QuickSet: Multimodal Interaction for Distributed Applications

Philip R. Cohen, Michael Johnston, David McGee, Sharon Oviatt,
Jay Pittman, Ira Smith, Liang Chen and Josh Clow
Center for Human Computer Communication
Oregon Graduate Institute of Science and Technology
P.O.Box 91000
Portland, OR 97291-1000 USA
Tel: 1-503-690-1326
E-mail: pcohen@cse.ogi.edu
http://www.cse.ogi.edu/CHCC

ABSTRACT

This paper presents an emerging application of multimodal interface research to distributed applications. We have developed the QuickSet prototype, a pen/voice system running on a hand-held PC, communicating via wireless LAN through an agent architecture to a number of systems, including NRaD's[1] LeatherNet system, a distributed interactive training simulator built for the US Marine Corps. The paper describes the overall system architecture, a novel multimodal integration strategy offering mutual compensation among modalities, and provides examples of multimodal simulation setup. Finally, we discuss our applications experience and evaluation.

KEYWORDS: multimodal interfaces, agent architecture, gesture recognition, speech recognition, natural language processing, distributed interactive simulation.

1. INTRODUCTION

A new generation of multimodal systems is emerging in which the user will be able to employ natural communication modalities, including voice, hand and pen-based gesture, eye-tracking, body-movement, etc. [Koons et al., 1993; Oviatt, 1992, 1996; Waibel et al., 1995] in addition to the usual graphical user interface technologies. In order to make progress on building such systems, a principled method of modality integration, and a general architecture to support it is needed. Such a framework should provide sufficient flexibility to enable rapid experimentation with different modality integration architectures and applications. This experimentation will allow researchers to discover how each communication modality can best contribute its strengths yet compensate for the weaknesses of the others.

Fortunately, a new generation of distributed system frameworks is now becoming standardized, including the CORBA and DCOM frameworks for distributed object systems. At a higher level, multiagent architectures are being developed that allow integration and interoperation of semi-autonomous knowledge-based components or "agents". The advantages of these architectural frameworks are modularity, distribution, and asynchrony — a subsystem can request that a certain functionality be provided without knowing who will provide it, where it resides, how to invoke it, or how long to wait for it. In virtue of these qualities, these frameworks provide a convenient platform for experimenting with new architectures and applications.

In this paper, we describe QuickSet, a collaborative, multimodal system that employs such a distributed, multiagent architecture to integrate not only the various user interface components, but also a collection of distributed applications. QuickSet provides a new *unification-based* mechanism for fusing partial meaning representation fragments derived from the input modalities. In so doing, it selects the best *joint* interpretation among the alternatives presented by the underlying spoken language and gestural modalities. Unification also supports multimodal discourse. The system is scaleable from handheld to wall-sized interfaces, and interoperates across a number of platforms (PC's to UNIX workstations). Finally, QuickSet has been applied to a collaborative military training system, in which it is used to control a simulator and a 3-D virtual terrain visualization system.

This paper describes the "look and feel" of the multimodal interaction with a variety of back-end applications, and discusses the unification-based architecture that makes this new class of interface possible. Finally, the paper discusses the application of the technology for the Department of Defense.

[1] NRaD = US Navy Command and Control Ocean Systems Center Research Development Test and Evaluation (San Diego).

2. QUICKSET

QuickSet is a collaborative, handheld, multimodal system for interacting with distributed applications. In virtue of its modular, agent-based design, QuickSet has been applied to a number of applications in a relatively short period of time, including:

- *Simulation Set-up and Control* — Quickset is used to control LeatherNet [Clarkson and Yi, 1996], a system employed in training platoon leaders and company commanders at the USMC base at Twentynine Palms, California. LeatherNet simulations are created using the ModSAF simulator [Courtmanche and Ceranowicz, 1995] and can be visualized in a wall-sized virtual reality CAVE environment [Cruz-Neira et al., 1993; Zyda et al., 1992] called CommandVu. A QuickSet user can create entities, give them missions, and control the virtual reality environment from the handheld PC. QuickSet communicates over a wireless LAN via the Open Agent Architecture (OAA) [Cohen et al., 1994] to ModSAF, and to CommandVu, each of which have been made into agents in the architecture.
- *Force Laydown* — QuickSet is being used in a second effort called ExInit (Exercise Initialization), that enables users to create large-scale (division- and brigade- sized) exercises. Here, QuickSet interoperates via the agent architecture with a collection of CORBA servers.
- *Medical Informatics* — A version of QuickSet is used in selecting healthcare in Portland, Oregon. In this application, QuickSet retrieves data from a database of 2000 records about doctors, specialties, and clinics.

