Dialogue management for embedded training

Abigail S. Gertner and **Brant A. Cheikes** and **Lisa A. Haverty**
The MITRE Corporation
202 Burlington Road, mail stop K302
Bedford, Massachusetts, 01730-1420 USA*

Abstract

We are developing an embedded training system to teach students to use complex software applications. Our tutor reasons strategically about its pedagogical plans. It uses a discourse model to communicate with the student and to guide its strategic decision making. We are developing the tutor using the Collagen platform for collaborative agent management, which provides a plan representation language, plan recognition capabilities, and a discourse model. Our embedded training system uses instrumentation software we have developed to allow Collagen to communicate with the application the student is learning. It also includes a student model to assess the state of the student's domain knowledge. In this paper we discuss the need for dialogue in a strategic embedded training system. We then describe the system architecture and present an example of a possible training scenario.

Introduction

Complex information systems play a critical and growing role in today's workplace environment. Given the complexity of these systems, effective, high-quality training is increasingly being recognized as crucial. Nonetheless, training is consistently subject to budgetary pressures, and in order to eliminate the cost of sending workers to remote training locations, managers are demanding that employees be trained at or near the workplace. As a result of these pressures, distance learning is proliferating, and computer-assisted training is becoming increasingly attractive.

Our group is in the first year of a project in the area of *embedded training technology* – instructional tools embedded within the computing environment of the application(s) being used. Our goal is to develop an instructional agent to teach individual users how to operate complex software applications. The agent will adapt its instruction to the needs of the student by following explicit *instructional strategies*. This work is part of an ongoing research program in embedded training that began in 1996 (Cheikes *et al.* 1999).

We are focusing on the use of instructional strategies because such strategies allow us to specify how a tutoring agent can go about achieving its instructional goals. We define instructional strategies as comprising both tutorial "plans" or "recipes for action," as well as reactive planning rules that guide the tutor as it is executing its plan. For example, a tutor may adopt a model-scaffold-fade strategy for instructing the student. This is a high-level recipe for action: first perform the task for the student, then coach the student as she performs the task, and finally remove the coaching support and have the student perform the task alone. However, the student may not perform exactly as expected during the execution of this plan – she might ask questions, get stuck, or make mistakes. The tutor's strategy, therefore, should include rules specifying how to adapt the plan when an unexpected event occurs. For instance, one strategy for responding to student errors could be to give immediate feedback to correct the error, while another might be to wait for the student to discover and correct the error on her own.

An important feature of our embedded training ITS is that it can choose between alternative strategies when there is more than one available. Every ITS uses some kind of tutorial strategy, but most do not model these strategies explicitly – rather they are hard-coded into the underlying design of the tutorial engine (cf. Andes (Gertner, Conati, & VanLehn 1998)). As a result, these systems have limited ability to flexibly adopt and switch between different strategies. Our tutor encodes rules allowing it to choose between competing instructional strategies based on information about the current instructional context. Currently, our agent only uses information about the student to make these decisions, but in the future it will also make use of information about the the domain of instruction, and the history of the tutorial interaction (discourse model).

A tutor that reasons strategically must not only be able to choose between different strategies, but it must also be able to recognize when its current strategy is failing, and to switch to a new strategy if possible. This will require the tutor to have a representation of the recent history of the tutoring interaction in order to indentify when the student is "floundering" or repeatedly failing to respond as expected to the current instructional strategy. The need for a model of the discourse history will be discussed further in the following section.

The role of dialogue in a strategic ITS

A major obstacle to the design of effective tutoring systems is the design of the user interface. Many ITS's require stu-

*This work is supported by the MITRE technology program, project 51MSR8AA. We would like to thank Chuck Rich, Neal Lesh, and Candy Sidner for all of their help with Collagen.

dents to learn a specialized user interface, which can put an unacceptable cognitive load on a student who is already struggling to learn a new subject. This problem is particularly salient in the case of embedded training because the task that the student is engaged in learning is precisely the use of a complex graphical user interface. Having to learn one user interface in order to learn how to use another seems extremely infelicitous. Natural language dialogue is an attractive potential solution to the user interface problem for embedded training because it would allow students to interact with the tutor in a familiar manner, thus reducing the overall cognitive load.

