Teachable Agents: Combining Insights from Learning Theory and Computer Science

Sean Brophy, Gautam Biswas, Thomas Katzlberger, John Bransford, and Daniel Schwartz
Box 45-GPC
Vanderbilt University
Learning Technology Center
Nashville, TN

Abstract. We discuss computer environments that invite students to learn by instructing "teachable agents" (TA's) who venture forth and attempt to solve problems that require knowledge of disciplines such as mathematics, science or history. If the agents have been taught properly they solve the problems they confront; otherwise they need to be further educated. The TA's have both a "knowledge dimension" and a "personality dimension" (e.g., some may be impetuous, not listen or collaborate well, need many examples to understand, etc.). This helps students focus on academic content plus the characteristics of "difficult agents" that interfere with learning. The paper briefly discusses learning by teaching, learning by programming, and relevant classroom research. This background helps identify key principles underlying teachable agent learning environments. The rest of the paper discusses a framework for instantiating these principles into a general teachable agent environment.

1. Introduction

In this paper we discuss computer environments that invite students to learn by instructing "teachable agents" (TA's) who venture forth and attempt to solve problems that require knowledge of disciplines such as mathematics, science or history. If the agents have been taught properly they solve the problems they confront; otherwise they need to be further educated. The TA's have both a "knowledge dimension" and a "personality dimension" (e.g., some may be impetuous, not listen or collaborate well, need lots of examples to understand, etc.). This helps students focus on academic content plus the characteristics of "difficult agents" that interfere with learning.

The TA environments that we discuss are designed to facilitate research on human learning (especially research on the potential advantages of "learning by teaching"). For example, in our environments, students do not know exactly what problem an agent will have to solve. Therefore, they need to think about "big ideas" that will prepare an agent to solve a class of problems. Presumably, teaching about big ideas will facilitate student learning and transfer as compared to teaching specific facts for solving a specific problem [1].

Our TA work is also intended to help us discover and refine computer science techniques for designing agents who can be taught by students and then display the effects of this teaching in their behavior. For example, students do not program TA's with procedural steps. Instead, they use representations common to the disciplines of knowledge we want the

students to learn. So, rather than teaching an agent with a pseudo-code algorithm for computing X given Y, students can construct a graph that shows the functional relationship.

In the following, we briefly review research on the benefits of learning by teaching and learning by programming. We then describe how classroom research led us to investigate the idea of having students learn by teaching agents. Next, we describe our current, and future, approaches to TA's where a student teaches an agent that is a virtual person. We conclude with a discussion of our current research on the development of TA environments.

2. Learning by Teaching

A belief in the value of learning by teaching is widespread. One example involves graduate students who become teaching assistants in areas like statistics and note that teaching helped them really learn. The literature relevant to learning by teaching includes reciprocal teaching [2], small group interactions, self explanation [3], and peer-assisted tutoring [4]. Much of this literature refers to the fact that tutors often learn as much as, if not more than, their tutees [5]. Nevertheless, strong empirical evidence on this point is difficult to find. Some benefits of learning by teaching seem to be that teachers have to anticipate what their students need to learn, that teachers are often confronted by "naive questions" that make them rethink their own understanding, that teachers need to organize their knowledge in clear, consistent, and communicable ways, and that teachers have opportunities to notice the importance of different behaviors for people's abilities to learn. Presumably, when students take the role of teacher, they partake of these benefits. Moreover, interacting with another individual has a motivational component that may not be found in interactions with inanimate instructional materials. People, for example, are more motivated to make sure they get it right if they have the responsibility of teaching it to someone else.

3. Learning by Programming

Learning by programming and the metaphor of computer agents provide opportunities that may yield benefits similar to those of learning by teaching. This work may be divided into two broad classes: research on the benefits of learning by programming, and research on techniques that make it possible for agents to learn.

There is a substantial history to the question of whether learning to program has general intellectual benefits that extend beyond programming [6]. In the context of agents, the idea of learning by programming was emphasized by Papert [7] who helped students learn by teaching a logo turtle (cf. [8]). The ability to improve domain independent thinking skills (e.g., planning) through programming has received mixed support. Consequently, there has been a shift towards domain specific knowledge that students learn as they program the domain knowledge into agents. This idea includes programming lego toys and robots to interact with one another explicitly [9], programming computer agents to collaboratively learn from one another [10], creating micro worlds, and creating software to help others learn topics such as mathematics [11].

