

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

**COLLAGEN:
Applying Collaborative Discourse Theory
to Human-Computer Interaction**

Charles Rich, Candace L. Sidner, Neal Lesh

TR-2000-38 November 2000

Abstract

We describe an approach to intelligent user interfaces based on the idea of making the computer a collaborator, and an application-independent technology for implementing such interfaces.

To appear in AI Magazine, Special Issue on Intelligent User Interfaces, 2001.

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Mitsubishi Electric Research Laboratories, Inc.; an acknowledgment of the authors and individual contributions to the work; and all applicable portions of the copyright notice. Copying, reproduction, or republishing for any other purpose shall require a license with payment of fee to Mitsubishi Electric Research Laboratories, Inc. All rights reserved.

Copyright © Mitsubishi Electric Research Laboratories, Inc., 2000
201 Broadway, Cambridge, Massachusetts 02139

Submitted in December 2000.

COLLAGEN:

Applying Collaborative Discourse Theory to Human-Computer Interaction

Charles Rich, Candace L. Sidner and Neal Lesh

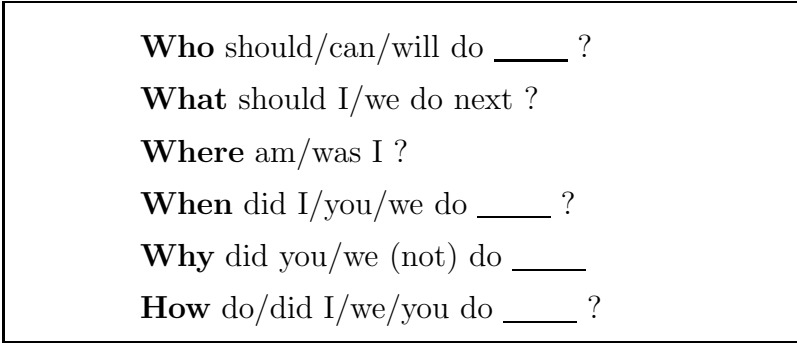
- We describe an approach to intelligent user interfaces based on the idea of making the computer a collaborator, and an application-independent technology for implementing such interfaces.

What properties of a user interface would make you want to call it *intelligent*? For us, any interface that is called intelligent should at least be able to answer the six types of questions from users shown in Figure 1. Being able to ask and answer these kinds of questions implies a flexible and adaptable division of labor between the human and the computer in the interaction process. Unlike most current interfaces, an intelligent user interface should be able to guide and support you when you make a mistake or if you don't know how to use the system well.

What we are suggesting here is a paradigm shift. As an analogy, consider the introduction of the *undo* button. This one button fundamentally changed the experience of using interactive systems by removing the fear of making accidental mistakes. Users today expect every interactive system to have an undo button and are justifiably annoyed when they can't find it. By analogy, to focus on just one of the question types in Figure 1, what we are saying is that every user interface should have a "What should I do next?" button.

Note that we are not saying that each of the questions in Figure 1 must literally be a separate button. The mechanisms for asking and answering these questions could be spoken or typed natural (or artificial) language, adaptive menus, simple buttons, or some combination of these. We have experimented with all of these mechanisms in the various prototype systems described later.

Finally, some readers may object that answering the question types in Figure 1 should be thought of as a function of the "application" rather than the "interface." Rather



Who should/can/will do ____ ?
What should I/we do next ?
Where am/was I ?
When did I/you/we do ____ ?
Why did you/we (not) do ____
How do/did I/we/you do ____ ?

Figure 1. Six Questions for an Intelligent Interface.

(Adapted from the news reporter's "five W's.") The blanks are filled in with application-specific terms, ranging from high-level goals, such as "prepare a market survey" or "shut down the power plant," to primitive actions, such as "underline this word" or "close valve 17."

than getting into a unproductive semantic argument about the boundary between these two terms, we prefer instead to focus on what we believe is the real issue, namely, whether this characterization of intelligent user interfaces can lead to the development of a reusable middleware layer that makes it easy to incorporate these capabilities into diverse systems.

Again, there is a relevant historical analogy. A key to the success of so-called WIMP (Windows, Icons, Menus and Pointers) interfaces has been the development of widely used middleware packages, such as Motif™ and Swing™. These middleware packages embody generally useful graphical presentation and interaction conventions, such as tool bars, scroll bars, check boxes, and so on. We believe that the next goal in user interface middleware should be to codify techniques for supporting communication about users' task structure and process, as suggested by the question types in Figure 1. This article describes a system, called Collagen™, which is the first step in this direction.

Collaboration

So what does all of this have to do with the “collaborative discourse theory” in the title of this article? The goal of developing generic support for communicating about the user’s task structure cannot, we feel, be achieved by taking an engineering approach focused directly on the questions in Figure 1. We therefore started this research by looking for an appropriate theoretical foundation, which we found in the concept of collaboration.

Collaboration is a process in which two or more participants coordinate their actions toward achieving shared goals. Most collaboration between humans involves communication. *Discourse* is a technical term for an extended communication between two or more participants in a shared context, such as a collaboration. “Collaborative discourse theory” (see theory sidebar) thus refers to a body of empirical and computational research about how people collaborate. Essentially, what we have done in this project is apply a theory of human-human interaction to human-computer interaction.

