Toward a Rapid Development Environment for Cognitive Tutors
Interactive Event during AIED-03

Kenneth R. KOEDINGER
Vincent A.W.M.M. ALEVEN
Human-Computer Interaction Institute
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213, USA
koedinger@cmu.edu, aleven@cs.cmu.edu

Neil HEFFERNAN
Computer Science Department
Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609-2280, USA
nth@WPI.EDU

1. **Introduction**

Cognitive Tutors have been very successful (Anderson, Corbett, Koedinger, Pelletier, 1995; Koedinger, Anderson, Hadley & Mark, 1997) but take a very significant amount of development effort. We are developing a suite of Cognitive Tutor Authoring Tools (CTAT), shown in Figure 1, 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 concrete goal is to experimentally demonstrate a reduction in development time by a factor of three. We are employing Human-Computer Interaction (HCI) methods and Cognitive Science principles to design development tools that reduce the time programmers and cognitive scientists need to spend in order to develop a Cognitive Tutor. Our preliminary analytic and empirical analyses compare use of CTAT with use of our current development environment and indicate a potential reduction in development time by a factor of about two.

The tools are currently in use by people in our research group at Carnegie Mellon University to develop a Cognitive Tutor for introductory genetics. One unit of this tutor that was partly developed with the tools is currently being pilot-tested at two universities. Also, the tools were used by the students in a graduate level/upper undergraduate level course on intelligent tutoring systems taught at CMU this spring semester. Finally, the tools were used by the participants of the 3rd Annual CIRCLE Summer School on Intelligent Tutoring Systems, which was held recently at Carnegie Mellon University.

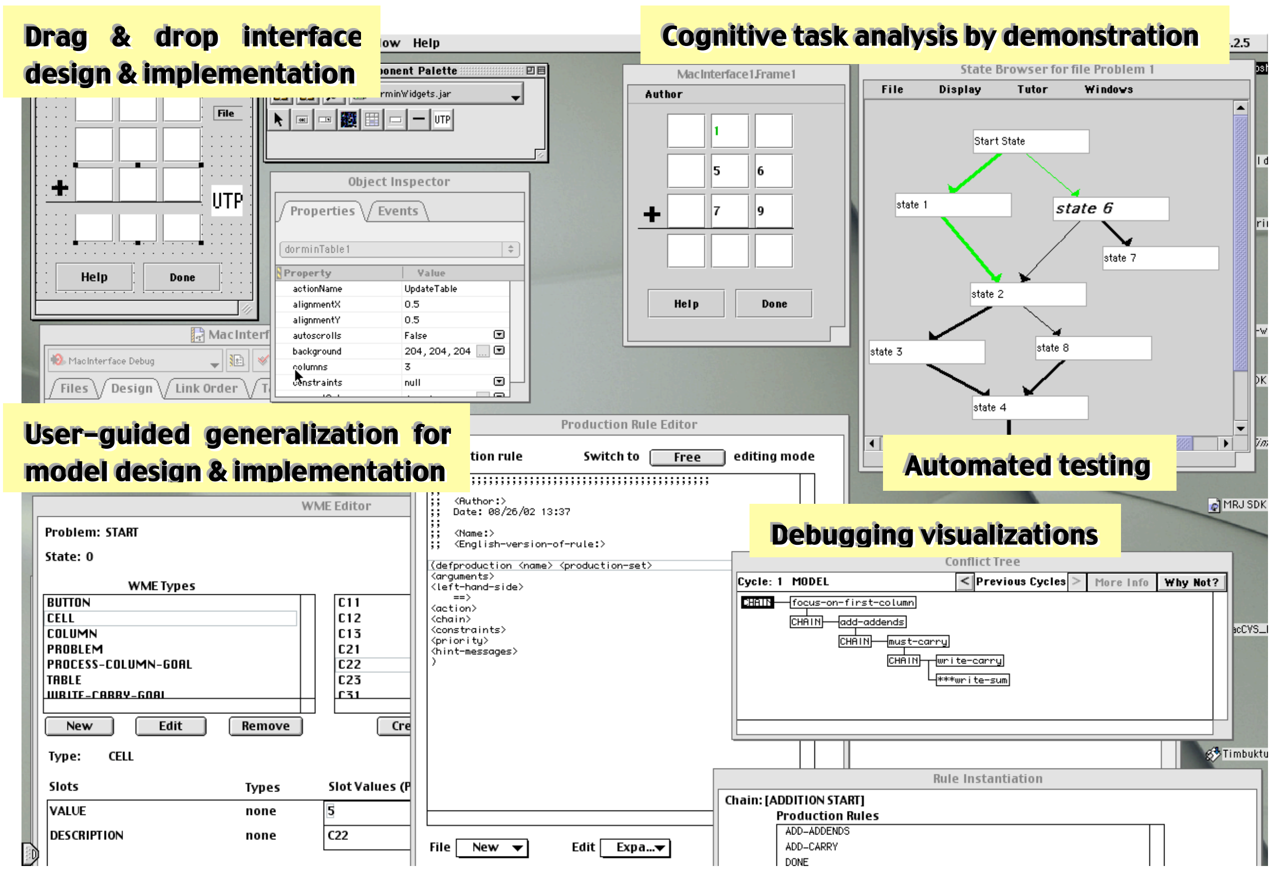
Drag & drop interface design & implementation
Cognitive task analysis by demonstration
User-guided generalization for model design & implementation
Automated testing
Debugging visualizations

