

Interview Summary	Application No.	Applicant(s)	
	09/225,198	CHEYER ET AL.	
	Examin r	**Art Unit**	
	Lewis A. Bullock, Jr.	2126	

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

(1) *Lewis A. Bullock, Jr.*.

(2) *Corina Tan*.

(3) *David Stringer-Calbert*.

(4)_____.

Date of Interview: *11 March 2004*.

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: *1-89*.

Identification of prior art discussed: *Kiss and Inventors publications*.

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 Continuation Sheet*.

(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.)

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, OR THE MAILING DATE OF THIS INTERVIEW SUMMARY FORM, WHICHEVER IS LATER, 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.

L A Bullock Jr
Examiner's signature, if required

Summary of Record of Interview Requirements

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. 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.

Continuation of Substance of Interview including description of the general nature of what was agreed to if an agreement was reached, or any other comments: Applicants argued that the prior art teachings of Kiss did not accomplish the inventors goal of the faciliatator agent using the goal satisfaction plan that stored the intelligence of the order of the sub-goals since Kiss teaches that the solution plan can be dynamically modifed. The examiner alluded that the claims make no mention that the solution plan cannot be modified and that Kiss's solution plan accomplishes the limitations of the claims as disclosed. The examiner pointed out that all the rejections regarding this application were made with publications written by the Applicants. The examiner pointed out that there are limitations in the specification regarding the Interagent Communication Language that were not disclosed in any of the inventors publications that can distinguish the claims from the prior art of record. In particular, the examiner pointed to page 17, lines 7-11 which describe the ICL as including a layer of conversational protocol and a content layer that distinguish the claims from any teaching disclosed in the publications. The examiner also pointed out that this teaching distinguishes the Applicant's interagent communication language from the well known communication language KQML. Applicants will submit a response amending the claims to the examiners suggestions. The interview concluded..

EXPRESS MAIL LABEL NO. EV 099152888 US

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.: 2126

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

Mail Stop AF
Commissioner of Patents
P. O. Box 1450
Alexandria, VA 22313-1450

AMENDMENT AND RESPONSE

Sir:

This is in response to the Final Office Action mailed November 28, 2003, the shortened statutory period for which runs until February 28, 2004.

IN THE CLAIMS

1. (Currently 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, 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; and

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. (Previously presented) 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. (Previously presented) 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. (original) 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. original) A computer-implemented method as recited in claim 1 further comprising the act of providing an agent registry data structure.

6. (original) 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. (original) 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. (original) 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. (original) 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. (original) 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. (original) A computer-implemented method as recited in claim 1 further comprising the act of establishing communication between the plurality of distributed agents.

12. (original) 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. (original) 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. (original) A computer-implemented method as recited in claim 13 wherein the natural language query was generated by a user interface agent.

15. (original) A computer-implemented method as recited in claim 1, wherein the base goal requires setting a trigger having conditional functionality and consequential functionality.

16. (original) 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. (original) 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. (original) 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. (original) 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. (original) A computer-implemented method as recited in claim 15 wherein the trigger is installed and executed within the facilitator agent.

21. (original) A computer-implemented method as recited in claim 15 wherein the trigger is installed and executed within a first service-providing agent.

22. (original) A computer-implemented method as recited in claim 15 wherein the conditional functionality of the trigger is installed on a facilitator agent.

23. (original) 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. (original) 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. (original) A computer-implemented method as recited in claim 15 wherein the consequential functionality of the trigger is installed on a facilitator agent.

26. (original) A computer-implemented method as recited in claim 1 wherein the base goal is a compound goal having sub-goals separated by operators.

27. (original) 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. (original) 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. (Currently amended) 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), wherein the ICL includes:

a layer of conversational protocol defined by event types and parameter lists associated with one or more of the 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 requested 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 to best complete the requested service; and

implementing the base goal satisfaction plan.

30. (original) 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. (original) 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. (original) 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. (original) 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. (original) 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. (original) 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. (original) 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. (original) A computer program as recited in claim 36 wherein the natural language query is generated by a user interface agent.

38. (original) 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. (original) 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. (original) 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. (original) 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. (original) 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. (original) A computer program as recited in claim 38 further including computer executable instructions for installing and executing the trigger within the facilitator agent.

44. (original) 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. (original) A computer program as recited in claim 29 further including computer executable instructions for interpreting compound goals having sub-goals separated by operators.

46. (original) 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. (original) 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. (Currently 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:

<u>the ICL having one or more of:</u>

<u>a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events; and</u>

<u>a content layer comprising one or more of goals, triggers and data elements associated with the events;</u>

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 conditional execution operator; and

a parallel disjunctive operation that indicates that disjunct goals are to be performed by different agents.

49. (original) An ICL as recited in claim 48, wherein the ICL is computer platform independent.

50. (original) 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. (original) 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. (original) 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. (original) An ICL as recited in claim 51 wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

54. (original) An ICL as recited in claim 48 wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

55. (original) 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. (original) An ICL as recited in claim 55 wherein an electronic agent's solvables define an interface for the electronic agent.

57. (original) An ICL as recited in claim 56 wherein the facilitator agent maintains an agent registry making available a plurality of electronic agent interfaces.

58. (original) 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. (original) 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. (original) 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. (Currently amended) 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), wherein the ICL includes:

a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events; and

a content layer comprising one or more of goals, triggers and data elements associated with the events;

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. (original) 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. (original) 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. (original) 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. (original) 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. (original) 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. (original) A facilitator agent as recited in claim 66 wherein the data repository is local to the facilitator agent.

68. (original) A facilitator agent as recited in claim 66 wherein the data repository is remote from the facilitator agent.

69. (original) A facilitator agent as recited in claim 64 wherein the trigger mechanism is a task trigger having a set of conditions.

70. (original) 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. (Currently amended) 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~~

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; 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. (Currently amended) A computer architecture as recited in claim 71, wherein the ~~basis for the computer architect is an~~ 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 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. (original) A computer architecture as recited in claim 72, wherein the ICL is computer platform independent.

74. (original) 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. (original) A computer architecture as recited in claim 73 wherein the ICL syntax supports explicit task completion constraints within goal expressions.

76. (original) 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. (original) A computer architecture as recited in claim 75 wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

78. (original) A computer architecture as recited in claim 73 wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

79. (original) 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. (original) A computer architecture as recited in claim 79 wherein an electronic agent's solvables define an interface for the electronic agent.

81. (original) 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. (original) 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. (original) 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. (Previously presented) 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. (Previously presented) 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. (Currently 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, 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; 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. (Previously presented) 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. (Previously presented) 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. (original) 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.

REMARKS

INTERVIEW:

A telephonic interview was conducted on March 11, 2004. The participants were Examiner Lewis A. Bullock, Jr., David Stringer-Calvert and Carina M. Tan. During the interview, an agreement with respect to all the claims were reached. Applicants argued that the prior art teachings of *KISS* did not disclose any intelligent reasoning when formulating a goal satisfaction plan. Applicants argued that *KISS* merely discloses a method of information retrieval from information repositories such as databases. The examiner disagreed. However, the examiner pointed out that certain features in Applicant's specification regarding ICL are novel. The Examiner indicated that the ICL features: 1) a conversational protocol layer, and 2) a content layer, would distinguish applicants' claims over the prior art. It was agreed that applicants would submit a response amending the claims to include the above novel ICL features.

The Examiner is thanked for the performance of a thorough search. By this response, claims 1, 29, 48, 61, 71, 72 and 86 have been amended. No claims have been cancelled or added. Hence, Claims 1-89 are pending in the Application.

IN THE SPECIFICATION

Compact Disc Containing Appendices

Applicants cancel the computer program listing appearing in the specification in Appendices A, B, C, D, and E. In compliance with 37 CFR 1.96(c), Applicants enclose a CD-ROM labeled as Copy 1 and an identical copy of the CD-ROM labeled as Copy 2 containing the identical contents of Appendices A, B, C, D and E as filed with the patent application on January 5, 1999.

Substitute Pages Of Specification

Enclosed are substitute Pages 1, 8 and 9. Substitute Page 1 of the specification has been amended to identify the compact disc and list the file names, size, and creation date of each file, and substitute Page 8 and Page 9 which have been amended to delete the "Brief Description of the Appendices." Also enclosed is a substitute ABSTRACT containing less than 150 words. The ABSTRACT as originally filed contained more than 150 words.

SUMMARY OF REJECTIONS/OBJECTIONS

In the Office Action, Claims 1-3, 5-11, 15-25, 29-34, 38-44, and 61-71 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Developing Tools for the Open Agent Architecture" by Martin1 in view of U.S. Patent No. 6,484,155 issued to Kiss.

Claims 4, 12-14, 26-28, 35-37, 45-47, and 72-85 are rejected under 35 U.S.C. 103(a) as being unpatentable over Martin1 in view of Kiss, and further in vie of "Information Brokering in an Agent Architecture" by Martin2.

Claims 48-60 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.

REJECTIONS UNDER 35 U.S.C. § 103(a)

CLAIMS 1, 29, 61, 71 and 86

Claim 1, as amended, recites in part, the features:

"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; and

a content layer comprising one or more of goals, triggers and data elements associated with the events;

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;"

Claim 1 includes the limitation of a inter-agent language, wherein the inter-agent language includes 1) a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, and 2) a content layer comprising one or more of goals, triggers and data elements associated with the events. The cited references do not disclose or suggest such a conversational protocol and content layer.

