COVER PAGE

Toward a Rapid Development Environment for Cognitive Tutors

Kenneth R. KOEDINGER
Vincent A.W.M.M. ALEVEN
Human-Computer Interaction Institute
Carnegie Mellon University, Pittsburgh, PA, USA

Neil HEFFERNAN
Computer Science Department
Worcester Polytechnic University, Worcester, MA, USA

Contact Information:
Kenneth R. Koedinger
5000 Forbes Avenue
Human-Computer Interaction Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Phone: 412 268 7667
Fax: 412 268 1266
Email: koedinger@cmu.edu

Abstract. We are developing a suite of Cognitive Tutor Authoring Tools (CTAT) intended to make tutor development both easier and faster for experienced modelers and possible for potential modelers who are not experts in cognitive psychology or artificial intelligence programming. Our goal is to demonstrate a reduction in development time by a factor of three. We employ Human-Computer Interaction (HCI) methods and Cognitive Science principles to design development tools that are both useful and useable. Our preliminary analytic and empirical analyses compare use of CTAT with use of our current develop environment and indicate a potential reduction in development time by a factor of about two.

Toward a Rapid Development Environment for Cognitive Tutors

Kenneth R. KOEDINGER
Vincent A.W.M.M. ALEVEN
Human-Computer Interaction Institute
Carnegie Mellon University

Neil HEFFERNAN
Computer Science Department
Worcester Polytechnic University

Abstract. We are developing a suite of Cognitive Tutor Authoring Tools (CTAT) intended to make tutor development both *easier and faster* for experienced modelers and *possible* for potential modelers who are not experts in cognitive psychology or artificial intelligence programming. Our goal is to demonstrate a reduction in development time by a factor of three. We employ Human-Computer Interaction (HCI) methods and Cognitive Science principles to design development tools that are both useful and useable. Our preliminary analytic and empirical analyses compare use of CTAT with use of our current develop environment and indicate a potential reduction in development time by a factor of about two.

Cognitive Tutors have been demonstrated to yield dramatic improvements in student learning. For example, evaluations of the Algebra Cognitive Tutor have demonstrated that students in tutor classes outperform students in comparison classes [4]. This tutor is being marketed and in use in over 1000 schools across the US (see www.carnegielearning.com). Despite the great potential of Cognitive Tutors to improve student learning in other areas, development of such systems is currently costly and has been limited to just a few research teams. Such teams currently require PhD level expertise in cognitive task analysis and advanced AI programming to create the cognitive models that drive Cognitive Tutors.

We have begun to create a development environment that addresses these difficulties. Our goal is to make tutor development both *easier and faster* for current developers and *possible* for researchers, trainers, and educators who are not experts in cognitive psychology or AI. We are designing, implementing, and evaluating Cognitive Tutor Authoring Tools (CTAT) that will support all phases of design and development. Creating an effective development environment is as much about getting the HCI details right as it is about innovation in algorithms. We use both empirical HCI methods, like Think Aloud user studies [3], as well as analytical methods, like Keystroke Level Modeling [2], to guide interface design. We employed such methods in an earlier project [5] involving the redesign of part of RIDES [6], an authoring environment for simulation-based intelligent tutoring systems, and were able to reduce programmer time by a factor of 2.6.

