AC 2008-1746: INTEGRATION OF AN INTELLIGENT TUTORING SYSTEM WITH A WEB-BASED AUTHORING SYSTEM TO DEVELOP ONLINE HOMEWORK ASSIGNMENTS WITH FORMATIVE FEEDBACK

Robert Roselli, Vanderbilt University

Stephen B. Gilbert, Clearsighted, Inc.

Larry Howard, Vanderbilt University

Stephen B. Blessing, University of Tampa

Aditya Raut, Vanderbilt University

Puvi Pandian, Iowa State University

© American Society for Engineering Education, 2008

Integration of an Intelligent Tutoring System with a Web-based Authoring System to Develop Online Homework Assignments with Formative Feedback

Abstract.

A web-based authoring tool, developed using VaNTH CAPE technology, is used to construct innovative online assignments that provide students with real time formative feedback as they attempt to solve quantitative engineering problems. The interactive system has found favor with instructors, teaching assistants and students. Because each step taken by the student in the problem solution is recorded by the accompanying learning management system, students and instructors can easily review modules to determine where the student went wrong. This approach also frees Teaching Assistants from the necessity of grading homework, most of which are worked correctly, and allows them to spend time with the students who most need their help. Because of the many options available in the authoring tool, novice developers often find it relatively difficult to design, construct and debug adaptive learning modules. The purpose of this work is to develop an Intelligent Tutoring System (ITS) that can be integrated with the authoring tool to provide personal guidance to new users as they develop homework problems. Previous ITSs have proved useful to learners in a wide variety of different domains, such as algebra, chemistry, and physics, resulting in gains of over one standard deviation. Clearsighted, Inc. has teamed with VaNTH to develop a version of the web authoring tool that allows information to flow between the ITS and the authoring tool. What is most interesting regarding this work is that the CAPE-based authoring tool was developed and works as its own stand-alone web-based application. Using Clearsighted, Inc.'s tools, an ITS was constructed that required no modification to the original authoring tool. The resulting ITS provides immediate feedback in a tutorial setting, offering help when requested and adaptive just-in-time messages, as well as noting incorrect actions. All of this feedback, from the user's point of view, seemingly comes from the authoring tool. A series of tutorials have been developed that will provide guidance to new users as they develop online homework assignments. Evaluation of the system is done by comparing authoring tasks performed by groups who learned to author without using the integrated system to groups performing the same tasks with the ITS.

Introduction

Most activities related to engineering coursework performed outside the classroom are passive, including reading assignments, web searches and typical engineering problem sets. Passive learning environments provide few opportunities for students to discover their misconceptions. Without timely feedback, students are often completely unaware of mistakes they may have made or misconceptions they had while completing a paper and pencil assignment. Immediate formative assessment in online assignments can help students realize that they made a mistake or did not fully understand a particular concept, and allow them to get back on track as they progress toward the learning goals[1].

The VaNTH ERC has developed a learning technology infrastructure[2] that enables educators to create innovative online assignments which provide students with real time formative feedback as they attempt to solve quantitative engineering problems[3]. This infrastructure consists of a visual language-based authoring technology called CAPE (Courseware Authoring and Packaging Environment) and a web-based delivery platform called eLMS (experimental Learning

Management System). We have used CAPE to develop a generic template for constructing homework problem sets that are capable of providing diagnostics and feedback. This generic module must be coupled with data that is specific for each individual problem before it can be presented to students via eLMS.

Web-based Authoring Tool

To assist authors in the process of building online homework sets with diagnostics and feedback, we have developed a Web-based Authoring tool (http://cape.vanth.org/WebAuthoring). Authors interact with the tool to develop their homework sets and, consequently, do not need to know how to construct modules with CAPE. Design of an online problem is quite straightforward and involves repetitive application of the following steps:

- pose a problem, followed by a series of questions
- acquire student responses to the questions
- compare the responses to anticipated (diagnosed) responses
- provide feedback or remediation based on the responses

The assignment can be adaptive, with the next problem offered to the student being dependent on the response to the current problem. Incorrect responses can be followed with new questions that are designed to discover where the student made an error. Alternatively, the author might elect to ask the same question for some maximum number of iterations, providing additional diagnostic information after each attempt (i.e., progressive remediation). If the maximum number of attempts is exceeded, the author can elect to suspend further interactions until the student has seen the instructor or TA. Alternatively, the author can display the correct result(s), or simply move on to the next question.