Further, the Office Action states that the "dynamic solution plan" in *KISS* is the equivalent of the "goal satisfaction plan" of applicants' Claim 1 above. The Office Action points to col. 5, lines 14-45; col. 8, line 21 - col. 9, line 26; and col. 10, lines 10-38, and col. 2, lines 50-67 for support.

The method for forming the "dynamic solution plan" in *KISS* is irrelevant to the method of forming the goal satisfaction plan in Applicants' Claim 1. It is respectfully submitted that *KISS* is irrelevant because *KISS* is an invention involving accessing knowledge repositories. Such knowledge repositories are represented by "knowledge agents." The Abstract of *KISS* states that "the invention solicits accessible knowledge repositories, represented by knowledge agents, for relevant knowledge..."

In other words, *KISS* is merely a method of information retrieval from information repositories or data sources. For example, the meta agent can ask questions involving facts or data and the agents attempt to retrieve the facts or data from the corresponding data repository. In contrast, the goal satisfaction plan of Claim 1 involves asking service providing agents to perform **actions** such as boil water, roast coffee beans, grind the roasted coffee beans as opposed to merely asking the agents to retrieve information from an information repository.

To further explain why *KISS* is irrelevant and completely different from the method of Claim 1, see col. 5 lines 39-43 where "[t]he meta agent 119 is configured to begin executing the solution plan even before the plan is complete." This underscores the fact that the solution plan in *KISS* merely involves information retrieval rather than asking the agent to perform intelligent actions such as roast coffee beans. In *KISS*, it is not fatal to begin executing the solution plan even before the plan is complete because no real harm is done if the meta agent begins by asking the wrong questions. To explain, *KISS* teaches "the meta agent 119 is capable of backtracking or replanning to permit escape from a dead-end." In other words, it is not fatal if the search for data is proceeding down an incorrect search path, as explained in *KISS*. In contrast, the facilitator of Claim 1 cannot begin execution of the goal satisfaction plan before the goal satisfaction plan is complete. For example, it would be fatal for the facilitator to ask a service-providing agent to boil the coffee beans instead of requesting that the coffee beans be first roasted and then ground. Such an action of boiling the coffee beans would be **irreversible** and would produce soggy beans. In other words, the service-providing agents of Claim 1 perform actions and are not merely sources of information.

Further, *KISS* does not use reasoning for "formulating the dynamic solution plan." In other words, *KISS* does not use the inferencing schemes as described in column 7 for generating the solution plan. In fact, *KISS* teaches away from using reasoning or inferencing for generating the solution plan. Column 8, lines 58-61 of *KISS* states that "**[a]fter** the solution plan is formulated, the meta agent 119 implements a distributed inference process to perform the search and execution phases of solving the problem, while maintaining control of the process" (emphasis added). Thus, the inference process is what the solution plan in *KISS* accomplishes and is not what is used to generate the solution plan.

In contrast, Claim 1 shows that the facilitating engine uses sophisticated reasoning when delegating sub-goal requests to best complete the requested service request. The facilitating engine's use of reasoning is supported by the specification on page 13, lines 342-347.

Assume that the facilitator agent of Claim 1 receives a request such as, "Make Coffee". The facilitator agent's facilitating engine uses reasoning to generate the following goal satisfaction plan:

Sub-goal request A: Please perform the act of roasting coffee beans
Sub-goal request B: Please perform the act of grinding coffee beans
Sub-goal request C: Please perform the act of boiling water, etc.

The facilitating engine is able to use reasoning to accomplish the base goal, "Make Coffee" by asking an appropriate agents to first roast the coffee beans before asking the agent to grind the beans, etc.

Neither *Cohen* nor *KISS*, either alone or in combination, disclose, teach, suggest or make obvious the novel features of claim 1. Thus, Claim 1 is allowable.

Claims 29, 61, 71 and 86, each contain similar features regarding "using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal." Thus, Claims 29, 61, 71 and 86 are allowable for at least the reasons provided herein in respect to Claim 1.

CLAIMS 2-28, 30-47, 62-70, 72-85 and 87-89

Claims 2-28 are either directly or indirectly dependent upon Claim 1 and include all the limitations of Claim 1 and therefore are allowable for at least the reasons provided herein in respect to Claim 1.

Claims 30-47 are either directly or indirectly dependent upon Claim 29 and include all the limitations of Claim 29 and therefore are allowable for at least the reasons provided herein in respect to Claim 29.

Claims 62-70 are either directly or indirectly dependent upon Claim 61 and include all the limitations of Claim 61 and therefore are allowable for at least the reasons provided herein in respect to Claim 61.

Claims 72-85 are either directly or indirectly dependent upon Claim 71 and include all the limitations of Claim 71 and therefore are allowable for at least the reasons provided herein in respect to Claim 71

Claims 87-89 are either directly or indirectly dependent upon Claim 86 and include all the limitations of Claim 86 and therefore are allowable for at least the reasons provided herein in respect to Claim 86.

CLAIM 48

Claim 48 as amended, recites in part:

"the ICL having one or more of:

a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events; and
a content layer comprising one or more of goals, triggers and data elements associated with the events;

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 operator that indicates that disjunct goals are to be performed by different agents."

The novel method recited in Claim 48 as amended requires that the inter-agent language include 1) a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, and 2) a content layer comprising one or more of goals, triggers and data elements associated with the events. The cited references do not disclose or suggest such a conversational protocol and content layer.

Further, the novel method recited in Claim 48 as amended requires that "goals within a single request" are "coupled by one or more operators from a set of operators". In amended Claim 48, the set of operators comprise, a conditional execution operator, and **a parallel disjunctive operator**.

In the Office Action, the Examiner states that triggers are conditional operators. It is respectfully submitted that triggers are not conditional operators in the sense of an being a syntactical operator in an expression.

Further, the Office Action states that page 10 of *Martin2* discloses **parallel disjunctive operators**. *Martin2* does NOT disclose parallel disjunctive operators. The "disjunction" in *Martin2* is the run-of-the-mill Prolog style disjunction. The expression, "Do task A OR Do Task B," is an example of a *Martin2* type disjunction. In contrast, a "**parallel disjunctive operator** is an operator that indicates that disjunct goals are to be performed by different agents. An example of a **parallel disjunctive operator** expression is "Ask agent Bob to do task A OR Ask agent Fred to do task B concurrently.

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 conditional execution operator** (such as "if" and "when", allowing for particular actions to be predicated on the state, or outcomes of earlier actions), and **a parallel disjunctive operator** (allowing for alternative actions to be performed at the same time, if resources allow, and a first-to-respond strategy may be used in their competition to perform the goal at hand). 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

Date: March 29, 2004

Carina M. Tan
Carina M. Tan
Registration No. 45,769

Correspondence Address:

Customer No. 22918
Perkins Coie LLP
P. O. Box 2168
Menlo Park, California 94026
(650) 838-4300

Software-Based Architecture for Communication and Cooperation Among Distributed Electronic Agents

By:

Adam J. Cheyer and David L. Martin

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

RECEIVED
JUN 0 8 2004
Technology Center 2100

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*

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.

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

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.

69

Please forward to Group Art Unit 2126

Amended Compact Discs

EXAMINER NOTE: THIS PAPER IS AN INTERNAL WORKSHEET ONLY. DO NOT ENCLOSE WITH ANY COMMUNICATION TO THE APPLICANT. ITS PURPOSE IS ONLY THAT OF AN AID IN HIGHLIGHTING A PARTICULAR PROBLEM IN A COMPACT DISC.

THE ATTACHED CD (COPY 1) HAS BEEN REVIEWED BY OIPE FOR COMPLIANCE WITH 37 CFR 1.52(E). ***Please match this CD with the application listed below.***

Date: 6/3/04
Serial No./Control No. 09/225198
Reviewed By: K. SMITH Phone: 308/215

- [x] The compact discs are readable and acceptable.
- [ ] Copy 1 and Copy 2 of the compact discs are not the same.
- [ ] The compact discs are unreadable.
- [ ] The files on the compact discs are not in ASCII.
- [ ] The compact discs contain at least one virus.
- [ ] Other

03-30-04 AF/2700

Attorney Docket No. 59501-8016.US01

EXPRESS MAIL LABEL NO. EV 099152888 US

OIPE JC102 MAR 2 9 2004 PATENT & TRADEMARK OFFICE

Applicants: CHEYER et al.
Application No.: 09/225,198
Filed: January 5, 1999
Examiner: L. A. Bullock, Jr.
Group Art Unit 2151
For: **SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS**

Mail Stop AF
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

RECEIVED
JUN 0 8 2004
Technology Center 2100

TRANSMITTAL FOR AMENDMENT AND RESPONSE AND

COMPUTER PROGRAM LISTING APPENDIX SUBMITTED ON COMPACT DISC

Sir:

This is in response to the Final Office Action mail by the U.S. Patent and Trademark Office on November 28, 2003. Applicants request a one month extension of time, thus allowing Applicants until March 28, 2004 to respond.

1. Transmitted herewith are the following:

☒ Check No. 2195 in the amount of $55.00
☒ Amendment and Response
☒ Copy 1 and Copy 2 of Compact Disc both containing the identical contents of Appendices A, B, C, D, and E as filed with the patent application on January 5, 1999.