In particular, we have taken the approach of adding a *collaborative interface agent* (see Figure 2) to a conventional direct-manipulation graphical user interface. The name of our software system, Collagen (for *Collaborative agent*), derives from this approach. (Collagen is also a fibrous protein that is the chief constituent of connective tissue in vertebrates.)

The interface agent approach mimics the relationships that typically hold when two humans collaborate on a task involving a shared artifact, such as two mechanics working on a car engine together or two computer users working on a spreadsheet together.

In a sense, our approach is a very literal-minded way of applying collaborative discourse theory to human-computer interaction. We have simply substituted a software agent for one of the two humans that would appear in Figure 2 if it were a picture of human-human collaboration. There may be other ways of applying the same theory without introducing the concept of an agent as separate from the application (Ortiz & Grosz 2000), but we have not pursued those research directions.

Notice that the software agent in Figure 2 is able both to communicate with and observe the actions of the user and vice versa. Among other things, collaboration requires knowing when a particular action has been done. In Collagen, this can occur two ways: either by a reporting communication (“I have done x ”)

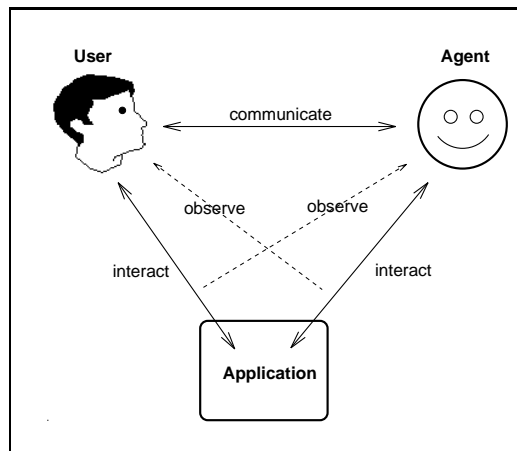


Figure 2. Collaborative Interface Agent.

or by direct observation. Another symmetrical aspect of the figure is that both the user and the agent can interact with the application program.

There are many complex engineering issues regarding implementing the observation and interaction arrows in Figure 2, which are beyond the scope of this article (Lieberman 1998). In all our prototypes, the application program has provided a program interface (API) for performing and reporting primitive actions and for querying the application state. Communication between the user and the agent has been variously implemented using speech recognition and generation, text and menus.

Outline of the Article

The remainder of this article lays out the work we have done in more detail, starting with snapshots of four interface agents built by us and our collaborators in different domains using Collagen. Following these examples comes a description of the technical heart of our system, which is the representation of discourse state and the algorithm for updating it as an interaction progresses. Next, we discuss another key technical contribution, namely how Collagen uses plan recognition in a collaborative setting. Finally, we present the overall system architecture of Collagen, emphasizing the application-specific versus application-independent components. We conclude with a brief discussion of related and future work.

Application Examples

This section briefly describes four interface agents built by us and our collaborators in

Theory

In 1986, Grosz and Sidner proposed a tripartite framework for modelling task-oriented discourse structure. The first (*intentional*) component records the beliefs and intentions of the discourse participants regarding the tasks and subtasks (“purposes”) to be performed. The second (*attentional*) component captures the changing focus of attention in a discourse using a stack of “focus spaces” organized around the discourse purposes. As a discourse progresses, focus spaces are pushed onto and popped off of this stack. The third (*linguistic*) component consists of the contiguous sequences of utter-

ances, called “segments,” which contribute to a particular purpose.

Grosz and Sidner extended this basic framework in 1990 with the introduction of SharedPlans, which are a formalization of the collaborative aspects of a conversation. The Shared-Plan formalism models how intentions and mutual beliefs about shared goals accumulate during a collaboration. Grosz and Kraus provided a comprehensive axiomatization of SharedPlans in 1996, including extending it to groups of collaborators.

Most recently, Lochbaum developed an algorithm for discourse interpretation using SharedPlans and the tripartite model of discourse. This al-

gorithm predicts how conversants follow the flow of a conversation based on their understanding of each other’s intentions and beliefs.

References

- Grosz, B. J., and Sidner, C. L. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics* 12(3):175–204.
- Grosz, B. J., and Sidner, C. L. 1990. Plans for discourse. In Cohen, P. R.; Morgan, J. L.; and Pollack, M. E., eds., *Intentions and Communication*. Cambridge, MA: MIT Press. 417–444.
- Grosz, B. J., and Kraus, S. 1996. Collaborative plans for complex group action. *Artificial Intelligence* 86(2):269–357.
- Lochbaum, K. E. 1998. A collaborative planning model of intentional structure. *Computational Linguistics* 24(4).

four different application domains using Collagen. We have also built agents for air travel planning (Rich & Sidner 1998) and email (Gruen *et al.* 1999). All of these agents are currently research prototypes.

Figure 3 shows a screen image and sample interaction for each example agent. Instances of the questions types in Figure 1 are underlined in the sample interactions

Each screen image in Figure 3 consists of a large application-specific window, which both the user and agent can use to manipulate the application state and two smaller windows, labelled “Agent” and “User,” which are used for communication between the user and agent. Two of the example agents (3a and 3d) communicate using speech recognition and generation technology; the other two allow the user to construct utterances using hierarchical menus dynamically generated based on the current collaboration state.

