Rapid Authoring of Intelligent Tutors for Real-World and Experimental Use

Vincent Aleven, Jonathan Sewall, Bruce M. McLaren, Kenneth R. Koedinger
Human-Computer Interaction Institute, Carnegie Mellon University
{aleven|sewall|bmclaren}@cs.cmu.edu, koedinger@cmu.edu

Abstract

Authoring tools for Intelligent Tutoring Systems are especially valuable if they not only provide a rich set of options for the efficient authoring of tutoring systems but also support controlled experiments in which the added educational value of new tutor features is evaluated. The Cognitive Tutor Authoring Tools (CTAT) provide both. Using CTAT, real-world "Example-Tracing Tutors" can be created without programming. CTAT also provides various kinds of support for controlled experiments, such as administration of different experimental treatments, logging, and data analysis. We present two case studies in which Example-Tracing Tutors created with CTAT were used in classroom experiments. The case studies illustrate a number of new features in CTAT: Use of Macromedia Flash MX 2004 for creating tutor interfaces, extensions to the Example-Tracing Engine that allow for more flexible tutors, a Mass Production facility for more efficient template-based authoring, and support for controlled experiments.

1. Introduction

Intelligent Tutoring Systems (ITSs) have been proven to be successful in improving students' learning [7, 13]. Inspired by such results, many authoring tool kits have been built for ITSs [1, 10, 11]. Typically, the goal is to make the development of tutors easier and faster. However, authoring tools may serve a second important purpose, namely, to make it easier to use ITSs as platforms for empirical experiments.

For example, researchers may want to test a hypothesis stating that a certain feature in the design of an ITS leads to deeper or more efficient learning on the part of students. To do so, they develop two tutor versions, one with the feature and one without, and have different groups of students work with each tutor version. They administer a pre-test and a post-test and instrument the software to write detailed logs of student-tutor interactions. Finally, they analyze the data to see if there are differences in learning gains between the conditions and differences in the interactions students had with the tutor. This kind of experimentation is of course not novel, but facilitating it by means of better tools will help move the empirical science of ITSs forward.

We are developing a set of authoring tools, called the Cognitive Tutor Authoring Tools (CTAT) that addresses these goals: easier and efficient creation of tutors for both real-world use and use in experimental scenarios [2, 6]. Currently, CTAT supports development of two types of tutors: Cognitive Tutors, which have a long and successful track record [7] but require development of a cognitive model through AI programming, and a new type of tutors, "Example-Tracing Tutors," which can be built entirely without programming. To create an Example-Tracing Tutor, an author demonstrates example solutions, generalizes the recorded examples, and annotates them with hints and feedback messages. CTAT also provides tools to support experimental scenarios such as the one described above, including facilities for detailed logging of student-tutor interactions and log analysis (see also [1, 11]).

In the current paper, we illustrate how CTAT supports experiments involving Example-Tracing Tutors, by means of two case studies, one in the area of chemistry, in particular stoichiometry, and one in the area of thermodynamics. The properties and authoring process of Example-Tracing Tutors have been described in a previous paper (Koedinger et al, 2004). Here, we focus on a number of new features: (1) new functionality to generalize recorded examples for more flexible Example-Tracing Tutors, (2) use of Macromedia Flash MX 2004 to create tutor interfaces, (3) a template-based approach to facilitate authoring and make it more efficient (4) support for delivering a fixed sequence of tutor problems on the web, and (5) support for experimentation: on-line tests and consent forms, as well as log recording and analysis.

CTAT is freely available for research and educational purposes (see http://ctat.pact.cs.cmu.edu). We organize a free annual summer school to help people get up to speed with CTAT. So far, over 200 instructors, researchers and students have used CTAT.