Another instructional method is to place the learner in the role of simulation designer. In this method the goal is to define a model by identifying the major factors in a system and to identify rules that govern those factors. Repenning and Sumner's AgentSheets [12], for example, makes it easy to create SimCity type simulations. Factors are represented as agents who each have a local set of rules that define their behavior. The design process resembles an

informal inquiry process where students generate a hypothesis of how a system works, then they translate this hypothesis into an agent's rule base. Students acting as designers learn about the underlying principles of a situation by evaluating how an agent acts with other agents when running the simulation. The design process of hypothesizing, implementing and testing provides an excellent model for learning.

Other work relevant to teachable agents comes from research on how to make agents that can learn. For example, the Persona project at Microsoft ([13]) has focused on agents that learn sophisticated user interactions, communication and social skills. Recent architectures have focused on agents that can learn from examples, advice and explanations [14][15]. These agents learn new knowledge through a range of techniques including natural language, monitoring users actions (demonstration), or learning through mistake correction. Our teachable agents could use similar approaches, but our end goal is different. Our teachable agents only need to "appear" that they are learning from the user. We are creating teachable agents that support student learning, not learning agents.

4. Learning environments that promote action, reflection and refinement.

Opportunities to teach and to program have both demonstrated potential for facilitating student learning. Our goal is to bring these ideas together to design TA computer environments. Our entrance into agent technologies followed a different path from that of research into computer applications or research into learning by teaching per se. Instead, we have come to TAs by way of classroom research where we have found it important for students to have opportunities to develop and assess their understanding and to interact with their peers. The following reviews several key features of our SMART project (SMART stands for "scientific and mathematical arenas for refining thinking) that lead to the design features and principles of our teachable agent project.

4.1 Complex Learning Activities and Opportunities for Frequent Assessments

One way to facilitate learning is to organize activities around anchoring problems that integrate and motivates multiple domain concepts [16][17][18]. Instruction begins with the presentation of a complex challenge. For example, in the video-based Jasper Adventure, Rescue at Boone's Meadow (RBM), students need to design a plan to rescue an injured eagle found in the mountains. Students and teachers identify subgoals for completing the challenge. Each subgoal creates a need to learn specific domain concepts. For example, students need to learn about rate-time-distance relationships for various vehicles (e.g., an ultralight and a car) to determine the optimal route and vehicle combination. Anchoring instruction in a larger problem-solving context helps students understand the value of new knowledge and how to apply it.

To help students develop and self-assess their understanding of concepts, we often provide simulations. We use the larger context of the challenge to help make the simulation more meaningful to students. For example, AdventurePlayer, is a planning simulation that compliments RBM. Students experiment with different plans and receive feedback on the plan's success. The larger context of RBM provides an interpretive framework for understanding the feedback and for making decisions about learning goals. This helps students (and teachers) determine whether they need to revise their understanding.

4.2 Introducing a Teachable Agent

Another way that we have helped students develop the abilities to solve a challenge is with a feature called, Kids on Line [19]. Kids on Line is a direct precursor of teachable agents. Students watched videotapes of students (actors we hired) as they explained their incomplete ideas on how to solve a challenge. The students critique the Kids on Line and provided suggestions for how to improve. Students (and teachers) found this activity extremely motivating, and they had many suggestions for how to improve the Kids on Line. When asked years later, students often spontaneously referred to the value of Kids on Line.

4.3 Representing and Working with Domain Knowledge using Smart Tools

Anchors provide familiar contexts that help students grasp the meaning of new ideas and skills. It is important, however, that student knowledge be given a chance to generalize. We do not want students to learn how to solve a specific rate-time-distance (RTD) problem about the time it takes to fly a plane to a particular location, we want them to be able to solve classes of RTD problems. Thus, we introduce students to the idea of SmartTools. SmartTools refer to the representational methods experts use to organize and make sense of complex information. For example, graphs can quickly illustrate the relationship of a dependent variable and an independent variable. Many of the "anchor" stories we have created attempt to set up situations that motivate the value of representational tools. For example, in one anchor, students must be prepared to make quick computations in real time to help potential clients of a travel firm. At first, they discover that unaided arithmetic can be too slow. Afterwards, they learn that graphs and tables can be tremendous aids. Students receive opportunities to invent their own SmartTools and to learn about tried and true conventions.

