MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Building Spoken Language Collaborative Interface Agents

Candace L. Sidner

TR2002-38 December 2002

Abstract

This article reports on experiences with collaborative interface agents using spoken dialogue to collaborate with users working with graphical user interface applications. Collaborative interface agents provide users with the means to manage tasks and leave many of the details to the agent. The article presents four different collaborative agents and associated applications. It reports on lessons learned in building these agents, including the importance of choosing tasks that relieve the user of unnecessary detail, and providing speech capabilities that are useable for a wide range of users. In particular, the article reports on the success in developing a subset language for speech understanding in one of the agents. Finally, the article discusses the advantages of using the explanation capabilities in collaborative agents to help users learn new interface functionality.

to appear in Ṕractical Dialogue Systemséditted by D. Dahl

Submitted June 2002.

Building Spoken-Language Collaborative Interface Agents

Candace L. Sidner
Mitsubishi Electric Research Laboratories

Keywords: collaboration, conversation, interfaces, collaborative interface agents, spoken dialogue, subset languages

Abstract: This article reports on experiences with collaborative interface agents using spoken dialogue to collaborate with users working with graphical user interface applications. Collaborative interface agents provide users with the means to manage tasks and leave many of the details to the agent. The article presents four different collaborative agents and associated applications. It reports on lessons learned in building these agents, including the importance of choosing tasks that relieve the user of unnecessary detail, and providing speech capabilities that are useable for a wide range of users. In particular, the article reports on the success in developing a *subset* language for speech understanding in one of the agents. Finally, the article discusses the advantages of using the explanation capabilities in collaborative agents to help users learn new interface functionality.

1. INTRODUCTION

This article reports on experiences with collaborative interface agents using spoken dialogue to collaborate with users using graphical user interface (GUI) applications. A team of researchers and software engineers[1]spanning two commercial research organizations (Lotus Development Corporation and Mitsubishi Electric Research Laboratories) built four such agents. This article reports on our original goals in building spoken-language collaborative interface agents, on our experience in developing agents for four different applications, and on the lessons we have learned throughout this exploratory process. As this article will make clear, many current GUIs do not easily lend themselves to adding a collaborative agent, so using a collaborative agent challenges developers to find tasks that reduce the user's interface burden.

Our goals have evolved over the several years during which these four agents were built. The original goal was to understand how interacting with a collaborative agent would differ from just giving commands to an interface. As the work progressed, we began to focus more on making use of the powerful capabilities of software agents in reducing the user's burden in using complex applications. We also found that the speech recognition technology needed to be better utilized to make the recognition highly reliable. Finally, we experimented with agents that provided explanations for how to achieve tasks using the interface. We pursued these goals with four collaborative interface agents and associated applications that are discussed in this paper.

Effective collaboration between a person and an interface agent depends fundamentally upon the agent being able to hold a conversation about the purposes of the collaboration. Computational linguistics research on conversation [1] has identified several key components to such conversations:

– the role of segmentation in determining the major units of a conversation,

---

[1] The systems reported here resulted from the efforts over time of several people in addition to the author. Thanks to: Carolyn Boettner, Clifton Forlines, Bret Harsham, David McDonald, Chris Maloof, Neal Lesh, Charles Rich, Bent Schmidt-Nielsen and Peter Wolf.

- the relation between the intentions of the conversational participants to the purposes conveyed in conversation,
- the mechanisms for change in the participants' purposes during the conversation,
- and the focus of attention, that is, the relative salience of purposes, objects, actions and other entities as the conversation unfolds.

Combining these aspects of conversation with collaboration requires relating the intentions conveyed in utterances to the collaboration. The collaborative interface agents discussed in this article use these aspects of conversation and the capacity to collaborate in practical interface applications. These agents collaborate with users in tasks for email, scheduling meetings and two versions of TV entertainment control.

One of the lessons learned in building collaborative interface agents, namely, to provide speech understanding that is feasible with today's technology, has been incorporated into the later interfaces reported on here. The first two agents, for collaboration on email and for scheduling meetings, understood only some of the utterances users typically would say to them. Without an extensive data collection effort, we found that not every user could talk to these agents because we only predicted a portion of the vocabulary and grammar that users spoke with. Unless every user could easily talk to the agent, the systems built were an insightful view of the future, but not very practical for the average user now. To make collaborative agents usable now required a focus on making speech intelligible for the agent. The fourth agent, for navigation of TV schedules and program recording tasks, constrains how the user can speak but does so with persistently present information about the legal utterances the user can say. The resulting conversations are direct and practical, but lack the informality and incompleteness typical in conversations among people. Improving the naturalness of these conversations depends upon not only more powerful speech recognition technology but also many other capabilities, such as better turn taking, observation the conversational partner, interpretation of gesture, and more knowledge of the implicit aspects of the domain.

A second lesson learned concerns the benefits of collaboration. Collaboration has the power to transform the user from a low level clerk seeing to every detail of his tasks into a manager of his tasks, who delegates details to an agent. Collaborative agents are most beneficial when the agent is given complete sub-tasks to perform rather than individual GUI acts. Simply replacing pointing at a menu with spoken commands ("File, Open") does not yield useful collaborations for several reasons. First, many GUIs have been designed to optimize the use of the mouse. Speech command of GUI actions is generally no faster than clicking and can be slower when correction for speech errors is included. Even for those actions (usually associated with search) that are faster with speech than the mouse, collaboration does not have enough overall payoff due to the small efficiency gained with speech. Second, the GUI requires users to control every aspect of the interface and associated task. In contrast, we are designing collaborative agents, who understand speech and the user's task well enough to do parts of it without detailed supervision. These agents offer even more powerful benefits to users than the point-and-click, manage-every-detail style interfaces users live with now.

2. COLLABORATIVE INTERFACE AGENTS

In order to understand how to use a collaborative interface agent with existing applications, several questions must first be addressed: What exactly is a collaboration? What is a collaborative interface agent? How are collaborative agents different from other systems that have conversations with users?

Collaboration is a process in which two or more participants coordinate their actions toward achieving shared goals. Collaboration between two people who are co-present involves:

– communication between them;
– interaction with the shared artifact that is the locus of their shared goals (e.g., when painting a house, the house is a shared artifact, when trouble shooting a problem under the hood of the car, the engine is the shared artifact);
– and observation of what the other person is doing with the shared artifact.

As the collaboration unfolds, the participants also come to share certain beliefs. Specifically, they come to share beliefs about their intentions to do actions, beliefs about how those actions support their shared goals, beliefs about how to advance the collaboration, and beliefs about the artifacts [2],[3].

A collaborative interface agent is designed to take the place of one of the human participants in a collaboration involving a shared interface to a computer application [4]. The agent therefore must be able to communicate with the human user, manipulate the interface, and observe the user's manipulation of the interface. It must also be able to come to have shared beliefs of the same type as in the human-human collaboration [2],[3],[5]. A diagram of collaboration between a user and a collaborative interface agent is shown in Figure 1.

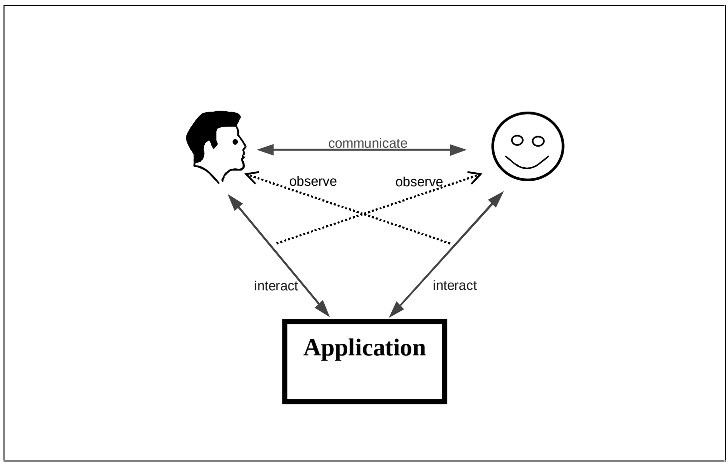


Figure 1: Human-Agent Collaboration

In this article we will focus on collaborative interface agents that communicate via spoken dialogue with a user who is using a software application as the shared artifact. Such an agent speaks utterances to a user and understands a user's spoken utterances. Spoken (as opposed to textual) communication is especially valuable when the user's hands and eyes are engaged with the application interface.

