Integrating a Spoken Language System with Agents for Operational Information Access

Jody Daniels

Lockheed Martin Advanced Technology Laboratories
1 Federal Street
Camden, NJ 08102 USA
jdaniels@atl.lmco.com

Abstract

Changing the way users interact with their data is the principal objective of the Listen, Communicate, Show (LCS) paradigm. LCS is a new paradigm being applied to Marine Corps tactical logistics. Using a spoken language understanding system a Marine converses with the system to place a supply or information request, which is passed to a mobile, intelligent agent to execute at the proper database. Upon successful return, the agent notifies the spoken language system of its results, and the Marine is given a verbal and/or visual response.

Introduction

Marines work in a dynamic, fluid environment where requirements and priorities are constantly subject to change. It currently takes 72 hours before a Marine in a Combat Service Support Operations Center (CSSOC) can confirm with a requesting unit that their order is in the logistics system. This is unacceptable for most requests, particularly if the request is for mission-critical supplies. CSSOC personnel spend a great deal of their time placing and then tracking requests, trying to keep the requesting units appraised of the status of their supplies. When it takes so long to place and check on a request, all involved become frustrated and operational plans may have to be altered to accommodate the slowness of the supply system.

The focus of the LCS - Marine project is to provide Marines in the field with the logistical support that they need, when they need it, where they need it. In an LCS system, the computer *listens* for information requests, *communicates* both with the user and networked information resources to compute user-centered solutions, and *shows* tailored visualizations to individual warfighters.

This is done by integrating a spoken language understanding system (SLS) (for assisting the user in placing a request) with mobile, intelligent agents (for information access). The SLS converses with the user to gather information for placing a new request, or to check status, amend, or cancel an existing request. Once sufficient information is obtained from the user, the SLS launches an agent to accomplish the requested task. The agent accesses a snapshot of the Marine databases by traveling over existing tactical communications networks. Once the agent's itinerary is complete, it returns to the SLS, which generates an appropriate response to the user. This may be visual in addition to verbal, depending on the available media. By integrating these AI technologies, we hope to reduce the 72-hour response time to less than 7.2 minutes.

Through a quick conversation between the user and the LCS system, many potential pitfalls can be reduced or eliminated in the processing of a request. For example, the system can observe that the user has requested an amount that exceeds the capacity of the requesting unit and can attempt to resolve this problem directly with the user. Additionally, the user can establish monitor agents that will track the request and send notification agents back to the user to report either status updates or observations that the request isn't being given the attention the user needs.

Achieving near real-time access to logistics databases via spoken language and agents greatly increases the confidence that the proper supplies will arrive and in a more timely fashion. Thus, commanders are able to more accurately generate operations plans that are less likely to be subject to supply shortfalls.

We next describe the LCS - Marine task. Then we discuss the system and its components. Finally we give an overview of the operational testing and results of using the LCS - Marine system.

Task Description

To submit requests through the logistics system, units pass their requests to the CSSOC, which prepares a Rapid Request form. LCS - Marine supports both ordinance (Class V) and subsistence (Class I) supply requests through the use of the Rapid Request form. LCS – Marine users provide their input to the CSSOC via radio and use radio protocols when communicating. This requires the system to understand call signs, military times, etc.

To support the flow of information using the Rapid Request form, the initial set of LCS - Marine activities are to submit a request, check on its status, change a request, and cancel a request. Figure 1 shows a partially completed Rapid Request form when placing a new request.