Next, we turn to the primary application of QuickSet technology.

3. NEW INTERFACES FOR DISTRIBUTED SIMULATION

Begun as SIMNET in the 1980's [Thorpe, 1987], distributed, interactive simulation (DIS) training environments attempt to provide a high degree of fidelity in simulating combat equipment, movement, atmospheric effects, etc. One of the U.S. Government's goals, which has partially motivated the present research, is to develop technologies that can aid in substantially reducing the time and effort needed to create large-scale scenarios. A recently achieved milestone is the ability to create and simulate a large-scale exercise, in which there may be on the order of 60,000 entities (e.g., a vehicle or a person).

QuickSet addresses two phases of user interaction with these simulations: creating and positioning the entities, and supplying their initial behavior. In the first phase, a user "lays down" or places forces on the terrain, which need to be positioned in realistic ways, given the terrain, mission, available equipment, etc. In addition to force laydown the user needs to supply them with behavior, which may involve complex maneuvering, communication, etc.

Our contribution to this overall effort is to rethink the nature of the user interaction. As with most modern simulators, DISs are controlled via graphical user interfaces (GUIs). However, GUI-based interaction is rapidly losing its benefits, especially when large numbers of entities need to be created and controlled, often resulting in enormous menu trees. At the same time, for reasons of mobility and affordability, there is a strong user desire to be able to create simulations on small devices (e.g., PDA's). This impending collision of trends for smaller screen size and for more entities requires a different paradigm for human-computer interaction with simulators.

A major design goal for QuickSet is to provide the same user input capabilities for handheld, desktop, and wall-sized terminal hardware. We believe that only voice and gesture-based interaction comfortably span this range. QuickSet provides *both* of these modalities because it has been demonstrated that there exist substantive language, task performance, and user preference advantages for multimodal interaction over speech-only and gesture-only interaction with map-based tasks [Oviatt, 1996; Oviatt, in press].[2] Specifically, for these tasks, multimodal input results in 36% fewer task performance errors, 35% fewer spoken disfluencies, 10% faster task performance, and 23% fewer words, as compared to a speech-only interaction. Multimodal pen/voice interaction is known to be advantageous for small devices, for mobile users who may encounter different circumstances, for error avoidance and correction, and for robustness [Oviatt, 1992; Oviatt 1995].

In summary, a multimodal voice/gesture interface complements, but also promises to address the limitations of, current GUI technologies for controlling simulators. In addition, it has been shown to have numerous advantages over voice-only interaction for map-based tasks. These findings had a direct bearing on the interface design and architecture of QuickSet.

4. SYSTEM ARCHITECTURE

In order to build QuickSet, distributed agent technologies based on the Open Agent Architecture[3] were employed because of its flexible asynchronous capabilities, its ability to run the same set of software components in a variety of hardware configurations, ranging from standalone on the handheld PC to distributed operation across numerous computers, and its easy connection to legacy applications. Additionally, the architecture supports user mobility in that less computationally-intensive agents (e.g., the map interface) can run on the handheld PC, while more computationally-intensive processes (e.g., natural language processing) can operate elsewhere on the network. The agents may be written in any programming language (here, Quintus Prolog, Visual C++, Visual Basic, and Java), as long as they communicate via an interagent communication language. The configuration of agents used in the QuickSet system is illustrated in Figure 1. A brief description of each agent follows.

