Reference on last page

Chapter Three

Technology Support for Complex Problem Solving

From SAD Environments to AI

Gautam Biswas, Daniel Schwartz, John Bransford, and the Teachable Agents Group at Vanderbilt

For the past decade, the Cognition and Technology Group at Vanderbilt (CTGV) has been studying how technology can help students learn to approach the challenges involved in solving complex problems and learning about new topics. Work centered around our video-based *Jasper Woodbury Problem Solving Series* represents one example; a book written by our group summarizes this work (CTGV, 1997). A summary of work that goes beyond the Jasper series appears in CTGV (1998).

We note in the Jasper book (CTGV, 1997) that our work began with simple, interactive videodisc technology, plus software for accessing relevant video segments on a "just-in-time" basis. We needed the interactivity because Jasper adventures are twenty-minute video stories that end with complex challenges for students to solve. All the data relevant to the challenges (plus lots of irrelevant data that students have to sort through) have been embedded in the story line. An overview of the Jasper Series is illustrated in figure 1; a brief description of one of the Jasper adventures, *Rescue at Boone's Meadow* (RBM), appears in figure 2.

After viewing a Jasper adventure, students usually work in groups to solve the challenge. (A CD-ROM that accompanies the Jasper book illustrates this process; see CTGV, 1997.) To succeed, students need access to the data embedded in the Jasper story. Even if they cannot remember the details about required data, they can usually remember where in the story the data had been provided. The software lets them return to the relevant part almost instantly. For example, in the RBM adventure, students may return to the flying field scene to review how

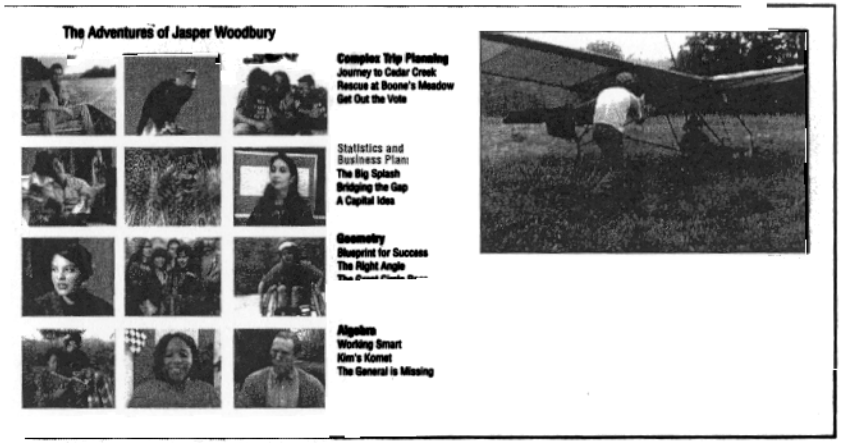

Figure 1. (left) The twelve adventures of Jasper Woodbury.
Figure 2. (right) Overview of the Jasper adventure, Rescue at Boone's Meadow.

much fuel an ultralight contained; go to the restaurant scene to find a conversation that explained how large the landing field was in Boone's Meadow; access Dr. Ramirez's office to review how far it was from one city to the next, and so forth. An example of the visually organized, random-access environment for RBM is illustrated in figure 3.

The software environments developed for each of the twelve Jasper adventures are very simple for students and teachers to use and extremely helpful as a support for student learning. From the perspective of AI, however, the software is trivial. Our approach to research has been to start with stone-age design (SAD) environments, and add sophistication and complexity only as necessary to achieve our instructional goals. Our SAD approach has allowed us to work closely with hundreds of teachers and students and, in this process, identify and test situations where increased technology support can further facilitate learning. In this chapter we especially emphasize situations where AI insights and techniques have become extremely helpful.

We will describe two examples where principles from AI have allowed us to improve student learning. One involves creating an AdventurePlayer program, plus offshoots of that program, to accompany the Jasper series (Crews et al. 1997). A second involves the use of AI techniques to create "teachable agents" whom students explicitly teach to perform a variety of complex activities. (The emphasis on teachable agents is different from an emphasis on learning agents that learn on their own without explicit teaching and without assessments of the adequacy of the agents' new knowledge. In other words, our teachable agents do not have machine learning algorithms embedded into their reasoning processes.) Our work on teachable agents is quite new, so the ideas and data we present are still preliminary. We hope

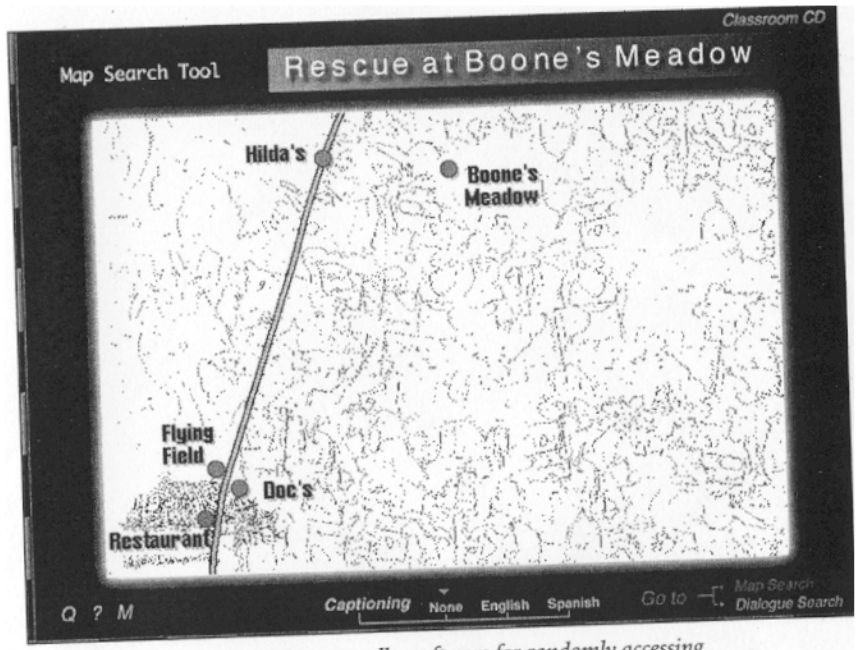

Figure 3. Map controller software for randomly accessing video scenes from Rescue at Boone's Meadow.

that our discussion of this project will help connect us with others who can provide insights about ways that we can strengthen and accelerate our current work.

An Example of Moving from SAD to AI

As noted above, our approach to research and development has been to begin with very simple uses of technology (SAD) and work in classroom settings to identify instances where increased technological sophistication could have a significant impact on student (and often teacher) learning. One example comes from our work on transfer. Our early data on Jasper showed that after students solved a Jasper challenge, they were better able to solve similar, complex challenges (e.g., CTGV, 1997, chapter 4; Van Haneghan et al. 1992). However, their abilities to transfer were still relatively brittle and inflexible (CTGV, 1997, chapters 4 and 5). One reason is that teachers sometimes had difficulty managing the learning of all the students in the classroom, so their degree of initial learning was not sufficient to support transfer (e.g., see Bransford, Brown, and Cocking 1999 for an overview of the transfer literature). Another reason was that research on transfer shows that learning can be "welded" to a

particular concrete context and that transfer can suffer unless it is facilitated by opportunities to see concepts applied in multiple contexts rather than only one (e.g., Bransford et al. 1999; Gick and Holyoak 1980, 1983).

Our original plan had been for teachers to always use at least two Jasper adventures that build on one another. For example, after students solved RBM (which involves concepts of distance, rate and time in the context of trip planning), teachers could present *Journey to Cedar Creek* (which involves distance, rate and time in the context of a boat trip). However, teachers often did not desire, or have the time, to ask their students to solve two adventures that related to the same topic. Instead, many wanted to move to Jasper adventures that focused on other topics like introductory statistics, geometry and algebra. As a consequence, we faced the challenge of designing a series of adventures that could be used flexibly yet also overcome impediments to transfer.

We found that we could increase the flexibility of students' learning by having them solve a series of "what-if" problems after solving a Jasper adventure (CTGV, 1997). For example, the challenge of RBM is to find the fastest way for Emily to rescue an eagle and to explain how long that will take (most students' solutions make use of an ultralight that is shown in the adventure). After students solve the initial challenge they complete what-if problems that ask questions such as: What if the speed of the ultralight was *x* rather than *y*, how would that have affected the rescue? What if the fuel capacity was *a* rather than *b*, how would that affect your current plan to rescue the eagle? Flexibility and the ability to transfer also increase when students solve analog problems like Lindbergh's flight from New York to Paris. (His planning for the trip is very similar to the planning required to rescue the eagle in RBM.)

