University of Alberta

ProAnalyser: A Multimedia Modeling and Authoring Framework for Discerning Student Learning Processes

by

Nathaniel Rossol

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering

Spring 2010
Edmonton, Alberta

Examining Committee

Dr. Mrinal Mandal, Electrical and Computer Engineering

Dr. Irene Cheng, Computing Science

Dr. Olenka Bilash, Faculty of Education

Dr. James Miller, Electrical and Computer Engineering

ABSTRACT

Online multimedia education systems traditionally tend to consist almost exclusively of multiple choice or numeric response style questions. However, many curricula such as math, physics, and chemistry typically involve students completing large, complex, multi-step problems where the process used to solve the problem is more important than the final answer. Current online multimedia systems are generally insufficient to model or assess problems like these.

In this thesis, I address this issue by implementing a Process Analyzer and its authoring tool. The Process Analyzer aims to improve student problem solving skills by acting as a self-tutoring tool that can analyze a student's problem-solving process and adapt accordingly, providing corrective guidance hints if necessary. Secondly, it provides instructors with an in-depth analysis of the process or thinking steps that students are using to solve complex problems. Instructors can therefore assess students based on their problem-solving process, and not just their final answer.

ACKNOWLEDGMENT

I would like to sincerely thank my family, colleagues, Dr. Mrinal Mandal, and Dr. Irene Cheng.

Table of Contents

Chapter 1 Introduction **1**

1.1 Motivation and Background **1**

1.2 Thesis Organization **4**

Chapter 2 Review of Related Works **5**

2.1 Learning Designs **5**

2.2 LAMS **6**

2.3 Parson's Programming Puzzles **8**

2.4 SIETTE **9**

2.5 AutoTutor **10**

2.6 Learning Beans **10**

2.7 SCORM **12**

2.8 KnowledgeTree **14**

Chapter 3 Problem Solving and Student Modeling **16**

3.1 System Overview **16**

3.2 Problem Solving as a Process or Workflow **23**

3.3 The Process Tree and Data Visualization of Student Interaction **29**

3.3.1 Assessment model 32

Chapter 4 System Architecture and Implementation Overview **34**

4.1 Software Architecture **34**

4.2 Authoring tool overview **36**

4.2.1 Creating a Simple Process Question 38

4.3 System Architecture **41**

4.3.1 Process Tree Implementation 42
4.3.2 Process Question XML Format 43

Chapter 5 Conclusion and Future Work 45

5.1 Conclusion 45

5.2 Future Work 46

Bibliography 47

List of Figures

Figure 2.1: A look at the LAMS authoring tool with a learning design under construction. Image is from [8]..... 7
Figure 2.2: A sample learning bean. Note the content bean which encapsulates all content-specific material into a single node. Image is from [14]. 11
Figure 2.3: An architectural overview of KnowledgeTree. Note that Activity servers do not need to only communicate with the central LMS portal, but can also communicate with various other useful services and modules directly. Image is from [15]. 15
Figure 3.1: Previous version of the Process Analyzer as published in [18]..... 17
Figure 3.2: The Process Editor with a sample process question loaded. The user can modify individual question items by selecting them and adjusting their parameters inside the parameter list. Users can also test out their process question in the Process Analyzer by clicking the “Run” button..... 18
Figure 3.3: The Process Analyzer running with the Process Tree Visualization enabled (right side). 20
Figure 3.4: The complete ProAnalyser Framework which the Process Analyzer and Question Authoring Tool are designed to fit into when in online mode..... 21
Figure 3.5: A simple geometry problem. 23
Figure 3.6: A workflow representing the sample solution to the example problem. 25
Figure 3.7: A workflow that accounts for both possible means by which to solve the problem. The paths branch after step 2 depending upon the way the student wishes to solve the problem. 26
Figure 3.8 A Sample Process Question in which the problem branches into two possible correct paths to the solution. The green colored box-shaped nodes represent Multimedia Question Nodes whereas circles represent data nodes. 27
Figure 3.9: A Process Analyzer tutoring session that teaches students how to create an exotic meal. 28
Figure 3.10: An example of the Data visualization used by VisTrails to

model a path taken through several workflows. Image is from [21]. 29
Figure 3.11: A Process Tree visualization of three students completing the sample problem. The nodes indicate the state the student is in (i.e. which Question sub-step they are on). 30
Figure 3.12: An example of a Process Tree visualization. Edges are numbered so as to make it even clearer as to the order in which the substeps were completed. 31
Figure 4.1: Overview of the Process Analyzer and Process Editor system Architecture. Arrows indicate dependencies and dotted lines represent optional dependencies. Note that the ProVisualizer is an optional plug-in to the client 35
Figure 4.2: An overview of the 4 main components of the ProEditor interface. 36
Figure 4.3: Creating a connection between 2 nodes. 37
Figure 4.4: Adding Process Nodes to the Main Editing Canvas 38
Figure 4.5: Configuring the Image Multiple Choice Question node. 39
Figure 4.6: The two Display Message Nodes are configured and connected to the Image Multiple Choice Question Node. 40
Figure 4.7: The results of two separate test runs are shown. Note that the in-editor version of the Process Analyzer does not have the Process Tree visualizer plug-in installed, but it will be available again when this process is run in applet mode. 40
Figure 4.8: The section of the Process Analyzer Game “21” that decides whether or not the student has just lost the game. 41
Figure 4.9: Save state nodes are placed just before question nodes so that if the student returns to the state by clicking on the corresponding node in the Process Tree, they can attempt the question item again. 43
Figure 4.10: A snippet of a saved XML file that defines a Process Question. 44

GLOSSARY

CAA Computer Adaptive Assessment

CBA Computer Based Assessment

IMS LD IMS Global Learning Consortium's Learning Design specification

Item Type Multimedia-based educational question

LAMS Learning Activities Management System

LD Learning Design

LMS Learning Management System

SCO Shared Content Object

SCORM Shared Content Object Reference Model

UML Unified Modeling Language

XML Extensible Markup Language

Chapter 1

INTRODUCTION

1.1 Motivation and Background

There is little doubt that the wide-scale adoption of computers into educational assessment is ultimately inevitable and a quick glance at the many benefits of such systems over traditional paper-and-pencil methods makes it quite clear as to why this is so. Some of the more notable benefits include savings both in terms of time and cost, reduced instructor marking times, the potential to increase assessment accuracy through adaptive testing, and the potential to measure a wider range of skills and knowledge through the use of graphics, animation and other types of multimedia-based questions (known as Multimedia "Item Types").

Intuitively, the cost and savings advantages of Computer-based testing and learning are quite clear. By placing course material online instead of printing it, not only are there cost savings related to printing expenses, but also time savings in terms of the time that would normally be spent photocopying and transporting these materials. Tests taken online can be marked automatically saving instructors a great deal of time. Furthermore, students can automatically check their results online as soon as the system is done marking it; perhaps even immediately after completing the exam. In [1] the authors suggest that Post Secondary institutions, in particular, that are looking to cut costs could benefit a great deal through the distribution of course material online and even go so far as to say, "As the technology is now available and relatively user-friendly, those universities which do not embrace it will be left behind in the race for globalisation and technological development."

Another key advantage of online computer-based education lies in the promises of the emerging field of Computer Adaptive Assessment (CAA). CAA is a form of Computer Based Assessment (CBA) where the next question a student is given during a Computer-based test depends on how well they answered the previous question [2][3]. For example, if a student answers a given question (or series of questions) very poorly, the system can recognize that the student does not understand the underlying concepts at all, and continuing to ask further questions of the same nature is pointless and will only serve to demoralize the student and waste time. The system may then adapt by asking simpler questions in an attempt to more accurately pinpoint exactly what level of understanding the student has achieved regarding the topic. Conversely, if a student is effortlessly completing a series of questions with great ease, then the system may recognize that there is no point in asking any more questions on the topic as the student has already demonstrated a solid understanding of it. It may then adapt by moving on to more difficult and/or complicated questions in order to narrow-in on exactly what level of excellence the student is performing at. This ability to automatically and precisely home-in on each individual student's skill level is simply not feasible with traditional paper-and-pencil based assessment. The Authors of [2] also believe that the CAA approach saves a great deal of test-taking time because it can narrow-in on individual student skill levels much faster than traditional assessment techniques so an adaptive test would typically need to ask fewer questions overall.

Perhaps the greatest promise of computer-based education lies in the ability to automatically assess a wide range of student skills and knowledge through the use of online multimedia types of questions (called Item Types). While traditional paper-and-pencil tests and course materials are, of course, capable of presenting static images and text, Multimedia is able to offer a much broader range of media including video, animations, and sound. Additionally, multimedia Item Types allow for interactive ways to view data. For example, a physics question might have a complicated 3 Dimensional structure in which a student is required to

calculate the stresses at various points. Using multimedia, the student could rotate the 3D diagram to learn its structure to help solve the problem. Furthermore, if the structure has moving parts, it could even be animated to demonstrate how it works. Traditional paper material, however, would be limited to only displaying an array of static cross-sections where the motion and operation of the setup could only be implied. Computer-based multimedia item types also allow for self-assessment and distance learning (if they are hosted online). This means that students need not be at an academic institution in order to receive new course material or to be assessed. New course material could instead be hosted through a website as the course progresses, and students can even learn and test their knowledge or problem-solving skills from home using an online multimedia-based tutoring system or online practice tests. Some even strongly advocate the advantages of hosting entire courses solely online which potentially allows anyone in the world to take such a course [1].

Unfortunately, as outlined in [4] current-day educational multimedia Item Types tend to mimic the system they are replacing: simple multiple choice and numeric response questions. The focus has largely been on multimedia Item Types that ask the student for only a single response in order to complete the problem; whether it is to ask the student to make a selection or some kind (i.e. Selected Response Items) or to enter a single number or keyword (Constructed Response Items) [4].

However, in many subjects, students are required to solve complex multi-step problems that require a non-trivial procedure and often the solving of smaller intermediate sub-problems (chemistry, physics, mathematics, and computer programming are all good examples of such subjects).

The desire to address this need in multimedia education systems is the motivation behind the implementation of the Process Analyzer, and the design of its encompassing system: the ProAnalyser Framework.

1.2 Thesis Organization

In Chapter 2 related works are presented.

In Chapter 3 the theory behind the design and operation of the proposed system is explained.

In Chapter 4, the implementation details of the system are covered and the system architecture of the Process Analyzer is explained. The justifications for the various software engineering design decisions are explored as well. A series of examples are used to further explain how users can implement a wide range of process-based questions through the combination and use of simple Item Types.

Chapter 5 discusses conclusions and future work.