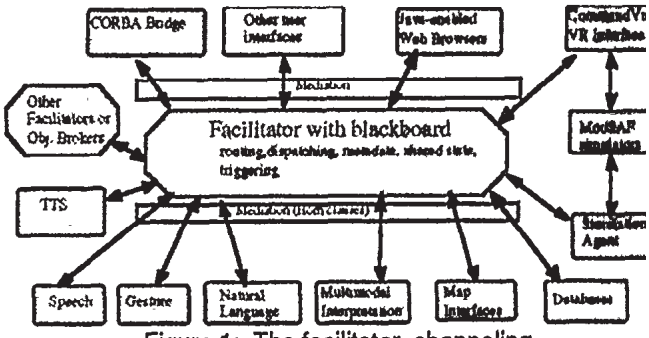

Figure 1: The facilitator, channeling queries to capable agents.

[2] Our prior research [Cohen et al., 1989; Cohen, 1992] has demonstrated the advantages of a multimodal interface offering natural language and direct manipulation for controlling simulators and reviewing their results.
[3] Open Agent Architecture is a trademark of SRI International.

5. EXAMPLES

5.1 Leathernet

Holding QuickSet, the user views a map from the ModSAF simulation. With speech and pen, she then adds entities into the ModSAF simulation. For example, to create a unit in QuickSet, the user would hold the pen at the desired location and utter: "red T72 platoon" resulting in a new platoon of the specified type being created. The user then adds a barbed-wire fence to the simulation by drawing a line at the desired location while uttering "barbed wire." A fortified line can be added multimodally, by drawing a simple line and speaking its label, or unimodally, by drawing its military symbology. A minefield of an amorphous shape is drawn and is labeled verbally. Finally an M1A1 platoon is created as above. Then the user can assign a task to the platoon by saying "M1A1 platoon follow this route" while drawing the route with the pen.

Figure 4: QuickSet running on a wireless handheld PC. The user has created numerous units, fortifications and objectives.

The results of these commands are visible on the QuickSet screen, as seen in Figure 4, as well as on the ModSAF simulation, which has been executing the user's QuickSet commands in the virtual world (Figure 5).

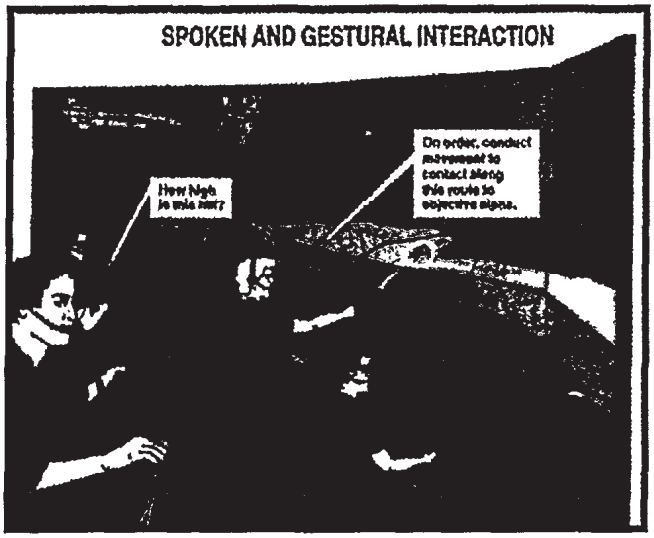

Figure 5: Controlling the CommandVu 3-D visualization via QuickSet interaction. QuickSet tablets are on the desks.

Two specific aspects of QuickSet to be discussed below are its usage as a collaborative system, and its ability to control a virtual reality environment.

5.1.1 Collaboration.

In virtue of the facilitated agent architecture, when two or more user interfaces connected to the same network of facilitators subscribe to and/or produce common messages, they (and their users) become part of a collaboration. The agent architecture offers a framework for heterogeneous collaboration, in that users can have very different interfaces, operating on different types of hardware platforms, and yet be part of a collaboration. For instance, by subscribing to the entity-location database messages, multiple QuickSet user interfaces can be notified of changes in the locations of entities, and can then render them in whatever form is suitable, including 2-D map-based, web-based, and 3-D virtual reality displays. Likewise, users can interact with different interfaces (e.g., placing entities on the 2-D map or 3-D VR) and thereby affect the views seen by other users. To allow for tighter synchronicity, the current implementation also allows users to decide to couple their interface to those of the other users connected to a given network of facilitators. Then, when one interface pans and zooms, the other coupled ones do as well. Furthermore, coupled interfaces subscribe to the "ink" messages, meaning one user's ink appears on the others' screens, immediately providing a shared drawing system. On the other hand, collaborative systems also require facilities to prevent users from interfering with one another. QuickSet incorporates authentication of messages in order that one user's speech is not accidentally integrated with another's gesture.

