The Assistment Builder: An Analysis of ITS Content Creation Lifecycle

Terrence E Turner, Abraao L. N. Lourenco, Neil T. Heffernan, Michael A. Macasek, Goss Nuzzo-Jones

Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609
1-508-831-5569
terrence.turner@gmail.com, aln@wpi.edu, nth@wpi.edu, macasek@wpi.edu, goss@wpi.edu

Abstract
Intelligent Tutoring Systems are notoriously costly to construct [1], and require PhD level experience in cognitive science and rule based programming. The purpose of this research was to ease the development process for building pseudo-tutors [6]. Pseudo-tutors are ITS constructs that mimic cognitive tutors but are limited in that they only apply to a single problem. The Assistment Builder is a tool designed to rapidly create, test, and deploy simple pseudo-tutors. These tutors provide a simplified cognitive model based upon a state graph designed for a specific problem. These tutors offer many of the features of rule-based tutors, but with shorter creation time. The system simplifies the process of tutor creation 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, testing, and deploying content easy for teachers. We present data to suggest that users can develop a tutor that can be released to students in under an hour.

Introduction
This research seeks to address the high development time of cognitive rule-based tutors in Intelligent Tutoring Systems (ITS). Despite the effectiveness of model-tracing rule based tutors, it has been shown that development time can be between 100-1000 hours per hour of content created [1][7]. Creating Cognitive tutors also requires high level computer science and cognitive psychology domain knowledge; and typically PhD level experience in artificial intelligence rule-based programming.

The Office of Naval Research funded the Assistment Project to create tools to reduce the cost of making intelligent tutoring systems. There are two ways to reduce these costs. One is to make tools that are faster to use. The other is to make them easier to use, thus removing the need for PhD level artificial intelligence rule-based programmers and cognitive scientists.

Our goal was to provide a tool to allow rapid content creation to users with little computer science or cognitive psychology background. In order to achieve this goal our research focused on developing “pseudo-tutors” [6].

Pseudo-tutors represent a simplified cognitive model that is comprised of a state graph. This graph is finite, and each node represents a possible state of the problem. User actions are represented by arcs in the graph, with specific user actions triggering state transitions [12]. A user’s location in the graph represents the problem’s current state, and student actions correspond to possible transitions from that state.

Despite having similar behavior to rule-based tutors, pseudo-tutors lack the ability to generalize over similar problems [5]. However, they can be designed to predict certain behaviors and respond accordingly. Pseudo-tutors can also combine the state graph with a branching problem structure known as scaffolding. Scaffolds are sub-problems usually designed to address a specific skill needed to solve the initial problem. Scaffolding problems in turn contain their own state graphs, and depending upon student actions, scaffolds can branch into other scaffolds. This provides a means for rich user interaction. The Assistment Builder was designed as a tool to create these types of scaffolding pseudo-tutors and is the basis of our research.

The Assistments Project
The Assistment Project is a research project led by Worcester Polytechnic Institute and Carnegie Mellon University and funded by grants from the Department of Education, the National Science Foundation, and the Office of Naval Research. The mission of the Assistment Project is to provide cognitively based assessment of students. This mission is supported by three goals [11]. The first goal is to provide tutoring content to students. The second goal is to provide useful and up-to-date reports on students to teachers. The final goal is to provide the tools to allow teachers to create their own tutoring content. Most ITS systems are built to assess students’ knowledge of a set of concepts (i.e. exams) or to assist them in acquiring a certain skill (i.e. tutorials). Students’ time is very valuable, and a system that provides both assessments while it assists is poised to make the best use of the time available. The Assistment system does just that: it assists while it assesses.
The Assistment system provides assessment through student reports to teachers. The reports are updated in real time, even as students are using the system. The system provides different types of reports to teachers based on statistical analysis. Some of the most important reports that we provide are the predicted MCAS score for a student, student effort score, the predicted student performance based on skills mapped to previous questions.

The final goal of the Assistment Project is to provide teachers with tools to allow them to easily create content for their own classes. The research involving the Assistment Builder is in support of this final goal. We have created a web based tool that allows teachers to create content online at their own leisure, using whichever platform they have available. We make claims regarding the ease of development for the Assistment Builder and present data regarding the performance of its users.