Chapter 2

REVIEW OF RELATED WORKS

It is virtually impossible to investigate online e-Learning today without quickly coming across Learning Management Systems (or LMS for short). Learning Management Systems are comprehensive online educational software systems that help organize the vast amounts of information that an education institution works with on a daily basis (for example, instructional notes, assignment descriptions, marks, attendance records, exam schedules, etc.). The idea is that, rather than have a separate system for managing each of these aspects, that there should instead be a single online consolidated system to manage all of these activities at once. Some real-world examples of successful Learning Management Systems include WebCT [5], BlackBoard [6], and Moodle [7] (WebCT actually currently enjoys wide usage at the University of Alberta by several faculties, and Moodle is popular with the Department of Computing Science). With these LMSs, students can log into the system, view and submit their upcoming homework assignments, download lecture notes, view their marks, participate in online discussions, and even complete simple multiple choice quizzes or practice tests. Teachers can log into these systems and perform a wide range of actions such as uploading or updating course notes, making announcements to their classes, creating online practice quizzes, and a variety of other tasks.

2.1 Learning Designs

Given that most instructors traditionally create semester-long course plans outlining all learning activities for the duration of the course and what sequence they will occur in, it seemed only natural that this should exist for online learning systems too [8]. Furthermore, online course plans would have some benefits that

traditional paper-based course plans wouldn't. For example, course plans usually reflect an instructors teaching method. If the instructor could separate the basic structure of the course plan from the course content, then the course plan (and teaching method) could theoretically be applied to an entirely different course. This ability to reuse course plans could save a lot of time and money if course plans/teaching methods that proved to be very effective could be shared and used by other instructors. In the online e-Learning world, these course plans are called "Learning Designs" (LD) and the most well-established LD specification to date is the IMS LD maintained by IMS Global Learning Consortium [9]. The main reason behind creating a LD specification is so that if an instructor, for example, creates a LD for their course in Moodle, then so long as they created that course LD with IMS LD - compliant software, then that LD should be fully portable to a different LMS (like WebCT) so long as that other LMS is also IMS LD compliant (at the time of this writing, WebCT has shown no indication of becoming IMS LD compliant though Moodle has plans to implement this feature for its next version according to the Moodle roadmap[10]).

2.2 LAMS

A system that *has* adopted the IMS LD, however, is LAMS - short for "Learning Activities Management System" [8]. The authors of [8] explain that by using LAMS, an instructor can plan a sequence of "Learning Activities" to teach a concept (Learning Activities include things like online group discussions, online quizzes, research assignments, etc.), and the system will automatically schedule them for students to participate in.

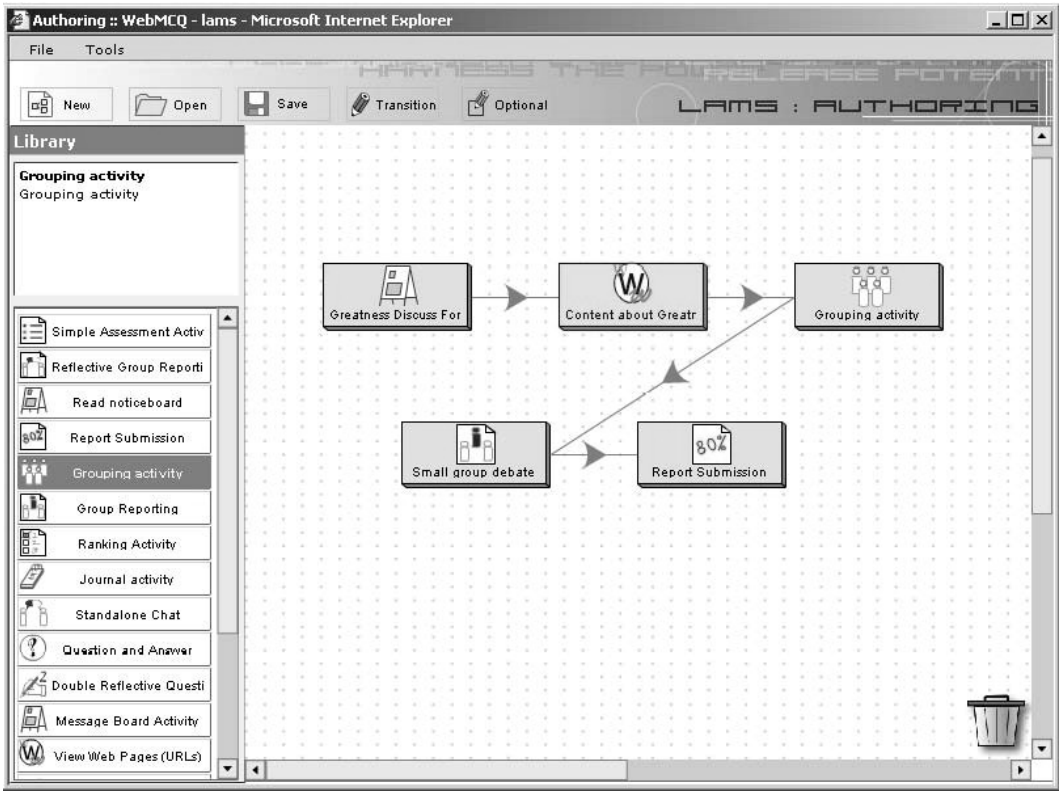

Figure 2.1: A look at the LAMS authoring tool with a learning design under construction. Image is from [8].

The authors of LAMS highlight the large degree of reusability of Learning Activity Designs created by their software thanks to the separation between the design and its content. In their paper, they even go so far as to claim that a learning design like the one shown in Figure 2.1 could be reused for an entirely different subject (i.e. different "content" but same design) and that the change could be made in as little as "5 minutes" (provided the course content for the other subject was already available) [8].

In comparison between LAMS and the Process Analyzer, the two systems are on different levels of course content management. That is to say that whereas systems such as LAMS are used to plan course learning activities such as online class discussions, online reports, and perhaps simple online multiple choice quizzes, innovative online educational systems like the Process Analyzer (and

others like it) are stand-alone educational systems not bound nor reliant on any LMS, and not concerned with greater course planning or learning designs, but rather just focused on teaching and/or assessing a specific subject, set of subjects, or specific types of problems across multiple subjects. The following are some such examples of other innovative online educational systems that share this important domain with the Process Analyzer.

2.3 Parson's Programming Puzzles

Parson's Programming Puzzles [11], is similar to the Process Analyzer in that it is also an online teaching and learning system designed to engage students through the use of multimedia. Like the Process Editor, Parson's Programming Puzzles also comes complete with an authoring tool that instructors can use to author their own content. Students (who are typically learning programming for the first time) use a single Selective Response Item in the form of a drag-drop Item Type to assemble a working program out of several code segment objects. The system also may use graphics to display the intended function of the system as a UML activity diagram.

As with the Process Editor, this system allows students to follow incorrect paths for a while, (giving them the opportunity to correct themselves when they realize this) and also supports multiple paths to the correct solution with some paths being better than others (i.e. some make better use of proper coding style and software engineering design principles). According to the authors, 82% of students sampled who used the system reported that they found it useful for learning a new programming language.

However, as the name suggests, Parson's Programming Puzzles system is solely designed for teaching introductory programming and is not general enough to simply be applied to another subject area. Furthermore, while students do in fact go through the process of constructing a program incrementally, step-by-step, only the final result is actually evaluated, and the process the students used to

construct the solution (including back-tracking and self-corrections) are not recorded and therefore such information is simply lost.

2.4 SIETTE

The "System of Intelligent EvaluaTion using TEsts" (SIETTE) [12] is a popular web-based adaptive testing system that attempts to use CAA in order to adaptively assess and evaluate student performance. Instructors populate a question bank with a large selection of questions on the subject that they are testing and define a large number of parameters to help tune how the system operates. These parameters include things such as the difficulty level of the questions, time limits, and the length of the test. There is also the option to allow the system to pose as many questions to the student as necessary in order to estimate their knowledge level within a defined level of statistical certainty. Experimental studies conducted by the group indicated a possible promising correlation between the use of their system and an increase in student performance. The group plans to verify this with future larger-scale experiments.

One important limitation of the system, though, is that the various parameters for each question Item must be set by hand (by the instructors). This can lead to problems if, for example, what an instructor believes is the correct difficulty level of a question is not very accurate compared to what students find difficult. Similarly, while an instructor may feel that a certain question tests a specific set of concepts, there may be others that were not considered, and thus the question might be incorrectly categorized by the software and thus asked at inappropriate times. A major limitation though, is that the system assumes that each question item is independent of the next. In fact, this is a requirement as the adaptive component of the system selects which question to run next based solely on its estimation of the student's knowledge levels. Instructors here do not have the ability to decide what question Items to load next based on a student's answer because the SIETTE system is in control of this instead. Using the Process

Analyzer, however, instructors are in direct control of which Item Type content to bring up next based on the student's actions within the confines of a single Process Question. The SIETTE system is also currently restricted in its design to only handle multiple choice questions.

2.5 AutoTutor

The AutoTutor system as described in [13] is one of the many systems that attempts to learn about a student's psychological state (i.e. Affective State) in order to adapt appropriately to tutor better. For example, if the system believes the user is becoming bored or uninterested, it could take adaptive action by providing a more engaging Item Type next or perhaps switching to a different topic for a short while. The AutoTutor system described in [13] uses natural language and questions in a conversation-based format to mimic a real human tutor. Arrays of visual, audio and tactile sensors focused on the learner are the way in which the system attempts to infer the emotional state of the learner. The version described was implemented to teach both Newtonian Physics and computer literacy.

Although the Process Analyzer was not designed to specifically deal with the emotional or Affective states of the students, it is flexible enough to allow for this data to be inputted into the system when a process question is loaded. Developers can implement Process Nodes to provide this data to the Process Analyzer, and Item Type Nodes can be implemented that would use this data to adapt in response to the data regarding the student's affective state during a tutoring session.

2.6 Learning Beans

Learning Beans is perhaps the most similar system to the Process Analyzer of all the systems mentioned thus far in the sense that it provides a flexible state-chart based flow from one process to another [14]. Also, drawing inspiration from LAMS, Learning Beans provides a separation of system behavior from question

content. System behavior encapsulates such aspects as how many tries the system will allow the user to make on a question, how much time to give the user, how to respond to an incorrect answer, whether or not to display the correct solution after so many tries, etc. The question content itself (i.e. the actual question and user interface the student interacts with) is encapsulated into a single node known as a "Content Bean". The idea is that the exact same question design can be re-used simply by plugging in a different Content Bean in the same manner that different LAMS designs can be used for different subjects and course content. Through proper configuration of the Content Bean, a series of Learning Bean questions can even be generated at random.

