The Extensible Problem-Specific Tutor (xPST): Evaluation of an API for Tutoring on Existing Interfaces

Stephen GILBERT[a,1], Stephen B. BLESSING[b], Sateesh KODAVALI[a]
[a]*Virtual Reality Applications Center, Iowa State University*
[b]*University of Tampa*

Abstract. The Extensible Problem-Specific Tutor (xPST) system was created to facilitate the creation of intelligent tutoring systems (ITSs) on existing software and web-based interfaces. This research introduces xPST and describes an exploration of the ease of authoring an xPST tutor for three web-based tasks. Participants demonstrated a quick learning curve and alternated their attentional focus regularly among the three components of the xPST authoring system.

Keywords. Intelligent tutoring system, authoring, web-based tutor, plug-in

Introduction

We address two challenges in the field of ITSs: 1) building tutors for pre-existing and web interfaces and 2) enabling easy authoring of such tutors. In particular, we describe xPST, an open-source architecture (http://code.google.com/p/xpst) for building tutors for existing interfaces and a study evaluating the xPST authoring process by graduate students with no experience with intelligent tutoring.

1. xPST: The Extensible Problem-Specific Tutor

Re-using an existing interface with a tutor reduces the time required to develop the tutor and any issues of learning transfer, a concern of past researchers [1]. If the ITS environment is the same as the non-ITS environment then such issues of transfer largely disappear. We developed the Extensible Problem-Specific Tutor (xPST) in order to perform software training within the software itself. Traditional software training often uses videos based on screen-recordings, but this passive technique to learning has been shown to be less effective than an ITS [2]. xPST is similar to the Cognitive Tutor Authoring Tool (CTAT; [3]), but allows for easier integration into existing software.

The xPST architecture is an instantiation of the architecture of plug-in tutor agents described in [4]. The xPST file, which contains information that allows for instruction akin to a model-tracing tutor, includes information describing the objects within the

[1] Corresponding Author: Stephen Gilbert, Virtual Reality Applications Center, Iowa State University, 1620 Howe Hall, Ames, IA 50011, USA Email: gilbert@iastate.edu

learning domain and rules that determine which feedback the student will receive. Every interface element of the application for which one needs learning instruction is mapped to an object and has one or more rules associated with it. The rules contain the instructional feedback. The Listener plugin eavesdrops on user actions in the third-party software and sends them to the xPST Tutoring Engine, which checks them with the xPST file. Feedback is mapped back to the client UI control and displayed *in situ*. Because software training does not require as repetitive of tasks as seen in many academic tutors, such problem specific models are simpler to produce and manage.

The xPST file is a text file written using a syntax that is designed to be easy to read and write for an inexperienced cognitive modeler. The file contains three sections: Sequence, Mappings, and Feedback. The sequence identifies a path of steps the user takes through the problem space to achieve the goal specified in the task. The mappings section maps interface identifiers to the steps noted in the sequence that the user takes. Finally, the feedback section provides hints and error messages for each step within the sequence. Because of this relatively simple syntax, the authoring tool for xPST can be a text editor. We have built an online editor for creating web-based tutors where authors can immediately jump to their target website and test the current xPST file.

We have confirmed that the xPST approach can be used to develop real tutors rapidly; our most extensive effort is described in [5], in which the tutor taught university faculty how to use a complex web-based homework authoring tool. Other efforts include tutoring on the NCBI biotechnology site and on Slashdot Journals.

2. The Study

We conducted a study examining the xPST authoring tool during Fall 2008 in an introductory HCI graduate class. Eighteen students, all non-programmers, attempted an extra credit assignment, and 10 agreed to allow their data into the study.

The assignment contained three parts: 1) information about ITSs in general; 2) a worked example of using the xPST authoring system; and 3) the assignment. The first two parts were composed of text documents, screen movies, and an annotated example. We estimated an average student would take one hour to understand the material. The assignment consisted of designing an xPST tutor for three different tasks, all centered on the issue of searching a particular library database. In doing the assignment, students had available to them the on-line xPST authoring tool, modified slightly for logging purposes. Students had three weeks to do the assignment. All materials were available via a webpage, and students were free to work on the assignment at any time or place until its due date. The materials suggested 5 - 10 hours to do the assignment.

