EXTRACTING CONCEPT MAPS FROM INSTRUCTOR AND LEARNER BEHAVIOR IN AN INTERACTIVE E-LEARNING ENVIRONMENT

THESIS

Presented to the Graduate Council of
Texas State University-San Marcos
in Partial Fulfillment
of the Requirements

for the Degree

Master of SCIENCE

by

Joshua Emmett Byrne, BChE

San Marcos, Texas
August 2007

DEDICATION

This thesis is dedicated to my mother, Kathy Byrne, who was my typist and proofreader throughout grade school, and who avoided the same fate in this endeavor thanks to the invention of word processing software for personal computers.

ACKNOWLEDGEMENTS

Thank you to my committee: Dr. Deborah East, Dr. John Durrett, Dr. Jawad Drissi, and Dr. Carol Hazlewood, who graciously accommodated my working on my thesis from a thousand miles away. A special thanks to Dr. John Durrett. Thank you to my wife, Lucia Byrne, who helped with proofreading, and along with my son Lex accommodated many absences. Many thanks to Bill Alfveby, who went above and beyond the call of duty to help with the experiment and testing UTS. Thanks to my parents John and Kathy Byrne who have always supported my academic endeavors. Finally, thanks to the many friends, colleagues, and students who participated in this study and provided comments on UTS, including Steve Byrne, Vijay Dayafule, Erik Mohr, Harold O'Dell, Mouli Paturi, Drew Thorstenson, and Ron Waara.

This manuscript was submitted on March 24, 2007.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION TO THE STUDY

The structure of knowledge as seen by an expert does not represent the order in which most novices naturally learn that information. The goal of this study is to experimentally explore that hypothesis by comparing concept maps designed by experts to concept maps derived from student and instructor behavior in a Web-based instructor facilitated training (WBIFT) course. Chapters 4 and 5 detail the experiments and their results.

The largest part of the effort in this study was creating the WBIFT software, hereafter called the Universal Tutoring System (UTS). UTS delivers learning content to students and assesses their learning. Students choose their own path through the course by submitting questions and by studying answers to questions submitted by other students. Two types of instructors—tutors and librarians—facilitate learning by answering student questions. Tutors answer students' questions by creating new content. Librarians answer questions by linking to existing content. Chapter 3 details the design and implementation of UTS.

This study involves several knowledge domains including computer science and instructional design. Chapter 2 provides a survey of the literature in those domains that

informed this study. Chapter 6 examines potentially useful findings in these areas that are not directly related to the basic hypothesis of the study and suggests further study.

One important application of this research is for intelligent tutoring systems. Intelligent tutoring has been shown to be more effective than the more typical linear sequenced courses (Murray, 1999). Unfortunately, intelligent tutoring is not widely used due to the considerable cost and difficulty of authoring (Oguejiofor, 2004), especially authoring the ontology that maps student needs to instructional content. Concept maps derived from UTS are useful for this application and could be developed at a fraction of the cost of expert designed concept maps. Others have alluded to automatic extraction of concept maps for ADL (Abdulah and others, 2004), but my literature search did not reveal any truly similar work.

The UTS software developed during this study is best characterized as a proof of concept sufficient for research purposes, but not yet robust enough for real-world use. Chapter 6 suggests further research and development needed for UTS to achieve broader applicability.

Changes From the Original Scope of the Study

While the primary objective and overall approach remained the same, one difference between this study and that envisioned in the original thesis proposal is the use of SCORM. My thesis proposal envisioned implementing the learner interface in a SCORM conformant learning management system (LMS). The SCORM standard defines the interface between learning content and conformant LMSs. Unfortunately, closer inspection reveled that SCORM does not allow enough visibility among learning objects to work easily with the Universal Tutoring System.

As a result, I built UTS as a single Web application. This added significantly to the development effort required, but gave a great deal of flexibility that was ultimately useful. The concept maps derived from UTS can still be used to drive the sequencing of SCORM conformant learning content that is delivered separately.

Nomenclature

The following terms are applied broadly in common usage, but I construe them more narrowly to improve clarity.

Question

In this thesis and in the UTS documentation, the term "question" refers to a question a student asks about a unit of content. An assessment question the student answers to demonstrate competence is referred to simply as an "assessment."

ADL and e-Learning

The US Department of Defense typically uses the term Advanced Distributed Learning (ADL) to refer to electronically delivered educational content. Outside of the military, the terms e-Learning and Computer Based Training (CBT) are more common. I have adopted the term ADL for content that is not instructor led and Web-Based Instructor Facilitated Training (WBIFT) for computer based learning that includes instructor intervention.

Concept Map vs. Ontology

Concept maps and ontologies are similar for the purposes of this study, and I occasionally refer to the literature for ontology development with the implication that the

findings apply to concept maps as well. For readability, I use the term concept map throughout. The reader should understand this to include node attributes, machine readability, and other features which are not formally part of concept mapping.

Instructor

In the context of UTS, there are two instructor roles: the tutor and the librarian. Because these two share many traits, I often refer to them collectively as an "instructor" role or user type. In discussion of ADL production, I refer to the "instructor" as the person or people creating a course. In practice ADL development typically involves a number of specialists including instructional designers, subject matter experts, visual designers, and media developers.

Sharable Content Object (SCO)

A SCO is a piece of ADL content conforming to the SCORM standard—typically centered on one or more Web pages and supporting media assets. In practice, the term SCO is often applied more generally to a collection of ADL content smaller than a module, which is in turn smaller than a course. In this study the term SCO refers to a single page of ADL content, regardless of whether it follows the SCORM standard.

UTS

Universal Tutoring System is the working name for the software system developed for this project. At the time of writing, a Web search showed that this name was not in common use by any other software system. Another search will be conducted before the system is publicly released. I am still trying to determine whether there is

commercial applicability for UTS, but readers interested in obtaining the software for research purposes are encouraged to contact me.

CHAPTER 2

LITERATURE SURVEY AND BACKGROUND INFORMATION

Instructional Technology

ADL Standards

Work on the Sharable Content Object Reference Model (SCORM) was initiated in 1997 by the US Department of Defense with the goal of enabling ADL interoperability, affordability, durability, reusability, and accessibility, which are sometimes referred to as the SCORM "ilities" (Advanced Distributed Learning, 2007).

At its core, SCORM has two major components. The first is a content packaging and metadata standard that defines how the files that compose a conformant SCORM package are to be structured and the XML metadata that it must include. The second is a standard set of JavaScript functions that a conformant learning management system must provide to an ADL course. For example, a conformant LMS must provide the loadPage() function (Advanced Distributed Learning, 2004).

Of its original goals, SCORM has been most successful at achieving portability. Today, SCORM conformant content will run on most conformant learning management systems with a small amount of testing and modification. SCORM has been less successful at enabling reusability, in large part because it is very difficult to author

context independent content. UTS attempts to overcome this problem by using student questions to provide context.

My original intention was to implement the student interface to UTS as SCORM packages on an open source LMS. After further analysis, I determined that this was not a good design. SCORM was designed before the limits imposed by modern browsers on cross-domain scripting (the ability of a page to load dynamic content from more than one domain). These restrictions make it difficult to provide content from a server different from the one that contains the learning management system, further limiting its usefulness for UTS (Brusilovsky, 2004).

Organizing content into small independent SCOs is important to evaluating my hypothesis because it is the most visible expression of the instructor's understanding of the structure of the content. In a typical ADL course design, content is organized in an outline that reflects the instructor's understanding of the best order to learn it in. If the generated concept map deviates significantly from this outline, it supports the hypothesis that learners conceptualize the content differently than an expert instructor.

SCORM also supports learning objectives, which might be a close analog to the nodes of the designed concept map. Unfortunately, it does not support relationships among the nodes, and the sequencing model for SCOs is primarily tree-based, not map-based. So, I did not use SCORM learning objectives as part of this study.

Intelligent Tutoring

The term "intelligent tutoring" has been applied to a broad range of computer-based training systems, but they typically share three common features (Evans and others, 1993) (Angelides and Paul, 1993).

1. The system has a model of the student's understanding
2. The system has a model of the subject domain
3. The system adapts the material presented to the student based on 1 and 2

Intelligent tutoring has been shown to be more effective than traditional linear ADL (Murray, 1999), but its application is limited due to the cost of developing domain models, learner models, and the additional content required to accommodate different learning styles (Abel, 2004). Intelligent tutoring is an important application of my research because the self organizing system approach embodied in UTS eliminates the need to expressly create these things.

Some intelligent tutoring methodologies draw extensively from the area of cognitive psychology, where there has been an enormous amount of effort to build models of how people learn, and how concepts are represented in the human brain. Rather than explore this work in great depth, I observe that despite great progress these theories have had limited impact on the way ADL is delivered. Instead, I focus on finding a self organizing approach that empirically delivers superior learning—even if the exact mechanism isn't completely understood.

Concept Maps and Ontologies

Concept Maps

Novak and Cañas (2006) describe concept maps as "... graphical tools for organizing and representing knowledge" composed of the following elements:

1. Concepts enclosed in circles or boxes
2. Connecting lines
3. Linking phrases
4. Cross links
5. Examples to clarify

These are typically organized so the most general or abstract concept is at the top of the concept map. Two or more concepts linked by a connecting line are said to form a proposition.

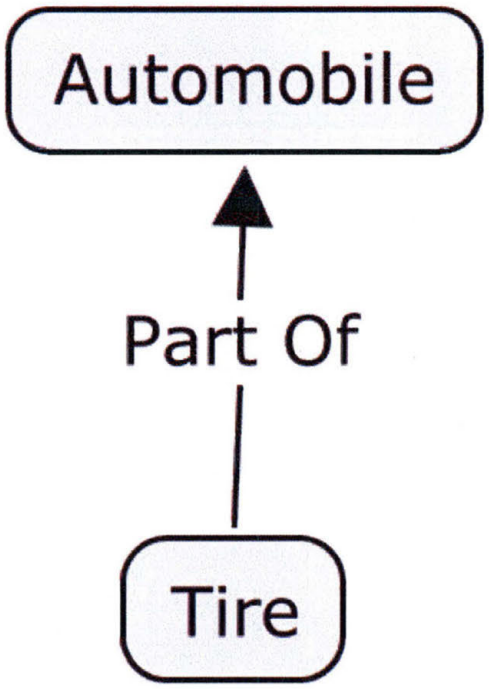

Figure 1: Proposition in a Concept Map

Concept map developers sometimes talk of a concept map "cloud" or "soup": a large collection of propositions, often formed by joining many different concept maps. This is similar to an upper ontology.

Constructing Concept Maps

Concept maps are most effective when created in a particular context. This context is typically provided by a "focus question" the map is intended to answer (Novak and Cañas, 2006). The following steps are recommended to produce useful concept maps.

1. Define focus question and domain
2. Identify 15 to 25 key concepts
3. Sort concepts by level of abstraction

4. Construct a preliminary map
5. Seek cross links
6. Revise many times
7. "Clean-up" visually to improve clarity before publishing

Concept Map Versus Ontology

The literature is not entirely consistent in defining the term ontology (Oguejiofor and others, 2004). Russell and Norvig (2003) state that "the ontology determines what kinds of things exist, but does not determine their specific properties and interrelationships." Sugumaran and Storey (2006) directly contradict this when they say "an ontology is conceptually represented as a semantic network where the nodes correspond to the ontology's concepts or terms, and the arcs correspond to various relationships." These are only two of many published definitions.

Concept maps are more narrowly defined, as described in the previous section, but are not quite sufficient for my purposes. There is not a rigorous mathematical definition of a concept map, and the literature is more focused on the educational benefits of the process of constructing them than on defining them as a means of machine useable knowledge representation.

In this study, I use both the term "concept map" to refer to a directed acyclic graph with nodes representing concepts and arcs representing the relationships between and among concepts. An arc connecting two nodes forms a proposition, and an abstract concept may encapsulate many propositions.

The concept maps generated by UTS are anonymous; nodes and relationships are not explicitly named by the system (Patel-Schneider and others, 2002). This is somewhat unusual because most concept mapping efforts focus on the question of how to name

things for human understanding. However, human readable names are not required to apply the concept map to intelligent tutoring, as long as the manner in which it was derived is well understood.

In principle, it might be possible to reduce all human knowledge to a set of propositions. In practice, it is only practical to develop high granularity concept maps for small domains. Efforts such as CYC to develop upper ontologies that cover large domains have been going on for many years and are incomplete despite having more than a million propositions (Cycorp, 2007). Yet concept maps are still useful for human understanding because the human brain encodes an enormous number of propositions, but in most cases we consciously deal with a smaller number of abstractions that encapsulate the more fundamental propositions. For example, a simple proposition such as chicken:buy-at:store encapsulates a large number of fundamental propositions, yet we are never consciously aware of many of them unless one turns out to be faulty.

Asking an expert to design a concept map is a good way to determine the abstract concepts and relations that make up the expert's view of a domain because that view is relatively stable. Student designed concepts maps are a poor way to measure novice understanding of a domain because the concept maps themselves are an excellent learning tool (Novak and Cañas, 2006) so the act of constructing the map is likely to significantly change the student's understanding. Instead, I deduce the student's conception of the domain from the questions they ask as they learn. For individual students, this approach has the same drawback as asking students to create concept maps—by the time they answer enough assessments and ask enough questions, they will know too much about the domain to be considered novices any longer. So, I look at

students in the aggregate when determining how the concept implied by many novices compare to the concept map designed by a group of experts.

Self-Organizing Systems

In self organizing systems, complex aggregate behavior is achieved through the actions of individuals who act locally and have little or no visibility on the system as a whole. Such systems appear in the physical world in biology, chemistry, and human societies.

UTS is a self organizing system in the sense that students, tutors, and librarians carry out a relatively simple set of tasks teaching and learning in a particular domain.

CHAPTER 3

UNIVERSAL TUTORING SYSTEM

This chapter describes the UTS process and WBIFT software that evolved as a result of my research. This chapter has two sections; the first describes the instructional methodology for using UTS. The second section describes the design of the UTS software.

UTS is required for testing my hypothesis because it generates data for the concept map representing the student conception of the domain being taught. This is a better approach than simply asking students to create their own concept maps because creating concept maps is an important teaching technique in itself, and the end product may not really represent the student's knowledge as a novice.

UTS Methodology

Over the course of this project, a process for authoring ADL content using UTS evolved. This section describes the state of the process and related best practices after the experiments and subsequent analysis. Chapter 4 provides additional information about how the process evolved over the course of development.

Figure 2 shows the UTS process for a single module. There is no rigid definition of the length of a module, but in my experiment the goal was that a typical student should complete a module in an hour.

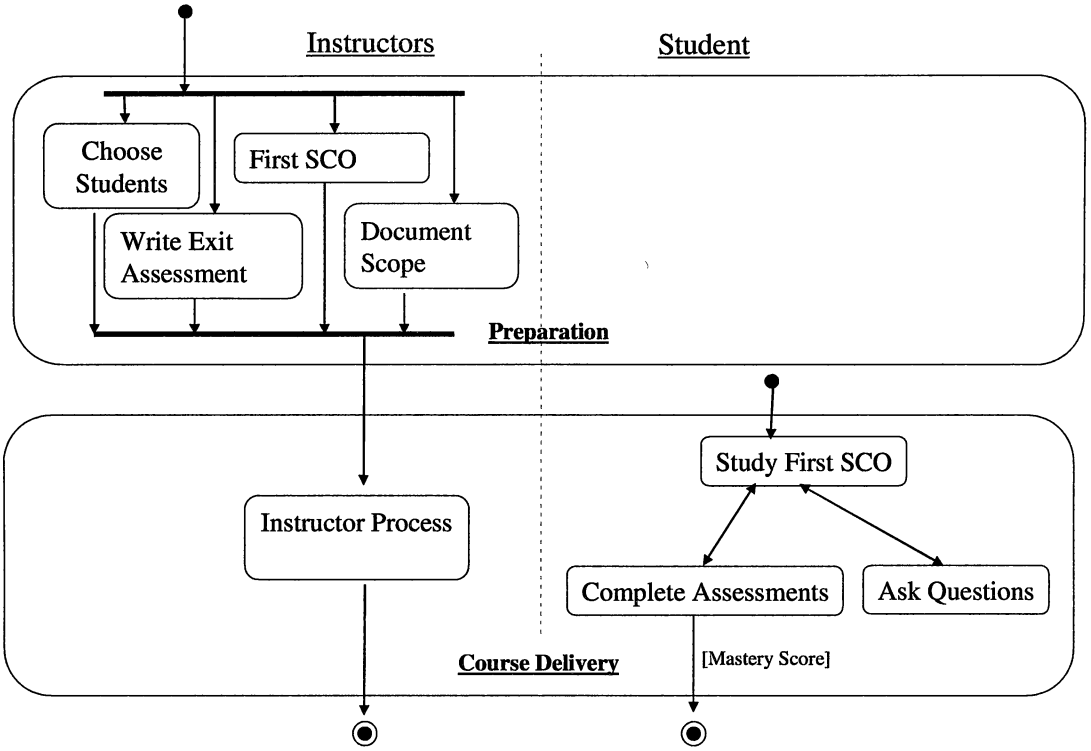

Figure 2: UTS Process Activity Diagram

UTS could be used to deliver training on an ongoing basis, but my objective was to use it as an experimental authoring environment where the interactions of students and instructors would ultimately evolve to ADL that would be useful without an instructor.