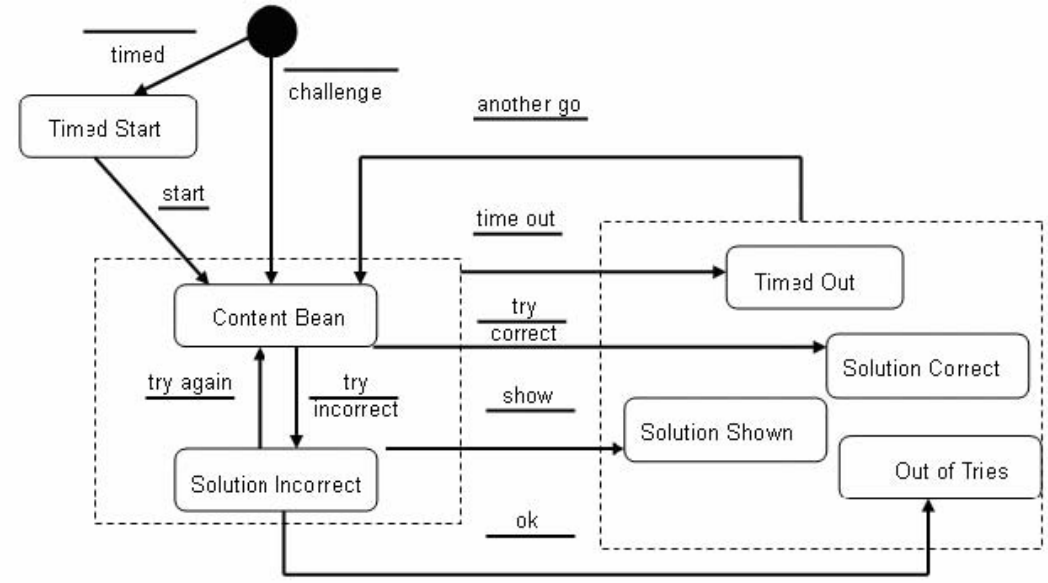

Figure 2.2: A sample learning bean. Note the content bean which encapsulates all content-specific material into a single node. Image is from [14].

However, while the system is capable of using innovative interactive multimedia Item Types like the Process Analyzer to assess students, it still only considers the final answer provided by the student, and does not record any intermediate thinking steps the student used to arrive at their solution. Similarly, there is no dialog between the system and the student (i.e. like the Process Tree) so students

cannot backtrack to a previous step or visualize the steps they've taken thus far in their attempts to solve the problem.

2.7 SCORM

While innovative online multimedia systems such as the ones examined above have been available for several years now, as the authors of [15] point out, "just a handful of these systems are actually being used for teaching real courses, typically in a class lead by one of the authors of the adaptive system." These authors then go on to explain that the reason for the low adoption of these systems is largely a matter of cost and commercial viability. Many of these systems are designed only for a very specific subject area or course, and would not be suitable for others. Therefore, an educational system hoping to capitalize on these systems would need to use a wide range of them to cover its various subject areas. When you consider that many of these systems require their own web server, with their own separate login, uses its own separate data format, and often has its own separate authoring tool, the costs and logistics of trying to manage (and train instructors for) such a large number of wildly different systems would be unmanageable[15]. So, to summarize, on one hand, you have an LMS which will allow you to plan for courses, assign discussions, readings, and other Learning Activities for your class, and on the other hand you have innovative online question Item Types and educational systems (like the Process Analyzer) which will be useful for teaching specific content or types of problems in your courses, but unfortunately, there is no way to connect these two different levels of educational systems together. In addition, such educators typically don't have the time or capital to learn how to use each and every educational system they would want to use. Learning something as large as a LMS alone takes a great amount of time.

Recognizing this, one solution to this tricky issue was proposed by the creators of SCORM (Sharable Content Object Reference Model) [16]. SCORM is similar to

LAMS in the sense that it is also a tool to assist LMSs in generating reusable, Learning Designs but its biggest advantage is that it provides a specification for a standardized interface for web-based Multimedia tutoring and assessment systems (like the ProAnalyser) to be used directly within LMS Learning Designs. Therefore, instructors could configure and use any of these online educational systems directly by treating them as just another form of content for their Learning Designs (i.e. the SCORM system will treat them as an SCO: Sharable Content Object). The idea is that any online teaching/assessment system that implements the SCORM standard can be used or ported to any SCORM-compliant LMS. The authors envisioned vast repositories of online teaching/assessment systems would be available that implemented SCORM, and that any and all LMSs would be strongly motivated to be SCORM compliant as well in order to allow instructors to leverage these vast repositories for teaching their courses (all through a common interface in their LMS).

However, several other authors were quick to point out some important limitations and drawbacks of SCORM. The authors of [17] point out that, in fact, the greatest advantage of SCORM - that its Learning Designs can be reused from one subject or course into another is indeed the source of a number of critical limitations. This reusability feature requires that all SCOs are all subject-content independent, and that, in fact, in order for the SCORM model to work, these learning systems may not know the context in which they are being used. For example, suppose an instructor creates a Learning Design for their programming course, and wants to include the use of Parson's Programming puzzles to teach some concepts of the course. However, if they wanted to reuse this Learning Design for, say, a History course, this would not be possible because Parson's Programming puzzles are limited to the domain of computer programming only, and thus any Learning Design using Parson's Programming Puzzles would fail to meet the requirement that it be reusable from one course to another. Therefore, even though Parson's Programming Puzzles is very useful for the specific subject

it is designed for, it can never be SCORM-compliant because it is too tightly bound to specific set of content or subject matter. This means that SCORM excludes a large set of high-quality tutoring and assessment systems simply because they were designed to teach specific concepts from specific subject matter - regardless of how useful they might be in that role.

Another limitation of SCORM pointed out by [17] is that current Learning Designs are focused on providing a Learning Design for an entire class and this poses an obstacle for adaptive web-based educational systems which work on the level of an individual student. According to the SCORM model, SCOs should not maintain state information between sessions, and thus adaptive education systems that want to be SCORM-compliant have no way to know what student they are currently tutoring or assessing, and must model the student again from scratch at the start of each session. However, the authors of [17] also point out that specification design is an iterative process, and hope that many of these issues will be resolved in future iterations of SCORM.

2.8 KnowledgeTree

In light of these current issues with SCORM, the authors of [15] proposed their own solution named "KnowledgeTree." KnowledgeTree takes a distributed approach to LMS design so that all Learning Activities or SCOs can exist on separate servers or as separate modules that all interact with one another.

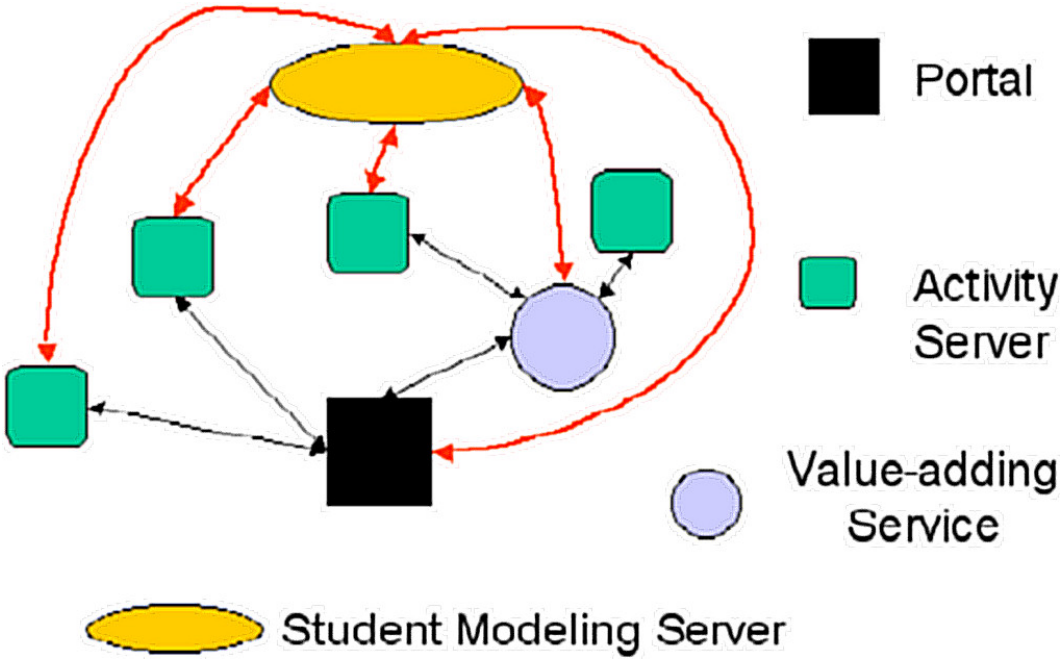

Figure 2.3: An architectural overview of KnowledgeTree. Note that Activity servers do not need to only communicate with the central LMS portal, but can also communicate with various other useful services and modules directly. Image is from [15].

The authors point out that such an approach leads to other important benefits. If, for example, the instructor wishes to use a different adaptive student modeling algorithm or strategy, they need only to make the change on the Student Modeling Server and all related Learning Activities Servers or modules will use the new student modeling data immediately. The same principle would naturally apply to any other servers or modules that perform a useful service to the system.

Given that most of these standards are still relatively early in their development and testing stages, it is difficult, at this time, to predict which one(s) will emerge as the dominant standard. However, the Process Analyzer is designed in a modular way such that implementation into a standard should ideally be straightforward once one is selected. In the meantime, integration into an LMS (such as Moodle) is also being considered.

Chapter 3

PROBLEM SOLVING AND STUDENT MODELING

3.1 System Overview

In this chapter, I propose the Process Analyzer as an effective means for the automated assessment and evaluation of complex, multi-step process-based educational problems. As the name implies, the Process Analyzer analyses the processes (or steps) students take to solve these complex problems so that student evaluation is based on their process rather than just their final answer. Instructors can also visualize the student processes afterwards to help narrow-in on trouble areas for specific students or even entire classrooms of students all at once.

The first version of the Process Analyzer I developed (and was published in [18]) was only designed to handle complex problems that could be broken down into a series of equations that could be solved one at a time as shown in Figure 3.1.