Other motivations for using dialogue in our strategic tutorial system include both the ability to implement a wider range of strategies than would be possible with a structured user interface, and the ability to change strategies smoothly by informing the student of what is happening using natural language cues (eg. "No, that's not quite right. Let's try a different approach..."). Freedman (Freedman 1999) also discusses the importance of mixed-initiative dialogue to support the implementation of different tutorial strategies. More generally, others have argued that the ability to communicate in natural language may itself be intrinsic to the effectiveness of human tutors (Graesser, Person, & Magliano 1995).

Finally, the ability to refer to a model of the ongoing discourse provides the agent with the opportunity to adapt its tutoring to the discourse context. The agent will refer to the discourse model, which incorporates both natural language utterances and GUI actions, to help select an instructional strategy, to determine whether its current strategy is succeeding or failing, and to decide whether and when to change strategies.

To build a discourse model for our tutorial agent, we are using Collagen, an application-independent collaboration management platform developed by the Mitsubishi Electric Research Laboratory (MERL) (Rich & Sidner 1998). Collagen provides tools for dialogue management which, when given a task-specific "recipe" library, will perform plan recognition, track the focus of attention in the tutor-student interaction, and maintain an "agenda" of actions that could complete the plan. Its underlying representation of beliefs and intentions is based on the SharedPlans formalism of Grosz and Sidner (Grosz & Sidner 1986) and Grosz and Kraus (Grosz & Kraus 1996). From observations of the utterances and GUI actions performed by the dialogue participants, Collagen builds a model of their *joint intentions* with respect to the domain task, which in the case of tutoring is to complete a particular exercise. Collagen also provides interface elements to support dialogue interaction between an agent and a user.[1]

Collagen is primarily a tool for modeling the interaction between agent and user as they communicate with each other about activity in the target application (the application the student is being trained to use). It treats the agent as a black box which is capable of accessing the current discourse agenda, performing utterances, and manipulating the application. The specific decision-making process guiding the agent's behavior is up to the developer of the agent. Our embedded training agent will implement two key components. The first will be a set of instructional strategies specifically designed to train users in the use of complex information applications. The second is a reasoner that can determine and execute the most appropriate strategy at any given time.

[1]At this point, Collagen does not yet do any natural language understanding, so the user is still required to learn a special interface for constructing utterances to communicate with the agent.

Tutorial strategies for embedded training

Our approach to developing instructional strategies for embedded training begins with the idea (first suggested by Norman (Norman 1988)) that users of graphical user interfaces proceed through a cyclical process of forming intentions, selecting actions, acting, interpreting results, and modifying their intentions – in other words, they are engaged in dynamic planning. We believe that the "operator as dynamic planner" theory can help us develop effective instructional strategies for embedded training. Some informal examples of potential strategies include:

- When the student is first learning to perform a task, the agent might choose among three possible instructional strategies: (1) demonstrate the steps involved in the task, (2) walk the student through the task by pointing to the parts of the GUI that are involved in each step, or (3) simply ask the student to perform the task.
- The student should be taught how to interpret state information presented on the system's graphical displays. The agent might point to the display and explain its content, or it might first assess the student's understanding by asking questions about it, or by asking her to perform some action that depends on correctly interpreting the state of the display.
- Students should be taught how to recognize, diagnose, and respond to execution failures. In the case of recognizing execution failures, the agent's strategy might be to explicitly point to the part of the display that reflects the failure, or to give more indirect feedback, such as a content-free hint or prompt.

Our pedagogical agent will rely on a library of instructional strategies and principles drawn from the cognitive science literature. For example, the tutor design principles delineated in (Anderson *et al.* 1995) provide some useful guidelines, such as facilitating successive approximations to the target skill, and minimizing working memory load. Decisions about how much support, or scaffolding, a student needs at any point during an exercise may be informed by research on identifying "teachable moments" and the student's zone of proximal development (Vygotsky 1978).

Our tutor is intended to train people in using complex software applications. One goal of our work, therefore, is to identify pedagogical strategies specific to this teaching task that differ from the strategies used by other types of intelligent tutoring systems. Embedded training involves teaching a procedural task in the context in which it will ultimately be applied. Thus, the best strategies for embedded training

are probably similar to strategies used to teach any procedural task. They will differ from strategies used to teach other types of knowledge, such as tutoring in a conceptual domain like mathematics. Embedded training strategies will differ from other types of procedural tutoring in the low-level details specific to using graphical user interfaces.