Authors interact with the web-based authoring tool by entering all of the information necessary to build an assignment. Four separate web pages have been designed to assist the author in construction of an assignment: 1) *Assessment*, 2) *Feedback*, 3) *Data*, and 4) R*esources* (Figure 1). The tool comes with a built-in html editor, online help, a data viewer, and an online debugger.

Questions are composed in the Assessment view. The system supports 'fill in the blank' questions (string, essay, integer, floating point), 'multiple choice' questions, 'selections', and 'true/false' questions. The author can provide an overall problem statement, as well as any number of questions relating to the problem. Numerical values used in problem statements can be randomized, so every student is presented with a unique numerical problem.

Comparisons between the student's response and various diagnosed responses (including the correct response(s)) are specified in the Feedback view. The syntax used for the comparison follows the syntax of the Python computer language. Feedback messages for each of the diagnosed responses can be entered via the Feedback view. If multiple attempts are allowed, then a different feedback message can be provided for each unsuccessful response.

Data and computations needed for the problem solution are entered in the Data view. The Python language is supported, which includes the following rich set of data types: string, floating point, integer, boolean, derived, struct, array, DateTime, and function. Data can be either local (available only within the problem in which it was defined) or global (available for all problems

in the assignment module). Data can be referenced in assessment questions and feedback messages.

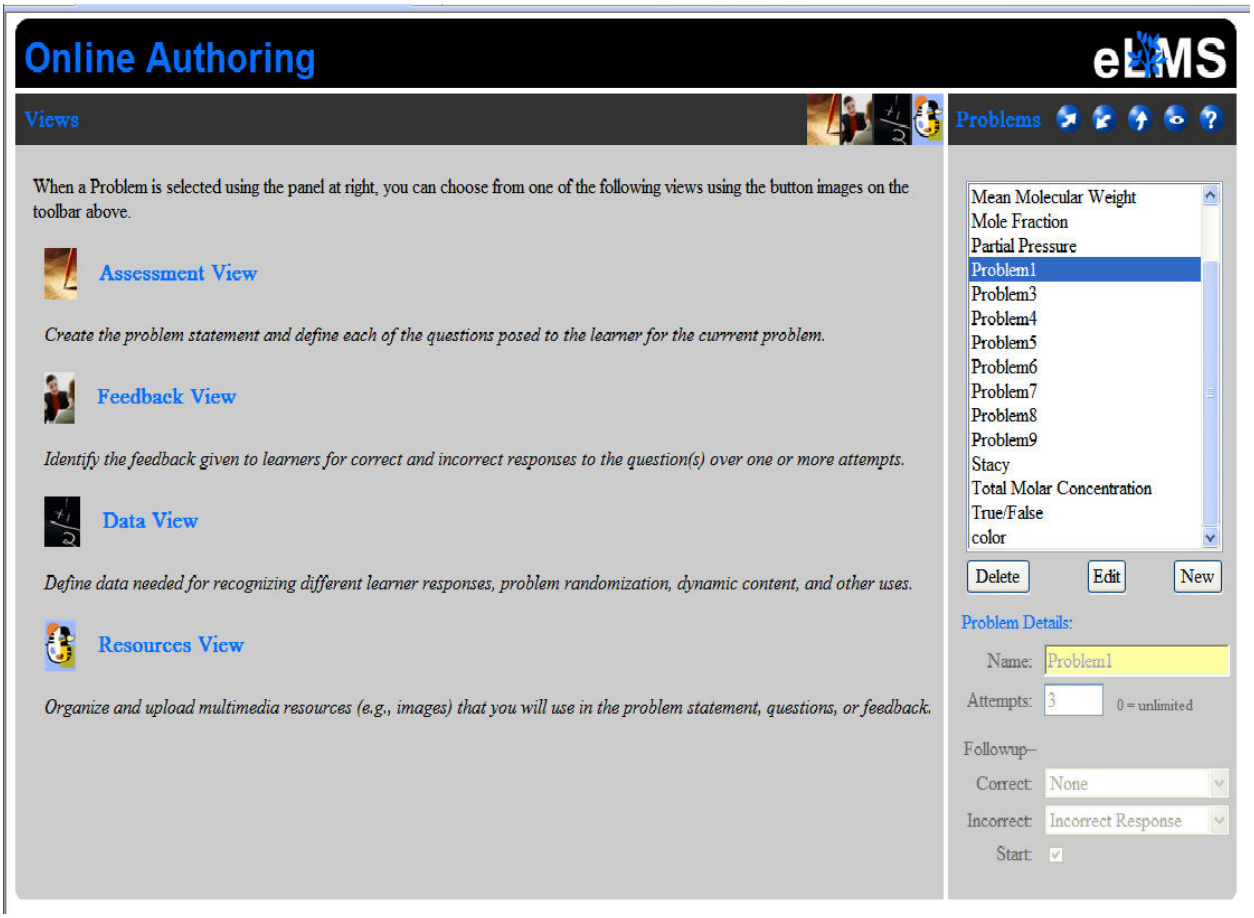

Figure 1: The opening screen of the web-based authoring tool.