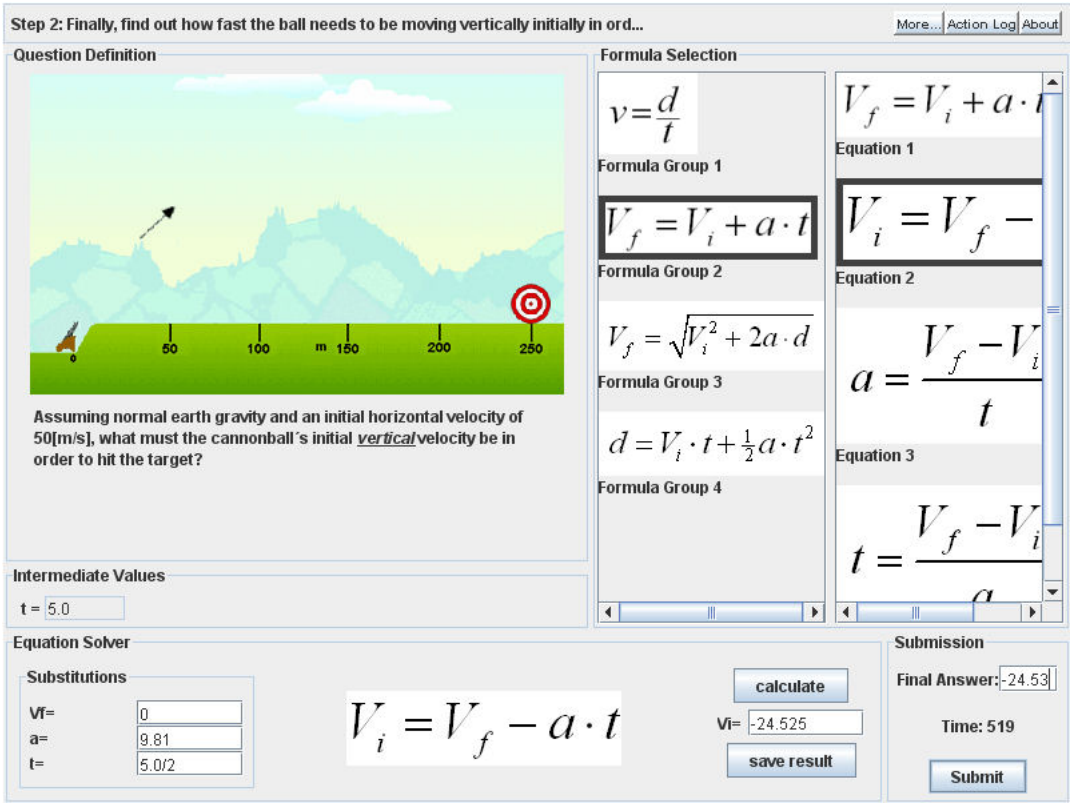

Figure 3.1: Version 1 of the Process Analyzer as published in [18].

The student first selected the appropriate equation/law from the "Formula Selection" panel, and then the needed permutation of the equation from the panel next to it. The student then substituted-in the appropriate values in the bottom "Equation Solving" Panel. After that, students would use the "calculate" button to compute the intermediate (or potentially final) result and would have the opportunity to revise this result before attempting to save it. The system was capable of providing corrective action in the form of instructor hints if a student calculated an intermediate step incorrectly. The instructions in the "Question Definition" panel were HTML-based allowing special user-specified formatting and multimedia in the form of animated diagrams.

However, this system has many important limitations. Obviously, being limited to only sequential equation-based problems is an issue, but the system is also limited in the sense that it only accepts a single process leading to the solution even if

multiple paths to the correct solution exist (as was discovered to actually be the case in the example in Figure 3.1).

Since that time, I have implemented a new Process Analyzer (version 2) that is capable of modeling any process-based question that can be broken down as a series of simpler Item Type responses. This to say that the new Process Analyzer's features are, in fact, a super-set of the features of the first version. The new Process Analyzer is capable of modeling all the same problems of version 1 in addition to many new types of problems. Also, unlike the first version, this second version of the Process Analyzer is now complete with a Multi-media Workflow-based Authoring tool (named the Process Editor) that allows instructors to create and preview Multimedia process-based questions (rather than requiring that the XML code be written by hand).

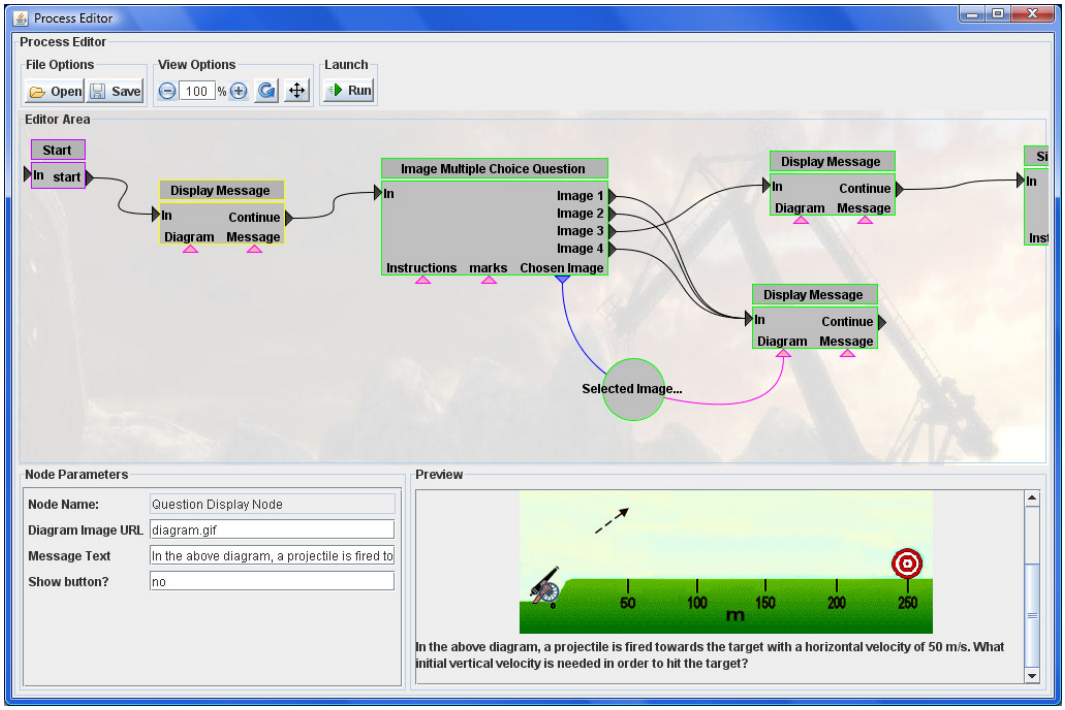

Figure 3.2: The Process Editor with a sample process question loaded. The user can modify individual question items by selecting them and adjusting their parameters inside the parameter list. Users can also test out their process question in the Process Analyzer by clicking the "Run" button.

The new Process Analyzer uses a modular approach to building process-based models of problems. The problem is built-up as a series of Item Type nodes (both Selective Response Items and Constructive Response Items) and branches differently depending upon student responses. Not all nodes in a process question are a Multimedia Item Type Node, however. Other nodes (named Process Nodes) exist which perform various other tasks such as processing data, displaying information to the user, and providing adaptive behavior. In Figure 3.2, for example, the "Image Multiple Choice Question" node is a multimedia Item Type node whereas the "Display Message" nodes are just basic Process Nodes. In this sense, Item Type nodes are simply one of the many types of Process Nodes that make up a process question. The Process Analyzer simply begins at the "Start" Node and proceeds to the next Process Node that it is connected to, branching if appropriate according to student responses or other conditions. Process Nodes (including Multimedia Item Type Nodes) are able to accept data in the form of input parameters and can also output data to another Process Node. In this fashion, Item Type Nodes can pass along any data they wish to each other.

Item type nodes are designed using a plug-in style interface approach. This allows for brand new item types to be implemented and integrated into the system with ease. A programmer need only create an Item Type class that implements the provided abstract class interface methods, and it will be available for use inside the editor and Process Analyzer immediately.

A key novel feature of the new Process Analyzer is that as a student solves a process question their solving process is visualized and displayed in a tree format. This Process Tree visualization (Figure 3.3) is also interactive in the sense that the student can click on any node of the tree to restore the process analyzer back to the corresponding state. This essentially allows a student to "backtrack" if they believe they've made a mistake on a previous step, or to switch back to a different path if they believe it might have been a simpler approach to solving the problem. In Figure 3.3, for example, the student attempted to solve step 1.2 twice before

using the Process Tree to backtrack all the way to step 1.0 and a try a brand new successful approach.

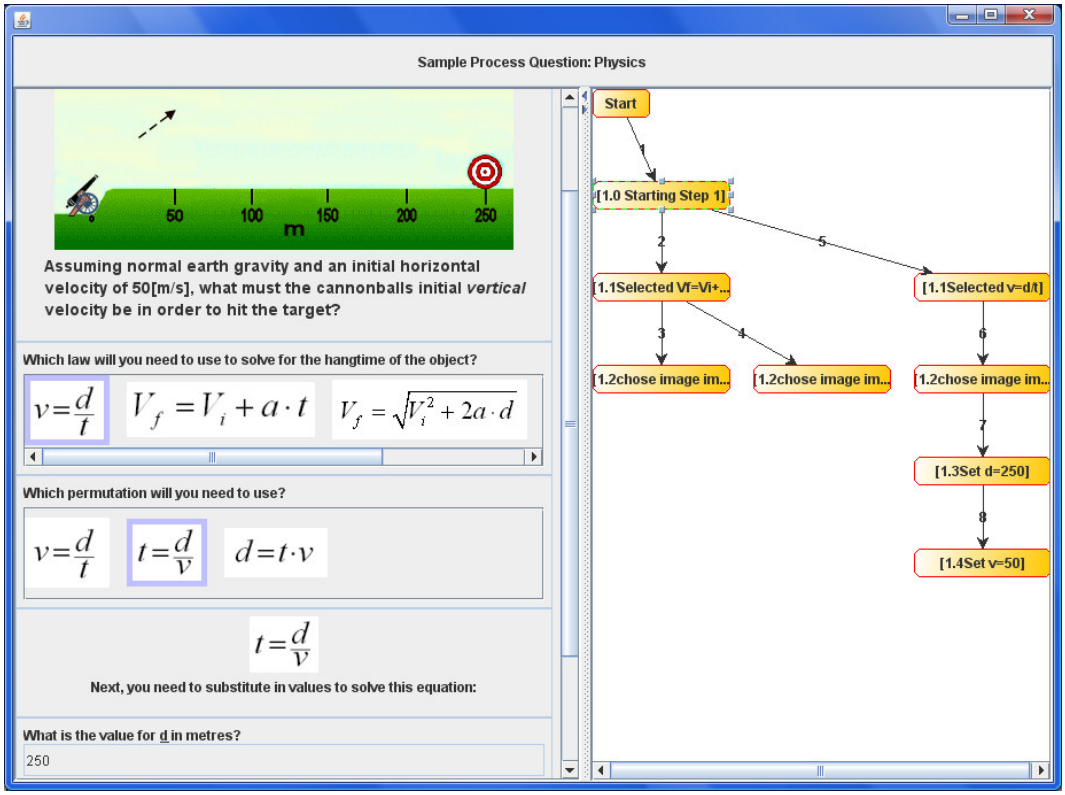

Figure 3.3: The Process Analyzer running with the Process Tree Visualization enabled (right side).