The UTS process starts with a group of at least four knowledgeable instructors and ten to twenty representative students. The students understand that they are engaged in a course development activity and that the content may not be as refined as fully developed courses they have taken. As shown in Figure 2, the UTS process has three major elements: preparation, course delivery, and analysis. Each is detailed below.

Preparation

During the preparation phase, the lead instructor starts by writing the module name and a narrative description of objectives for the course. This could be a paragraph

or a list of five to ten items. For my experiments I used the concept map focus question. The lead instruction then gathers the other instructors to create a list of post requisite assessments that will define the bounds of the module. I found 10-15 questions to be sufficient, as the number grows significantly during module delivery. The instructors also work together to create the introductory SCO and its initial assessments.

After the course definition, exit assessments, and initial SCO, the student test group is chosen. It is important to choose a group that understands they are involved in course development, and are willing to overlook some delays at the beginning. Students in the development group should be chosen to reflect the diversity of the ultimate target audience. For example, if the course is intended for students with a wide range of prior experience in the topic, then the test group should include students who have significant prior knowledge as well as those with little or no background in the topic. This situation is typical of a corporate training environment where the goal is for all students to reach a certain competency level, but recognizes that some of them might be 90% of the way there already and others only 10%. By including both kinds of students in the development group, one ensures that the final course will have paths for each type of student to complete the course in the most efficient way.

An academic course might have the opposite situation if it has well defined prerequisites and students have fairly consistent prior domain knowledge. In this case it might be necessary to diversify the development student group in a different way; it might be desirable for non-native English speakers to be represented, for example.

The test group can all be registered in UTS at the same time, but I discovered that it is a good idea to start them two or three at a time. Otherwise too many students will

quickly overwhelm the instructors with questions, leading to a lot of equivalent questions. There is a risk that this approach leads to a course heavily weighted to the predilections of the first couple students, but I doubt that it is a serious problem in practice because I saw many questions on the initial SCOs, even from students who joined the course after a significant amount of content had been created.

Course Delivery

When the students first log into the course there is only a single SCO, so their first steps are to read the content, attempt the assessments, and ask questions. Where possible, it is helpful to start with a synchronous session where the students and instructors are online at the same time and can rapidly ask questions and build out the initial SCOs. Over subsequent hours or days, depending on the difficulty of the exit assessments, students continue asking questions and completing SCOs until they have scored sufficient points on the exit assessments to pass the module. As the content grows, additional students are added.

During course delivery, the tutors look for questions from students and answer them with new SCOs. For each new SCO, the tutor also creates three assessments and evaluates the relationship of other assessments to the new SCO. Tutors are encouraged to go back to the most popular SCOs to improve the writing and add interactivity and assessments.

During course delivery, librarians also look for questions, and answer them using existing SCOs. Similar to tutors, librarians create new assessments and evaluate existing assessments after answering a question. It is critical that both tutors and librarians rephrase the student questions where necessary. In some cases, they should even

communicate directly with the students to determine the student's intent for a particular question.

Analysis and Production

After a number of students have completed the course, the number of new questions will decline, or the ratio of linked SCOs from librarians to new SCOs from tutors will rise. The average time to complete the course should also drop, as the most common questions are answered. These measures indicate when the topic has been adequately covered for the target student population. At this point it is possible to simply disable the question component on UTS, and use it to deliver the course as ADL instead of WBIFT. This is undesirable in most practical situations because UTS is not a full featured LMS, and because the development process may lead to an excess of closely related questions in some areas, making it hard for students to find what they are looking for. Instead, the data from UTS should be extracted for further analysis and refinement.

Ideally the course delivery phase will include experienced media developers who can take the basic SCOs created by the instructors and augment them with interactive components and high quality graphics. In the analysis phase, the most frequently used SCOs should be identified for further improvement. The least used SCOS and questions should be considered for removal. The course content can then be published in a SCORM or AICC conformant framework for portability to a wide variety of learning management systems. For my experiments, the goal was not the production of a polished ADL course, but the evaluation of student behavior, so I did not go through the final production step.

Software Design

UTS evolved significantly over the course of the three experiments. This section describes UTS at the end of this process. This section is organized into three subsections. The Overall Architecture subsection covers design and implementation issues important to all of the user types: student, librarian, and tutor. The second subsection covers the student user features The third subsection covers instructor features, both tutor and librarian.

Overall Architecture

Components and Deployment

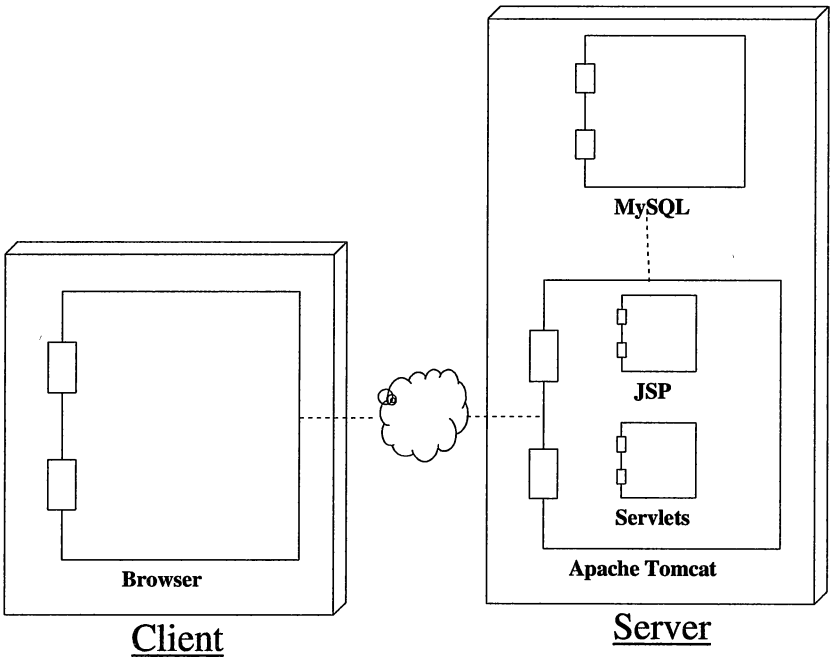

Figure 3: UTS Deployment Diagram

As shown in Figure 3, UTS is built on the MySQL database management system and Apache Tomcat application server. Its user interface is implemented using HTML,

CSS, and JavaScript. Server side coding is in the Java programming language using JSP, Servlets, and JDBC.

Common Patterns

The following patterns are noteworthy because they are used consistently across different parts of the system.

Ajax

The student user interface makes extensive use of Asynchronous Javascript and XML (Ajax) to improve interactivity in a number of places. The instructor interface also uses Ajax in some places. For example, when the instructor chooses a new item from the list box on the assessment editing page, the form elements are updated without refreshing the entire page. This improves response time and general user experience. Figure 4 shows a sequence diagram for this scenario, which is typical of all Ajax use in UTS.

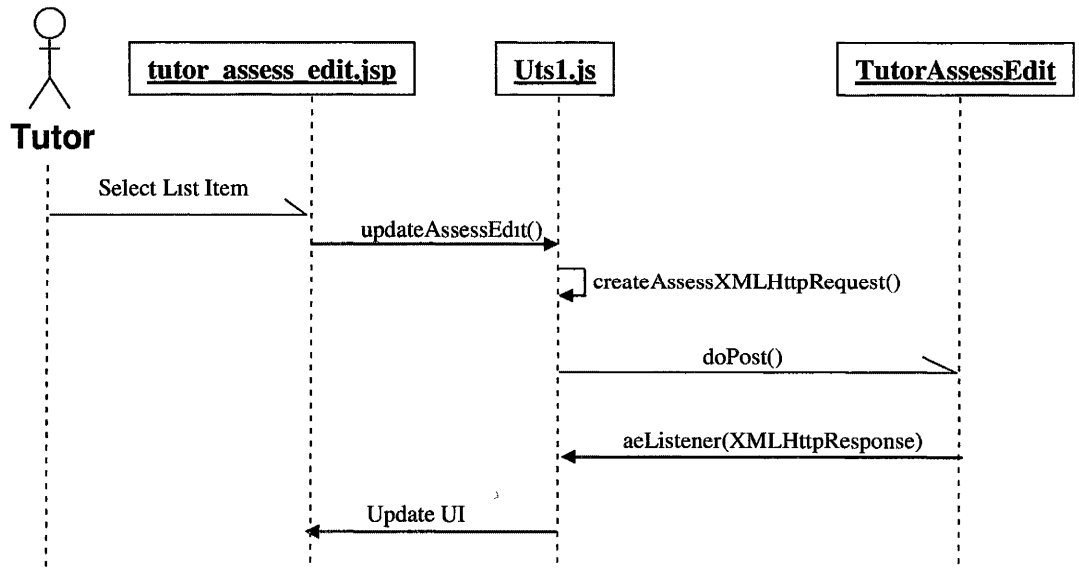

Figure 4: Select New Assessment Sequence Diagram

Readers who are not familiar with J2EE and Ajax should note that this diagram elides (intentionally hides for the sake of readability) many details. For example, the JSP

is really compiled to a servlet that generates HTML which is sent to the browser. It is in the browser that the user selects a list item; JavaScript calls are also internal to the browser. When the listener is invoked, it is the browser's implementation of the Document Object Model (DOM) that is used to update the UI. Figure 4 only shows the Ajax specific aspects of this scenario. Instead of posting directly to the server, the Web page generated by the JSP uses JavaScript to make an asynchronous post, and selectively update the UI based on the response. In a less dynamic implementation, selecting a list item would simply get a whole new page from the server.

JSP vs. Servlets

Two major capabilities of J2EE are Java Server Pages (JSP) and Servlets. JSP allows the developer to mix HTML and Java code in a single source document. This document is then compiled to a Servlet. JSP is particularly convenient for building user interfaces because the developer doesn't need to explicitly write code to output static HTML in a servlet. The drawback of JSP is that the mix of HTML and Java makes them very difficult to read if there is too much Java code. They are also inappropriate for returning data types other than HTML. UTS uses JSP to generate each page of the user interface, including loading data from the database, but does not use JSP to write to the database.

Servlet classes inherit from the HTTPServlet class, which provides methods for handling GET and POST requests from the client browser. UTS uses servlets where a component needs to write to the database, or where a component needs to return XML. The sequence diagram in Figure 5 shows the scenario where a students navigates to the previous SCO. This flow is typical of the use of JSP and Servlets in UTS for both

student and instructor roles, except for doGet() calling doPost(). This was done to accommodate the hyperlink. Most other areas of the US use onclick() or buttons, and call doPost() directly.

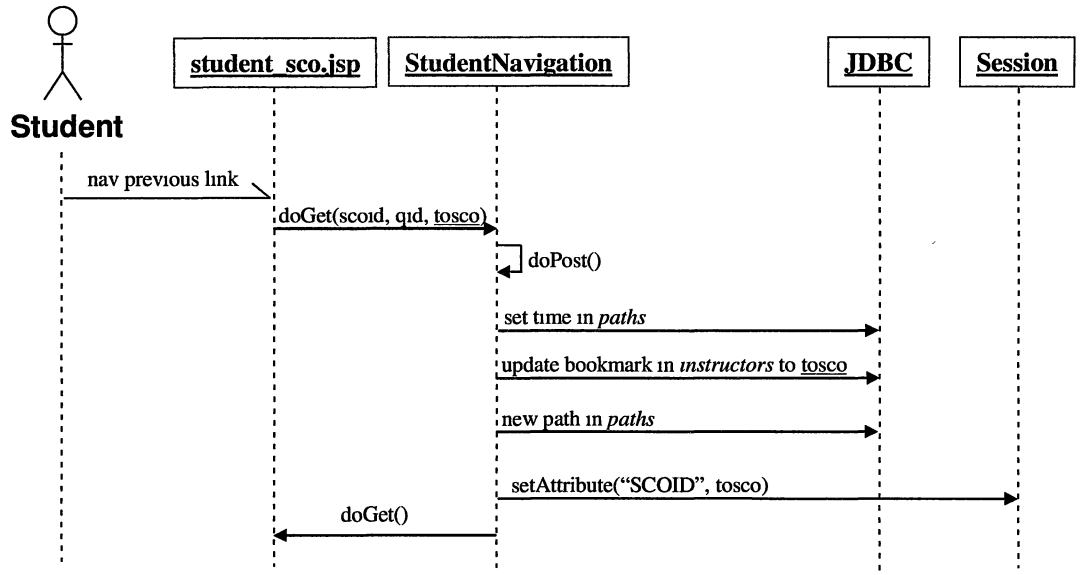

Figure 5: Student Navigate Back Sequence Diagram

In Figure 5, the student presses a link to navigate to the previous SCO. The Web page issues a GET request to the StudentNavigation servlet, which forwards the request to the servlet's doPost(). This extra step makes it easier for HTML to treat a hyperlink to the servlets as a plain HTML request. The doPost() method then makes various calls to the database, and sets the SCO number in the session so that when it redirects back to the JSP, it will display the previous SCO.

Code Conventions

Some code conventions for UTS evolved over the course of the project, so these are not universally followed in the current build. However, they are followed on all new development, and added to old code as time permits.

Array Indexing

Where user interface elements with 1 based indexing correspond to code elements with zero based indexing, make the conversion as close as possible to the user interface.

JSP Logic

Where practical, place Java code in JSP files at the top of the JSP. For example, database queries should be in the <HEAD> of the document with the results assigned to variables that are output by <%= %> in the body of the document. A small amount of Java code is still required in the body of the HTML for iterative building of tables and the like.

JSP Formatting

One unfortunate aspect of the way JSP works is that many uses of whitespace that make the JSP readable render the generated HTML very difficult to read. I chose to err on the side of JSP readability.

File Naming

Where a JSP posts to a single Servlet, use filenames role_pagedescription.jsp and RolePageDescription.java (e.g. tutor_question_list.jsp and TutorQuestionList.jsp). Each JSP should post to its own Servlet. In cases such as student_sco.jsp with many different posts, there should be a servlet for each major functionality.

Student Features

This section details the requirements and design of the student interface to UTS. Students are users who log in to UTS for the purpose of taking a course during the development process. The greatest emphasis was placed on ease of use for the Student interface because it has the most total user interface time, and the least time is available for training each student users.