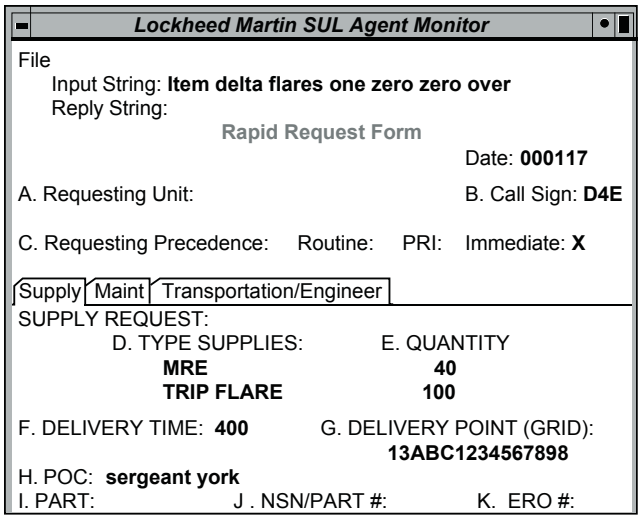

Figure 1. Partially Completed Marine Corps Rapid Request Form

Overview of the LCS – Marine System

The LCS - Marine system consists of four major components: an SLS, a collection of agents for information access, real-world operational databases, and communications networks to connect the user to the SLS and the agents to the databases. Operationally, the Marine speaks into a microphone attached to a hand-held device. The recognition process begins on the hand-held and sends a compressed version of the speech over a wireless communications link to the main portion of the SLS, which resides a short distance away.

The other SLS components complete the recognition process, parse the *n*-best possibilities, create a semantic representation, add historical context from the on-going human-machine interchange, and decide what step to take next. Possible next steps include prompting the user for additional information, requesting a clarification, or sending an agent to an information source to process a completed user request.

When the dialogue is sufficiently progressed, the SLS will send a mobile agent over another communications link (either SINCGARS or Wavelan) to the remote database management system (dbms). After processing the request at the dbms, the agent returns to the SLS to confirm placing the request (or the modification or deletion). The SLS then generates a response for the user, which may be both verbal and visual. The SLS transforms the verbal response from a semantic frame into text, which is then synthesized into speech on the hand-held device. Visual updates are handled by agents, which manage a custom display server. Figure 2 provides an overview of the current LCS – Marine architecture.

We next describe each of the components of the system, starting with the SLS, then the extendible mobile agent architecture system, the dbms, and finally the communications system.

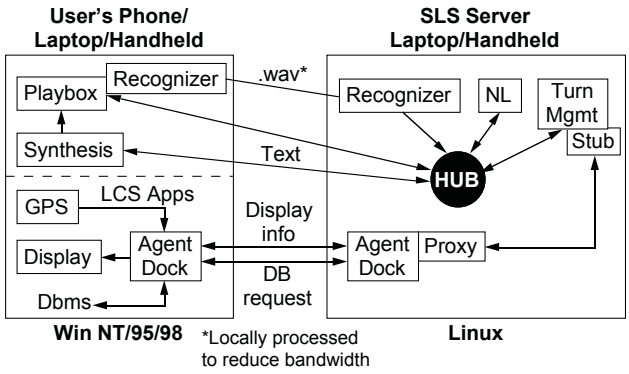

Figure 2. LCS - Marine Architecture

Spoken Language Understanding System (SLS)

The SLS uses the MIT Galaxy II architecture customized with the domain application and specialized servers (Seneff, Lau, and Polifroni 1999). The architecture is designed to be "plug and play."

Galaxy II is a DARPA-developed, GOTS, distributed, component-based middleware product maintained by the MITRE Corporation. Specialized servers handle specific tasks, such as translating audio data to text. All Galaxy II-compliant servers communicate with each other through a central server known as the Hub. The Hub manages flow control, handles traffic among distributed servers, and provides state maintenance.

Using the Hub architecture, information is passed in the form of a frame. Specific frames are built throughout the processing to handle and store such items as the input utterance, prior context, and the response to the user.

In terms of overall processing, speech is moved from the Playbox to the Recognizer. Speech is translated, prior context added, and processed using the Natural Language (NL) and Turn Manager servers to verify the new input's validity. The Turn Manager generates a response, NL converts it to text, and the Synthesis server generates the verbal response. The Playbox then speaks the waveform file to the user.

Audio. The Audio Input/Output server, known as the Playbox, uses the COTS Microsoft SAPI Speech Development Kit (Beta). Using SAPI audio objects enables LCS - Marine to capture and play back speech. The server captures speech and sends it as an audio waveform file to the Recognizer.

To support low-bandwidth communications, such as tactical and cellular, the initial steps of the Recognizer (spectral compression) were added to the audio capture modules to reduce the amount of bandwidth needed to move the waveform file.

At the other end of a processing cycle (sometimes called a turn) the Synthesis server converts text to speech, creating an audio waveform file. Synthesis also uses Microsoft SAPI audio objects. This waveform file is then passed directly to the Playbox for audio generation, completing the user-machine dialogue loop.

