Task-Oriented Tutorial Dialogue: Issues and Agents

Jeff Rickel & Rajaram Ganeshan
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292
rickel, rajaram@isi.edu
http://www.isi.edu/isd/carte

Charles Rich, Candace L. Sidner & Neal Lesh
Mitsubishi Electric Research Laboratory
201 Broadway
Cambridge, MA 02139
rich, sidner, lesh@merl.com
http://www.merl.com/projects/collagen

Abstract

Towards our goal of developing computer tutors that can collaborate with students on tasks in simulated environments, we have built a series of four prototype tutors. These tutors integrate two independent but related strands of research: intelligent tutoring systems and task-oriented dialogue systems. While the tutors share a core approach to teaching procedural tasks, each was designed to explore a different set of issues. This paper outlines the issues that arise in task-oriented tutorial dialogue and the ways they have been addressed in these four tutors.

Introduction

To master complex tasks, such as operating complicated equipment, people need hands-on experience facing a wide range of situations. They also need a mentor that can demonstrate procedures, answer questions, monitor their performance, and provide feedback. Since it is often impractical to provide such training on real equipment, we are exploring the use of simulation-based training. Since mentors are often unavailable when students need them, we are developing computer tutors that can take their place. Thus, our research objective is to develop computer tutors that can collaborate with students on tasks in simulated environments.

Our work integrates two independent but related strands of research: intelligent tutoring systems and task-oriented dialogue systems. Research on intelligent tutoring systems (Carbonell 1970; Sleeman & Brown 1982; Wenger 1987) focuses on computer tutors that can adapt to individual students through the use of artificial intelligence. Such systems dynamically adapt their teaching based on a representation of the target knowledge the student is expected to learn and a representation of the student's presumed state of knowledge. Research on task-oriented dialogue (Grosz [Deutsch] 1974; Lochbaum 1998; Traum 1994), which has an equally long history, focuses on computational models of human dialogue for collaborative tasks. While these two research communities have focused on different issues, they are equally important to our goal of supporting task-oriented, tutorial dialogue between students and computer tutors.

Our research on this topic has resulted in a series of computer tutors. While these tutors share a core approach to teaching procedural tasks, each was designed to explore a different set of issues. In this brief paper, we review the similarities and differences among these tutors and the issues they raise for task-oriented, tutorial dialogues.

TOTS, STEVE, PAT, and PACO

Our work on this topic began when one of the authors (Rickel) began studying the problem of intelligent tutoring for procedural tasks. His work resulted in the first domain-independent shell for constructing such systems (Rickel 1987; 1988), called TOTS (Task-Oriented Tutoring System). The shell included a rich representation for procedural tasks, based on Sacerdoti's procedural networks (Sacerdoti 1977), an overlay representation of the student's knowledge (Goldstein 1977), and the ability to dynamically interleave teaching and coached practice based on a student model.

Subsequent work by Rickel and Johnson (1999; 2000) focused on a different aspect of task-oriented, tutorial dialogue: face-to-face interaction in a shared environment. This work resulted in STEVE (Soar Training Expert for Virtual Environments), a graphical human figure that cohabits three-dimensional virtual environments with students to teach them physical tasks. STEVE has many of the same pedagogical capabilities as TOTS. However, because it has an animated body, and cohabits the virtual world with students, it can provide additional types of assistance. For example, it can demonstrate actions, use gaze and gestures to direct the student's attention, and guide the student around the virtual world (Johnson, Rickel, & Lester 2000).

Our third agent, PAT (Pedagogical Agent for Training), was developed by Ganeshan and Rickel to support task-oriented, tutorial dialogues on the Web. PAT was designed to combine the tutorial capabilities of TOTS, the task reasoning capabilities of STEVE, and a more complete model of task-oriented dialogue than either of those prior systems. Like STEVE, PAT appears to the student as an animated human figure. However, to simplify the interface for Web delivery, PAT is animated from two-dimensional bitmap images rather than

a full three-dimensional graphical model, and appears in a separate window from the simulated environment.