The agent in Figure 3a helps a user set up and program a video cassette recorder (VCR). The image in the figure, which a real user would see on her television screen, includes the VCR itself so that the agent can point at parts of the VCR during explanations (see line 12). In the first part of the VCR agent transcript, the agent helps the user eliminate the annoying blinking “12:00” that is so common on VCR clocks. Later on, the agent walks the user through the task of connecting a camcorder to the VCR.

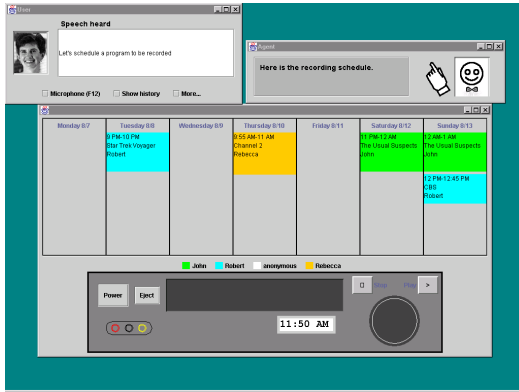
The agent in Figure 3b was developed in collaboration with the Industrial Electronics and Systems Laboratory of Mitsubishi Electric in Japan. The application program in this case is a sophisticated graphical interface development tool, called the Symbol Editor. Like many such tools, the Symbol Editor is difficult for novice users because there

are too many possible things to do at any moment and the system itself gives no guidance regarding what to do next. Our agent guides a user through the process of achieving a typical task using the Symbol Editor, automatically performing many of the tedious subtasks along the way.

The agent in Figure 3c was developed in collaboration with the Information Sciences Institute of the University of Southern California (Rickel *et al.* 2001). This agent teaches a student user how to operate a gas turbine engine and generator configuration using a simple software simulation. The first time the agent teaches a new task or subtask, it walks the student through all the required steps. If a task has already been performed once, however, the agent tells the student to “take it from here” (line 3). If the student later asks for help (line 5), the agent will describe just the next step to be performed.

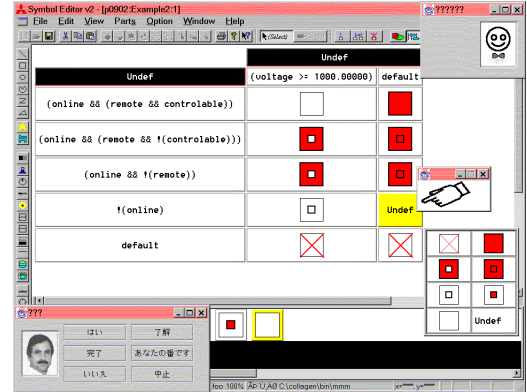
The gas turbine agent is part of a larger effort, which also involves the MITRE Corporation (Gertner *et al.* 2000), to incorporate application-independent tutorial strategies into Collagen. Teaching and assisting are best thought of as points on a spectrum of collaboration (Davies *et al.* 2001), rather than as separate capabilities.

Finally, Figure 3d shows an agent being developed at the Delft University of Technology to help people program a home thermostat (Keyson *et al.* 2000). The transcript here illustrates only a very simple interaction with the agent. This agent will eventually be able to help people analyze their behavior patterns and construct complicated heating and cooling schedules to conserve energy. This work is part of a larger research project at Delft to add intelligence to products.



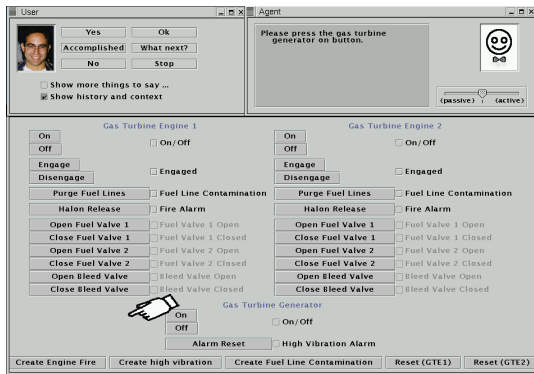
- 1 User says "Hello."
- 2 Agent says "Hello, what's your name?"
- 3 User says "Candy."
- 4 Agent says "Hello, Candy!"
- 5 User says "Let's set the clock."
- 6 Agent says "What is the time?"
- 7 User says "The time is 11:00 AM."
- 8 Agent sets the clock to 11:00 AM.
- ...
- 9 User says "How do I play back a video from my camcorder?"
- 10 Agent says "First, plug in the camcorder cables."
- 11 User says "How do I plug in the camcorder cables?"
- 12 Agent points to where you plug in the camcorder cables.
- 13 Agent says "The red cable plugs into the red terminal, the black cable plugs into the black terminal, and the yellow cable plugs into the yellow terminal. Make sure the other ends of the cables plug into the corresponding colors on the camcorder."
- 14 User says "What next?"
- 15 Agent says "Now press Play on your camcorder. You should see a picture on the TV screen."
- 16 User says "Ok."
- 17 Agent says "I don't see a picture. Make sure the camcorder is plugged in or has fresh batteries, and is turned on."
- 18 User says "Ok."
- 19 Agent says "Good. It's working."

(a) VCR Agent.