Upon completion of the problem, the process tree is submitted to the server along with the student's answer so that instructors can inspect a student's Process Tree and visually identify where a student (or group of students) ran into difficulties whilst attempting to solve the problem. It can also help give instructors insight into the decision-making process a student used when attempting to solve the problem.

The Process Analyzer is capable of both online and offline use. In the case of offline use, it can be used for tutoring purposes and will read the problems from .xml files off the local hard disk. In the case of online use, the Process Analyzer can run inside a web browser and can receive the process question's XML data

from a database. Process Trees submitted to the server-side database can then potentially be visualized using a separate visualization module. This complete system when considered as whole, has been named the ProAnalyser Framework, and is laid out in Figure 3.4.

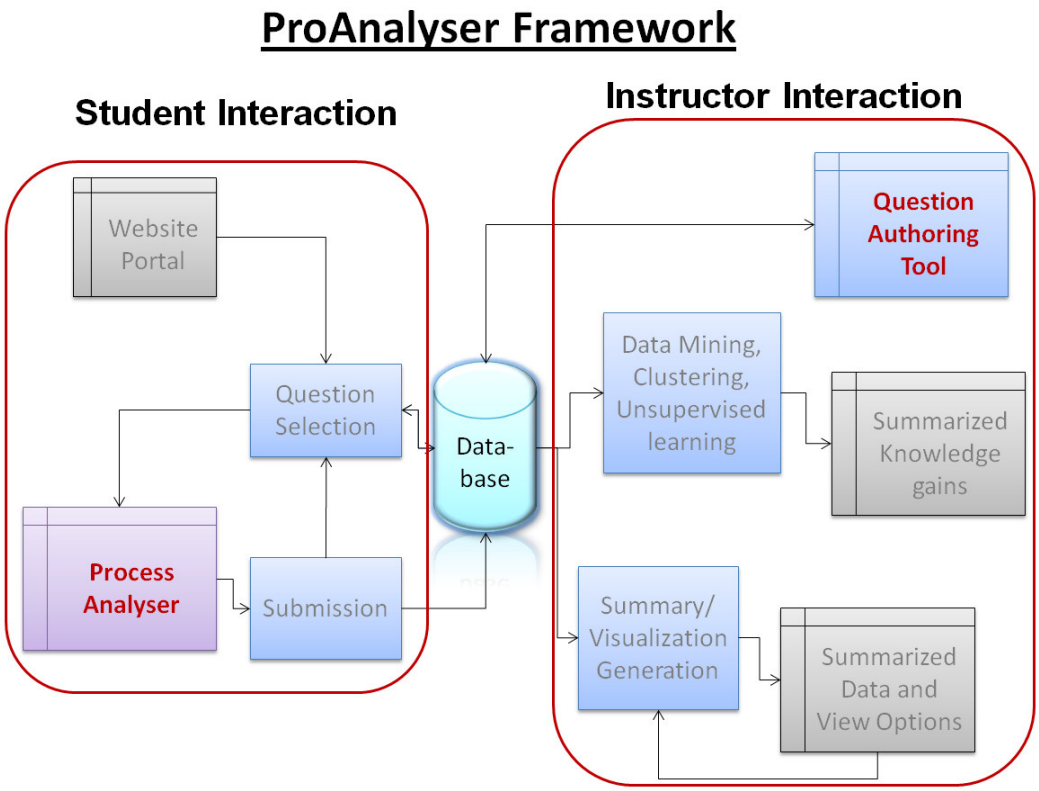

Figure 3.4: The complete ProAnalyser Framework which the Process Analyzer and Question Authoring Tool are designed to fit into when in online mode.

As Figure 3.4 shows, students interact with the system by answering problems through the Process Analyzer. Once a problem is completed and submitted by the student, a potentially adaptive server-side process [19] updates the database and decides which problem to retrieve next and give to the student.

As shown on the right hand side of the diagram, instructors can use data visualization software to analyze the Processes that various students went through in order to solve their problems. The potential also exists to use data-mining,

clustering, and/or unsupervised machine learning techniques to try to gain some useful knowledge regarding re-occurring patterns in student achievement regarding the problems. If the Process Trees of a large number of students are collected in this fashion, general trends may present themselves regarding the manner in which students are solving the problem. As such, it may be possible for the system to predict (within a certain degree of certainty) the likelihood of a student becoming "stuck" when they deviate onto a path that will not result in a correct answer. If the system is in tutoring or teaching mode, it may then provide necessary guidance or take preventative action in the form of corrective hints to prevent the student from becoming hopelessly frustrated or irrecoverably lost.

Thus far, however, only the Process Analyzer (and its authoring tool) is implemented, and thus will be the primary focus of this Thesis.

3.2 Problem Solving as a Process or Workflow

Throughout our education as youths, we were constantly told that the best way to solve a complex problem is to break it down into smaller parts. This approach makes sense because most large problems are in fact just a collection of simpler sub-problems. If one can solve all the smaller sub-problems, their results can be combined to produce the final solution to the original problem. Consider the following trivial geometry problem as an example:

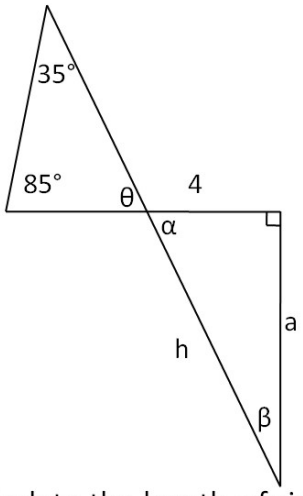

Figure 3.5: A simple geometry problem.

In this problem the student might solve the problem like so:

Step 1: The Interior angles of a triangle always sum up to 180 °, therefore:

$$35 + 85 + \theta = 180$$

$$\theta = 180\text{-}35\text{-}85$$

$$\theta = 60$$

Step 2: Opposite interior angles are always equal, therefore:

$$\alpha = \theta$$

$$\alpha = 60$$

Step 3: And finally, because we are dealing with a right angle triangle:

$$\cos(\alpha) = 4/h$$

$$h = 4/ \cos(\alpha)$$

$$h = 8$$

In this example, the student broke the problem down into 3 smaller sub-problems, namely: finding θ, finding α, and finally solving for h. Steps 1 and 3 also had 3 sub-steps each, namely: identifying the appropriate law to use (and the appropriate permutation of that law), substituting in the appropriate values, and finally solving the resulting expression. Step 2 simply involved recognizing the correct geometry law.

Given that each step (and sub-step) was dependent upon the step before it (as is typical of these types of problems) it seems quite natural to model it as a workflow. The diagram below shows a possible modelling of this problem as a workflow:

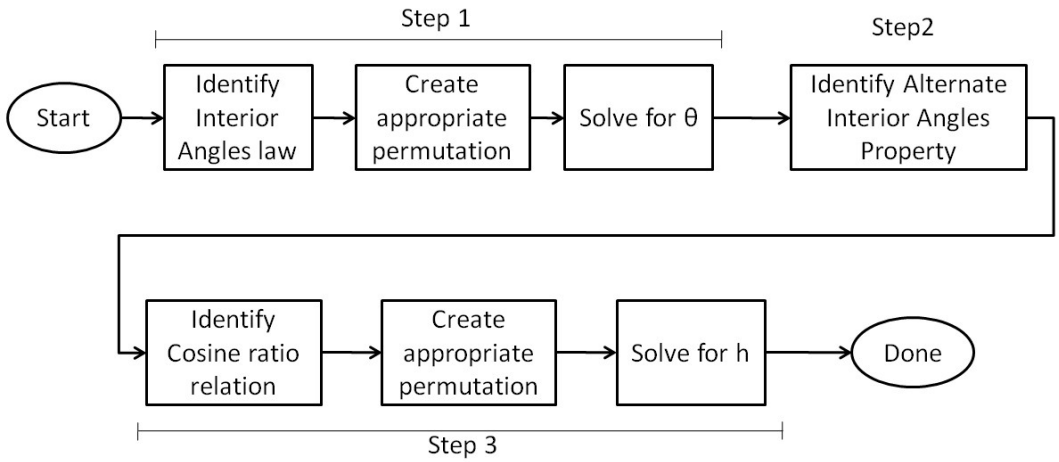

Figure 3.6: A workflow representing the sample solution to the example problem.

However, in the provided sample question, there exists another route to the final solution that the student may take. Suppose that the student is not aware of the relationship between the cosine of an angle and the sides of a right-angle triangle, the student may then instead use tangent relationship to compute the length of side 'a' instead after step 2 like so:

$$\tan(\alpha) = a/4$$

$$a = 4 * \tan(\alpha)$$

$$a \approx 6.928$$

Afterwards the student could use the Pythagorean Theorem to compute the length of side h like so:

$$h^2 = a^2 + b^2$$

$$h = \sqrt{((6.982)^2+4^2)}$$

$$h = 8$$

If we call the previous solution A and this latter solution B, a work flow that accounts for both of these acceptable paths to solve the problem might look like so:

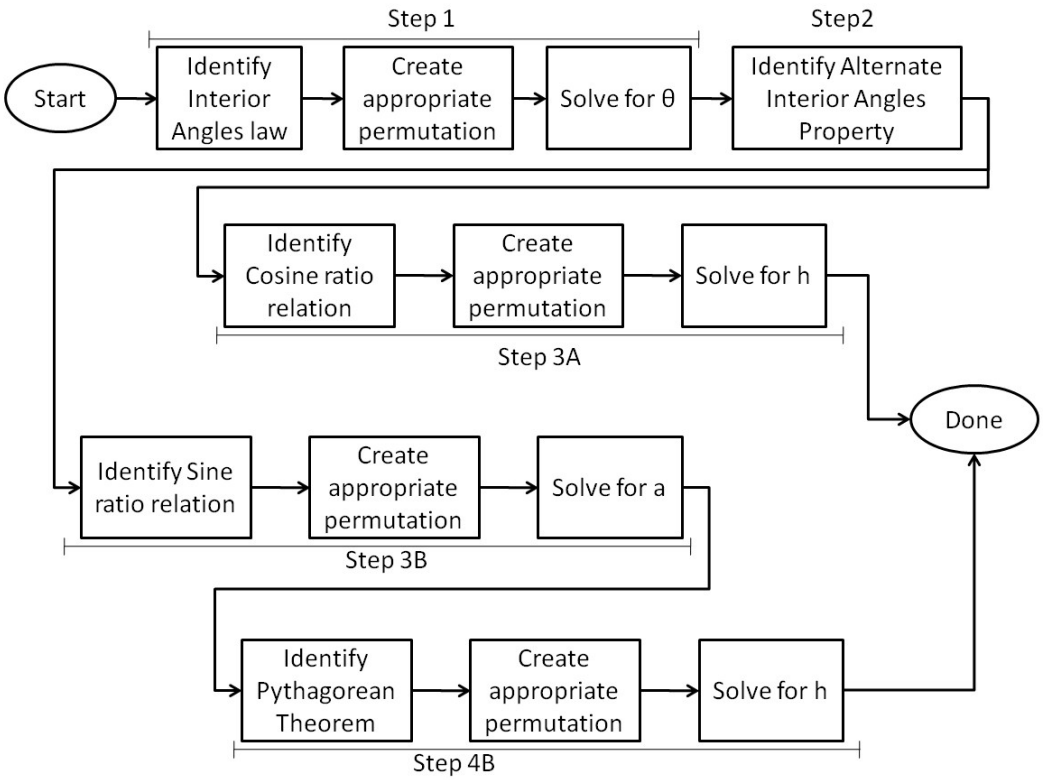

Figure 3.7: A workflow that accounts for both possible means by which to solve the problem. The paths branch after step 2 depending upon the way the student wishes to solve the problem.