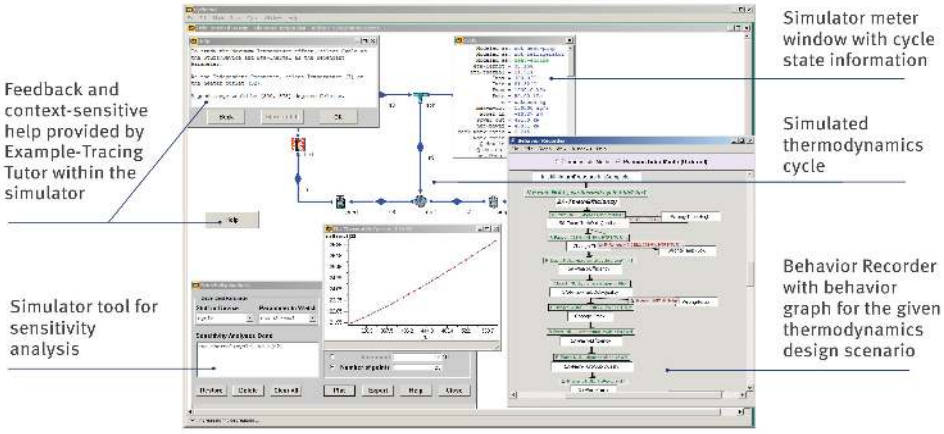

Figure 1: Use of CTAT to develop tutoring capabilities for the CyclePad thermodynamics simulator

2. Brief overview of CTAT

When developing Example-Tracing Tutors for use in an experiment, an author is likely to use the following CTAT tools and components:

- the Behavior Recorder, a tool to create Behavior Graphs, which are maps of the solution space used by Example-Tracing Tutors (see Figures 1-3);
- a Mass Production facility for template-based authoring of Example-Tracing Tutors (Figure 3);
- the Tutor Shop, a component used to sequence tutor problems in a web-based environment and to support experimental use of CTAT-based tutors;
- the Data Shop, a fully-integrated web-based service for logging, log analysis, and reports.

A key advantage of Example-Tracing Tutors is that a full-fledged, real-world tutor can be built "by demonstration," that is, without programming. An author can either start with an existing problem-solving environment or simulator (when the goal is to provide tutoring within that environment), or create a new tutor interface from scratch. Hooking up an external "tool" requires some programming. Creating a new tutor GUI on the other hand can be done without programming, using a set of CTAT-compatible interface components with off-the-shelf GUI building tools such as Netbeans (for Java) or Flash MX 2004 (see Figure 2).

Next, the author must demonstrate examples of correct and incorrect behavior for each problem on which students will be tutored. The examples are recorded by CTAT in a Behavior Graph with links and nodes that represent problem-solving steps and states respectively (see Figures 1-3). An author may demonstrate alternative ways of solving a problem, which are recorded as separate paths in the graph. An author may also demonstrate common student errors that the tutor must recognize. The corresponding links in the graph must be marked as representing incorrect actions. Finally, the author annotates the Behavior Graph with hints, feedback messages, and skill labels.

CTAT's Example-Tracing Engine uses the Behavior Graph to guide a student through a problem, comparing the student's problem-solving behavior against the graph. It provides positive feedback when the student's behavior matches steps in the graph, and negative feedback otherwise. If the student's input matches a link in the graph that was marked as an incorrect action, then any error feedback message attached to that link is presented to the student. When the student requests a hint, the hint messages attached to a link out of the current state in the graph are displayed. As discussed further in [6], Example-Tracing Tutors thus provide the same key features in tutor behavior as Cognitive Tutors. Our experience gained during workshops, summer schools, and courses indicates that non-programmers typically learn to build Example-Tracing Tutors in an afternoon.

3. Case study 1: More flexible tutors

The first of our two case studies involves the use of CTAT to add a tutor agent to an existing "articulate simulator" for thermodynamics, called CyclePad, shown on the left in Figure 1 [3, 5]. (The Behavior Recorder is on the right.) The CyclePad simulator enables students to build, analyze, and optimize thermodynamic systems (or "cycles"), without having to perform extensive manual computations. The case study illustrates the use of CTAT to provide tutoring within an external simulator and the options that the Behavior Recorder provides for generalizing examples.

The goal of the thermodynamics project is to evaluate the value of computer-generated natural language tutorial dialogue in a context where, unlike earlier tutorial dialogue systems, students have the freedom to explore. We therefore added a dialogue