2. Machine format is ISO-9660 file system:

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

03/31/2004 SSESHE1 00000104 09225198
01 FC:2251 55.00 OP

3. Fee Authorization

Check No. 2195 in the amount of $55.00 is enclosed for the required fees for one month extension of time, however, the Commissioner is authorized to charge any underpayment of fees to Deposit Account No. 50-2207. This paper is submitted in duplicate.

Respectfully submitted,
Perkins Coie LLP

Date: March 29 , 2004

Carina M. Tan

Carina M. Tan
Registration No. 45,769

Correspondence Address:
Customer No. 22918
Perkins Coie LLP
P. O. Box 2168
Menlo Park, California 94026-2168
(650) 838-4300

RECEIVED
CENTRAL FAX CENTER
JUN 08 2004

EXPRESS MAIL LABEL NO. EV 099152888 US

OFFICIAL

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:

CHEYER et al.

Serial No.: 09/225,198

Filed on: January 5, 1999

Atty Dkt. No. 59501-8016.US01

Group Art Unit No.: 2126

Examiner: L. A. Bullock, Jr.

For: SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS

Mail Stop AF
Commissioner of Patents
P. O. Box 1450
Alexandria, VA 22313-1450

AMENDMENT AND RESPONSE

Sir:

This is in response to the Final Office Action mailed November 28, 2003, the shortened statutory period for which runs until February 28, 2004.

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

IN THE CLAIMS

1. (Currently 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, 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; and

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. (Previously presented) 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. (Previously presented) 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. (original) 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. original) A computer-implemented method as recited in claim 1 further comprising the act of providing an agent registry data structure.

6. (original) 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. (original) 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. (original) 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. (original) 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. (original) 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. (original) A computer-implemented method as recited in claim 1 further comprising the act of establishing communication between the plurality of distributed agents.

12. (original) 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. (original) 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. (original) A computer-implemented method as recited in claim 13 wherein the natural language query was generated by a user interface agent.

15. (original) A computer-implemented method as recited in claim 1, wherein the base goal requires setting a trigger having conditional functionality and consequential functionality.

16. (original) 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. (original) 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. (original) 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. (original) 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. (original) A computer-implemented method as recited in claim 15 wherein the trigger is installed and executed within the facilitator agent.

21. (original) A computer-implemented method as recited in claim 15 wherein the trigger is installed and executed within a first service-providing agent.

22. (original) A computer-implemented method as recited in claim 15 wherein the conditional functionality of the trigger is installed on a facilitator agent.

23. (original) 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. (original) 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. (original) A computer-implemented method as recited in claim 15 wherein the consequential functionality of the trigger is installed on a facilitator agent.

26. (original) A computer-implemented method as recited in claim 1 wherein the base goal is a compound goal having sub-goals separated by operators.

27. (original) 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. (original) 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. (Currently amended) 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), wherein the ICL includes:

a layer of conversational protocol defined by event types and parameter lists associated with one or more of the 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 requested 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 to best complete the requested service; and

implementing the base goal satisfaction plan.

30. (original) 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. (original) 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. (original) 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. (original) 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. (original) 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. (original) 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. (original) 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. (original) A computer program as recited in claim 36 wherein the natural language query is generated by a user interface agent.

38. (original) 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. (original) 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. (original) 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. (original) 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. (original) 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. (original) A computer program as recited in claim 38 further including computer executable instructions for installing and executing the trigger within the facilitator agent.

44. (original) 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. (original) A computer program as recited in claim 29 further including computer executable instructions for interpreting compound goals having sub-goals separated by operators.

46. (original) 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. (original) 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. (Currently 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 of:

a layer of conversational protocol defined by event types and parameter lists associated with one or more of the 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 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 conditional execution operator; and

a parallel disjunctive operation that indicates that disjunct goals are to be performed by different agents.

49. (original) An ICL as recited in claim 48, wherein the ICL is computer platform independent.

50. (original) 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. (original) 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. (original) 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. (original) An ICL as recited in claim 51 wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

54. (original) An ICL as recited in claim 48 wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

55. (original) 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. (original) An ICL as recited in claim 55 wherein an electronic agent's solvables define an interface for the electronic agent.

57. (original) An ICL as recited in claim 56 wherein the facilitator agent maintains an agent registry making available a plurality of electronic agent interfaces.

58. (original) 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. (original) 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. (original) 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. (Currently amended) 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), wherein the ICL includes:

a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events; and

a content layer comprising one or more of goals, triggers and data elements associated with the events;

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. (original) 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. (original) 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. (original) 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. (original) 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. (original) 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. (original) A facilitator agent as recited in claim 66 wherein the data repository is local to the facilitator agent.

68. (original) A facilitator agent as recited in claim 66 wherein the data repository is remote from the facilitator agent.

69. (original) A facilitator agent as recited in claim 64 wherein the trigger mechanism is a task trigger having a set of conditions.

70. (original) 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. (Currently amended) 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~~

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; 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. (Currently amended) A computer architecture as recited in claim 71, wherein the ~~basis for the computer architect is an~~ 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 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. (original) A computer architecture as recited in claim 72, wherein the ICL is computer platform independent.

74. (original) 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. (original) A computer architecture as recited in claim 73 wherein the ICL syntax supports explicit task completion constraints within goal expressions.

76. (original) 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. (original) A computer architecture as recited in claim 75 wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

78. (original) A computer architecture as recited in claim 73 wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

79. (original) 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. (original) A computer architecture as recited in claim 79 wherein an electronic agent's solvables define an interface for the electronic agent.

81. (original) 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. (original) 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. (original) 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. (Previously presented) 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. (Previously presented) 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. (Currently 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, <u>and an Interagent Communication Language (ICL), wherein the ICL includes:</u>

<u>a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events; and</u>

<u>a content layer comprising one or more of goals, triggers and data elements associated with the events;</u>

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. (Previously presented) 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. (Previously presented) 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. (original) 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.

REMARKS

INTERVIEW:

A telephonic interview was conducted on March 11, 2004. The participants were Examiner Lewis A. Bullock, Jr., David Stringer-Calvert and Carina M. Tan. During the interview, an agreement with respect to all the claims were reached. Applicants argued that the prior art teachings of *KISS* did not disclose any intelligent reasoning when formulating a goal satisfaction plan. Applicants argued that *KISS* merely discloses a method of information retrieval from information repositories such as databases. The examiner disagreed. However, the examiner pointed out that certain features in Applicant's specification regarding ICL are novel. The Examiner indicated that the ICL features: 1) a conversational protocol layer, and 2) a content layer, would distinguish applicants' claims over the prior art. It was agreed that applicants would submit a response amending the claims to include the above novel ICL features.

The Examiner is thanked for the performance of a thorough search. By this response, claims 1, 29, 48, 61, 71, 72 and 86 have been amended. No claims have been cancelled or added. Hence, Claims 1-89 are pending in the Application.

IN THE SPECIFICATION

Compact Disc Containing Appendices

Applicants cancel the computer program listing appearing in the specification in Appendices A, B, C, D, and E. In compliance with 37 CFR 1.96(c), Applicants enclose a CD-ROM labeled as Copy 1 and an identical copy of the CD-ROM labeled as Copy 2 containing the identical contents of Appendices A, B, C, D and E as filed with the patent application on January 5, 1999.

Substitute Pages Of Specification

Enclosed are substitute Pages 1, 8 and 9. Substitute Page 1 of the specification has been amended to identify the compact disc and list the file names, size, and creation date of each file, and substitute Page 8 and Page 9 which have been amended to delete the "Brief Description of the Appendices." Also enclosed is a substitute ABSTRACT containing less than 150 words. The ABSTRACT as originally filed contained more than 150 words.

SUMMARY OF REJECTIONS/OBJECTIONS

In the Office Action, Claims 1-3, 5-11, 15-25, 29-34, 38-44, and 61-71 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Developing Tools for the Open Agent Architecture" by Martin1 in view of U.S. Patent No. 6,484,155 issued to Kiss.

Claims 4, 12-14, 26-28, 35-37, 45-47, and 72-85 are rejected under 35 U.S.C. 103(a) as being unpatentable over Martin1 in view of Kiss, and further in vie of "Information Brokering in an Agent Architecture" by Martin2.

Claims 48-60 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.

REJECTIONS UNDER 35 U.S.C. § 103(a)

CLAIMS 1, 29, 61, 71 and 86

Claim 1, as amended, recites in part, the features:

"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; and

a content layer comprising one or more of goals, triggers and data elements associated with the events;

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;"

Claim 1 includes the limitation of a inter-agent language, wherein the inter-agent language includes 1) a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, and 2) a content layer comprising one or more of goals, triggers and data elements associated with the events. The cited references do not disclose or suggest such a conversational protocol and content layer.

Further, the Office Action states that the "dynamic solution plan" in *KISS* is the equivalent of the "goal satisfaction plan" of applicants' Claim 1 above. The Office Action points to col. 5, lines 14-45; col. 8, line 21 - col. 9, line 26; and col. 10, lines 10-38, and col. 2, lines 50-67 for support.

The method for forming the "dynamic solution plan" in *KISS* is irrelevant to the method of forming the goal satisfaction plan in Applicants' Claim 1. It is respectfully submitted that *KISS* is irrelevant because *KISS* is an invention involving accessing knowledge repositories.' Such knowledge repositories are represented by "knowledge agents." The Abstract of *KISS* states that "the invention solicits accessible knowledge repositories, represented by knowledge agents, for relevant knowledge..."