2.1 Collaborative Conversations versus Imperative Interactions

While speaking to an interface can be easily argued for, one might wonder why a **collaborative** agent is needed. Why not just replace GUI operations with spoken commands? Our answer largely concerns how to reduce the burden of what users must convey to computers. The imperative command approach treats the computer as a kind of slave to the user's request. It puts the entire burden for managing the overall task on the user so that the user must tell the system exactly what to do at each step.

The imperative command approach suffers from a significant conceptual flaw. Users want to know and say less about the details of applications than they currently do. The mantra of users should be: do more with less input. Instead, in current GUIs, the user cannot leave any action out, and furthermore, all of the actions are low-level and detailed. Providing GUI commands in speech rather than by pointing and clicking does not address this problem.

Collaboration offers the user something new: a means to be freed from having to even think about portions of his or her tasks. The user describes an action or outcome to the agent, and then the agent undertakes to accomplish the part of the outcome that it knows how to do. The action might take the agent several sub-actions to accomplish, but the user will be freed from performing or even supervising those actions. In accomplishing its sub-tasks, the agent is responsible for asking for the information it lacks when accomplishing its task. However, for this describe-and-act paradigm to work, the agent needs to have a larger context for interpreting what the user is undertaking. The larger context is essential for choosing actions and objects. Collaborative agents are built to have knowledge of the overall task and how actions fit together with other actions to achieve portions of the task.

In the collaboration framework, the agent is not a slave, but functions as a somewhat knowledgeable assistant. It cannot do everything a user can do because it still is not as sophisticated as a human, either in its ability to do tasks in the world, in its understanding users' beliefs and intentions, or in making everyday inferences about the world. It can, however, undertake portions of a user's task and accomplish those actions in light of the user's goals and the known means of achieving them.

Interfaces to applications do not offer such help because they cannot. The interface has extremely little information about what the user is doing. In most existing computer applications, the information that the interface or the application itself does have is generally not used to reason about what the user is doing or wants to achieve. While inroads in certain applications are being made in this direction (for example, the Microsoft paper clip models user activities with Bayesian networks), such software additions do not converse with the user about the user's goals or means of achieving those goals. So at best, the interface "agent" guesses what is going on. Our approach relies on using the human-human collaboration model more closely and makes use of the capability to communicate about goals and tasks directly.

User requests generally reflect the larger activities that users are undertaking. A part of collaboration includes interpreting the relations between simpler actions and more complex ones, so that an agent can automatically recognize the relation between the current action and more general ones. This recognition process, called *intended plan recognition* [6], can be supported algorithmically in collaborative agents so that the agent can interpret the more general purposes underlying user actions.

2.2 Collagen-A Reusable Architecture

To support an interface agent performing as a collaborator, the research group at Mitsubishi Electric Research Laboratories has developed a Java middleware program, called Collagen[TM], for building COLLaborative interface AGENts to work with users on GUI applications [7],[4]. Collagen is designed with the capability to participate in collaboration and in conversation based on [1],[5] and the SharedPlan theory of collaboration [2],[3].

The spoken-language architecture of Collagen is shown in Figure 2. Collagen makes use of a discourse state consisting of a focus stack and a recipe tree, which is created using a combination of the discourse interpretation algorithm of [5] and plan recognition algorithms of [8], [9]. Plan recognition provides Collagen with the ability to infer how one or more actions can be used to accomplish a more general action. The discourse interpretation algorithm in Collagen takes as its input all user and agent utterances and interface actions, and accesses a library of recipes. The recipes represent the actions of the domain as action types, which are instantiated for each activity that occurs with a particular user in a particular interaction situation. After updating the discourse state, Collagen makes two resources available to an interface agent: the discourse state, and an agenda of next possible actions computed from the discourse state. The Collagen middleware requires an adapter, which is a sub-system that transforms the descriptions of action types and objects in the agent's internal language to program calls in the application programmer's interface (API). The adapter also informs the discourse interpretation of any GUI events done by either the user or the agent so that these can be incorporated in the agent's understanding of the interaction.

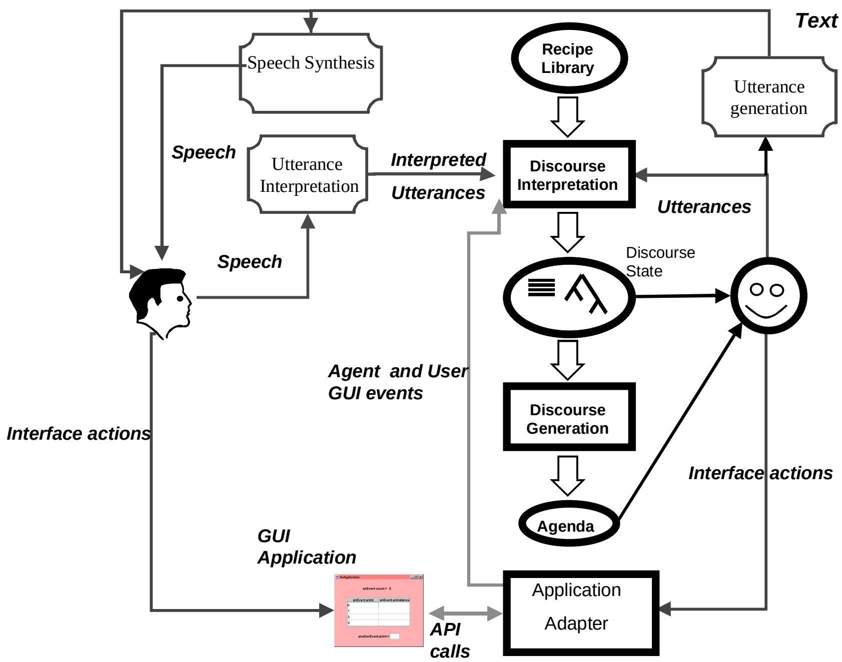


Figure 2: Collagen architecture for spoken-language collaborative agents

All spoken-language agents must have a means for understanding user input and generating spoken output. In the Collagen architecture, utterance interpretation takes a spoken utterance and produces an interpretation for the utterance. Several sub-components make this possible in our spoken agents. A speech recognition engine takes user utterances and produces strings that are fed to a syntactic analyzer. The analyzer's results are semantically interpreted to form descriptions in an utterance intention language, which captures both utterance's intentional and propositional content. Interpreted utterances are interpreted independent of context, and later processing uses the discourse state to modify the interpretation. The utterance intention language is based on an artificial discourse language [10]; it serves as the internal language for all Collagen agents. Results of semantic and intention processing are passed to the discourse interpretation module that is central in the Collagen architecture shown in Figure 2.

When any of the agents communicate, they make use of the Collagen facilities for sentence output, which translates the internal intention language to English phrases using string template mapping. The strings are also sent to a synthesizer for speech output. Both text and spoken forms of utterances can be presented to the user. In some interfaces, the recipes in the recipe library specify special sentence forms including ones with markings for emphasis, which can be interpreted by the synthesizer.

3. FOUR SPOKEN LANGUAGE COLLABORATIVE AGENTS

Over the past several years, the Collagen team has built four spoken language collaborative interface agents as well as a number of other types of interface agents that do not involve speech [7]. Two of the spoken-language agents assisted users on business tasks, one for email (the email agent) and one for scheduling meetings with sales contacts in Notes™ databases (the scheduling agent). The other two agents, the VCR agent and the Entertainment Center agent, helped users operate time-shifted video recording systems. Each of these agents used the same underlying Collagen middleware for modeling collaboration and conversation. Three of the agents used commercially available IBM speech recognition and synthesis software. The email agent used a research prototype developed at IBM's T.J. Watson Research Labs.