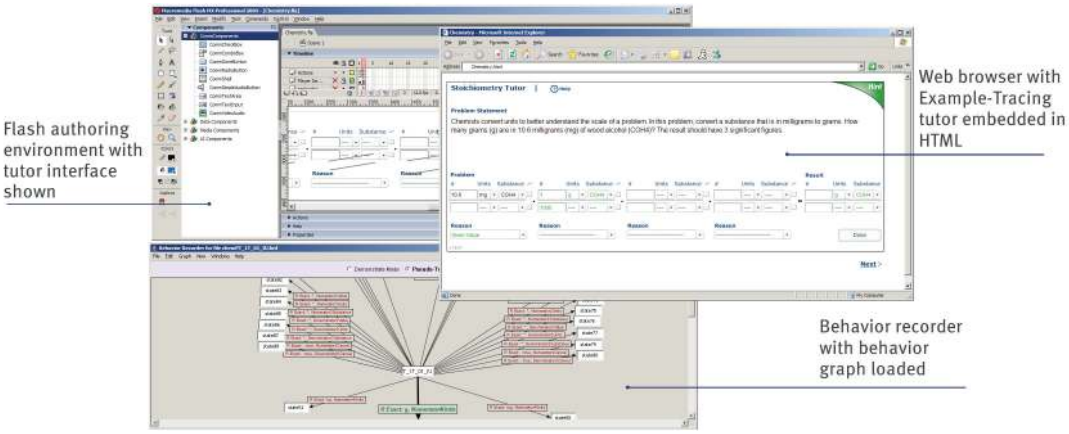

Figure 2: Macromedia Flash MX 2004 used to create a GUI for an Example-Tracing Tutor

component, built outside of CTAT. In a study with mechanical engineering students at Carnegie Mellon University in Pittsburgh (USA), it was shown that the dialogues enhance the effectiveness of the system [8].

In addition to the dialogue component, an Example-Tracing Tutor built in CTAT was added to the simulator, to provide tutorial support at points in the thermodynamics scenarios where dialogue was deemed unnecessary. After doing some programming to hook up the simulator [3], we created Behavior Graphs for three scenarios dealing with different designs for power plants. We took advantage of the options that CTAT offers to generalize examples for the purpose of flexible matching of student behavior. Early versions of the Example-Tracing Engine required that the student complete a problem scenario by exactly replicating one of the solution paths in the Behavior Graph, but that turned out to be too restrictive. Thus, we extended the Example-Tracing Engine in a number of ways. We added unordered and partially-ordered modes in which the restriction is lifted that the student must carry out the problem steps in the demonstrated order. In unordered mode, the students can do the steps in any order. In partially-ordered mode, "unordered groups" can be defined; the groups themselves must be completed in the order that they appear in the graph, but the steps within a group can be carried out in any order. The partially-ordered mode was useful for the thermodynamics scenarios, which naturally break down into two phases, one in which the student initializes a cycle by setting parameters such as pressure and temperature, and a second one in which they optimize the thermal efficiency of the cycle by adjusting the parameters. Within each scenario, we created unordered groups corresponding to each of these phases. As a result, the Example-Tracing Tutor appropriately requires that students finish the two phases in order, without placing any other constraints on the order of the actions within each phase.

Second, we used CTAT's "advanced matching options," which enable an author to generalize the conditions under which a demonstrated step is matched. For example, an author can specify that a numeric value input by the student should be within a certain range. This capability was useful, since in the optimization phase of the thermodynamics scenarios, we expect students to change the value of certain cycle parameters, but we do not know exactly what the new values will be, other than for example that they should be higher than the given initial value. Further, CTAT lets an author specify a regular expression. We used that facility to deal with situations in which different student actions essentially mean the same thing. For example, the working fluid can be specified at many different points within a thermodynamic cycle.

A third and final way in which CTAT allows an author to generalize examples is by specifying that certain steps in a problem solution are optional. Although we did not use this facility in the thermodynamics scenarios, we could have. In the optimization phase, for example, a student may change various cycle parameters an unspecified and unpredictable number of times. Modeling these actions as being optional will thus be very helpful.

4. Case study 2 : Flash and templates

We present a second case study [9] to illustrate two further CTAT features: the use of Flash to create tutor interfaces and the use of CTAT's Mass Production facility. Stoichiometry is the basic algebra used to account for substance quantities in chemical reactions. In this study, we explored how student learning with stoichiometry tutors might be improved by (a) personalized language in the tutor's hints (e.g., hints directly addressing the student using "you") and (b) the interleaving of worked examples and problem-solving exercises, as opposed to just problem solving. Each of