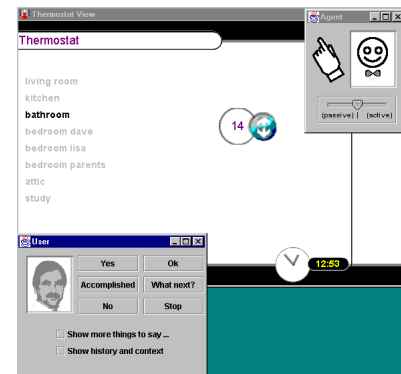
- 1 Agent says "Please list the other (non-boolean) attributes used in the columns of the logic table."
- 2 User says "The attributes are: voltage."
- 3 Agent creates a new user-defined attribute.
- 4 Agent enters "voltage" as the name of the new attribute.
- 5 Agent says "Please select the data type of the new attribute."
- 6 User selects float as the data type of the new attribute.
- 7 User clicks ok to accept the definition of voltage.
- 8 Agent says "Please edit the initial setting for the voltage attribute (if necessary) and then click ok."
- 9 User clicks ok to accept 0.0 as the initial setting for voltage.
- 10 Agent shows the expression editor for column 1.
- 11 User enters (voltage > 0.0) as the title for column 1.
- 12 Agent says "Please define an expression for column 1." ...[user goes to lunch]...
- 14 User says "Where are we?"
- 15 Agent says "You are defining an expression for column 1, as part of making a logic table."
- 16 User enters (voltage >= 1000.0) as the title for column 1.
- 17 Agent shows the logical operator window.
- 18 Agent selects the >= operator.
- 19 Agent drags and drops >= into expression for column 1.
- 20 Agent selects the voltage attribute.
- 21 Agent drags and drops voltage into expression for column 1.
- 22 Agent types "1000.0" into expression for column 1.

(b) Symbol Editor Agent.



- 1 Agent says "Let's start the generator."
- 2 User says "Ok."
- 3 Agent says "You take it from here."
- 4 User presses the alarm reset button.
- 5 User says "What next?"
- 6 Agent says "Please press the gas turbine generator on button."
- 7 Agent points to the gas turbine generator on button.
- 8 User presses the gas turbine generator button.
- 9 Agent says "Good. We have successfully started the generator."

(c) Gas Turbine Agent.



- 1 Agent says "Let's set the temperature in a room."
- 2 User selects the bathroom.
- 3 User says "What next?"
- 4 Agent says "What should the temperature setting be?"
- 5 User says "18 degrees Centigrade."
- 6 Agent sets the temperature in the bedroom to 18 degrees.

(d) Thermostat Agent.

Figure 3. Example Agents Built with Collagen.

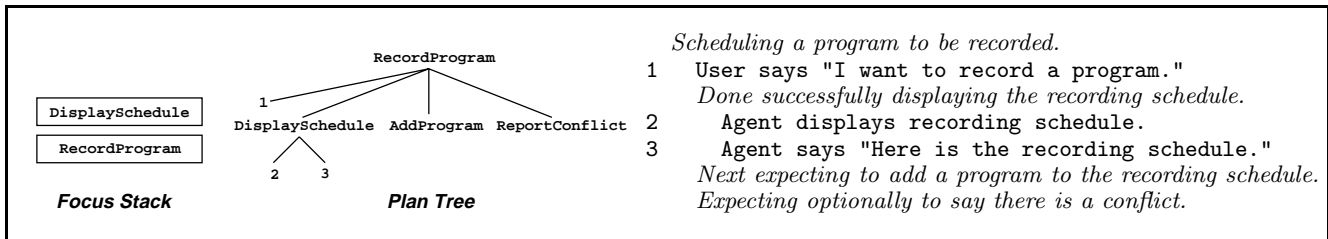


Figure 4. Example Discourse State and Segmented Interaction History for VCR Agent.

Discourse State

Participants in a collaboration derive benefit by pooling their talents and resources to achieve common goals. However, collaboration also has its costs. When people collaborate, they must usually communicate and expend mental effort to ensure that their actions are coordinated. In particular, each participant must maintain some sort of mental model of the status of the collaborative tasks and the conversation about them—we call this model the *discourse state*.

Among other things, the discourse state tracks the beliefs and intentions of all the participants in a collaboration and provides a focus of attention mechanism for tracking shifts in the task and conversational context. All of this information is used by an individual to help understand how the actions and utterances of the other participants contribute to the common goals.

In order to turn a computer agent into a collaborator, we needed a formal representation of discourse state and an algorithm for updating it. The discourse state representation currently used in Collagen, illustrated in Figure 4, is a partial implementation of Grosz & Sidner’s theory of collaborative discourse (see theory sidebar); the update algorithm is described in the next section.

Collagen’s discourse state consists of a stack of goals, called the *focus stack* (which will soon become a stack of focus spaces to better correspond with the theory), and a *plan tree* for each goal on the stack. The top goal on the focus stack is the “current purpose” of the discourse. A plan tree in Collagen is an (incomplete) encoding of a partial SharedPlan between the user and the agent. For example, Figure 4 shows the focus stack and plan tree immediately following the discourse events numbered 1–3 on the right side of the figure.

Segmented Interaction History

The annotated, indented execution trace on the right side of Figure 4, called a *segmented interaction history*, is a compact, textual rep-

resentation of the past, present and future states of the discourse. We originally developed this representation to help us debug agents and Collagen itself, but we have also experimented with using it to help users visualize what is going in a collaboration (see discussion of “history-based transformations” in (Rich & Sidner 1998))

The numbered lines in a segmented interaction history are simply a log of the agent’s and user’s utterances and primitive actions. The italic lines and indentation reflect Collagen’s interpretation of these events. Specifically, each level of indentation defines a segment (see theory sidebar) whose purpose is specified by the italicized line that precedes it. For example, the purpose of the toplevel segment in Figure 4 is *scheduling a program to be recorded*.