For each agent, a recipe library of the action types, both primitive and abstract, of the domain was created, and an adapter was programmed to access the API of the application. In addition each of the agents procedurally encoded a very small amount of specialized knowledge about activities it might perform for its domain.

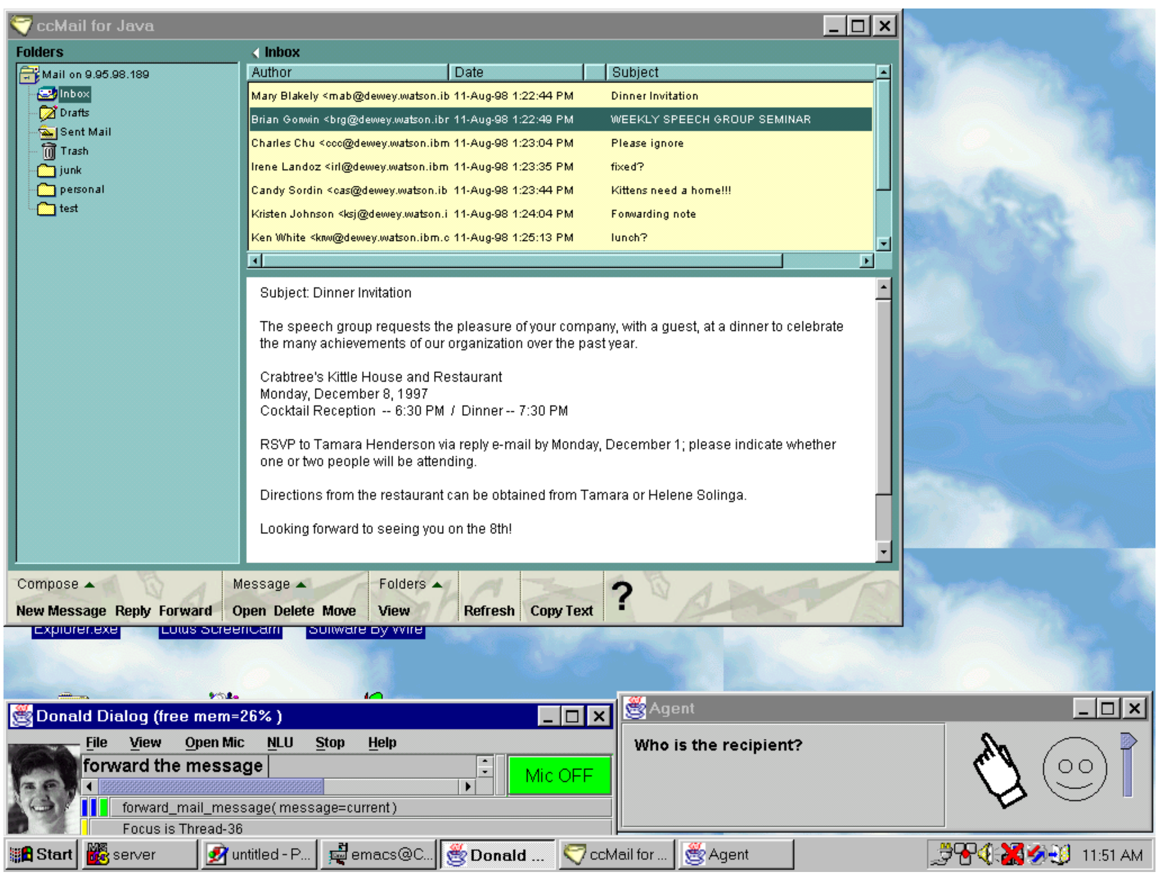


Figure 3: The Email agent and Esuite email application

3.1 The Email Agent

The email agent, shown in Figure 3, was the first collaborative interface agent to incorporate speech understanding and synthesis. Figure 4 presents a conversation, which was captured from system logs, between a user and the email agent while operating the Lotus Development Corporation Esuite ™ email application. This conversation is annotated at major task boundaries for easy perusal by the reader.

In this conversation, the email agent knows enough about email take on several tasks. It opens messages that a user selects, asks about the times for scheduling meetings and then changes the user's schedule as needed. It also knows that messages are likely to be responded to (by forwarding as well as replying) and then sent, and that email is managed by storing, printing and deleting after the message has been read.

User: "Let's work on email."

Agent: "Ok."

Choosing a message

User: Selects message from Brian Gorwin about Weekly Speech Group Seminar.

Agent: Opens message from Brian Gorwin about Weekly Speech Group Seminar.
"Shall we use the contents of message from Brian Gorwin?

Meeting Planning

User: "This is about a meeting with Brian."

Agent: Setups meeting with Brian Gorwin.
"Let's schedule event tentatively at a time."

User: "Schedule the meeting at three p.m. Thursday."

Agent: Schedules event tentatively at three p.m. Thursday.

Agent: "There is a conflict of meeting with Brian Gorwin at three p.m. Thursday with meeting with Irene Landoz at three p.m. Thursday. Let's remember to re-schedule meeting with Irene Landoz at three p.m. Thursday."

User: "ok."

Agent: Finalizes meeting with Brian Gorwin at three p.m. Thursday.

Forwarding the message

"Shall we react to message from Brian Gorwin about Weekly Speech Group Seminar?

User: "Forward the message."

Agent: "Who is the recipient."

User: "The recipient is Steve Krantz."

Agent: Presents forward template for message from Brian Gorwin about Weekly Speech Group Seminar to Steve Krantz.
"Shall we fill in message from Mary User about Fwd: Weekly Speech Group Seminar?"

User: "Ok."

Agent: Adds signature to message.
"Shall we cc message from Mary User about Fwd: Weekly Speech Group Seminar?"

User: "No."

Agent: "Ok, your turn."

User: "Send the message."

Agent: Sends the message from Mary User about Fwd: Weekly Speech Group Seminar.

Cleanup of the message

"Shall we store, print or delete this message?"

User: "Print the message."

Agent: Prints the message from Mary User about Fwd: Weekly Speech Group Seminar

Agent: "Shall we work on email?"
User: "No."

Figure 4: A conversation with the email agent

The email agent had a recipe library for knowledge about the actions of the domain. Of the four agents, the email agent had the largest recipe library, which included about thirty-five primitive action types (which correspond to interface action types accessible with the API), and twenty-nine non-primitive action types, with at least one, and sometimes several, recipes for non-primitives. The most general action types in the recipe library of actions are: doing email, sending email, searching email, managing email documents and reading a message. Each of these action types is described by a recipe of several steps, some steps being optional and some containing steps with partial orders. Recipes can included pre and post condition tests, and they also describe objects in the domain that are used in the action descriptions. For example, doing email includes the steps reading a message, responding to the message, managing the message in the inbox, and then repeating that process on additional messages. Some action types in the recipe library have several recipes corresponding to different ways to perform that action type.

Modelling a user's email tasks to describe in a recipe library is a non-trivial task. To accomplish it, WOZ studies were conducted at Lotus in which users spoke to a human wizard as they used their email. Users could choose at any time to perform GUI operations themselves or to request that an email task (described as they wished) be performed by the wizard. User email actions were gleaned from transcripts of these sessions and described in the library.

From observing the types of conversations possible with the email agent, we concluded that the existing GUI interface precluded the agent being very helpful. In fact, the email agent actually only does very minor tasks for the user: opening and closing message windows without being asked, changing the user's schedule (when approved), and adding signatures to email messages. In large part, doing more for users would have required a very different interface so that the agent could have taken on, for example, the whole burden of constructing email messages. In designing the second agent, we decided to focus on tasks where the agent could relieve the user from as many of the time consuming details as possible.

3.2 The Scheduling Agent

The second collaborative interface agent used the Lotus product Notes™ and is shown in Figure 5. The scheduling agent was intended to help Notes users who had databases of information and who performed routine scheduling tasks involving their colleagues and clients. The agent made use of existing data in Notes databases to gather profiles, check schedules of people and rooms, and added to that data to create profiles, new meeting entries and send email messages. A sample conversation of the scheduling agent is shown in Figure 6.

