The Assistment Builder: A Rapid Development Tool for ITS

Terence E. TURNER, Michael A. MACASEK, Goss NUZZO-JONES, Neil T. HEFFERNAN
Worcester Polytechnic Institute
100 Institute Rd, Worcester, MA 01609
(508) 831-5569
tinylox@wpi.edu, macasek@wpi.edu, goss@wpi.edu, nth@wpi.edu
Ken KOEDINGER
Human-Computer Interaction Institute
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract. Intelligent Tutoring Systems are notoriously costly to construct [1], and require PhD level experience in cognitive science and rule based programming. The goal of this research was to ease the development process for building pseudo-tutors [4], which are ITS constructs that mimic cognitive tutors but are limited in that they only work for a single problem. The Assistment Builder is a system designed to rapidly develop, test, and deploy simple pseudo-tutors. These tutors provide a simple cognitive model based upon a state graph tailored to a specific problem. These tutors offer many of the features of rule-based tutors, but without the expensive creation time. The system simplifies the process of tutor construction to allow users with little or no ITS experience to develop content. The system provides a web-based interface as a means to build and store these simple tutors we have called Assistments. This paper describes our attempt to make the process of developing content easy for teachers. We present some evidence to suggest that these novice users can develop a tutor for a problem in under thirty minutes.

1. Introduction

This research aims to develop tools for the rapid development and deployment of Intelligent Tutoring Systems (ITS). Specifically, this research focused on so-called "pseudo-tutors" that are a simplification of cognitive rule-based tutors 4]. Model tracing rule-based tutors [1] have been shown to be effective [5], but development time on them is highly prohibitive, from 100-1000 hours of development time per hour of content [6][1]. Development also requires a very specialized knowledge set. Tutor developers are required to be expert system programmers, in addition to developing the cognitive model, to say nothing of being a content expert. Another aim of this research was to make our tools accessible to novices, with no programming experience, and less than an hour of training.

A pseudo-tutor is a simplified cognitive model based on a state graph. State graphs are finite graphs with each arc representing a student action, and each node representing a state of the problem interface [2][8]. Student actions trigger transitions in the graph, and the current state of the problem is stored by the graph. Pseudo-tutors have nearly identical behavior to a rule-based tutor, but suffer from having no ability to generalize to different problems [3]. This pseudo-tutor approach allows for predicted behaviors and provides feedback based on those behaviors. We also combined this state graph with a conceptually broader branching structure referred to as scaffolding. Scaffolding provides sub-problems to the initial question, often designed to address specific concepts within the initial question. Both initial and scaffold questions can branch to different scaffolding questions depending on a student's actions. This allows for a higher-level of predicted actions to be handled.

1.1 Overview of the Assistments Project

These pseudo-tutors are being developed and deployed as part of the Assistments Project [7]. The Assistments architecture is an interactive content delivery system designed to deploy both pseudo-tutors and full cognitive model tutors over the web with centralized database logging of student actions. A problem consists of an interface definition and behavior definition. The interface definition provides a collection of simple widgets to be displayed to the student. The behavior definition is a representation of the state graph and its transitions, or a cognitive model. Many types of behaviors are possible within the representation and architecture. These two parts of the representation are consumed by the runtime Assistment architecture, and presented to the student over the web. Student actions are then fed back to the representation, and compared with the state graph or used to model trace.

1.2 Purpose of the Assistment Builder

We sought to create a tool that would provide a simple web-based interface for creating these pseudo-tutors. Upon content creation, we could rapidly deploy the tutor across the web, and if errors were found with the tutor, bug-fixing or correction would be quick and simple. Finally, the tool had to be usable by someone with no programming experience and no ITS background. This applied directly to our project of creating tutors for the mathematics section of the Massachusetts Department of Education (MCAS) test [7]. We wanted the teachers in the public school system to be able to build pseudo-tutors. These pseudo-tutors are often referred to as *Assistments*, but the term is not limited to pseudo-tutors.

A secondary purpose of the Assistment Builder was to aid the construction of a Transfer Model. A Transfer Model is a cognitive model construct divorced from specific tutors. The Transfer Model is a directed graph of *knowledge components* representing specific concepts that a student could learn. These *knowledge components* are then associated with a specific tutor (or even sub-question within that tutor) so that the tutor is associated with a number of *knowledge component* arcs associated with it. This allows us to maintain a complex cognitive model of the student without necessarily involving a production rule system. The basic structure of an *Assistment* is a top-level question that can then branch to scaffolding questions based on student input

The scaffolding questions mentioned above are all queued as soon as a user gets the top-level question incorrect, or requests help in the form of a hint. Upon successfully completing the displayed scaffolding question the next is displayed until the queue is empty. Many *Assistment* authors also use text feedback on certain incorrect answers. These feedback messages are called bug messages. Bug messages address the specific error made, and match common or expected mistakes.

Content creators can also use the Assistment Builder to add hint messages to problems, providing the student with hints attached to a specific scaffolding question. This combination of hints, buggy messages, and branched scaffolding questions allow even the simple state diagrams described above to assume a useful complexity.

We constructed the Assistment Builder as a web application for accessibility and ease of use purposes, a teacher or content creator can create, test, and deploy an *Assistment* without installing any additional software. A user can design and test his *Assistment* and then instantly deploy it. By making the *Assistment* Builder available over the web, if a new feature is added users do not need to update any software.

