Presented at the Workshop on Adaptive Hypermedia and Collaborative Web-Based Systems (AHCW'04), Munich, Germany, July 2004

TOWARD COGNITIVE TUTORING IN A COLLABORATIVE, WEB-BASED ENVIRONMENT

BRUCE M. MCLAREN, KENNETH R. KOEDINGER, MIKE SCHNEIDER

Carnegie Mellon University
Pittsburgh, PA USA
bmclaren@cs.cmu.edu, koedinger@cmu.edu, mike.schneider@cs.cmu.edu

ANDREAS HARRER, LARS BOLLEN

Collide Research Group
University Duisburg-Essen
Duisburg, Germany
harrer@collide.info, bollen@collide.info

While intelligent tutoring has been applied to collaborative learning environments, it has met with little success so far because of the complexity involved in adding a tutoring component to a collaborative environment. We propose to tackle this problem by using Cognitive Tutors as the basis for our approach and by applying a technique we call Bootstrapping Novice Data (BND). The BND approach involves feeding student log files from a problem-solving tool into tutor development software to create the beginnings of a tutor for the tool. We describe an initial implementation of our approach in which Cool Modes, a collaborative software tool, is integrated with the Behavior Recorder, tutor-authoring software that supports development by demonstration. We show how our initial implementation provides a foundation for an intelligent tutor for collaboration. We also discuss some of the challenges ahead.

1 Introduction

Intelligent Tutoring Systems (ITS) [13] have typically been developed for one-on-one (machine-to-student) instruction. We are interested, however, in how we could integrate tutoring into a collaborative, web-based work environment. While efforts have been made toward providing tutoring in a collaborative environment (e.g., [3, 5, 6, 15]), there are many and varied challenges still ahead. Consider, for instance, the complexities of building tutors for a computer-mediated collaborative environment in which users in remote locations, communicating over the Internet, work together to solve a problem. The large space of possible actions and interactions between the users, in addition to the problem representation itself, make it more challenging to analyze learner behavior and build tutors for collaborative tasks than for the single-student case.

As an alternative to the traditional approach to tutor development – and as a step toward addressing the complexities of a collaborative environment – we propose a tutor-development approach that leverages actual problem-solving data not only to guide tutor design, as has been done before (e.g., [8]), but also to contribute directly to tutor implementation. In this approach, called bootstrapping novice data (BND), we provide groups of collaborating novice users with a computer-based tool, let them attempt to solve problems with the tool, and record that problem-solving activity in a tutor-specific representation. This integrated record of user activity has two important benefits: (1) we learn a great deal about how users engage in collaborative problem-solving activities, and (2) it provides the initial representational structure for a tutor that could support collaboration.

Our initial step in this direction has been to develop a prototype integration between a collaborative software tool, Cool Modes (Collaborative Open Learning and MODEling System) [12], and a tutor authoring environment, the Cognitive Tutor Authoring Tools (CTAT) [7]. The Cool Modes software generates computer log files of user activity that are, in turn, used by CTAT as an initial representation of the tutor. In this paper, we discuss how we have effected this integration and how we could use this model as a way of collecting user data to create an initial, skeletal tutoring system.

An important underpinning of this work is the notion of component-based development. Our approach relies on taking an existing software application (what we term a "tool") and integrating it, with little or no modification, with a tutor or tutor agent. Using off-the-shelf or pre-existing software as the basis for building tutoring systems could result in substantial time savings, as compared to the traditional approach of building tutors "from scratch" [10, 14]. This is particularly important in developing tutors for collaboration, as the underlying interaction model is much more complex than in the single-student scenario.

A longer-term aim of our work – which will be greatly facilitated by both the component-based approach and our initial steps in developing BND – is to explore how we can fully integrate cognitive tutoring techniques in a computer-mediated collaborative environment. We want to use the integration of Cool Modes and CTAT not just as a means of leveraging user log files to help in building a skeletal, initial tutor, but also as a step toward developing a fully integrated, pedagogical model to provide real-time tutoring in Cool Modes and other collaborative software environments.