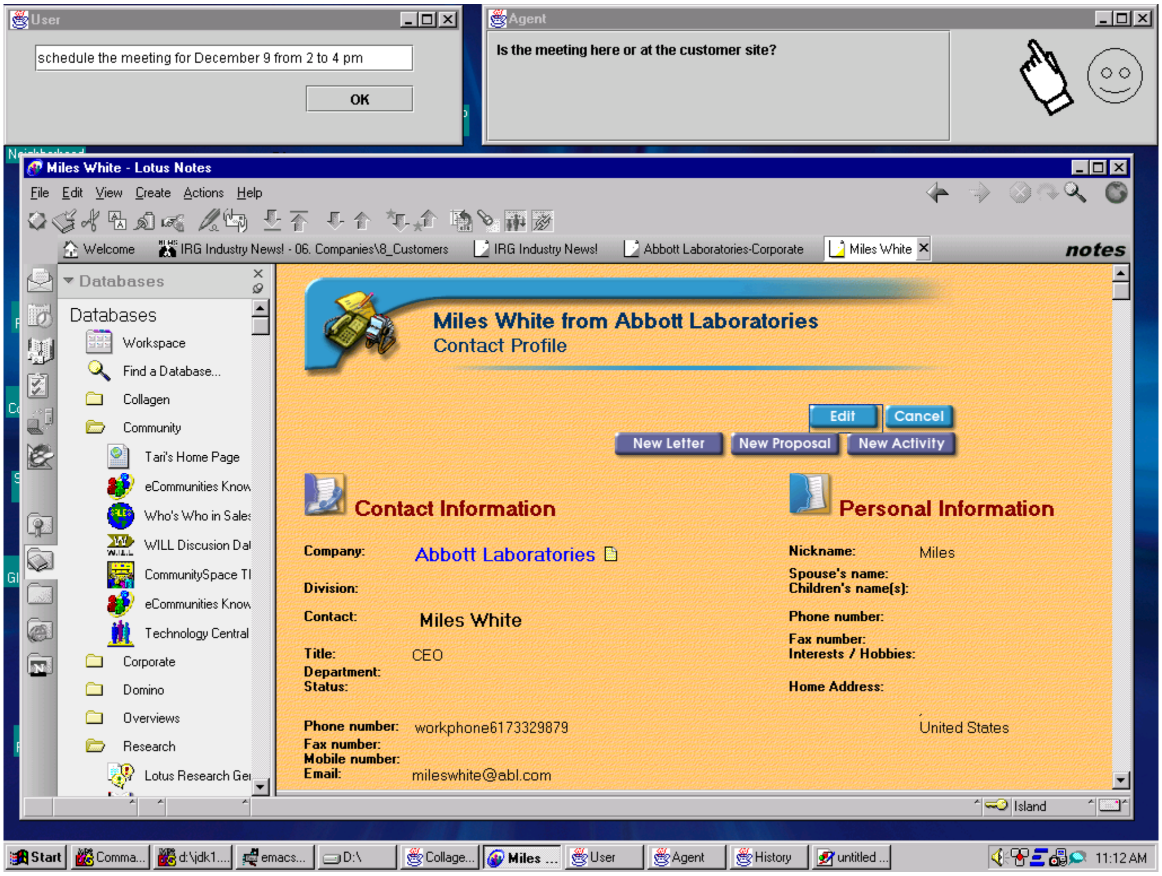


Figure 5: The scheduling agent with a Lotus Notes application

---

Agent: "Let's manage scheduling contacts."

Creating a profile

User: "Let's create profile for Miles White from Abbott Laboratories."

Agent: "What is the email address?"

User: "The email address is mileswhite@abl.com."

Agent: "What is the work phone?"

User: "The work phone is 617-332-9280."

Agent: Goes to profile for Abbott Laboratories.

Creates profile for Miles White.

Scheduling a presentation

"Do you want to schedule a presentation for Miles White?"

User: "Schedule the meeting for December 9 from 2 to 4 p.m."

Agent: Makes meeting entry for Miles White on December 9.

"Is the meeting here or at the customer site?"

User: "The site is here."

Agent: "Which building?"

User: "The building is Rogers Street."

Agent: Locates room 5003 in Rogers Street.

"I am going to reserve room 5003 in Rogers Street."

User: "Ok."

Agent: Reserves room 5003 in Rogers Street.

Finding attendees for the presentation

"Who will attend the meeting?"

User: "Invite Fred Smith and Jerry Johnson."

Agent: Invites Fred Smith and Jerry Johnson.

"Will the customer need consulting services?"

User: "Invite Pam Bridges and Jimmy Barnes from Lotus Consulting."

Agent: Invites Pam Bridges and Jimmy Barnes.

"The following people: Pam Bridges have a conflict with the meeting time."

User: "Instead invite Julie Knox."

Agent: Invites instead Julie Knox.

"I will get back to you to confirm this meeting."

User: "Ok."

Agent: Sends invitations to Fred Smith, Jerry Johnson, Julie Knox and Jimmy Barnes.

Figure 6: A conversation with the scheduling agent

The speech understanding sub-systems for the scheduling agent, as well as for the VCR and Entertainment Center agents, were designed similarly. Grammars and semantic interpreters were created by the Collagen team and used with the Java Speech API as implemented by IBM, and with IBM commercially available speech recognition engines (IBM Via Voice products). Like the email agent architecture, results of semantic interpretation were passed to the discourse interpretation component of Collagen.

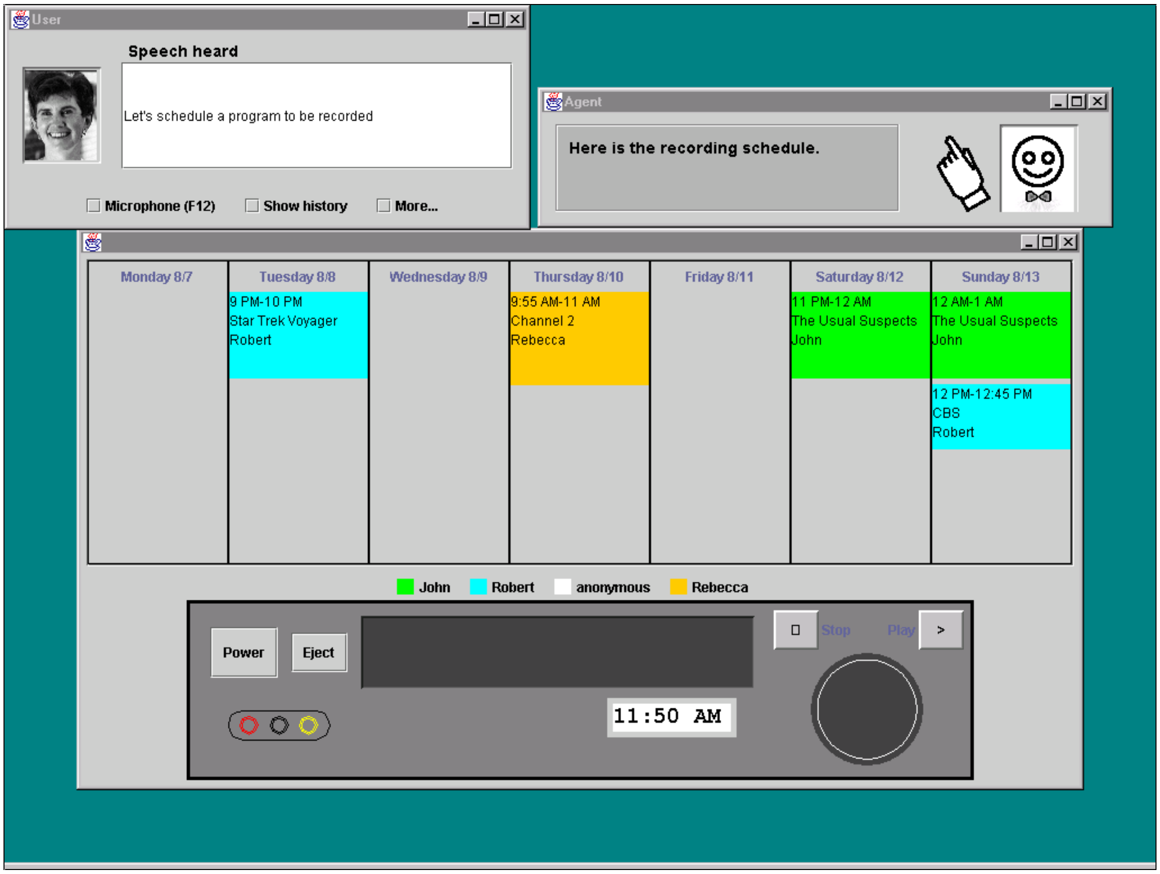
In constructing the email and scheduling agents, we built adapters to access the underlying Esuite email and Notes applications. In the case of the Notes application, the adapter did not access Notes directly but rather acted through a server, in part because Notes is not a Java application, and in part because it is a large and complex system. Details of the architectural changes for the Notes application are given in [11].