Use Cases and Feature Requirements

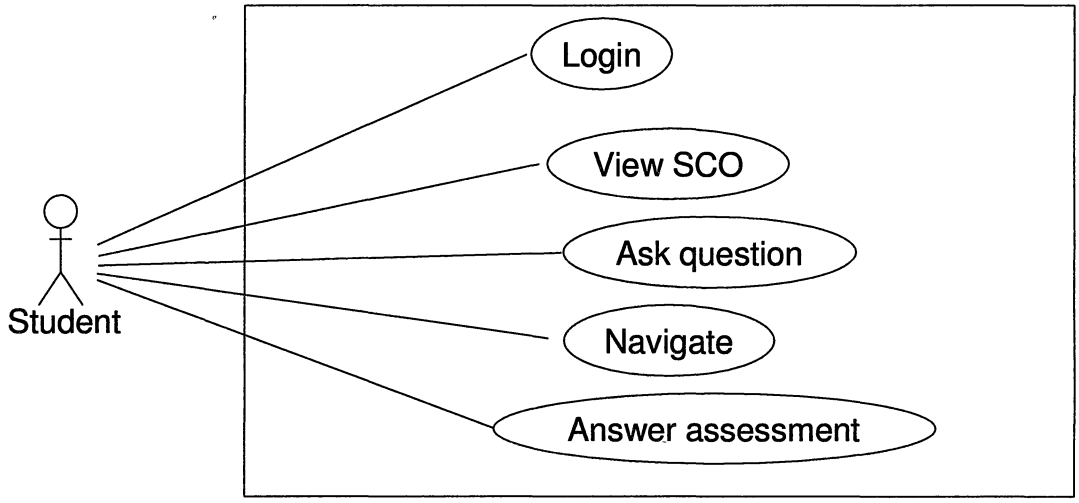

Figure 6: UTS Student Use Cases

Figure 6 shows the overall Unified Modeling Language (UML) use case diagram for UTS students. A use case diagram shows the functionality of a system from the perspective of an outside observer. It shows neither process, nor how that functionality is implemented. The rectangle represents the system, and the ovals use cases. A stick figure—an official part of the UML specification—represents an actor, an entity external to the system that interacts with it in some way. In this example the student is a role for a human being, but an actor could as easily be another computer system.

Comments on Instructional Design

To enable its goal of concept map generation, UTS enforces certain instructional design constraints that would be less restrictive in most other ADL courses. UTS assesses the student on each page because the assessments are the foundation of the concept map extraction algorithm. This would generally be considered excessive assessment in a linear ADL course, so UTS encourages but does not require assessment at any given time. Students are free to browse a course and ask questions as long as they like without taking any assessments. The only restriction is that a student cannot complete the module until they have achieved the mastery score on the exit assessments. Likewise, students can take as many assessments as they like on a particular SCO. If the system runs out of new questions on that SCO, it repeats the ones the student has already taken. Only the most recent attempt for each assessment is counted.

UTS limits each SCO to a single scrollable page. There is not an enforced limit to the length of the page, but I strongly encouraged instructors to create content that fit on an 800x500 screen. UTS also limits the instructor to one figure, graphic, or code sample per SCO. This is a simplification for the sake of the experiment. I envision a future version of UTS accepting SCO uploads from any commercial or open source ADL authoring tool.

UTS also draws assessments for a SCO randomly from four sources: prerequisite, post-requisite, unrelated, and module exit assessments. The proportion of these is configurable, but in the experiment I found 10:60:5:25 to be effective. Delivering assessments that are not post requisite to the SCO is necessary for UTS to determine the relationship of the concepts underlying one SCO to the concepts underlying another.

However it is also distracting to students if they are not warned about it, so students are told that they should expect some questions they cannot answer because the system is trying to determine the best path for them.

User Interface

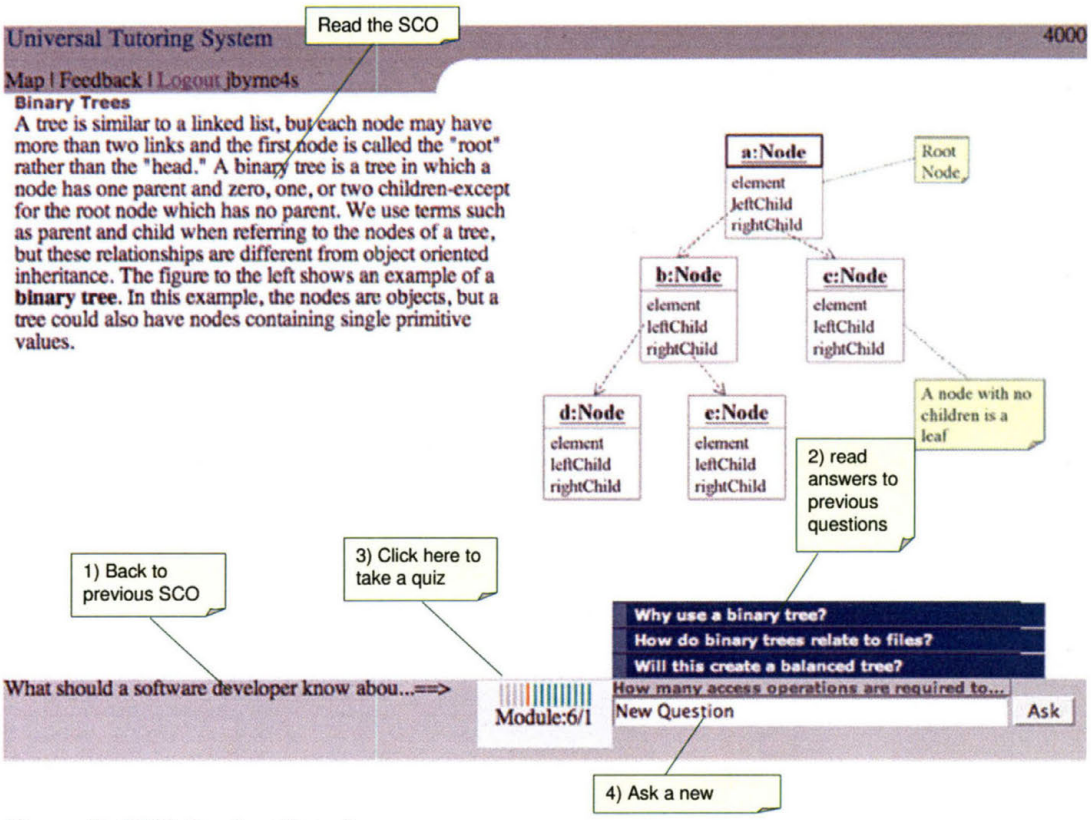

Figure 7: UTS Student Interface

Figure 7 shows the student user interface. On the browser side it is implemented using HTML, CSS, and JavaScript. To provide the greatest simplicity and ease of use for students, it makes extensive use of Ajax—students can ask questions and take assessments without refreshing the whole page.

The back link (1) takes the student to the prior SCO, except when they are in the first (root) SCO of the module, in which case the link is inactivated as shown here. The text of the back link is the question the student followed in order to get to this SCO. In

the case of the root SCO, it is the focus question for the module. The question is truncated in the link if it exceeds a certain number of characters, but the student can see the entire question if they roll over the link.

The forward link (2) shows the recommended next question. For experiment 3 the suggested link is randomly chosen to avoid biasing the student's chosen path through the material. In future iterations of UTS, it could be based on the shortest path to complete the post assessment, on the student's stated preferences, or on a number of other factors. Rolling the mouse over the suggested next question link, the student sees the other question links to choose from.

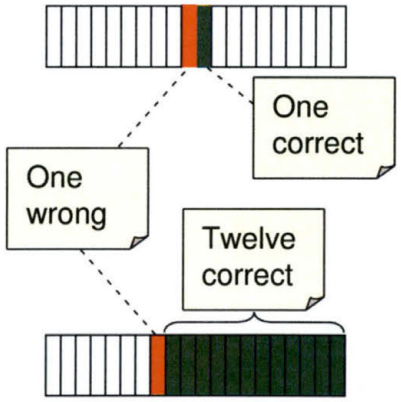

Figure 8: Score Component Concept in Two Different States

By clicking on the assessment component (3) the student shows or hides the semi-transparent assessment area. This design allows the student to see the SCO content while completing the assessment. The top bar on the assessment component shows the student's mastery of this SCO. If the student answers an assessment correctly, a green bar is added; incorrectly and a red bar is added. For example, Figure 8 shows the score component in two different states—first with one correct and one wrong, and then with twelve correct and one wrong. If the student answers enough post requisite assessments,

all bars will be green. My intention is to make a game of the assessments so students are encouraged to answer more than if they were required to answer a fixed number.

The new question area (4) allows the student to submit a new question. Pressing the submit button posts the question asynchronously and shows the student a confirmation message. The question will not appear in the question link list until a tutor creates new content for it, or a librarian links it to a new SCO.

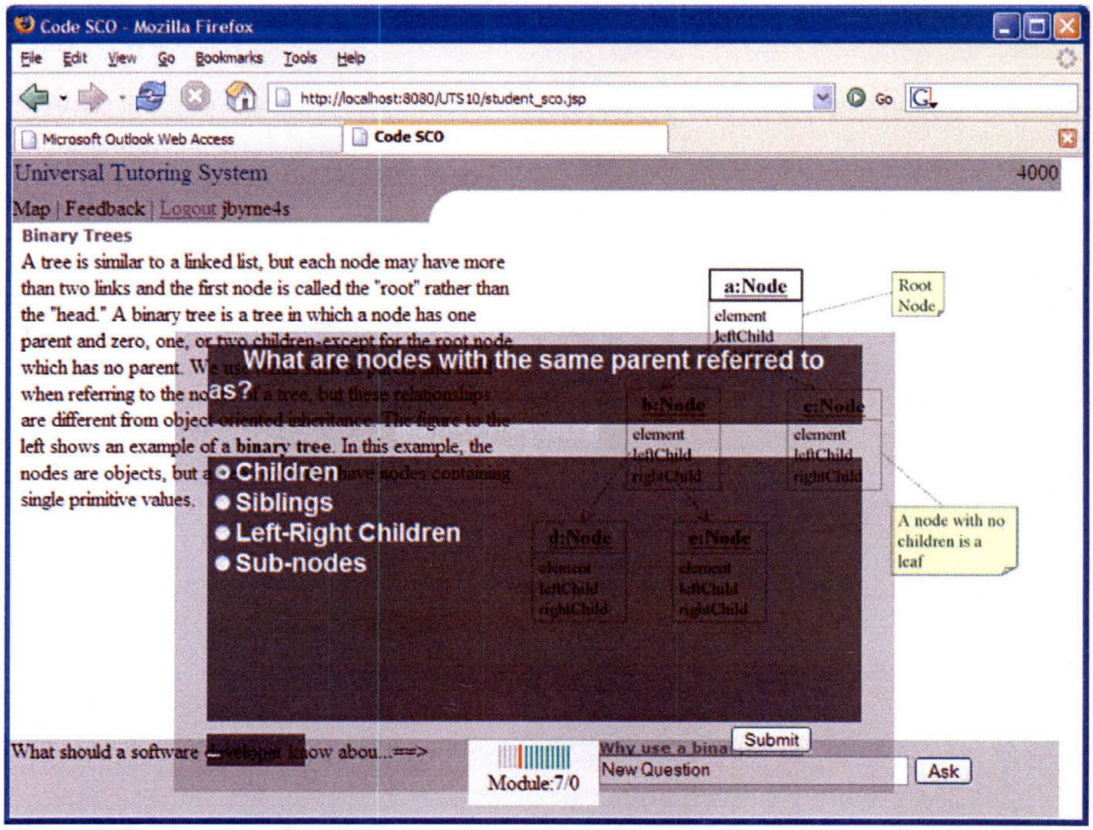

Figure 9: Student Interface With Assessment Interface Visible

Figure 9 shows the student interface with the assessment screen activated. It is semitransparent, so students can review the SCO as they complete the quiz. By making it easy to access the assessments without going to a different page, this interface encourages students to take as many assessments as possible.

Static Design and Database

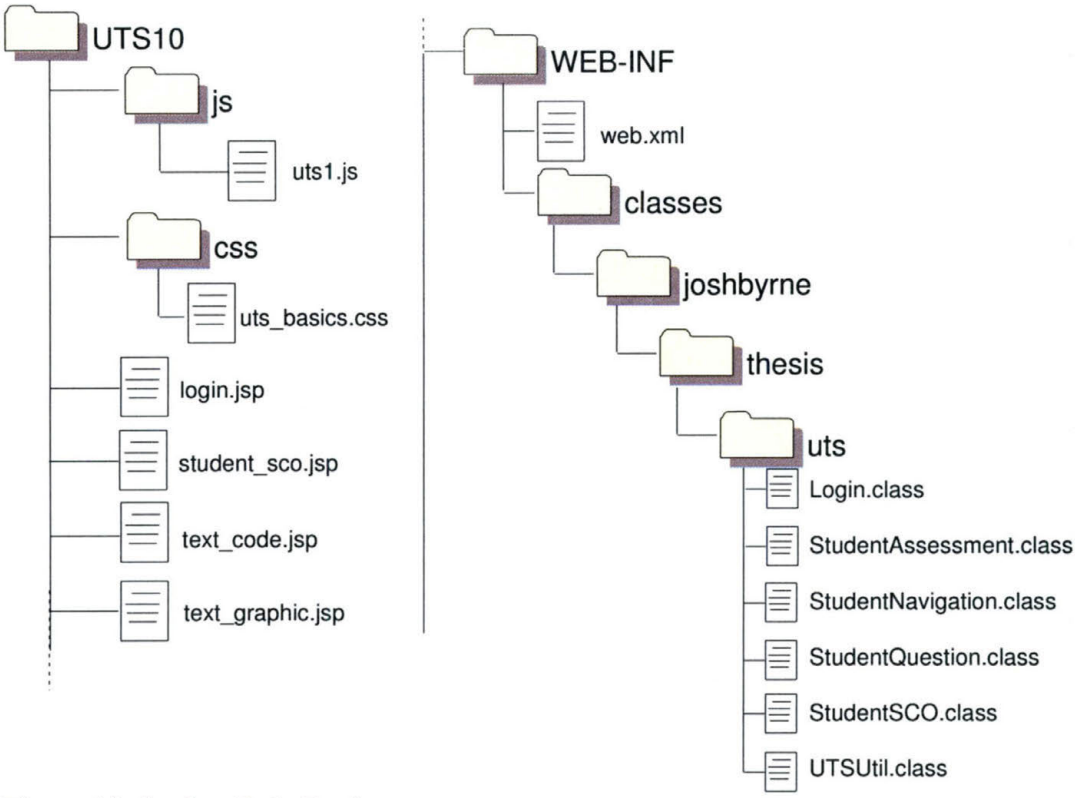

Figure 10: Student Role Deployment

Figure 10 shows the deployment of the files that implement the student role. The primary student interface is provided by student_sco.jsp, and each of its major interactive components is handled on the server side by a servlet called by a JavaScript function. For example, StudentQuestion loads assessments from the database and evaluates student responses.

The text_code and text_graphics JSPs are included by student_sco depending on what type of content a SCO has. This design keeps the individual JSPs from becoming too complex, and makes it easy to provide a SCO preview in the instructor components. Due to extensive use of Ajax, most of the student interface is handled by student_sco, but it is supported by four different servlets for different types of server side data handling.

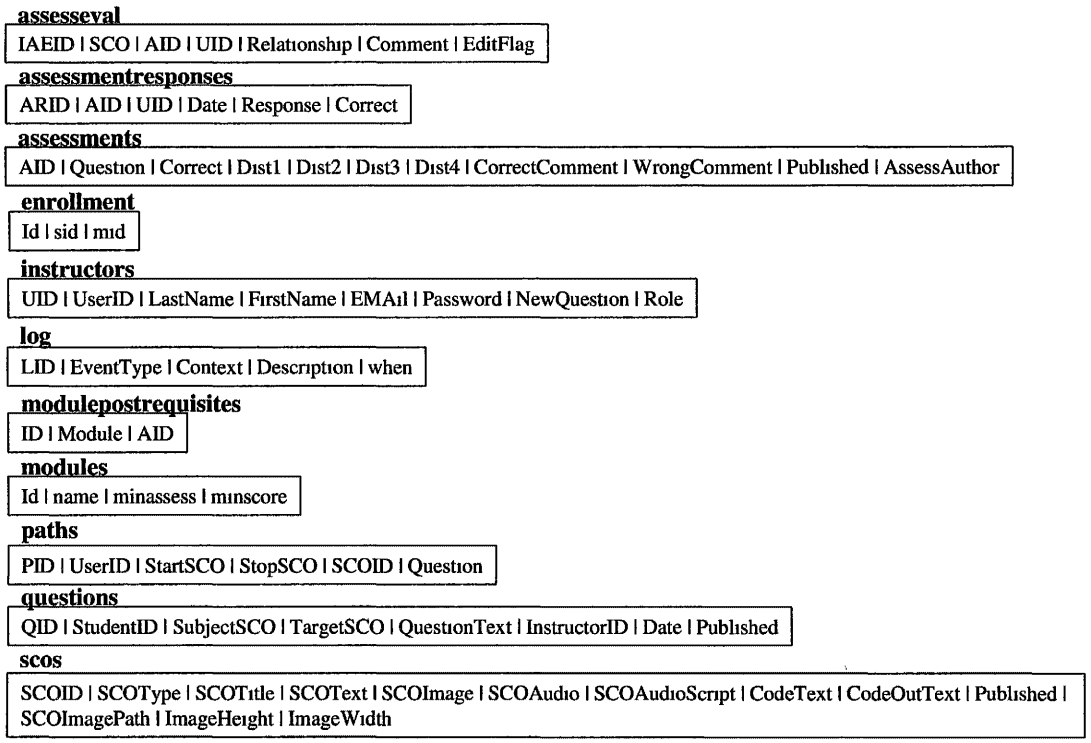

Figure 11: Database Schema

Figure 11 shows the database schema for UTS. Some of the specific fields are instructor related, but the student role uses all of the tables. Most of this will be self explanatory to readers familiar with database design, but one unusual naming convention that has not been factored out of the system is noteworthy: the instructors table holds user information for students, tutors, and librarians.