2 Bootstrapping Novice Data: Creating the Initial Representation of a Tutor for Collaboration

The BND process we have developed to create an initial, skeletal model of a tutor through log data involves the integration of two software components, Cool Modes and the Behavior Recorder, each with complementary features.

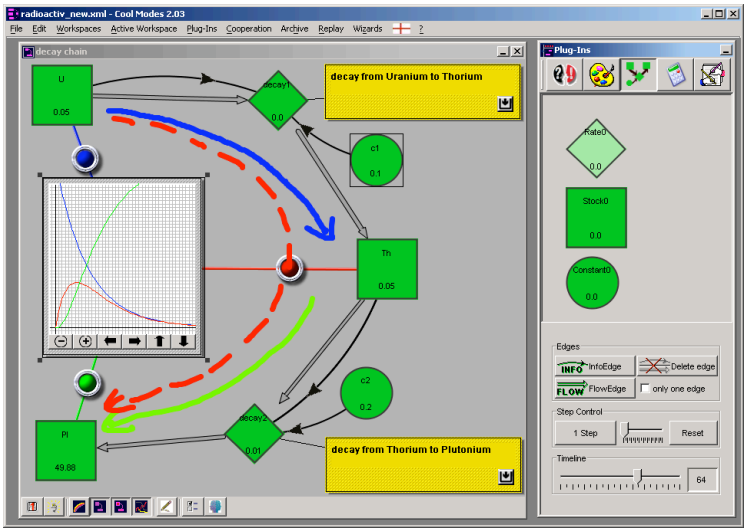

Figure 1. An example Cool Modes client. Here students are collaborating on a nuclear decay problem in a shared workspace.

Cool Modes, depicted in Figure 1, is a collaborative software tool designed to support "conversations" and shared graphical modelling facilities between collaborative learners [12]. It is a domain-independent tool that supports a variety of modelling and learning tasks and provides users with various plug-in objects, such as Petri nets, a turtle programming environment, text widgets, and a "chat" area, each of which has its own semantics and underlying representation. All of the Cool Modes objects are available on a palette from which students may drag and drop objects into workspaces to build models. Cool Modes is extensible; new objects adhering to a well-defined API may be added to the palette as required. Translation between different object types is achieved through reference frames, a set of entities and rules that facilitate semantic object mapping. Cool Modes also provides operational facilities, such as execution of simulations and automated calculations.

Each Cool Modes user runs a client program that contains a private workspace in which objects can be privately created and updated. In addition, all users have access to a shared workspace, such as that shown in Figure 1, which is rendered in all of the collaborating clients and may be updated by any of the participants. The Cool Modes client program communicates with a server, called MatchMaker [4],

which maintains the shared workspace and handles all communication between the collaborating clients.

Cool Modes does not assess or critique students' solutions apart from helping the students create syntactically correct models and allowing them to execute and observe the models. Thus, the tutoring component of the integration is provided by CTAT [7], an authoring tool for intelligent tutors. It supports authors in building Cognitive Tutors, a form of "model-tracing" tutor based on cognitive psychology theory. As of the spring of 2004, Cognitive Tutors have been deployed in over 1700 schools in the United States. A Cognitive Tutor is composed of a problem representation and a set of production rules that model both desired and buggy behavior and is able to tutor students on a range of problems within a particular domain (e.g., geometry, algebra). Model tracing compares actual student behavior during problem solving with the desired behavior represented in the production rules and identifies deviations from that behavior. Cognitive Tutors are difficult to develop, typically requiring AI programming expertise.

On the other hand, a specialized type of Cognitive Tutor also supported within CTAT is a *Pseudo Tutor*, a tutor that behaves much like a regular Cognitive Tutor, except that it provides instruction for only a single problem instance and is much easier to develop. Pseudo Tutors are developed using a "programming by demonstration" approach [9] that allows authors with no programming skills to build tutors.