Our fourth agent, PACO (Pedagogical Agent for COLLAGEN), which is currently under development by Ganeshan and Rickel, is an attempt to build a computer tutor on top of COLLAGEN (Rich & Sidner 1998). COLLAGEN is a separate piece of software developed at the Mitsubishi Electric Research Laboratory as a general model of task-oriented dialogue. While COLLAGEN does not embody any tutorial capabilities, it provides a more general model of collaborative conversation than our previous systems.

Each of these four tutors consists of a set of domain-independent capabilities that utilize a declarative representation of domain knowledge. To teach students about the tasks in a new domain, someone must provide the appropriate domain knowledge. We assume that this domain author will be someone with enough domain knowledge to create a course for teaching others. Importantly, we do not assume that this person has any programming skills, so we have tried to ensure that our tutors only rely on types of knowledge that a domain author can provide.

Learning Environment

All of our tutors are designed for simulation-based training. That is, students learn tasks by performing them in a simulation of the real work environment. Of course, if the target work environment is actually a software application, that application can serve as the simulator. Our tutors make few assumptions about the nature of the simulated environment and the student's interface to it. Each of the tutors has an API through which it receives messages from the simulator describing state changes and student actions and sends messages to take action itself. STEVE is unique among our agents in actually appearing in the simulated environment with the student; the other tutors appear in separate windows. (Although COLLAGEN does allow a pointing hand to move over the simulator window to point at objects.)

All of our tutors' instruction and assistance is situated in the performance of domain tasks in the simulated environment. That is, the tutor chooses a scenario (task to perform starting from a particular simulation state), works through it with the student, and then repeats until all scenarios have been covered. Our goal is to support the apprenticeship model of learning (Collins, Brown, & Newman 1989). This requires two capabilities. First, the tutor must be able to perform and explain the task, in order to teach the student how to do it. Second, it must be able to monitor the student as she performs the task, providing assistance when it is needed. As the student gains proficiency, the assistance provided by the tutor should decrease. Ideally, students should learn to apply well-defined procedures to a variety of situations.

Our tutors support both actions and utterances by students. All four tutors allow the student to perform actions in the simulated environment. They differ in the range of utterances they allow. None of our tutors support natural language understanding; the student uses a GUI to construct utterances. (STEVE supports speech recognition, but the student is still limited to a relatively small range of allowable utterances.) TOTS only allowed the student to ask what should be done next. STEVE and PAT additionally allow the student to ask "Why?" following most suggestions or explanations, and STEVE allows the student to ask for a demonstration, interrupt a demonstration and ask for the task initiative, and use domain-specific utterances. PACO uses the dialogue interface provided by COLLAGEN, which allows the user to construct a rich set of utterances appropriate for task-oriented collaboration. The set of utterances is based on Sidner's (1994) artificial discourse language. Students can, for example, propose a goal or action, ask or propose who should perform a goal or action, ask about or propose the value of a parameter (input) to a goal or action, and ask or propose how a given goal should be achieved.

Our work focuses on mixed-initiative dialogue with the student, where either the tutor or student can act or speak at any time. Our early work on TOTS followed the more typical approach in intelligent tutoring systems in which the tutor is in complete control. To maintain control, all student actions were passed through TOTS before being sent to the simulator; if an action was incorrect, the student received feedback from TOTS but the action was never sent to the simulator. While this approach is simple to implement, it prevents students from seeing the consequences of mistakes and learning to recover from them. For this reason, STEVE, PAT and PACO share equal control with the student. This requires an additional ability to recover from student errors and other unexpected events, as discussed shortly.

Agent Architecture

There are four key modules in our tutors. The *expert module* embodies the knowledge and capabilities that should be taught to the student; it knows how to solve the problems being given to her. The *dialogue manager* maintains the state of the tutorial dialogue. *Plan recognition* uses the output of the expert module and dialogue manager to interpret the student's actions and utterances; it decides what the student is trying to do. Finally, the *pedagogical executive* uses the output of the other three modules to decide what to say or do next. Below, we discuss these four modules and the issues they raise for task-oriented tutorial dialogue.