Overall, our work in classrooms indicates that students learn well when they have an opportunity to learn in anchored problem-solving contexts that include 1) scaffolds and frequent opportunities for assessment, 2) when they interact with models of behavior, and 3) when they can create SmartTools to generalize their knowledge. As a group, these ideas have evolved into the concept of helping students learn by teaching agents to solve particular sets of problems. One reason we moved to agent technology is because we found students do not always get sufficient opportunities to exercise and test their understanding when working on large projects. In this light, computer environments make an excellent complement to project-based instruction because they can provide increased opportunities for individual exploration.

5. Designing Teachable Agents Environments

Our TA environments fold our classroom research into work on programming to learn. For example, we have found that teaching someone else can be very motivating. Therefore, the agents that students teach are virtual humans. A recent classroom study helps to clarify the potential of teachable agents. In this study, students began their inquiry by meeting a cartoon character named "Billy Bashinall", they watched him attempt to perform in an environment that required knowledge of ecosystems and water quality. The challenges for the students consisted of teaching Billy so he could perform correctly when tested. The students eventually teach him about water quality, how to assess it, and how pollution affects dissolved oxygen and hence life. To meet these challenges, they did research , and were able to observe the effects of this teaching on his behavior. Figure 1 helps clarify this process. It shows drawings

that represent snippets from an 8 minute video anchor that introduces students to Billy Bashinall.

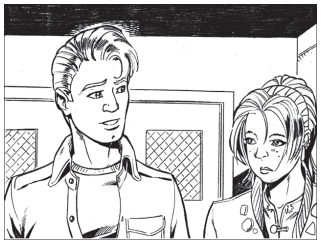
A particularly interesting aspect of this study was that the students showed great perseverance in their attempts to teach Billy. They used resources and revised their own understanding for several weeks without flagging interest. For example, we gave students repeated multiple choice tests. We told the students they were using these tests to determine whether they were ready to teach Billy. Our measures indicate that students did not view the tests as tests, but instead embraced them as opportunities to assess their own preparedness. And, as expected, they showed strong learning gains.

6. Creating Teachable Agent Environments: Computer Science Issues

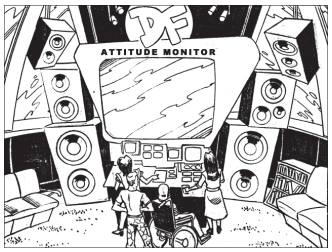
In the preceding example, Billy was not computerized. He was simply a scripted cartoon character shown on video. Classrooms of fifth through seventh grade students voted on the knowledge that Billy should receive, we tallied their votes, and then showed one of a few "canned" Billy behaviors based on the majority of votes. The level of interactivity was low, but this study provides important information about the design of TA's. It indicates that students did not need complete realism or interactivity to enjoy and benefit from teaching virtual agents. The students did not feel the need to create Billy from the bottom up, and they were willing to treat a cartoon character with the intent they might bring to teaching a real human. Our next goal is to create a more interactive TA environment.

Adding interactivity to our environment involves designing small simulations where a TA performs. The first step is to identify the knowledge of a domain and construct a working simulation. We then convert the standard simulation into the TA framework. The TA framework moves the student's interaction away from directly manipulating the variables that control the simulation to providing input that instructs the TA on how to decide what adjustments to make to these variables. Figure 2 provides an overview of one method for accomplishing this outcome.