Our collaboration design goals were more successfully met for the scheduling agent than the email agent. Compared to the email agent, the scheduling contact agent undertook far more tasks for the user: it found meeting rooms in specified buildings, checked calendars for proposed participants to see if the participants were free at specified times, and sent email with preformatted invitations to proposed participants. The user was never required to oversee a GUI scheduling program. Instead, the agent simply collected the needed information and undertook appropriate scheduling tasks on its own. While only an experimental prototype, this second agent had the potential to shorten user involvement in setting up meetings with groups of participants because the user simply did not need to pay attention to the details that normally surface in GUI control of application tasks. We concluded that agents were more valuable when they offered to perform time-saving tasks for the user.

3.3 The VCR and Entertainment Center agents

The remaining two agents, the VCR agent and the Entertainment Center agent, are related because the Entertainment Center agent and application are a second generation version of the VCR agent and application. The first of these agents was created to explore collaborative agents working with users who knew what they wanted to but had never been successful in using an onscreen interface to accomplish their tasks. Since most users do not know how to program their VCRs and navigate online TV schedules, a collaborative agent was designed to take a high level simple request and gather all the details it needed itself.

The VCR agent, shown in Figure 7, focused on assisting a user to record programs on his VCR and also provided information about the TV schedule. A sample conversation is shown below the screen shot.



Recording a program

User: "I want to record a program."

Agent: "Here is the recording schedule."

User: "Show me the TV guide for Thursday."

Agent: [Displays TV guide for Thursday.]

"What is the name of the program?"

User: "The program is the X-Files."

Agent: [Adds X-Files to recording schedule.]

"Ok. What next?"

Recording a program

User: "I want to record another program."

Agent: "What is the name of the program?"

User: "It's on NBC."

Agent: "What is the starting time?"

User: "It starts at 8 p.m."

Agent: "What is the ending time?"

User: "It ends at 10 PM."

Agent: "What is the date?"

User: "Monday."

Agent: "I'm sorry that there is a recording conflict. Robert has already scheduled the recording of Star Trek at that time."

User: "Ok."

Figure 7: The VCR agent and application and a conversation with the agent

The Entertainment Center agent is depicted in Figure 8. It used the Entertainment Center (EC), which was a more complex shared artifact than the VCR agent used. The EC agent undertook four basic tasks: navigating a week-long TV show schedule, recording a program from that schedule, playing back a pre-recorded program and presenting a TV show currently being broadcast.

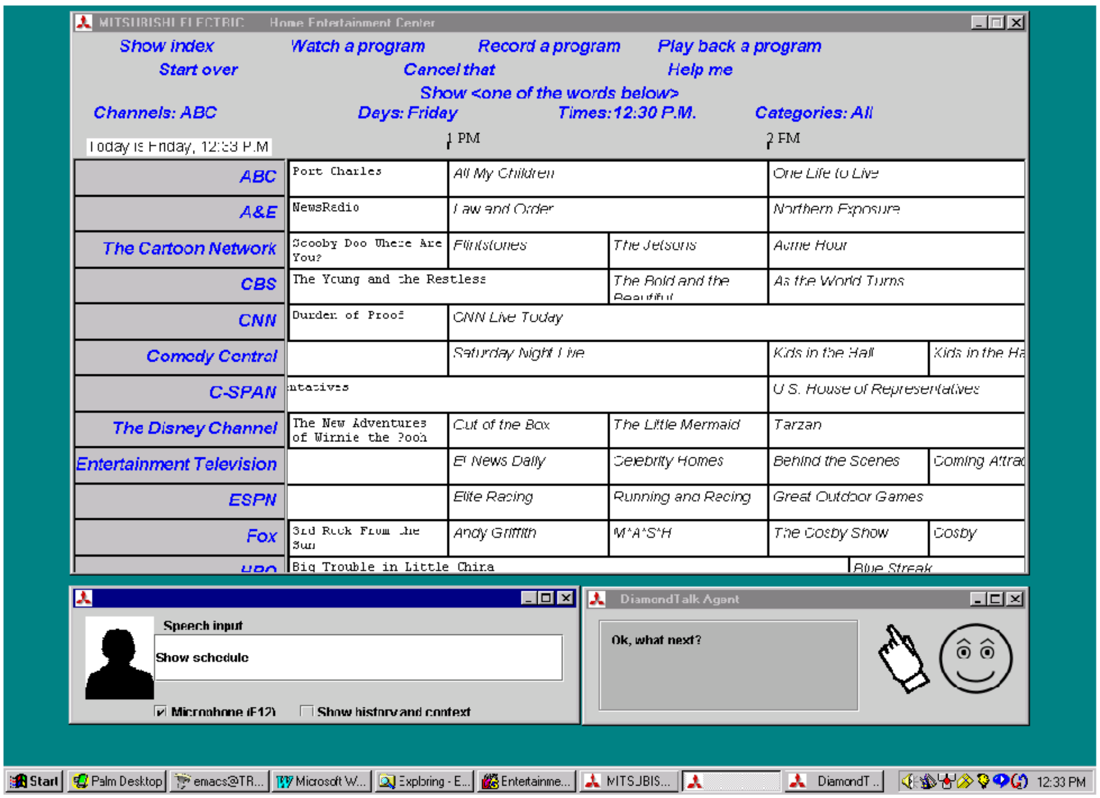


Figure 8: The Entertainment Center and Agent

While the VCR agent interpreted much more complex user utterances than the EC agent could, the simplicity of spoken language in the EC agent made it useable by many people. A sample conversation for the EC agent is presented in Figure 9.

Agent says "Welcome to Diamond Talk. Try saying something shown in blue."

Navigation activities

User says "Show schedule."

Agent says "Ok, what next?"

User says "Show Friday."

Agent shows schedule for Friday 10:30 A.M.

Agent says "Ok, what next?"
User says "Show 7 P M."
Agent shows schedule for Friday 7:00 P.M.
Agent says "Ok, what next?"
User says "Show the W B."
Agent scrolls to the WB.
Agent says "Ok, what next?"
User says "??? Record a program ???"
Agent says "I didn't get that. Please repeat."

Recording a program

User says "Record a program."
Agent says "What is the program title? (exactly as on the screen)."
User says "The Invisible Man."
Agent shows episodes of The Invisible Man to be broadcast.
Agent says "Should I record The Invisible Man Friday 8:00 P.M.?"
User says "No."
Agent says "What day and time do you want to record The Invisible Man?"
User says "Friday 11 P M."
Agent scrolls to The Invisible Man Friday 11:00 P.M.
Agent adds The Invisible Man Friday 11:00 P.M. to the recording schedule.
Agent says "The Invisible Man has been added to the recording schedule Friday 11:00 P.M."
Agent says "Ok, what next?"