### Builder Interaction with the CTOP

At the core of the Assistment Project is the Common Tutor Object Platform (CTOP), a lightweight component framework for creating and deploying all applications in the Assistment Project [10]. The CTOP was designed with extensibility in mind it consists of a core object model and a data layer [10]. The core object model contains components considered to be universally applicable to ITS software [10]. The Assistment Builder uses the problem component and its subcomponents, the interface and the behavior. The interface subcomponent is made up of high-level widgets which are interpreted by the runtime application for viewing and interacting with the user [10]. The behavior subcomponent defines the result of an action on the interface; i.e. whether a specific answer corresponds to a transition to a new state in the state graph representing the tutor [10].

The Assistment Builder allows a user to specify the high level widgets to be used for an interface as well as the properties associated with that interface. It does this by using the Interface component API to provide a form based GUI that exposes the configurable parts of the interface in an easy to modify manner. Similarly, the Assistment Builder uses the Behavior component API to display the state graph linking states and strategies in form based GUI that is easy to update. Strategies currently supported include message strategies (messages that are displayed when the user enters a specific answer or requests help), and scaffolding questions, which are represented in a nested list structure not dissimilar from a hierarchical tree. The Assistment Builder also updates the interface and behavior as each one is changed.

### The Assistment Builder

The main goals of the Assistment Builder are ease of use and accessibility during content creation. The initial prototype of the Assistment Builder was developed without the CTOP and suffered from maintenance and stability problems. To address these issues our research focused on pseudo-tutors and used the CTOP component framework for ease of development and maintainability. The web was chosen as the delivery medium to make the tool immediately available to users. The only requirement to use the tool is registration on our website; no software needs to be obtained or installed. Our primary users are middle school and high school teachers in the state of Massachusetts who are teaching the curriculum of the Massachusetts Comprehensive Assessment System; thus, the Assistment Builder was designed with an interface simple enough for users with little or no computer science and cognitive psychology background. The Assistment Builder also includes other tools to allow teacher themselves to create content and organize it into curriculums and assigned to classes, all of which can be done by the teachers themselves. This provides teachers with a total web-based solution for content management and deployment.
Assistments

The pseudo-tutors created by the Assistment Builder are a subset of the tutors possible under the CTOP. These tutors and pseudo-tutors are referred to as Assistments throughout this paper.

An example of a basic Assistment is a top-level problem that branches into scaffolding problems depending on the student’s actions. To simplify content creation there are only five choices of high level widgets for the interface available to content creators: radio-buttons, pull-down menus, checkboxes, text-fields, and algebra text fields. The Assistment Builder also allows users to add images to a problem’s interface. A problem’s state graph consists of only two states. The student will remain in the initial state until they answer the problem correctly, or they are programatically moved forward. Other incorrect student actions will keep them in the initial state, but may be mapped to specific tutoring strategies. These strategies include branching into scaffolding problems, or specific textual and/or visual feedback called buggy messages that address common student errors.

Scaffolding problems are queued immediately after the behavior consumes an interface action that results in a transition to a state containing scaffolds. One or more scaffolding problems can be mapped to a specified user action. In the Assistment Builder an incorrect answer to the top-level problem or a request for hints on the top-level problem will immediately queue a list of scaffolding problems specified by the content creator. Upon answering a scaffolding problem correctly the student is presented with the next one in the queue until it is empty. When a Assistment has no more problems in queue it is considered to be finished.

Aside from buggy messages and scaffolds, a problem can also contain hint messages. Hint messages provide insights into methods to solve the given problem. Combing hints, buggy messages, and scaffolds together provides a means to create Assistments that are simple but can address complex behavior. Content creators can create complex tree structures of problems each with their own specific buggy messages, hints, and possibly sub-scaffolds.

Assistment Builder Structure

We constructed the Assistment Builder as a web application for accessibility and ease of use purposes. A content creator can build, test, and deploy an Assistment without installing any additional software. It is a simple task to design and test an Assistment and release it to students. If further changes or editing are needed the Assistment can be loaded into the Assistment Builder, modified, and saved; all changes will immediately be available in all curriculums that contain the Assistment. By making the Assistment Builder available over the web, new features are instantly made available to users without any software update. The central storage of Assistments on our servers makes a library of content available to teachers which they can easily combine with their own created content and release to their classes organized in curriculums.

Another goal was to redesign the Assistment Builder to make use of the CTOP component framework. To do this the Apache Struts Framework was used in conjunction with the CTOP to maintain a strict MVC architecture. By following a strong Model 2 Model View Controller (MVC) design pattern extending the Assistment Builder is also easy. The CTOP is designed to be extendable with new types of tutors, widgets, and user interfaces. The Assistment Builder is only concerned with a specific portion of the CTOP, but whenever new widgets or functionality is added all that needs to be done is adding new controllers and views. Sharing code between the Assistment Builder and CTOP means less code to write as well as swift benefit from improvements to the CTOP. The decoupled nature of the Assistment Builder also makes it easy to change or update the web forms that are presented to users.