An example Pseudo Tutor for fraction addition is shown in Figure 2. A Pseudo Tutor is developed as follows. First, the author builds a graphical user interface (GUI) with a specialized set of CTAT widgets. The GUI for the fraction addition tutor is shown on the right side of Figure 2. Second, the author demonstrates sequences of correct, alternative correct, and incorrect actions on these widgets. A CTAT tool known as the Behavior Recorder (BR for short) records all of these actions and builds a structure known as a behavior graph, shown on the left side of Figure 2. Each edge of the graph represents an action taken by the student on a particular widget of the GUI. The thicker edges represent the preferred or primary action taken from a particular node. The student's action is represented as a triple (selection, action, input): the *selection* identifies the GUI widget selected, such as "TextArea1"; the *action* is the type of user action taken in the GUI, such as "Update Text"; and the *input* is the value provided by the student, such as "20." Each node of the graph represents a state of the interface after a path of edges from the root to that node has been traversed[1]. Third, after the behavior graph has been created by problem demonstration, the author can annotate the graph by labelling buggy edges (e.g., the incorrect path represented by "2, F13num" in Figure 2), inserting hints and

[1] There can be multiple paths to a node and thus a node can represent multiple states.

feedback messages, and associating skills with edges. Finally, the author can test the model by running the Pseudo Tutor, acting like a student or observing actual student use. During the testing step the Behavior Recorder no longer builds the graph but rather traces student actions on it. The whole process typically iterates several times, as the tutor author refines the graph and accounts for alternative action sequences.

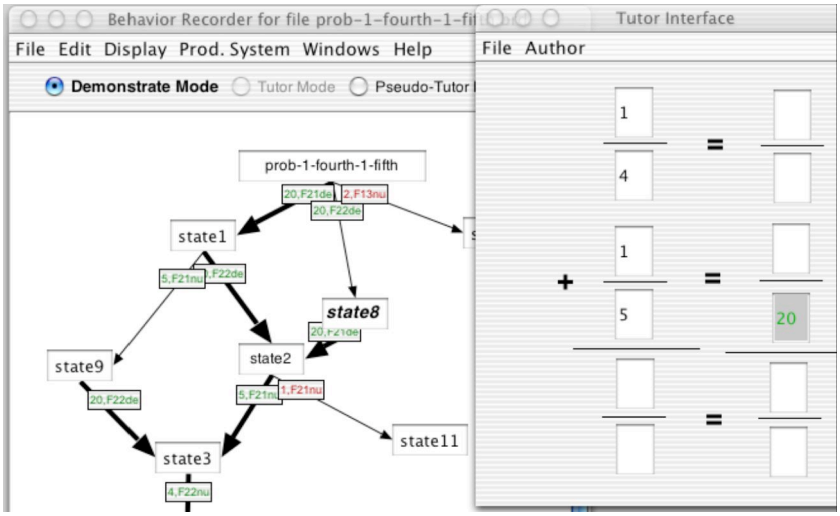

Figure 2. The Behavior Recorder records authors' actions in any interface which use CTAT's specialized GUI widgets. The author demonstrates alternative correct and incorrect paths. From the start state (labeled "prob-1-fourth-1-fifth") there are two correct paths ("20, F21den" and "20, F22den"), in which a common denominator is entered in either of the converted fractions, and one incorrect path ("2, F13num"), in which the numerators of the addend fractions are incorrectly summed to a value of "2" and placed in the numerator of the result. Although not visible in this black and white figure, the incorrect path has a red edge label and the two correct paths have green labels. One of the edges emanating from each node is thicker than the others; this indicates that it is the "preferred" path from that node. In the figure state8 is selected in the Behavior Recorder; this can be seen because it is boldfaced and italicized. The Tutor Interface displays the currently selected state, as can be seen by "20" appearing in the second converted fraction of the GUI.