Recognizer. The Recognizer is the server responsible for speaker-independent speech recognition. Audio waveform data is analyzed for component sounds and examined with respect to models of known combinations of sounds. Candidate results are selected using an *n*-best algorithm.

The *n*-best algorithm produces a list of the *n* most likely possible utterances produced that map to the waveforms being examined. The list is merged into a single network, a word graph. Three conceptual models are used in this process. The acoustic model maps sounds to phones, the base unit of pronunciation. The pronunciation model maps groups of phones to words, and the language model groups words into sequences.

The LCS - Marine acoustic model is derived from data collected by the Spoken Language Systems Group at MIT for another speech application (Glass and Hazen 1998). The data is from narrow-band telephone speech and contains 24,000 samples. LCS - Marine has adapted and expanded upon this model as the vocabulary and the operational environment evolves.

The Recognizer combines the pronunciation and language models into one process, a class bigram model. This combined model is constructed and trained using a pronunciation lexicon of vocabulary words, a collection of word classifications, and a training set of common sentences and phrases.

Classification rules assist the language model by identifying groups of words that could appear in similar locations within an utterance. These could be “numbers,” “city names,” “colors” or any other group of related terms.

A large set of textual sentences, phrases, and other types of user inputs exist to train the class bigram model. These range in length from “Third platoon needs four thousand m sixteen rounds at checkpoint bravo before sixteen hundred hours zulu” to “no.” The text samples provide order information to the Recognizer and are used to generate the relative probabilities found in the language model.

The Recognizer merges the *n*-best sequences into a single directed word graph. The nodes in the graph correspond to particular pinpoints in time. The edges connecting the nodes are labeled with a word and score. NL will use these scores when matching against the parse rules.

Natural Language (NL). The NL component handles text parsing and generation. A semantic frame is built to represent knowledge and the user's meaning. This knowledge representation is initially built by NL and is passed to the other components. NL parses text, both syntactically and semantically, to create a semantic representation (the semantic frame) to pass to the Turn Manager. NL also takes semantic frames representing replies from the system back to the user and transforms the frame into text to be synthesized into speech.

NL first syntactically parses the possible sequences in the word graph provided by Recognizer. NL uses a grammar specific to the application domain. The most likely parsable word sequence from the word graph is converted into a syntax tree for further manipulation.

A second domain-specific grammar maps the resulting syntax tree into a semantic tree. The semantic tree is an intermediate step in which meanings are mapped onto salient parts of the syntax tree, ready to be rearranged into a semantic frame.

Using another set of rules, NL constructs a semantic frame from the semantic tree. While both the semantic tree and the semantic frame represent the meaning of an utterance, the structure of the semantic tree reflects the form of the utterance, while the semantic frame reflects the form of the ideas expressed within that utterance.

Information from previously constructed frames (which represent the history of the discourse) may be used to augment the current frame through a collection of rules. The final semantic frame incorporates this historical context and represents the current semantic state of the dialogue. It is passed to the Turn Manager.

When the Turn Manager has completed its processing and has created a reply to speak to the user, NL again takes action. NL translates turn management output, also a semantic frame, into natural language text. This is accomplished by selecting the appropriate text templates from a catalog of templates and filling in the blanks with information from fields within the semantic frame. The text generated by NL is then passed to the Synthesis server.

Turn Manager. The Turn Manager manages dialogue with the user and interfaces with the agent system. The logic contained in the Turn Manager governs the order of clarifying questions from the system back to the user and triggers agent creation and tasking. This turn management stage reflects the system's end of the dialogue with the user.

The Turn Manager examines the current semantic state of the dialogue, represented by the semantic frame created by NL. The Turn Manager checks the information contained in the semantic frame to determine which data elements of a request are present, which are missing, and whether the information is consistent with domain knowledge. If the semantic frame contains insufficient or inconsistent information, a request for clarifying information is generated and sent back to the NL server in the form of a semantic frame.