Resources, such as pictures stored on your computer, can be imported and referenced in materials presented to students.

The author can test the module by stepping through it one question at a time, entering various responses to test the comparisons and the feedback. A debugger is available to assist with this process. After the module is tested, it can be uploaded to eLMS for delivery to students.

The system provides benefits to students, teaching assistants and instructors. Students no longer need to wait until their homework is graded and returned before receiving timely feedback. Delivery is also adaptive, so students who answer a problem correctly the first time move on, but students who have problems may go through additional iterations. Every interaction with the system is stored and can be reviewed later by the student or with the instructor to see where the student may have gone wrong. Teaching Assistants are freed from grading homework assignments, most of which are correct. Instead, TAs can focus their attention on helping those students who need help. The instructor can quickly evaluate performance by the entire class on each question to determine if the class as a whole is having difficulty with particular concepts. If so, the instructor can review those concepts immediately, rather than waiting for the next exam

or a communication from the grader. It is easy to post assignments, import grades, and enforce submission deadlines. Randomization of input variables for each student makes the assignment of the same good problems possible from year to year. The preparation time for homework is greatly reduced after the initial year or two. Homework problems developed by others at different universities can be shared and modified via the VaNTH portal. Examples of two biomechanics assignments and feedback for one of the diagnosed incorrect responses for each are shown in Figure 2.