The fundamental idea of Bootstrapping Novice Data, depicted in Figure 3, is to use a tool, in this case Cool Modes, to generate a log of user actions and have those actions recorded in the BR, providing the beginnings of a real tutor. Translated user log files, which contain recorded user actions, are provided as input to the BR. Because the BR is a component with a well-defined message interface (i.e., it accepts XML messages we call "Dormin Messages"), this integration was relatively easy to develop. All that was required was a Translator, developed in XSLT, to take the Cool Modes XML log files and convert them to XML Dormin messages.

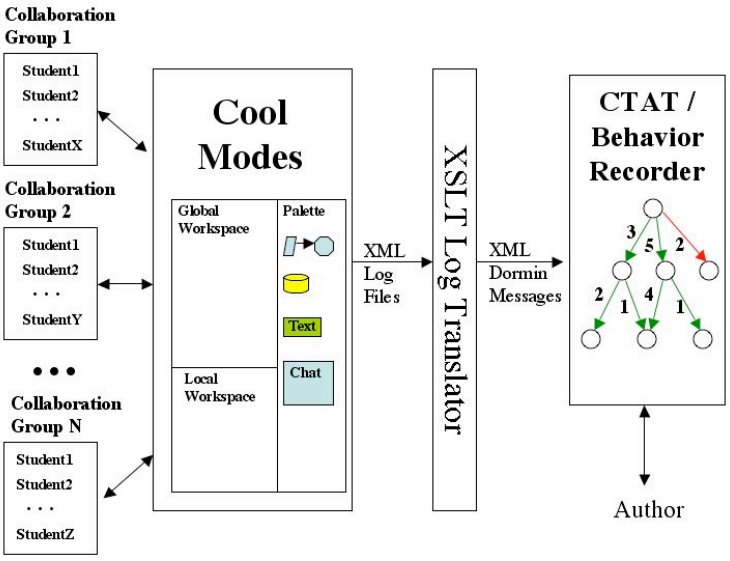

Figure 3. Bootstrapping Novice Data through the integration of Cool Modes and the Behavior Recorder.

The BND approach involves having groups of collaborating users, depicted on the left side of Figure 3, generate (possibly different) correct and faulty solutions to the same problem. The logs of the different collaborating groups are then translated into a single behavior graph in the Behavior Recorder. After the behavior graph is generated, a tutor author manually updates it by adding hints and bug messages, annotating buggy paths, and adding skills to the edges. Not only does the BND approach provide the tutor author with examples of actual correct and buggy paths taken by users, it also presents the author with traversal frequencies of those paths. The edge traversal counts are good indicators of which of the correct solution paths might be considered primary, which secondary, as well as which types of error paths occur frequently enough to merit writing specific bug messages.

After the behavior graph is generated, the tutor author manually updates it by adding hints and bug messages, annotating buggy paths, and adding skills to edges. Not only does the BND approach provide the author with examples of *actual* correct and buggy paths taken by students, it also presents the author with another important piece of information: traversal frequencies of those paths. The edge traversal counts are good indicators of which of the correct solution paths might be considered primary, which secondary, and the counts along incorrect paths provide real data to show which errors occur frequently enough to merit writing specific buggy messages. The traversal counts can also help authors identify slips and careless errors (e.g., accidental item selections): an edge with a traversal count of 1, as compared to much higher counts on alternative edges, may indicate that an accidental action was taken by a student and thus can be deleted from the graph.

The power of the BND approach is that instead of authors building Pseudo Tutors from scratch, tapping only their individual experience or incorporating student data "by hand" as in traditional ITS development, they can semi-automatically leverage the empirical data of a wide range of students engaged in actual problem-solving activity. This approach contrasts markedly with the usual ITS development method in which a domain expert author first creates "expert" problem solutions. In our approach, the student novices create initial solutions. The expert's judgement as to which novice solutions are correct is, of course, critical to creating a final version of the tutor. It may also be the case that an expert will have to augment the model by demonstrating a correct solution or solutions, if the student novices fail to generate any correct solution paths. But the critical aspect of BND is how it directly captures and encodes incorrect and inefficient novice solutions, information that will prove invaluable in building a full ITS for a collaborative system.

