Intelligent Agent-based User Interfaces

Douglas B. Moran, Adam J. Cheyer
Artificial Intelligence Center, SRI International
333 Ravenswood Avenue, Menlo Park, CA 94025 USA
E-mail: {moran,cheyer}@ai.sri.com, URL: http://www.ai.sri.com/~{moran,cheyer}/

Abstract

We have developed, and are extending, a multimodal human-computer interface in which the selection of modalities to be used takes into account the available resources and the user's current environment and preferences. Our current work focuses primarily on interpreting multimodal inputs and adapting to available output modalities. We anticipate providing more intelligent production of multimodal output in future systems.

We have an agent-based system architecture that we use to allow us to incrementally add functionality, and to substitute alternative handling of individual modalities.

Our Background and Focus

Our work in multimodal interfaces grew out of two interrelated research projects. The first project was a multimedia conferencing system [6] where the users could query a database to create and update graphical displays (primarily maps). Since the primary modalities of the conference were speech and graphical pointing (mouse cursor), spoken natural language was the most natural mechanism for accessing the database.

The second project focused on a practical role for natural language in accessing databases [3, 5]. Graphical interfaces using direct manipulation were advertised as quick and natural, and we observed that they often were thus for smaller problems, but on larger problems, they became slow, difficult and unnatural. Natural language access had complementary characteristics. However, within a query, the choice between direct manipulation and natural language was often not a choice of one or the other, but a combination of both. We built a series of prototypes allowing the user to choose the appropriate modality for each portion of a query. These prototypes were the basis for a series of experiments by colleagues [9, 10] on when users chose which modality.

In discussing these systems with potential users, we were impressed by the variability of computing environments within user communities. For example, a manufacturing engineer would be working with managers, design engineers, and factory floor personnel, and each of these groups would have different hardware and software environments. These users wanted to be able to use their computational tools not only from their own desktops, but from the computers of all the other people with whom they worked. Also, more and more people have multiple computers—for example, a workstation in the office, a home computer, and a PDA (Personal Digital Assistant) or portable computer for meetings and travel—and these computers have overlapping uses.

Since we wanted our multimodal interface to run on a wide range of hardware and software configurations, we needed an architecture that would support different subsets of the possible modalities and different implementations of individual modalities. Furthermore, restrictions on which of the available modalities are appropriate can be imposed by the environment in which the user is working. For example, even though a system is capable of spoken input and output, that modality would be unsuitable in a noisy environment, or when the user does not want his interactions with the computer to be overheard or to intrude on the conversation of nearby people.

Some of the example sets of modalities that we are particularly interested in are

- A traditional graphical user interface using speech instead of a keyboard, to be used by people whose hands are in use (*e.g.*, factory workers) or those who cannot use, or prefer to avoid, the keyboard (*e.g.*, people with injuries to their hands or wrists)
- A PDA or portable computer with a small screen that supports handwriting
- A speech-only interface (*e.g.*, a telephone)

Open Agent Architecture

Our multimodal interfaces are part of a larger agent-based architecture, called the *Open Agent Architecture* [1, 4], which is based loosely on Schwartz's FLiPSiDE system [12], and has been influenced by work being done as part of ARPA's I3 (Intelligent Integra-

tion of Information) program[1] and Knowledge Sharing Effort[2]. In this architecture, agents register their capabilities[3] with a special agent, called the *Facilitator*. Agents are independent processes and may be distributed over multiple computers, including ones from different manufacturers.

One system that we built with this architecture was an *Office Assistant*, which converted conventional applications such as a mail, calendar, and database lookup into agents. Our agent-based system had two major advantages over a simple collection of these applications. First, it allowed us to add handwriting and speech (and potentially other modalities) as the input to applications that assumed keyboard input. Second, it allowed us to combine the actions of multiple applications; thus, the command *"Send mail to Smith's boss"* does a database lookup to find Smith's boss, and then sends his e-mail address to the mail agent (the user follows up by filling in the body of the message). In this system, the major agents were readily identifiable because each has its own distinctive graphical interface.

In a second type of system we have built, a community of agents shares a common graphical interface: in all our examples, the graphical display is a map. From the typical user's perspective, interactions with such a system are indistinguishable from interacting with a large unitary application.

Our research on this architecture had two primary motivations. While we were interested in multiagent systems for their own sake, we also see them as a way to overcome the problems we have traditionally had in integrating closely related technology developed on separate projects. The systems we have built using this architecture have been driven in large part by the component technologies that were available to us and that we were most interested in exploiting.

Multimodal Input

Our treatment of multimodal input can be thought of as a hierarchy in which increasing amounts of intelligence are applied to the interpretation of the input. However, the actual organization of the agents is flat, and thus it is easy for different applications to make use of input modalities at differing levels of interpretation. However, our focus has been on applications that require high-level interpretation of multimodal inputs.

In our current multimodal interfaces, the user has audio and pen-based inputs in additional to the conventional computer interface. The audio input is used only for spoken natural language. The pen input is used for handwritten input, gestures, and conventional pointing (an alternative to a mouse).