We used videodisc technology to deliver the what-if scenarios and analog problems. A limitation of this approach was that we could only present a limited set of "canned" problems. In addition, our computer environment did not allow us to present feedback to students about their planning and thinking—this had to be left to the teacher. When one is teaching 20 to 30 students, providing "just-in-time" feedback is an extremely difficult task. In addition, we found that students liked to explore their own what-if scenarios and even create scenarios for others. This was not possible with the "canned" problems that we had used.

Efforts to develop a more flexible, feedback–rich environment brought us to our first attempt to use some of the techniques made possible through AI. The result was AdventurePlayer, developed initially by Thad Crews and Gautam Biswas in consultation with the Jasper group. It allows students to work either alone or in groups to attempt to solve a Jasper problem and what-if analogs, and to see the effects of their efforts via a simulation (figure 4). For example, if they have not accounted for fuel needs, the plane has to land prematurely or it crashes. Students can then go back and revise their planning. If they get terribly stuck, they have access to a coach. AdventurePlayer is designed both to facilitate initial learning of each Jasper adventure and to promote flexible transfer with "what-if" scenarios.

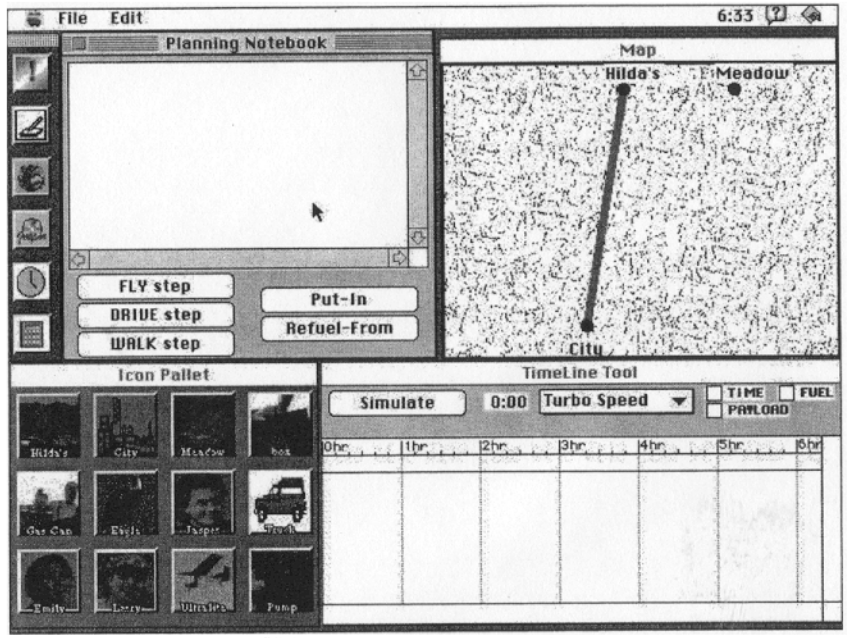

Figure 4. AdventurePlayer interface.

AdventurePlayer combines features of intelligent tutoring systems (ITS) (Wenger 1987) and cognitive tools (Lajoie and Derry 1993). The system provides an intelligent simulation environment that enables students to test partial (and complete) solutions to problems and receive feedback. Unlike traditional ITS (e.g., Anderson et al. 1990) the focus is not on student modeling (i.e., discovering the students' misunderstandings) or how their errors should be corrected. The feedback mechanisms borrow from the cognitive tools framework in which the emphasis is on highlighting important aspects of the domain and the problem solving techniques. In AdventurePlayer the feedback is designed to make explicit the consequences of errors and incompleteness in defining and executing solution steps. For example, if a student fails to check the payload on the ultralight, and the total weight is too great, the ultralight fails to take off. Similarly, if the student specifies an excessive flying time from the city to Boone's Meadow, the ultralight flies past its destination. We have tried to stage the feedback so students can readily map explicit failures onto the relevant parameters of their plans. Ideally, this helps students learn to identify, reflect upon, and correct their own errors. However, if students continually experience difficulty, they also have access to the coach (we say more about this below).

AI techniques and representation mechanisms also underlie a suite of Ad-

venturePlayer tools that assist students in their problem solving efforts. For example, the environment includes an information pallet that enables students to access information about people, locations, vehicles, and distances that are part of the video adventure. There is a planning notebook that students use to sequence their solution steps while considering the resources they need to make their solution steps work. There is also a timeline tool to help students organize their solution steps, and to assign start and completion times for each of the steps.

The planning notebook and timeline tool present important representational structures that we hope students will appropriate as their own. They also serve as scaffolds by simplifying the planning and calculation tasks that may otherwise overwhelm students in their problem solving. Our pedagogical strategy is to offer scaffolds that permit students to freely explore the problem space and experience complex problem solving without excessive floundering. We gradually remove the scaffolds as students move on to solve analogous "what if" problems and other related adventures.

Overall, the intelligent simulation and its suite of tools provide students with an exploratory environment for guided discovery learning. Such environments have been criticized because they can frustrate students who cannot recover from errors during problem solving or who get stuck at a plateau of performance. This is why we have also implemented a coaching system (cf. Burton and Brown 1979) that observes student problem solving, and intervenes to make suggestions at specific points in the problem solving process. The coaching system is useful because the task of generating working plans in the RBM domain is complex for middle school students, and in many cases they generate incomplete plans (Van Haneghan et al. 1992). The trip planning coach helps move students to the next level; for example, by assisting them in generating an optimal solution to the trip-planning problem once they have a feasible plan to rescue the eagle. The reasoning engine of the coach employs a generic algorithm that combines hierarchical planning and best first search. A small set of predefined heuristics guide the search process. After a student generates a complete solution, the coach intervenes to ask if the student could find a better solution. If the student says no, the coach compares the optimal solution to the student's solution, and based on the differences, makes a number of suggestions to the student directing him or her toward a more optimal solution. Details of the coach and the AdventurePlayer system appear in Crews et al. (1997).

Data show that the use of the Jasper AdventurePlayer software greatly facilitates students' abilities to solve RBM (Crews et al. 1997). For example, we know from early studies on Jasper that students working alone can have difficulty solving the challenges (e.g., CTGV 1997, chapter 4; Van Haneghan et al. 1992). Their performance levels are higher when teachers guide the learning process and when they can work collaboratively in groups (see CTGV, 1997). However,

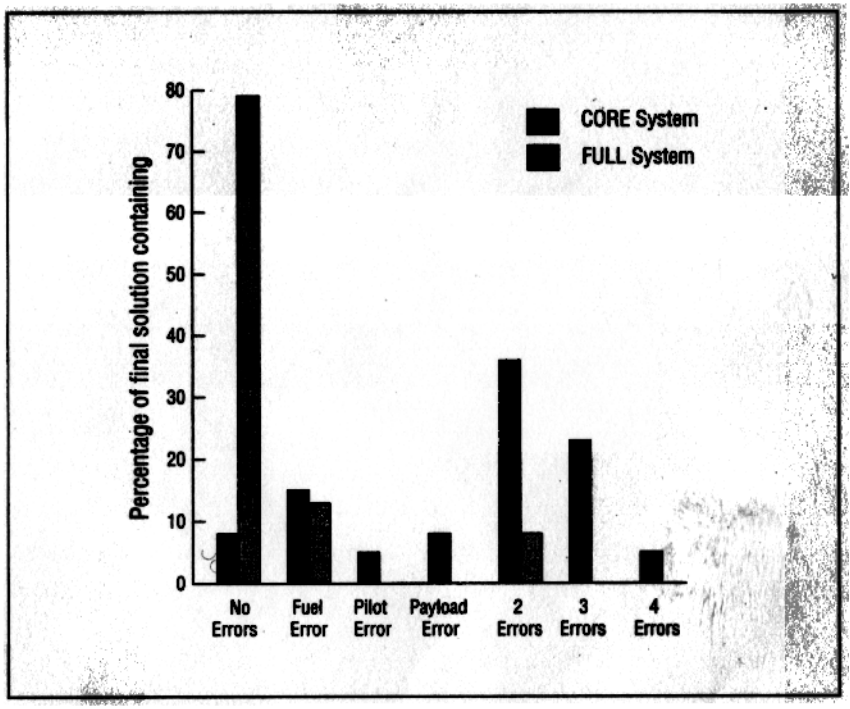

Figure 5. Student gains using the full AdventurePlayer system versus a core system that only includes the planning tool and the information pallet.

teachers often have difficulty managing the complexity of helping each student learn to solve each adventure. AdventurePlayer is very effective in managing this complexity.

