Cognitive Tutors as Research Platforms: Extending an Established Tutoring System for Collaborative and Metacognitive Experimentation

Erin Walker[1], Kenneth Koedinger[1], Bruce McLaren[1], and Nikol Rummel[2]

[1] Human Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, PA, USA
{erinwalk, koedinger, bmclaren}@cs.cmu.edu
[2] Department of Psychology, Albert-Ludwigs-Universitat Freiburg, Germany
rummel@psychologie.uni-freiburg.de

Abstract. Cognitive tutors have been shown to increase student learning in long-term classroom studies but would become even more effective if they provided collaborative support and metacognitive tutoring. Reconceptualizing an established tutoring system as a research platform to test different collaborative and metacognitive interventions would lead to gains in learning research. In this paper, we define a component-based architecture for such a platform, drawing from previous theoretical frameworks for tutoring systems. We then describe two practical implementation challenges not typically addressed by these frameworks. We detail our efforts to extend a cognitive tutor and evaluate our progress in terms of flexibility, control, and practicality.

1 Introduction

A cognitive tutor is an intelligent tutor that compares student action during problem-solving to a model of correct action, and provides context-sensitive hints and error feedback. These tutors have been shown to be effective at increasing student learning in long-term classroom studies by as much as one standard deviation over traditional instruction [1,2]. In general, cognitive tutors have focused on instruction to increase domain knowledge but have lacked support for collaborative activities or metacognitive tutoring. However, cognitive tutors could become even more effective if used in combination with collaborative and metacognitive interventions.

There is a need for research on which interventions are most effective. Collaboration can increase student mastery of domain knowledge, reasoning strategies, and social skills, but it is only effective when designed to encourage particular behaviors [3]. Although cognitive tutors like the Cognitive Tutor Algebra I (CTAI) are used in conjunction with collaborative classroom activities, it is difficult to control whether these activities occur as intended and difficult to measure their effectiveness. It is necessary to determine which activities produce desired learning effects and to control their execution. Similarly, preliminary research on supporting metacognition in intelligent tutoring systems has yielded encouraging results [4]. More research is needed to evaluate different interventions in a classroom context.

Tutoring systems are ideal environments for experimentation with collaborative and metacognitive interventions. They are useful settings for the implementation of

M. Ikeda, K. Ashley, and T.-W. Chan (Eds.): ITS 2006, LNCS 4053, pp. 207 – 216, 2006.

collaborative activities: The controlled environment adds structure, and actions can be tutored so that desired behaviours are exhibited. Further, implementing metacognitive instruction within the context of an intelligent tutoring system can provide monitoring and support mechanisms for metacognition that may increase learning. Trying many interventions on the same system allows researchers to examine varying effects and explore how the interventions might complement each other. There are additional benefits to using an *established* tutoring system: savings in development time, preexisting relationships with classrooms that are using the tutor, and a proven baseline of effectiveness. For example, the CTAI is an integral part of the Pittsburgh Science of Learning Center (PSLC) which facilitates experimental studies in real-world contexts by providing access to schools and programmers.

An established tutoring system is a solid basis for a research platform. A research platform should be *flexible* in terms of the number of experiments that can be conducted, *controlled* in terms of the number of factors that can be compared in an experiment, and *practical* in terms of its ability to facilitate interventions that are used in a real classroom. The idea of a cognitive tutor as a research platform has been exploited before; existing tutoring systems like Project LISTEN have been used for large-scale data collection and analysis and have been extended to test hypotheses about tutoring cognitive skills [5]. However, these extensions have not contributed to a flexible framework for experimentation with more complex interventions. Preexisting tutoring systems have been extended to test collaborative and metacognitive hypotheses [6], but these projects have not created a platform that researchers can use to compare scenarios. Building a collaborative or metacognitive tutoring system from scratch to test certain research hypotheses [7,8] can provide flexibility and control, but because these systems are not based on an established tutor, they may not be practical for classroom use and would require much development to make them so.

In this paper, we describe the extension of an established tutoring system into a research platform that provides support for *both* collaborative and metacognitive interventions. We define a theoretical architecture for such a platform, discuss practical challenges in implementing the architecture, and evaluate our specific efforts in extending the CTAI into a platform for experimentation.

2 Component-Based Architecture

A research platform must allow flexibility in terms of the number of tutoring experiments that can be run and control within an experiment to facilitate ablation studies. These requirements can be met using a component-based architecture that emphasizes the development of independent, reusable components [9]. An ideal implementation of this approach yields a situation where components can be created for use in a variety of situations and can be added or removed without much effort.