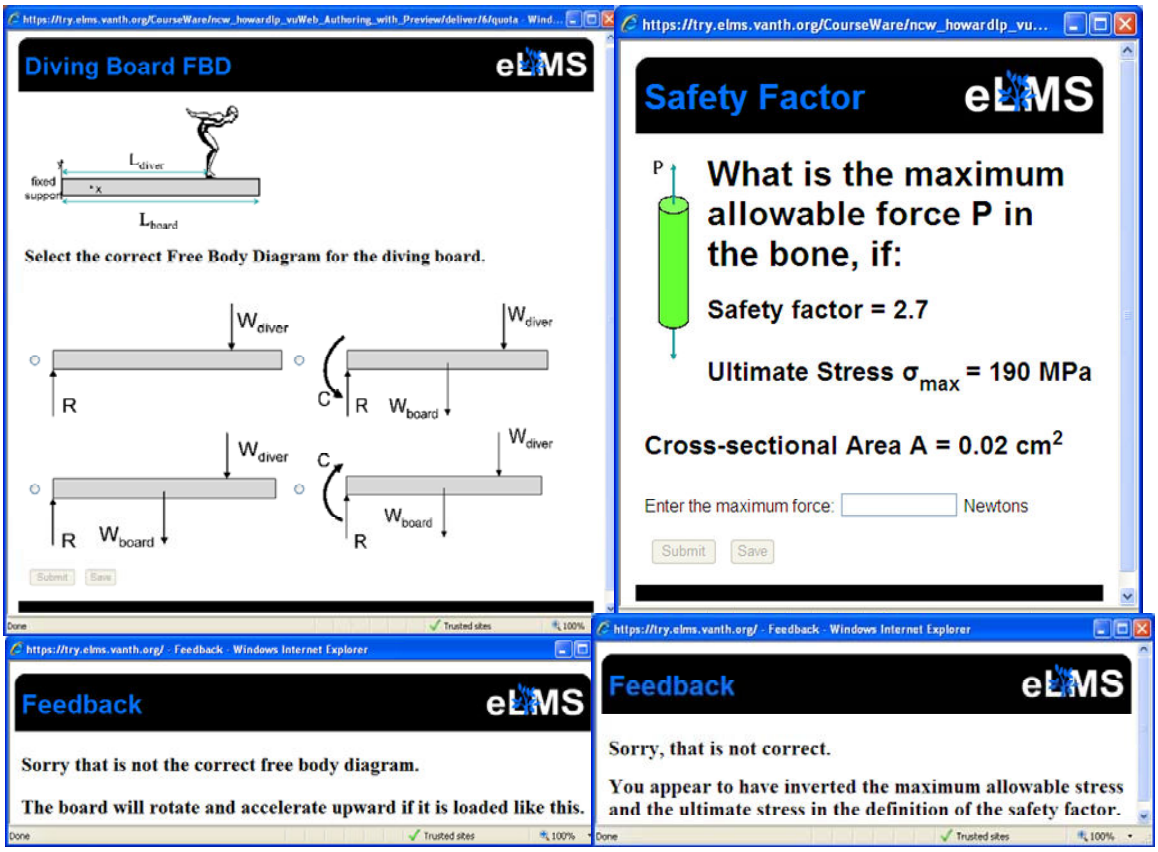

Figure 2. Examples of questions and feedback for diagnosed incorrect responses. Left: Feedback for selection of the top left free body diagram. Right: Feedback for a force calculation in which the student has inverted terms in the definition of the safety factor.

Despite its many advantages, novice developers often find it relatively difficult to design, construct and debug adaptive learning modules with the web-based authoring tool. The goals of this work are to develop an Intelligent Tutoring System (ITS) that can be integrated with the authoring tool to provide personal guidance to new users as they develop homework problems, and to design workshops for training faculty interested in constructing their own online assignments with diagnostics and feedback.

Intelligent Tutoring Systems

An intelligent tutoring system (ITS) observes a learner's behavior while working through a tutorial on a computer system. At any time, the learner can request hints about what to do next, or the tutor may offer friendly error messages if the learner takes incorrect steps in the software. A model-tracing ITS is based on a cognitive model, which contains information about the learning domain and about common mistakes that learners make.

ITSs have been successfully used to tutor on a variety of domains such as mathematics, geometry, and economics. The benefits of ITSs include personalized training, learning with real-world tasks, and multiple levels of help[4-9]. Studies have shown that students who use an ITS to learn can master the material in a third less time[10].

The ITS was developed by Clearsighted, Inc., which develops ITSs for software training. Clearsighted partners with Carnegie Learning, Inc., the leader in creating and deploying cognitive tutors, a particular form of ITS, in mathematics classrooms. The ITS approach used with the VaNTH web-based authoring tool is similar to an approach Clearsighted used to develop an ITS for learning Paint.NET, an open-source image manipulation application[11].

Building an ITS for the Web-Based Authoring Tool

The prototype ITS provided assistance for a single 'true/false' problem, the Color of Royalty problem. We developed a complete cognitive model for this problem with goalnodes mapped to each meaningful action that the learner can take towards the goal of creating the problem successfully. We provided the learner with a problem statement, and the learner can proceed with the problem uninterrupted if no help is needed. If the learner takes a step that would lead away from the solution, a just-in-time message (JIT) appears. A partial screenshot of the prototype is shown in Figure 3.