Unachieved purposes that are currently on the focus stack are annotated using the present tense, such as *scheduling*, whereas completed purposes use the past tense, such as *done*. (Note in Figure 4 that a goal is not popped off the stack as soon as it is completed, because it may continue to be the topic of conversation, for example, to discuss whether it was successful.)

Finally, the italic lines at the end of each segment, which include the keyword *expecting*, indicate the steps in the current plan for the segment’s purpose which have not yet been executed. The steps which are “live” with respect to the plan’s ordering constraints and preconditions have the added keyword *next*.

Discourse Interpretation

Collagen updates its discourse state after every utterance or primitive action by the user or agent using Lochbaum’s discourse interpretation algorithm (see theory sidebar) with extensions to include plan recognition (see next section) and unexpected focus shifts (Lesh *et al.* 2001).

According to Lochbaum, each discourse event is explained as either: (i) starting a new segment whose purpose contributes to

the current purpose (and thus pushing a new purpose on the focus stack), (ii) continuing the current segment by contributing to the current purpose, or (iii) completing the current purpose (and thus eventually popping the focus stack).

An utterance or action contributes to a purpose if it either: (i) directly achieves the purpose, (ii) is a step in a recipe for achieving the purpose, (iii) identifies the recipe to be used to achieve the purpose, (iv) identifies who should perform the purpose or a step in the recipe, or (v) identifies a parameter of the purpose or a step in the recipe. These last three conditions are what Lochbaum calls “knowledge preconditions.”

A *recipe* is a goal-decomposition method (part of a task model). Collagen’s recipe definition language (see Figure 5) supports partially ordered steps, parameters, constraints, pre- and post-conditions, and alternative goal decompositions.

Our implementation of the discourse interpretation algorithm above requires utter-

```
public recipe RecordRecipe achieves RecordProgram {
  step DisplaySchedule display;
  step AddProgram add;
  optional step ReportConflict report;
  constraints {
    display precedes add;
    add precedes report;
    add.program == achieves.program;
    report.program == achieves.program;
    report.conflict == add.conflict;
  }
}
```

Figure 5. Example Recipe in VCR Task Model.

Definition of the recipe used in Figure 4 to decompose the non-primitive `RecordProgram` goal into primitive and non-primitive steps. Collagen task models are defined in an extension of the Java language which is automatically processed to create Java class definitions for recipes and act types.

Scheduling a program to be recorded.

- 1 User says "I want to record a program."
Done successfully displaying the recording schedule.
- 2 Agent displays recording schedule.
- 3 Agent says "Here is the recording schedule."
- 4 User says "Ok."
Done identifying the program to be recorded.
- 5 Agent says "What is the program to be recorded?"
- 6 User says "Record 'The X-Files'."
*Next expecting to add a program to the recording schedule.
Expecting optionally to say there is a conflict.*

Figure 6. Continuing the Interaction in Figure 4.

ances to be represented in Sidner’s (1994) artificial discourse language. For our speech-based agents, we have used standard natural language processing techniques to compute this representation from the user’s spoken input. Our menu-based systems construct utterances in the artificial discourse language directly.

Discourse Generation

To illustrate how Collagen’s discourse state is used to generate as well as interpret discourse behavior, we briefly describe below how the VCR agent produces the underlined utterance on line 5 in Figure 6, which continues the interaction in Figure 4.

The discourse generation algorithm in Collagen is essentially the inverse of discourse interpretation. Based on the current discourse state, it produces a prioritized list, called the *agenda*, of (partially or totally specified) utterances and actions which would contribute to the current discourse purpose according to cases (i) through (v) above. For example, for the discourse state in Figure 4, the first item on the agenda is an utterance asking for the identity of the program parameter of the `AddProgram` step of the plan for `RecordProgram`.

In general, an agent may use any application-specific logic it wants to decide on its next action or utterance. In most cases, however, an agent can simply execute the first item on the agenda generated by Collagen, which is what the VCR agent does in this example. This utterance starts a new segment, which is then completed by the user’s answer on line 6.

Plan Recognition

Plan recognition (Kautz & Allen 1986) is the process of inferring intentions from actions. Plan recognition has often been proposed for improving user interfaces or to facilitate intelligent help features. Typically, the computer watches “over the shoulder” of the user and jumps in with advice or assistance when it thinks it has enough information.