The appropriate choice of strategy will often depend on the student's level of domain knowledge, which is represented in the training system's student model. We are using a basic overlay student model which reflects the student's successful achievement of individual domain actions and procedures. Additionally, the agent's strategy will depend on such considerations as whether it has given the same explanation before, how much time the student has spent on the current problem (and hence her potential level of frustration), and how many problems the student has solved correctly so far (she may need a boost to improve her motivation level). This information will be available to the agent by querying the discourse model.

System architecture and instrumentation

We have implemented a prototype embedded training system that can choose between two simple coaching strategies. The tutor refers to information from its student model to determine which strategy to use. A high-level view of the architecture of our tutoring system is illustrated in Figure 1. The student and the tutoring agent can communicate with each other via natural language through Collagen and can both act directly on the domain application as well. The tutor also has a pointing hand that can move around the screen and point at objects that are being discussed or manipulated. Collagen observes the utterances generated by both student and tutor, and it also observes the actions performed on the application by both participants. These observations are stored in the discourse model.

Normally, Collagen requires that the domain application it is working with be written (and documented) in such a way that the Collagen system can register itself to be notified of "semantic events" that occur in the application. This is not necessary for our tutor because we are taking advantage of JOSIT (Java Observation, Simulation and Inspection Tool), a tool for "instrumenting" applications that was one of the products of our earlier embedded training research.[2] Instead of relying on the application to provide information about semantic events as they happen, JOSIT allows another application (in this case, Collagen) to observe the low-level interface events such as window openings and button clicks. The observer can then construct a model of the semantic events based on the information about what is happening on the interface.

As its name suggests, JOSIT provides not just the ability to *observe* user interfaces, but also to *simulate* events on the interface, and to *inspect* the state of the interface at any given time. Simulation is used by the agent to demonstrate actions or perform actions for the user. Inspection makes it possible for the agent to develop a more complete model of the state of the situation than is possible to develop based on observing interface actions alone.

[2] JOSIT is named after its predecessor, WOSIT (Widget Observation, Simulation and Inspection Tool) which works with X-Windows/Motif applications. Information about WOSIT can be found at `http://www.mitre.org/technology/wosit/`

A training scenario

For the initial prototype of our embedded training agent, we are working with an application called TARGETS (Terminal Area Route Generation Evaluation Traffic Simulation), which is a tool for developing new arrival and departure routes into and out of airports. TARGETS is a standalone Java application being developed by MITRE's Center for Advanced Aviation System Development (CAASD). To develop a route, the TARGETS user first creates a new "project file" associated with the desired airport. The user imports data about the surrounding area, including topographical data, existing navigation stations, and data on actual flight paths flown around the airport in the past. These data are displayed on a graphical view window. The user then defines a new path using a drawing tool to specify the lateral position fixes, and then enters the altitude and speed of the airport at each fix point using a table. Displayed data are used to guide the definition of the new flight path. Finally, the user can check the "flyability" of the new path giving information about the typical aircraft and wind conditions. Any parts of the path that are considered not flyable must be edited and re-checked.

The student learning how to use TARGETS with the embedded training agent will go through a sequence of exercises designed to introduce new features of the system a few at a time. In each exercise, the tutor will introduce the new feature and then observe as the student practices using it. The tutor will adapt its strategies for introducing the material and scaffolding the student's practice based on the student model, the discourse model and the domain model. These sources of information will control such parameters as whether the tutor will demonstrate the new feature for the student, whether it will volunteer hints before the student asks for them, and how it will respond to errors.

If the tutor is teaching a feature of the application that the student has never seen before, it might give a short introduction to the new feature, demonstrate it by performing actions on the interface, and then walk the student step-by-step through performing the actions herself. If, on the other hand, the feature is very similar to one the student already knows, the tutor might introduce the feature by comparing it with the related item, perhaps give a demonstration, and then ask the student to perform the actions without giving a hint at every step. We are now in the process of implementing these strategies in our tutoring agent.