In one study we tested AdventurePlayer by asking sixth grade students, who had no prior experience solving Jasper problems, to solve the problem by themselves. From previous studies (e.g., Van Haneghan et al. 1992, CTGV, 1997), we knew that even students who scored very high on standardized tests of mathematical skills and understanding were not prepared to deal with Jasper-like problems; instead, they were used to dealing with simple one- and two-step word problems. So we wanted to see if AdventurePlayer could help the students succeed. Overall, students performed much better in the AdventurePlayer environment than when solving a Jasper adventure on their own.

In our study, we also varied the support provided by AdventurePlayer by turning off some of its key features and comparing students' learning in these situations with ones where all the features were operable. Figure 5 shows that the full system led to fewer errors in sixth-grade students' final RBM solutions compared to a partial system called the CORE system. The CORE system simply provided students with the map interface, information pallet, and planning

tool, but not the simulation environment, timeline tool, and coach. Additional data showed that seventy-nine percent of the students who used the complete simulation environment generated a complete plan compared to only eight percent of the students who used the CORE system.

In more recent work (Katzlberger 1998), the AdventurePlayer design has been generalized using a generic object-oriented architecture for problem solving with visual interfaces that retain the pedagogic characteristics of the original AdventurePlayer system. The environment uses Java internet foundation classes (Netscape) making it accessible remotely via the Web. Domain experts can build problem-solving environments as subclasses of the abstract simulation class. The simulation contains and displays actors implemented as "agents" that perform roles based on properties assigned to them. Class libraries provide support for implementing many different kinds of actors.

This framework has been extended by introducing agents that add additional "intelligence" to the environment and to the interaction between the student and the environment (cf., Lester et al. chapter 8). Agents come in various forms. Some are assistants that help users retrieve relevant data and remind them of salient features about the domain. Others watch user actions to keep track of their preferences and their progress in problem solving tasks. Still others act as coaches to help students who make repeated errors or generate suboptimal solutions with their problem solving steps. Agents have been designed to take on characteristics of some of the actors in the Jasper adventures (Balac, Katzlberger, and Leelawong 1998). As a result, students encounter multiple interaction styles and a variety of opportunities for learning.

One of the major goals of recasting AdventurePlayer in an object-oriented architecture has been to support a reconceptualization of the Jasper series—a reconceptualization that emphasizes the importance of invention and modeling in order to "work smart" in particular environments. As an illustration, note that the Jasper Adventure *Rescue at Boone's Meadow* (RBM) asks students to solve a single problem—rescuing the eagle (see figure 2). The what-if analogs to this adventure change the parameters of the problem, but it's still the same problem. The Lindbergh analog adds another problem, but it is still a single problem involving flight time and fuel from New York to Paris.

Problem-solving in the Jasper series is reconceptualized when one moves from attempts to solve one or two complex problems to attempts that prepare one for a large class of recurring problems. This is done by inventing tools that allow one to "work smart." For example, a "working smart" extension for RBM involves Emily (the heroine who saves the Eagle) setting up a rescue and delivery service where she flies into various areas in her region depending on the needs specified by customers. She and her employees need to be able to tell the customer—as quickly as possible—the type of plane needed (depending on payload constraints), the flying time for the trip (depending on the speed of the plane), the fuel charges (which vary by plane size), and so on. To calculate

these anew for each problem is cumbersome. Students learn to work smart by inventing tools like graphs, charts, and spreadsheets that help them solve these problems at a glance. Examples of smart tools, and of the learning processes and motivation involved in creating, testing and using them, are discussed in Bransford, Zech, et al. (2000).

The new object-oriented architecture of the Smart environment makes it easy for students to transport tools, like rate-time-distance graphs, from one working smart environment to another. This helps students see both the generality and possible limitations of their representational tools as they move to new environments. For example, a spreadsheet is more likely to be generally useful than a particular graph or chart, but students may still prefer the latter, once they tailor it to a specific set of constraints in a new environment (see Bransford, Zech, et al. 2000).

The object-oriented architecture has also facilitated the development of an AdventureMaker environment. AdventureMaker lets students create their own simulated problems for other students to solve. A current version of the environment provides students with a map background on which they can construct locations, paths for travel between locations, and a number of vehicles. The system allows students to create a variety of challenge problems that include RBM-type planning problems, overtake and catch up problems, and even versions of the traveling salesman problem. Students responding to the challenge have to import tools they created in other problem solving environments, tailor them for the particular problems, and then find the best solution as quickly as possible. We find that this is highly motivating for students, and that prior experience with the Jasper series affects the quality of the problems they construct (see CTGV, 1997). One of our hypotheses about why AdventureMaker activities are so motivating to students is that it puts them in the position of creating programs that others can learn from (see also Kafai et al. 1998). This fits the general idea of "learning by teaching," which is discussed below.

Creating Designs That Give Students More Responsibility by Asking Them to Teach

Over time, our work in classrooms helped us see that some types of activities were consistently motivating to students and helped them appreciate feedback and opportunities for revision. These involved cases where students were preparing to present their ideas to outside audiences (e.g., adults), preparing to teach others to solve problems that they had learned to solve previously (e.g., college students; see CTGV 2000; Bransford, Brophy, and Williams 2000), and creating new problems to present to other students (see the preceding discussion of AdventureMaker). These observations led us to consider the value of

using AI-techniques to create intelligent social agents (teachable agents) for whom students could take the responsibility of teaching.

We differentiate teachable agents from other agent technologies such as agents that coach people as they learn new skills and knowledge, or agents that search the web for a user and gradually learn preferences to tailor more efficient searches. Our agents currently have no automated learning algorithms built into their learning processes. For us, teachable agents are social agents who need explicit instruction to do well. Students provide this instruction to the agent as knowledge structures or procedures that can be directly executed by the agent in response to given queries and problems. For example, an agent may need to be taught how to deal with trip planning problems like the "working smart" version of RBM, or an agent may need to learn how to monitor the quality of rivers to discover evidence of possible pollution. We situate our agents in particular task environments that provide a focus for teaching and assessment that is more domain targeted than simply asking students to "teach this agent to do something—anything." The agents do poorly or well in these task environments depending on how well they are taught.

A number of factors motivated us to explore the idea of creating "teachable" agents whose behavior would depend on the quality of the teaching provided by students. One is that the challenge of teaching others appears to create a sense of responsibility that is highly motivating to individuals of all ages. In a study that interviewed sixth graders about the highlights of their year as fifth graders, doing projects that helped the community and tutoring younger students received the highest praise from the students (CTGV, 1997). In "reverse mentoring" studies headed by Kay Burgess, inner-city students who had solved a Jasper adventure were highly motivated to help adults and college students solve an adventure for the first time (see Bransford, Brophy, and Williams 2000). In work with teachers, we consistently find that the opportunity to teach their peers is highly motivating and develops a strong learning community among the teachers (CTGV, in press).

The motivation to teach others also carries over to virtual environments. In the newest release from the Little Planet Literacy Series that the Learning Technology Center at Vanderbilt helped create (CTGV, 1998), students are highly motivated to write letters to a character named Maria to help her learn to read (see *The Dougout Collection* Sunburst, 2000). In early versions of our SMART Challenge series (Barron, et al. 1995; Vye, et al. 1998), students eagerly wrote e-mails to virtual students who asked for help in solving the Jasper Adventure that they were working on. In video games, students are consistently motivated to affect the fate of agents as they attempt various adventures. However, the fate of these agents usually depends on physical and mystical powers. We want to change the paradigm so that their fate hinges on the development of useful knowledge, attitudes and skills.