In contrast, our main motivation for adding plan recognition to Collagen was to reduce the amount of communication required to maintain a mutual understanding between the user and the agent of their shared plans in a collaborative setting (Lesh *et al.* 1999). Without plan recognition, Collagen’s discourse interpretation algorithm onerously required the user to announce each

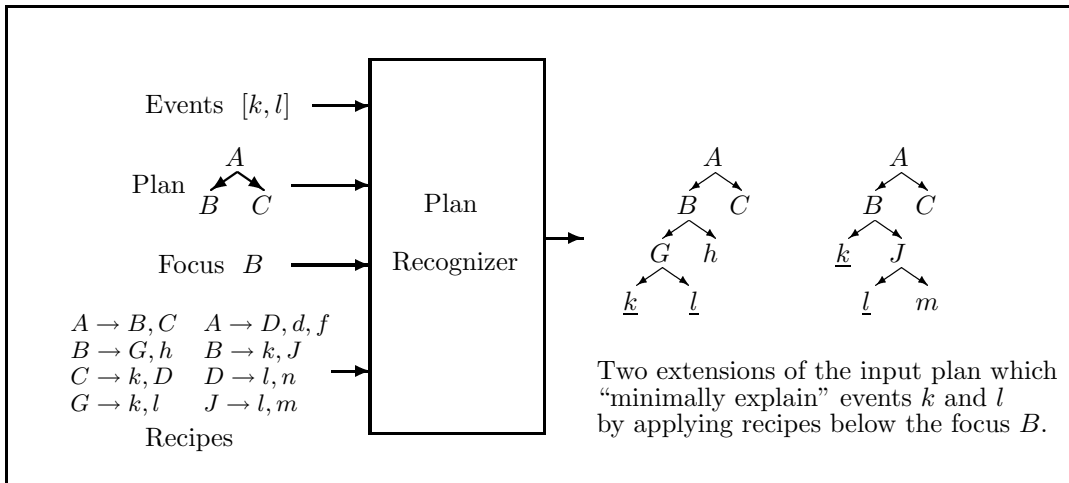


Figure 7. Example Plan Recognizer Inputs and Outputs in a Collaborative Setting.

goal before performing a primitive action which contributed to it.

Although plan recognition is a well-known feature of human collaboration, it has proven difficult to incorporate into practical computer systems due to its inherent intractability in the general case. We exploit three properties of the collaborative setting in order to make our use of plan recognition tractable. The first property is the focus of attention, which limits the search required for possible plans.

The second property of collaboration we exploit is the interleaving of developing, communicating about and executing plans, which means that our plan recognizer typically operates only on partially elaborated hierarchical plans. Unlike the “classical” definition of plan recognition, which requires reasoning over complete and correct plans, our recognizer is only required to incrementally extend a given plan.

Third, it is quite natural in the context of a collaboration to ask for clarification, either because of inherent ambiguity, or simply because the computation required to understand an action is beyond a participant’s abilities. We use clarification to ensure that the number of actions the plan recognizer must interpret will always be small.

Figure 7 illustrates roughly how plan recognition works in Collagen. Suppose the user performs action k . Given the root plan (e.g., A) for the current discourse purpose (e.g., B) and a set of recipes, the plan recognizer determines the set of minimal extensions to the plan which are consistent with the recipes and include the user performing k . If there is exactly one such extension, the extended plan becomes part

of the new discourse state. If there is more than one possible extension, action k is held and reinterpreted along with the next event, which may disambiguate the interpretation (which l does not), and so on. The next event may in fact be a clarification.

Our algorithm also computes essentially the same recognition if the user does not actually perform an action, but only proposes it, as in, “Let’s achieve G .” Another important, but subtle, point is that Collagen applies plan recognition to both user and agent utterances and actions in order to correctly maintain a model of what is mutually believed.

System Architecture

Figure 8 summarizes the technical portion of this article by showing how all the pieces described earlier fit together in the architecture of collaborative system built with Collagen. This figure is essentially an expansion of Figure 2, showing how Collagen mediates the interaction between the user and the agent. Collagen is implemented using Java Beans™, which makes it easy to modify and extend this architecture.

The best way to understand the basic execution cycle in Figure 8 is to start with the arrival of an utterance or an observed action (from either the user or the agent) at the discourse interpretation module at the top center of the diagram. The discourse interpretation algorithm (including plan recognition) updates the discourse state as described above, which then causes a new agenda to be computed by the discourse generation module. In the simplest case, the agent responds by selecting and executing an entry in the