Component-based architectures have been proposed as a way to enhance the effectiveness of intelligent tutoring systems, but it is often difficult to integrate components created for different purposes [10]. Standards for curriculum representation like the Scaleable Content Object Reference Model have been proposed to resolve this problem [11], and distributed architectures like KnowledgeTree [12] and multi-agent approaches such as I-help [13] have been developed to integrate individual web-based

services more effectively. Architectures designed for collaborative activities have also been developed, with a focus on synchronizing objects between components to create shared activity spaces [14]. We model our approach after these approaches, focusing on developing reusable components and interaction standards.

We have developed a component-based architecture (Figure 1) for a research platform for collaborative and metacognitive tutoring, based on Ritter and Koedinger's architecture for plug-in tutoring agents [15]. We focus on students solving problems in a single application, unlike the above research, which focuses on the integration of multiple applications. We intend the components we discuss to be situated in a larger web-based system for the delivery of multiple applications (e.g., [12]), and to be capable of receiving curriculum information independently.

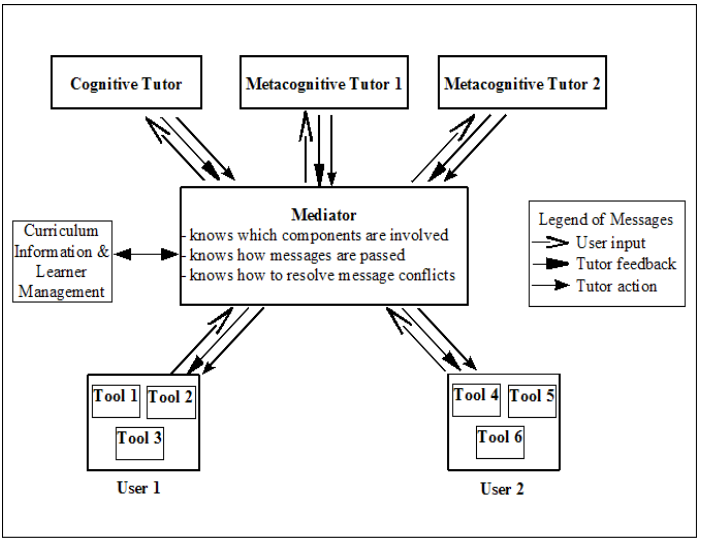

Fig. 1. Our component-based architecture. The figure depicts 3 tutors, 6 tools, and 2 users, but in theory, it could have any number of tools, tutors, and users.

We separate the components in our architecture into tools and tutors. Tool components are the part of the application that the user sees, while tutor components are the part of application that evaluates and offers advice to the student [15]. A spreadsheet is a tool, while the software module that gives feedback to a student on completing the spreadsheet is a tutor. Adding multiple tool components to a given application can facilitate collaboration; each collaborator might use a different tool on a different computer. Adding multiple tutor components to a given application can facilitate metacognitive tutoring. A tool that incorporates self-explanation could have two corresponding tutors, one that is responsible for the cognitive elements of the task, and one that is responsible for the self-explanation elements of the task [6]. To facilitate reuse, tools should be built to be compatible with any corresponding tutor, and tutors should be built to be usable with any corresponding tool.

Establishing standards for messages exchanged by components during a tutoring session, based on the protocol in [15], can also enable component reuse. In general, tools send messages that communicate information about user actions, while tutors send messages that provide feedback about user actions. A proposed set of messages is shown in Table 1. To facilitate reuse, any message that can be sent by a tool should be understood by a tutor, and any message that can be sent by a tutor should be understood by a tool. Although this protocol was originally designed to support cognitive tutoring, it can be applied to metacognitive and collaborative tutoring, where message content may be different but message structure would be the same.

Table 1. Message types for tool and tutor components

Component	*Message Type*	*Message*	*Meaning*
Tool	User action	Note input	Student uses an interface element
		Note create	Student creates an element
		Note delete	Student deletes an element
		Note hint request	Student asks for a hint
		Note done	Student indicates problem complete
Tutor	Tutor feedback	Approve	Answer is correct
		Flag	Answer is incorrect
		Point to	Points to an element
		Send message	Gives feedback
		Update assessment	Changes student assessment
	Tutor action	Send input	Performs user action
		Undo input	Undoes user action
		Send create	Creates element
		Send delete	Deletes element

Following Ritter [16], we define a remotely located *mediator* to control the interaction between components. The mediator is aware of which components are involved in a collaborative session, and during tutoring, receives messages and passes them to the appropriate components. The mediator also holds a set of rules for dealing with message conflicts (sample rules are described in [16]). Because knowledge for how to deal with interacting components is located within the mediator, only the mediator has to be changed when adding or removing components.