A second reason for exploring the idea of teachable agents stems from the

Task	Preparation for Teaching	Preparation for Testing
Students Initial Framing of the Preparation Task		
Consider larger context of studies	100%	0%
Must memorize details	0%	50%
Spontaneously Questioned Purpose of Experiment (percent of challenged experiments)	92%	33%
Spontaneously Mentioned Flaw in an Experiment	83%	17%
Spontaneously Mentioned Alternative Experiment	50%	0%
Successfully Graphed Experiments Afterwards (successful graphs)	89%	56%

Table 1. Comparative effects of preparing to teach versus preparing to take a test on students' emphasis during study and final understanding.

strongly shared intuition that attempts to teach others is an especially powerful way to learn. There is research literature on learning by teaching. Research on mentoring has shown that tutors learn as much or more than tutees (Webb 1983), and that lessons in which students tutor each other are beneficial (King 1998) especially if well-scaffolded (e.g., reciprocal teaching; Palinscar and Brown 1984). Nevertheless, the research on learning by teaching does not consistently show a benefit for teaching over and above learning for oneself (Bargh and Schul 1980; Willis and Crowder 1974; Cohen, et al. 1982). We believe that some of this has to do with a lack of a research base about where to look for the benefits of teaching. It seems unlikely that the unique payoff of teaching would appear in memory tests, although memory tests are typically used. It seems more likely that the payoff would be in the structure of people's knowledge and their readiness to learn from further instruction and feedback (Bransford and Schwartz 1999). In addition, there are many aspects of learning by teaching that have not been explored.

We identify at least three phases of teaching that might be expected to enhance learning: planning to teach, explaining and demonstrating during teaching, and interpreting the questions and feedback that come from students

during and after teaching. Research has concentrated on the effects of planning to teach. For example, Bargh and Schul (1980) found that people who prepared to teach someone else to take a quiz on a passage learned the passage better than people who prepared to take the quiz themselves. Chi, deLeeuw, Chiu, and LaVancher (1994) showed benefits of explanation, even when there was no audience. For example, directions to "self-explain" while reading a passage on the heart improved comprehension relative to students who simply studied without directions to self-explain.

In our initial studies on preparing to teach, we are finding benefits that suggest preparing to teach can spontaneously affect the way students learn and self-explain. In a recent study, we videotaped twelve students separately as they studied a psychology article that described a series of experiments on memory. Half of the students heard they were studying in preparation for a class test, and the other half heard they would have to teach their class about the article. They had up to thirty minutes to prepare.

The teaching students spent twice as much time studying the article. More interesting is the way they prepared and what they learned. Table 1 presents some of the important contrasts. Students who prepared to teach spent a substantial amount of time trying to understand "the why" of the studies, whereas the students who prepared for the test tried to memorize the results of the studies. As a consequence, the latter were less successful at reconstructing the studies, their results, and their rationale (see table 1).

Written reflections collected by X. D. Lin and J. D. Bransford also show benefits of learning by teaching. They asked different groups of graduate students in a class on cognition, culture, and technology to teach the undergraduates in the class about some articles on stereotypes that included empirical tests of various theories. Some of the graduate students worked individually to prepare to teach, and others worked in small groups.

After their teaching experiences, the graduate students were asked to discuss any benefits of being asked to teach the material to the undergraduates (compared to simply studying the materials in preparation for a test). All the graduate students were convinced that preparing to teach, plus actually doing the teaching, resulted in levels of learning that exceeded what they would have experienced if they had only studied for a test.

Planning to Teach: Some students focused on the fact that the responsibility of teaching forced them to make sure they understood the materials:

> The article that described the three theories was very hard for me to read.... I had to be sure that I understood this article.... After my clear understanding of how the theories work I was then able to prepare a comprehensive and compact presentation.... In short, I cannot prepare a presentation on something I do not understand.

Other students focused on the increased importance of a clear conceptual organization:

> To teach something in a specified amount of time means that you need to be able to differentiate what is important from what is less important and identify component parts and relationships. In other words, it's necessary to conceptualize the framework of ideas that are presented in the article. When I read an article with the mind set of discussing it, I am not so diligent about understanding the hierarchy of the ideas presented. If I'm teaching something, I have to categorize and prioritize the ideas. This insures I present the important ideas in a coherent manner.

It seems clear that the planning that takes place depends on knowledge of one's audience and the constraints of the teaching opportunities. For example, the graduate students in the Lin and Bransford course had a good idea of their audience (the undergraduates in their course) and the time constraints on their presentations, and these seemed to affect their thinking. Similarly the data illustrated in table 1 involved graduate students planning to teach other graduate students. The students relied on their knowledge of their audience to prepare for the types of questions they might receive. But it is worth noting that all students envisioned giving a lecture rather than teaching in some other, more interactive manner. Moreover, students in primary and secondary school may not readily anticipate the demands of teaching, or their expectations, may be unduly constrained by the experiences that they have had in classrooms (e.g., prior experiences with lectures). This suggests that it is worthwhile to explore different ways to set the preparation "stage" for students rather than simply ask students to prepare to teach.

Learning During the Act of Teaching

We noted earlier that the advantages of learning by teaching involve more than just planning to teach. For example, there would appear to be additional advantages from the actual act of teaching—especially from the opportunity to get feedback from one's students' about what they do and do not understand. In a review of the literature on self-explaining and explaining to other people (as might occur during a collaboration), Ploetzner, Dillenbourg, Preier, and Traum (1999) conclude that feedback from a collaborator is a significant component of other-explanation, yet its effects on learning have not been investigated.

The graduate students in the Lin and Bransford study at Vanderbilt University spontaneously noted some of the things they learned by actually attempting to teach their subject matter. The following quote comes from a graduate student who collaborated with three colleagues in order to prepare to teach the undergraduates. The student discusses the preparation process and then writes about his group's actual attempts to teach.

> We had a list of ideas and feelings that we wanted the class to experience with us. When it came time to present, however, I realized how difficult it was to explain the emotions of our small group discussion to a large group who were not all that familiar with the article. Our presentations became more a dissemination of facts

instead of a sharing of emotion, as I had hoped (and planned) that it would. I think the undergraduates got our basic points and left having a better understanding of stereotypes and our charge to open up to each other, but I don't feel that they had the same experience with the message that I had. I feel my experience was more intense and more memorable because my small group took the article down to its bare issues and discussed how those made us feel and think about stereotypes.

The student's comments suggested that the goal of planning in order to teach helped his group learn effectively. In addition, the act of teaching helped him experience the differences between merely "transmitting" information and helping people experience the effects of their own stereotypes.

Despite the benefits, it seems clear that not all teaching experiences will create significantly new learning on the part of the teacher. Most teachers are familiar with pupils who make them think and learn and with pupils who do not. This is one reason why the idea of creating AI-based teachable agents can be so valuable; it allows us to provide students with teachable agent pupils that optimize their chances of learning, for example by ensuring the teachable agent asks questions most relevant to the domain and the student's level of development. And, unlike peer tutoring, the teachable agent is not hurt if its teacher is really quite bad.

The idea of teachable agents has its precursors in activities such as teaching the "turtle" to do things in LOGO (Papert 1980). This is a very motivating task environment; we have worked with a large number of middle school students in this context and know how motivating it can be (e.g., Littlefield et al. 1988). However, it is also clear from the literature on Logo that it can be very difficult to demonstrate clear advantages of these kinds of activities unless one structures them around particular types of goals and feedback structures (see Klahr and Carver 1988; Mayer 1988). Our approach to teachable agents differs from LOGO in the sense that we situate our agents in particular, anchored task environments, which require specific sets of knowledge, skills and attitudes in order for the agents to succeed. An example of a teachable agent is discussed below.