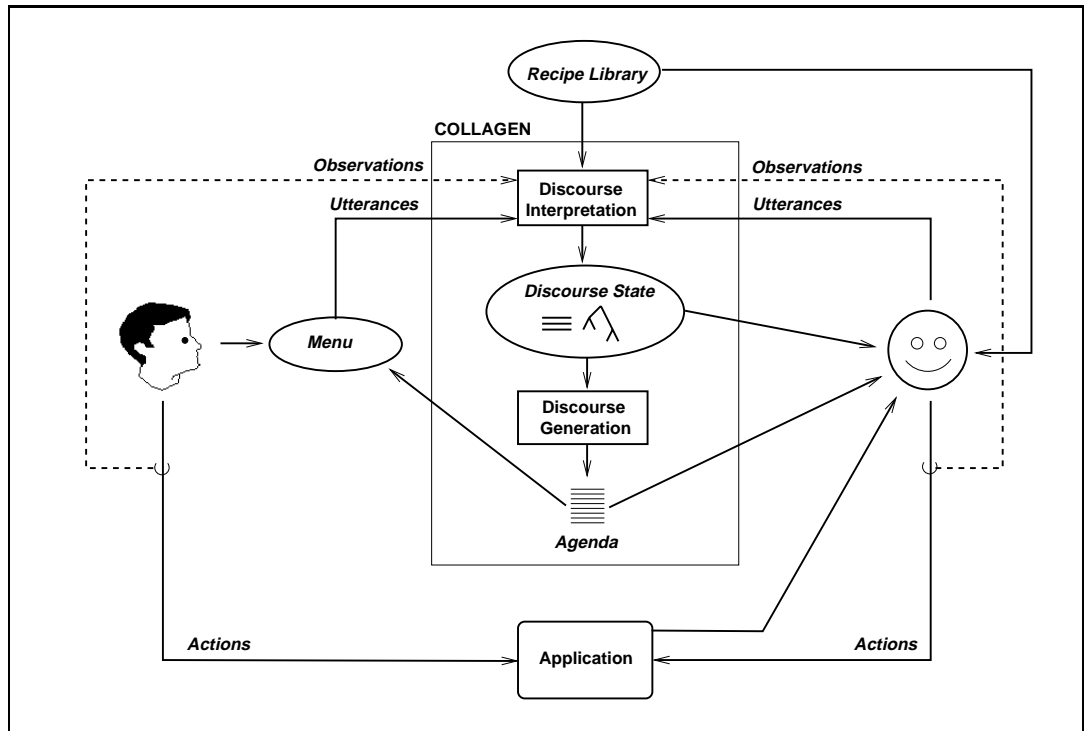


Figure 8. Architecture of a Collaborative System Built with Collagen

new agenda (which may be either an utterance or an action), which provides new input to discourse interpretation.

In a system without natural language understanding, a subset of the agenda is also presented to the user in the form of a menu of customizable utterances. In effect, this is a way of using expectations generated by the collaborative context to replace natural language understanding. Because this is a mixed-initiative architecture, the user can, at any time, produce an utterance (e.g., by selecting from this menu) or perform an application action (e.g., by clicking on an icon), which provides new input to discourse interpretation.

In the simple story above, the only application-specific components an agent developer needs to provide are the recipe library and an API through which application actions can be performed and observed (for an application-independent approach to this API, see (Cheikes *et al.* 1999)). Given these components, Collagen is a turnkey technology—default implementations are provided for all the other needed components and graphical interfaces, including a default agent which always selects the first item on the agenda.

In each of the four example applications in Figure 3, however, a small amount (e.g., sev-

eral pages) of additional application-specific code was required in order to achieve the desired agent behavior. As the arrows incoming to the agent in Figure 8 indicate, this application-specific agent code typically queries the application and discourse states and (less often) the recipe library. An agent developer is free, of course, to employ arbitrarily complex application-specific and generic techniques, such as a theorem proving, first-principles planning, etc., to determine the agent's response to a given situation.

Related Work

This work lies at the intersection of many threads of related research in artificial intelligence, computational linguistics, and user interface. We believe it is unique, however, in its combination of theoretical elements and implemented technology. Other theoretical models of collaboration (Levesque *et al.* 1990) do not integrate the intentional, attentional and linguistic aspects of collaborative discourse, as SharedPlan theory does. On the other hand, our incomplete implementation of SharedPlan theory in Collagen does not deal with the many significant issues in a collaborative system with more than two participants (Tambe 1997).

There has been much related work on implementing collaborative dialogues in

the context of specific applications, based either on discourse planning techniques (Chu-Carroll & Carberry 1995; Ahn *et al.* 1995; Allen *et al.* 1996; Stein *et al.* 1999) or rational agency with principles of cooperation (Sadek & De Mori 1997). None of these research efforts, however, have produced software that is reusable to the same degree as Collagen. In terms of reuseability across domains, a notable exception is the Verbmobil project (Verbmobil 2000), which concentrates on linguistic issues in discourse processing, without an explicit model of collaboration.

Finally, a wide range of interface agents (Maes 1994) continue to be developed, which have some linguistic and collaborative capabilities, without any general underlying theoretical foundation.

Future Work

We are currently extending our plan recognition algorithm to automatically detect certain classes of so-called “near miss” errors, such as performing an action out of order, or performing the right action with the wrong parameter. The basic idea is that, if there is no explanation for an action given the “correct” recipe library, the plan recognizer should incrementally relax the constraints on recipes before giving up and declaring the action to be unrelated to the current activity. This functionality is particularly useful in tutoring applications of Collagen.

We have, in fact, recently become increasingly interested in tutoring and training applications (Gertner *et al.* 2000; Rickel *et al.* 2001), which has revealed some implicit biases in how Collagen currently operates. For example, Collagen’s default agent will always, if possible, itself perform the next action that contributes to the current goal. This is not, however, always appropriate in a tutoring situation, where the real goal is not to get the task done, but for the student to learn how to do the task. We are therefore exploring various generalizations and extensions to Collagen to better support the full spectrum of collaboration (Davies *et al.* 2001), such as creating explicit tutorial goals and recipes, and recipes that encode “worked examples.”

We are also working on two substantial extensions to the theory underlying Collagen. First, we are adding an element to the attentional component (see theory sidebar) to track which participant is currently in control of the conversation. The basic idea is that, when a segment is completed, the de-

fault control (initiative) goes to the participant who initiated the enclosing segment.

Second, we are beginning to codify the negotiation strategies used in collaborative discourse. These strategies are different from the negotiation strategies used in disputations (Kraus *et al.* 1998). For example, when the user rejects an agent’s proposal (or vice versa), the agent and user should be able to enter into a sub-dialogue in which their respective reasons for and against the proposal are discussed.