The capability for process-based problems to have multiple branches all leading to a correct solution was one of the main rationales behind the workflow-based design for Process Analyzer questions. As shown in Figure 3.8, the Process Analyzer can branch differently depending upon the choices a student makes to try to solve the problem whereas most existing computer-based tutoring systems (as shown in Chapter 1) usually allow only a single path to the correct answer.

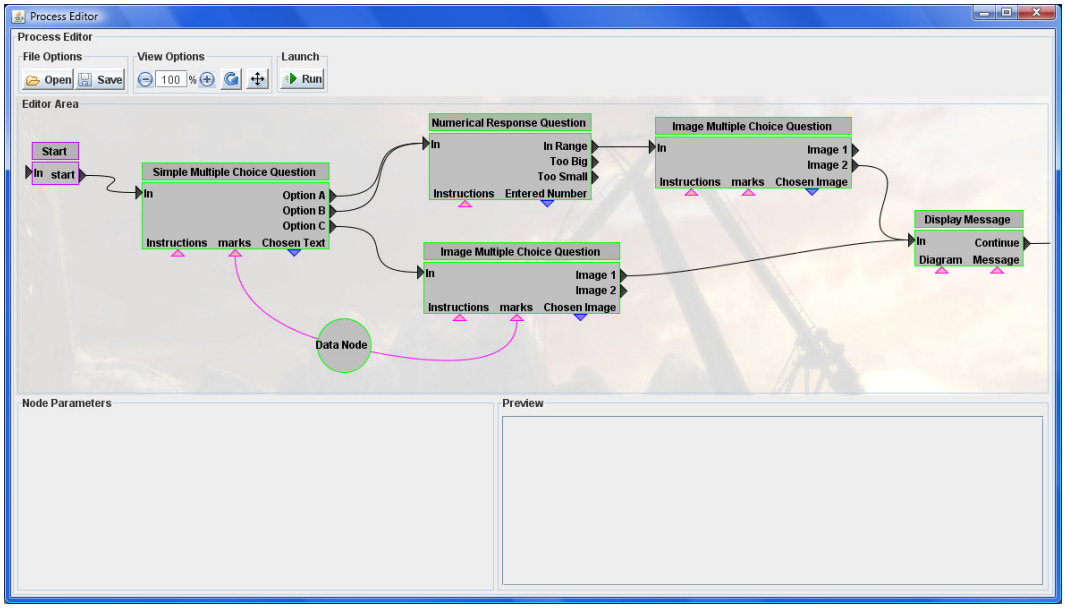

Figure 3.8 A Sample Process Question in which the problem branches into two possible correct paths to the solution. The green colored box-shaped nodes represent Multimedia Question Nodes whereas circles represent data nodes.

Each Multimedia Question Item Node in the Process Analyzer has the ability to branch differently depending upon the student's response. The system can also branch based on some logic described by the author (for example, comparing the current response with a past one). It is important to note, therefore, that when the Process Analyzer is executing, it is not modeling an FSM (Finite State Machine) because the state is not dependent solely upon which node in the Process Question the system is currently on. Process Nodes inside these workflow-based Process Questions are capable of saving data in variables for use by other nodes later in the process. The combination of the current state and all saved variables together define a state in the Process Analyzer.

It is also important to note that although the example Process Questions shown so far are DAGs (Directed Acyclic Graphs) the Process Question workflows are actually fully capable of containing cycles. This feature allows for the system to provide corrective action as needed. For example, if a student branches away from

any of the defined paths that can lead to the correct answer, the instructor authoring the Process Question can have the system branch to a "Hint node" that will display a corrective hint geared directly towards the mistake that the student made. After the hint is displayed, the instructor can have the workflow loop-back to the Process node the student made the mistake on. This feature would naturally be ideal for use in the system's Tutorial Mode.

Finally, in spite of the examples given in this section thus far, it should be made clear that the Process Analyzer is not limited to just the fields of mathematics and science, but that, in fact, any problem that can be broken down into a series of testable process steps can be modeled with the Process Analyzer. Figure 3.9 shows an example of a Process Analyzer tutorial that teaches and assesses the process of cooking and preparing a dish.

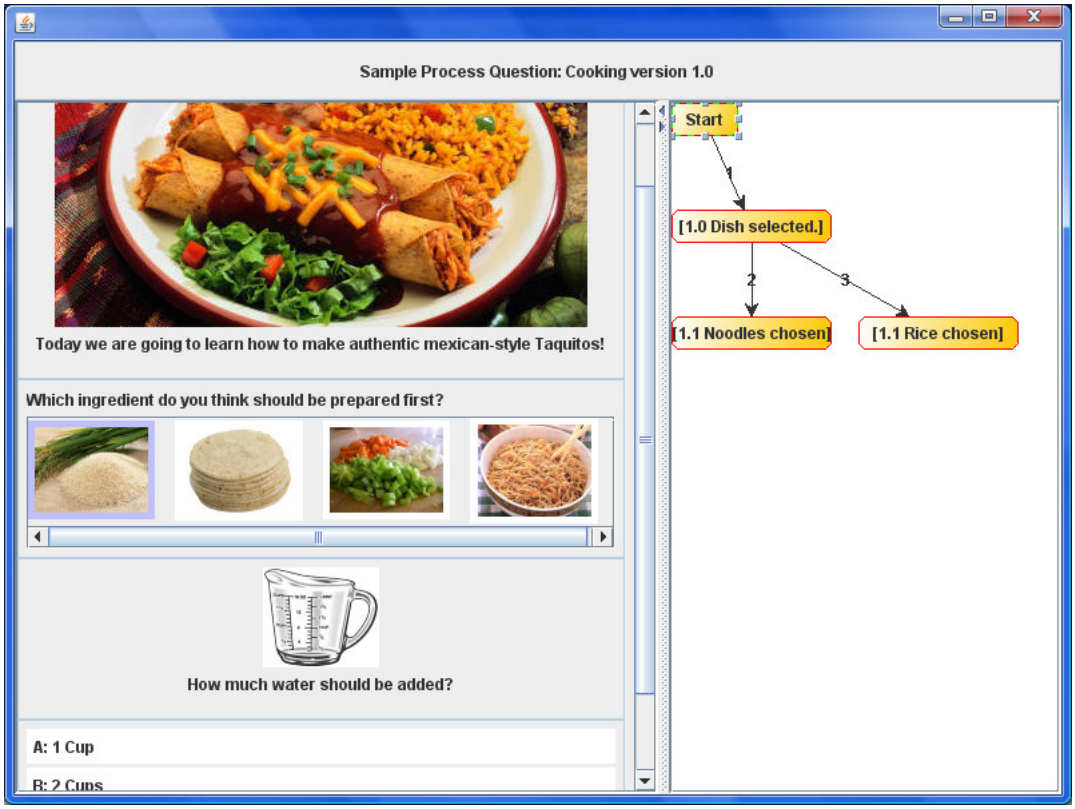

Figure 3.9: A Process Analyzer tutoring session that teaches students how to create an exotic meal.

3.3 The Process Tree and Data Visualization of Student Interaction

As the authors of VisTrails explain in [20], data without understanding is meaningless. Data needs to be interpreted in order to be of any use to humans and data visualization is seen as an effective means by which to summarize data for easy interpretation. An effective data visualization can give the user an accurate overview of the data at a glance, whilst providing more in-depth details as needed.

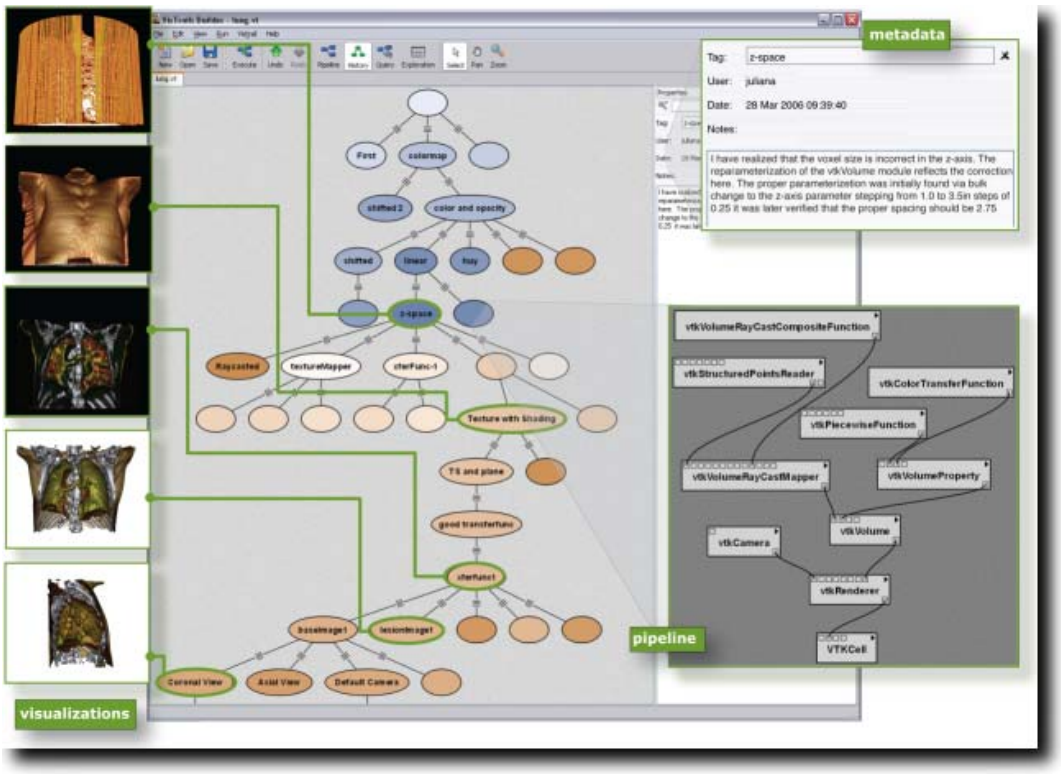

Figure 3.10: An example of the Data visualization used by VisTrails to model a path taken through several workflows. Image is from [21].