The 10 participants produced 26 models: 1 participant produced only 1 task model, 2 produced 2 task models, and the other 7 participants completed all 3 task models. We rated the models on a 5-point qualitative scale based on actually running the model. A majority of models (18 out of 26, 69%) received a score of either 4 or 5 (more than sufficient for a learner to do the task, in terms of help and JIT messages). One model scored a 2, and the remaining seven scored a 3.

The authoring tool kept track of how long participants spent working on the three parts of an xPST model. Table 1 displays the average time participants spent on various parts of the models, split by the three different tasks. The times decrease as the participants move from task to task. Though not required to work on tasks in order, all participants did so. Participants got more noticeably proficient in using the tool and

creating these models. The average participant spent 9.37 hr producing all three models. The quickest time was 2.58 hr for all three models, and the slowest was 20.97 hr (a non-native English speaker who took 13.55 hr on the first model alone).

Table 1. Time to complete model actions within a task (times in hr, with percent of total in parentheses)

	Task A	Task B	Task C
Sequence	1.30 (25.4%)	1.02 (40.3%)	0.58 (33.5%)
Mappings	1.63 (31.9%)	0.58 (22.9%)	0.40 (23.1%)
Feedback	2.18 (42.7%)	0.93 (36.8%)	0.75 (43.4%)
Total	5.11	2.53	1.73

3. Discussion

We would like to note two items from the study. First, people were able to use the xPST authoring tool. None of the participants considered themselves programmers. No one who attempted to start the task failed to produce at least one working model, with minimal instruction. Second, there was a spread in terms of time to create the models, but all users reduced their times upon successive models. By the third model, participants finished in about 1.5 hrs, with more improvement to come. That time provided 10 min worth of instruction. The small up-front cost of training coupled with a small time in producing the training could make the xPST approach attractive to those who want to provide model-tracing feedback to existing interfaces.

To conclude, we have discussed the xPST tutoring system and authoring tool. An experiment has demonstrated the tool's ease of learning, and we have produced xPST tutors in a variety of domains. Additional work though is needed to expand the contexts in which xPST can be used and to make it a more user-friendly authoring environment.

Acknowledgements

We thank Steven Ourada as the senior architect of xPST, and Mike Oren for assisting in running the experiment. This work was supported in part by the National Science Foundation under OII-0548754 and by the Air Force Office of Scientific Research.

References

[1] Corbett, A. T., Koedinger, K. R., & Anderson, J. R. Intelligent tutoring systems. M. G. Helander, T. K. Landauer, & P. Prabhu, (Eds.) *Handbook of Human-Computer Interaction*, 2nd edition Elsevier Science, (1997), 849-874.

[2] Hategekimana, C., Gilbert, S. & Blessing, S. Effectiveness of using an intelligent tutoring system to train users on off-the-shelf software. In K. McFerrin et al. (Eds.), *Proc. of Society for Info. Tech. and Teacher Education Int'l Conf., AACE* (2008), 414-419.

[3] Aleven, V., Sewall, J., McLaren, B. M., & Koedinger, K. R. Rapid authoring of intelligent tutors for real-world and experimental use. In Kinshuk, R. Koper, P. Kommers, P. Kirschner, D. G. Sampson, & W. Didderen (Eds.), *Proceedings of the 6th IEEE International Conference on Advanced Learning Technologies* (ICALT 2006), 847-851.

[4] Ritter, S., & Koedinger, K. (1996) An architecture for plug-in tutor agents, *Journal of AIED*, 7(3-4), (1992), 315-347.

[5] Roselli, R.J., Gilbert, S., Howard, L., Blessing, S. B., Raut, A., & Pandian, P. Integration of an Intelligent Tutoring System with a Web-based Authoring System to Develop Online Homework Assignments with Formative Feedback. American Society for Engineering Education 2008 Conference.