In other words, *KISS* is merely a method of information retrieval from information repositories or data sources. For example, the meta agent can ask questions involving facts or data and the agents attempt to retrieve the facts or data from the corresponding data repository. In contrast, the goal satisfaction plan of Claim 1 involves asking service providing agents to perform **actions** such as boil water, roast coffee beans, grind the roasted coffee beans as opposed to merely asking the agents to retrieve information from an information repository.

To further explain why *KISS* is irrelevant and completely different from the method of Claim 1, see col. 5 lines 39-43 where "[t]he meta agent 119 is configured to begin executing the solution plan even before the plan is complete." This underscores the fact that the solution plan in *KISS* merely involves information retrieval rather than asking the agent to perform intelligent actions such as roast coffee beans. In *KISS*, it is not fatal to begin executing the solution plan even before the plan is complete because no real harm is done if the meta agent begins by asking the wrong questions. To explain, *KISS* teaches "the meta agent 119 is capable of backtracking or replanning to permit escape from a dead-end." In other words, it is not fatal if the search for data is proceeding down an incorrect search path, as explained in *KISS*. In contrast, the facilitator of Claim 1 cannot begin execution of the goal satisfaction plan before the goal satisfaction plan is complete. For example, it would be fatal for the facilitator to ask a service-providing agent to boil the coffee beans instead of requesting that the coffee beans be first roasted and then ground. Such an action of boiling the coffee beans would be **irreversible** and would produce soggy beans. In other words, the service-providing agents of Claim 1 perform actions and are not merely sources of information.

Further, *KISS* does not use reasoning for "formulating the dynamic solution plan." In other words, *KISS* does not use the inferencing schemes as described in column 7 for generating the solution plan. In fact, *KISS* teaches away from using reasoning or inferencing for generating the solution plan. Column 8, lines 58-61 of *KISS* states that "**[a]fter** the solution plan is formulated, the meta agent 119 implements a distributed inference process to perform the search and execution phases of solving the problem, while maintaining control of the process" (emphasis added). Thus, the inference process is what the solution plan in *KISS* accomplishes and is not what is used to generate the solution plan.

In contrast, Claim 1 shows that the facilitating engine uses sophisticated reasoning when delegating sub-goal requests to best complete the requested service request. The facilitating engine's use of reasoning is supported by the specification on page 13, lines 342-347.

Assume that the facilitator agent of Claim 1 receives a request such as, "Make Coffee". The facilitator agent's facilitating engine uses reasoning to generate the following goal satisfaction plan:

Sub-goal request A: Please perform the act of roasting coffee beans
Sub-goal request B: Please perform the act of grinding coffee beans
Sub-goal request C: Please perform the act of boiling water, etc.

The facilitating engine is able to use reasoning to accomplish the base goal, "Make Coffee" by asking an appropriate agents to first roast the coffee beans before asking the agent to grind the beans, etc.

Neither *Cohen* nor *KISS*, either alone or in combination, disclose, teach, suggest or make obvious the novel features of claim 1. Thus, Claim 1 is allowable.

Claims 29, 61, 71 and 86, each contain similar features regarding "using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal." Thus, Claims 29, 61, 71 and 86 are allowable for at least the reasons provided herein in respect to Claim 1.

CLAIMS 2-28, 30-47, 62-70, 72-85 and 87-89

Claims 2-28 are either directly or indirectly dependent upon Claim 1 and include all the limitations of Claim 1 and therefore are allowable for at least the reasons provided herein in respect to Claim 1.

Claims 30-47 are either directly or indirectly dependent upon Claim 29 and include all the limitations of Claim 29 and therefore are allowable for at least the reasons provided herein in respect to Claim 29.

Claims 62-70 are either directly or indirectly dependent upon Claim 61 and include all the limitations of Claim 61 and therefore are allowable for at least the reasons provided herein in respect to Claim 61.

Claims 72-85 are either directly or indirectly dependent upon Claim 71 and include all the limitations of Claim 71 and therefore are allowable for at least the reasons provided herein in respect to Claim 71

Claims 87-89 are either directly or indirectly dependent upon Claim 86 and include all the limitations of Claim 86 and therefore are allowable for at least the reasons provided herein in respect to Claim 86.

CLAIM 48

Claim 48 as amended, recites in part:

"the ICL having one or more of:

a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events; and
a content layer comprising one or more of goals, triggers and data elements associated with the events;

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 operator that indicates that disjunct goals are to be performed by different agents."

The novel method recited in Claim 48 as amended requires that the inter-agent language include 1) a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, and 2) a content layer comprising one or more of goals, triggers and data elements associated with the events. The cited references do not disclose or suggest such a conversational protocol and content layer.

Further, the novel method recited in Claim 48 as amended requires that "goals within a single request" are "coupled by one or more operators from a set of operators". In amended Claim 48, the set of operators comprise, a conditional execution operator, and **a parallel disjunctive operator**.

In the Office Action, the Examiner states that triggers are conditional operators. It is respectfully submitted that triggers are not conditional operators in the sense of an being a syntactical operator in an expression.

Further, the Office Action states that page 10 of *Martin2* discloses **parallel disjunctive operators.** *Martin2* does NOT disclose parallel disjunctive operators. The "disjunction" in *Martin2* is the run-of-the-mill Prolog style disjunction. The expression, "Do task A OR Do Task B," is an example of a *Martin2* type disjunction. In contrast, a "**parallel disjunctive operator** is an operator that indicates that disjunct goals are to be performed by different agents. An example of a **parallel disjunctive operator** expression is "Ask agent Bob to do task A OR Ask agent Fred to do task B concurrently.

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 conditional execution operator** (such as "if" and "when", allowing for particular actions to be predicated on the state, or outcomes of earlier actions), and **a parallel disjunctive operator** (allowing for alternative actions to be performed at the same time, if resources allow, and a first-to-respond strategy may be used in their competition to perform the goal at hand). 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

Date: March 29, 2004

Carina M. Tan
Carina M. Tan
Registration No. 45,769

Correspondence Address:

Customer No. 22918
Perkins Coie LLP
P. O. Box 2168
Menlo Park, California 94026
(650) 838-4300

Software-Based Architecture for Communication and Cooperation Among Distributed Electronic Agents

By:
Adam J. Cheyer and David L. Martin

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

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*

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.

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

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.

RECEIVED
CENTRAL FAX CENTER
JUN 08 2004

OFFICIAL

Perkins Coie

101 Jefferson Drive
Menlo Park, CA 94025-1114
PHONE: 650.838.4300
FAX: 650.838.4350
www.perkinscoie.com

FACSIMILE COVER SHEET

CONFIDENTIAL AND PRIVILEGED

If there are any problems with this transmission, please call:
☐ *Sender's name and phone number

DATE: *June 8, 2004* COVER SHEET & 35 PAGE(S)

CLIENT NUMBER: *59501-8016.US01*

RETURN TO: (NAME) *Sharyl Brown* (EXT.) *4314* (ROOM NO.) *Error! No document variable supplied.*

ORIGINAL DOCUMENT(S) WILL BE: ☐ SENT TO YOU ☒ HELD IN OUR FILES

SENDER:	TELEPHONE:	FACSIMILE:
Sharyl Brown	*(650) 838-4314*	(650) 838-4350

RECIPIENT:	COMPANY:	TELEPHONE:	FACSIMILE:
Examiner L. A. Bullock, Jr.	*USPTO, Group Art Unit 2126*	*(703) 305-0439*	*(703)*872-9306

RE: ***Serial No. 09/225,198***
Atty. Dkt. No. 59501-8016.US01

Dear Examiner Bullock:

Pursuant to your request, attached hereto is a copy of the Amendment and Response which was filed on March 29, 2994, including the return postcard stamped by the USPTO.

We would appreciate receiving status of the Notice of Allowance at your earliest convenience.

If you have any questions or comments, please contact Carina Tan, Reg. No. 45,769 at (650) 838-4311.

Sincerely,
PERKINS COIE LLP

Sharyl Brown

Sharyl Brown
Secretary to Carina M. Tan

BEST AVAILABLE COPY

Attorney Docket No.:	Date Mailed:	Express Mail No.
59501-8016.US01	March 29, 2004	EV 099152888 US

Applicant: CHEYER et al.