3 Practical Challenges

Although theoretical frameworks can provide guidelines for the design of independent, reusable components, there is a question as to whether these architectures are practical for development and classroom use. It is not always feasible to design for reuse because usability can be in conflict with reusability, and it can be impractical to develop reusable components on a realistic schedule. Architectures must be more specific about expectations for component reuse.

3.1 Feasibility of Designing for Reuse

There is a conflict between designing for reuse and designing for usability. Different tools can be easier to use together if their relation is represented in the interface. When a student has to input values in a spreadsheet and then plot the values on a graph, it is easier to have each point plotting widget located next to the appropriate spreadsheet cell, so that the connection between the two tasks is clear. However, tools then become dependent on each other, making it more difficult for developers to reuse them. Theoretical architectures need to account for compromises that must be made between usability and reuse. Developers could consider linked tools as a single tool, making the larger tool reusable.

Further, developing for reuse is not always an attainable goal. On a tight development schedule, it is unlikely that programmers can give tools the higher level of functionality required for reuse. When a tool was not originally designed for collaborative use, adding functionality for tutor actions is not a high priority. It also may be unreasonable to modularize components to the level required for reuse. For small tutoring interventions, it may be simpler to modify the primary tutor rather than to add an adjunct tutor. Frameworks should prioritize reuse requirements so that developers can implement critical functionality for reuse but code flexibly enough to facilitate the future implementation of other requirements. Explicit implementation guidelines for usability and reuse would aid in design decisions.

3.2 Separation Between Tool and Tutor Components

Taking usability and a practical development schedule into account can create difficult decisions when attempting to implement a theoretical framework that does not specify tool and tutor responsibilities. We attempt to define tool responsibilities to maximize the flexibility and experimental control provided by a multi-agent architecture. Tools should act as more than an interface to the tutor. For instance, in an equation solver, when a student subtracts both sides of an equation by five the result needs to be displayed by the tool. If the tool behaves as an interface, some agent would be needed to compute the result of the student action, which means that the tool is dependent on that agent for updating its state. One might argue that it is not practical to emphasize independence for tools that do not mimic anything outside a tutor. However, one should design with future extensions in mind. In an architecture where components can be easily added and removed, it is necessary that the tool hold the logic for calculating the result of student actions.

A more difficult question is whether the tool should require permission from a tutor to update its state based on student actions. When a student requests to create a point on a graph, does the tool wait for permission from a tutor to approve the request before creating the point (synchronous operation), or create the point, allow the student to continue, and then deliver tutor feedback (asynchronous operation)? Waiting for permission means that the tool cannot function independently of a particular tutor, but providing asynchronous feedback might decrease usability. One solution could be to increase tool functionality so that it can always undo the previous action. However, this solution is problematic for usability, as it may be confusing to the student to see their action mysteriously reversed (e.g., creating a point and then immediately delet-

ing it). The best solution may be to sacrifice reuse and have certain actions, like user "create" actions, require an immediate response from a tutor.

The final decision in component separation is whether to give the tool knowledge about tutoring. When an entry in a spreadsheet is flagged, does the tool or the tutor know to display the error by turning the text red? If the knowledge is located in the tool, tutors will not need to know about the specifics of each tool in order to send approve and flag messages. However, having the tool know about the tutoring might be unreasonable in situations where tools are pre-designed; one would not expect Microsoft Excel to be developed with a tutoring framework in mind! Developers have to be prepared both to use pre-designed tools and to design tools specifically for tutoring. When using an off-the-shelf tool in a tutoring scenario, a translator component can be built to transform a flag message into a more specific series of messages for that particular tool. When designing or modifying a tool for use in a component-based tutoring framework, it should have feedback behaviors built in. Although theoretical architectures specify that tool and tutor components should be separated, examining the issue yields a set of guidelines on where that separation should occur.

4 Evaluation of Progress

We now evaluate our efforts in extending the CTAI into a research platform. We wished to retain the strengths of the CTAI while adding collaborative functionality. To this end, we expanded the CTAI to incorporate a peer tutoring script (PTS). We have implemented the architecture from Section 2, while negotiating the challenges discussed in Section 3. Our changes form the beginning of a flexible, controlled, and practical framework for future extensions.