We have agents that handle the individual input modalities, passing those results to other agents that interpret those results. For example, the audio input is directed to a speech recognizer that produces a string of words that is then passed to a natural language understanding system. The natural language system produces a logical form representing the user's request and passes it to the Facilitator agent, which decomposes it into requests that can be handled by the individual application agents.

A major advantage of using an agent-based architecture is that it provides simple mix-and-match for the components. In developing systems, we have used three different natural language systems: a simple one, based on Prolog DCG (Definite Clause Grammar), for rapid prototyping, then an intermediate one, based on CHAT [11], and finally, our most capable research system GEMINI [7, 8]. Similarly, we have used different speech recognition systems, substituting to meet different criteria. We use research systems developed by another laboratory in our company[4] [2] and by a commercial spin-off from that group.[5] We have also created Japanese-language systems by replacing our English-language speech recognizer and natural language systems with equivalent Japanese ones. In the near future, we expect to also support Korean. An agent-based architecture permits speech recognizers and natural language systems for multiple languages to be incorporated into the same system. We do not attempt to automatically identify the language being spoken, but have the user explicitly specify which language he will be using.

Our interface supports a rich set of interactions between natural language (spoken, written, or typed) and gesturing (*e.g.*, pointing, circling) – much richer than that seen in the *put-that-there* systems. Deictic words (*e.g.*, *this, them, here*) can be used to refer to many classes of objects, and also can be used to refer to either individuals or collections of individuals. Both the natural language expression and the gesture can be ambiguous, and a *Modality Coordinator* agent sends to each agent dealing with an individual modality the information from other active modalities that could be used to resolve the ambiguities.

We have found that including a pen in the user interface has several significant advantages. First, the gestures that users employ with a pen-based system are substantially richer than those employed by other pointing and tracking systems (*e.g.*, a mouse). Second, handwriting is a important adjunct to spoken language. Speech recognizers (including humans) can have problems with unfamiliar words (*e.g.*, new

[1] URL: http://isx.com/pub/I3

[2] URL: http://www-ksl.stanford.edu/knowledge-sharing/

[3] These capabilities are analogous to routines in a library or to an object's methods in object-oriented programming.

[4] URL: http://www-speech.sri.com/

[5] Corona Corporation, 333 Ravenswood Avenue, Menlo Park, CA 94025 (domain: coronacorp.com)

names). Users can use the pen to correct misspelled words, or may even anticipate the problem and switch from speaking to handwriting. Third, our personal experience is that when a person who has been using a speech and gesture interface faces an environment where speech is inappropriate, replacing speech with handwriting is more natural.

Multiagent Systems

In current multiagent systems, the user is typically interacting with a single agent at a time, although that agent may have hidden interactions with other agents. This assumption enables relatively simple interfaces. Our agent architecture is designed to handle complex requests that involve multiple agents. In some of these requests, the components will correspond to the capabilities of various application agents, and therefore can be handled by simple decomposition. In other requests to the system, the user's model of the task will not match the structure of the community of agents, and the system will have to translate the user's model into the system's model before apportioning subtasks to the agents. We believe that natural language is an important component of the user interface because it provides a very natural mechanism for stating complex queries and for allowing the user to work relative to his model of the tasks, rather than the system model.

One of the big problems that arises for using natural language in a multiagent system is how the language related to the individual agents is combined. We have created an Agent Development Toolkit to help the programmer create agents, including specifying the vocabulary and logical form predicates for a new agent. However, our experience in building natural language systems is that a good ontology is crucial, and key to achieving good performance. How people who are independently developing agents would provide language specifications that would fit into a common ontology is a problem that we are just beginning to address. Our initial approach is to provide a substantial initial ontology that other agent developers would not substantially change or extend—most of their changes would be minor variations or additional instances of entries already in the ontology.

User as Agent

One of the desirable side effects of using agents to implement the user interface to a multiagent system is that the user appears to be just another agent to the application agents. This simplifies the design of the application agents because they can have a simple interaction (single modality) with the user interface agents that handle the complexities of multimodal interaction with the user. It also simplifies the design of the application agents because they do not need to treat interactions with the user as distinct from interactions with the automated agents. A consequence of this is that what appears to the application agent to be a single interaction may actually involve the user and automated agents.

A coarse-grained mix would have the automated agent as a fall-back for the user, or *vice versa*. For example, if an agent queried the user and he did not respond within a specified time, an automated agent would be queried as to its best guess about what the user wanted done. Or, the system could query this automated agent first, and interrupt the user only if the automated agent cannot produce an answer that exceeds some confidence threshold. In producing this answer, the automated agent could look at the user's history (past interactions) and specified defaults (values and rules), or combinations of these.

A finer-grained mix would see automated agents handling parts of more complex interactions between the user and an application (*e.g.*, completion). We see this as an extension of the notion of adding more and more intelligent layers to the interpretation of multimodal input.