Figure 1: The prototype Cognitive Tutor Authoring Tools

2. The Cognitive Tutor Authoring Tools

Our rapid development environment, illustrated in Figure 1, consists of the following tools:

- *A GUI Builder* (top left) to create a graphical user interface (GUI) for the tutor.
- *A Behavior Recorder* (top right), used (a) to record solution paths for a given problem, (b) to create a problem-specific "pseudo tutor" and (c) to test a production rule model.
- *A WME Editor* and a *Production Rule Editor*, dedicated editors to create a production rule model.
- A debugging tool called the *Cognitive Model Visualizer*.

3. Event outline

The event will have three main sections: introduction, hands-on, and discussion. During the *introduction*, we will discuss the goals and planned contributions of the project and provide background information on Cognitive Tutors. Each Cognitive Tutor has a cognitive model that specifies how a student might reasonably go about solving a problem in the given application area. The tutor uses the model to follow the student's actions and provide step-by-step guidance, in a process called model-tracing (Anderson, et al, 1995).

During the *hands-on part* of the event, the participants will work with the tools to develop part of a tutor for a simple domain such as multi-column addition. First, they will create a "pseudo tutor" which, on a single problem, behaves like a full-blown Cognitive Tutor but is much easier to build. The pseudo tutor will then be used as the basis for creating part of a cognitive model that can drive a "full" Cognitive Tutor.

In order to build the pseudo tutor, the participants will develop a user interface using the GUI Builder tool, dragging and dropping widgets as needed. Having completed the GUI, they will set up a tutor problem in the GUI and demonstrate one or more solution paths, simply by performing solution steps in the GUI. The Behavior Recorder will automatically record the demonstrated paths in a "behavior diagram". The nodes in the diagram represent problem-solving states (which typically correspond to the states that the GUI goes through in the process of solving the given problem), the links potential student actions. The participants will attach hint sequences to links in the diagram. They will also add links that represent common student errors and attach "bug messages" to these links. At this point, the pseudo tutor will be complete. The participants will test it to see that it provides useful help and feedback on the given problem. The capability to build pseudo tutors is useful to do rapid prototyping or to build tutors in domains where there are few problem instances per problem type. It is useful also because the behavior diagram guides the construction of a cognitive model.

The next step for the participants of the Interactive Event will be to construct some components of a cognitive model that can be used to drive tutoring over a wider range of problems. In general, a production rule model consists of a specification of the objects in "working memory" and a set of production rules (or "if-then" rules). CTAT takes over a significant portion of the first task: it generates an initial version of the working memory objects needed to represent the initial state of a given problem. The participants will write a few production rules that model some of the problem-solving steps recorded in the behavior diagram —the diagram now functions as a specification of the desired behavior of the model to be built, on the given problem. The Production Rule Editor supports a process that we call "user-guided generalization": the author first constructs a concrete or "instantiated" version of the rule and then generalizes the concrete rule by removing constants, introducing variables, and adding list matching patterns. The behavior diagram is now used for semi-automated testing. Upon the author's request, the Behavior Recorder checks which links in the diagram have been "modeled correctly" and color codes the links accordingly. When the production rule model does not behave as intended, the participants will use the debugging tools to find out how to fix the problem. Finally, they will test their partial Cognitive Tutor, verifying that it provides useful hints and feedback to students.

During the *discussion* section, we would like to discuss with the participants what they see as desirable features for an authoring environment for intelligent tutoring systems and what existing components they might want to hook up. We would also like to establish contacts with participants who are interested in using the tools in their own projects (versions of some of these tools can be found at http://nth.wpi.edu/CTAT.htm).

Acknowledgements

This research is supported by ONR grants N00014-02-1-0443 and N00014-03-1-0220. Thanks to programmers Vanessa DeGennaro, Chang-Hsin Chang, Mike Schneider, Noble Shore, Zhenhua Zhang, Yogi Samant, and Sanket Choksey and for comments from Ryan Baker.

References

Anderson, J. R. (1993). *Rules of the Mind.* Hillsdale, NJ: Erlbaum.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences,* 4 (2), 167-207.

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.