Features

The initial view presented to users of the Assistment Builder is a top level problem. The view has been redesigned based on user input. At the very top of the screen are several links to help manage Assistments. These are show in figure 3. The problem is blank and users can enter answers, buggy messages, question text and/or images as well as selecting the interface widget they wish. A content creator can also add hints. However, hints and scaffolds are mutually exclusive in the top level problem, and a user must select either one for the top level problem. Each section in the problem view is collapsible to allow users to conserve screen space.

Figure 3

The question section is the first section that content creators will usually use. This section allows a user to specify a problems question text using html and/or images as well as select the interface widget they wish to use and the ordering method answers. There are currently three ways to order answers: random, alphabetic, or numeric. This interface is shown in figures 4 and 5.

Figure 4
The answer section of the problem view allows a content creator to add correct answers and expected incorrect answers. Users can map buggy messages to a specific incorrect answer. Users can also edit answers or toggle their correct or incorrect status. The answer section is shown in figure 6.

Figure 6
The hint section allows users to enter a series of hints to the applicable problem. Hints can be reordered. This section contains an option to create a bottom out hint for the user that just presents the student with the solution to the problem. This is shown in figure 7.

Figure 7
A typical Assistment will contain scaffolds and after a user is finished creating the top level problem they will proceed with adding scaffolds. The view for a scaffolding problem is exactly the same as that for the top level problem, only slightly indented to mark it as a scaffold.

Knowledge Component Tagging
The Assistment Builder supports others applications besides content creation. One of these applications is the mapping of knowledge components, which are organized into sets known as transfer models. Knowledge components are a means to map certain skills to specific problems to specify that a problem involves knowledge of that skill. This mapping between skills and problems allows the reporting system to track student knowledge over time using longitudinal data analysis techniques [3].

In a separate paper submitted to WWW2006, we report on the ability to track the learning of individual skills using a coarse-grained model provided by the state of Massachusetts that classifies each 8th MCAS math item in one of five categories (i.e. knowledge components in our project): Algebra, Measurement, Geometry, Number Sense, and Data Analysis [3].

The current system has more than twenty transfer models available, each with up to three hundred knowledge components. In order to more efficiently manage transfer models, the Assistment Builder makes use of the preference architecture, allowing users to specify the transfer models they will use. Once those are specified, the user is allowed to browse the knowledge components within each transfer model and to map the ones they select to the problem.

Figure 8

Metadata Tagging
Another application that the Assistment Builder supports is the association of metadata with problems. Users are allowed to associate information such as problem source (i.e. MCAS), year, number, season, comment, description, and whether calculators are allowed with each question. This information may be displayed to the user (i.e. whether calculators are allowed) or it may be used solely for accounting (i.e. determining whether all the items in for a specific year have been built).

Figure 9

Methods
The Assistment Builder was designed to log user actions while building Assistments. Each log message contained
the action logged (e.g. editing a hint, adding an incorrect answer, uploading an image, etc.) the user who performed the action, as well as a timestamp. Most content creators also spend time outside of the Assistment Builder planning out content and editing images. We logged the creation and editing of various types of Assistments. Some Assistments were simply a single MCAS problem entered into the system with no scaffolds, hints, or bug messages. Others were more typical Assistments that contained multiple scaffolds. Some were simply Assistments that had been built and were now being modified with different numbers, otherwise known as morphs. A significant portion of user time is spent outside of the Assistment Builder planning out content and creating images. Thus we also performed a survey with content creators and asked them to estimate how much time they spent building specific items in the logs. They were asked to break down the times according to time spent on each task.

Results & Analysis

Data was obtained for over 271 Assistments being created and edited. Some were simply a single MCAS problem entered into the system with no scaffolds, hints, or bug messages. Others were more typical Assistments that contained scaffolds, hints, and buggy messages. Some were simply Assistments that had been created and were now being modified with different numbers, these are known as morphs.

Many teachers were able to use the Assistment Builder as part of a University course, and a teacher was observed in our lab creating 3 items in about two hours. In the past 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 several months, otherwise known as curriculums and Assistments were simply a single MCAS problem entered into the system with no scaffolds, hints, or bug messages. Some were simply Assistments that had been created and were now being modified with different numbers, these are known as morphs.