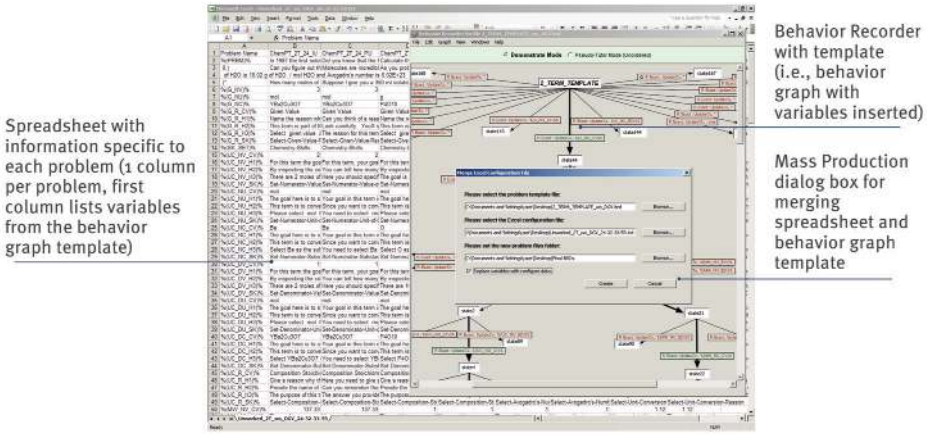

Figure 3: CTAT's Mass Production facility provides for template-based authoring using a spreadsheet

these interventions is supported by prior research (e.g., [4]) but has not been investigated in the context of ITSs. The experiment involved a 2x2 design, based on the two factors mentioned above. It was executed at the University of British Columbia (Canada) as an optional, online activity in two chemistry courses. The 69 subjects were randomly assigned to one of the four conditions. All were given an online pre-test, then worked on 15 tutor problems, presented according to the different experimental conditions, and finally, took an on-line posttest. We found, surprisingly, that personalization and worked examples had no significant effect on learning, although we did find a significant improvement from pre-test to post-test [9].

The first CTAT innovation used in this study was the use of the Flash programming tools in conjunction with CTAT to create a student interface (see Figure 2). Use of Flash led to a more modern user interface and easier web deployment. The Macromedia Flash IDE is also arguably easier to use than equivalent tools for Java, such as NetBeans and CodeWarrior.

We also made extensive use of CTAT's new "Mass Production" facility, which addresses a limitation of demonstration-based authoring of Example-Tracing Tutors, namely, that a significant amount of problem-specific authoring is necessary. While simply demonstrating a large number of solutions is not difficult, it is time consuming, especially when multiple authoring iterations are required. Further, maintaining consistency of hints and error messages across problems is a formidable task. CTAT's Mass Production facility considerably streamlines this process. Using this facility, an author first creates a single Example-Tracing Tutor by demonstration, as described above, and then turns it into a problem template by inserting variables for items that vary across problems, such as the values of steps, hints, and error feedback messages. The author then can create similar problems, of the same structure, simply by providing problem-specific information in an Excel spreadsheet. For each problem, values for all of the variables must be provided. A merge step, akin to that in Microsoft Word's mail merge facility, completes the process (see Figure 3). In the stoichiometry study, we used the Mass Production facility to develop 48 stoichiometry problems, both tutor problems and items for the on-line tests.

A significant advantage of Mass Production is that all of the problem instances are represented in a single file. This allows the author to easily copy across problems, making it easier to keep hints consistent across problems. It also facilitates maintenance: when a change must be made across all problems, one (often) only needs to change the problem template file. Of course, mass production works only if problems share the same interface widgets and the same problem structure. In the stoichiometry study, we needed 4 different templates, so, in spite of this requirement, the Mass Production facility was very helpful.

In the stoichiometry study, CTAT was used for the first time to field on-line pre-test and post-tests with automatic grading. We extended CTAT so that special Example-Tracing Tutors could be created that evaluate student input in the usual manner but do not provide hints or feedback. Student actions are logged in the usual manner (all CTAT-based tutors have logging capabilities built in), together with the tutor responses identifying correct and incorrect answers. Thus, the tests are essentially graded on the spot, with results stored in a database. Further, creating Behavior Graphs for test problems is easier than creating them for tutored problems. Since no hints or error messages are displayed to the student in pre- and post-tests, the author need only demonstrate correct solutions.

The stoichiometry study illustrates the use of two more components. In order to have the students

proceed through the problems in a fixed order, we created a web-based module, the "Tutor Shop," This module controls problem sequencing, using a data base to keep track of where each student is in the problem sequence. It serves several other purposes as well. First, it provides a programming interface to external authentication systems. We used this interface to permit the learner management system we were using, OLI [12], to perform user authentication both for account creation and subsequent login. Second, the Tutor Shop assigns students to experimental conditions. For our stoichiometry study, a simple round-robin algorithm sufficed, but other condition-assignment schemes are possible. Finally, the module can handle (pluggable) presentation pages to deliver content other than tutors, such as instructional videos, in its sequencing. This facility was used to provide online directions and to present the worked-out examples, video clips with voice narration showing the tutor interface as an expert solved problems.