4.1 Proposed Extension to the Cognitive Tutor Algebra-1

The CTAI focuses on "the mathematical analysis of real-world situations and the use of computational tools" [1]. Students read a word problem and use various tools to solve the problem. As students work, the cognitive tutor monitors their progress based on a model of performance and gives immediate feedback. When students make an error, the cognitive tutor will immediately "flag" it (e.g., by turning input text red) and, for common errors, output a message that explains the misconception. At any time, the student can request help, and the cognitive tutor will provide hints. The tutor keeps an estimate of student mastery of skills. Skill levels are displayed so that students are aware of their progress, and problems are chosen based on skills that students have not yet mastered.

We are integrating collaboration and metacognitive tutoring into the CTAI using a peer tutoring script. Instead of the computer tutoring students on math, students tutor each other, while the computer provides collaborative support. The human peer tutors prepare by solving the problem that they will be teaching with the help of the computer tutor. While tutoring, peer tutors mark tutees' answers as wrong or right, provide hints and feedback in a chat window, and assess the tutees' skill mastery. Peer tutees can also engage the tutor in discussion in the chat window. See Figure 2 for a screenshot of the peer tutor's interface. Peer tutoring scripts have improved learning, particularly when peer tutors prepare ahead of time, peer tutors provide elaborated explanations which

peer tutees use constructively, and students set goals for the tutoring and monitor skills being acquired [17]. We believe that the PTS encourages these elements and will enhance the effectiveness of the CTAI.

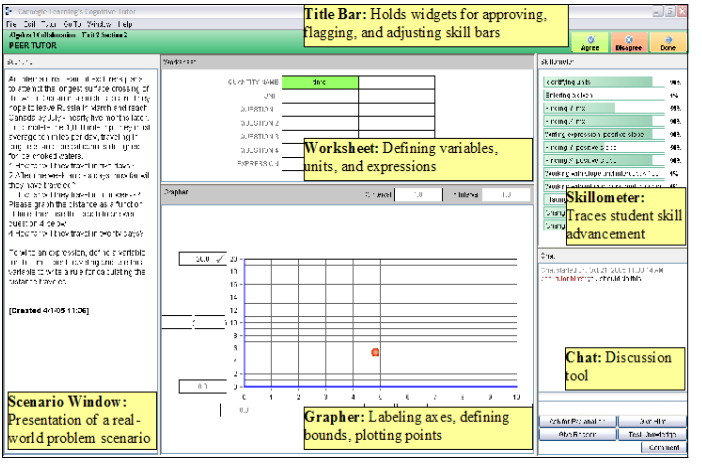

Fig. 2. Screenshot of peer tutor's interface in the peer tutoring script (PTS)

4.2 Implementation of Extensions

To implement PTS, we changed the CTAI so that the tool and tutor components function independently of each other, are remotely located, and communicate through a mediator, as illustrated in Figure 3. Although the CTAI was designed to follow the architecture described in [15], development constraints lead its current state to evolve from this ideal. Components were dependent on each other for launching, and some tool functionality was located within the tutor. The tool and tutor were capable of communicating remotely using the message protocol defined in Section 1, but there was no mediator in place. We created central classes that could function remotely to control functions such as beginning a session, launching the tutor, and shutting down the tutor. We added a mediator to intercept remote messages between the tool and tutor.

We then created some new tutor components, which required further negotiation of the tool/tutor separation. We built an echoing module to echo input from one user's screen onto the other. The echoing tutor receives user action messages, and transforms them into the appropriate tutor feedback and action messages. For example, a "note input" message on a given widget would be changed into a parallel "send input" message. To make the echoing tutor effective and reusable, we implemented most of the tool-side functionality detailed in Section 3.2, and improved tool response to tutor action messages, which were rarely used in the CTAI. We tested the echoing tutor in a configuration with the original cognitive tutor, and two original tools (see Figure 3). We also implemented a prototype metacognitive tutor to listen for certain messages and provide metacognitive instruction when appropriate.