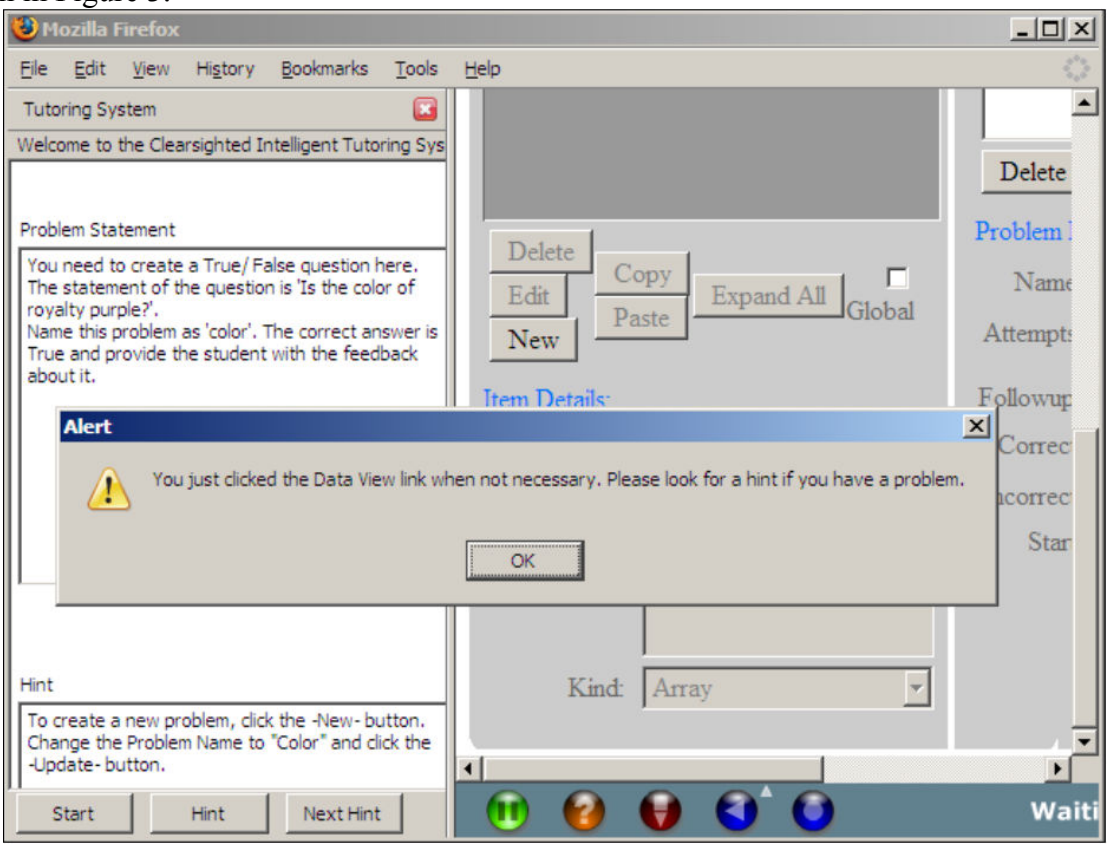

Figure 3: A screenshot of the ITS prototype for the Web-based Authoring Tool. The tutor on the left side panel shows the initial problem statement to address and a hint. A JIT (just-in-time message) gives feedback on top of a partial screenshot of the Web-based Authoring Tool on the right.

The structure of two of the problem's goalnodes are shown in Figure 4. The first row is the first step of the problem. If the learner clicks a different "New" button, e.g., the "New" button to create data items in the Data view, a JIT appears explaining the error and suggesting what to do differently. It also provides an image of the correct 'New' button for this action. The learner can also request hints associated with each step by clicking the Hint button in the tutor side panel. The bulleted phrases in the hints are successful feedback that the learner sees upon requesting further help.

ITS problems (including this prototype) contain an internal sequence of steps that lead to the solution. With an ITS (unlike linear tutorials such as videos), the learner can often choose between multiple paths along the way. For example, in Assessment View, the user has a choice to fill in the 'Title of the Problem' or the 'Problem Statement' first, before proceeding with the other.

ITS Design Decisions: Instructional Approach

The instructional design of an ITS is based on principles summarized by Bransford et al.[12] that emphasize 'learning by doing.' The tutorial poses a real-world problem to be solved using the actual software itself rather than a simulation. However, the designer must make several decisions about the extent to which the ITS takes control of the learner's experience with the software. Should the ITS actually block features, so that beginning users do not get confused by complicated interface elements? How much previous knowledge about computing software should the ITS assume?

Goalnodes	Steps	Hints	JITs
Click-New	Create a new problem with name as color	• You need to create a new problem first. Use "Color" as your Problem Name. • To create a new problem, click the -New- button. Change the Problem Name to "Color" and click the -Update- button.	You have clicked the wrong New button. The correct button is in the Problems panel. 
Text-Assess-Title	Provide the title for the problem	• You are creating the problem. Provide its title, description, create a question and add details • Use the following information. Title - "Color of Royalty", Problem Statement - "No problem statement is necessary here.", Question - "Q1", Question type - "True/ False problem." and Question Statement - "The color of royalty is purple."	
Link-Assess	Move to the Assessment view	• You need to create the question for your problem	

Figure 4: Two steps of the Color of Royalty problem, along with feedback that the student would receive upon request (hints) or upon error (JITs). Successive bulleted hint phrases appear to the learner upon request for further help.

Our ITS design follows several basic principles: 1) Don't remove features of the software (graying out features, etc), 2) Interrupt the learner as little as possible, 3) Give feedback in many small doses, so that an expert can be satisfied with a little but a beginner can find more detail, and 4) Only stop the learner if he or she is about to take a step that would lead down an irreparable path (e.g., deleting a key element). These principles still leave several options for reacting to errors, however. For example, the ITS could block a step and say, "Sorry! This step would lead to…." That approach is highly invasive, however, and a preferable approach when possible is to allow the step but give the message, "Note that you have just… A better way is…" This method allows the learner to fail, a key element in learning[13], but informs why it's incorrect. One might suggest that more complex failure would offer a better learning experience in the long run. Research from ACT-based ITSs suggest that it's better to have the tutor intervene much sooner rather than later, and not have the student explore longer fruitless paths[14].