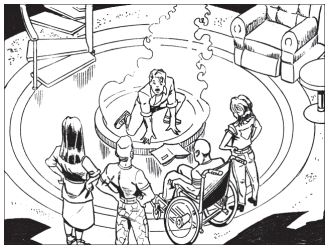
Adding TA's to a simulation does not necessarily represent an application of machine learning. Just like the larger context helped students view the non-interactive Billy as a "real" agent worth teaching, the challenge context reduces the degrees of freedom a TA must have to seem teachable to students. The challenge context defines the boundaries of the amount of domain knowledge an agent needs to know and the knowledge the students need to convey. TA's can greet students already possessing most of the knowledge they need including domain knowledge, knowledge of how to interact with the simulation, and knowledge of how to plan and solve key aspects of the problem. Students only need to provide a little knowledge to make the TA work.



Billy Bashinall is ready to turn in his group's report on water monitoring. He tells his friend, Sally, that he's confident his group's report is good enough because "five pages is always good for a 'C' in Mr. Hogan's class."



Billy's negative attitude comes to the attention of the Dare Force, a group of individuals who learned the hard way that it pays to work hard in school. They have dedicated their lives to helping (daring) others do well.

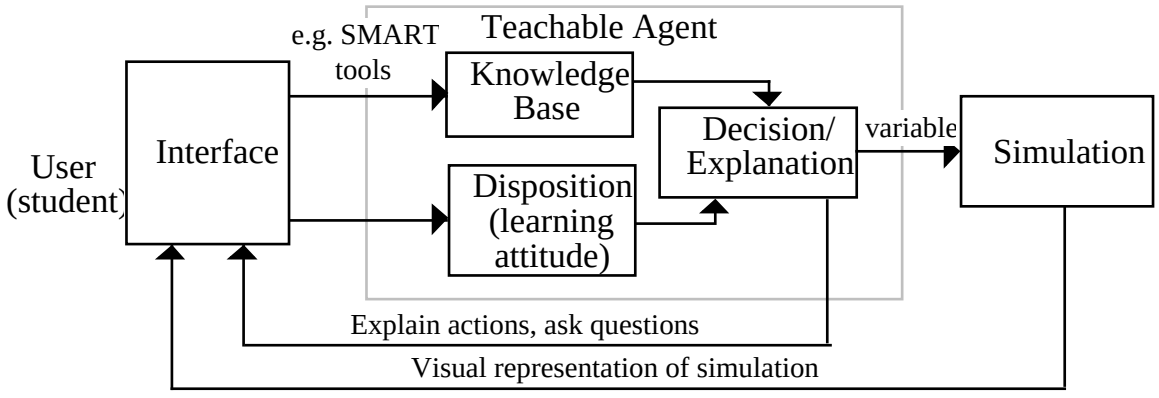


The Dare Force interviews Billy about his understanding of water quality, including the use of indicator species such as macroinvertebrates, and relationships between pollution and dissolved oxygen. Many of the interviews take the form of showing Billy visual scenes from actual river monitoring projects and asking him to explain what's going on.

Billy starts out over-confident. He answers some questions correctly but also displays a number of preconceptions that need to be repaired (e.g., a healthy stream is clear and bug-free). Billy also chooses various tools to help with water monitoring that will not work well. The Dare Force makes it clear to Billy that he has a lot of learning to do (without specifying his exact strengths and weaknesses.) Billy finally agrees and asks for help.

The Dare Force asks students in the classroom to teach Billy and help him reinterpret his data. When they feel they are ready, students can have Billy return to the Dare Force context and see how well he fares.

Figure 1. An Introduction to a Teachable Agent and Its Problem Environment



Instructing a teachable agent may take several different forms. One method of instruction centers around an agent who doesn't have the tools to make decisions about how to formulate a plan. Students could fill out a SmartTool that allows the TA to make decisions and calculations. This method of designing SmartTools is one approach we are exploring for converting the AdventurePlayer software to a TA design. The new version of the AdventurePlayer program provides multiple interfaces that a student can use to teach Billy. For example, students must teach Billy how to derive time parameters by constructing a graph that shows different time over distance slopes for various rates of travel. To teach him, they choose the frame of a line graph which presents two axes. It is their task to enter relevant information including axes labels (miles, hours, etc.), unit increments (10's, 100's, etc.), and lines indicating the time-distance relationship (e.g., the distance over time slope for a 25 mph rate). The completed graph becomes a part of Billy's knowledge base. A complete and correct graph will enable Billy to solve the problem correctly. A graph with incorrect quantities or quantitative relationships will cause the plane to leave too soon or too late in lawful ways. A graph that is missing the labels can lead to idiosyncratic behaviors. One possibility is that the program randomly inserts labels of the same ontology but of different scale (e.g., seconds instead of minutes). The simulation would reflect this insertion, and the graph would highlight the fact that Billy just guessed and put a value onto the axis himself. A second possibility is that the program could insert labels that do not maintain ontology (e.g., pounds instead of minutes). How this manifests itself in the simulation will depend on the specific simulation (e.g., it may misinterpret pounds as the amount of gas needed for a given distance). Determining the best forms of feedback and instruction is a topic for continued research.