2 Methods

To analyze the effectiveness of the Assistment Builder, we developed a system to log the actions of an author. While authors have been constructing items for nearly six months, only very recently has the Assistment Builder had the capability to log actions.

Each action is recorded with associated meta-data, including author, timestamps, the specific series of problems being worked on, and data specific to each action. These actions were recorded for a number of Assistment authors over several days. The authors were asked to build original items and keep track of roughly how much time spent on each item for corroboration. The authors were also asked to create "morphs," a term used to indicate a new problem that had a very similar setup to an existing problem. "Morphs" are usually constructed by loading the existing problem into the Assistment Builder, altering it, and saving it with a different name. This allows rapid content development for testing transfer between problems. We wanted to compare the development time for original items to that of "morphs" [7].

Another trial of the Assistment Builder with less rigorous methodology was testing how authors with little experience would react to the software. To test the usability of the Assistment Builder, we were able to provide the software to two high-school teachers in the Worcester, Massachusetts area. These teachers were computer literate, but had no previous experience with intelligent tutoring systems, or creating mathematics educational software. Our tutorial consisted of demonstrating the creation of a problem using the Assistment Builder, then allowing the teacher to create their own with an experienced observer to answer questions. Finally, we hope to allow them to author *Assistments* on their own, without assistance.

3 Results & Analysis

Prior to the implementation of logging within the Assistment Builder, we obtained encouraging anecdotal results of the software's use. A high-school mathematics teacher was able to create 15 items and morph each one, resulting in 30 *Assistments* over several months. Her training consisted of approximately four hours spread over two days in which she created 5 original *Assistments* under supervision. While there is unfortunately no log data to strengthen this result, it is nonetheless encouraging.

The logging data obtained suggests that the average time to build an entirely new *Assistment* is approximately 25 minutes. Entirely new *Assistments* are those that are built using new content and not based on existing material. This data was acquired by examining the time that elapsed between the initialization of a new problem and the problem save time. Creation times for *Assistments* with more scaffolds naturally took longer than those with fewer scaffolds. Experience with the system also decreases *Assistment* creation time, as end-users who are more comfortable with the Assistment Builder are able work faster. Nonetheless, even users who were just learning the system were able to create *Assistments* in reasonable time. For instance, Users 2, 3, and 4 (see Table 1) provide examples of end-users who have little experience using the Assistment Builder. In fact, some of them are using the system for the first time in the examples provided.

Table 1: Full Item Creation

Username	Number of Scaffolds	Time Elapsed (min)
User 1	10	35
User 1	2	23
User 2	3	45
User 2	2	31
User 2	0	8
User 3	2	21
User 4	3	37
User 4	0	15
User 5	4	30
User 5	2	8
User 5	4	13
User 5	4	35
User 5	3	31
User 5	2	24
		Average: 25.4 minutes

We were also able to collect useful data on morph creation time and Assistment editing time. On average morphing an *Assistment* takes approximately 10-20 minutes depending on the number of scaffolds in an Assistment and the nature of the morph. More complex Assistment morphs require more time because larger parts of an *Assistment* must be changed. Editing tasks usually involve minor changes to an *Assistments* wording or interface. These usually take less than a minute to locate and fix.

4 Conclusions

The Assistment Builder has been in use over six months by a variety of users involved in the Assistments project. Teachers, developers, and others have used it to develop pseudo-tutor *Assistments.* The end result has been over a thousand individual pseudo-tutors deployed on the web. The breadth of users who developed these *Assistments* and the number created would not have been possible without the Assistment Builder.

References

1. Anderson, J. R. (1993). Rules of the mind. Hillsdale, NJ: Erlbaum.
2. Jackson, G.T., Person, N.K., and Graesser, A.C. (2004) Adaptive Tutorial Dialogue in AutoTutor. *Proceedings of the workshop on Dialog-based Intelligent Tutoring Systems at the 7th International conference on Intelligent Tutoring Systems. Universidade Federal de Alagoas, Brazil, 9-13*.
3. Jarvis, M., Nuzzo-Jones, G. & Heffernan. N. T. (2004) Applying Machine Learning Techniques to Rule Generation in Intelligent Tutoring Systems. Proceedings of 7th Annual Intelligent Tutoring Systems Conference, Maceio, Brazil. Pages 541-553
4. Koedinger, K. R., Aleven, V., Heffernan. T., McLaren, B. & Hockenberry, M. (2004) Opening the Door to Non-Programmers: Authoring Intelligent Tutor Behavior by Demonstration. *Proceedings of 7th Annual Intelligent Tutoring Systems Conference, Maceio, Brazil*. Page 162-173
5. 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.
6. 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.
7. Razzaq, L., Feng, M., Nuzzo-Jones, G., Heffernan, N.T., Aniszczyk, C., Choksey, S., Livak, T., Mercado, E., Turner, T.E., Upalekar. R, Walonoski, J.A., Macasek. M.A., Rasmussen, K.P. (2005) The Assistment Project: Blending Assessment and Assisting. *12th Annual Conference on Artificial Intelligence in Education 2005, Amsterdam*
8. Rose, C. P. Gaydos, , A., Hall, B. S., Roque, A., K. VanLehn, (2003), *Overcoming the Knowledge Engineering Bottleneck for Understanding Student Language Input , Proceedings of AI in Education*.