The ITS follows the above principle of minimal interruption by offering hints only when requested. With this approach, a learner who knows what to do can complete the tutorial quickly with no tutor interaction. The system follows the principle of feedback in small doses by making all feedback dependent on the context. The hints are not simply, "Do this now;" they first remind the learner of his or her current subgoal, "Here's what you need to accomplish next." Then, if the learner requests further hint information, it will elaborate to explain how to accomplish that subgoal. Similarly, JIT messages, which occur upon an error, do not say simply, "That's wrong." They explain what's wrong and why.

ITS Design Decisions: Technical Architecture

Enabling students to be tutored from within any application, even one that was designed without thought of an ITS, has been investigated by us and our colleagues in past work[15,16]. The web-based authoring tool ITS implements a system shown in Figure 5 and described in Blessing, Gilbert, Ourada, and Ritter[17].

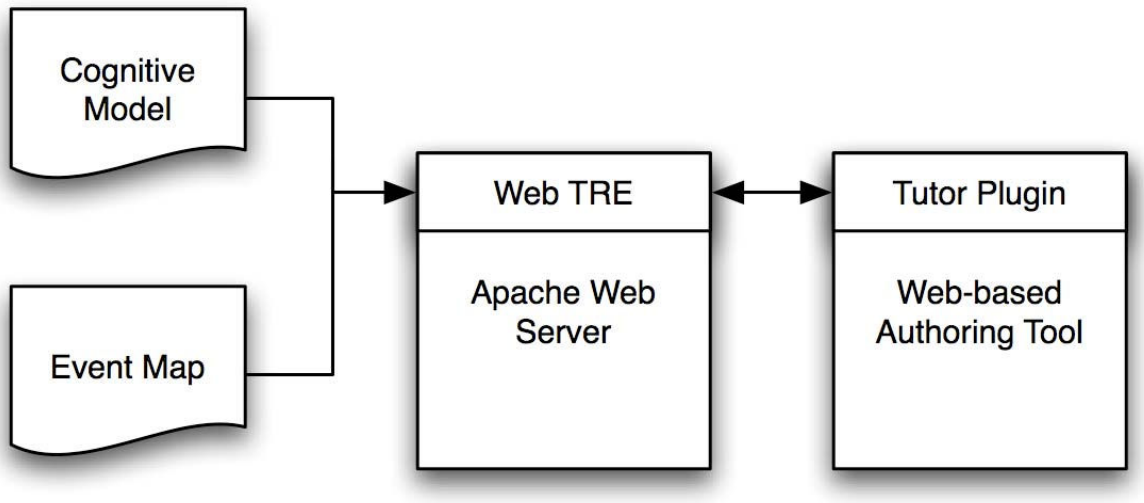

Figure 5: Architecture of the ITS-enhanced Web-based Authoring Tool

The cognitive model includes information describing the objects within the learning domain and production rules that determine which feedback the student will receive at any given moment. Every interface element of the web-based authoring tool for which we need learning instruction is mapped to an object and has one or more rules associated with it. This association between the tool's website and cognitive model is done with event mapper files. The model and the mapper files are provided as input to the Web Tutor Runtime Engine (WebTRE). The TRE is run as a part of an Apache web server. The Tutor Plugin is installed as a plugin for the Firefox browser. The WebTRE web server communicates with the Tutor Plugin via TCP/ IP.

An example of the communications follows. The Tutor Plugin inspects the Document Object Model (DOM) of the webpage in the Firefox window, where the web-based authoring tool is run. When the user clicks on a button in the authoring tool, this action is tracked and sent across to the WebTRE web server. The TRE, with the help of the Event Mapper files, checks whether any feedback is associated with the action. If there is feedback, such as a hint or an error message, it sends that message back to the plugin, which displays it in a sidebar next to the tool.

The current ITS architecture makes the assumption that learners are using the Firefox web browser so that they can use the Tutor plugin. Although other browsers support plugins, the APIs are not standardized at this point. While the core code of the plugin could be shared across different kinds of web browsers, each plugin would require a different packaging code because of its reliance on the DOM model of web pages. More generally, the WebTRE approach is limited to web pages that do not use Flash and AJAX extensively. The more complex the webpage becomes, the more difficult it is for the tutor to monitor the learner's behavior.