In the future, we will provide a subgrouping mechanism for users, such that there can be multiple collaborating groups using the same facilitator, thereby allowing users to be able to choose to join collaborations of specific subgroups. Also to be developed is a method for handling conflicting actions during a collaboration.

5.1.2 Multimodal Control of Virtual Travel

Most terrain visualization systems allow only for flight control, either through a joystick (or equivalent), via keyboard commands, or via mouse movement. Unfortunately, to make effective use of such interfaces, people need to be pilots, or at least know where they are going. Believing this to be unnecessarily restrictive, our virtual reality set-up follows the approach recommended by Baker and Wickens [unpublished ms]., Brooks [1996], and Stoakley et al., [1995] in offering two "linked" displays — a 2-D "birds-eye" map-based display (QuickSet), and the 3-D CommandVu visualization. In addition to the existing 3-D controls, the user can issue spoken or multimodal commands via the handheld PC to be executed by CommandVu. Sample commands are:

"CommandVu, heads up display on,"

"take me to objective alpha"

"fly me to this platoon <gesture on QuickSet map>" (see Figure 4).

"fly me along this route <draws route on QuickSet map> at fifty meters"

Spoken interaction with virtual worlds offers distinct advantages over direct manipulation, in that users are able to describe entities and locations that are not in view, can be teleported to those out-of-view locations and entities, and can

QuickSet interface: On the handheld PC is a geo-referenced map of some region,[4] such that entities displayed on the map are registered to their positions on the actual terrain, and thereby to their positions on each of the various user interfaces connected to the simulation. The map interface provides the usual pan and zoom capabilities, multiple overlays, icons, etc. Two levels of map are shown at once, with a small rectangle shown on a miniature version of the larger scale map indicating the portion of it shown on the main map interface.

Employing pen, speech, or more frequently, multimodal input, the user can annotate the map, creating points, lines, and areas of various types. The user can also create entities, give them behavior, and watch the simulation unfold from the handheld. When the pen is placed on the screen, the speech recognizer is activated, thereby allowing users to speak and gesture simultaneously. The interface offers controls for various parameters of speech recognition, for loading different maps, for entering into collaborations with other users, for connecting to different facilitators, and for discovering other agents who are connected to the facilitator. The QuickSet system also offers a novel map-labeling algorithm that attempts to minimize the overlap of map labels as the user creates more complex scenarios, and as the entities move (cf. [Christensen et al., 1996]).

Speech recognition agent: The speech recognition agent used in QuickSet is built on IBM's VoiceType Application Factory and VoiceType 3.0, recognizers, as well as Microsoft Whisper speech recognizer.

Gesture recognition agent: QuickSet's pen-based gesture recognizer consists of both a neural network [Pittman, 1991, Manke et al., 1994] and a set of hidden Markov models. The digital ink is size-normalized, centered in a 2D image, and fed into the neural network as pixels. The ink is also smoothed, resampled, converted to deltas, and given as input to the HMM recognizer. The system currently recognizes 68 pen-gestures, including various military map symbols (platoon, mortar, fortified line, etc.), editing gestures (deletion, grouping), route indications, area indications, taps, etc. The probability estimates from the two recognizers are combined to yield probabilities for each of the possible interpretations. The inclusion of route and area indications creates a special problem for the recognizers, since route and area indications may have a variety of shapes. This problem is further compounded by the fact that the recognizer needs to be robust in the face of sloppy writing. More typically, sloppy forms of various map symbols, such as those illustrated in Figure 3, will often take the same shape as some route and area indications. A solution for this problem can be found by combining the outputs from the gesture recognizer with the outputs from the speech recognizer, as is described in the following section.