Another noteworthy feature is the paths table, which keeps track of all of the SCOS the student visits. This information is used in testing my hypothesis for comparing the typical student's path through a set of SCOs to the instructor's intended path. It will also be important when UTS is used to design an ADL course because it will help determine which SCOs and links are most important. At some point in the future with a larger data set it could also be useful for identifying groups of students (e.g. those who

take a certain path through the content) and creating concept maps specific to those student types.

Important Design Details

Module Completion

Each time a user submits a response to an assessment, the system evaluates all of the user's submissions to see if the user has met the completion criteria for a module. Once a user completes the exit criteria, it is reflected in the score component, but they are free to continue studying the module and completing additional assessments.

The system tracks the relationship between assessments using modules and modulepostrequisites tables. The modules table lists course name and ID, modulepostrequisites relates course ID to assessment ID. For simplicity, the system currently only supports student enrollment in one module at a time, but the database is structured to accommodate multiple simultaneous enrollments for future use.

The system calculates the student's completion score based on the most recent submission for each assessment using the following query.

```
SELECT assessmentresponses.aid, MAX(date), response, correct
FROM   enrollment,modulepostrequisites,assessmentresponses
WHERE  enrollment.sid='student' AND
           enrollment.sid=assessmentresponses.uid AND
           enrollment.mid=modulepostrequisites.module AND
           assessmentresponses.aid=modulepostrequisites.aid
GROUP BY aid
```

Path and Event Tracking

To support the experimental goals of UTS, I added a number of features for robust user behavior tracking, including path tracking and event logging. Every time a user navigates from one page to another, an entry is made in the path table. These data

are also required for the forward and back functionality which is non-trivial because UTS is a map, and not a linear path.

Instructor Features

This section describes the UTS features for the tutor and librarian roles. The tutor is the most complicated role in terms of number of features, but was somewhat simpler than the student role to implement because it is less interactive. As shown in Figure 12, the librarian role is quite similar to the tutor—reusing many of the same screens.

Use Cases and Feature Requirements

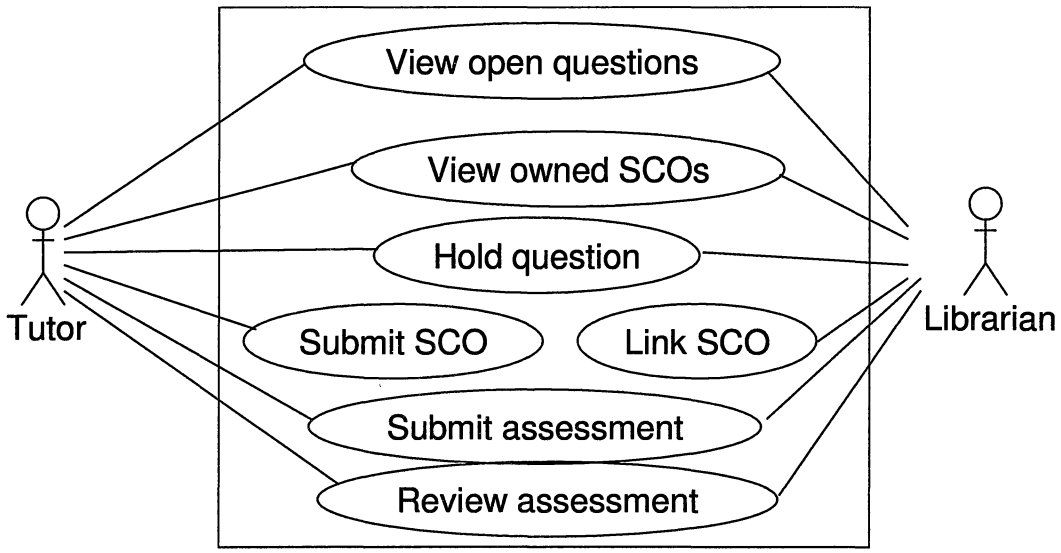

Figure 12: Instructor Use Case Diagram

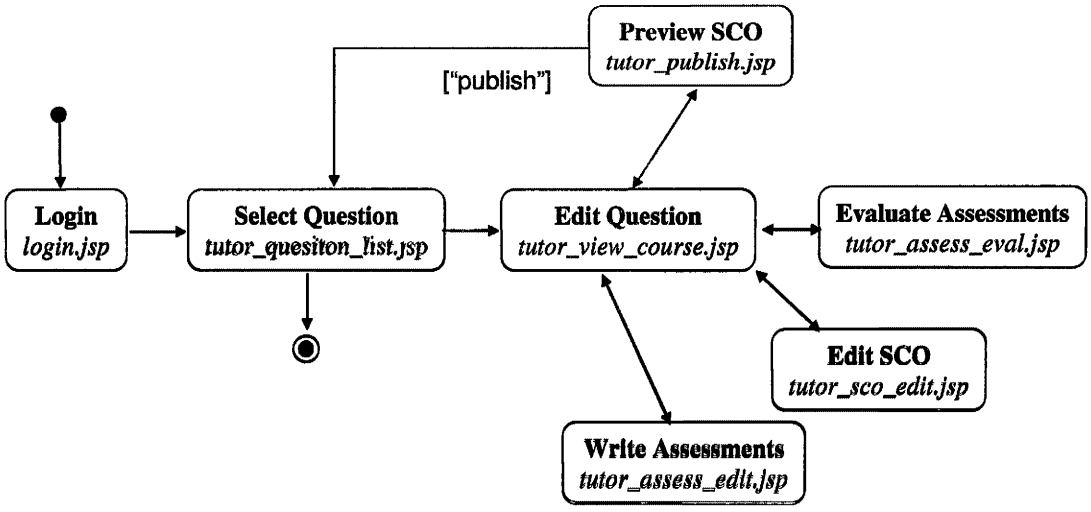

Figure 13: UTS Tutor Activity Diagram for a New Question

Tutors follow the steps outlined in Figure 13 in the UTS process. Upon logging in, the tutor is presented with a list of questions that need to be answered, questions they have already answered, and whether they already have a question open for editing. If the tutor selects a new question to edit, the system creates a new SCO in the database and navigates to an editor page that allows the tutor to edit the student's question. Enabling the instructor to edit the student's question was not an original feature of UTS, but proved necessary because students' questions were often unclear or error ridden.

Once in editing mode, the instructor can navigate freely among pages for editing the SCO content, adding new assessments, characterizing existing assessments, and previewing the target (answer) SCO. After editing the tutor publishes the course, makeing it accessible to students, and allowing the tutor to open a different question for editing. From the publish page, the tutor may also cancel editing. Saved edits are not lost, but other instructors have the option to edit and publish the SCO.

User Interface

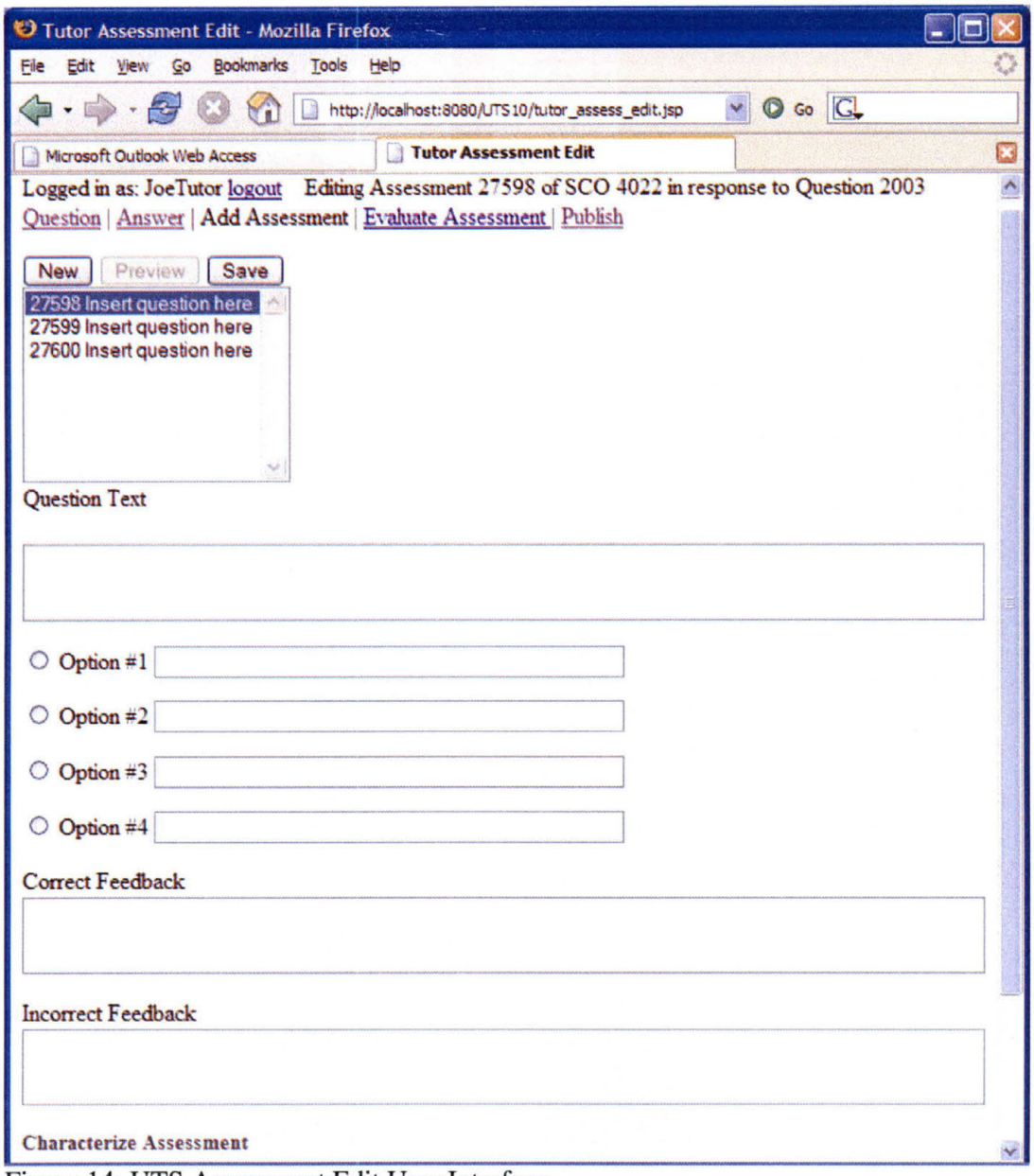

Figure 14: UTS Assessment Edit User Interface

The user interface in Figure 14 is typical of the UTS tutor interface. In this particular example, the tutor edits assessments for the target SCO. Any time a new SCO is created, UTS creates a blank SCO and three blank assessments to go with it. The tutor may choose to create additional assessments as well. This screen does have some client-

side interactivity—when an assessment is selected in the list area, the data for the new form are loaded using Ajax, so there is no reload. The other screens of the instructor interface (not shown) are of similar design and interactivity.

Static Design and Database

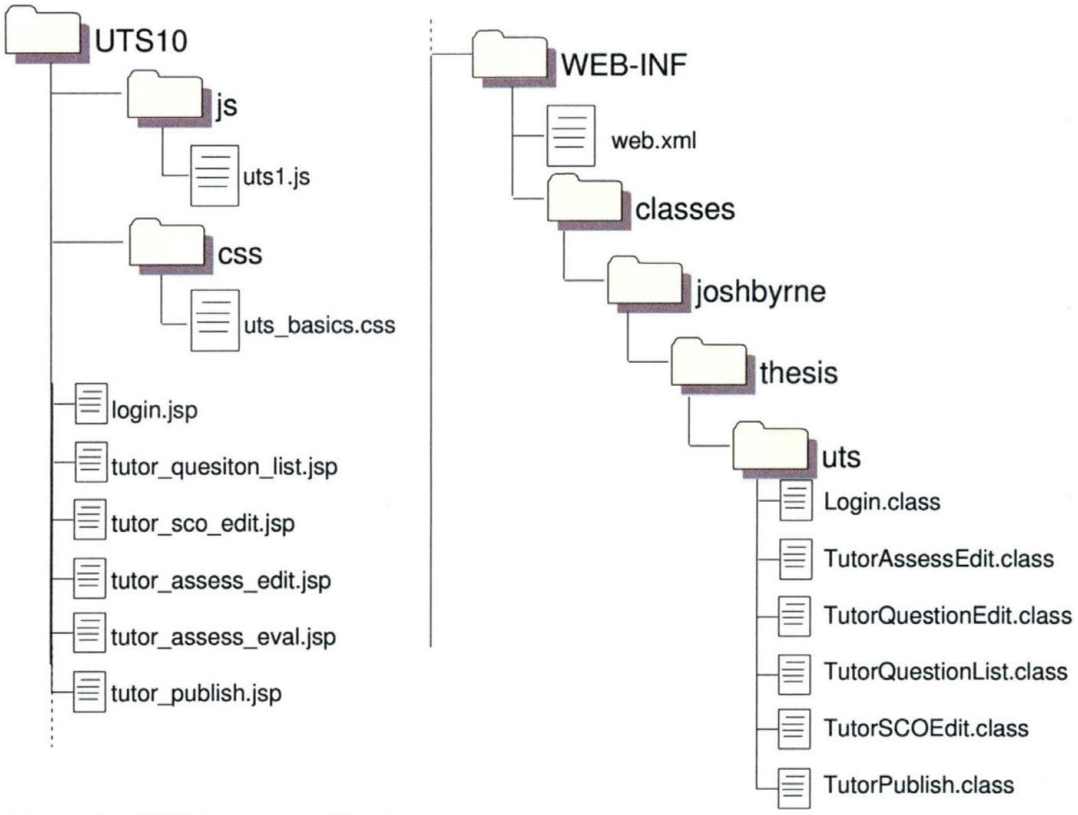

Figure 15: UTS Instructor File Structure

Figure 15 shows the deployment of the instructor role in the file system. Because the instructor interface uses relatively little client side interaction there is a one to one correspondence between JSPs and the servlets that post data for them. The instructor roles use the same database tables as the student role shown in Figure 11.

CHAPTER 4

EXPERIMENTAL RESULTS

This study revolves around two test modules and an experiment–each composed of a Web-based instructor facilitated training (WBIFT) session using UTS, a concept map design, and subsequent data analysis. This chapter maps out the overall approach to the modules, and the goals for each with respect to testing my hypothesis. Chapter 5 provides analysis of results.

The UTS software was developed over the course of this project, so gathering information for its development and improvement was also an important objective of each module. UTS development objectives are also described here.

Common Aspects of the Modules

Test the Basic Hypothesis

The most important objective of all three modules was to test the hypothesis that novices see a knowledge domain in a different way than the expert instructors. My basic premise is that there is cloud of low level propositions that describe the universe as we understand it. While people can argue about the naming and connecting phrases of these concepts, they are essentially fixed from person to person, except at the fringes of knowledge where experts disagree on the fundamental nature of our world.