Working with Users

VaNTH offered a number of workshops in 2007 to approximately 70 college and university instructors. A full day was directed at the use of CAPE, and the web-based authoring tool (without the ITS) was demonstrated. Feedback from these workshops indicated that many potential developers would be interested in using the web-based tool to construct homework assignments with diagnostics and feedback. A set of tutorials was developed and a three hour training workshop on the web authoring tool was offered to faculty in the School of Engineering in November, 2007. There were many positive comments about the capabilities of the modules produced by the web authoring tool. However, the time spent by participants on most tasks was 2-3 times longer than anticipated. Criticism focused on procedural issues encountered while building the module, difficulties in interpreting some of the error messages, and difficulties with syntax when making comparisons between diagnosed responses and student responses. A new help file and hints provided by the ITS have been designed to clarify each of these issues. For the syntax issues raised, the ITS addresses a whole range of common errors such as case sensitiveness of Python, usage of '=' instead of '==' for comparison, and unnecessary blank spaces in the comparison expression. For example, if an author needs to type the Python formula:

```
int(4.*pi*pow(R,3)/3.+.5)
```

but the author changes the order of operations and neglects some decimal points and types

```
int(pi*4*pow(R,3)/3+.5)
```

the ITS can recognize that type of mistake and offer appropriate feedback. Our goal is to reduce the feelings of intimidation that non-computational authors may experience by providing ITS-based feedback that can walk them through the construction of an appropriate Python expression.

A one-day workshop has been designed to teach up to 25 potential authors how to develop homework problems with diagnostics and feedback using the ITS-enhanced web-based authoring tool. This workshop is scheduled to take place on June 22, 2008 at the University of Pittsburgh. It is based on successful workshops delivered by VaNTH in the past. Participants will be asked to bring with them one or more problems they would like to assign to their students.

The initial presentation focuses on the importance of formative assessment, provides many examples of its use in online homework assignments, and compares student responses following diagnosed errors to student responses following undiagnosed errors. The second presentation focuses on the web authoring tool and how to use it. This is followed by a demonstration of a working module. Attendees are then led through a simple True/False question that introduces them to the intelligent tutoring system and to the basic procedures necessary for constructing a problem with diagnostics and feedback. Since the workshop is open to educators from all scientific and engineering disciplines, the tutorial problems are quite simple in nature, and are not specific to biomedical engineering.

After initial training, workshop participants will be provided with the two sets of tutorials used in the previous workshop. Written instructions will be much less specific than in the original workshop, and participants will be expected to rely on the ITS for assistance. These tutorials are designed to familiarize participants with key aspects of the web authoring tool, diagnostics design, and the ability to thread problems so the assignment is adaptive to student responses. After completing these tutorials, participants should be prepared to design their own modules. However, before we open the workshop to the development of modules for their own classes, all participants will be asked to design a module with a given set of specifications, but without the use of the ITS. For example:

> "Design and test a problem to illustrate Newton's second law, F = ma, where the mass is given in grams, the acceleration in cm s^{-2} and the force is requested in Newtons. Specifically, diagnose the case where the student fails to express the force in Newtons (1 N = 10^5 g cm s^{-2})."

We anticipate that workshop participants will have approximately 1.5 – 2 hours to develop their own problems at the end of the day.

Evaluation of the ITS

Effectiveness of the workshop will be evaluated at several different levels. We will compare performance by participants at this workshop with the performance on the same problems developed by participants at a previous workshop without use of the ITS-enhanced web authoring tool. Comparisons will be made for the two instrumented tutorials on the basis of time to completion for various tasks, accuracy and completeness of the modules developed, and responses to a survey developed for the previous workshop, which included Likert scale questions and free responses.

Results from the baseline workshop were quite diverse. Of the eight participants, only two finished authoring both of the tutorial problems in the allotted 2 hour period. Two others finished the first tutorial problem, then began to author their own problems, rather than authoring on the second tutorial problem. The others did not finish authoring all portions of the first tutorial problem. The average time spent by participants at the workshop was 106 ± 37 min (range: 56 – 131 min). All participants finished authoring and testing the initial portion of

problem 1 in 37 ± 13 minutes (range: 13 -55 min), which represented a complete problem. The remainder of problem one consisted of a number of extensions to the problem. The four participants who completed all of the extensions to problem 1 did so in 82 ± 19 min. The other four were still authoring extensions to the first problem at the end of the session.

On the post-workshop survey, participants found the the tool relatively difficult to use (score of 2 on a Likert scale which ranged from difficult [1] to easy [7]), but were relatively confident that they could use the web authoring tool to author their own assignments (4.33/7). Criticism of the tool included comments that too many clicks were required, using the proper syntax is important, buttons should be relocated on the screen to prevent repetitive scrolling, and authors were unfamiliar with the Python language. Comments about the workshop were generally favorable, but one participant suggested that the workshop might have been improved if participants were allowed to author their own problems, rather than confining instruction to the tutorial problems.