After several Assistments are built creating a curriculum is a simple task; in the curriculum creation screen the user is presented with a list of all available Assistments, each of which can be selected for inclusion in the curriculum. Once all Assistments have been selected the curriculum is made. This usually takes less than one minute per Assistment.

We wanted to focus on Assistments that had been created, organized into a curriculum, and deployed to classrooms. These Assistments are presumably of higher quality, and they were the majority of Assistments logged in the system. This is because many of the curriculums and Assistments currently in use by students were created before actions were logged.

We obtained data for four users who created a combined total of 25 Assistments that were then released to students. Each of these users has created several Assistments and was familiar with the system. These users log data were compared with their estimates on the time spent creating each item. The data is presented in table 1. The columns in the table are identified as follows: S is the number of scaffolds in the problem, I is the time spent creating images outside of the Assistment Builder, P is time spent planning the Assistment outside of the Assistment Builder, B is the time spent inside the Assistment Builder to create the item, and L is the time spent on the Assistment Builder according to the logs.

A single Assistment is approximately two minutes of content; this suggests a 10:1 ratio of creation time to content, a 10 fold improvement from the previous literature. However, users reported an average time of about 60 minutes to build an Assistment. This time

<table>
<thead>
<tr>
<th>User</th>
<th>Assistment</th>
<th>S</th>
<th>I</th>
<th>P</th>
<th>B</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>10</td>
<td>30</td>
<td>60</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>45</td>
<td>18</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>25</td>
<td>19</td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>30</td>
<td>33</td>
</tr>
<tr>
<td>A</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>35</td>
<td>37</td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>60</td>
<td>10</td>
<td>17</td>
</tr>
<tr>
<td>A</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>45</td>
<td>14</td>
</tr>
<tr>
<td>A</td>
<td>8</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>30</td>
<td>36</td>
</tr>
<tr>
<td>A</td>
<td>9</td>
<td>3</td>
<td>3</td>
<td>60</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>A</td>
<td>10</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>25</td>
<td>17</td>
</tr>
<tr>
<td>A</td>
<td>11</td>
<td>4</td>
<td>3</td>
<td>60</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>A</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>60</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>B</td>
<td>13</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>15</td>
<td>17</td>
</tr>
<tr>
<td>B</td>
<td>14</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>15</td>
<td>3</td>
<td>0</td>
<td>5</td>
<td>7</td>
<td>27</td>
</tr>
<tr>
<td>B</td>
<td>16</td>
<td>3</td>
<td>5</td>
<td>50</td>
<td>15</td>
<td>13</td>
</tr>
<tr>
<td>B</td>
<td>17</td>
<td>3</td>
<td>30</td>
<td>10</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>B</td>
<td>18</td>
<td>6</td>
<td>150</td>
<td>40</td>
<td>30</td>
<td>25</td>
</tr>
<tr>
<td>B</td>
<td>19</td>
<td>4</td>
<td>60</td>
<td>15</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>B</td>
<td>20</td>
<td>5</td>
<td>40</td>
<td>15</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>B</td>
<td>21</td>
<td>4</td>
<td>60</td>
<td>20</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>D</td>
<td>22</td>
<td>11</td>
<td>0</td>
<td>5</td>
<td>50</td>
<td>40</td>
</tr>
<tr>
<td>D</td>
<td>23</td>
<td>1</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>D</td>
<td>24</td>
<td>3</td>
<td>0</td>
<td>10</td>
<td>40</td>
<td>30</td>
</tr>
<tr>
<td>D</td>
<td>25</td>
<td>8</td>
<td>0</td>
<td>10</td>
<td>30</td>
<td>20</td>
</tr>
</tbody>
</table>

A single Assistment is approximately two minutes of content; this suggests a 10:1 ratio of creation time to content, a 10 fold improvement from the previous literature. However, users reported an average time of about 60 minutes to build an Assistment. This time
includes estimated times spent outside of the actual builder application planning and editing images. This leads to an estimated ratio of 30:1 which is still an improvement in content creation times.

Conclusions

The goal of the Assistment Builder was to provide a system that easily allowed users to create and edit content. The data collected so far suggest that the tool does allow content to be created in a relatively short amount of time. In order to do so limits were placed on the type of content that could be created with a focus on pseudo-tutors and a web driven interface. The Assistment Builder has been in use for over a year and utilized by many users, including teachers, have been able to create over a thousand Assistments. These pseudo-tutors are now deployed on the web. Without the Assistment Builder much of this content would not exist.

Acknowledgements

We would like to acknowledge the US Department of Education, National Science Foundation, Office of Naval Research, and the Worcester Public School System for providing the means to accomplish this research.

References