As mentioned previously in this document, the Process Analyzer records every action a student takes in the process of solving a problem including any dead-end branches where the student deviated from any of the correct paths that would have led to a solution. For example, using the mathematics example from the previous section, suppose student A completed each step flawlessly and Student B

completed steps 1 and 2 flawlessly, but was not able to immediately identify all the variables needed to complete the cosine relationship, so instead used the sine relationship and the Pythagorean Theorem. Also consider a third student, C, who first tried to apply the law of Sines in step 1 and then the law of Cosines before finally realizing the need to use the Interior Angles Law. Using a tree format, we could model each of the student's problem solving processes like so:

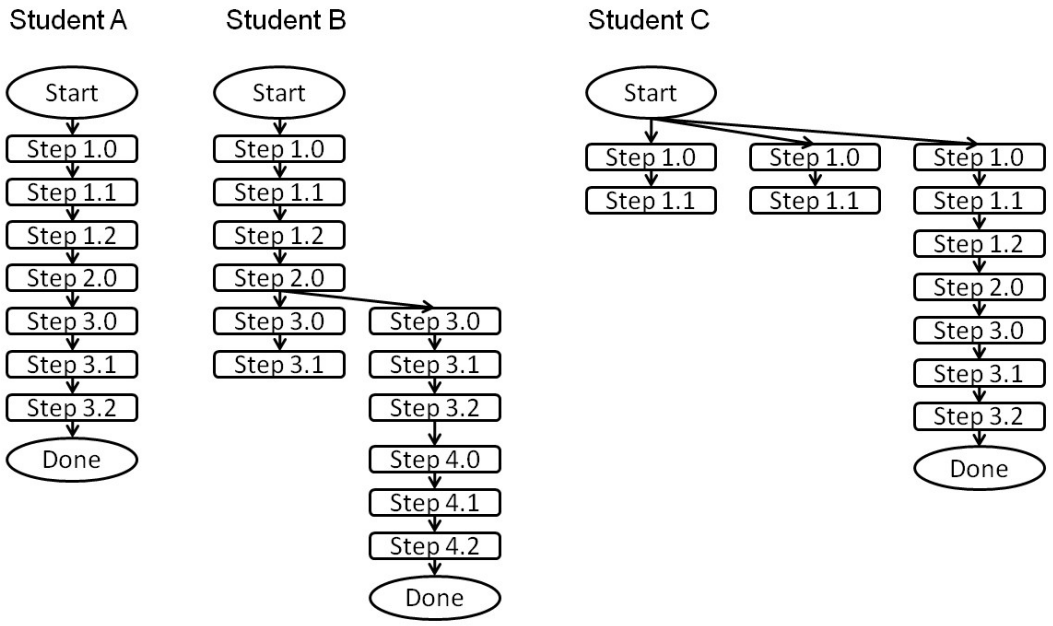

Figure 3.11: A Process Tree visualization of three students completing the sample problem. The nodes indicate the state the student is in (i.e. which Question sub-step they are on).

As we can see in the above figure, student A completed one step after another without any issue and thus has no branching. Student B started on one approach for step 3, but became "stuck" at 3.1 and abandoned it. The student then took a different approach to step 3 which led to a fourth step which then lead to the final answer. Student C attempted 3 strategies for the first step before finally settling on one that worked. From this visualization, much can be gained at an initial glance. Instructors can easily recognize that students with little or no branching in their Process Trees obviously had little difficulty with the problem. However, students

with a great deal of branching (i.e. Process Trees with greater "breadth") clearly had much more difficulty in solving the problem. In paper-and-pencil style exams, such repeated failed attempts at solving a step would often be erased by students and lost. However, a great deal can be learned from failed attempts, and as such, the Process Analyzer records every action and solution attempt that the student makes.

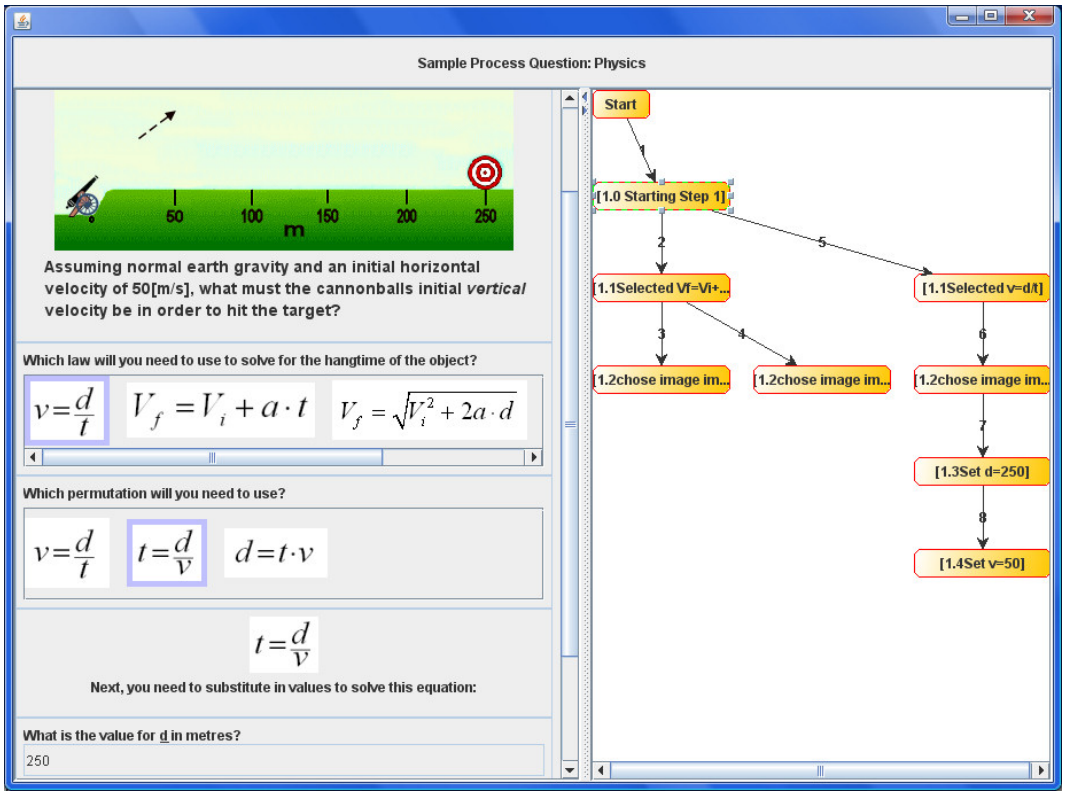

Figure 3.12: An example of a Process Tree visualization. Edges are numbered so as to make it even clearer as to the order in which the substeps were completed.

However, the Process Tree visualization is not only useful to instructors after the problem is completed, but also of great use to the student whilst they are solving the problem. In the Process Analyzer, the Process Tree is dynamically generated as the student works through the problem. Each node in the tree represents a state that the student was in for a certain Question Item step. Students are free to return to any past state they were in simply by clicking on it in the Process Tree

visualization. The Process Tree Visualization will then create a new branch as the student continues solving from that state. This way, no student work is ever lost; if a student abandoned an approach but then later realized that it was the correct approach, the student can click on the state where he or she left off and continue working from there. The Process Tree nodes are annotated by the question author (i.e. instructor) for easier identification of past states.

Finally, after the student has completed the Process Question, the entire Process Tree is submitted to the educational server-side database for automated assessment by server-side scripts. If a CAA module is implemented on the server, the next Process Question may be selected adaptively too (although instructors may choose to embed assessment logic directly into the Process Analyzer itself). Instructors may then use data visualization tools to visualize the Process Trees of a single student or perhaps summarized Process Tree data from the entire class. For example, if step 1.0 had 30 branches coming from it but step 2.0 had 80 (when summed up across the entire classroom), that would indicate to the instructor that students are struggling a great deal with either the concepts tested for in step 2.0, or perhaps the way the Question Item is set up or presented needs to be improved. The instructors can then quickly narrow-in on these issues at a glance and take corrective action immediately.

3.3.1 Assessment model

The assessment model for the Process Analyzer is also somewhat different than conventional assessment. Back in 1998, Bennett, in his paper "Reinventing Assessment" [22] proposed his views on how computer based assessment would evolve over the coming years. He described such systems as advancing through 3 different generations of design. First generation Computer Based Assessment (CBA) would be simple in design (mainly multiple choice) but would be completely computer based and also adaptive so that the next question that the student received would be based on their previous answer. Obviously, such systems already exist today in the form of the well-known adaptive Graduate

Record Examinations (GRE) used by many universities across North America. Bennet describes the second generation as being similar to the first, but with a larger focus on incorporating high quality video, audio, animation and other innovative uses of multimedia and interaction to test a much wider range of human skills. As shown in the related works section of Chapter 1, there are already many people working on such systems, and they are now beginning to emerge into useful tools for instructors. Finally, Bennett describes the 3rd generation, or "Generation R" which will be a somewhat more radical shift as assessment becomes continuous throughout the learning process, and will no longer take the form of large exams. Instead, as students are taught, their level of understanding of the material will be continually and automatically measured and charted over time as they work through their learning activities.

Although Generation R is probably still quite some time away, it is interesting to note that the Process Analyzer shares a few concepts with the idea. For example, even when used in tutorial/teaching mode, the Process Analyzer still records all the student's actions as they proceed through the instructor's content and questions. The potential exists, therefore, for instructors to analyze the Process Trees of students not only in assessment mode, but also as they go through the various tutorials and teaching modes, and to draw some conclusions regarding the student's level of understanding or knowledge retention on the subject being taught. Such continuous assessment could one day replace large tests just as Bennett hypothesized.

Chapter 4

SYSTEM ARCHITECTURE AND IMPLEMENTATION OVERVIEW

4.1 Software Architecture

For the implementation of the Process Analyzer and its Process Editor, Java was selected as the programming language. There were several reasons behind this decision, but the main one was the ease with which Java's built-in AWT and Swing libraries would allow for the Process Analyzer to be implemented for both the online and offline environments simultaneously with only a few lines of code to differentiate between the two modes. Java also offers automatic cross-platform capability meaning educators and students can make use of the system regardless of what operating system they are running.

In terms of overall architecture, the system breaks down into 3 major modules: The ProAnalyser Core, the ProAnalyser Client, and the ProEditor. Figure 4.1 shows a dependency graph outlining how these modules are associated with one another.