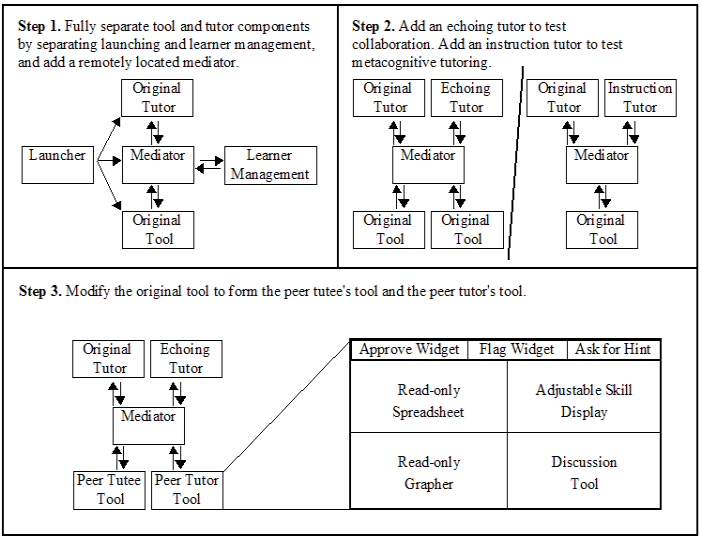

Fig. 3. Three steps in extending the CTAI to implement the peer tutoring script

Finally, we created the tool components required for the PTS, attempting to plan for both usability and reuse. We added a chat tool for the peer tutor and tutee. We also modified the peer tutor's tools, disabling widgets for inputting answers, and adding widgets for approving and flagging to the title bar. This setup is more usable because these new widgets are in one place, but reusable in that changing the title bar removes peer tutoring functions from the tools while retaining other characteristics.

4.3 Results and Discussion

We evaluate our implementation with respect to our criteria for a good research platform: flexibility, control, and practicality. The framework that we have developed is flexible in terms of the potential reuse of components that have been implemented and the variety of new components that could be integrated. In addition to the peer tutoring extension to the CTAI, developed components suggest other extensions. Combining the echoing agent with two regular tools yields a collaborative setup for solving cognitive tutor problems. Combining the echoing agent and the cognitive tutor agent with a regular tool and a modified peer tutor tool so that the peer tutor cannot input values or perform tutoring actions yields an actor/observer configuration, where one person solves the problem, the other watches.

A variety of new components can be integrated into this framework, illustrating the extensibility of the CTAI. The mediator potentially allows any components to connect to it, as long as they include a translator class to translate the messages into the protocol we have developed. For example, we intend to use the framework to implement another collaborative session type, called the *collaborative problem-solving script* (CPS), where two students alternate between working independently and together to solve problems [18]. Students collaborate at the same computer terminal. They require a tool modified for the requirements of the CPS, the cognitive tutor, an instruction tutor that listens for certain messages from the cognitive tutor and provides

scripted instruction, and a collaborative help tutor that listens for certain messages from the tool and provides adaptive support. Adding components to the CTAI would not be possible without the implemented framework.

The framework can also ensure experimental control in research by facilitating ablation studies to examine the independent effects of components. Because components have no knowledge of each other but are connected through the mediator, specifying a different configuration in the mediator can remove the component. For example, it is simple to compare versions of the PTS with or without a specific tutoring agent, simply by changing the mediator to include or exclude that agent. One could also make comparisons between different versions of the tools; for example, comparing the differing effects of using the regular skill display to the effects of using the peer tutorable display. In the CPS, removing the instruction tutor and/or the collaborative help component allows different interventions to be compared. The implemented framework allows examination of why an intervention is effective.

Practicality is the final criteria for evaluating our implementation. We have attempted to design for usability in addition to reuse, and tried to keep the development demands for reuse to a minimum. We have extended the CTAI, a tutor that has been shown to be effective, which means we are comparing our interventions to a gold standard. The tutor is already used in roughly 2000 classrooms across the United States, and is an integral part of the Pittsburgh Science of Learning Center (PSLC), which is engaged in facilitating experimental research in real classrooms. Using the CTAI means there is institutionalized support with accessing teachers, schools, and CTAI developers. At this early stage, it appears that the extended CTAI is practical for development and classroom use.

We have developed a theoretical architecture for extending an established cognitive tutor into a platform for collaborative and metacognitive experimentation, discussed some practical challenges with the implementation, and evaluated our efforts to expand the CTAI using a peer tutoring script. We will soon be evaluating the PTS in the classroom, and using the framework with other scenarios such as the CPS, which should give us a clearer idea of the effectiveness of the CTAI as a research platform. Other established tutoring systems can be extended in a similar manner, using a multi-component architecture that is specific about component responsibilities and takes usability needs and development schedule into account. Reconceptualizing the cognitive tutor as a research platform is a powerful idea for furthering educational research and improving intelligent tutoring technology.