Meeting a Teachable Agent in Its Native Context

An example of a teachable agent is Billy Bashinall, high school student (figure 6). He and friend Sally have been monitoring a local river to test for water quality. Billy is ready to hand in the report, which says that the river is in excellent shape. Sally is not so sure that their findings are accurate. She worries that the river is polluted and will eventually kill the fish and other aquatic life. Billy's response is, "Lighten up Sally. This is only a school assignment. Besides, five pages is always good enough for a *C* in Mr. Hogan's class." Billy's negative atti-

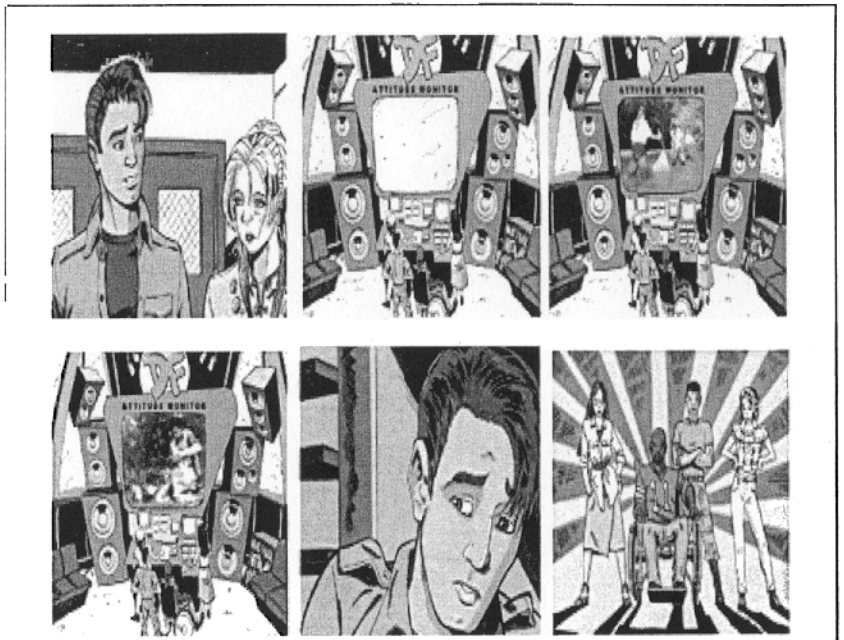

Figure 6. Billy, a teachable agent, learns that his work is not good enough, and the D-Force challenges students to teach Billy.

tude comes to the attention of the D-Force—a group dedicated to helping students avoid the mistakes that they made as students. Billy needs their help.

Billy is transported to D-Force headquarters, where he is shown videos of students monitoring a river by collecting macroinvertebrates, calculating a water quality index, measuring dissolved oxygen and so forth. Billy is asked to explain what is happening. His answers reveal that he understands some things (e.g., macroinvertebrates can provide an index of the health of the river). However, he seriously misunderstands other things like how and why some types of macroinvertebrates need considerable oxygen and are more sensitive indicators of water quality than other macroinvertebrates that need less oxygen. Consequently, he does not understand why the formula for water quality weights some macroinvertebrates more than others. And he does not understand how the amount of dissolved oxygen in the water is related to water quality and pollution.

The episode ends with Billy realizing that he needs help, and the D-Force asking people to help teach Billy. This becomes the task of the students in the classroom. By adjusting Billy's attitude and helping him learn important skills and concepts, students learn by being teachers. By seeing how Billy performs

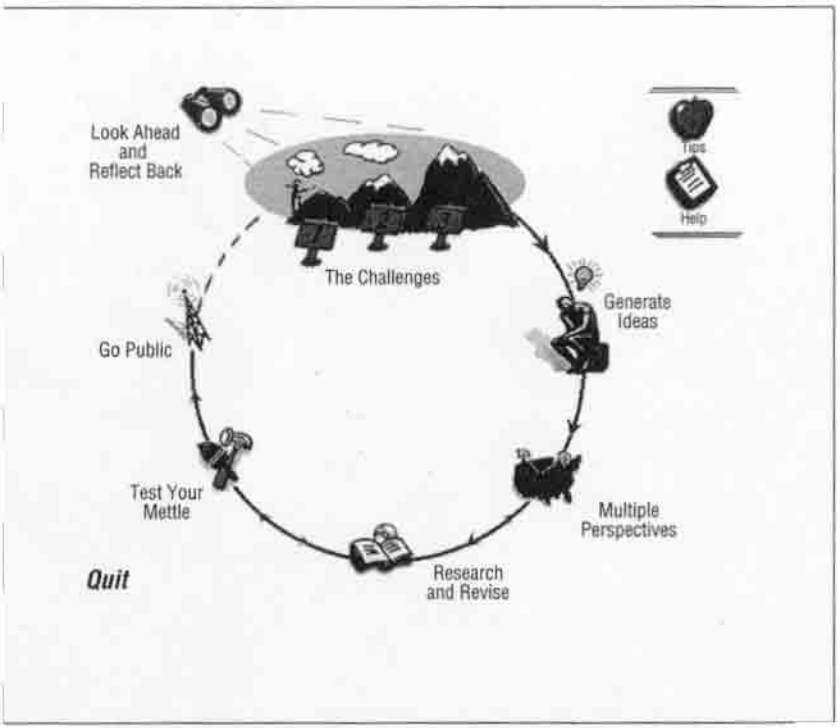

Figure 7. The STAR.legacy interface that organizes complex inquiry activities.

following their teaching, students can assess the quality of their teaching. After a series of iterative teaching-assessment cycles, students eventually learn about rivers as ecosystems and see that Billy has learned, too.

Initial SAD Studies Using Teachable Agents

As was true with our work on Jasper, our research on teachable agents began with a SAD approach. In particular, we began with a pre-scripted agent in order to test our ideas in fifth-grade classrooms. Over time, we are replacing our scripted agents with ones that have intelligence.

In a recent set of experiments led by Nancy Vye, fifth grade students began their inquiry by meeting "Billy Bashinall," the character introduced earlier (figure 6). Students saw him attempt to perform in a particular environment that required knowledge of ecosystems and water quality. They did research in order to teach Billy, and they observed the effects of this teaching on his behavior. The Billy Bashinall environment was a scripted environment in the sense that we pre-specified everything beforehand.

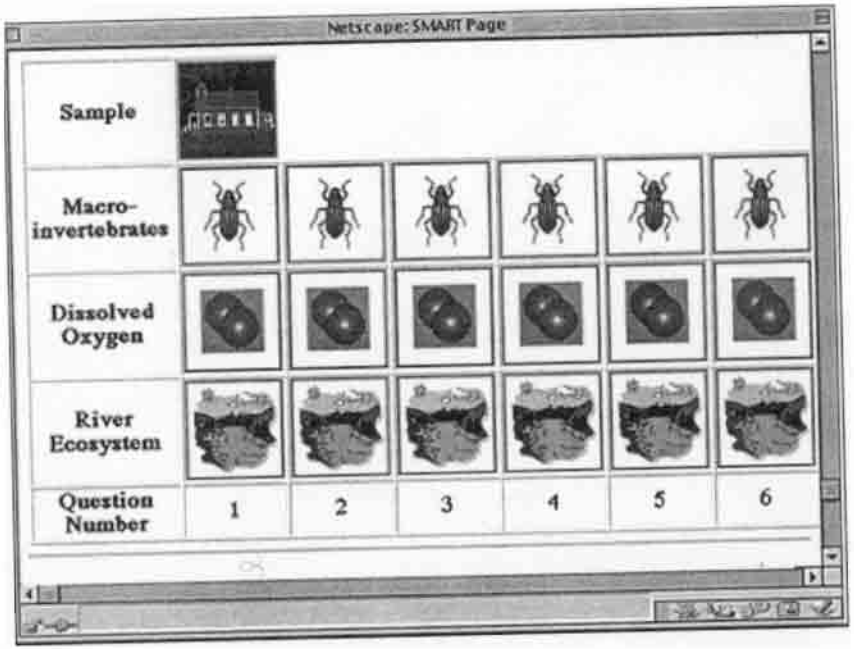

Figure 8. Formative assessment interface.

We used a software program called STAR.Legacy to organize students' inquiry into water quality monitoring (Schwartz et al. 1999). STAR.Legacy is a multimedia shell that helps teachers, students, and instructional designers manage complex inquiry activities. The interface shown in figure 7 organizes different stages of inquiry: students click on the icons to access relevant multimedia content. Students begin with a Challenge that, in this case, asks students to help Billy learn about macroinvertebrates and why they are used to check water quality.

Using the Legacy framework, students then generate their own ideas on the topic of macroinvertebrates (generate ideas) and hear some ideas from experts (multiple perspectives). In the course of discussing these ideas, students generate many questions about macroinvertebrates, which set the course for their research (research and revise). Students also have the opportunity to access specially-designed computer tools to help them learn; for example, they use a simulation in which they learn how to sample and sort macroinvertebrates to check water quality.