Playing back a program

User says "Play back a program."
Agent shows index of programs already and to be recorded.
Agent says "What is the title of the recorded program? (exactly as on the screen)."
User says "The Enemy Within."
Agent says "Should I play back The Enemy Within recorded Wednesday 7:30 P.M.?"
User says "Yes."
Agent plays The Enemy Within recorded earlier.
Agent says "Ok, what next?"

Navigation activity

User says "Show schedule."
Agent shows this week's broadcast schedule.
Agent says "Ok, what next?"
User exits.

Figure 9: A conversation with the EC agent

For navigation tasks, the EC agent simply performed the action requested by the user, while for the program tasks, the agent performed a simple form of collaboration by seeking missing information needed to do the task. When the agent was requested to do an action, it responded just as all Collagen agents can: it requested from the user the additional information that it needed, in this case, program names or channel names. Where there were several choices of times for a program of a given name, the agent also asked for clarification of times and confirmed that the user wanted to proceed. While clarification could be obtained using dialogue scripting languages, the collaboration model [5] provides general agent behavior that need not be written by the system programmer. Instead, a programmer, using the collaborative interface agent framework to build an application such as this one, focuses on describing the tasks to be accomplished and leaves the conversational capabilities to Collagen.

3.4 A comparison of the four agents

The four dialogues between a user and a collaborative interface agent share some similarities concerning the agent's behavior and abilities. All four agents require that the user set the overall context of the interaction by providing an intention to focus the interaction (for example, "Let's work on email;" "Let's create a profile for Miles White;" "I want to record a program;" or "Show schedule."). The generality of the user intention varies significantly in these four interfaces. For the email and scheduling agents, the user's intention is very generally stated, while for the VCR and EC agents, the intentions are much more specific.

The user directs the agent differently with these applications. For the email agent, the user is able to simply undertake the first primitive action (selecting an email from the onscreen email inbox using the mouse) and expects that the agent will understand how this action contributes to doing email. For the scheduling agent, the user agrees to the one overall activity that the agent is aware of, and then sets the direction in more detail by creating a profile for the client Miles White. The agent then begins to gather the details needed for creating the profile itself. For the VCR agent, the agent responds to the user's general intention of recording a program by providing the schedule of shows already recorded and waiting for the user to determine the next action. For the EC agent, the user's intention is so directly stated that the agent can simply perform the required action and then return the conversation to the user. All of these varieties are possible in Collagen based agents, but the variation results from the complexity of the recipe library and the way that the conversation begins.

Each of the sample dialogues differs on the use of mixed initiative in the conversation. The user initiates the email conversation, but once the agent understands what task is being done, it directs the conversation with questions about next activities and about how to accomplish a given activity. For example, given the task of forwarding the message, it asks about recipients, filling in the message and carbon copying the message to others. In the email conversation, the user also mixes in his initiative by performing GUI actions such as selecting a message to read. In the scheduling conversation, almost all the initiative rests with the agent. By comparison, the VCR and EC conversations are much more balanced, with the user initiating each task and the agent taking initiative to ask about information that it needs. The variation in initiative across these four examples results from experience in developing the email and scheduling agents. After building the email and scheduling agents, we concluded the user was better off having more initiative and hence more control of the interaction. To pass off initiative, the agent in the TV related applications uses the simple technique of asking "Ok, what next?" at the conclusion of each task. While this repetitive question is somewhat annoying, it also clearly gives the user initiative and control in the conversation.

The generation capabilities of the Collagen middleware are decidedly primitive compared to that state of the art. As a result, in some conversations (especially those for email and scheduling), the agent's utterances are less natural than human ones would be. In particular, the agent does not use reduced noun phrases or pronominal forms after first mention of an object. For example, "the message from Brian Gorwin about weekly Speech Group seminar" is repeated where a reduced noun phrase would have been appropriate. Because Collagen generates language using string template matching, its techniques do not perform as well conversationally as is possible. A richer generative capability would improve the linguistic capacity of collaborative agents that must use descriptive noun phrases. This capability awaits future work in Collagen.

Collaborative interface agents understand more about conversations than interpreting user goals and performing tasks that contribute to those goals. They track the discourse and retain a history of it. This history, called a *segmented interaction history*, can be shown to a user (or a developer) and is a English readable "dump" of the internal data structures that are the state of the agent's understanding of the complete conversational history. Figure 10 provides such a Collagen generated segmented interaction history for the email conversation from Figure 4. The conversation is segmented into units that are headed by the most general-purpose action for that segment. Segments are subdivided by more specific actions in sub-segments. The indentation illustrates to which purpose each action contributes. An early prototype of a Collagen agent for air travel planning experimented with commands to manipulate the interaction history to replay or re-use portions of the conversation [4]. In the four agents reported here, no effort has been given to these manipulations because no natural instances arose where this capability was useful. All four interface agents have segmented interaction histories that are produced as the conversation unfolds and are available at any time during the conversation.

Done working on email.

User: "Let's work on email."

Agent: "Ok."

Done reading a message from Brian Gorwin about Weekly Speech Group Seminar

Done agent selecting a message from Brian Gorwin about Weekly Speech Group Seminar

User: Selects message from Brian Gorwin about Weekly Speech Group Seminar.

Agent: Opens message from Brian Gorwin about Weekly Speech Group Seminar.

Done using the contents of a message from Brian Gorwin about Weekly Speech Group Seminar

Agent: "Shall we use the contents of message from Brian Gorwin?

Done scheduling a meeting with Brian at 3 p.m. Thursday

User: "This is about a meeting with Brian."

Agent: Setups meeting with Brian Gorwin.

Done agent scheduling event tentatively at 3 p.m. Thursday

Agent: "Let's schedule event tentatively at a time."

User: "Schedule the meeting at three p.m. Thursday."

Agent: Schedules event tentatively at three p.m. Thursday.

Agent: "There is a conflict of meeting with Brian Gorwin at three p.m. Thursday with meeting with Irene Landoz at three p.m. Thursday. Let's remember to re-schedule meeting with Irene Landoz at three p.m. Thursday

User: "Ok."

Agent: Finalizes meeting with Brian Gorwin at three p.m. Thursday.

Done reacting the message from Brian Gorwin about Weekly Speech Group Seminar

Agent: Shall we react to message from Brian Gorwin about Weekly Speech Group Seminar?

Done forwarding the message from Brian Gorwin about Weekly Speech Group Seminar

User: "Forward the message."

Agent: "Who is the recipient."

User: "The recipient is Steve Krantz."

Agent: Presents forward template for message from Brian Gorwin about Weekly Speech Group Seminar to Steve Krantz.

Done filling in the message from Mary User about Fwd: Weekly Speech Group Seminar

Agent: "Shall we fill in message from **Mary User** about Fwd: Weekly Speech Group Seminar?"

User: "Ok."

Agent: Adds signature to message.

"Shall we cc message **Mary User** about Fwd: Weekly Speech Group Seminar?"

User: "No."

Agent: "Ok, your turn."

Done sending, canceling or storing this message as a draft

User: "Send the message."

Agent: Sends the message from **Mary User** about Fwd: Weekly Speech Group Seminar.

Done storing, printing or deleting the message from Mary User about Fwd: Weekly Speech Group Seminar

Agent: "Shall we store, print or delete this message?"

User: "Print the message."

Agent: prints the message from Mary User about Fwd: Weekly Speech Group Seminar

Agent: “Shall we work on email?”

User: “No.”

Figure 10: The segmented interaction history for an email conversation

Building these interface agents sharpened our understanding of the agent’s contribution to reducing the user’s burden when performing tasks. From observing the email agent and user interaction, we decided that the agent would be more effective if it could do more for the user. In building the scheduling agent, we learned that it indeed relieved the user in meeting planning of much of the painful details. For the television recording-based agents, our learning played out in a different way. Providing high level but straightforward capabilities to users meant they did not have to learn what the agent needed in advance; they could rely on the agent to request additional information that it needed. As will be discussed below, anticipating the user’s needs is part of collaborative activity as well.