Once students have taught Billy, they may place him back into the simulation environment. The simulation environment provides mini-assessments (specific condition and/or configuration of the simulation). These assessments offer small, manageable simulations where students can "debug" Billy under relatively controlled circumstances. Billy is introduced to a specific problem to solve. Students can see Billy's behaviors and whether his solution works. This provides feedback that helps students determine what Billy (and they) have yet to learn correctly. The challenge of debugging Billy can be made more or less difficult by varying the number of "free parameters" that he must fix to make the simulation work.

A problem with many attempts to help students learn by programming is the "overhead problem"-- learning to program often gets in the way of learning important mathematical or scientific content. In the hands of teachers who know the relevant content knowledge and have the programming skills, programming projects involving real and computer agents have resulted in successful content-based learning [20]. However, the burden on teachers to monitor and structure students' learning experiences is very large [20]. The fact that our TA environments are focused on specific goals for content and skills makes the "overhead problem" much less severe. Students can program Billy by using graphs, timelines, and other "SmartTools" that they need to learn to understand the domain.

Figure 2 shows TAs have a dispositional component. This component determines the consistency of the behavior and learning of the agent in plausible ways; for example, the agent may jump to a quick, reasonable solution (a brief search in its knowledge space), or it may strive slowly for a precise solution (an exhaustive search). Students may see short cartoons of Billy solving various simple problems. One cartoon may show Billy responding very quickly

and confidently. Another may show him checking his answers five times. They have to choose which cartoons contain the dispositions that they would like Billy to develop. In this case, probably neither because when they put Billy into a mini-assessment, neither will enable Billy to use his knowledge in an optimal manner. For the former disposition, he may answer fast, but randomly make stupid mistakes. For the latter disposition, he may always get the precise answer but take forever. One can imagine many other dispositions such as simply refusing to do the task, or asking for clarification on a faulty graph before entering the mini-assessment.

We want students to understand that dispositions have a large influence on success, regardless of one's knowledge state. Thus, in our computational representation of Billy we have separated his disposition "module" from his knowledge "module." We are currently exploring different ways to represent dispositional information and have it interact with knowledge to create performances. This represents something of a switch from the Platonic paradigm that dominates Artificial Intelligence. In this paradigm, correct knowledge leads to correct behavior. In our paradigm, correct knowledge may be deployed poorly if an agent has a poor disposition. Moreover, a poor disposition may make it difficult for the agent to learn. We believe that making disposition variables explicit in TA environments should have interesting consequences on students' ability to reflect on their own attitudes and those of their peers.

7. Conclusion

Teachable agents provide a method to facilitate learning in a motivating and natural way for students. The introduction of a virtual human agent captures students' attention and motivates them to help this virtual agent learn new knowledge to accomplish a goal. Our instructional approach emerges from what we know about classroom learning including the value of an anchoring context for making learning meaningful, the need for multiple opportunities for assessing and revising one's knowledge, and the importance of promoting generalization. Students interact with teachable agents in the larger context of a meaningful challenge. Teachable agent simulations provide an excellent mechanism for students to apply what they know and receive feedback. Asking students to teach agents with SmartTools promotes generalization. The focus of our activities with teachable agents relate more to how humans represent knowledge rather than to how computers represent knowledge. This perspective opens new doors for exploring various computer science techniques for representation, accessing and displaying knowledge to facilitate human learning. We offered several example of teachable agents ranging from low interactivity to very high level of interactivity and complex representations of agents and their knowledge base. We are continuing to explore interesting challenges that capitalize on teaching agents and to explore new methods for virtual characters to interact with the human counterparts.