1. The Cognitive Tutor Authoring Tools

Our rapid development environment, illustrated in Figure 1, consists of the following tools:
- *An Intelligent GUI Builder*, whose windows are shown in the top-left of Figure 1, can be used to create a graphical user interface (GUI) to be used in the tutor. The modeler can use the interface to demonstrate how to carry out the task to be modeled.
- A *Behavior Recorder* (top right), which records solution paths through a given problem scenario, as the modeler demonstrates these paths in the GUI.
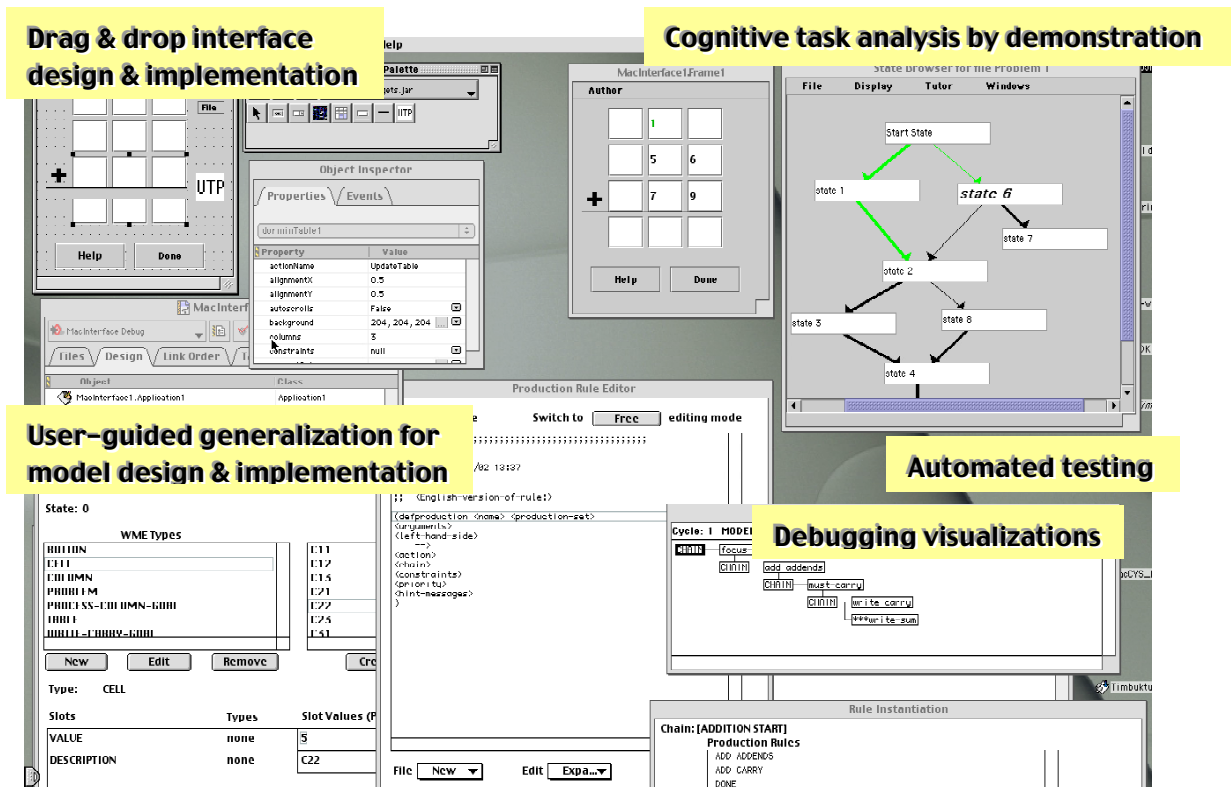
Drag & drop interface design & implementation

Cognitive task analysis by demonstration

User-guided generalization for model design & implementation

Automated testing

Debugging visualizations

Figure 1: The prototype Cognitive Tutor Authoring Tools

- A *WME Editor* and a *Production Rule Editor*, dedicated editors used to implement the production rules that model the demonstrated paths (bottom left and bottom middle).
- A debugging tool called the *Cognitive Model Visualizer, which* has two windows shown in the bottom right ("Conflict Tree" and "Rule Instantiation").

2. Preliminary Empirical and Analytic Evaluations to Guide Design

In order to get an initial impression of the savings afforded by CTAT and of the ways in which the tools might be improved, we conducted preliminary analytical and empirical evaluations. Our preliminary analysis of CTAT used a method called the Keystroke Level Model (KLM) [2]. KLM is a way of estimating the time required for expert performance on routine tasks in a computer interface. The analyst creates a detailed specification of the task at the level of keystrokes and mouse clicks and uses it to estimate the time the task will take. Time estimates derived from KLM correlate well with expert performance times [2].