Expert Module

The expert module of an intelligent tutoring system provides a model of the skills that the student is expected to acquire. Since our tutors teach procedural tasks, they require an expert module that can perform such tasks. That is, they must be able to decide, based

on the state of the simulator, what the next appropriate actions are and why. Since our objective is to design a domain-independent tutor that can be applied to a wide variety of domains, our expert module must provide two things: a representation language for domain task knowledge, and algorithms that can use knowledge expressed in that language to perform domain tasks.

Intelligent tutoring systems typically represent procedural knowledge in one of two ways. Some, notably those of Anderson and his colleagues (Anderson *et al.* 1995), use detailed cognitive models built from production rules. Such systems perform domain tasks by directly executing the rules. Other systems use a declarative representation of the knowledge, usually some variant of a procedural network representation (Sacerdoti 1977) specifying the steps in the procedure and their ordering. Such systems perform tasks by using a domain-independent interpreter to "execute" the procedural network (i.e., walk through the steps). To simplify authoring, all of our tutors use a declarative representation.

All four tutors use the same basic representation, drawn from research on planning within artificial intelligence, although they differ in some details. First, each task consists of a set of steps, each of which is either a primitive action (e.g., press a button) or a composite action (i.e., itself a task). Composite actions give tasks a hierarchical structure. Second, there may be ordering constraints among the steps; these constraints define a partial order over the steps. In addition to steps and ordering constraints, STEVE and PAT represent the rationale for steps in the task as a set of causal links; each causal link specifies that one step in the task achieves a goal that is a precondition for another step (or for termination of the task). For example, pulling out a dipstick achieves the goal of exposing the level indicator, which is a precondition for checking the oil level. In addition to steps, ordering constraints, and causal links, each of the tutors supports other constructs that are useful for representing task knowledge, but these three form the core of the representation.

Whether demonstrating a task for a student or monitoring the student as she performs the task, the tutors use their task knowledge to decide what must be done next. TOTS assumes that steps must be done in the order specified by the task knowledge, although it does allow conditional links in the task knowledge so that the exact execution sequence can depend on parameters of the problem given to the student. PACO relies on COLLAGEN's representation and use of task knowledge; COLLAGEN has some abilities to use the state of the simulation to recognize when a step can be skipped or must be redone. PAT and STEVE both incorporate a partial-order planning algorithm (Weld 1994) that uses the causal links to construct a plan for completing the task based on the simulator state (Rickel & Johnson 1999). This gives them the most complete ability to recover from errors by the student and other unexpected events created by the simulator (e.g., a simulated equipment failure). The causal links also allow PAT and STEVE to automatically generate explanations for how the steps they suggest to the student contribute to completing the task (Rickel & Johnson 1999).

Dialogue Manager

Perhaps no other module of our tutors has evolved as much as the dialogue manager. TOTS had almost no representation for the state of the dialogue. It only kept track of the focus node (i.e., the step in the task that is the current topic of discussion). To a large extent, this simple representation was sufficient because TOTS retained full control over the flow of the dialogue, which followed a relatively fixed pattern.

The more dynamic nature of STEVE's interactions with students required a more explicit representation of the dialogue state (Rickel & Johnson 2000). Because utterances take time (synthesized speech for STEVE's voice, speech recognition for the student), STEVE must keep track of whether it and/or the student is speaking, and avoid overlaps. It must keep a record of who has the task initiative (i.e., responsibility for deciding which task steps to do next), because either STEVE or the student can have it and the task initiative can change during the course of a scenario. STEVE keeps a record of which steps have been performed already; this allows it to acknowledge when a step must be redone, rather than implying that the second performance of the step is normal. When STEVE answers a student question ("What next?" or "Why?"), it records its answer in case the student asks a follow-up question ("Why?"). For the task step currently in focus, STEVE records the status of the collaboration with the student (e.g., whether it proposed the step, whether it explained it, whether it or the student performed it, and whether they discussed the result); among other things, this helps STEVE decide what to say if collaboration on the step must be aborted (e.g., if a change in the simulated world makes it irrelevant or inapplicable). Finally, since errors by the student or unexpected simulator events (e.g., a high-priority alarm) can interrupt a subtask on which STEVE and the student are working, STEVE maintains a focus stack (Grosz & Sidner 1986); this allows STEVE to return to the interrupted task after handling the interruption (Rickel & Johnson 1999).