3 An Example: The NASA Exercise for Group Dynamics

We tested our BND approach and integration of Cool Modes and BR using a scenario called the NASA exercise. In this exercise, a group of students is presented with the following fictional space travel problem. After their shuttle crashes on the moon, the students must decide which of a set of 15 items on the shuttle (e.g., matches, a bottle of oxygen, a bottle of water) are most important for survival while journeying to the mother ship 200 miles away on the moon's surface. The task facing the students is to assign a strict priority ordering to all items (e.g., bottle of oxygen 1 (highest), matches 15 (lowest)). Using scientific knowledge about light, the need for oxygen, the atmosphere of the moon, etc., NASA has proposed an optimal solution to the problem.

The game is typically first played by each individual and then as a collaborative group. Interestingly, the collaborative solution is typically better than the average individual solution and even better than the best individual's. Both the individual and group phases of the exercise can be easily conducted in Cool Modes using a visual discussion language which employs text cards for the items, numerical indicators for priorities, and links to connect items to priorities.

The task description of a simplified version of the NASA exercise, with 5 items instead of 15, is shown in Figure 4.

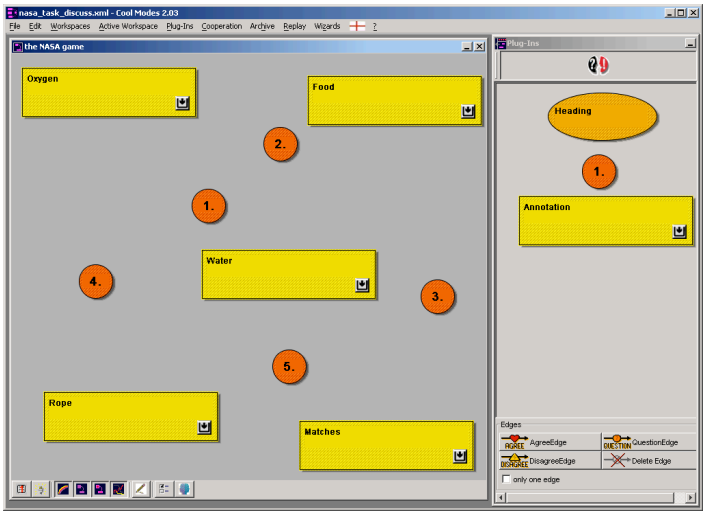

Figure 4. Task description of the simplified NASA exercise within Cool Modes.

We obtained several potential solutions, both optimal according to NASA's proposed solution (shown in Figure 5) and deviating from that, by logging Cool Modes sessions with different groups of students.

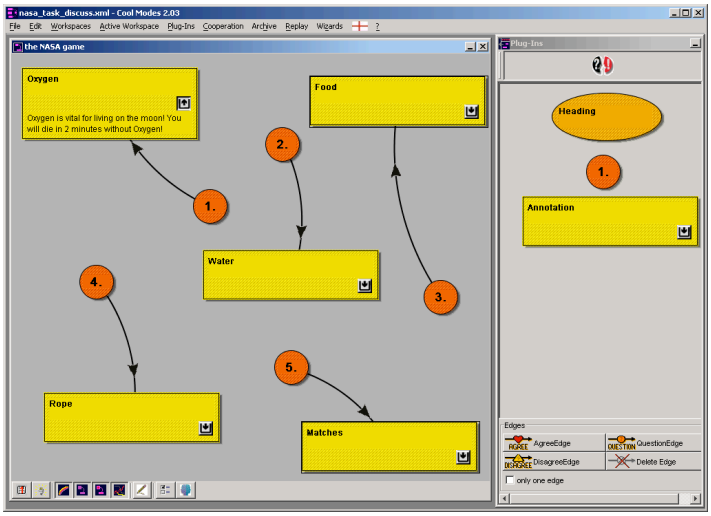

Figure 5. Correct solution of the simplified NASA exercise within Cool Modes.