Multimodal Output

Our systems support only coarse-grained multimodal output. For example, one of our application agents provides airline flight-schedule information, which is usually presented as a graphical table. However, if the user has a speech-only interface (*e.g.*, a telephone), the table is read to him.

Our system also uses some elementary planning and reasoning capabilities in selecting the appropriate modality. For example, a user can request to be notified when a specified event occurs. Our *Notify* agent has a set of rules for making the notification. If the user is not then using his computer, the agent will try to give him the message by telephone. This notification-by-telephone task is also specified by a set of rules. First, the agent checks the user's calendar (via the calendar agent) to attempt to determine where he currently is. If it finds an entry with an identifiable location, it queries the telephone book (another agent) to get the phone number. It then has the telephone agent place the call. We use a PIN (personal identification number) to determine that the intended person is the one answering the telephone, but automatic speaker-verification technology may be used in future systems. If the attempt to notify by telephone fails (no one answers, or the user is not available), the Notify agent falls back to the next-best scheme given by the rules (*e.g.*, call his pager, send e-mail, call his secretary).

We are working on providing more sophisticated control of the use of output modalities. We are also interested in developing more fine-grained interaction between the various potential output modalities, but this is not one of our current priorities.

Lightweight Interfaces

One of the big advantages of our agent-based approach is that it enables a PDA or other low powered computer (*e.g.*, a portable) to have a user interface that incorporates substantial amounts of intelligence. Only the low-level user interface agents need to be running on the user's computer—all the other agents can run on remote computers. Thus, our current systems running on PCs or PDAs make use of speech recognizers, natural language systems, and other systems that require powerful workstations.

Summary

Much of the work in multimodal (multimedia) communication uses the information to be conveyed as the primary determinant of the modalities to be used. Our work assumes that much of the information can be adequately conveyed in more than one modality, and thus we assume that the user and the available resources will determine the appropriate modality or mix of modalities.

Acknowledgements

This paper is based on work that was supported in part by a contract to SRI from the Electronics and Telecommunications Research Institute (Korea). Phil Cohen (now at the Oregon Graduate Institute) was project leader until August 1994, and responsible for many of the design decisions in the systems described here. Any opinions expressed in this paper are strictly those of the authors.

References

[1] Adam Cheyer and Luc Julia. Multimodal maps: An agent-based approach. In *Proc. of the International Conference on Cooperative Multimodal Communication (CMC/95)*, Eindhoven, The Netherlands, May 1995.

[2] Michael Cohen, Hy Murveit, Jared Bernstein, Patti Price, and Mitchel Weintraub. The DECIPHER speech recognition system. In *IEEE ICASSP*, pages 77–80, 1990.

[3] P. R. Cohen. The role of natural language in a multimodal interface. In *The 2nd FRIEND21 International Symposium on Next Generation Human Interface Technologies*, Tokyo, Japan, November 1991.

[4] P. R. Cohen, A. Cheyer, M. Wang, and S. C. Baeg. An open agent architecture. In O. Etzioni, editor, *Proc. of the AAAI Spring Symposium Series on Software Agents*, pages 1–8, Stanford, California, March 1994. American Association for Artificial Intelligence.

[5] Philip R. Cohen, Mary Dalrymple, Douglas B. Moran, Fernando C. N. Pereira, Joseph W. Sullivan, Robert A. Gargan, Jon L. Schlossberg, and Sherman W. Tyler. Synergistic use of direct manipulation and natural language. In *Human Factors in Computing Systems: CHI'89 Conference Proc.*, pages 227–234, New York, 1989.

[6] Earl Craighill, Douglas Moran, and Ruth Brungardt. Research in information structures and software architectures for C3I systems. In *Joint Defense Laboratories Conference on Command, Control and Communications (C3)*, Ft. McNair, Washington, D.C., June 1987.

[7] John Dowding, J. Mark Gawron, Douglas Appelt, John Bear, Lynn Cherny, Robert Moore, and Douglas Moran. GEMINI: A natural language system for spoken-language understanding. In *Proc. of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 54–61, Ohio State University, Columbus, Ohio, 22–26 June 1993.

[8] John Dowding, Robert Moore, Francois Andry, and Douglas Moran. Interleaving syntax and semantics in an efficient bottom-up parser. In *Proc. of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 110–116, New Mexico State University, Las Cruces, New Mexico, 27 June – 1 July 1994.

[9] S. L. Oviatt. Pen/voice: Complementary multimodal communication. In *Proc. of Speech Tech'92*, pages 238–241, 1992.

[10] S. L. Oviatt, P. R. Cohen, and M. Wang. Reducing linguistic variability in speech and handwriting through selection of presentation format. In K. Shirai, editor, *Proc. of the International Conference on Spoken Dialogue: New Directions in Human-Machine Communication*, Tokyo, Japan, November 1993.

[11] F. C. N. Pereira. *Logic for Natural Language Analysis*. PhD thesis, U. of Edinburgh, 1983.

[12] D. G. Schwartz. *Cooperating Heterogeneous Systems: A Blackboard-based Meta Approach*. PhD thesis, Case Western Reserve University, Cleveland, Ohio, 1993.