These dialogue management issues that arose in STEVE convinced us of the importance of representing the state of the dialogue. Beginning with our work on PAT, we sought a more principled representation. PAT's dialogue manager is based largely on the ideas of Grosz and Sidner (1986). However, rather than discuss PAT's dialogue manager, we turn instead to our work on PACO. PACO's dialogue manager is COLLAGEN, which employs a similar but more general approach than PAT.

In COLLAGEN, the attentional component of the dialogue state, i.e., what are we talking about and/or

working on *now*, is represented by a hierarchy of dialogue segments and a dialogue segment (focus) stack. A dialogue segment is a contiguous sequence of actions and utterances that contribute to some purpose. The intentional component of the dialogue state, i.e., the status of goals, is represented by plan trees which are a partial implementation of SharedPlans (Grosz & Sidner 1990; Grosz & Kraus 1996). The heart of COLLAGEN's dialogue management is the discourse interpretation algorithm, based on (Lochbaum 1998), which specifies how to update the dialogue state given a new action or utterance by either the student or tutor. This algorithm links the goals and plans of collaboration to the dialogue state via the actions and utterances.

Plan Recognition

When a student takes an action or says something, the tutor must decide what the student is trying to do. This allows the tutor to update a model of the student's knowledge as well as provide appropriate feedback in the case of errors. When the tutor is maintaining a representation of the dialogue state, it also allows appropriate updates to that state. Recognizing someone's intentions from their actions and utterances is typically called plan recognition, an area with a long research history. In intelligent tutoring, this area is often called student diagnosis (Wenger 1987), since research on tutoring systems focuses on diagnosing why a student performed an erroneous action in order to give appropriate feedback.

Plan recognition, especially for intelligent tutoring, can exploit two types of knowledge. The first is the correct knowledge of the tasks, as embodied in the expert module. The second is buggy knowledge (Burton 1982) representing common errors in the domain. The addition of buggy knowledge increases the burden on the domain author, who must encode the knowledge, and increases the search space for the plan recognition module, but it can result in more appropriate feedback to the student. For generality, all our tutors focus on the use of the correct knowledge rather than requiring any buggy knowledge.

Since TOTS did not allow student utterances except "What next?", it focused on interpreting the student's domain actions. It did so by searching for a step in the task knowledge that matches the student's action and provides the most likely interpretation. It recognized and responded to five general cases. (1) The student's action was correct if it matched one of the actions that the expert module thought could be done next. (2) The student's action was invalid if it didn't match any action in the domain knowledge. (3) The student's action may be appropriate for the given goal except that the problem given to the student precluded the action; for example, she may be proceeding as for a two-barrel carburetor when the problem stated that the car has a four-barrel carburetor. TOTS recognized this case by asking the expert module to relax all problem constraints and search again for all valid next actions; if the student's action was found, TOTS could identify which problem constraints it violated and give appropriate feedback. (4) The student may have taken the wrong action for the right goal; for example, she may have violated an ordering constraint by omitting a necessary action. (5) The student's action may be valid only for an irrelevant goal; for example, she may be changing the tire to fix a carburetor problem. Cases 3-5 above are examples of what is often called "near miss" plan recognition.

PAT also has a set of cases it can distinguish by searching through the task knowledge for an interpretation of the student's action. (1) Like TOTS, it classifies a student's action as correct if it matches one of the actions that the expert module thinks should be done next. (2) An action is irrelevant if it does not appear anywhere in the expert module's plan for completing the task (i.e., it does not contribute to any remaining goals of the task). (3) If an action is relevant (i.e., in the tutor's plan), it is classified as inapplicable if its preconditions are not all satisfied. (4) Finally, if an action is relevant and applicable, it is classified as precluded if an ordering constraint requires another relevant, applicable step to be done first. If a student's action matches multiple steps in the task knowledge, PAT uses the current dialogue focus to disambiguate when possible; that is, it prefers an interpretation of the student's action that is a continuation of the current subtask over other interpretations that would represent a focus shift (Grosz & Sidner 1986).