Application No.: 09/225,198

Filing Date: January 5, 1999

Title: SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS

Papers Enclosed

- ☒ Check No. 2195 in the amount of $55.00
- ☒ Transmittal for Amendment and Response...
- ☒ Amendment and Response
- ☒ Copy 1 and Copy 2 of Compact Disc both containing the identical contents of Appendices A, B, C, D, and E as filed with the patent application on January 5, 1999

CMT

Received by the U.S. Patent and Trademark Office

OIPE MAR 2 9 2004 PATENT & TRADEMARK OFFICE

2195

PERKINS COIE LLP
BAY AREA PATENT
101 JEFFERSON DRIVE
MENLO PARK, CA 94025-1114

US BANK
1420 5TH
SEATTLE, WA 98101
19-10/1250

3/26/2004

PAY TO THE ORDER OF Commissioner for Patents $ **55.00

Fifty-Five and 00/100*** DOLLARS

UNITED STATES PATENT and TRADEMARK OFFICE

Shellina Berger

MEMO 59501-8016.US01

⑈002195⑈ ⑆125000105⑆153592271574⑈

PERKINS COIE LLP 2195

Commissioner for Patents	Carina Tan	3/26/2004	55.00

U. S. Bank	59501-8016.US01	55.00

PERKINS COIE LLP 2195

Commissioner for Patents	Carina Tan	3/26/2004	55.00

U. S. Bank	59501-8016.US01	55.00

Attorney Docket No. 59501-8016.US01

EXPRESS MAIL LABEL NO. EV 099152888 US

Applicants: CHEYER et al.
Application No.: 09/225,198
Filed: January 5, 1999
Examiner: L. A. Bullock, Jr.
Group Art Unit 2151
For: **SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS**

Mail Stop AF
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL FOR AMENDMENT AND RESPONSE AND COMPUTER PROGRAM LISTING APPENDIX SUBMITTED ON COMPACT DISC

Sir:

This is in response to the Final Office Action mail by the U.S. Patent and Trademark Office on November 28, 2003. Applicants request a one month extension of time, thus allowing Applicants until March 28, 2004 to respond.

1. Transmitted herewith are the following:

   ☒ Check No. 2195 in the amount of $55.00
   ☒ Amendment and Response
   ☒ Copy 1 and Copy 2 of Compact Disc both containing the identical contents of Appendices A, B, C, D, and E as filed with the patent application on January 5, 1999.

2. Machine format is ISO-9660 file system:

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

Attorney Docket No. 59501-8016.US01

3. Fee Authorization

Check No. 2195 in the amount of $55.00 is enclosed for the required fees for one month extension of time, however, the Commissioner is authorized to charge any underpayment of fees to Deposit Account No. 50-2207. This paper is submitted in duplicate.

Respectfully submitted,
Perkins Coie LLP

Carina M. Tan

Date: March 29, 2004

Carina M. Tan
Registration No. 45,769

Correspondence Address:
Customer No. 22918
Perkins Coie LLP
P. O. Box 2168
Menlo Park, California 94026-2168
(650) 838-4300

RC

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
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

22918 7590 07/12/2004

PERKINS COIE LLP
P.O. BOX 2168
MENLO PARK, CA 94026

EXAMINER	
BULLOCK JR; LEWIS ALEXANDER	
ART UNIT	**PAPER NUMBER**
2126	

DATE MAILED: 07/12/2004

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

BEST AVAILABLE COPY

PTO-90C (Rev. 10/03)

Advisory Action	Application No.	Applicant(s)	
	09/225,198	CHEYER ET AL.	
	Examiner	Art Unit	
	Lewis A. Bullock, Jr.	2126	

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

THE REPLY FILED 08 June 2004 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): CD Requirements and Abstract objections.

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: *See Continuation Sheet*.

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: *1-89*.

Claim(s) withdrawn from consideration: _____.

BEST AVAILABLE COPY

8.☐ The 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: _____

L A Bullock Jr

Continuation of 2. NOTE: Applicant amended the claims to language that overcomes the prior art references, however, the examiner has been able to find references that meets the new claim limitations.

Continuation of 5. does NOT place the application in condition for allowance because: Applicant's arguments are unpersuasive. Applicants amendment of the agent language including a conversational protocol layer and a content layer would overcome the applied prior art references, however, the examiner has now found references that teach KQML having a a layer of conversational protocol defined by event types, i.e. a type of ask (ask one or ask_all primitive) along with parameters associated with the event types and a content layer comprising data elements associated with the event as disclosed in all independent claims. Also regarding claim 48, prior art references published by some of the Applicants detailed that ICL has either one of the layers, in particular the content layer, as disclosed in that claim however, the references do not allude to the ICL having both layers. Page 17, lines 12-30 attempts to illustrate that the events are different from the communication acts of KQML, however, the Examiner has not been able to ascertain how they are different from this portion of the specification or any other parts of the specification. It would seem that KQML's ask primitives are events that contain parameter information. Applicant would have to amend the claims or explain how the primitives of KQML would not represent events in order for the Examiner to not equate a layer of KQML primitives having parameter data to Applicant's conversational protocol layer defining events. In regards to claims 1-47 and 61-89, Applicant argues that the applied references, in particular Kiss, teaches the knowledge repository are represented by knowledge agents and merely ask the agents to retrieve information and is irrevelevant to Applicants method of forming the goal satisfaction plan in order to perform actions. The examiner disagrees. The examiner cannot find any language within the claims that details that the service is not a data retrieval service. Therefore, the plan generated to retrieve information is a satisfaction plan to perform actions, i.e. to retrieve the data. In addition, Applicant's example of actions such as boil water, roast coffee beans, and grind the roasted coffee beans are illustrated actions that the invention could perform when solving a goal. It is equally seen from the claim language that the actions can also be the tasks distributed by the meta agent when processing its solution plan to accomplish its overall goal. Applicant argues that the meta agent is capable of backtracking and replanning is another illustrations that Kiss does not teach the invention. In response, the Examiner cannot find any limitations that the plan can not be reevaluated or modified while being implemented. Therefore, the teachings of Kiss just adds another benefit, but still meets the limitations of the claims as disclosed. Applicant then argues that Kiss does not teach using reasoning to formulate the dynamic solution plan. The examiner disagrees. Column 5, lines 25-27 detail that the meta agent contains knowledge of problem solving methodologies and distributed inferencing procedures. Column 5, lines 30-32, detail that the meta agent may maintain the domain-specific knowledge necessary to answer the query itself. Column 5, lines 33-39 detail that meta agent formulates a solution plan and formulates sub-plans in order to perform iterative and recursive procedures. Therefore, the solution plan is generated by the planning component of the meta agent based on domain independent coordination strategies or domain specific reasoning. The cited paragraph Applicant refers to refute the teachings of Kiss refers to how the plan is replanned and backtracked. Applicant then argues that in regards to claim 48, the combination, i.e. Martin1 and Martin2, do not teach a single request are coupled by one or more operators from a set of operators comprising a conditional execution operator or a parallel disjunctive operator. The examiner disagrees. First, it is pointed out that only one operator has to be shown in order for the limitation to be met. Applicant discloses that a conditional execution operator is represented by an arrow (pg. 23, lines 2-5). Page 10, details a mapping rule (request) submitted in ICL format by an information agent which denotes an arrow as well as other control operators that affect the interpretation of a rule. Therefore, the cited reference teaches conditional execution operators and meets the claim language as disclosed.

RECEIVED
CENTRAL FAX CENTER
JUN 08 2004

EXPRESS MAIL LABEL NO. EV 099152888 US

OFFICIAL

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: CHEYER et al.

Atty Dkt. No. 59501-8016.US01

Group Art Unit No.: 2126

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

Mail Stop AF
Commissioner of Patents
P. O. Box 1450
Alexandria, VA 22313-1450

AMENDMENT AND RESPONSE

Sir:

This is in response to the Final Office Action mailed November 28, 2003, the shortened statutory period for which runs until February 28, 2004.

ENTER IN PART
ENTER AMENDMENTS TO SPECIFICATION & ABSTRACT

DO NOT ENTER AMENDMENT TO CLAIMS

fes 7/7/04

BEST AVAILABLE COPY

CONFIDENTIAL PRIVILEGED

Perkins Coie LLP-Menlo Park

RECEIVED
CENTRAL FAX CENTER
AUG 2 5 2004

Facsimile Transmittal Sheet

Date: August 25, 2004

Please confirm receipt ☒

Total Number of Pages (including cover sheet): 24

Confirmation by mail ☐

Attorney Docket No.: 59501-8016.US01

To:	From:
Name: Examiner L. A. Bullock, Jr.	Name: Carina M. Tan
Company: USPTO	Company: Perkins Coie LLP
FAX No.: (703) 872-9306	Phone No.: 650 838-4311
	FAX No.: 650 838-4350

I HEREBY CERTIFY THAT THIS CORRESPONDENCE IS BEING TRANSMITTED VIA FACSIMILE TO (703) 872-9306, THE UNITED STATES PATENT AND TRADEMARK OFFICE, ALEXANDRIA, VA, ON:

Date: August 25, 2004 By: Sharyl Brown

Sharyl Brown

Re: Serial No.: 09/225,198
Filing Date: January 5, 1999

Dear Examiner Bullock:

Attached hereto please find a Transmittal for Supplemental Amendment and Response (in duplicate) and a Supplemental Amendment and Response for the above-identified patent application.

Respectfully submitted,
Perkins Coie LLP

Carina M. Tan

Carina M. Tan
Registration No. 45,769

Perkins Coie LLP • 101 Jefferson Drive • Menlo Park, CA 94025

Attorney Docket No. 59501-8016.US01

CERTIFICATE OF FACSIMILE TRANSMISSION (37 CFR 1.8a)
I hereby certify that this correspondence is being transmitted to the United States Patent & Trademark Office, Central Fax Service Center via facsimile number (703) 872-9306 on August 25, 2004.

Date: August 25, 2004 By: Sharyl Brown
Sharyl Brown

RECEIVED
CENTRAL FAX CENTER
AUG 25 2004

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**

Mail Stop AF
Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

Transmittal For Supplemental Amendment and Response

Sir:

1. Transmitted herewith are the following:
   ☒ Supplemental Amendment and Response
   ☒ Facsimile Cover Sheet

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

   Applicants believe that no fees are due, however, the Commissioner is authorized to charge any underpayment in fees for timely filing to Deposit Account No. 50-2207.