I contend that in a Web-based self organizing system, assessments created by instructors are a good proxy for these fundamental propositions. That is not to say that a particular assessment corresponds one-to-one with a single proposition, but that experts can have an objective discussion and come to good agreement on the correctness of an assessment. Any given assessment will deal in part with the instructors' abstract notions of the domain, but by comparing a large number of assessments, get a more and more detailed picture of the fundamental proposition cloud.

Similarly, I contend that an effective way to measure the student's abstract conception of the domain is to look at the questions they ask. It is true that this measurement will be colored by the content of the SCOs they are asking questions about, but I believe that influence will diminish with a sufficient number of students. So, one objective of each module is to collect student and instructor data for the purpose of generating a student concept map and comparing it to the concept map designed by instructors.

Create a Concept Map

It is impractical to directly gather student concepts of a domain by asking them to create a concept map, because the process of constructing the map significantly changes their perception of the domain, and because they often lack the vocabulary to express the concepts they do have. Instructors on the other hand have a fairly stable concept of the domains they teach, so simply asking them to design a concept map is a good way to find out how they view the domain.

An important part of the test modules and experiment was obtaining the instructor designed concept map. For test module one and the experiment, the map was designed

by the instructors. For module two, I used a published concept map designed by third party experts.

Start With a Single SCO

Each module starts with a single SCO answering the same focus question as the target concept map. Students are informed that to complete the module they must ask questions and check back for answer SCOs until they learn enough to pass the exit assessment.

In order to bound the content, instructors were asked to submit assessments they think a student should be able to pass after completing the course. Students pass the course by scoring 80% or better on a minimum of ten exit assessments. These exit questions are mixed in with the SCO specific questions, so there is an opportunity for students to demonstrate mastery of the material as they learn it.

I considered an alternative approach where the starting point would be a linear multi-SCO ADL course, but decided that this would bias the derived concept map too strongly towards the instructors' model of the domain, and require a large number of students before the typical path deviated significantly from the initial design.

The chosen approach also guarantees a significant number of questions and answers, and has the potential to generate good path information. The disadvantage is that it requires a short response time from instructors.

Test Module #1: Single User in All Three Roles

Overview

The experimental objective for test module #1 was to generate enough data to test the basic concept map extraction algorithm, and put in place the evaluation procedures. It also creates a baseline for maximum user awareness of the UTS process—how the derived and designed concept maps compare if the instructor is also the student and simply uses UTS as a programmed learning design tool. To the extent that the results show a closer correlation between derived and designed concept maps in test module #1 than in the later experiments, it supports the hypothesis that students organize abstract concepts differently than instructors. It also provides additional support for the hypothesis because it helps to show that any lack of correlation in later experiments is not simply because the concept map extraction methodology is ineffective.

The UTS development goal of test module #1 was to verify the completeness and usability of the student interface. It was also interesting to determine whether UTS could be useful purely as a programmed learning design and authoring tool with the designer acting as both the student and instructor during course development. During the initial stages of test module #1, I manipulated the database directly for the instructor tasks. The experimental process was as follows:

1. Write description, prerequisites, and post requisites
2. Write exit assessments
3. Write introductory SCO
4. Populate UTS
5. Ask and answer questions
6. Author Concept Map
7. Generate Concept Map

Initial SCO Design

The topic of test module #1 was "Using UML Sequence Diagrams in a Design Document." The focus question was "How should a software developer use sequence diagrams in a design document?"

Module Prerequisites

- Basic knowledge of an object oriented programming language
- Understand the basic goals of a design document

Module Post Requisites and Exit Assessment

After completing this module, students will know how and when to apply simple sequence diagrams in a design document. Students will know where to look for authoritative information on sequence diagrams. The following questions formed the exit assessment.

1. A sequence diagram is best for showing:
    a. Database design
    b. Class composition
    c. Recursive functions
    d. Interactions among objects (X)

2. In a sequence diagram, time is shown:
    a. Relative to vertical position (X)
    b. Relative to horizontal position
    c. By numbering function calls
    d. Sequence diagrams are time independent

3. Which is typically not shown on a sequence diagram?
    a. Object
    b. Message
    c. Link (X)
    d. Lifeline

4. A sequence diagram is NOT good for showing:
    a. Interactions between objects
    b. Complex iteration (X)
    c. Time dependent behavior

d. Multiple objects

5. Use a sequence diagram to show
    a. All dynamic aspects of a design
    b. The most important dynamic aspects of a design (X)
    c. The system from a user perspective
    d. A class's members

6. In a sequence diagram, a box around the lifeline shows:
    a. Focus of control (X)
    b. An asynchronous function
    c. Object initializations
    d. A comment

7. In a sequence diagram, a dotted line descending from an object is
    a. A swimlane
    b. The object's life line (X)
    c. A function call
    d. The object's focus of control

8. In a sequence diagram, an asynchronous message is indicated by:
    a. A solid arrow
    b. A stick arrow (X)
    c. A dotted line
    d. A blue arrow

9. Where can you find the latest UML specification
    a. w3c.org
    b. omg.org (X)
    c. ibm.com
    d. microsoft.com

10. Sequence diagrams are defined in what specification
    a. UML (X)
    b. XML
    c. HTML
    d. Java Programming Language

Initial SCO

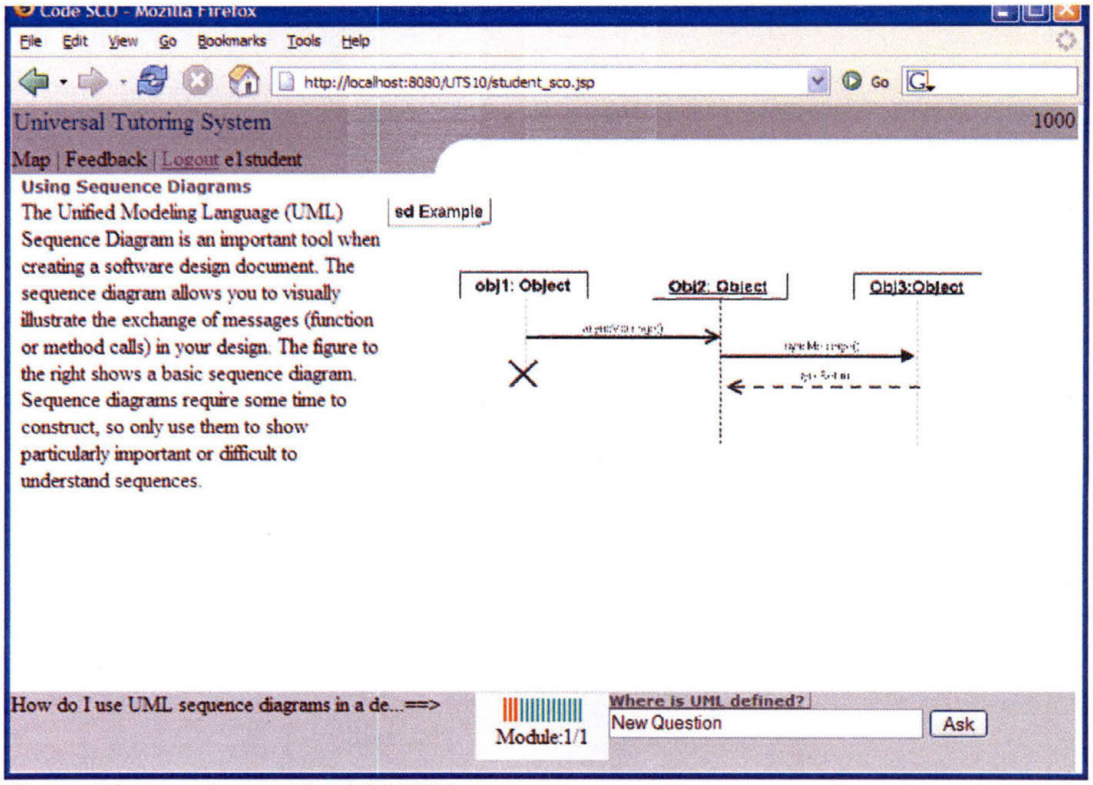

Figure 16: Experiment #1 Initial SCO

Initial SCO Assessments

1. Messages shown in a sequence diagram are often implemented in a programming language as:
    a. Variables
    b. Class definitions
    c. Destructors
    d. Method or function calls (X)

2. Which UML diagram shows the exchange of messages among objects?
    a. Class diagram
    b. Object diagram
    c. Use case diagram
    d. Sequence Diagram (X)

3. Why not create a sequence diagram for an entire program?
    a. Sequence diagrams are expensive to create (X)
    b. Sequence diagrams are not useful for modeling software
    c. The exchange of messages is not important to document
    d. A visual approach is more appropriate

Interactive Content Generation

After posting the exit assessments and first SCO, I alternated in the role of student, tutor, and librarian—asking and answering questions. The following are observations made during test module #1.

Observations

It quickly became obvious that it was difficult to avoid adding very similar questions, even with a single instructor and a single student. I ultimately decided that there wasn't a good way to avoid this in small scale use of UTS. With a sufficient number of students, it will be possible to eliminate questions that always evaluate the same against a given set of SCOs.

One of the first observations on UTS in Experiment #1 was that it did not properly handle navigation related to the initial SCO. The design and code were modified so that when a new module is started, the focus question is added as a question in UTS, and set to refer to the root SCO as both its source and target. The user interface was updated to inactivate the back link for self-referential questions.

At the start of test module #1, "exit" status for assessments was still just a flag in the assessments table. This limited UTS to a single module, so I added a lookup table so that each module could have different exit assessments.

After completing only two or three SCOs as a student, it became obvious that having the assessments on a separate page from the SCO content was unwieldy. This observation inspired the first use of Ajax in the user interface for the assessment popup shown in Figure 9. Test course #1 also drove the first of several upgrades of the bottom

navigation bar with the arrows being removed, and the module score added to the assessment component.

Test Module #2: Separate Student and Instructor

Overview

The experimental objective for test module #2 was to determine the differences in generated concept map when the student and tutor are different individuals. The UTS development goal was to verify the completeness and usability of the tutor interface. For this test I continued to play both instructor roles, but did so using the UTS interface where possible. Four other people were the students in this test.

The UTS evaluation goal was to further test the student interface and provide information for the design of the tutor and librarian interfaces. The experimental process was as follows:

1. Write description, prerequisites, and post requisites
2. Write exit assessments
3. Write introductory SCO
4. Populate UTS
5. Ask and answer questions
6. Author Concept Map
7. Generate Concept Map

Initial SCO Design

For the second test module, I chose concept maps as a topic, and most of the students were people who had been chosen as instructors for the final experiment. This met the dual goals of further testing the student interface and preparing the instructors for the experiment. Rather than create my own concept map for this experiment, I used the one previously published by Novak (2006), who invented concept maps. The topic and

focus question (also from Novak) were "Introduction to Concept Mapping" and "What is a concept map?"

Module Post Requisites and Assessment

After completing this module, students will be able to use concept maps to explain a knowledge domain. For this module I used the following exit assessments, one for each concept on the designed concept map.

1. Which of the following is not a good use of a concept map?
    a. Graphically illustrate relationships between concepts
    b. Replace a flow chart (X)
    c. Assist in learning a new idea
    d. Assist in teaching a new idea

2. Who invented concept maps?
    a. Robert Novak
    b. Joseph Novak (X)
    c. Robert Joseph
    d. Joseph Roberts

3. Which of the following are not used to label concepts in a concept map?
    a. Symbols
    b. Words
    c. Pictures
    d. Linking Words (X)

4. In a concept map, what is a concept?
    a. A perceived pattern (X)
    b. A relationship between two things
    c. A place on a map
    d. An innovative idea

5. How should concept maps be structured?
    a. Radially
    b. As a taxonomy
    c. Randomly
    d. Hierarchically (X)

6. What does a concept map show?
    a. Effective learning
    b. Effective teaching
    c. Organized knowledge (X)

d. A geographic region

7. Which is a benefit of a concept map?
    a. Organizes knowledge without respect to context
    b. Aids creativity (X)
    c. Supports rote learning
    d. Useful for modeling large knowledge domains

8. In terms of a concept map, what is a proposition?
    a. An event or object
    b. A concept map created by an expert
    c. Two or more concepts connected by linking words (X)
    d. The starting point for a new concept map

9. What is a crosslink on a concept map?
    a. An interrelationship between different map segments (X)
    b. Any linking words
    c. A symbol in a concept icon
    d. A perceived regularity or pattern

10. How is organized knowledge useful?
    a. enables effective learning
    b. enables effective teaching
    c. enables both effective learning and effective teaching (X)
    d. it can represent a concept map

11. When learning to use concept maps, start with
    a. a large number of concepts
    b. a familiar domain (X)
    c. four or five linking words
    d. a new concept

12. Concept maps are most effective when
    a. created by learners
    b. crosslinks are created first
    c. used to answer broad questions
    d. created in a particular context (X)

13. What is used to identify new interrelationships on a concept map?
    a. Focus questions
    b. Creativity (X)
    c. Inference
    d. Iteration

First SCO

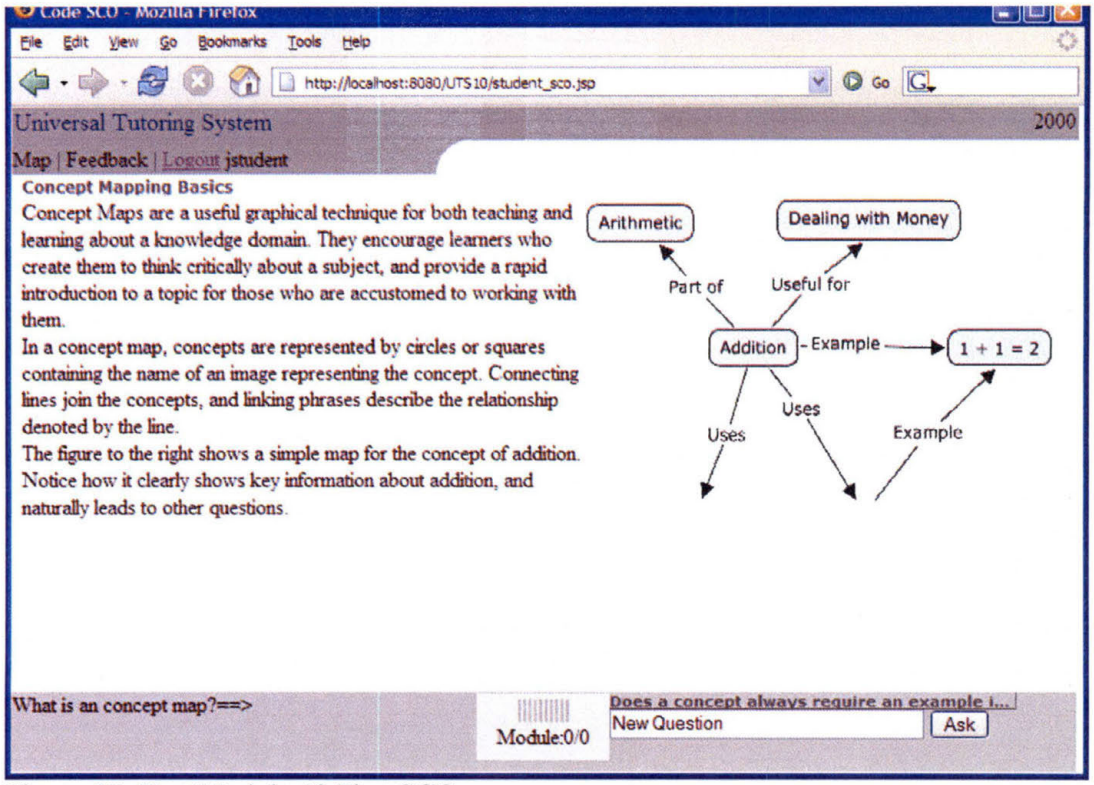

Figure 17: Test Module #2 First SCO

First SCO Assessment

1. Concept maps show which of the following
    a. Concepts, connecting lines, and linking phrases (X)
    b. Concepts, mathematical expressions, and linking phrases
    c. Concepts, domains, and connecting lines
    d. Concepts, domains, and linking phrases

2. Concepts maps model
    a. A knowledge domain (X)
    b. A process
    c. A computer program
    d. Questions

3. Lines in a concept map show
    a. Limits of the map
    b. Geographic regions
    c. Important concepts
    d. A relationship between concepts (X)

Interactive Content Generation

After posting the exit assessments and first SCO, the UTS process was explained to several students. As they posted questions, I answered them either as a tutor or as a librarian, depending on the question. This was the first test of the instructor interface, so a significant amount of development accompanied this test module.