COLLAGEN's plan recognition is more general than what is used in our earlier tutors because COLLAGEN must interpret a variety of student utterances in addition to student actions. However, since plan recognition is well known to be intractable in the general case (Kautz 1990), COLLAGEN exploits properties of the collaborative setting in order to make plan recognition practical (Lesh, Rich, & Sidner 1999). Specifically, it exploits the following properties: the focus of attention (as described above for PAT), the use of partially elaborated hierarchical plans (which indicate the dialogue acts that would contribute to progress on the task), and the possibility of asking for clarification. COLLAGEN's plan recognizer also provides very general and extensible facilities for near-miss plan recognition.

Pedagogical Executive

The previous three modules – the expert module, dialogue manager, and plan recognition – serve mainly to guide the pedagogical executive. The pedagogical executive is the module that decides what the tutor should say or do next. The other three modules simply provide the information it needs to make such decisions.

In teaching students how to perform tasks, our goal is to have our tutors support an apprenticeship model of instruction (Collins, Brown, & Newman 1989), first showing the student how to perform the task, then providing assistance as the student practices the task, and gradually providing less assistance as the student ap-

proaches mastery. If there were no overlap among tasks and scenarios, this approach could be implemented in the obvious way: the tutor would first demonstrate the entire task, then repeatedly let the student practice the task, providing assistance where necessary. However, different tasks often share common subtasks or actions, and different scenarios often require variants of the same task. Therefore, at any moment, a student's level of mastery may differ across the different parts of a task. For example, a new scenario may require branches of a task that the student has not yet seen while also requiring steps and subtasks that have been mastered already. Thus, our tutors must use a student model to adapt their instruction accordingly. Except for STEVE, which does not use a student model, all our tutors use an overlay model (Goldstein 1977) that records, for each element of domain knowledge, whether the student has been exposed to it and her current degree of mastery.

TOTS, PAT, and PACO use the student model to dynamically interleave demonstration and coached practice. As the student and the tutor progress through a task, the expert module will repeatedly identify the next possible steps that could be done. The tutor will consult the student model to see whether the student has sufficient knowledge to choose the next step. If so, the tutor will expect the student to take the next step, and will provide assistance only if the student requests it or makes a mistake. If not, the tutor will intervene and teach the student what to do next. Thus, as the tutor and the student work through tasks, task initiative will pass back and forth between them based on the student's prior experience. Whenever the tutor decides that the task initiative should shift, it will let the student know through verbal comments (e.g., "Let me show you what to do next" or "You take it from here").

Our research has focused more on deciding *what* the tutor should say next than on *how* it should say it. All of our tutors use relatively simple text templates for natural language generation. The domain author associates elements of domain knowledge with text fragments that the tutor dynamically plugs into the text templates. One issue we have addressed is the use of cue phrases. Because STEVE and PAT include planners that can dynamically decide how steps should be ordered in the face of unexpected events, text fragments cannot assume the order in which steps will be done. Therefore, both those tutors dynamically insert cue phrases (e.g., "First," "Next," "Now we can", "As I told you before") into their utterances to help show the relationship of new utterances to old ones (Rickel & Johnson 1999). The use of cue phrases has been identified as an important element of human tutorial dialogues (Moore 1996).

Conclusions

Through our four tutors, we have explored the issues that arise in task-oriented, tutorial dialogues. Despite their many similarities, our work on each tutor focused on different issues. Together, they have helped us understand the required interplay between an expert module, a dialogue manager, plan recognition, and a pedagogical executive.

Our current work is progressing in two directions. First, through new funding provided by the Army Research Office (via the USC Institute for Creative Technologies), we are continuing our research on STEVE. The project focuses on face-to-face tutorial interaction in a shared environment, especially the interplay between verbal and nonverbal communication. Second, we are continuing work on PACO. Because COLLAGEN offers a general, modular model of task-oriented dialogue, it provides an opportunity to integrate the independent but related strands of research on intelligent tutoring and task-oriented dialogue in a more complete and principled way than was previously possible.