4. DESIGN ISSUES FOR COLLABORATIVE AGENTS

In this section we will discuss design issues in the development of the MERL EC and EC agent, the most recent agent built with Collagen. While the EC and EC agent are not intended as a commercial product, the EC agent was designed to perform with users in as realistic a consumer environment as possible. The EC and EC agent required approximately four person months of effort, from initial design to fully operational system. Because they are quite useable by an untrained user, in this section we will focus on the agent’s capabilities and on lessons learned in building it. We also will discuss our visions for agent explanations to expand the tasks that the user can perform with the agent.

To understand the EC and EC agent better, first we describe the function of the entire system. Figure 8 illustrates that in addition to the TV schedule display, windows are displayed in which the user's utterances to the agent and the agent's utterances to the user are presented. At appropriate times, windows appear for the display of the index of prerecorded programs, for a “help me” page, for showing in simulation the currently broadcast channel, and for a list of the multiple times a program could be presented, so that the user can disambiguate at which time a program is to be recorded. With the EC, the user does not use a keyboard or a mouse, but interacts with the agent solely by speaking.

4.1 A Subset Language for the EC agent

The user cannot communicate in free form language with the EC agent. Instead the user is constrained to use what we call a *subset* language, that is, a small, artificially constructed subset of a natural language (in our examples, English) in which the user can make requests and state constraints to the agent [2]. In Figure 8, the words in bold black at the top of the TV window present the utterances that the user can say to the agent when the user is navigating the TV schedule. For every different window display that a user sees, context sensitive utterance descriptions appear. These utterances teach new users what they can say and remind experienced users in the same way. These utterances also describe the tasks that users can perform for the current application state.

The subset of English assumed in most GUI interfaces is similar to the one designed for the EC agent. However, the language is designed to make its membership boundaries easily inferable, so that users can reliably predict the utterances that are members of the subset language. For spoken circumstances, such prediction is critical to ease of use under the constraint of real-time interaction. Because speech recognition of utterances is far from perfect, it is difficult for users to distinguish when they have mis-predicted the utterance's membership in the subset from when the recognition engine has failed (and the speech recognition subsystem provides little help in this regard). In such circumstances, users quickly become frustrated and cease interacting with speech systems. To avoid such frustrating circumstances for users, the subset language for the EC agent was designed to be simple, both syntactically and semantically. The EC agent grammar, lexicon and semantic interpretation were built at MERL; the grammar can be described by fourteen context free rules. The lexicon is medium in size, about 1100 items, of which 999 are TV show names, and twenty are TV channel names. There are no synonymous terms in the lexicons in order to eliminate the problem of the user predicting whether a word has a synonymous term.

Subset languages do not offer a natural, free flowing conversational style that is typical of human speech. A subset language interaction capability was chosen due to our experience with the other three Collagen interface agents. Those agents all made use of more complex English utterances. However, none of these agents had the property that a user could walk up and simply use the system. In those agents, users attempted to speak a wide variety of utterances that corresponded to what they wanted to do. The grammars for those agents covered some of the utterance types users would use, but not nearly enough of them to be widely usable. Furthermore, users could not easily be taught which utterances were allowable in the grammar because the grammar size, while not large, was large enough to preclude any type of short tutorial description. The EC agent was developed to function in conditions that were similar to the ones consumers find themselves in with new products. That is, consumers might have a user manual (which they might not look at or understand), but generally would not have someone to explain to them how to interact with the system.

The speech recognition system used with the EC agent is IBM Embedded Via Voice. It requires no training (and, in fact none is possible), and experience suggests that it understands user utterances at about 85% reliability for utterances that are correct grammatical strings for the EC agent grammar. For utterances that are not in the grammar, the EC agent does not make correct interpretations, but due to threshholding values of the recognition engine, it often says "I did not understand. Please repeat your utterance," in addition to presenting the utterance that it heard. Users are generally able to recover with this combination of information.

The Collagen team's experience with the EC agent suggests that we have progressed towards the goal of making the EC and agent usable for any speaker of English. Pilot user studies (12 subjects) show the users can use the subset created for the EC agent easily. They can perform a series of eight tasks in about eighteen minutes on average with no training on the system and no experience with speech systems. Many more users (about 40, several non-native speakers of

---

[2] Subset languages can be contrasted with sublanguages [12] because subset languages are artificially constructed from a parent natural language and not intended to be extended by human speakers of the parent language.

English) have talked with the EC agent after seeing a demo of the system. These users were all able to navigate the display, record and play back programs.

4.2 Explaining Complex Tasks to Users

Of the four basic tasks that users can undertake with the EC agent, none are as complex as the ones for email or scheduling. However, because means to perform these tasks are not already familiar to users, this small handful of tasks that every user needs to conduct their TV show viewing was chosen. Simple actions such as turning on and off the TV and changing the volume were not included since these are already easily undertaken with existing TV remotes.

There are more complex goals that can be undertaken by the agent, and we chose to keep these from being presented to new users. Complex tasks include those for searching the TV schedule, and comparing programs. For example, the task of comparing programs requires that the agent be told the names of the programs to be compared (any number of programs can be handled this way), after which it can provide the user with descriptions of the programs. A sample such interaction is shown in Figure 11. It is also natural for the agent to query users at the end of this activity about the user's desire to record one of those shows because recording is a likely reason for doing comparisons. This predictive behavior allows an agent to anticipate the user's desires and to shorten the conversation. The use of plan recognition and the recipe library in Collagen makes this predictive behavior possible.

User: Compare programs.
Agent: What is the name of the first program?
User: Star Trek
Agent: In this episode of Star Trek….
Agent: What is the name of the next program?
User: Angel.
Agent: In this episode of Angel….
Agent: What is the name of the next program?
User: No more programs.
Agent: Want to record one of these programs?
User: Yes. Angel.

Figure 11: A sample conversation to compare programs.

Actions such as compare programs have not yet been introduced to the EC. The delay results from a concern about how users learn about more complex capabilities in agent/user interactions. Most commercial applications ignore the issue of user learning and simply provide a wide array of features to users, who must then experiment to decide which features to use. Given the dizzying array of features, users find that this trial and error process takes a long time and is quite demanding. Our design of the EC and agent proceeds in a different way. The interface displays the essential actions necessary for basic activities by presenting their invocation commands on the screen (and also providing a "help me" page where the actions are briefly described).

How are more complex actions to be presented to users? Our proposed approach makes use of two concepts: keeping track of what the user already knows how to do, and explaining new functionality to users. The Collagen middleware is already able to undertake both these activities, in part through use of a student model, and in part by using explanation recipes.

Explanations of actions that can be undertaken have been central to our research on collaborative agents. For example, in the interaction of users with email agents, users were assumed to be experienced at email use. They were able, however, to ask for the agent's understanding of how to do email. In response, the agent would step them through its understanding of the task (see the dialogue in Figure 12, taken from a system run).

User: "What should I do?" [to read email]

Agent: "First you read a message."

User: "Ok."

Agent: "Then, you might store, print or delete the message."

User: "How do I read a message?"

Agent: Points to where to select message.

"First, you might select a message by clicking here."

User: "Ok."

Agent: Points to where to open a message.

"Next, you open the message by clicking here."

User: "Ok."

Agent: "Then you might use the contents of the message. Then you might react to the message."

User: "Ok."

Agent: Points to where to close a message.

Agent: "Finally , you close the message by clicking "Close" on pop-up menu here."

Points to where to close a message.

Figure 12: Explaining how to use email

Similarly with the VCR agent, the user could ask for an explanation of how to plug in a camcorder to the VCR. In response, the agent both explained the task and did simple sensing (in simulation but consistent with the capabilities of current VCRs) to check that the user had done what the agent suggested.