Concept Map

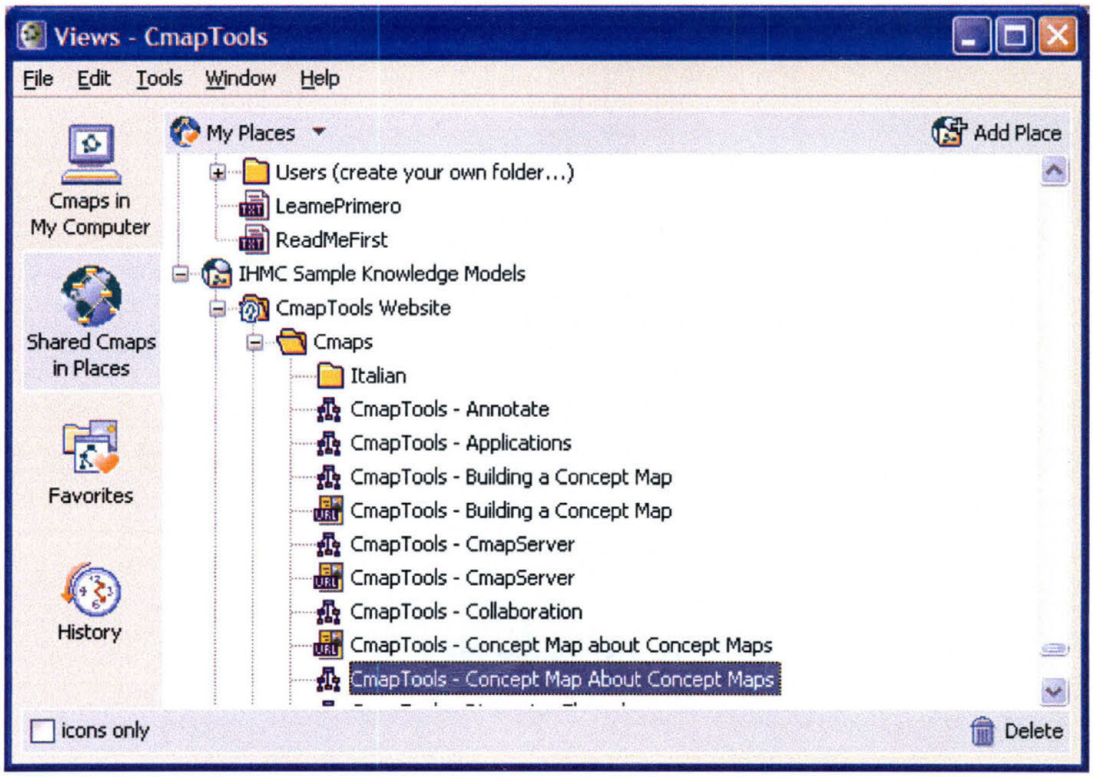

Figure 18: Location of Novak's "What is a Cmap" Concept Map in the CmapTools (CmapTools, 2007) Application

Rather than create a new concept map, I used the "What is a Cmap" concept map by Novak. A version of this concept map is available in published IHMC reports (Novak

and Cañas, 2006), and in electronic form using the CmapTools application. Figure 18 shows the location of this concept map within CmapTools.

Observations

Writing good multiple choice questions remains a challenge. Any large scale UTS deployment would need an incentive mechanism to encourage instructors to write them. Chapter 6 also discusses the possibility of using multiple select questions. These are easier to write and typically more challenging for students. They also hold more information for concept map extraction, if a system for easily evaluating them could be developed. Adding a wider variety of question types and supporting commercial assessment authoring tools would also make writing questions easier.

As soon as there were actual students in the course, it became obvious that UTS needed a mechanism to handle questions that were based on errors in the SCO, comments about UTS, and other topics that needed to be addressed but were of little value to other students. Based on this observation, I added a "do not publish" option for the instructor to the UTS requirements. Future versions of UTS will also have e-mail integration, so the student who posted a question will receive a reply, but their question will not be available as a link to other students.

Experiment: Large Student Data Set

Overview

The primary objective of this experiment was to evaluate the generated concept map in a scenario that approximates "real-world" use of UTS and in so doing answer the basic scientific question of the study—how does a concept map designed by

knowledgeable instructors compare to one derived automatically? The UTS development goal was to verify the completeness and usability of UTS in a real-world situation with multiple instructors, and more than ten students. The experimental process was as follows:

1. Write description, prerequisites, and post requisites
2. Write exit assessments
3. Write introductory SCO
4. Design concept map
5. Populate UTS
6. Ask and answer questions
7. Author Concept Map
8. Generate Concept Map

Initial SCO Design

After considering many options, I chose "binary trees" as the topic for the experiment. I considered using a non-computer science topic, but ultimately decided that having a good selection of instructors and students was more important.

Module Prerequisites

Basic knowledge of an object oriented programming language is required.

Module Post Requisites and Exit Assessment

After completing this module, students will know how to construct and manipulate a binary tree, and when it is appropriate to use one. Students must successfully answer 80% of at least ten module post requisite questions to pass the module. The system evaluates only the student's most recent attempt when determining module completion.

1. What is the most child nodes a parent can have in a binary tree?
   a. Zero
   b. One
   c. Two (X)
   d. Unlimited

2. What are nodes with the same parent referred to as?
   a. Children
   b. Siblings (X)
   c. Left-Right Children
   d. Sub-nodes

3. A leaf node has how many children
   a. Zero (X)
   b. One
   c. Two
   d. Unlimited

4. What Java structure can be used to store a heap?
   a. java.lang.Integer
   b. java.util.Heap
   c. int
   d. int[] (X)

5. Which is not an application of a binary tree?
   a. Company organizational structure (X)
   b. Sorting
   c. Heap
   d. Yes/No decision tree

6. After inserting a value to the bottom of the heap, what process turns a binary tree into a heap?
   a. Reheapification (X)
   b. Reorder
   c. Sort
   d. Removing the root node

7. Which of these is not a feature of a heap?
   a. A heap is balanced
   b. All nodes have two children (X)
   c. At the deepest level the elements are sifted to the left
   d. A node's value may not be less than either of its children

8. Which is the top node of the tree?
   a. Parent
   b. Head
   c. Heap
   d. Root (X)

9. Heap sort works because?
   a. The root node is always largest (X)

b. Every parent has the same number of children
c. The depth of the tree is predictable based on the value of the root node
d. The leaf nodes at the lowest depth are largest

10. After you remove the root of a heap, which element should be moved to the root in the first step of reheapifying?
a. The node with the highest value
b. The node that was the left-most child to the former root
c. The node that was the right-most child to the former root
d. The last element of the binary tree (X)

11. Why do many algorithms call for removing the first node of the heap?
a. Because it is the node with the lowest value
b. Because it is the only node with more than two children
c. Because it is the only node whose relationship to the others is completely known (X)
d. Because its value can not be the highest

First SCO

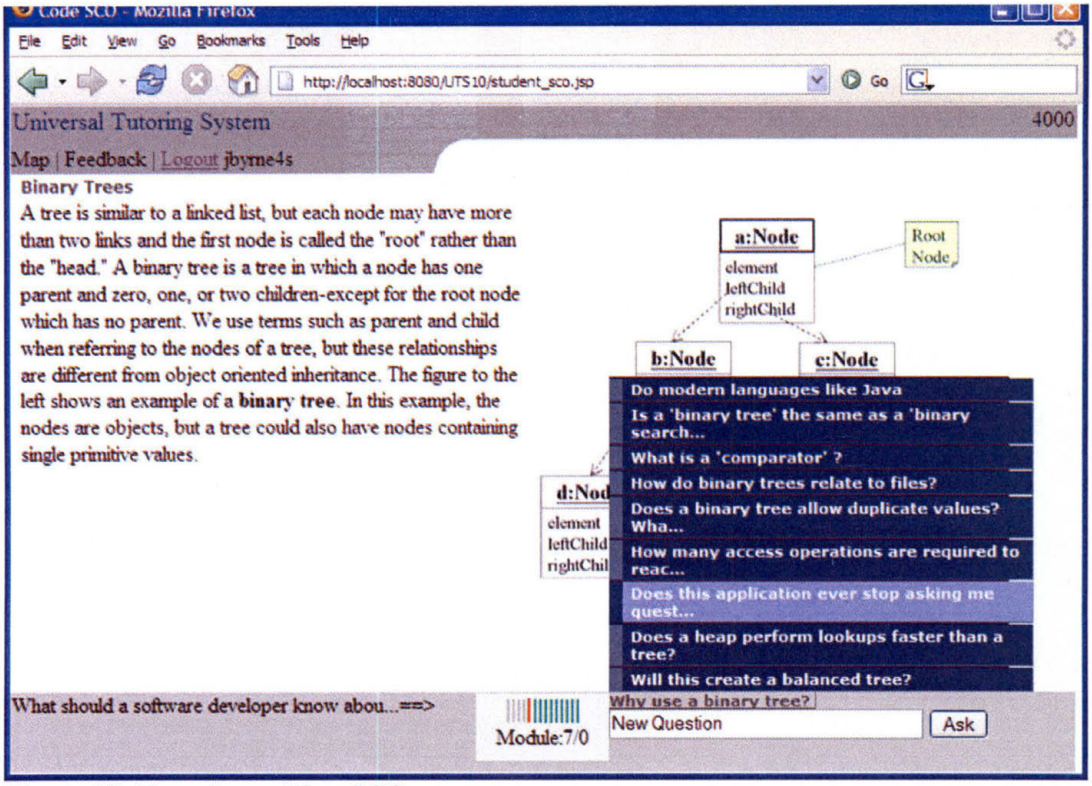

Figure 19: Experiment First SCO

Figure 19 shows the initial SCO for the experiment. It also shows the question roll over that gives the student a list of available questions.

First SCO Assessment

1. How is a tree different from a linked list?
    a. It doesn't have a "first" node
    b. A node can have two children instead of one (X)
    c. There are no links
    d. A tree can only contain primitive values

2. How is a binary tree different from other trees?
    a. Its nodes hold Boolean values
    b. Its nodes hold primitive values
    c. Its root node has a parent
    d. Each node has a maximum of two children (X)

3. Which node of a tree has no parent?
    a. Leaf node

b. Head node
c. Root node (X)
d. Binary node

Interactive Content Generation

For this experiment, students were introduced to the system 2 and 3 at a time over a number of days to avoid overwhelming the instructors and frustrating the students because their questions were not answered fast enough. I also played the instructor role in a limited capacity, but primarily for the purpose of handling exceptions and keeping the process moving. For example, there were a number of questions about UTS itself, and I created some "help" SCOs to explain the system. There were also several situations where changes that were not accommodated by the instructor interface needed to be made directly in the database.

Concept Map

To generate the concept map we used the Cmap software (CmapTools, 2007) and followed the approach suggested by Novak. (Novak and Cañas, 2006) The instructors chose the focus question "what should software developers know about the binary tree data structure?" Next, we created a "parking lot"—a list of concepts that are likely to be on the map. These concepts were then organized into propositions during an interactive online session. The concept map in Figure 20 is the result of that session.

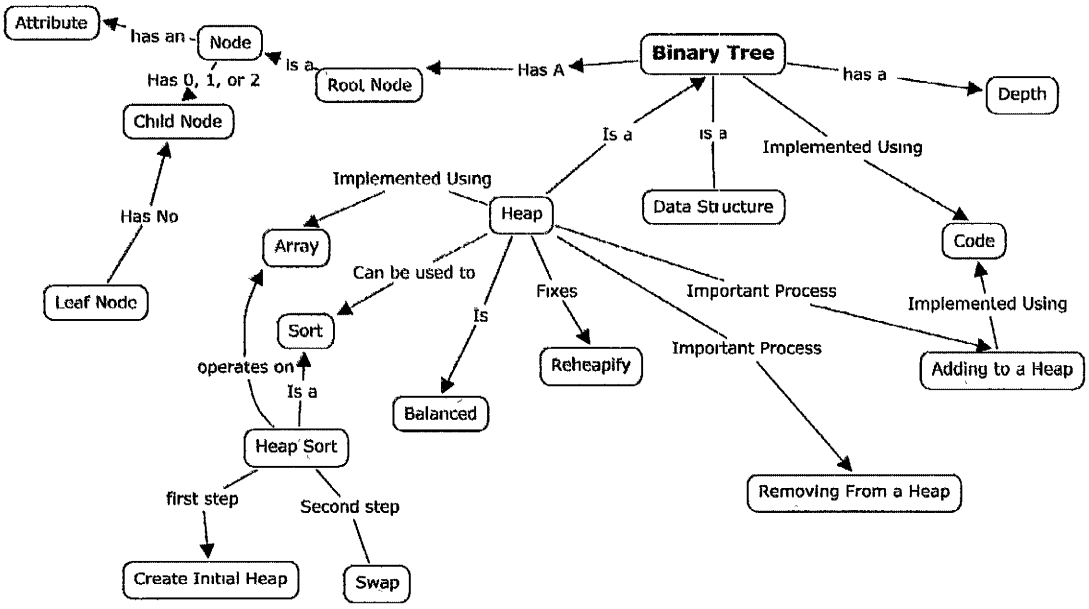

Figure 20: Binary Tree Concept Map for Experiment

Observations

The initial SCO in Figure 20 is a good example of the power of the UTS and of intelligent tutoring. In a conventional course, the instructional designer would need to worry about whether the students had a prerequisite such as knowing what a linked list is. In the experiment, students who didn't know asked, and that information was added to the course.

With the expanded student group in the experiment it quickly became clear that the ability for the instructor to rephrase student questions is critical; many questions were too long, incomplete, or unclear. This feature was added to UTS during the course of the experiment.

It was also observed that student questions quickly went outside of the specified scope of the module. It only took three or four questions to go from binary trees to obscure details of file system design. Over the long term, this is a good thing because

one of the goals of UTS is that new modules will be spontaneously generated as students follow lines of inquiry they are interested in. To avoid distracting instructors from the focus question, instructors were asked to simply delay a couple of days in answering questions that were outside the initial scope. This proved an effective technique because by the time the question was answered, it was mixed in with a lot of other questions, and few students selected it.

The initial design for UTS envisioned an interactive help function for students who had trouble using the system. However, I found that simply allowing students to submit help questions in the same way they do content questions was effective. It kept the interface simple, and once the help SCOs were developed, the questions were easy for the librarians to answer. In future applications of the UTS methodology, I plan to start with an "about UTS" SCO with the focus question linking it to the initial content SCO.

Requiring instructors to evaluate three assessments when they created or linked a SCO proved to be insufficient. To generate a useful data set, we had to work through a complete matrix of assessments versus SCOs. For future development a complete data set might not be necessary, but more than three evaluations per SCO will be.

CHAPTER 5

ANALYSIS AND CONCLUSIONS

Derived Concept Map Algorithm

This section describes the algorithms used to extract concept maps from student behavior and the results of applying it to the experimental data. The algorithms I examined has several basic premises. First, it is assumed that each concept relates to one or more SCOs. Second a relationship exists between two concepts exists when the first concept relates to a SCO that is the subject of a question, and the second concept relates to the SCO that answers that question. A relationship is also presumed to exist between concepts that relate to the same SCO. Each SCO has a set of pre and post requisite assessments. Likewise, each assessment is prerequisite, post requisite, or unrelated to a given SCO.