Acknowledgments

The work described in this paper has been supported by several different funding sources. TOTS was supported by funding from Texas Instruments Incorporated. STEVE was supported by funding from the Office of Naval Research under grant N00014-95-C-0179 (subcontract to Lockheed Martin Incorporated) and AASERT grant N00014-97-1-0598. PAT was supported by funding from the Air Force Research Laboratory (subcontract to Intelligent Systems Technology Incorporated) under grants F41624-97-C-5018 and F41624-98-C-5008. COLLAGEN and PACO are supported by funding from the Mitsubishi Electric Research Laboratory.

References

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

Burton, R. R. 1982. Diagnosing bugs in a simple procedural skill. In Sleeman, D., and Brown, J., eds., *Intelligent Tutoring Systems*. Academic Press. 157–183.

Carbonell, J. R. 1970. AI in CAI: An artificial-intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems* 11(4):190–202.

Collins, A.; Brown, J. S.; and Newman, S. E. 1989. Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In Resnick, L., ed., *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Goldstein, I. P. 1977. Overlays: A theory of modelling for computer-aided instruction. Artificial Intelligence Laboratory Memo 495, Massachusetts Institute of Technology, Cambridge, MA.

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. 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.; Morgan, J.; and Pollack, M., eds., *Intentions in Communication*. MIT Press. chapter 20, 417–444.

Grosz [Deutsch], B. J. 1974. The structure of task oriented dialogs. In *Proceedings of the IEEE Symposium on Speech Recognition*. Pittsburgh, PA: Carnegie-Mellon University. Also available as Stanford Research Institute Technical Note 90, Menlo Park, CA.

Johnson, W. L.; Rickel, J. W.; and Lester, J. C. 2000. Animated pedagogical agents: Face-to-face interaction in interactive learning environments. *International Journal of Artificial Intelligence in Education* 11:47–78.

Kautz, H. 1990. A circumscriptive theory of plan recognition. In Cohen, P. R.; Morgan, J. L.; and Pollack, M. E., eds., *Intentions and Communication*. Cambridge, MA: MIT Press. 105–133.

Lesh, N.; Rich, C.; and Sidner, C. L. 1999. Using plan recognition in human-computer collaboration. In *Proceedings of the Seventh International Conference on User Modeling*, 23–32.

Lochbaum, K. E. 1998. A collaborative planning model of intentional structure. *Computational Linguistics* 24(4):525–572.

Moore, J. D. 1996. Making computer tutors more like humans. *Journal of Artificial Intelligence in Education* 7(2):181–214.

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.

Rickel, J., and Johnson, W. L. 1999. Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. *Applied Artificial Intelligence* 13:343–382.

Rickel, J., and Johnson, W. L. 2000. Task-oriented collaboration with embodied agents in virtual worlds. In Cassell, J.; Sullivan, J.; Prevost, S.; and Churchill, E., eds., *Embodied Conversational Agents*. Boston: MIT Press.

Rickel, J. W. 1987. An intelligent tutoring framework for task-oriented domains. Master's thesis, University of Texas at Dallas.

Rickel, J. 1988. An intelligent tutoring framework for task-oriented domains. In *Proceedings of the International Conference on Intelligent Tutoring Systems*, 109–115. Montréal, Canada: Université de Montréal.

Sacerdoti, E. 1977. *A Structure for Plans and Behavior*. New York: Elsevier/North-Holland.

Sidner, C. L. 1994. An artificial discourse language for collaborative negotiation. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, 814–819. Menlo Park, CA: AAAI Press.

Sleeman, D., and Brown, J., eds. 1982. *Intelligent Tutoring Systems*. Academic Press.

Traum, D. R. 1994. *A Computational Theory of Grounding in Natural Language Conversation*. Ph.D. Dissertation, Department of Computer Science, University of Rochester, Rochester, NY.

Weld, D. S. 1994. An introduction to least commitment planning. *AI Magazine* 15(4):27–61.

Wenger, E. 1987. *Artificial Intelligence and Tutoring Systems*. Los Altos, CA: Morgan Kaufmann.