As per Figure 3, the set of Cool Modes XML log files were transformed by XSLT into XML Dormin messages. The messages were then fed into CTAT's Behavior Recorder, resulting in a behavior graph with weighted edges, showing the frequency of steps taken by the users across all solution attempts.

The resulting behavior graph is shown in Figure 6. The left branch at the top represents the fact that 4 of the users correctly chose a bottle of oxygen as the highest priority item (by connecting priority 1 to the text card labelled "oxygen" in Cool Modes), while the right branch at the top indicates that 2 users first (and also correctly) assigned matches as the lowest priority item. The branch emanating from state11 to state12 is a buggy path, in which oxygen was assigned a low priority (4). Since some of the users naturally chose different, yet still correct sequences of priority assignments, the graph has some confluent paths. These are equally valid paths in solving the problem.

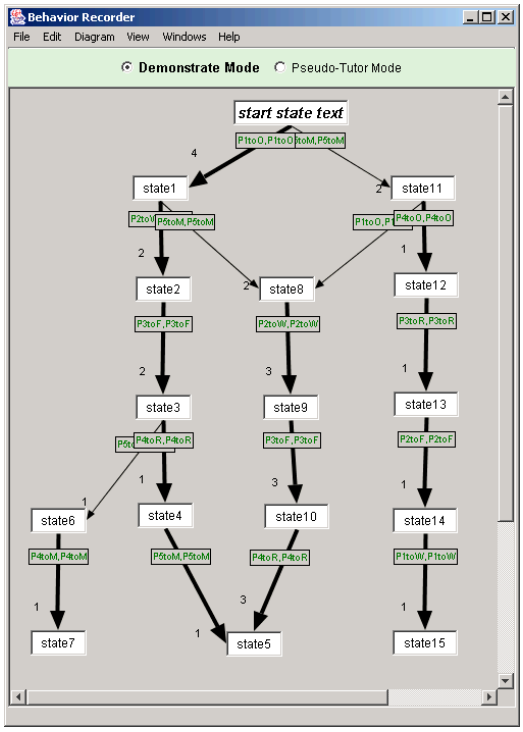

Figure 6. The Behavior Recorder after multiple user solutions of the NASA exercise.

The author has not yet manually updated the behavior graph shown in Figure 6. For instance, the author would ultimately mark the path on the right below state11 as "buggy," which would change the selection/action/input of the rightmost edge emanating from state11 to red (the color we use to indicate a "buggy" action). This, in turn, would result in the deletion of all states on the path below state12, since the Behavior Recorder assumes that all buggy states are "dead ends".

4. Discussion

The NASA exercise illustrates the value of the BND approach and the Cool Modes / Behavior Recorder integration. By having students attempt to solve this problem, we not only record the variety of paths taken, but we actually produce the backbone representation of an actual tutor. Given this initial representation, an author then annotates the behavior graph with hints, buggy actions, and bug messages. The traversal counts guide the author by indicating which paths are anomalous and which are common solutions/mistakes. An annotated behavior graph can be run as a fully functioning Pseudo Tutor.

But there is still much work to be done. The NASA problem, while intended to be tackled collaboratively, was, for our purposes, attempted only by individual users in order to test the Cool Modes / Behavior Recorder integration. To make the BND process work for actual collaboration, there is still research and development work required. For instance, the behavior graphs in typical mid-range to difficult problems are likely to be quite bushy, interconnected, and thus difficult to read and manipulate manually; for highly complex problems it will be difficult to grade *all* actions as either correct or buggy. One of our goals, therefore, is to write software to automatically read and evaluate the graphs and, in turn, provide advice to authors (for instance, that an infrequently traversed edge may be buggy).