One of the Turn Manager's greatest strengths is its support of mixed-initiative dialogue. That is, while the Turn Manager requests information in a predictable order, it is able to accept information from the user that it has not yet requested. For example, if the system asks the user, “Where should the shipment be delivered?” and the user replies “Deliver by noon to Checkpoint Charlie,” the Turn Manager recognizes that both a location and a time have been given. It will, therefore, not ask for a time, which might otherwise have been requested later.

When the request by the user is sufficiently complete, the second responsibility of the Turn Manager comes into play. The Turn Manager distributes tasks to agents to gather and/or disseminate information to and from data sources. For example, this might mean making an SQL query to a database. When an agent returns information, the Turn Manager accepts it and translates it into a semantic frame. This semantic frame is passed back to the NL server and will be used to generate a response.

Extendible Mobile Agent Architecture System

LCS - Marine's information discovery and dissemination is accomplished using the Extendible Mobile Agent Architecture (EMAA) (Lentini, et. al. 1998) (Hofmann, McGovern, and Whitebread 1998). This agent architecture is used to integrate the LCS - Marine system with logistical databases, whether Lotus Notes, Oracle, or another dbms.

The agent architecture is a framework on which to build autonomous, intelligent, mobile agent systems. It defines an object-oriented interface for programming agent systems. It follows the principles of component-oriented and reusable object-oriented software. The agent architecture provides a mobile agent platform that gives an application the ability to dynamically create asynchronous mobile software agents. These agents can migrate among computing nodes in a network and can exploit resources at those nodes.

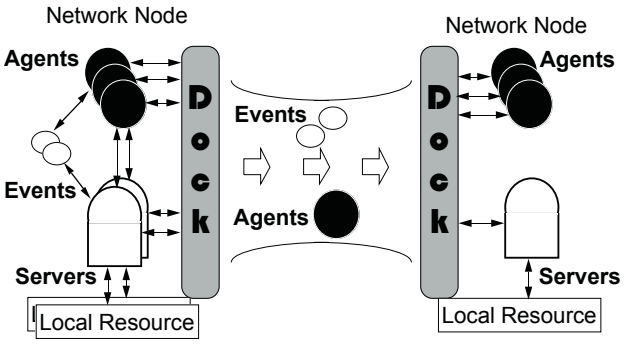

Figure 3. Extendible Mobile Agent Architecture

The agent architecture has three major components: the agents, servers, and the dock. At the most basic level, the agents perform the specialized, user-defined work (i.e., tasks). Agents travel through the system of computing nodes via the docks. The dock serves as a daemon to send and receive agents between docks at other nodes. Nodes that offer specialized services to agents do so via servers that provide packaged components. The relationship of the agents, servers, and docks is shown in Figure 3.

Agents. Mobile agents carry and execute tasks at different nodes. The goal of the agent is to complete all of its tasks. Since the agent is mobile, one must consider both the task and where that task is performed when creating the agent itinerary. EMAA supports mapping tasks to nodes; the set of these task/node pairs is defined as the itinerary of the agent.

EMAA Agent itineraries are Finite State Machines. Agents constructed using this model can exhibit complex behavior while maintaining the ability to execute itinerary subgoals.

Mobile agents can be tasked with database actions (i.e., query, submit, monitor, modify). They migrate to the dbms machine and execute the database actions and return with the results. The mobile agents gain access to the database using the Java Database Connection Standard (JDBC). A server component is run at the node that provides the functionality to make the JDBC connection. The agent accesses the server when it arrives at the node to obtain the functionality to establish the database connection. Once a connection is established, interaction to the database is done using standard SQL commands.

Servers. The agents rely on servers at each node to provide the underlying mechanism to interface with resources; they do not carry programs, merely well defined tasks. It is beneficial to package the code needed to access resources into separate components known as servers. Servers provide a repository of common functionality and processes that an agent may access at a given node. The agent architecture provides the following features with its resource servers. First, they are code repositories: migrating agents carry as little code as possible, thus conserving bandwidth. Second, the servers provide common APIs for the code or logic needed to exploit the resources at a node that keeps the agent machine-independent. Third, because of the component design of servers, the implementation of node-specific resources is separate from the implementation of the agent application.

Dock. The dock is the operating environment for agents and servers. The dock consists of four components: communication server, agent manager, server manager, and event manager.

A communications server is the daemon process itself and it handles all connections to any other computing nodes. It manages the transmission of agents to and from the local node.