Respectfully submitted,
Perkins Coie LLP

Date: August 25, 2004

Carina M. Tan
Carina M. Tan
Registration No. 45,769

Correspondence Address:
Customer No. 22918
Perkins Coie LLP
P.O. Box 2168
Menlo Park, CA 94
(650) 838-4300

[59501-8016/BY042380.033]

CERTIFICATE OF FACSIMILE TRANSMISSION (37 CFR 1.8a)
I hereby certify that this correspondence is being transmitted to the United States Patent & Trademark Office, Central Fax Service Center via facsimile number (703) 872-9306 on August 25, 2004.

Date: August 25, 2004

By: Sharyl Brown
Sharyl Brown

RECEIVED
CENTRAL FAX CENTER
AUG 2 5 2004

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: CHEYER et al.

Atty Dkt. No. 59501-8016.US01

Group Art Unit No.: 2126

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

Mail Stop AF
Commissioner of Patents
P. O. Box 1450
Alexandria, VA 22313-1450

SUPPLEMENTAL AMENDMENT AND RESPONSE

Sir:

This is a supplemental amendment to the Final Office Action mailed November 28, 2003, the shortened statutory period for which runs until February 28, 2004. A first amendment and response to Final Office Action mailed November 28, 2003 was filed on March 29, 2004.

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: ______________________

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

IN THE CLAIMS

1. (Currently 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, 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. (Previously presented) 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. (Previously presented) 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. (original) 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. original) A computer-implemented method as recited in claim 1 further comprising the act of providing an agent registry data structure.

6. (original) 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. (original) 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. (original) 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. (original) 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. (original) 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. (original) A computer-implemented method as recited in claim 1 further comprising the act of establishing communication between the plurality of distributed agents.

12. (original) 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. (original) 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. (original) A computer-implemented method as recited in claim 13 wherein the natural language query was generated by a user interface agent.

15. (original) A computer-implemented method as recited in claim 1, wherein the base goal requires setting a trigger having conditional functionality and consequential functionality.

16. (original) 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. (original) 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. (original) 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. (original) 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. (original) A computer-implemented method as recited in claim 15 wherein the trigger is installed and executed within the facilitator agent.

21. (original) A computer-implemented method as recited in claim 15 wherein the trigger is installed and executed within a first service-providing agent.

22. (original) A computer-implemented method as recited in claim 15 wherein the conditional functionality of the trigger is installed on a facilitator agent.

23. (original) 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. (original) 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. (original) A computer-implemented method as recited in claim 15 wherein the consequential functionality of the trigger is installed on a facilitator agent.

26. (original) A computer-implemented method as recited in claim 1 wherein the base goal is a compound goal having sub-goals separated by operators.

27. (original) 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. (original) 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. (Currently amended) 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), 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;

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 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 to best complete the requested service; and

implementing the base goal satisfaction plan.

30. (original) 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. (original) 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. (original) 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. (original) 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. (original) 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. (original) 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. (original) 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. (original) A computer program as recited in claim 36 wherein the natural language query is generated by a user interface agent.

38. (original) 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. (original) 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. (original) 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. (original) 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. (original) 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. (original) A computer program as recited in claim 38 further including computer executable instructions for installing and executing the trigger within the facilitator agent.

44. (original) 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. (original) A computer program as recited in claim 29 further including computer executable instructions for interpreting compound goals having sub-goals separated by operators.

46. (original) 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. (original) 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. (Currently 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:

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 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 conditional execution operator; and

a parallel disjunctive operation that indicates that disjunct goals are to be performed by different agents.

49. (original) An ICL as recited in claim 48, wherein the ICL is computer platform independent.

50. (original) 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. (original) 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. (original) 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. (original) An ICL as recited in claim 51 wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

54. (original) An ICL as recited in claim 48 wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

55. (original) 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. (original) An ICL as recited in claim 55 wherein an electronic agent's solvables define an interface for the electronic agent.

57. (original) An ICL as recited in claim 56 wherein the facilitator agent maintains an agent registry making available a plurality of electronic agent interfaces.

58. (original) 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. (original) 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. (original) 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. (Currently amended) 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), 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. (original) 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. (original) 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. (original) 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. (original) 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. (original) 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. (original) A facilitator agent as recited in claim 66 wherein the data repository is local to the facilitator agent.

68. (original) A facilitator agent as recited in claim 66 wherein the data repository is remote from the facilitator agent.

69. (original) A facilitator agent as recited in claim 64 wherein the trigger mechanism is a task trigger having a set of conditions.

70. (original) 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. (Currently amended) 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. (Previously presented) 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 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. (original) A computer architecture as recited in claim 72, wherein the ICL is computer platform independent.

74. (original) 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. (original) A computer architecture as recited in claim 73 wherein the ICL syntax supports explicit task completion constraints within goal expressions.

76. (original) 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. (original) A computer architecture as recited in claim 75 wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

78. (original) A computer architecture as recited in claim 73 wherein the ICL syntax supports explicit task completion advisory suggestions within goal expressions.

79. (original) 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. (original) A computer architecture as recited in claim 79 wherein an electronic agent's solvables define an interface for the electronic agent.

81. (original) 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. (original) 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. (original) 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. (Previously presented) 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. (Previously presented) 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. (Currently 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, 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. (Previously presented) 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. (Previously presented) 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. (original) 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.

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: ____________________________

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

REMARKS

INTERVIEW:

A telephonic interview was conducted on August 10, 2004. The participants were Examiner Lewis A. Bullock, Jr., and Carina M. Tan. During the interview, an agreement with respect to all the claims was reached. Applicants distinguished KQML from ICL.

The Examiner is thanked for the performance of a thorough search. By this response, claims 1, 29, 48, 61, 71, and 86 have been amended. No claims have been cancelled or added. Hence, Claims 1-89 are pending in the Application.

CONCLUSION

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

Date: August 25, 2004

Carina M. Tan
Carina M. Tan
Registration No. 45,769

Correspondence Address:

Customer No. 22918
Perkins Coie LLP
P. O. Box 2168
Menlo Park, California 94026
(650) 838-4300

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

NOTICE OF ALLOWANCE AND FEE(S) DUE

22918 7590 09/10/2004

PERKINS COIE LLP
P.O. BOX 2168
MENLO PARK, CA 94026

EXAMINER	
BULLOCK JR, LEWIS ALEXANDER	
ART UNIT	PAPER NUMBER
2126	

DATE MAILED: 09/10/2004

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/225,198	01/05/1999	ADAM J. CHEYER	SRI1P016	2756

TITLE OF INVENTION: SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1330	$0	$1330	12/10/2004

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 above is to be removed, check box 5b on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and twice the amount of the ISSUE FEE shown above, 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 box 5a on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and 1/2 the ISSUE FEE shown above.

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 Mail Stop 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. 09/04) Approved for use through 04/30/2007.

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

or Fax **(703) 746-4000**

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 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: Use Block 1 for any change of address)

22918 7590 09/10/2004

PERKINS COIE LLP
P.O. BOX 2168
MENLO PARK, CA 94026

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 Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (703) 746-4000, on the date indicated below.

(Depositor's name)

(Signature)

(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/225,198	01/05/1999	ADAM J. CHEYER	SRI1P016	2756

TITLE OF INVENTION: SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1330	$0	$1330	12/10/2004

EXAMINER	ART UNIT	CLASS-SUBCLASS
BULLOCK JR, LEWIS ALEXANDER	2126	709-310000

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. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. 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 (No small entity discount permitted)

☐ 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 Director is hereby authorized by charge the required fee(s), or credit any overpayment, to Deposit Account Number ____________ (enclose an extra copy of this form).

5. **Change in Entity Status** (from status indicated above)

☐ a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27. ☐ b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

The Director of the USPTO 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.
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.

Authorized Signature ____________ Date ____________

Typed or printed name ____________ Registration No. ____________

PTOL-85 (Rev. 09/04) Approved for use through 04/30/2007. 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 FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
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

22918 7590 09/10/2004

PERKINS COIE LLP
P.O. BOX 2168
MENLO PARK, CA 94026

EXAMINER	
BULLOCK JR, LEWIS ALEXANDER	
ART UNIT	PAPER NUMBER
2126	

DATE MAILED: 09/10/2004

Determination of Patent Term Extension under 35 U.S.C. 154 (b)

(application filed after June 7, 1995 but prior to May 29, 2000)

The Patent Term Extension is 0 day(s). Any patent to issue from the above-identified application will include an indication of the 0 day extension on the front page.

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Extension 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) WEB site (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. 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. 09/04) Approved for use through 04/30/2007.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
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

22918 7590 09/10/2004

PERKINS COIE LLP
P.O. BOX 2168
MENLO PARK, CA 94026

EXAMINER	
BULLOCK JR, LEWIS ALEXANDER	
ART UNIT	PAPER NUMBER
2126	

DATE MAILED: 09/10/2004

Notice of Fee Increase on October 1, 2004

If a reply to a "Notice of Allowance and Fee(s) Due" is filed in the Office on or after October 1, 2004, then the amount due will be higher than that set forth in the "Notice of Allowance and Fee(s) Due" because some fees will increase effective October 1, 2004. See Revision of Patent Fees for Fiscal Year 2005; Final Rule, 69 Fed. Reg. 52604, 52606 (May 10, 2004).