Figure 2: Pen drawings of routes and areas. Routes and areas do not have signature shapes that can be used to identify them.

[4] QuickSet can employ either UTM or Latitude/Longitude coordinate systems.

Figure 3: Typical pen input from real users. The recognizer must be robust in the face of sloppy input.

Natural language agent: The natural language agent currently employs a definite clause grammar and produces typed feature structures as a representation of the utterance meaning. Currently, for the force laydown and mission assignment tasks, the language consists of noun phrases that label entities, as well as a variety of imperative constructs for supplying behavior.

Text-to-Speech agent: Microsoft's text-to-speech system has been incorporated as an agent, residing on each individual PC.

Multimodal integration agent: The task of the integrator agent is to field incoming typed feature structures representing individual interpretations of speech and of gesture, and identify the best potential unified interpretation, multimodal or unimodal. In order for speech and gesture to be incorporated into a multimodal interpretation, they need to be both semantically and temporally compatible. The output of this agent is a typed feature structure representing the preferred interpretation, which is ultimately routed to the bridge agent for execution. A more detailed description of multimodal interpretation is in Section 6.

Simulation agent: The simulation agent, developed primarily by SRI International [Moore et al., 1997], but modified by us for multimodal interaction, serves as the communication channel between the OAA-brokered agents and the ModSAF simulation system. This agent offers an API for ModSAF that other agents can use.

Web display agent: The Web display agent can be used to create entities, points, lines, and areas, and posts queries for updates to the state of the simulation via Java code that interacts with the blackboard and facilitator. The queries are routed to the running ModSAF simulation, and the available entities can be viewed over a WWW connection.

CommandVu agent: Since the CommandVu virtual reality system is an agent, the same multimodal interface on the handheld PC can be used to create entities and to fly the user through the 3-D terrain.

Application bridge agent: The bridge agent generalizes the underlying applications' API to typed feature structures, thereby providing an interface to the various applications such as ModSAF, CommandVu, and Exinit. This allows for a domain-independent integration architecture in which constraints on multimodal interpretation are stated in terms of higher-level constructs such as typed feature structures, greatly facilitating reuse.

CORBA bridge agent: This agent converts OAA messages to CORBA IDL (Interface Definition Language) for the Exercise Initialization project.

To see how QuickSet is used, we present the following examples.

ask questions about entities in the scene. We are currently engaged in research to allow the user to gesture directly into the 3-D scene while speaking, a capability that will make these more sophisticated interactions possible.

5.2 Exercise Initialization: ExInit

QuickSet has been incorporated into the DoD's new Exercise Initialization tool, whose job is to create the force laydown and initial mission assignments for very large-scale simulated scenarios. Whereas previous manual methods for initializing scenarios resulted in a large number of people spending more than a year in order to create a division-sized scenario, a 60,000+ entity scenario recently took a single ExInit user 63 hours, most of which was computation.

ExInit is distinctive in its use of CORBA technologies as the interoperation framework, and its use of inexpensive off-the-shelf personal computers. ExInit's CORBA servers (written or integrated by MRJ Corp. and Ascent Technologies) include a relational database (Microsoft Access or Oracle), a geographical information system (CARIS), a "deployment" server that knows how to decompose a high-level unit into smaller ones and position them in realistic ways with respect to the terrain, a graphical user interface, and QuickSet for voice/gesture interaction.

In order for the QuickSet interface to work as part of the larger ExInit system, a CORBA bridge agent was written for the OAA, which communicated via IDL to the CORBA side, and via the interagent communication language to the OAA agents. Thus, to the CORBA servers, QuickSet is viewed as a Voice/Gesture server, whereas to the QuickSet agents, ExInit is simply another application agent. Users can interact with the QuickSet map interface (which offers a fluid multimodal interface), and view ExInit as a "back-end" application similar to ModSAF. A diagram of the QuickSet-ExInit architecture can be found in Figure 6. Shown there as well is a connection to DARPA's Advanced Logistics Program demonstration system for which QuickSet is the user interface.