An agent manager registers the agent and initializes it for execution. It is used in agent collaboration and agent validation for security reasons.

A server manager handles the server components at a node and provides a control mechanism to start and stop a service.

An event manager "listens" for component events. An event is an announcement that some component in the system has reached an important state, which may be of interest to other components. Often it is unknown if and when a component will generate an event. For this reason, agents and servers dependent on events must have an event manager "listen" for an event.

How Agents Are Used in LCS - Marine. The LCS - Marine system uses the agent architecture to interact with the Marines' logistical dbms. Mobile agents make ordinance (Class V) and subsistence (Class I) requests such as for water, 5.56 rounds, or flares. When making a request, agents are created with request criteria, (e.g., the amount of a particular type of supply to order, delivery location, etc.). If necessary, an agent will first make sure that the request criteria satisfy all dynamic request constraints. Values for these constraints are stored external to the agent system and need to be accessed for validity checking.

When a unit places a request for the water, the agent retrieves the allowable limit from the unit information and checks to see if the requested amount is allowable. If so,

then the agent migrates to the logistical database, connects with the database, and executes the request.

Logistical Database (dbms)

LCS - Marine's focus has been on simplifying the logistics process by gathering and disseminating information in a more effective and timely manner. If data can be inserted into and obtained from a database quickly and efficiently, it can effectively provide the information needed to make knowledgeable decisions for appropriate actions.

LCS - Marine uses an Oracle-based logistical database provided by the 1st Marine Expeditionary Force (MEF) for research, development, testing, and prototyping potential spoken user interactions with logistical data sources. The database supports logging, processing, and monitoring logistics requests. It incorporates a number of logistical systems including the Rapid Request Tracking System, Command Information System, and various administrative tools. The front end to these systems is a web-based interface using Cold Fusion to dynamically update the web pages. These systems support both the tactical community ordering supplies through the logistical database, and the watch officers in the rear monitoring the incoming requests and outgoing shipments of supplies.

By using a subset of the 1st MEF's database, which contains “real” logistic data gathered during one of their training exercises (data is spread across 121 tables), LCS - Marine can approximate the interaction that users have with the system.

Communications System

To benefit Marines in the field, the LCS system has been made more portable. The system's basic architecture was reduced to lightweight equipment (PC laptops) without a loss in functionality. Hand-held computing devices have been added and the microphone replaced with an ear/microphone headset. Increased mobility and accessibility were added via wireless access.

Testing

To measure the effectiveness of the LCS paradigm under operational conditions—real users placing real requests, accessing a live dbms, and using existing communications links—we slated four Integrated Feasibility Demonstrations (IFDs) over a twelve-month period. The first was held in September of 1999, the second in December, and the third is projected for April 2000, with the final one to be in July 2000. The IFDs range from scripted dialogue, replicated databases, and testing in the lab with prior military personnel, to active duty Marines using the system operationally over a series of days as their sole means of interaction with the logistics system for rapid requests.

The first IFD used five former military personnel as test subjects, although none were a Marine and none had used the Rapid Request form. All were familiar with radio operations and the military alphabet. All were native English speakers. Four were male and one was female. Each person attempted to accomplish three tasks: input a simple request, check on the status of the request, and make a modification to an existing request.

The second IFD took place concurrent with the Marine exercise Desert Knight. In this IFD, ten active-duty Marines attempted five tasks: the prior three plus a more complex request and canceling of an order. Nine of the subjects had used radio communications before, but only one had used the Rapid Request form before.

In a strenuous test of the speaker independence portion of the system, the users represented a diverse set of demographics: three of the subjects came from the Western US, two from the Southern US, one from the Mid-Atlantic Coast, one from the Midwestern US, one from Algeria, one from Mexico, and one from Burma. Seven subjects were male; three were female.

The third IFD is to take place in April 2000 concurrent with a Marine combined arms exercise and the fourth IFD will follow in July. The third IFD will allow greater flexibility in the dialogue and will integrate the LCS - Marine system into the existing logistics system: agents will be accessing the live dbms, rather than a replica. For the fourth IFD the system will be operational for a longer period of time and from a variety of locations.