The current fee schedule is accessible from WEB site (http://www.uspto.gov/main/howtofees.htm).

If the 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 of Pay Balance of Issue Fee" will be mailed to applicant. In order to avoid processing delays associated with mailing of a "Notice of Pay Balance of Issue Fee," if the response to the Notice of Allowance is to be filed on or after October 1, 2004 (or mailed with a certificate of mailing on or after October 1, 2004), the issue fee paid should be the fee that is required at the time the fee is paid. See Manual of Patent Examining Procedure (MPEP), Section 1306 (Eighth Edition, Rev. 2, May 2004). 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 MPEP Section 1308.01.

Effective October 1, 2004, 37 CFR 1.18 is amended by revising paragraphs (a) through (c) to read as set forth below.

Section 1.18 Patent post allowance (including issue) fees.

(a) Issue fee for issuing each original or reissue patent, except a design or plant patent:
By a small entity (Sec. 1.27(a))...................... $685.00
By other than a small entity......................... $1,370.00
(b) Issue fee for issuing a design patent:
By a small entity (Sec. 1.27(a))...................... $245.00
By other than a small entity........................... $490.00
(c) Issue fee for issuing a plant patent:
By a small entity (Sec. 1.27(a))...................... $330.00
By other than a small entity........................... $660.00

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. 09/04) Approved for use through 04/30/2007.

Notice of Allowability	Application No. 09/225,198	Applicant(s) CHEYER ET AL.
	Examiner Lewis A. Bullock, Jr.	Art Unit 2126

-- 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 8/25/04.
2. ☒ The allowed claim(s) is/are 1-89.
3. ☒ The drawings filed on 05 January 1999 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: _____.

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.

5. ☐ 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.
6. ☐ CORRECTED DRAWINGS (as "replacement sheets") 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./Mail Date _____.
   (b) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

   Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).
7. ☐ 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 Draftperson's Patent Drawing Review (PTO-948)
3. ☐ Information Disclosure Statements (PTO-1449 or PTO/SB/08), Paper No./Mail Date _____
4. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
5. ☐ Notice of Informal Patent Application (PTO-152)
6. ☒ Interview Summary (PTO-413), Paper No./Mail Date ______ .
7. ☒ Examiner's Amendment/Comment
8. ☒ Examiner's Statement of Reasons for Allowance
9. ☐ Other _____.

LEWIS A. BULLOCK, JR.
PRIMARY EXAMINER

EXAMINER'S AMENDMENT

1. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with Corina Tan on September 3, 2004.

The application has been amended as follows:

- The claims are amended as listed in the Attachment.

2. The following is an examiner's statement of reasons for allowance: All of the claims are allowable for at least the following reasons: All of the claims detail the inter-agent language including: a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameters lists further refine the one or more events; and a content layer comprising one or more goals, triggers and data elements associated with the events. The cited prior art of record do not teach the inter-agent language having the cited layers as disclosed. Prior Art article entitled, "Building Distributed Software Systems with the Open Agent Architecture", published by some of the inventors teaches the cited layers however, the reference has been disqualified by the 1.132 Affidavit filed on 11/25/02. In addition, prior art article "Software Agent Technologies" published by Nwana et al. teach an

agent communication language (KQML) that comprises three layers: a content layer, a message layer, and a communication layer. The content layer specifies the actual content of the message for which KQML standard itself has nothing to say about its structure (pg. 4). The message layer provides the performative that specifies the protocol for delivering the message that subsumes the content, i.e. the rules that agents must use when initiating and maintaining an exchange (pg. 5). The communication layer encodes low level communication parameters, such as the identities of the sender and the recipient, and unique identifiers for the particular speech act (pg. 5). The disclosed agent communication language does not read upon the cited agent language because the layer does not define an event type as well as the parameter lists that further refines the event. Nwana's language at best has separate layers for the event and the parameters associated with the event. By Applicant providing these parameters in the same layer as the event such that they further refine the event, a standard set of events are dynamically extensible based upon the parameter list which is not possible with the teachings of Nwana. Therefore, the claims are allowable over the prior art 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."

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, Meng An can be reached on (703) 305-9678. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

September 3, 2004

LEWIS A. BULLOCK, JR.
PRIMARY EXAMINER

Examiner-Initiated Interview Summary	Application No.	Applicant(s)
	09/225,198	CHEYER ET AL.
	Examiner	Art Unit
	Lewis A. Bullock, Jr.	2126

All Participants:

(1) Lewis A. Bullock, Jr..

(2) Corina Tan.

Status of Application: *Allowed*

(3) _____.

(4) _____.

Date of Interview: 2 September 2004

Time: _____

Type of Interview:

☒ Telephonic
☐ Video Conference
☐ Personal (Copy given to: ☐ Applicant ☐ Applicant's representative)

Exhibit Shown or Demonstrated: ☐ Yes ☒ No
If Yes, provide a brief description: .

Part I.

Rejection(s) discussed:
All

Claims discussed:
All

Prior art documents discussed:

Part II.

SUBSTANCE OF INTERVIEW DESCRIBING THE GENERAL NATURE OF WHAT WAS DISCUSSED:
See Continuation Sheet

Part III.

☒ It is not necessary for applicant to provide a separate record of the substance of the interview, since the interview directly resulted in the allowance of the application. The examiner will provide a written summary of the substance of the interview in the Notice of Allowability.

☐ It is not necessary for applicant to provide a separate record of the substance of the interview, since the interview did not result in resolution of all issues. A brief summary by the examiner appears in Part II above.

(Examiner/SPE Signature)

(Applicant/Applicant's Representative Signature – if appropriate)

Continuation of Substance of Interview including description of the general nature of what was discussed: In an informal interview, the examiner explained his position as disclosed in the 1st after final response. Applicant and the examiner agreed upon more language in the claims with the prior language that would place the application in condition for allowance as disclosed in the Reasons for allowance. The examiner also explained to Applicant that the after final response is non-compliant in that it is not readable in later pages, and the all new language is not underlined. The examiner will correct this defect by Examiner's Amendment..

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 2126	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-2003/0167247	09-2003	Masuoka, Ryusuke	706/46
	B	US-2001/0039562	11-2001	SATO, AKIRA	709/202
	C	US-			
	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	Nwana, Hyacinth et al. "Software Agent Technologies". BT Technology Journal. 1996.
	V	Busetta, Paolo et al. "The BDIM Agent Toolkit Design." 1997.
	W	Mayfield, James et al. "Desiderata for Agent Communication Languages." March 27-29,1995.
	X	Khedro, Taha et al. "Concurrent Endineering through Interoperable Software Agents. August 1994.

*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.

CERTIFICATE OF FACSIMILE TRANSMISSION (37 CFR 1.8a)
I hereby certify that this correspondence is being transmitted to the United States Patent & Trademark Office, Central Fax Service Center via facsimile number (703) 872-9306 on August 25, 2004.

Date: August 25, 2004

By: Sharyl Brown
Sharyl Brown

RECEIVED
CENTRAL FAX CENTER
AUG 25 2004

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:

CHEYER et al.

Serial No.: 09/225,198

Filed on: January 5, 1999

Atty Dkt. No. 59501-8016.US01

Group Art Unit No.: 2126

Examiner: L. A. Bullock, Jr.

For: SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS

Mail Stop AF
Commissioner of Patents
P. O. Box 1450
Alexandria, VA 22313-1450

SUPPLEMENTAL AMENDMENT AND RESPONSE

Sir:

Do Not ENTER jcb

This is a supplemental amendment to the Final Office Action mailed November 28, 2003, the shortened statutory period for which runs until February 28, 2004. A first amendment and response to Final Office Action mailed November 28, 2003 was filed on March 29, 2004.

SERIAL NUMBER	FILING DATE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
09/225,198	01/05/99	~~395~~ 719	~~2755~~ 2126	SRI1P016

APPLICANT

ADAM J. CHEYER, PALO ALTO, CA; DAVID L. MARTIN, SANTA CLARA, CA.