Acknowledgements

This research is funded by the National Science Foundation, award #0354220. I thank Jonathan Steinhart, Dale Walters, Steve Ritter, and Dejana Diziol for their help.

References

1. Koedinger, K.R.; Anderson, J.R.; Hadley, W.H.; and Mark, M.A.: Intelligent Tutoring Goes to School in the Big City. *Journal of Artificial Intelligence in Education, 8.* 30–43. (1997).
2. Morgan, P., & Ritter, S. (2002). An experimental study of the effects of Cognitive Tutor® Alegbra I on student knowledge and attitude. (Available from Carnegie Learning, Inc., www.carnegielearning.com/research/research_reports/morgan_ritter_2002.pdf).

3. Johnson, D. W. and Johnson, R. T. (1990). Cooperative learning and achievement. In S. Sharan (Ed.), *Cooperative learning: Theory and research* (pp. 23-37). NY: Praeger.
4. Aleven, V., McLaren, B. M., Roll, I. and Koedinger, K. R. (2004). Toward Tutoring Help Seeking: Applying Cognitive Modeling to Meta-Cognitive Skills; *Proceedings of the 7th International Conference on Intelligent Tutoring Systems* (ITS-2004).
5. Beck, J.E., Mostow, J., & Bey, J. (2004). Can automated questions scaffold children's reading comprehension? *The 7th International Conference on Intelligent Tutoring Systems.*
6. Aleven, V., Roll, I., McLaren, B. M., Ryu, E. J., & Koedinger, K. R. (2005). An architecture to combine meta-cognitive and cognitive tutoring: Pilot testing the Help Tutor. *Proceedings of 12th International Conference on Artificial Intelligence in Education*.
7. Weinberger, A., Reiserer, M., Ertl, B., Fischer, F. & Mandl, H. (2003). Facilitating collaborative knowledge construction in computer-mediated learning with structuring tools. In R. Bromme, F. Hesse & H. Spada (Eds.), *Barriers and Biases in network-based knowledge communication in groups*. Dordrecht: Kluwer.
8. McLaren, B. M., Bollen, L., Walker, E., Harrer, A., and Sewall, J (2005). Cognitive Tutoring of Collaboration: Developmental and Empirical Steps Toward Realization. *Proceedings of the Conference on Computer Supported Collaborative Learning*,.
9. Krueger, Charles W.. Software reuse. *Computing Surveys*, 24(2):131–183, June 1992.
10. Roschelle, J., Kaput, J., Stroup, W., & Kahn, T. M. (1998). Scaleable integration of educational software: Exploring the promise of component architectures. *Journal of Interactive Media in Education* (6).
11. ADL (2004a). *Sharable Content Object Reference Model 2004 2nd Edition Overview*..
12. Brusilovsky, P. KnowledgeTree: A distributed architecture for adaptive e-learning. In *Proc. of WWW2004 - The Thirteen International World Wide Web Conference*, 2004.
13. Vassileva, J., McCalla, G., and Greer J. (2003). "Multi-Agent Multi-User Modeling", *User Modeling and User-Adapted Interaction*, *13* (1-2), 179-210.
14. Muhlenbrock, M., Tewissen, F. and Hoppe, H. U.: A framework system for intelligent support in open distributed learning environments, *International Journal of Artificial Intelligence in Education,* 9, 256-274.
15. Ritter, S. and Koedinger, K.R. An architecture for plug-in tutor agents. Journal of Artificial Intelligence in Education, 7, 3/4 (1996), 315-347.
16. Ritter, S. (1997). Communication, cooperation and competition among multiple tutor agents. In du Boulay, B. and Mizoguchi, R. (Eds). *Artificial Intelligence in Education: Knowledge and media in learning systems* (pp. 31-38). Amsterdam: IOS Press.
17. Walker, E. (2005). Mutual peer tutoring: A Collaborative Addition to the Cognitive Tutor Algebra I. Accepted as a Young Researcher's Track paper at the *International Conference on Artificial Intelligence and Education*.
18. McLaren, B. M., Walker, E., Koedinger, K., Rummel, N., Spada, H., and Kalchman, M. (2005). Improving Algebra Learning and Collaboration through Collaborative Extensions to the Algebra Cognitive Tutor., Poster Presented at CSCL-05, Taipei, Taiwan.