A final component used in the stoichiometry study was the Pittsburgh Science of Learning Center (PSLC) Data Shop. The PSLC is an NSF-sponsored research center spanning Carnegie Mellon and the University of Pittsburgh (http://www.learnlab.org). The Data Shop is the PSLC's principal data repository: it bears the responsibility for standardized data collection for all experiments conducted in the center. It also provides data reporting and analysis services, including:

- 24-hour data logging server with web access;
- security and backup of collected data;
- student identity-masking and other measures to enforce privacy constraints;
- web access to queries that extract identity-masked data by time, course, unit, etc.;
- broadly-applicable data analyses, such as reports on error rates and learning curves, also available on demand from the web site.

CTAT is fully integrated with the Data Shop, meaning that CTAT-based tutors can automatically log the student-tutor interactions into the Data Shop data base. Thus, authors can obtain data extracts and view standard reports on data from their studies minutes after students finish with their tutors. In both the stoichiometry and the thermodynamics studies, we used the Data Shop to collect data about student-tutor interactions. In the stoichiometry study, in addition, the Data Shop was used to analyze the data and was instrumental in implementing on-line tests with automated grading.

Conclusion

The goals of the CTAT project are to make ITS development fast and accessible to non-programmers and to support the use of tutors in learning science experiments, including those run over the web. The CTAT tool suite supports authoring of two types of ITSs, Example-Tracing Tutors and Cognitive Tutors. Two case studies illustrate that Example-Tracing Tutors provide a viable way to create tutoring capabilities in advanced domains such as stoichiometry and thermodynamics. They also illustrate the way CTAT supports the use of tutors in learning science experiments. The innovations that were developed in support of the two experiments, meant to facilitate authoring and to increase the flexibility of Example-Tracing Tutors, are fully reusable in other projects.

References

[1] Ainsworth, S. & Fleming, P. (2005) Evaluating a mixed-initiative authoring environment: Is REDEEM for real? In *Proceedings AIED 2005* (pp. 9-16). Amsterdam: IOS Press.

[2] Aleven, V., McLaren, B., Sewall, J., & Koedinger, K. (in press) The Cognitive Tutor Authoring Tools (CTAT): Preliminary evaluation of efficiency gains. In *Proceedings ITS 2006*.

[3] Aleven, V., & Rosé, C. (2005). Authoring plug-in tutor agents by demonstration: Rapid, rapid tutor development. In *Proceedgins AIED 2005* (pp. 735-737). Amsterdam: IOS Press.

[4] Clark, R., & Mayer, R. (2003). *e-Learning and the Science of Instruction.* Jossey-Bass/Pfeiffer.

[5] Forbus, K., Whalley, P., Evrett, J., Ureel, L., Brokowski, et al. (1999). CyclePad: An articulate virtual laboratory for engineering thermodynamics. *Artificial Intelligence 114*(1-2), 297-347.

[6] Koedinger, K., Aleven, V., Heffernan, N., McLaren, B., & Hockenberry, M. (2004) Opening the door to non-programmers: authoring intelligent tutor behavior by demonstration. In *Proceedings ITS-2004* (pp. 162-174). Berlin: Springer.

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

[8] Kumar, R., Rosé, C., Aleven, V., Iglesias, A., & Robinson, A. (in press). Evaluating the effectiveness of tutorial dialogue instruction in an exploratory learning context. In *Proceedings ITS 2006*.

[9] McLaren, B., Lim, S., Gagnon, F., Yaron, D., & Koedinger, K. (in press) Studying the effects of personalized language and worked examples in the context of a web-based intelligent tutor. In *Proceedings ITS 2006.*

[10] Murray, T., Blessing, S., & Ainsworth S. (Eds.) (2003). *Tools for Advanced Technology Learning Environments.* Amsterdam: Kluwer.

[11] Nuzzo-Jones, G., Walonoski, J.A., Heffernan, N., Livak, T. (2005). The eXtensible tutor architecture: a new foundation for ITS. In *Proceedings AIED 2005* (pp. 902-904). Amsterdam: IOS Press.

[12] Carnegie Mellon University Open Learning Initiative, http://www.cmu.edu/oli.

[13] VanLehn, K., Lynch, C., Schultz, K., Shapiro, J. A., Shelby, R. H., Taylor, L., et al. (2005). The Andes physics tutoring system: *Lessons learned. International Journal of Artificial Intelligence and Education, 15*(3), 147-204.