CONTINUING DOMESTIC DATA*******************
VERIFIED

None fcb

371 (NAT'L STAGE) DATA*******************
VERIFIED

None fcb

FOREIGN APPLICATIONS**********
VERIFIED

None fcb

FOREIGN FILING LICENSE GRANTED 01/28/99

Foreign Priority claimed / 35 USC 119 (a-d) conditions met / Verified and Acknowledged	STATE OR COUNTRY	SHEETS DRAWING	TOTAL CLAIMS	INDEPENDENT CLAIMS
☐yes ☒no / ☐yes ☒no ☐Met after Allowance / fcb Examiner's Initials _____ Initials	CA	16	89	6

ADDRESS

BRIAN R COLEMAN
HICKMAN STEPHENS & COLEMAN
P O BOX 52037
PALO ALTO CA 94303-0746

TITLE

SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS

FILING FEE RECEIVED		
$2,236	FEES: Authority has been given in Paper No. _______ to charge/credit DEPOSIT ACCOUNT NO. _______ for the following:	☐ All Fees ☐ 1.16 Fees (Filing) 1.17 Fees (Processing Ext. of time) 1.18 Fees (Issue) Other _______ Credit

Index of Claims

Application No.	Applicant(s)
09/225,198	CHEYER ET AL.
Examiner	**Art Unit**
Lewis A. Bullock, Jr.	2126

√	Rejected	—	(Through numeral) Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

Claim		Date								
Final	Original	7/17/02	3/3/03	11/28/03	9/3/04					
1	1	√	√	√	=					
2	2									
3	3									
4	4									
5	5									
6	6									
7	7									
8	8									
9	9									
10	10									
11	11									
12	12									
13	13									
14	14									
15	15									
16	16									
17	17									
18	18									
19	19									
20	20									
21	21									
22	22									
23	23									
24	24									
25	25									
26	26									
27	27									
28	28									
29	29									
30	30									
31	31									
32	32									
33	33									
34	34									
35	35									
36	36									
37	37									
38	38									
39	39									
40	40									
41	41									
42	42									
43	43									
44	44									
45	45									
46	46									
47	47									
48	48									
49	49									
50	50	√	√	√	=					

Claim		Date								
Final	Original	7/17/02	3/3/03	11/28/03	9/3/04					
51	51	√	√	√	=					
52	52									
53	53									
54	54									
55	55									
56	56									
57	57									
58	58									
59	59									
60	60									
61	61									
62	62									
63	63									
64	64									
65	65									
66	66									
67	67									
68	68									
69	69									
70	70									
71	71									
72	72									
73	73									
74	74									
75	75									
76	76									
77	77									
78	78									
79	79									
80	80									
81	81									
82	82									
83	83									
84	84									
85	85									
86	86									
87	87									
88	88									
89	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									

Issue Classification	Application No.	Applicant(s)	
	09/225,198	CHEYER ET AL.	
	Examiner	**Art Unit**	
	Lewis A. Bullock, Jr.	2126	

ISSUE CLASSIFICATION

ORIGINAL		CROSS REFERENCE(S)						
CLASS	SUBCLASS	CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)					
719	317	709	202					
INTERNATIONAL CLASSIFICATION		717	114					
G 0 6 F	09/54							
	/							
	/							
	/							
	/							

(Assistant Examiner) (Date)	Lewis A. Bullock, Jr. 09/03/04	Total Claims Allowed: 89
(Legal Instruments Examiner) (Date)	(Primary Examiner) (Date)	O.G. Print Claim(s): 1 / O.G. Print Fig.: 4

☒ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original
	1		31		61		91		121		151		181
	2		32		62		92		122		152		182
	3		33		63		93		123		153		183
	4		34		64		94		124		154		184
	5		35		65		95		125		155		185
	6		36		66		96		126		156		186
	7		37		67		97		127		157		187
	8		38		68		98		128		158		188
	9		39		69		99		129		159		189
	10		40		70		100		130		160		190
	11		41		71		101		131		161		191
	12		42		72		102		132		162		192
	13		43		73		103		133		163		193
	14		44		74		104		134		164		194
	15		45		75		105		135		165		195
	16		46		76		106		136		166		196
	17		47		77		107		137		167		197
	18		48		78		108		138		168		198
	19		49		79		109		139		169		199
	20		50		80		110		140		170		200
	21		51		81		111		141		171		201
	22		52		82		112		142		172		202
	23		53		83		113		143		173		203
	24		54		84		114		144		174		204
	25		55		85		115		145		175		205
	26		56		86		116		146		176		206
	27		57		87		117		147		177		207
	28		58		88		118		148		178		208
	29		59		89		119		149		179		209
	30		60		90		120		150		180		210

Search Notes

Application No.	Applicant(s)
09/225,198	CHEYER ET AL.
Examiner	**Art Unit**
Lewis A. Bullock, Jr.	2126

SEARCHED

Class	Subclass	Date	Examiner
719	317	8/31/2004	LAB
709	202	8/31/2004	LAB
717	114	8/31/2004	LAB

INTERFERENCE SEARCHED

Class	Subclass	Date	Examiner
719	**317**	**9/3**/2004	**LAB**
709	202	9/3/2004	LAB
717	114	9/3/2004	LAB

SEARCH NOTES (INCLUDING SEARCH STRATEGY)

	DATE	EXMR
EAST ACM IEEE CITESEER INTERNET	8/31/2004	LAB

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail   Mail Stop ISSUE FEE
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

or Fax (703) 746-4000

OIPE · OCT 0 4 2004 · PATENT & TRADEMARK OFFICE

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 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: Use Block 1 for any change of address)

22918 7590 09/10/2004

PERKINS COIE LLP
P.O. BOX 2168
MENLO PARK, CA 94026

10/05/2004 GWORDOF2 00000107 09225198

01 FC:1501 1330.00 OP
02 FC:8001 30.00 OP

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 Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (703) 746-4000, on the date indicated below.

Sharyl Brown (Depositor's name)
Sharyl Brown (Signature)
September 29, 2004 (Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/225,198	01/05/1999	ADAM J. CHEYER	SRI1P016	2756

TITLE OF INVENTION: SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1330	$0	$1330	12/10/2004

EXAMINER	ART UNIT	CLASS-SUBCLASS
BULLOCK JR, LEWIS ALEXANDER	2126	709-310000

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 Perkins Coie LLP
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. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. 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 (No small entity discount permitted)
- ☒ 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 Director is hereby authorized by charge the required fee(s), or credit any overpayment, to Deposit Account Number 50-2207 (enclose an extra copy of this form).

5. **Change in Entity Status** (from status indicated above)
- ☐ a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27.
- ☒ b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

The Director of the USPTO 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.
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.

Authorized Signature Carina M. Tan   Date September 29, 2004

Typed or printed name Carina M. Tan   Registration No. 45,769

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, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

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.

PTOL-85 (Rev. 09/04) Approved for use through 04/30/2007. OMB 0651-0033 U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

OCT 0 4 2004 OIPE JC22 PATENT & TRADEMARK OFFICE

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: Mail Stop Issue Fee, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450, on:

Date: September 29, 2004 By: Sharyl Brown

Sharyl Brown

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF:	EXAMINER: L. A. BULLOCK, JR.
CHEYER ET AL.	ART UNIT: 2126
APPLICATION NO.: 09/225,198	
FILED: January 5, 1999	NOTICE OF ALLOWABILITY: SEPTEMBER 10, 2004
FOR: **SOFTWARE-BASED ARCHITECTURE FOR COMMUNICATION AND COOPERATION AMONG DISTRIBUTED ELECTRONIC AGENTS**	

Transmittal of Issue Fee and Advance Order

Mail Stop Issue Fee
Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

Sir:

In response to the Notice of Allowance dated September 10, 2004, applicants herewith submit the following:

☒ Form PTOL-85B (in duplicate)

☒ Check in the amount of $1,360.00 for:

1) Issue Fee ($1,330.00) – Large Entity
2) Fee ($30) for 10 advance copies of the printed patent.

☒ Please charge any additional fees necessary for consideration of this paper to Deposit Account No. 50-2207.

Respectfully submitted,
Perkins Coie LLP

Date: September 29, 2004

Carina M. Tan
Carina M. Tan
Registration No. 45,769

Correspondence Address:
Customer No. 22918
Perkins Coie LLP
P. O. Box 2168
Menlo Park, CA 94026-2168
(650) 838-4300

2126

CERTIFICATE OF MAILING (37 CFR 1.8(a))

I hereby certify that this paper (along with any referred to as being attached or enclosed) is being deposited with the U.S. Postal Service as first class mail in an envelope addressed to: Mail Stop Issue Fee, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on:

Date: September 29, 2004

Sharyl Brown

By: Sharyl Brown

UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Cheyer et al.	Docket No.:	59501-8016.US01
Serial No.:	09/225,198	Group Art Unit:	2126
Filing Date:	January 5, 1999	Examiner:	L.A. Bullock, Jr.

For: Software-Based Architecture For Communication And Cooperation Among Distributed Electronic Agents

Mail Stop Issue Fee
Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

NOTIFICATION OF ERROR IN PAYMENT OF FEE(S) AS A SMALL ENTITY (37 C.F. § 1.28(c))

1. The present application is no longer entitled to small entity status. On November 18, 2002 and on March 29, 2004, Applicants filed Amendment and Response to Office Actions, each requesting a one month extension of time.

Error

2. The error in the payment of fee(s) as a small entity was as follows:

   ☒ Applicant believed itself entitled to small entity status, and has discovered that it is no longer be entitled to small entity status.

Fee Payment for Deficiency

3. ☒ Payment is attached for the deficiency between the amount of fees paid and the amount due.

Fee Payment

4. ☒ The attached check in the amount of $110.00 includes fees for the deficiency of the filing of the one month extension of time filed on November 18, 2002 and on March 29, 2004.

10/05/2004 HLE333 00000072 09225198

01 FC:1251 110.00 OP

☒ In the event that: a) no check to cover the filing fee is enclosed, b) any above-referenced check is inadvertently omitted or lost, or c) any enclosed check is in an amount less than or greater than the required fee, the Commissioner is authorized to charge any required fees, additional fees, or credit any overpayment to Deposit Account 50-2207.

Further Status as a Small Entity

☒ Status as a small entity is hereby withdrawn.

☒ Attached is a postcard for date-stamped return as confirmation of receipt of these materials.

Date: September 29, 2004

Carina M. Tan
Reg. No. 45,769

PERKINS COIE LLP
Customer No.: 22918
P.O. Box 2168
Menlo Park, CA 94026
Tel: (650) 838-4300
Fax: (650) 838-4350