References

[1] Bransford, J. D., & Schwartz, D. L. (in press). Rethinking transfer: A simple proposals with educational implications. Review of Research in Education. Washington, DC.

[2] Palinscar, A. S., & Brown, A. L. (1984). Reciprocal teaching of comprehension monitoring activities. *Cognition and Instruction*, 1, 117-175.

[3] Chi, M. T. H., Bassok, M., Lewis, M., Reimann, M. & Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science,* 13, 145 182.

[4] Fuchs, L. S., Fuchs, D., Karns, K., and Hamlett, C. L. (1996). The relation between students ability and the quality of effectiveness of explanations. *American Educational Research Journal,* 33, 631-644.

[5] Webb, N. M. (1983). Predicting learning from student interaction: Defining the interaction variables. *Educational Psychologist,* 18, 33-41.

[6] Nickerson, R. S. (1983). Computer programming as a vehicle for teaching thinking skills. *Thinking, The Journal of Philosophy for Children, 4,* 42-48.

[7] Papert, S. (1980). *Mindstorms*. New York: Basic Books.

[8] Ableson, H. and diSessa, A. (1980). *Turtle geometry: The computer as a medium for exploring mathematics*. Cambridge, MA: MIT Press.

[9] Kafai, Y., & Resnick, M. (Eds.). (1996). Constructionism in practice. Mahwah, NJ: Lawrence Erlbaum & Associates.

[10] Dillenbourg, P. (Ed.). (in press). *Collaborative learning: Cognitive and computational approaches*. New York: Elsvier Press.

[11] Harel, I., and Papert, S. (1991). *Constructionism*. Norwood, NJ: Ablex.
[12] Repenning, A., & Sumner, T. (1995). Agentsheets: A medium for creating domain oriented visual languages. *Computer,* 28, 17-25.

[13] Ball, G., Ling, D., Kurlander, D., Miller, J., Pugh, D., Skelly, T., Stankosky, A., Theil, D., Van Dantzich, M., & Wax, T. (1997). Lifelike computer characters: The persona project at Microsoft research. In J. M. Bradshaw (Ed.), *Software Agents* (191-222). Menlo Park, CA: AAAI/MIT Press.

[14] Huffman, S. B., and Laird, J.E. (1995). Flexible instructable agents, *Journal of Artificial Intelligence Research,* 3, 271-324.

[15] Lieberman, H. and Maulsby, D. (1996). Instructible agents: Software that just keeps getting better. I*BM Systems Journal,* 35(3&4), 539-556.

[16] Cognition and Technology Group at Vanderbilt. (1997). The Jasper Project: Lessons in curriculum, instruction, assessment, and professional development. Mahwah, NJ: Lawrence Erlbaum Associates.

[17] Cognition and Technology Group at Vanderbilt. (1998). Designing environments to reveal, support, and expand our children's potentials. In S. A. Soraci & W. McIlvane (Eds.), Perspectives on fundamental processes in intellectual functioning, Vol. 1 (pp. 313-350). Greenwich, CT: Ablex.

[18] Sherwood, R., Petrosino, A., Lin, X. D., and the Cognition and Technology Group at Vanderbilt. (in press). Problem based macro contexts in science instruction: Design issues and applications. In B.J. Fraser & K. Tobin (Eds.), *International handbook of science education*. Dordrecht, Netherlands: Kluwer.

[19] Vye, N. J., Schwartz, D. L., Bransford, J. D., Barron, B. J., Zech, L. and Cognition and Technology Group at Vanderbilt. (1998). SMART environments that support monitoring, reflection, and revision. In D. Hacker, J. Dunlosky, & A. Graesser (Eds.), Metacognition in Educational Theory and Practice. Mahwah, NJ: Lawrence Erlbaum & Associates.

[20] Littlefield, J., Delclos, V., Lever, S., Clayton, K., Bransford, J., & Franks, J. (1988). Learning logo: Method of teaching, transfer of general skills, and attitudes toward school and computers. In R. E. Mayer (Ed.), Teaching and learning computer programming (pp. 111-135). Hillsdale, NJ: Lawrence Erlbaum Associates.