Algorithm Step #1: Unique Set of Pre and Post Assessments

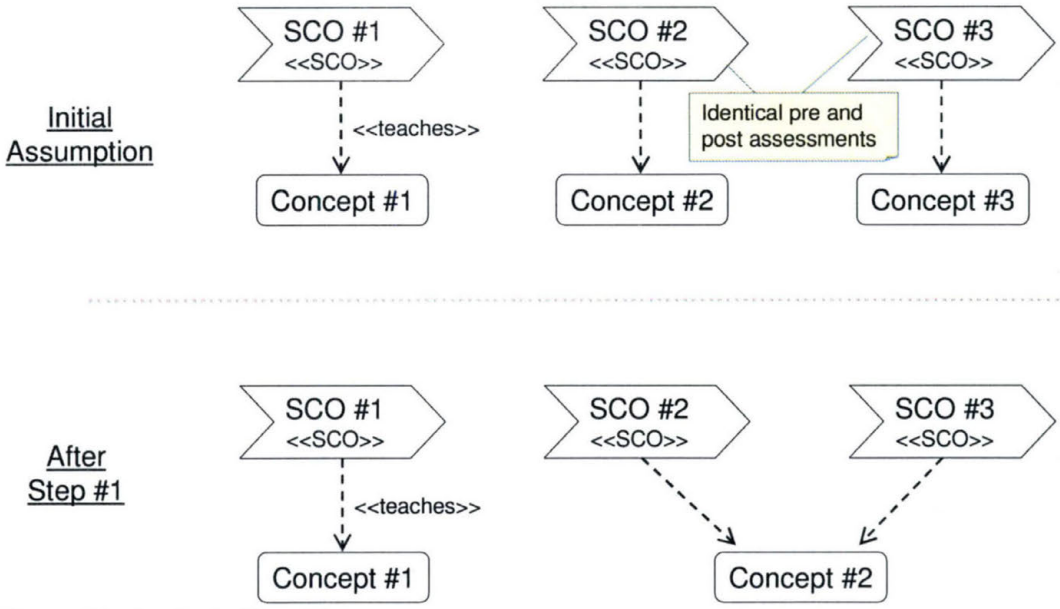

Figure 21: Analysis Step #1

This algorithm defines a concept for each unique set of pre and post assessments, and a connecting line for each question. In practice, this is implemented by starting with a concept corresponding to each SCO. If two concepts have the same pre and post requisites, they are merged because requisites of one concept are the same as another, the two are deemed to teach the same underlying concept.

In practice, I found the set of assessment to SCO relationships generated by UTS too sparse to be useful for this step, so I reviewed the assessments and SCOs with the instructors outside of UTS and generated a complete set of data for the relationship between all SCOs and all assessments. This technique provided useful data, but is problematic for larger experiments—for this experiment we evaluated more than two thousand SCO and assessment pairs for approximately twenty-five SCOs and a hundred assessments.

In order to improve the efficiency of the algorithm for comparing the (ordered) sets of assessment relationships for each SCO, I first summed the values of the index numbers corresponding to the relationships, and only compared those columns with the same value. In itself, this step was not a good discriminator for the data set I had; as there were never any SCOs combined in this way. Its potential effectiveness is likely to diminish as the data set grows.

Algorithm Step #2: Similarity Threshold

The next step was to extend the algorithm from step one by recognizing that there is likely to be some variation in any set of experimental data, and allowing for a similarity threshold. I defined similarity threshold as a percentage of items in the sets (SCO to assessment relationships) that could be different, and still consider the corresponding SCOs equivalent. The data was evaluated for similarity thresholds of 10% and 20%. At a 10% threshold, only two percent of SCOs were deemed equivalent. At a 20% threshold twelve percent of SCOs were deemed equivalent.

I took this as strong evidence that few if any of the SCOs in my experiment covered the same exact set of concepts, because the number of concepts in the derived map was only slightly reduced by this algorithm. At the same time, I demonstrated that the number of concepts can be reduced using this algorithm. It proved useful for recombining SCOs after step #3, and will likely become more effective as the number of SCOs grows.

Algorithm Step #3: Divide At Question

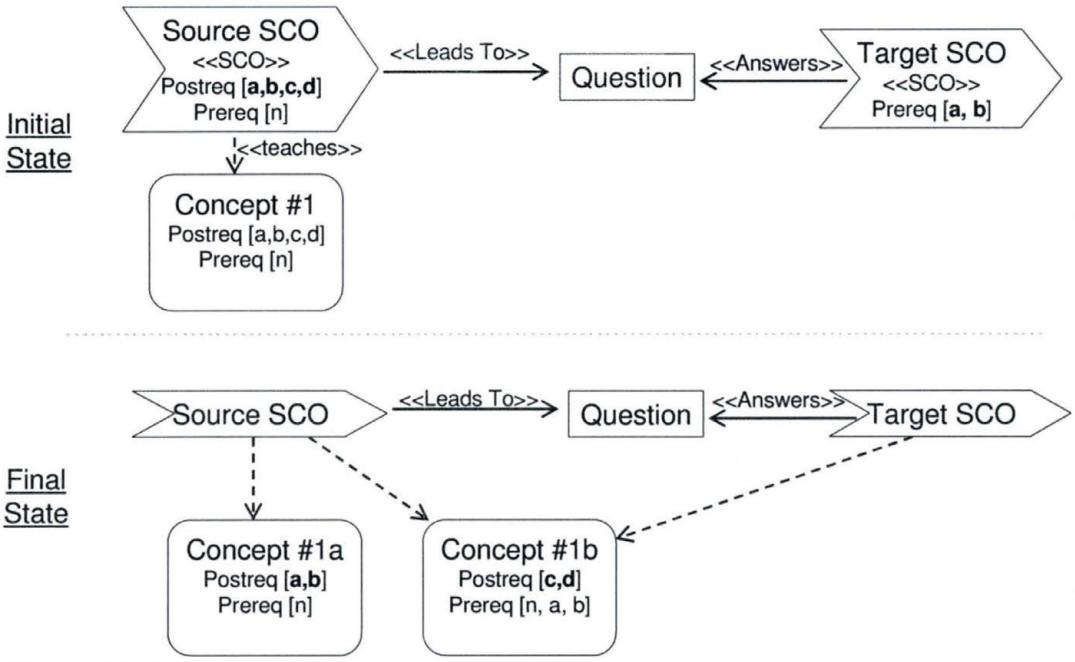

Figure 22: Analysis Step #3

Step #3 addresses the possibility that a concept will be divided when a question is asked from the corresponding SCO. When assessments are post requisite to a subject SCO but prerequisite to the target SCO, I deduce that there is a concept learners could learn from the subject SCO before linking to the target SCO. Therefore, the concept corresponding to the subject SCO is divided into two concepts. Both have the same prerequisites, but one has the post requisites that are prerequisite to the target SCO, and the other has the remainder of the post requisites.

This analysis yielded a good number of opportunities to break the concepts linked to SCOs into multiple concepts. After splitting off the new concepts, they were again compared for recombination using step #3. After division there were twelve new concepts, but they subsequently recombined back to just six concepts. This is excellent

preliminary evidence that the algorithm is effective, and won't simply create an unlimited number of new concepts. For example, qualitative analysis of the results from the experiment very clearly show that one of the concepts that was separated by questions, and subsequently recombined into a single concept was the basic definition of a binary tree. Intuitively this makes sense—even if a students doesn't understand everything in the root SCO, they should grasp the idea that a binary tree is a tree where a node can have no more than two children.

Concept Map Analysis

Quantitative Comparison to Expert Concept Map

To measure the similarity between the derived and extracted concept maps I evaluated each concept in the designed map against each assessment question as prerequisite, post-requisite, or unrelated. Concepts which had 80% or greater similarity were deemed to represent the same abstract concept, with similarity defined as the number of assessments with the same relationship divided by the total number of assessments.

Unfortunately, the comparison of concepts from the concept map to assessments yielded very few pre and post requisite relationships. This seemed to be because the nodes of the designed concept maps are too granular, and the assessments tend to involve multiple concepts. Consider the question "What is the most child nodes a parent can have in a binary tree?" When evaluated against the concept "node" there is clearly some relationship but one can neither say that someone should be able to answer the assessment before they can understand the concept of "node," nor is it correct to say that

someone who understands "node" could necessarily answer the question. The most useful observation from the expert concept map analysis was that even instructors seem to take a much broader view of a domain when constructing a concept map, as compared to designing a linear course.

Qualitative Comparison to Expert Concept Map

Because the differences in level of abstraction made it difficult to compare the student and instructor concepts maps on the basis of prerequisites and post requisites, I also applied a second analytical approach composed of three steps. In the first step, I compared SCOs generated in UTS to the instructor designed concept map by identifying each concept taught by each SCO.

In the second step I used the path data for each student—the data showing the order in which they studied the SCOs—to determine the order in which they navigated the SCOs. Because most students visited a number of SCOs multiple times, I simplified this analysis by only considering the first time a student visited a SCO. The sequence data for each concept was then averaged to develop a picture of how the average student approaches this domain.

In the final step, I cross-referenced the results of steps one and two to determine the order in which students studied the concepts. I also performed a similar analysis on an existing linear lesson on binary trees (Byrne and Alfveby, 2005) in order to shed further light on the relationship between student and instructor chosen sequencing. Figure 23 shows the result of this analysis mapped onto the instructor designed concept map.

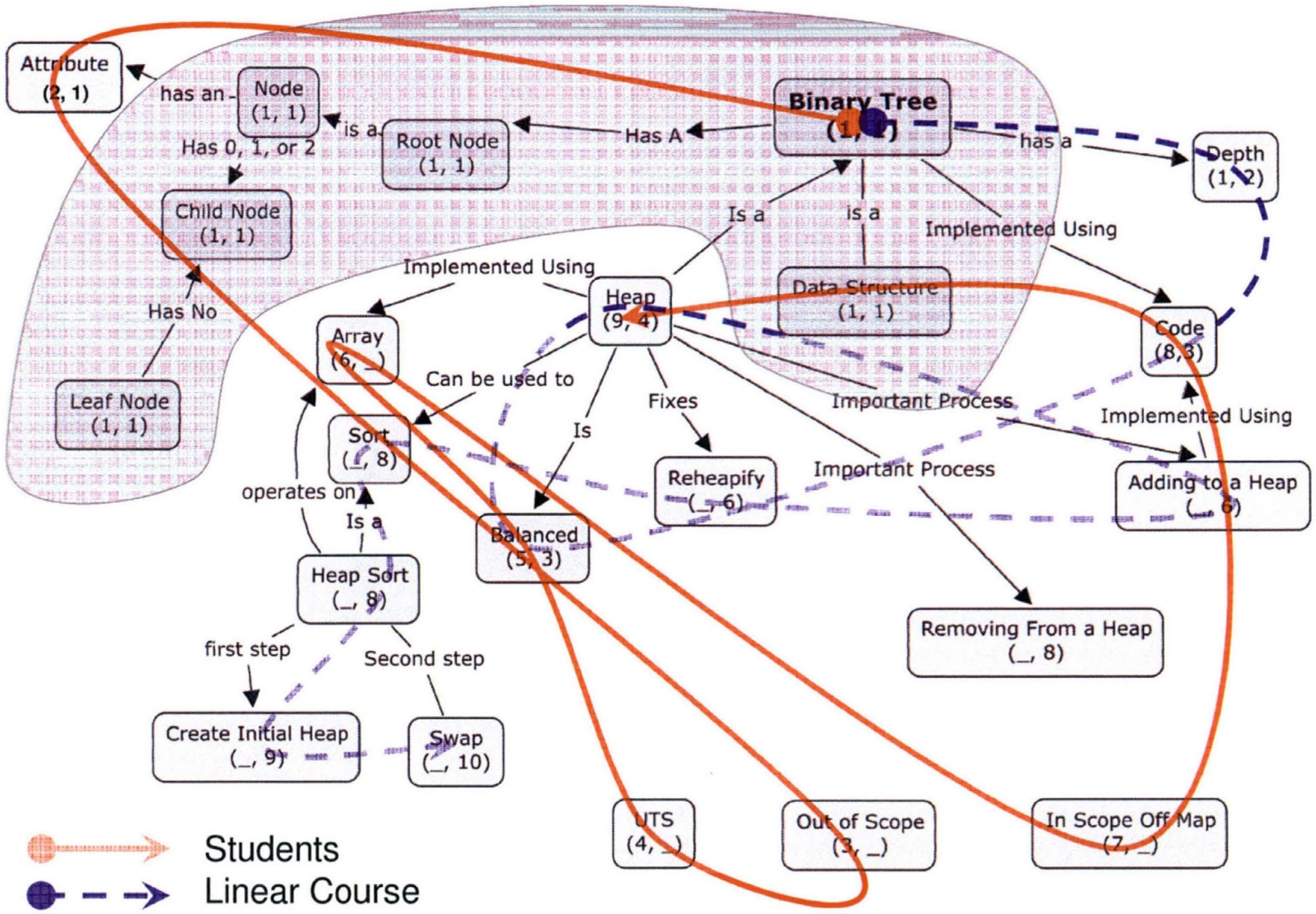

Figure 23: Student and Instructor Paths Overlaid on Instructor Designed Concept Map

The pair of numbers appended to each concept is the sequence number of that concept in the student and instructor paths respectively. For example, the idea of a balanced tree was covered fifth (on average) by students, but was taught third in the sample linear course.

The initial SCO students were required to start with was very similar to the first section of the instructor sequenced lesson, so the first concepts in the sequence are also very similar. The grey area of Figure 23 highlights these concepts.

The bottom three concepts are not on the original instructor designed map, but represent additional information about the student path. The first is for information about UTS, and the experiment. The typical student asked about this fairly early on. The second additional concept encapsulates all concepts that are out of scope of the lesson. This analysis shows that the typical student went very quickly off topic (at least in part). Remember that these are just concepts, and some SCOs taught both in and out of scope concepts. Finally I added a concept for topics that were in scope, but not on the concept map designed by the instructors. Many of these were additional binary tree application examples.

There are several notable differences between the student and instructor paths through the concept map. The first is early and frequently the students went outside the instructor's concept map. This isn't altogether surprising because students exhibit the same behavior in a classroom, asking questions that go beyond the prepared material or about administrative aspects of the class. The main difference in the WBIFT is that it

provides the opportunity to go into greater depth on these questions for small groups of students without using the time of the class as a whole.

The second notable difference is that the instructors turned to code examples much earlier than students asked for them, and the instructor concept map goes into more detail about code than students ever get to through their own questions. This may be in part due to UTS's focus on multiple choice assessments, which do not lend themselves to code related questions. However, it may also reflect the fact that instructors find it important for students to learn to code, but students do not find it intuitive to deduce concepts from the underlying code.

Finally, the instructor path seems to the expert eye to follow a much more logical progression that the student path. The instructor goes from a high level explanation, to more detail, to an example. The student path seems much more fractured. Additional study will be required to determine if this simply reflects the student's lack of understanding of the big picture, or if allowing students to choose their own path can enhance the learning experience by helping them learn more thoroughly, more quickly, or more comfortably.

Summary of Findings

Over the course of the experiment, I compared the way students organize knowledge as measured by their questions in a WBIFT system to the way instructors organized knowledge as measured by their design of concept maps and of courses. In comparing the derived concept maps to designed concept maps and to an existing course,

the key finding was that the domain is much larger than the instructor captures in either a designed concept map or a traditional linear ADL course.

The results of the concept map generation algorithm are also promising, clearly delineating concepts that were identifiable when the questions involved were examined. Unfortunately, achieving that level of result required me to gather data outside of the UTS process, in a way that isn't scalable. It took almost six hours to evaluate the approximately two thousand relationships to produce the data described here. The UTS methodology shows real promise for concept map generation, but won't be scalable until a technique for easily evaluating a much larger number or relationships is developed. I believe such a method is possible because the relationship sets tend to be fairly sparse—most relationships were "unrelated." Chapter 6 provides some ideas for additional research that could shed light on this, including the idea that any future experimentation will need to address the problem of how to characterize a very large set of assessments against a large number of SCOs.

Analyzing the experimental data by mapping instructor and student paths to the concepts by qualitatively correlating the concepts to SCOs yielded more information about the core hypothesis. It showed that the student path is very different from the instructor path in some ways that were expected (e.g. going off topic, and taking a more erratic path through the concepts). It also showed tendencies that merit further study (e.g. lack of student interest in code examples early in the learning process).

In conclusion, the experiments supported my hypothesis that "the structure of knowledge as seen by an expert does not represent the order in which novices naturally learn that material," but the data were not robust enough to make any firm conclusions. I

did raise some interesting questions, and verify the usefulness of several analysis techniques. With some enhancements to UTS, and a much larger data set, I believe that my approach could lead to a number of useful conclusions about the way students learn.