Using this method, we compared our existing modeling tools, an environment called TDK [1], which has been used for over a decade to create many large-scale Cognitive Tutors to both 1) the initial CTAT environment created in four months and 2) a preliminary redesign of CTAT that was not implemented at the time of the initial evaluation. We created simplified KLM models for three common modeling tasks, namely (1) creating the initial configuration in working memory for a problem scenario, (2) writing a production rule of medium complexity, and (3) debugging why a rule that was expected to fire did not. As shown in Table 1, the KLM analysis predicts that the current CTAT will reduce the time needed to create an initial working memory configuration by a factor of 2.3. It predicts lesser savings for the other tasks. However, the results in the future CTAT column indicate significant future savings in debugging, where programmers spend much of their time.

We also conducted a preliminary *empirical* analysis comparing the amount of time it takes to complete a modeling task with the existing TDK and the current preliminary version of CTAT. The task was to implement, test, and debug a working memory

Table 1: Results of a preliminary evaluation of CTAT using Keystroke Level Models: estimates of the time (in seconds) spent at the keystroke level for three commonly-occurring modeling tasks

	TDK	Current CTAT	Future CTAT
Initialization	209.3	90.6	(90.6)
Writing rule	203.1	207.3	130.1
Debugging	39.3	30.8	12.5

representation of a problem scenario and a single production rule. One of the authors completed this task in 50 minutes using TDK. He then did the same task using CTAT, this time taking only 15 minutes. Finally, to address the possibility of confounding effects due to learning, the same author re-did the task using the TDK tools. This time he needed 30 minutes. Thus the time savings due to CTAT were considerable. As shown in Table 2, the majority of savings occurred at the debugging stage. We anticipate that there will be further savings as we continue to develop the tools and scale up to larger evaluation studies.

Table 2: Minutes spent on various sub-tasks for each trial in a preliminary empirical evaluation of CTAT.

	1. TDK	2. CTAT	3. Redo TDK
Initialization	10	5	8
Writing Rule	12	5	5
Testing & Debugging	28	5	17
Total	**50**	**15**	**30**

3. Conclusion

CTAT integrates cognitive task analysis, knowledge acquisition, and model building to yield efficiencies. Current estimates are that it takes 100-1000 hours of development for one hour of ITS instruction [7]. CTAT will be successful if we can demonstrate that it can reduce development time by at least a factor of three while maintaining or increasing system quality. We presented preliminary evidence that CTAT, even in its early stage of development, may substantially reduce the time needed to build and test cognitive models and tutors. More detailed disaggregation of our analytic and empirical results will provide specific guidance for where redesign efforts are most likely to yield further cost savings. A further criterion for success is that the tools make Cognitive Tutor development feasible for a larger range of developers and also a broader range of education and training domains.

Acknowledgement

References

[1] Anderson, J.R., & Pelletier, R. (1991). A development system for model-tracing tutors. In *Proceedings of the International Conference of the Learning Sciences,* 1-8.

[2] Card, S.K., Moran, T.P., & Newell, A (1983). *The Psychology of Human-Computer Interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates.

[3] Ericsson, K. A., & Simon, H. A. (1984). *Protocol Analysis: Verbal Reports as Data*. Cambridge, MA: The MIT Press.

[4] Koedinger, K. R., Anderson, J. R., Hadley, W. H., & Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, *8,* 30-43.

[5] Mathan, Koedinger, Corbett, & Hyndman (2000). Effective strategies for bridging gulfs between users and computer systems. In *Proceedings of HCI-Aero 2000*. Toulouse, France. pp 197 - 202

[6] Munro, A. (1994). RIDES Authoring Reference. Behavioral Technology Labs, USC.

[7] Murray, T. (1999). Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education, 10*, pp. 98-129.