After studying macroinvertebrates, students take an assessment in *Test Your Mettle*. Figure 8 shows the software interface that organizes the assessment questions (the software was developed by Jay Pfaffman). By clicking on a square, students are brought to a question from the D-Force that Billy has

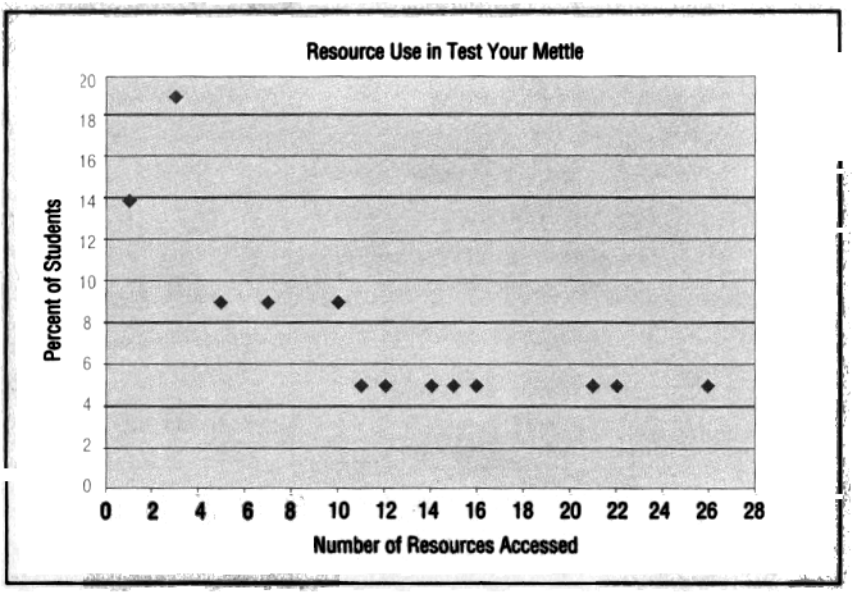

Figure 9. Students consulted resources numerous times in preparation for teaching Billy.

already tried to answer. Students give "advice" to Billy on how he should have answered the question. For example, one of the questions on macroinvertebrates begins, "The D-Force asked Billy to explain how scientists use macroinvertebrates to check water quality." Students then see Billy's response to the question, in this case, "Scientists count the total number of macroinvertebrates that are found in their sample." Students are asked to teach Billy by either telling him that his answer is correct or by choosing a better answer for him from a set of alternatives that we provide. If the students select correct advice, the relevant square in the grid turns green. If they are wrong, the square turns red. The teacher (and class) can look at the accuracy of the students' advice to Billy to get an assessment of class understanding. Although the students believe they are assessing and remediating Billy's understanding in *Test Your Mettle*, they are actually completing assessment activities relevant to their own understanding.

One of the goals of the assessment environment is to help students self-assess whether they are ready to advise Billy, and if they are not, to learn to consult resources. Before advising Billy, students have the option to see online resources such as relevant readings or animations By structuring the environment so students may learn during an assessment, we encourage students to be reflective about whether they understand well enough to teach Billy. The system's backend database tracks student use of resources. Among other things, this feature provides a measure of student motivation to teach Billy; if students are motivated to give Billy good advice, we assume that they will consult resources when they are uncertain or want to verify their advice. Figure 9 suggests that students are highly intent on teaching Billy well. All students from the class represented in the figure spontaneously consulted resources at least once before advising, and most consulted multiple resources. As noted by a fifth grader, "It feels good to help somebody out even though they are computer animated."

In *Test Your Mettle*, students give advice on all aspects of water quality monitoring, even aspects that they have not yet studied. They give advice on issues related to macroinvertebrates and also on dissolved oxygen and river ecosystems. In this way, the assessment is formative and students can use it to self-assess what they have learned or have yet to learn. Once students have completed their teaching in *Test Your Mettle*, we give them class-level feedback on their performance. The feedback shows average performance for the class on each water quality topic.

Having "taught" Billy in *Test Your Mettle*, students watch (in *Go Public*) a cartoon vignette in which Billy answers questions from the D-Force. Billy's behavior is scripted and it is not closely contingent on what the students actually taught Billy in *Test Your Mettle*. Instead, it is contingent only at a general level. For example, if the class as a whole does relatively well in *Test Your Mettle*, the subsequent vignette shows Billy successfully answering the set of questions that were the focus of attention in that particular legacy cycle. If the class did very poorly, the "go public" video shows the D-Force asking the students to try again.

We created a different vignette for each challenge in the legacy cycle. The vignette for challenge one on macroinvertebrates shows Billy doing a good job answering questions about macroinvertebrates, but the D-Force makes it clear that he still does a poor job with questions about dissolved oxygen and ecosystems— topics that students had not yet studied and that we can reasonably assume students will not know much about. At the end of challenge 2, which focuses on learning about dissolved oxygen, students see an epilogue in which Billy is now able to answer questions about dissolved oxygen (and macroinvertebrates), but is still unprepared to answer questions about ecosystems. In this way, we create the idea that student performance in *Test Your Mettle* changes Billy's ability to answer queries from the D-Force.

One important goal of our research is to study whether asking students to teach an agent is effective in helping them learn. Our findings show that over time, students improve their performance on the assessment. And, these improvements occur in a predictable time course. After each challenge, students improve most on the topic related to that challenge.

Bringing Agents to Life with AI

As noted earlier, the long-term goal of our work is to use insights from AI to "bring Billy to Life." Our major focus is not to make Billy learn on his own through inductive, machine learning methods. Billy is a teachable agent not a learning agent. Our goal is to create an environment that allows students to learn both by preparing to teach agents, and by observing how the agents behave once the teaching is completed (if there are problems, students can revise their teaching). We are building teachable agents that can respond to different forms of teaching (complete and incomplete) in flexible and informative ways so that students may learn.

A simple example of this approach comes from *Betty's Brain*, which we developed for instruction in the life sciences. (Betty is a second member of the Bashinall family.) Betty, as an agent, is able to execute and portray inferences implicit in a semantic network. Students and teachers can ask Betty questions as a way to assess the quality of her knowledge. To develop her semantic network, students teach Betty by drawing concept maps and taxonomic trees. In their drawings, students define entities that are of interest in pollution studies (e.g., sunlight, carbon dioxide, dissolved oxygen, algae, fish, etc.). And, they define the relations between these entities (e.g., produce, breathe, and so on). Figure 10 illustrates a concept map that was created by a student early in his learning. In contains misconceptions that are quite typical for middle school students. After students create their concept map, it serves as Betty's semantic network and permits her to draw inferences. In this way, students can get feedback on the quality of what they believe and have taught. Given the map in figure 10, the agent offers many mistaken answers that the student then has to explore and correct.

One of our tasks is to make sure that the effort of teaching an agent does not incur the overhead of learning to program. For example, students do not need to teach Betty how to draw inferences with a concept map. She already knows how to do that. Moreover, the "programming" the students perform occurs in the form of manipulating representational tools, like concept maps, which students need to learn in the course of their normal studies. (Novak 1998) Similar to asking students to develop their AdventurePlayer plans with the timeline tool, we scaffold student acquisition of useful tools and concepts by asking them to teach using "smart tools," in this case, networks that help organize complex declarative knowledge.

Figure 11 shows an interface that students can use to build Betty's network. To teach Betty, students can select from a list of entities relevant to a river ecosystem using a pull down menu. They can also create new entities not available in the menu. Students must also construct relations among the entities they select or create. Students can use relation names from a pull down menu or create relation names of their own. When they create new relation names,

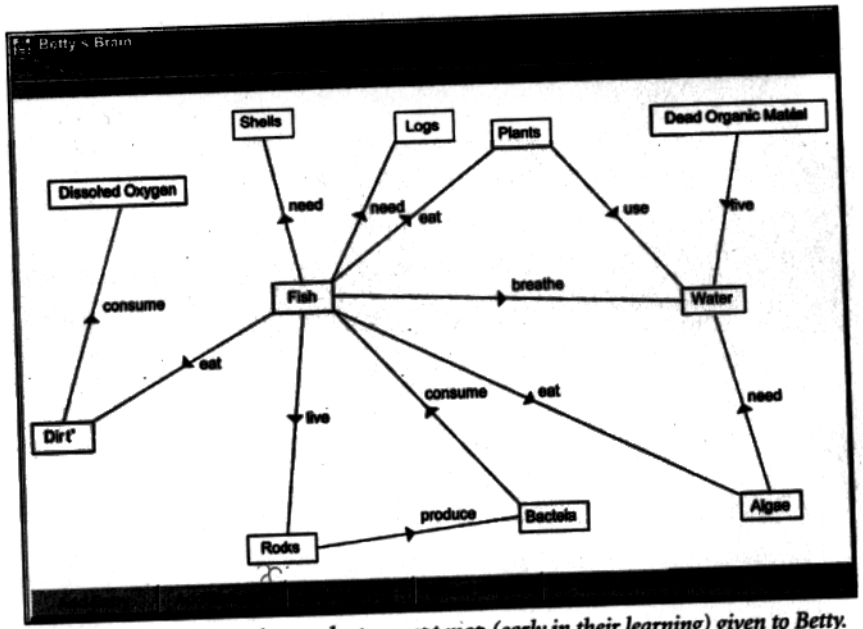

Figure 10. A representative, student concept map (early in their learning) given to Betty.