References

- Ahn, R., et al. 1995. The DenK-architecture: A fundamental approach to user-interfaces. *Artificial Intelligence Review* 8:431–445.
- Allen, J., et al. 1996. A robust system for natural spoken dialogue. In *Proc. 34th Annual Meeting of the ACL*, 62–70.
- Cheikes, B., et al. 1999. Embedded training for complex information systems. *Int. J. of Artificial Intelligence in Education* 10:314–334.
- Chu-Carroll, J., and Carberry, S. 1995. Response generation in collaborative negotiation. In *Proc. 33rd Annual Meeting of the ACL*, 136–143.
- Davies, J., et al. 2001. Incorporating tutorial strategies into an intelligent assistant. In *Proc. Int. Conf. on Intelligent User Interfaces*.
- Gertner, A.; Cheikes, B.; and Haverty, L. 2000. Dialogue management for embedded training. In *AAAI Symposium on Building Dialogue Systems for Tutorial Applications*, Tech. Report FS-00-01. Menlo Park, CA: AAAI Press.
- Gruen, D.; Sidner, C.; Boettner, C.; and Rich, C. 1999. A collaborative assistant for email. In *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*.
- Kautz, H. A., and Allen, J. F. 1986. Generalized plan recognition. In *Proc. 5th National Conf. on Artificial Intelligence*, 32–37.
- Keyson, D., et al. 2000. The intelligent thermostat: A mixed-initiative user interface. In *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*, 59–60.
- Kraus, S.; Sycara, K.; and Evanchik, A. 1998. Argumentation in negotiation: A formal model and implementation. *Artificial Intelligence* 104(1-2):1–69.
- Lesh, N.; Rich, C.; and Sidner, C. 1999. Using plan recognition in human-computer collaboration. In *Proc. 7th Int. Conf. on User Modelling*, 23–32.
- Lesh, N.; Rich, C.; and Sidner, C. 2001. Collaborating with focused and unfocused users under imperfect communication. In *Proc. 9th Int. Conf. on User Modelling*. Submitted.
- Levesque, H. J.; Cohen, P. R.; and Nunes, J. H. T. 1990. On acting together. In *Proc. 8th National Conf. on Artificial Intelligence*, 94–99.

Lieberman, H. 1998. Integrating user interface agents with conventional applications. In *Proc. Int. Conf. on Intelligent User Interfaces*, 39–46.

Maes, P. 1994. Agents that reduce work and information overload. *Comm. ACM* 37(17):30–40. Special Issue on Intelligent Agents.

Ortiz, C. L., and Grosz, B. J. 2000. Interpreting information requests in context: A collaborative web interface for distance learning. *Autonomous Agents and Multi-Agent Systems*. To appear.

Rich, C., and Sidner, C. 1998. COLLAGEN: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction* 8(3/4):315–350.

Rickel, J.; Lesh, N.; Rich, C.; Sidner, C.; and Gertner, A. 2001. Using a model of collaborative dialogue to teach procedural tasks. In *Proc. 10th Int. Conf. on Artificial Intelligence in Education*. Submitted.

Sadek, D., and De Mori, R. 1997. Dialogue systems. In Mori, R. D., ed., *Spoken Dialogues with Computers*. Academic Press.

Sidner, C. L. 1994. An artificial discourse language for collaborative negotiation. In *Proc. 12th National Conf. on Artificial Intelligence*, 814–819.

Stein, A.; Gulla, J. A.; and Thiel, U. 1999. User-tailored planning of mixed initiative information seeking dialogues. *User Modeling and User-Adapted Interaction* 9(1-2):133–166.

Tambe, M. 1997. Towards flexible teamwork. *J. of Artificial Intelligence Research* 7:83–124.

Verbmobil. 2000. <http://verbmobil.dfki.de>.

Charles Rich is a Senior Research Scientist at Mitsubishi Electric Research Laboratories in Cambridge, Massachusetts. He received his Ph.D. from MIT in 1980. The thread connecting all of Rich’s research has been to make interacting with a computer more like interacting with a person. As a founder and director of the Programmer’s Apprentice project at the MIT Artificial Intelligence Laboratory in the 1980s, he pioneered research on intelligent assistants for software engineers. Rich joined MERL in 1991 as a founding member of the research staff. He is a Fellow and past Councillor of AAAI and was Co-Chair of AAAI’98. His email address is rich@merl.com.

Candace L. Sidner is a Senior Research Scientist at Mitsubishi Electric Research Laboratories in Cambridge, Massachusetts. She received her Ph.D. from MIT in 1979. Sidner has researched many aspects of user interfaces, especially those involving speech and natural language understanding and human-computer collaboration. Before coming to MERL, she was a member of the research staff at Bolt Beranek Newman, Inc., Digital Equipment Corp., and Lotus Development Corp., and a visiting scientist Harvard at University. She is a Fellow and past

Councillor of AAAI, past President of the Association for Computational Linguistics, and Co-Chair of the 2001 International Conference on Intelligent User Interfaces. Her email address is sidner@merl.com.

Neal Lesh is a Research Scientist at Mitsubishi Electric Research Laboratories in Cambridge, Massachusetts. He received his Ph.D. from the U. of Washington in 1998. His research interests of late lie primarily in human-computer collaborative problem solving. He has also worked in the areas of interface agents, interactive optimization, goal recognition, data mining, and simulation-based inference. After completing his thesis on scalable and adaptive goal recognition at U. Washington with Oren Etzioni, Lesh was a postdoc at U. Rochester with James Allen. His email address is lesh@merl.com.