At the conclusion of the IFDs we will review progress and evaluate the system’s robustness. Depending on the outcome, LCS - Marine could be either directly integrated into CSSOC operations or development could continue with further testing in preparation for a future integration.

Results

There are a variety of metrics that can be used to measure an SLS. Among these are measures for task completion, accuracy, and user satisfaction. For the IFDs we prepared and revised a user satisfaction survey and instrumented the system to collect a variety of pieces of data. We also made manual annotations within the conversation log files to collect additional metrics.

We reported task complexity, task completion rate, system understanding, and user satisfaction across the set of three or five tasks. We also attempted to pinpoint errors down to a specific component(s) of the system, although we do not discuss specific results here.

We computed task complexity as the number of information items that the system needed to understand before it could task an agent. For example, stating: “Item delta M R E three hundred,” represents three information items, specifying the line in the form, the type of supply, and the quantity. Table 1 gives a breakdown by task.

In IFD-1 an overall task completion rate for all tasks was 60%, which improved to 89% in IFD-2 for these same three tasks. For the set of five tasks in IFD-2, the average task completion rate was 82%. (The data from one subject was not included in the final results because of the difficulty of other humans in understanding this person.)

Task	Items	Turns	Time (Min)	Completion Rate
Simple request	11	21.0	7.0	78%
Complex request	22	27.0	16.6	44%
Status check	4	6.0	2.8	100%
Change request	6	11.5	4.1	89%
Cancel request	4	8.0	3.3	100%

Table 1. IFD-2 results.

For system understanding, we measured the number of "turns" to complete each task. We considered a turn to be a complete cycle, including both a user input and a system response.

For both completed IFDs, we measured user satisfaction after the completion of the entire test session. In the future, we may measure user satisfaction after the testing of each task, rather than at the end of the session. Because there is currently no standard for measuring user satisfaction of dialogue systems, we are still adapting our survey to attempt to glean the most information. Unfortunately, in retrospect, the changes we made to the survey between IFD-1 and IFD-2 were likely to have hurt our results. (For example, there were several questions where a negative response would have been the best response, and, if these questions were not closely read, the user could easily misinterpret the question.) For IFD-1, overall user satisfaction was at 1.85 on a 1 to 5 scale where 1 was best. For IFD-2, overall user satisfaction was at 6.4 on a 1 to 10 scale where 10 was best.

It is important to note that the amount of time spent training personnel to use the LCS - Marine system is generally less than 10 minutes. After a short introduction, the user is shown a sample dialogue for familiarization. The user is also given information about meta-instructions, such as how to start over or clear their previous statement, before they begin the tasks.

Conclusion

We have built a system that integrates an SLS with a mobile, intelligent agent system that allows users to place and access data requests via a conversational interface. The system is speaker independent and requires little training. The time to accomplish a task is significantly lower than the old, manual input method, but can still be improved. Being able to rapidly access, insert, modify, and delete data gives the users greater confidence in the supply system and allows commanders to generate operations plans that are less likely to be subject to supply shortfalls.

Acknowledgements

Thanks to members of the LCS - Marine team: Josh Brody, Phil Crawford, Jerry Franke, Michael Frew, Steve Knott, Lizz McCormick, Jose Rivera, and Kathy Stibler.

This research was supported by DARPA contract N66001-98-D-8507 and Naval contract N47406-99-C-7033.

References

Glass, J., and Hazen, T.J. 1998. Telephone-Based Conversational Speech Recognition in the Jupiter Domain. In *Proceedings of ICSLP '98*. Sydney, Australia.

Hofmann, M.O., McGovern, A., and Whitebread, K.R. 1998. Mobile Agents on the Digital Battlefield, in *Proceedings of the 2nd International Conference on Autonomous Agents (Agents '98)*. Minneapolis/St. Paul.

Lentini, R., Rao, G., Thies, J., and Kay, J. 1998. EMAA: An Extendable Mobile Agent Architecture. In *Proceedings for the Software Tools for Developing Agents Workshop at the 15th National Conference on Artificial Intelligence (AAAI '98)*. Madison, Wisconsin.

Seneff, S., Lau, R., and Polifroni, J. 1999. Organization, Communication, and Control in the GALAXY-II Conversational System. In *Proceedings for Eurospeech '98*. Budapest, Hungary.