they are prompted to categorize them into one of four classes that enable Betty to draw inferences. One type of relation is qualitative causal, where an increase or decrease in one entity can cause an increase or decrease in the other; for example, fish breathe (decrease) dissolved oxygen. For Betty's future inferences, this implies an increase in the fish population results in a decrease in dissolved oxygen. All causal links are defined as an increase (+) or decrease (-) relation between two quantities. For example, if quantity *A* has a produce relation with quantity *B*, then an increase (decrease) in *A* causes an increase (decrease) in *B*. On the other hand, if quantity *A* has a consume relation with quantity *B*, an increase (decrease) in *A* will cause a decrease (increase) in *B*. A qualitative amount relation, defined as a large, normal, or small change, may further qualify the qualitative relations. (If the amount is not specified, a default value of a normal change is assumed.) This way, an increase in quantity *A* may cause a large increase in quantity *B*. This framework allows the inference mechanism of Betty's Brain to implement a simple qualitative arithmetic scheme for propagating amount values through a chain of causal relations in a concept map (see Leelawong et al. 2001 for details). Work done in the qualitative reasoning community (e.g., deKleer and Brown 1984, Forbus 1984a, and Kuipers 1986) define more sophisticated qualitative reasoning mechanisms for dynamic systems that reason simultaneously with amounts and rate of change.

A second type of relation is dependency where one entity needs another without implying quantitative changes; for example, plants need sunlight but

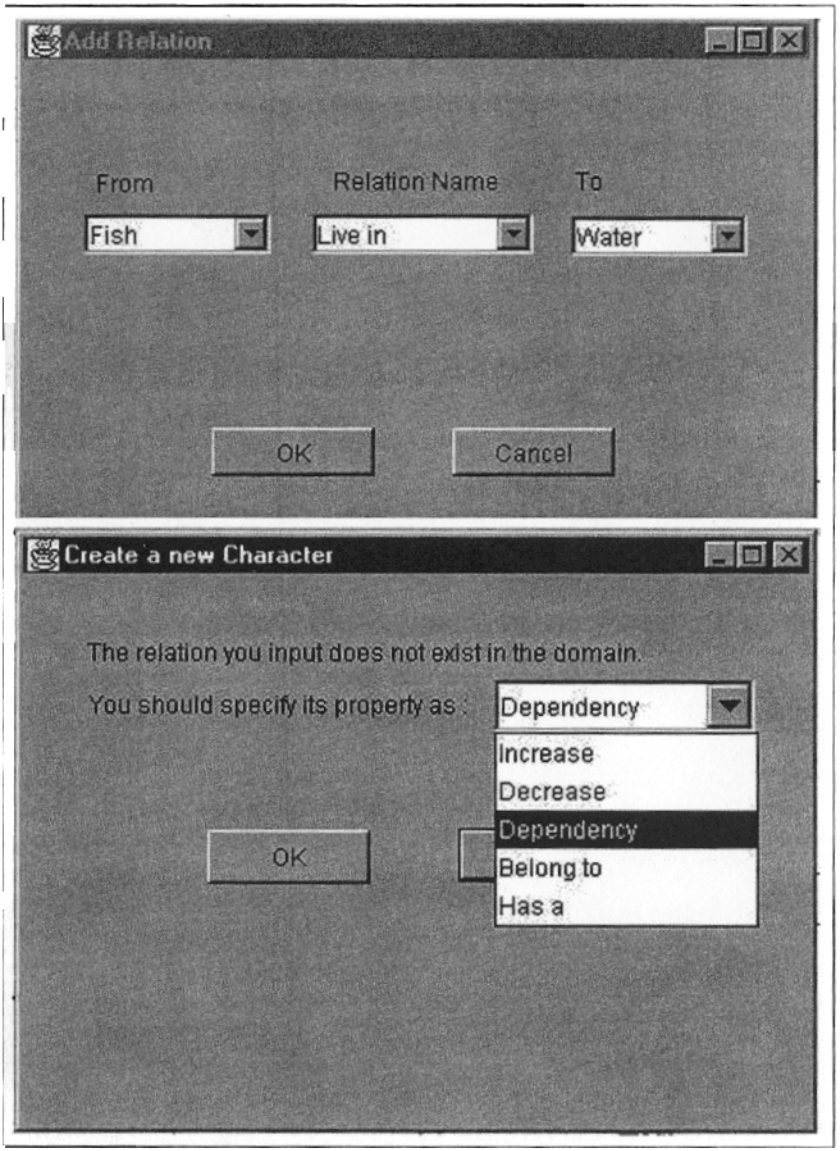

Figure 11. The interface students use to construct Betty's concept map.

do not use it up. A third relationship is belongs-to where one entity belongs to another entity; for example, fish belong to the class of living objects. Finally, the has-a relationship allows students to define entity attributes; for example, water has dissolved oxygen.

After students teach Betty's Brain by building her concept map, they can

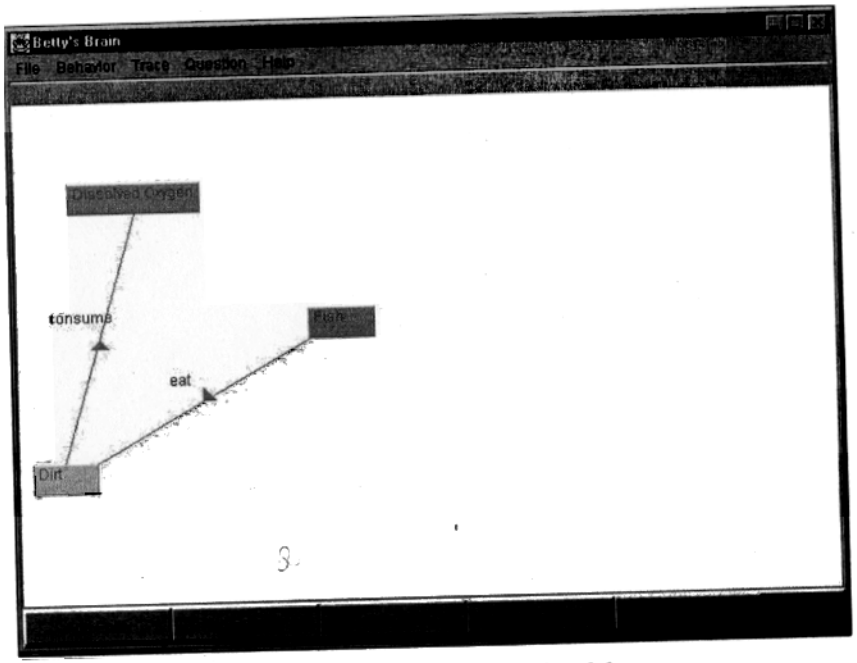

Figure 12. Betty portrays how she inferred that an increase in fish increases the amount of dissolved oxygen.

"test her mettle" by asking questions composed from a pull-down menu. For example, a student can ask, "How does an increase in fish change dissolved oxygen?" In full-discourse mode, Betty uses an exhaustive forward chaining algorithm (a brute force, depth-first search) to find all possible paths connecting the entities in the query. If there are multiple paths with conflicting solutions, the system weights the paths to arrive at a conclusion. To present her conclusion, Betty can explain her answer in stylized natural language, or she can highlight the relevant portion of her concept map. For the concept map in figure 10, there is only one path and Betty replies, "I think an increase in fish increases dissolved oxygen," and she shows her inference path as in figure 12. If there were more than one path, Betty would offer a single answer, but show that there is more than one chain of inference.

In more detail, students and teachers can ask Betty three kinds of questions:

1. What happens to quantity *B* when the amount of quantity *A* increases (decreases)?
2. What is the effect of an increase(decrease) in quantity *A*?
3. What quantities can cause an increase(decrease) in quantity *A*?