Based on these comments, we are making improvements to the authoring tool interface that should reduce the number of clicks, minimize repetitive scrolling and assist authors with use of the Python language. The ITS will help authors with syntax and will assist them in determining what needs to be done next.

User logs at the Pittsburgh workshop will be analyzed to determine the amount of time participants spent in the assessment, feedback, data and resources views, how much time was spent testing and debugging their module, and the number and nature of errors detected by the authoring tool and the ITS (JITs). Comparisons will also be made between the highly structured tutorial problems with integrated ITS and the more open-ended problem that is to be authored without the aid of the ITS. New survey questions directed at analyzing and improving the ITS will be added to the post-workshop survey.

Workshop results will be used to help improve both the authoring tool and the ITS. Additional hints and JITs will be developed for tasks in which user errors are detected by the authoring tool. Excessive JITs can be used to design modifications to the authoring interface that can prevent certain classes of errors.

Acknowledgement.

This work was supported primarily by the Engineering Research Center Program of the National Science Foundation under Award Number EEC9876363.

Bibliography

1. Black, P, and Dylan, W. (1998). Assessment and classroom learning. *Assessment in Education*: 5:7-74.
2. Howard L.P. (2003). Adaptive learning technologies for bioengineering education. Engineering in Medicine and Biology Magazine 22: 58-65.
3. Roselli RJ, LP Howard and SP Brophy (2006). Integration of formative assessment into online engineering assignments, *Computers in Engineering Journal*, *14(4):281-290*.
4. Aleven, V. et al. (2006). Toward meta-cognitive Tutoring: A Model of Help-Seeking with a Cognitive Tutor. *International Journal of Artificial Intelligence in Education, 16*, 101-130.

5. Arruarte, A. et al. (1997). The IRIS Shell:"How to Build ITSs from Pedagogical and Design Requisites. *International Journal of Artificial Intelligence in Education, 8*, 341-381.

6. 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.

7. Koedinger, K. R., Aleven, V., Heffernan, N., McLaren, B. M., & Hockenberry, M. (2004). Opening the Door to Non-Programmers: Authoring Intelligent Tutor Behavior by Demonstration. In the *Proceedings of the Seventh International Conference on Intelligent Tutoring Systems*.

8. Graesser, A.C., Lu, S., Jackson, G.T., Mitchell, H., Ventura, M., Olney, A., & Louwerse, M.M. (2004). AutoTutor: A tutor with dialogue in natural lan-guage. *Behavioral Research Methods, Instruments, and Computers, 36*, 180-193.

9. VanLehn, K., Lynch, C., Schulze, K., Shapiro, J.A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., & Wintersgill, M. (2005). The Andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence and Education, 15(3).*

10. Corbett, A.T. (2001). Cognitive computer tutors: Solving the two-sigma problem. In the *Proceedings of the Eighth International Conference of User Modelling*.

11. Hategekimana, C., Gilbert, S., Blessing, S. (2008, in press) Effectiveness of using an intelligent tutoring system to train users on off-the-shelf software. In the *Proceedings 19th Annual Conference of the Society for Information Technology & Teacher Education.*

12. Bransford, J.D., Brown, A. L. Brown, and Cocking, R. R. (2000) *How People Learn: Brain, Mind, Experience and School*. National Academies Press; 1st edition.

13. Schank, Roger (2002). *Designing world-class e-learning*. New York: McGraw-Hill.

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

15. Ritter, S. & Koedinger, K. R. (1997). An architecture for plug-in tutoring agents. In *Journal of Artificial Intelligence in Education, 7 (3/4)*, 315-347. Charlottesville, VA: Association for the Advancement of Computing in Education.

16. Ritter, S., Blessing, S. B., & Wheeler, L. (2003). Authoring tools for component-based learning environments. In T. Murray, S. Blessing, & S. Ainsworth (Eds.), *Authoring Tools for Advanced Technology Educational Software*, Dordrecht, The Netherlands: Kluwer Academic Publishers.

17. Blessing, S., Gilbert, S., Ourada, S., & Ritter, S. (2007). Lowering the bar for creating model-tracing intelligent tutoring systems. In *Proceedings of the 13th International Conference on Artificial Intelligence in Education* (pp. 443-450), Marina del Rey, CA. Amsterdam, Netherlands: IOS Press.