To illustrate the use of QuickSet for ExInit, consider the example of Figure 7, in which, a user has said: "Multiple boundaries," followed in rapid succession by a series of multimodal utterances such as "Battalion <draws line>," "Company <draws line>," etc. The first utterance tells ExInit that subsequent input is to be interpreted as a boundary line, if possible. When the user then names an echelon and draws a line, the multimodal input is interpreted as a boundary of the appropriate echelon.

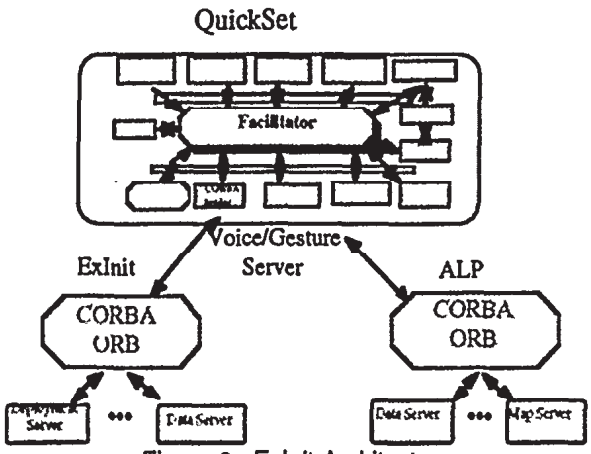

Figure 6: ExInit Architecture

Numerous features describing engineering works, such as a fortified line, a berm, minefields, etc. have also been added to the map using speech and gesture. Then the user creates a number of armored companies facing 45 degrees in defensive posture; he is now beginning to add armored companies facing 225 degrees, etc. Once the user is finished positioning the entities, he can ask for them to be deployed to a lower-level (e.g., platoon).

An informal user test was recently run in which an experienced ExInit user (who had created the 60,000 entity scenario) designed his own test scenario involving the creation of 8 units and 15 control measures (e.g., the lines and areas shown in Figure 7). The user first entered the scenario via the ExInit graphical user interface, a standard Microsoft Windows mouse-menu-based GUI. Then, after a relatively short training session with QuickSet, he created the same scenario using speech and gesture. Interaction via QuickSet resulted in a two-fold to seven-fold speedup, depending on the size of the units involved (companies or battalions). Although a more comprehensive user test remains to be conducted, this early data point indicates the productivity gains that can potentially be derived from using multimodal interaction.

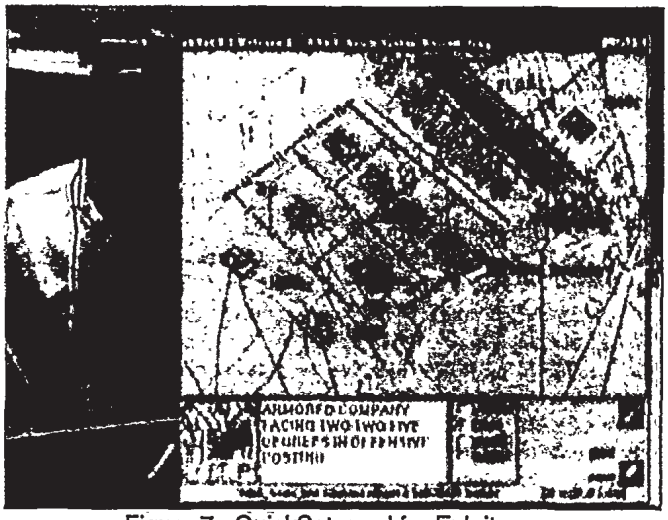
Figure 7: QuickSet used for ExInit — large-scale exercise initialization

5.3 Multimodal Interaction with Medical Information: MIMI

The last example a QuickSet-based application is MIMI, which allows users to find appropriate health care in Portland, Oregon. Working with the Oregon Health Sciences University, a prototype was developed that allows users to inquire using speech and gesture about available health care providers. For example, a user might say "show me all psychiatrists in this neighborhood <circling gesture on map>". The system translates the multimodal input into a query to a database of doctor records. The query results in a series of icons being displayed on the map. Each of these icons contains one or more health care providers meeting the appropriate criterion. Figure 8 show the map-based interaction supported by MIMI.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.