These attempts reflect experienced gained with PEdogogical COllagen agents [13], better known as PACO agents. PACO agents tutor students on procedural tasks, so they can learn to do the tasks for themselves. Among PACO agent capabilities are a student model [14], which is a model to track the actions a student has been told about and those actions the student has tried performing. PACO agent facilities in Collagen are available to the more assistant-style agents used in the four spoken collaborative agents. Using PACO style tutoring, a future EC agent could employ the student model to capture the tasks that the user has used in the interface so far, before deciding it was time to introduce more complex tasks. Then the future EC agent would use an explanation recipe for comparing programs to explain that task to the user. While a student model is not used with the current EC agent, this general Collagen capability offers significant new help for users in future versions of the EC agent.

In summary, with the EC agent we have explored several aspects of user interaction with a collaborative agent. The EC agent collaborates with a user in more limited ways than any of the previous agents. It allows the user to manage EC tasks by seeking information it needs to undertake tasks that the user specifies. To make the EC and EC agent useable by a wide range of users, we developed a highly constrained but usable spoken language, though the use of subset languages. To keep the interface tasks understandable to users, because the EC and EC agent are a new type of user interface, we provided limited but essential tasks in the interface for the user and agent to perform. To help a user learn about additional tasks that are available in the interface, we discussed how to use the tracking and explanation capabilities of Collagen for additional user tasks.

5. RELATED WORK

The field of conversational speech interfaces has grown enormously in the past few years (see [15]. Rather than a comprehensive treatment of all such works, a few relevant systems will be reviewed here. The most sophisticated conversational speech interface developed to date is the VERBMOBIL system [16]. This speech to speech conversational system translated each user's input to a conversation in the languages of German, Japanese and English. The discussion domain was travel matters, hotel reservations and ticketing. While VERBMOBIL did not use an agent that must contribute to a conversation, its overall dialogue capabilities were quite rich. The dialog management subcomponent generated dialog summaries, and it classified all utterances according to their utterance type before proceeding to speech translation.

Glass and Weinstein[17] report on SpeechBuilder, a facility for quickly developing spoken dialogue systems. The architecture relied on the Galaxy Hub architecture [18] and the types of utterances available were restricted to those that are semantically related to case frame filling. SpeechBuilder used the dialogue manager of the Galaxy architecture, which was limited to a dialogue state variable rather than the more general models used in VERBMOBIL, Collagen, Artimis [19] or Allen's TRIPS and TRAINS systems.

Allen's recent TRIPS system [20], a follow-on to the TRAINS system, focused on producing and interpreting spoken language in an incremental fashion. This system aimed to interpret utterances that were partially complete, contained pauses, were grounded by acknowledgements such as "mm-mm." Like the TRAINS system, TRIPS had a rich model of dialogue, separated domain and dialogue reasoning and modelled turn taking explicitly.

6. A RECAP OF LESSONS LEARNED

Collaboration is essential to spoken language conversational systems. Users converse with computer applications in order to accomplish tasks with those applications. Without collaboration, the user is forced to manage the entire task himself, and must be very literal in what he asks the computer to do for him. With collaboration, the user can delegate parts of his tasks for the agent to perform, and respond to agent requests for information it needs to accomplish the task on the user's behalf.

In this article, we have reported on our experience with four collaborative interface agents for GUI applications. All these agents make use of their knowledge of the conversational structure, the intentions of the user and the relations between the two to perform tasks in service of the user's goals. Our agents are most effective in collaboration when the application provides significant sub-tasks that the agent can undertake on behalf of the user. Even in those cases where the sub-tasks are simple, the ability of the agent to decide itself what information it needs, reduces the user burden to manage all the details of the actions that contribute to the user's goals.

Human collaborations are far richer than those that can be undertaken with computer agents, in part because people know and can do more than our computer systems. Equally important, human collaborations involve richer conversations than users can yet have with their collaborative agents due to the limitations of speech technology. Our experience with collaborative interface agents indicates that for GUI applications, spoken conversations are effective and can be available to many users by employing a subset language of a natural language.

7. REFERENCES

1. B. J. Grosz and C. L. Sidner. "Attention, intentions, and the structure of discourse." *Computational Linguistics*, 12(3):175--204, 1986.
2. B.J. Grosz and C. L. Sidner. "Plans for discourse," in *Intentions and Plans in Communication and Discourse*. P. Cohen, J. Morgan, and M.Pollack (eds.), MIT Press, 1990.
3. B. J. Grosz and S. Kraus. "Collaborative Plans for Complex Group Action," *Artificial Intelligence*, 86(2): 269-357, 1996.
4. C. Rich and C. L. Sidner. "COLLAGEN: A Collaboration Manager for Software Interface Agents," *User Modeling and User-Adapted Interaction*, Vol. 8, No. 3/4: 315-350, 1998.
5. K. E. Lochbaum "A Collaborative Planning Model of Intentional Structure," *Computational Linguistics*, 24(4): 525-572, 1998.
6. P. Cohen, R. Perrault and J. Allen. "Beyond Question Answering," in *Strategies for Natural Language Processing*. W. Lehnert and M. Ringle (eds.). Lawrence Erlbaum Associates, Hillsdale, NJ. pp. 245-274, 1982.
7. C. Rich, C. L. Sidner and N. Lesh, "COLLAGEN: Applying Collaborative Discourse Theory to Human-Computer Interaction," *AI Magazine*, *Special Issue on Intelligent User Interfaces*, AAAI Press, Menlo Park, CA, Vol. 22: 4: 15-25, 2001.
8. N. Lesh, C. Rich and C. L. Sidner. "Using plan recognition in human-computer collaboration," in *Proceedings of the 7th Int. Conference on User Modeling*, pp 23--32, 1999.
9. N. Lesh, C. Rich, and C.L. Sidner. "Collaborating with Focused and Unfocused Users," in *Proceedings of the 8th International Conference on User Modeling*, Springer Verlag, New York, pp. 64-73, 2001.
10. C. Sidner. "An Artificial Discourse Language for Collaborative Negotiation," in *Proceedings of the Twelfth National Conference on Artificial Intelligence*, MIT Press, Cambridge, MA, Vol.1: 814-819, 1994.
11. C.L. Sidner, C. Boettner and C. Rich. "Building Spoken Language Collaborative Interface Agents," Lotus Technical Report TR2001-01, Lotus Development Corporation, 2001.
12. R. Kittredge. "Variation and Homogeneity of Sublanguages," in *Sublanguage: Studies of Language in Restricted Semantic Domains*, R. Kittredge and J. Lehrberger (eds.), Walter de Gruyter, New York, 1982.
13. J. Rickel, N. Lesh, C. Rich, C. L. Sidner, and A. Gertner, "Collaborative Discourse Theory as a Foundation for Tutorial Dialogue," *Proceedings of Sixth International Conference on Intelligent Tutoring Systems*, pp. 542-551, Springer Verlag, New York, 2002.
14. I. Goldstein. "Overlays: A theory of Modelling for Computer-Aided Instruction," Artificial Intelligence Laboratory Memo 495, MIT, Cambridge, MA, 1977.
15. C. Sidner (ed.). *ANLP/NAACL 2000 Workshop on Conversational Systems*, Association for Computational Linguistics, New Brunswick, N.J., 2000.
16. W. Wahlster (ed.) *Verbmobil: Foundations of Speech-To-Speech Translation*. Springer Verlag, New York, 2000.
17. J. Glass and E. Weinstein. "SpeechBuilder: Facilitating Spoken Dialogue System Development," in *Proceedings of Eurospeech 2001*, Alborg, Denmark, September, 2001.
18. S. Seneff, R. Lau and J. Polifroni. "Organization, Communication and Control in the Galaxy-II Conversational System," in *Proceedings of Eurospeech 1999*, Budapest, 1999.
19. M. D. Sadek, P. Bretier and F. Panaget. "ARTIMIS: Natural dialogue meets rational agency," in *Proceedings of IJCAI-97*, Nagoya, Japan, Morgan Kaufman Publishers, San Francisco, pp. 1030-1035, 1997.

20. J. Allen, G. Ferguson and A. Stent. "An Architecture for More Realistic Conversational Systems," in *Proceedings of the 2001 Conference on Intelligent User Interfaces*, ACM Press, New York, pp. 1-8, 2001.