For query types 1 and 2, the mechanism does an exhaustive search over the entire concept map, identifying all possible forward paths from quantity A. A single step in the forward search process involves identifying a forward causal

link from the current (source) concept, and applying the qualitative reasoning mechanism to compute the resultant value for the destination concept. When multiple paths exist between two entities, the reasoning mechanism applies qualitative rules of the form, a large increase and a small decrease imply a normal increase, and a normal decrease and small increase, imply a small decrease. Situations, such as a normal increase and a normal decrease lead to an ambiguous situation, and Betty responds by saying that she is unable to tell whether the result is an increase or a decrease. For query type 3, the reasoning mechanism employs an exhaustive backward search instead of a forward search.

In addition to the concept map provided by the student, the Betty system can include an age-appropriate network created by the instructor (using the same interface tools) that can also draw inferences. If needed, these inferences can be used to help students rectify their concept maps. Betty's Brain is similar to AdventurePlayer in that it can operate with and without this type of domain model. The students can simply create and test their own concept maps, just like they can create their own simulations in AdventureMaker. When there is a domain model available, the Betty system, also like AdventurePlayer, can compare the discrepancies between Betty's inferences based on the map provided by a student and the domain model's inferences to determine possible points for instruction. However, unlike AdventurePlayer, the Betty system produces indirect coaching based on discrepancies. We say "indirect" because we make the feedback come through Betty to maintain the student's sense of responsibility. For example, Betty can ask, "I know I figured that an increase in fish would increase the dissolved oxygen, but I remember my teacher said the opposite. Where have we gone wrong?" The system can gradually increase the directness of the feedback depending on student difficulties. For example, Betty can ask, "Are you sure that fish eat dirt?" because the teacher's domain model does not include a dirt entity. At an even greater level of support, Betty can say she asked a teacher or friend who handed her a concept map. She can show the student the map that is the subset of the domain model deemed most relevant to the question.

After students assess and revise Betty's Brain, they ask her to go public. This can take a variety of forms. For example, each student in a class may create a separate Betty. When ready, the students submit their Betty to the teacher's website. (Betty has Java genes. She is implemented as a platform-independent Java applet.) The teacher can then ask the same question of all the Betty's, and students can see the different responses. This serves as an excellent context for classroom discussion as students try to resolve which answers are correct and why. A similar form is that students generate questions for one another that they know their own Betty can answer. A third form of going public involves collaborative problem solving. Under this model, students join their Betty's into team Betty. (The Betty system supports the automatic merging of databases and the redrawing of the subsequent concept map.) Team Betty then tries to

answer questions. In the best of cases, team Betty can answer questions that none of the Betty's could resolve in isolation. In other cases, team Betty contradicts herself, because the students had conflicting concept maps. Our hope is that it will be highly motivating for teams of students who build a team Betty, and it will provide an excellent context for developing classroom discussions about knowledge organization.

One of the exciting possibilities with teachable agents is that they can help students reflect on the dispositions of effective learners. The students can explore how different attitudes affect an agent's ability to learn and reason. For example, students can adjust Betty's "attention" parameter that determines how well she "encodes" the relations she is taught. If her attention is poor, she may put an entity into the wrong relationship or leave it out all together. Of course, the goal is not to teach children that good attention is all that is needed for learning. Therefore, we plan to include a complex of different disposition parameters that students can adjust, balance, and discuss. For example, Betty can be stubborn and refuse to do homework so she forgets some relationships over time.

Students can adjust disposition parameters that affect her reasoning. Although students do not program the search algorithms, they can indirectly change them via Betty's disposition. For example, if they receive a lazy Betty and do not adjust her "thoroughness" parameter, Betty only searches one level or stops after finding one solution. Betty can be too wordy and report every step of every inference chain she tries, right or wrong. Betty can be a little too loose in her reasoning and chain across nodes that have no actual relationship. Students get a chance to explore and change disposition parameters as they work with Betty. Ideally, this activity can help students reflect upon and discuss the appropriateness of their own dispositions for learning.

Summary

Our goal in this chapter was to describe some examples where AI techniques have helped to improve learning in classroom environments. We noted that our research strategy has been to begin with SAD applications of technology. This has allowed us to identify situations where increased technological sophistication could have a significant impact on student (and often teacher) learning. We discussed two examples of moving from stone-age designs to designs enriched through AI.

One example involved creating the AdventurePlayer program to accompany the Jasper Series (Crews et al. 1997). Students can work on the program either individually or in groups. Tests of the program (Crews et al. 1997) show that it facilitates initial learning and leads to more flexible transfer. Teachers who have

taught Jasper with and without the aid of AdventurePlayer have noted how helpful the program was for managing the complexity of the learning environments that Jasper adventures entail.

We also noted that we have created object-oriented variations on AdventurePlayer that allow students to create tools for "working smart," plus create simulations for other students. The "smart tools" application supports the reformulation of Jasper from a series where students only solve a single complex problem, to one where they must learn to work smart in order to solve large classes of frequently recurring problems (Bransford, Zech et al. 2000). The AdventureMaker applications allow students to create new adventures of their own.

A second example of moving from SAD to the use of AI techniques centered around the concept of "teachable agents" whom students explicitly teach to perform a variety of complex activities. Our focus on teachable agents is part of a larger effort to explore the potential benefits of learning by teaching. There is a great deal of intuitive support for the benefits of learning by teaching—both motivationally and conceptually. By contrast, the research literature on the topic is extremely modest. Our research is helping us understand the conditions under which learning by teaching can have powerful effects. We find benefits of planning to teach, as well as benefits of actually trying to teach and getting feedback from students. But we noted that even these categories are very general and need to be explored more carefully. For example, how one plans to teach is affected by knowledge of (1) who will be taught; (2) under what conditions (e.g., time constraints), and (3) the range of possible teaching strategies that might be used (e.g., lecture versus alternatives).

We also argued that people learn during the act of teaching (and when reflecting on their teaching), and that what a teacher learns will depend on the quality of his or her students (e.g., the kinds of questions they ask). And we wanted to ensure that students are not harmed by inexperienced or underprepared teachers. Therefore, we are creating virtual "teachable agents" who are not hurt by poor teaching, and who provide the kinds of questions and feedback that best enable their teachers to learn from them.

Our effort to design teachable agents follows the tradition of viewing computers as not only tools and tutors, but also as tutees (Taylor 1980). Attempts to program the Logo turtle represents an excellent example of the computer as tutees approach (e.g., Papert 1980). However, unlike Logo, we situate our teachable agents in particular environments that require specific sets of attitudes, skills, and knowledge to function effectively. This allows us to focus on the acquisition of conceptual knowledge that is important for areas such as science, mathematics and history.

Our work on teachable agents has followed our strategy of beginning with SAD approaches and adding complexity as needed. Our initial SAD designs involved agents whose behaviors were all prespecified. This work has allowed us

to learn a great deal about student motivation and learning, and about ways to provide feedback to students that facilitate this process. Subsequent work is allowing our agents to be much more flexible by using agent-based AI techniques.

Our work on teachable agents is quite new, so the ideas and data we presented were preliminary. Nevertheless, students' reactions to the "faux" agents have been highly promising, and the AI-based agents are making the learning situations even more exciting. One of our ultimate goals is to change the nature of video games from environments that primarily emphasize weapons and fighting abilities to ones that highlight important sets of knowledge, skills, and attitudes. Combing our teachable agent software with the AdventurePlayer type simulation environments will allow students to design agents for particular challenge environments, and then evaluate and cheer for the superior problem-solving performance of their agents in these environments. As the environments become more complex, the knowledge students will need to teach will become increasingly sophisticated and flexible. In this way, students can develop an appreciation for the "big ideas" that organize thinking in different domains.

Acknowledgements

This chapter was made possible by the Teachable Agents grant—NSF REC-9873520T, and by a grant for The Challenge Zone—NSF ESI-9618248. The opinions expressed in the chapter do not necessarily reflect those of the granting agency. Members of the Teachable Agents Group at Vanderbilt who contributed to this chapter are Gautam Biswas, John Bransford, Sean Brophy, Thomas Katzlberger, Xiaodong Lin, Taylor Martin, Jay Pfaffman, Daniel Schwartz, Nancy Vye, and Yingbin Wang.

Smart Machines in Education

The Coming Revolution in Educational Technology

Edited by

Kenneth D. Forbus and Paul J. Feltovich

2001

AAAI Press / The MIT Press
Menlo Park, California Cambridge, Massachusetts London, England