Conclusion

In this paper we have discussed our goal of developing an intelligent agent for embedded training that is able to reason strategically about how to achieve its instructional goals. The agent will choose between alternative strategies using information about the student, the domain, and the tutoring interaction. It will execute its strategies, monitoring and re-

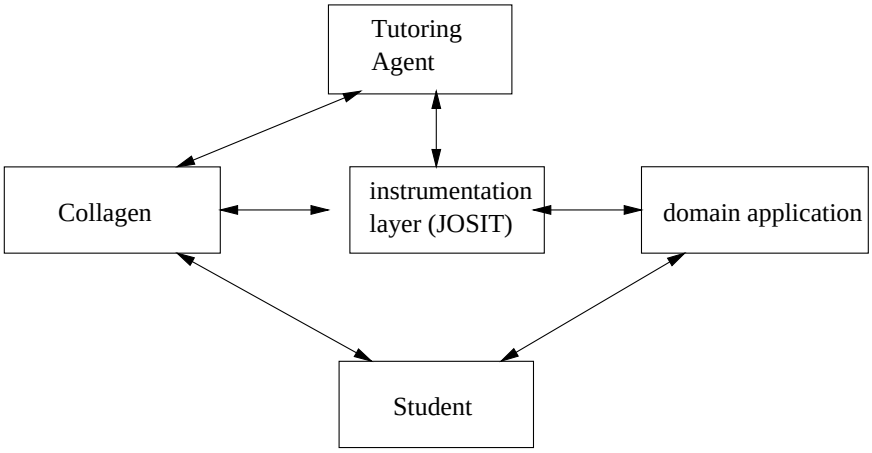

Figure 1: Tutoring system architecture

acting to the student's responses, and replanning when necessary. Instructional strategies will determine not only the steps of the tutorial plan, but also the ways in which the tutor will react to unexpected events.

We discussed the motivations for using dialogue in our tutoring system. These include the need for an easy to use user interface, the ability to implement many different tutorial strategies using natural language, and the utility of a representation of the discourse history to make strategic decisions.

We have begun to implement a very simple tutorial agent, using JOSIT to observe and simulate events in a domain application (TARGETS), and Collagen to incorporate these observations into a model of the current plan and the discourse history. Future work includes defining and implementing instructional strategies specific to the embedded training domain, enhancing the agent's strategy-choice rules to include information about the domain model and the discourse model, and implementing a decision procedure to allow the agent to abandon one strategy and adopt a new one.

A final thought: Given that our tutoring system is making use of Collagen, which was originally developed to support intelligent assistants, rather than tutors, the question of the relationship between tutoring and assisting naturally arises. We believe that the two functions are closely related (Davies *et al.* 2000), and that the strategic reasoning capabilities we are developing for our tutor could serve an important role in an intelligent assistant as well. Ideally, the tutor will fade away over time as the student becomes familiar with the domain, and its behavior will start to look more and more like an assistant. This transition would be handled naturally by strategies like the ones we are developing for tutoring.

References

Anderson, J.; Corbett, A.; Koedinger, K.; and Pelletier, R. 1995. Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences* 4(2):167–207.

Cheikes, B. A.; Geier, M.; Hyland, R.; Linton, F.; Riffe, A. S.; Rodi, L. L.; and Schaefer, H.-P. 1999. Embedded training for complex information systems. *International Journal of Artificial Intelligence in Education* 10:314–334.

Davies, J. R.; Lesh, N.; Rich, C.; Sidner, C. L.; Gertner, A. S.; and Rickel, J. 2000. Incorporating tutorial strategies into an intelligent assistant. submitted to the 2001 International Conference on Intelligent User Interfaces.

Freedman, R. 1999. Atlas: A plan manager for mixed-initiative, multimodal dialogue. In *Proceedings of the AAAI'99 Workshop on Mixed-Initiative Intelligence*.

Gertner, A. S.; Conati, C.; and VanLehn, K. 1998. Procedural help in Andes: Generating hints using a Bayesian network student model. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*.

Graesser, A. C.; Person, N. K.; and Magliano, J. P. 1995. Collaborative dialogue patterns in naturalistic one-to-one tutoring. *Applied Cognitive Psychology* 9:495–522.

Grosz, B. J., and Kraus, S. 1996. Collaborative plans for complex group action. *Artificial Intelligence* 86(2):269–357.

Grosz, B. J., and Sidner, C. L. 1986. Attentions, intentions and the structure of discourse. *Computational Linguistics* 12:175–204.

Norman, D. A. 1988. *The Psychology of Everyday Things*. Basic Books.

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

Vygotsky, L. S. 1978. *Mind in Society: The Development of Higher Psychological Processes*. Cambridge, MA: Harvard University Press.