One very basic change to the BR required to support collaboration is a way to record the user who performed each action, information Cool Modes has readily available. Since the BR was originally designed to support single-student tutoring, this data is currently not captured in the behavior graph, so we will extend the selection-action-input representation to include and reason about an "Actor" field. Since collaboration environments typically involve dynamic instantiation of objects (e.g. when creating a model from scratch with a visual modelling language like Petri Nets or System Dynamics), the Behavior Recorder must also be capable of handling dynamic object definition and recognition. In other words, the BR must be able to recognize similar selection-action-inputs across collaborative sessions (which would result in different paths in the behavior graph) and map these actions and objects to a single entity. Since Cool Modes uses a consistent, internal naming scheme, we can leverage this to identify common objects across sessions. Ultimately, for collaborative systems in general, we will need mapping tables to help the BR identify similar nodes and paths across sessions.

We may also need a more powerful representation than behavior graphs for modelling behavior and delivering tutoring for more complex collaborative activities. Existing collaborative systems typically employ more complex models, such as action-based collaboration analysis [11], which derives higher-level activity descriptions from user actions using plan recognition, or Hidden Markov Models [16], which have been used to model effective and ineffective student interactions.

The production system approach of full Cognitive Tutors appears to be a promising way to model more complex collaborative behavior. In particular, we could treat collaboration as a special type of meta-cognitive behavior, by specifying a desirable model of collaboration, similar to what we've done in the area of help seeking [1, 2]. Given this direction, the collaborative tutor building approach described in this paper could be viewed as an initial cognitive modelling / cognitive task analysis, initial test case development, and programming-by-demonstration input for building full Cognitive Tutors.

A clear, but we believe appropriate, limitation of the direction outlined in this paper is emphasis on student problem-solving actions, rather than on collaborative social activities such as dialogue between collaborators [3]. While we acknowledge that dialogue is important to many, if not most, collaborative activities, we believe that a significant class of collaborative problems can be dealt with by computer support at the level of observed problem-solving actions. Our focus is also a matter of tackling one piece of the complex collaboration puzzle at a time.

5. Conclusion

In summary, we have discussed the use of log files and log analysis as a starting point in developing a tutoring component for a pre-existing software tool. Our approach, called Bootstrapping Novice Data, involves the transformation of student log files generated by a problem-solving software tool into a sequence of student-action messages useable by tutor authoring software. To implement an initial version of BND, we used a component-based approach in which we integrated an existing collaborative software tool, Cool Modes, with tutor-authoring software, realized in the Behavior Recorder software. The BND approach is potentially quite powerful, as it obviates the need to (a) build a tutor "from scratch" and (b) rely primarily on a domain expert to build a tutor.

The ultimate aim of our work is to explore how we can fully integrate cognitive tutoring techniques in a computer-mediated collaborative environment. In other words, we want to use the integration of Cool Modes and CTAT as a first step toward developing a fully integrated, pedagogical model to provide real-time tutoring in Cool Modes and other collaborative software environments.

Currently, we are working on a real-time integration of Cool Modes and the Behavior Recorder by using an extended message protocol and a socket connection, instead of XSLT, between the applications. We plan to conduct a field test and study with small groups (or dyads) of students collaborating on modelling problems in the domain of software engineering in the fall of 2004. This study will allow us to further explore the technical and practical feasibility of the BND approach and to collect data to guide us in providing a cognitive tutoring approach to a collaborative environment.

In general, to reach the ultimate goal of deploying cognitive tutoring in a collaborative work environment, we need to better understand the intersection of collaborative workspaces, such as Cool Modes, and individualized cognitive tutoring, such as that provided by CTAT. By applying BND techniques and taking a component-based approach, we have created the foundation for continued exploration and experimentation with cognitive tutoring in a collaborative environment.