With respect to UTS, I found that even in its relatively primitive state, it is a useful ADL design tool because it expands the instructors view of what concepts need to be covered, and gives immediate feedback on the questions students will have about a piece of content before a lot of effort has been expended on it and it is difficult to change. It also provides a useful way for a group of instructors to collectively address a domain much more thoroughly than they could alone. A key goal for future research should be to determine whether this ultimately speeds or slows student learning of the domain.

CHAPTER 6

FUTURE STUDY

This chapter outlines questions that merit further study, but are beyond the scope of this work. Some are directly related to UTS and concept map generation. Others are broader observations on learning technology

UTS Software Features

This section describes recommended design and feature changes to UTS based on observations during the three experiments. These are significant changes that will require thought, effort, and in some cases experimentation. The UTS source code also includes many smaller suggestions in comments starting with "TODO." Developers using the Eclipse IDE can automatically generate a consolidated list of these suggestions.

Standards Refactor

In order to have full control over UTS, I built some functionality that is also available off-the-shelf in a more robust and functional, if less flexible way. With my experiment completed, the way forward for UTS is to focus on its core business logic and a few aspects of instructional design, and start integrating third party components for less unique functionality. The first step in this process should be for UTS to accept SCORM and AICC compliant content packages. This will allow instructors to upload content

from almost any open source or commercial ADL authoring tool, leading to much more interactive content on UTS.

The next step will be to configure UTS as a SCORM and AICC capable repository. This will enable anyone with a learning management system to deploy UTS courses to their students, while retaining the enterprise class user management and portal features of the LMS. As mentioned earlier, this is a challenge with SCORM due to cross domain scripting issues, but they can be addressed by co-installation with the LMS or by use of a proxy server.

My analysis was performed in Microsoft Excel, but it would be useful to implement it in UTS, and provide for output to a standard ontology description language such as OWL (Heflin, 2004) (McGuiness and van Harmelen, 2004).

Finally, the student and instructor parts of UTS should be extended to accept and deliver QTILite standard assessments. This would enable UTS to support questions from a number of commercial and open course assessment authoring tools, which are sometimes separate from ADL authoring tools.

Assessment Component and Bottom Navigation Effectiveness

The assessment component still requires explanation when new students start a course. This needs to be refactored to be more intuitive. The assessment component is also not very robust in handling longer questions or different question types. It should be extended on the instructor side to accept QTILite standard questions, as mentioned above.

The entire visual metaphor for the bottom navigation needs to be redesigned to make it more intuitive. The typical student should recognize by looking at it that the

back link is the question that was asked from the previous SCO to get to the current SCO, and that the forward links are questions other students have asked about the current SCO.

How Much to Keep

In most cases, it is undesirable to keep all of the material added to a course during the UTS process unless the final course will have a robust question ranking algorithm. This is especially true when UTS results are extracted to create an ADL course. Presumably, it is desirable to retain the paths through the content that are chosen by the most students, but further research is required to determine how much content can be removed without significantly diminishing the effectiveness of UTS.

Exit Assessment Balance

With a well designed initial exit assessment, there is a balance of questions over the topics in the module. As instructors add new questions and mark them as exit assessments, the exit assessment may become heavily weighted towards a small number of topics. A technique needs to be developed to ensure that exit assessments delivered to students are balanced across concepts.

Number of Choices

The current version of UTS randomly selects the links presented to a user. This has the advantage of encouraging exploration, but is probably not optimal for moving students quickly to completion of a module.

Oguejiofor (2004) suggests a continuum from strictly scripted courses where the user must view SCOs in a particular sequence to entirely free form courses where the SCOs may be viewed in any order. The UTS POC is closer to the latter, but students may

be more comfortable choosing from a smaller number of carefully selected options. This is a question that requires further research to answer. Another possibility is to rank the question links by the shortest path through SCOs that teach the remaining exit assessments.

UTS Cloud

Concept map developers talk of a "cloud" or "soup"—a large collection of relations, often formed by integrating multiple concept maps. UTS has the potential to form such a cloud if multiple groups adopt it and start building content from different focus questions. Integrating these sites will be reasonably straight forward in a purely open content model (all instructors license their content for free use). However, if ownership of the content is restricted, the problem becomes more difficult, and a federation model is more appropriate.

UTS Process and Applications

Algorithmic Assessment Relationship Determination

Is it possible or desirable to eliminate the need for instructors to specify the relationship between assessments and SCOs? On one hand, such a change would require less work of instructors, allowing them to focus more on content and driving broader acceptance of the UTS methodology. On the other hand, automating this process might be critical to scaling UTS because instructor evaluation alone cannot lead to enough evaluations for the analysis to work properly.

The need for instructors to evaluate the relationship between SCOs and assessments could be eliminated in a system with a large number of students by

evaluating what students know before and after a SCO, and deducing that the SCO teaches those assessments where there is an n% increase in correct answers before and after viewing the SCO. Fischer (2001) has done some work in this area that might be expanded to benefit UTS.

Applicability to the Semantic Web

One major roadblock in implementing the semantic Web (Zhadanova and others, 2004) is the difficulty agreeing to a common vocabulary and writing all of the necessary propositions. Because UTS generates an anonymous concept map, it isn't certain that it would be applicable, but the possibility of integrating UTS with a controlled vocabulary and creating large amounts of semantic Web metadata as a byproduct of learning activity is compelling enough to merit further study.

Assessment Type

For the experiments in this study, UTS used multiple choice questions—students were asked to select one and only one correct answer from among four possible answers (distracters). In later iterations, UTS will support other question types, perhaps even complex simulation questions. This raises the question of whether a single question can yield more information for concept map extraction than a simple pre, post, or unrelated.

Consider the example of multiple select questions. In a multiple select question, the user may select any number of distracters, making it essentially four interrelated true and false questions. This raises the possibility of characterizing the user's response to each distracter independently. I considered using multiple select questions in the

experiments for this study, but was unable to find a simple enough way for instructors to characterize the distracters independently.

An added advantage of multiple select questions is that they are easier to author than multiple choice because "which of the following" questions do not need to be phrased in the negative, and it isn't necessary to think of three plausible but incorrect choices for each question.

Simplified Assessment Authoring

Inducing instructors to write and evaluate the number of assessments required for UTS to work on a large scale will be a challenge. One approach is to set up a monetary or other incentive. Another approach is to simplify the question authoring process.

The most time consuming part of authoring multiple choice and multiple select questions is thinking of incorrect distracters. It might be possible to configure the system so the instructor writes the question and correct answer as a fill-in-the-blank question. Incorrect distracters would then be drawn from subsequent student answers.

Spontaneous Content Generation

An observation from the experiment was that if instructors kept answering off topic questions, they would go far a field very quickly. It would be interesting to assemble twenty or thirty instructors in a large domain and see what evolved from a single initial SCO.

UTS Learning Effectiveness

A study should be completed to determine the learning effectiveness of UTS both in terms of the rate at which students learn, and the amount they learn. For the former, a

good experiment would be to apply the UTS process to a course that already exists as linear ADL. Both UTS and the existing ADL would be extended to accurately measure the time students spend working on the course. Randomly selected students would be assigned to UTS or the ADL over the course of several weeks. Time to master the post assessment would be tracked. I believe this experiment would show that the first students through UTS would take longer than ADL students, but that students who started after UTS had evolved for several weeks would master the material faster.

After UTS performance surpasses ADL, the "ask" option will be removed so that UTS runs in intelligent tutoring ADL mode. Continued tracking will demonstrate whether the superior performance of UTS continues, or even improves as students are forced to more closely consider the questions others have asked.

Multi Page SCORM Player

One limitation of the UTS approach that became obvious is that many answers do not require a full page. A large number of five word SCOs are likely to annoy students. I envision a UI approach that shows content objects as a series of sections on a page, more like "Normal View" in Microsoft Word or the portlets in a JSR 168 portal. The left margin could contain some navigation showing the flow of questions and so on. When to go to a new page could even be left to the student. This idea could also be applied to general SCORM content. A SCORM runtime could be developed that displays existing SCORM content as a series of sections instead of a series of pages.

Appendix A: Experimental SCO Names

"Binary Trees"

"Uses of a Binary Tree"

"Binary Trees and File Organization."

"Depth of a Binary Tree"

"Binary Search Trees"

"Benefits of Sequential and Random Access"

"Balanced Binary Trees"

"Making an Array that Represents a Binary Tree"

"Values Within a Binary Tree"

"Some Special Types of Binary Trees"

"Benefits of a Heap Versus the Binary Tree"

"Ways to Find the Deepeset Node in a Binary Tree"

"About the Universal Tutoring System"

"UTS Architecture"

"Trees and Various Types of Trees"

"The Values of the Elements of a Binary Tree"

"Heapsort"

GLOSSARY

Distracters – The possible answers a student selects from in a multiple choice or multiple select question.

Instructor – In UTS there are two instructor roles: librarian and tutor.

J2EE—The Java application programming interface (API) for server side coding. Originally the acronym Java 2 Enterprise Edition, it is now simply the name of the API.

Learning Management System (LMS) – A class of Web server designed to manage students and deliver courses.

Module – A multi-SCO unit of instruction.

Module Post Requisite – Narrative description of the expected student capabilities after completing a module. Used as a guideline for writing the completion assessment.

Module Prerequisite – Narrative description of the expected student capabilities going into a module. Used as a guideline for tutors and librarians.

Multiple Choice Assessment – The user chooses one and only one distracter.

Multiple Select Assessment – The user chooses any number of the distracters.

Programmed Learning – A technique for organizing learning materials where a question leads students from one module to the next.

Scaffolding -- In educational theory, the strategy of providing temporary support to students in accomplish a task to facilitate learning to complete the task on their own.

Self-Organizing System – An information or physical system that exhibits sophisticated overall behavior based on interactions components which have only local visibility.

Sharable Content Object Reference Model (SCORM) – A US Department of Defense sponsored standard to provide learning content portability among learning management systems.

Universal Tutoring System (UTS) – The working title of the software developed for this thesis.

BIBLIOGRAPHY

Abel, Marie-Helene. 2004. Using two ontologies to index e-learning resources. Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI'04): 553-557.

Discusses the relationship between domain ontology, which defines the knowledge in the domain being trained, and the application ontology, which specifies "organization of theoretical notions which are studied."

Abdulah, N.A. Christopher Bailey, and Hugh Davis. 2004. Augmenting SCORM Manifests with Adaptive Links. Proceedings of the fifteenth ACM conference on Hypertext and hypermedia (HYPERTEXT '04) (August):183-184.

Describes a project to extract concept maps from SCORM manifests using the AuldLinky server. Their goal is to add adaptive links, but they don't really seem to have accomplished that in this paper.

Advanced Distributed Learning. 2004. Sharable Content Object Reference Model (SCORM) Sequencing and Navigation (SN) Version 1.3.1. Internet. Available from http://www.adlnet.org/downloads/index.cfm; accessed August 2005.

This specification contains a detailed description of SCORM requirements.

Advanced Distributed Learning. 2007. SCORM General Common Questions. Internet. Available from http://www.adlnet.gov/help/CommonQuestions/SCORMGeneral Questions.cfm; accessed March 2007.

This site contains interesting historical background on the genesis and evolution of the SCORM standard.

Angelides, Marios C, and Ray J. Paul. 1993. Towards a Framework for Integrating Intelligent Tutoring Systems and Gaming-Simulation. Proceedings of the 35th Conference on Winter Simulation. 1281 - 1289.

The authors identify three capabilities a tutoring system needs to be considered intelligent: 1) draw inferences in the domain being taught, 2) approximate the student's current capabilities. 3) implement strategies to reduce the difference between expert and student capabilities.

Brusilovsky, Peter. KnowledgeTree: A Distributed Architecture for Adaptive E-Learning. WWW 2004. Internet. Available from http://www2004.org/proceedings/docs/2p104.pdf; accessed 5 December 2005.

This paper proposes an architecture that divides responsibility for intelligent tutoring over four server types. This architecture supports the delivery of the same learning objects in or pre-defined or intelligent tutoring way. Many good ideas, but doesn't fit well with SCORM. Excellent overview of existing e-Learning standards.

Byrne, Joshua and William Alfveby. 2005. Java Course Notes. Unpublished book.

Draft of a textbook commissioned by McGraw Hill, but not yet published.

CmapTools Version 4.09, Institute for Human and Machine Cognition, Pensacola, FL, 2007.

Free software tool used to create the designed concept maps in this thesis.

Cycorp. 2007. The Cyc Knowledge Base. Internet. Available from http http://www.cyc.com/cyc/technology/whatiscyc_dir/whatsincyc; accessed 10 March 2007.

Overview of the history and current status of Cycorp's upper ontology and software products.

Fischer, Stephan. 2001. Course and exercise sequencing using metadata in adaptive hypermedia learning systems. Journal on Educational Resources in Computing (JERIC) v.1 (March): 1-21.

The authors focus on creating exercises automatically. Refers to an IEEE LTSC upper ontology. Contains a good overview of various metadata standards.

Heflin, Jeff. 2004. OWL Web Ontology Language: Use Cases and Requirements. W3C 2004 (February). Internet. Available at http://www.w3.org/TR/webont-req/; accessed September 1995.

This document provides a high level introduction to OWL, a language intended to support the creation of a semantic Web.

McGuiness, Deborah L, and Frank van Harmelen. 2004. OWL Web Ontology Language Overview. Internet. Available from http://www/w3/org/TR/2004/REC-owl-features-20040210/; accessed August 2005.

Murray, Tom. 1999. Authoring Intelligent Tutoring Systems: An Analysis of the State of the Art. International Journal of Artificial Intelligence in Education 10: 98-129.

Novak, J. D. & A. J. Cañas. 2006. The Theory Underlying Concept Maps and How to Construct Them. Technical Report IHMC CmapTools 2006-01 Florida Institute for Human and Machine Cognition Internet. Available from: http://cmap.ihmc.us/Publications/ResearchPapers/TheoryUnderlyingConceptMaps.pdf; accessed February 2007.

This paper is an excellent overview of concept maps and how to use them for teaching and learning, by the person who invented concept maps.

Oguejiofor, Emeka, Rafal Kicinger, Elena Popovici, Tomasz Arciszewski, and Kenneth De Jong. 2004. Intelligent tutoring systems: an ontology-based approach. International Journal of IT in Architecture, Engineering and Construction: 115-128.

This article provides a good description of the type of tutoring system the UTS derived ontology could be applied to.

Patel-Schneider, Peter F and Dieter Fensel. 2002. Layering the Semantic Web: Problems and Directions. First International Semantic Web Conference: 16-29.

This paper contains some interesting observations on anonymous nodes, and a good introduction to Resource Description Frameworks (RDF).

Russel, Stuart and Peter Norvig. 2003. Artificial Intelligence: A Modern Approach. Upper Saddle River, NJ: Prentice Hall.

Sugumaran, V. and V. C. Storey. 2006. The Role of Domain Ontologies in Database Design: An Ontology Management and Conceptual Modeling Environment. ACM Trans. Database Syst. 31, 3 (Sept) : 1064-1094.

Zhadanova, Anna V., Jos de Bruijn, Kerstin Zimmerman, Francois Scharffe. 2004. Ontology Alignment Solutions D1.4 V2.0: 3.

VITA

Joshua Emmett Byrne was born at Fort Lewis, Washington, on September 1, 1970, the son of Kathryn Ann Byrne and John Eccleston Byrne. After completing his work at Cloquet Senior High School, Cloquet, Minnesota, in 1988, he attended the University of Delaware. He received the degree Bachelor of Chemical Engineering from the University of Minnesota in May of 1993. During the following years he was employed in various engineering and management positions at Applied Materials in Santa Clara, California, founded Mahnomen, Inc in Austin Texas, and is currently employed as Chief Technology Officer by Adayana, Inc in Minneapolis, Minnesota. In 2003 he entered the Graduate College of the University of Texas State University-San Marcos.

Permanent Address: 9972 Oxborough Road

Bloomington, Minnesota 55437

This thesis was typed by Joshua E. Byrne.