References

1. Aleven, V., McLaren, B. M., Roll, I., and Koedinger, K. (2004). Toward Tutoring Help Seeking: Applying Cognitive Modeling to Meta-Cognitive Skills. In the Proceedings of the Seventh Annual Conference on Intelligent Tutoring Systems (ITS), Maceio, Brazil, September, 2004.
2. Aleven, V., McLaren, B. M., and Koedinger, K. (forthcoming). Towards Computer-Based Tutoring of Help-Seeking Skills. In: Help Seeking in Academic Settings: Goals, Groups and Contexts, Karabenick, S. ,Newman, R. (eds).
3. Goodman, B., Hitzeman, J., Linton, F., and Ross, H. (2003). Towards Intelligent Agents for Collaborative Learning: Recognizing the Role of Dialogue Participants. In the Proceedings of Artificial Intelligence in Education (AIED-03), IOS Press, Amsterdam.
4. Jansen, M. (2003) Matchmaker - a framework to support collaborative java applications. In the Proceedings of Artificial Intelligence in Education (AIED-03), IOS Press, Amsterdam.
5. Jermann, P., Soller, A., and Muehlenbrock, M. (2001). From Mirroring to Guiding: A Review of State of the Art Technology for Supporting Collaborative Learning. In the Proceedings of the European Conference on Computer-Supported Collaborative Learning, Maastricht, The Netherlands.
6. Lesgold, A., Katz, S., Greenberg, L., Hughes, E., Eggan, G. (1992). Extensions of Intelligent Tutoring Paradigms to Support Collaborative Learning. In S. Dijkstra, H. Krammer, J. van Merrienboer (Eds.), Instructional Models in Computer-Based Learning Environments. Berlin: Springer-Verlag, 291-311.
7. Koedinger, K. R., Aleven, V., Heffernan, N., McLaren, B. M., and Hockenberry, M. (2004). Opening the Door to Non-Programmers: Authoring Intelligent Tutor Behavior by Demonstration. In the Proceedings of the Seventh Annual Conference on Intelligent Tutoring Systems (ITS), Maceio, Brazil, September, 2004.
8. Koedinger, K. R. and Terao, A. (2002). A cognitive task analysis of using pictures to support pre-algebraic reasoning. In C. D. Schunn, W. Gray (Eds.), Proc. of Cognitive Science Society, 542-547.
9. Lieberman, H. (ed) (2001). Your Wish is My Command: Programming by Example. Morgan Kauffman Publishers.

10. McArthur, D., Lewis, M. W., and Bishay, M. (1996). ESSCOTS for learning: Transforming commercial software into powerful educational tools. Journal of Artificial Intelligence in Education, 6 (1), 3-33.
11. Muehlenbrock, M. (2001). Action-based Collaboration Analysis for Group Learning. IOS Press, Amsterdam.
12. Pinkwart, N. (2003). A Plug-In Architecture for Graph Based Collaborative Modeling Systems. In the Proceedings of Artificial Intelligence in Education (AIED-03), IOS Press, Amsterdam.
13. Polson, M. C. and Richardson, J. J. (1988). Foundations of Intelligent Tutoring Systems. Lawrence Erlbaum Associates Publishers.
14. Ritter, S. and Koedinger, K. R. (1996). An Architecture For Plug-In Tutor Agents. In: Journal of Artificial Intelligence in Education, 7 (3 / 4), 315-347.
15. Suthers, D. D. (2003). Representational Guidance for Collaborative Learning. In the Proceedings of Artificial Intelligence in Education (AIED-03), IOS Press, Amsterdam.
16. Soller, A. and Busetta, P. (2003). An Intelligent Agent Architecture for Facilitating Knowledge Sharing Communication. Proc. of Workshop on Humans and Multi-Agent Systems at AAMAS-03, Melbourne, Australia, 94-100.