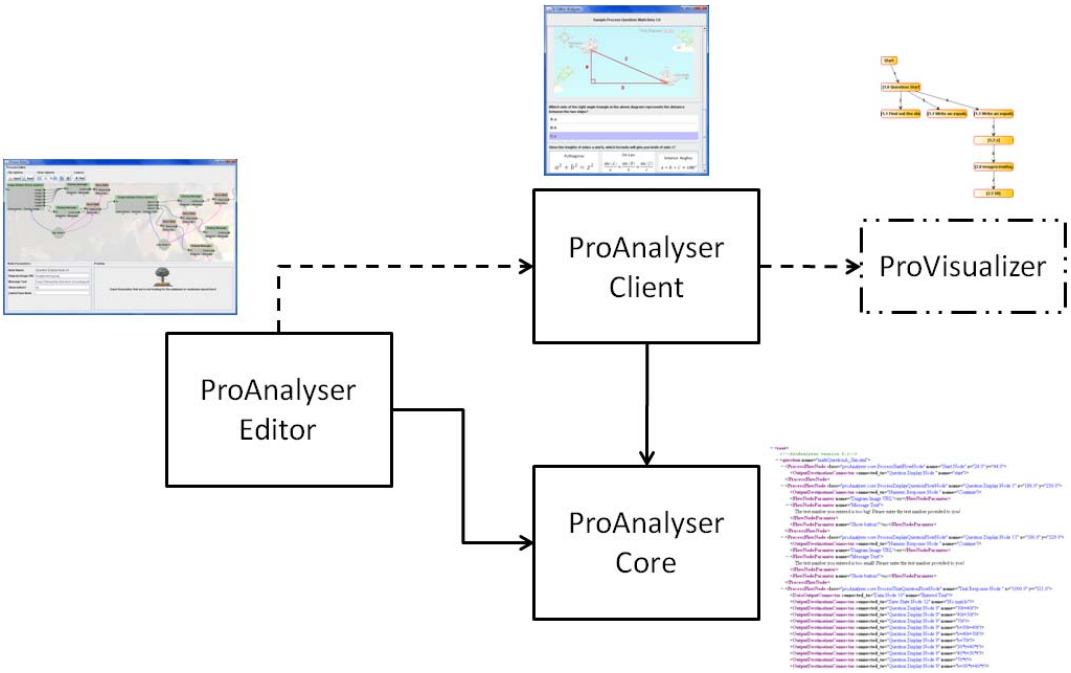

Figure 4.1: Overview of the Process Analyzer and Process Editor system Architecture. Arrows indicate dependencies and dotted lines represent optional dependencies. Note that the ProVisualizer is an optional plug-in to the client.

As shown above, the Editor and the Client are the two main user interfaces of the system, and all shared implementation between them is stored in the Core module. This modular software engineering design allows for several important advantages. For example, this means that a "Student Version" of this system could be smaller in size because it would only need the Core and Client modules (with the ProVisualizer Plug-in installed to allow students to use the Process Tree). The "Instructor version" of the Process Analyzer, however, would need to include the Editor module (and the Core) so that instructors can author questions for their students. Also, it would be most practical to include the Client module as well so that instructors can test out their questions as they make them (although this is technically optional). The modular design also allows any particular module of the system to be upgraded without concern of the rest of the system (except for large changes to the Core module of course).

These modules are all realized as Java "packages" within the development workspace and are named according to the names in Figure 4.1. The Core package contains 24 class files, the Client package contains 3 source files, the Editor contains 10, and the visualizer plug-in contains 3 (which are special in that they were implemented in collaboration with Parisa Naeimi). The various Utility classes and functions could be considered part of the Core module.

4.2 Authoring tool overview

The Process Editor is the authoring tool designed to allow instructors to create new, potentially adaptive, process-based questions without the need to learn a programming language. As shown in Figure 4.2, the Process Editor user interface is divided into 4 main components: the Toolbar, the Main Editing Canvas, the Process Node Configuration Panel, and the Process Question Preview Panel.

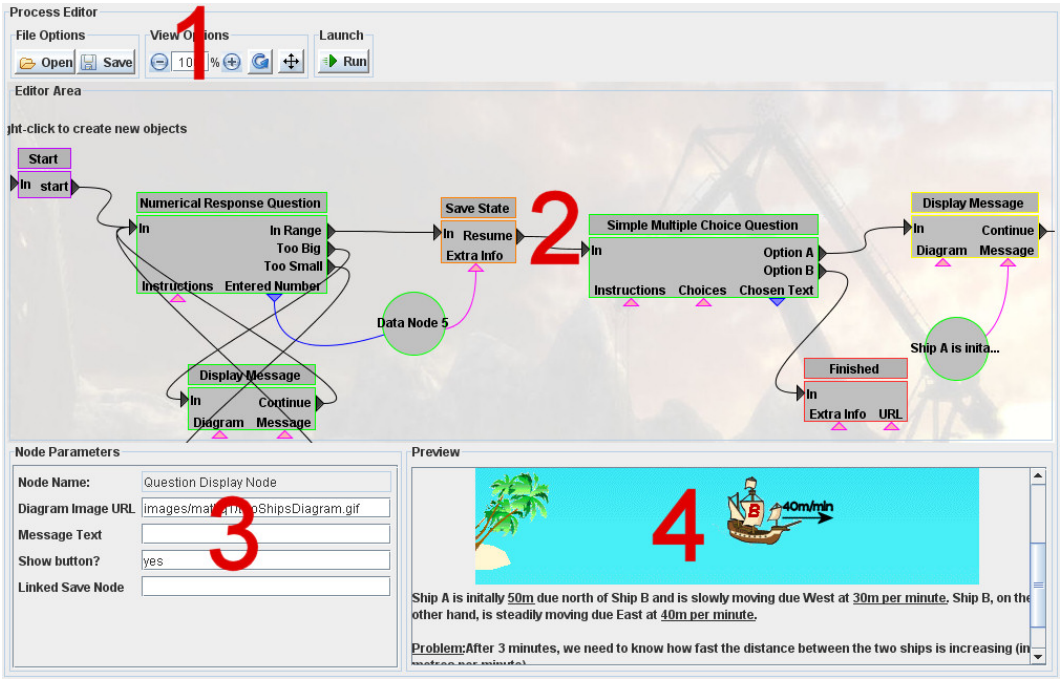

Figure 4.2: An overview of the 4 main components of the ProEditor interface.

The Toolbar (labeled "1" in Figure 4.2) provides the standard Open/Save functionality and contains various options for zooming or moving around in the Main Editing Canvas. It also contains the "Run" button under the Launch category so that users (i.e. instructors) can test out their process-based questions in a new pop-up window directly from the editor without having to embed the question in a web-browser first. The Main Editing Canvas (2) is where the process nodes are created and connected to one another. The user simply opens the right-click menu anywhere on the canvas to select a node type and add it there. Users can then click and drag a wire from one node connector to another to connect nodes with each other.

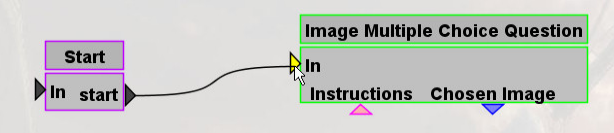

Figure 4.3: Creating a connection between 2 nodes.

Nodes can be moved around simply by clicking and dragging them, and entire groups of nodes can be selected at once by clicking and dragging a selection box over them. Selected nodes can be copy-pasted and moved or deleted en masse in this fashion. Nodes are also colour-coded so that all Question Item nodes are bright green, nodes that perform utility functions are shades of blue and also dark green, the start node is purple, and the finish node is red. Orange indicates nodes that perform special actions such as logging text to the console or saving states.

The Process Node Configuration Panel (3) is responsible for editing the properties of a selected node (or in the case of data nodes, to set the default value for that node). This panel lists all the parameters the node has as well as their current values. The user can change them simply by editing the text fields.

Finally, the Process Question Preview Panel (4) simply shows a preview of what the selected Process Question Node will look like when it comes up in the Process

Analyzer based on the current value of its parameters and/or any data nodes connected to it. This panel is blank if a node besides a Process Question Node type is selected.

4.2.1 Creating a Simple Process Question

In this section, a trivial-scale process question is created and run to demonstrate several features of the Process Analyzer and Process Editor. The goal will be to create a trivial Process question that simply asks the student to find, from a list of equations, an equation that solves for velocity. If the student finds the answer, they move on, otherwise, they are simply shown a static, non-adaptive hint.

The user begins by placing in the nodes they will need for this process question: the Start node, an Image Multiple Choice Node, and two Display Message Nodes. These can all be added through the right-click menu on the canvas as shown in Figure 4.4.

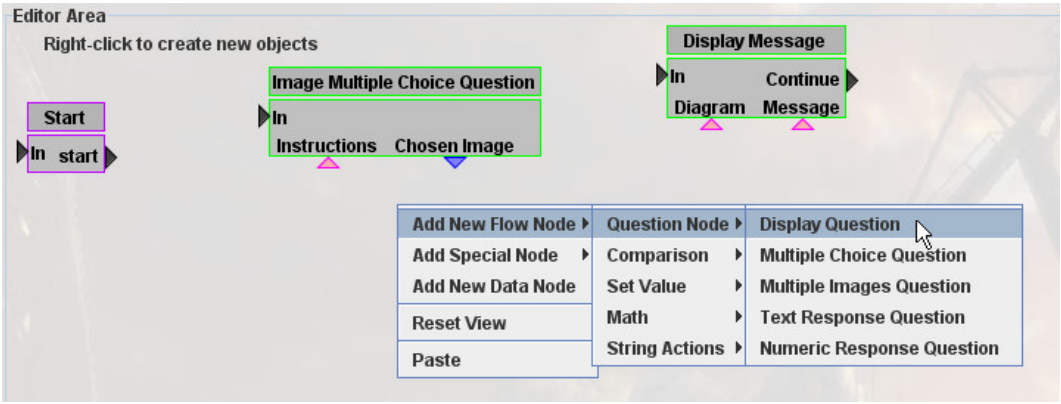

Figure 4.4: Adding Process Nodes to the Main Editing Canvas.

The Start Node indicates where to begin the process. The Multiple Choice Images Node is a subclass of Process Question Node that simply displays a multiple choice question using images instead of text so it will be used to ask our question. The Display Message Node can display Images and HTML formatted text to the screen inside the Process Analyzer so it will be used to tell the user whether or not their choice was correct.

After connecting the Start node to the Image Multiple Choice Question node, the next step would be to add the 3 image options to the “Image URLs” parameter inside the node as shown in Figure 4.5. Note that the parameter “Description Text” is also set so that a descriptive question appears above the image choices.

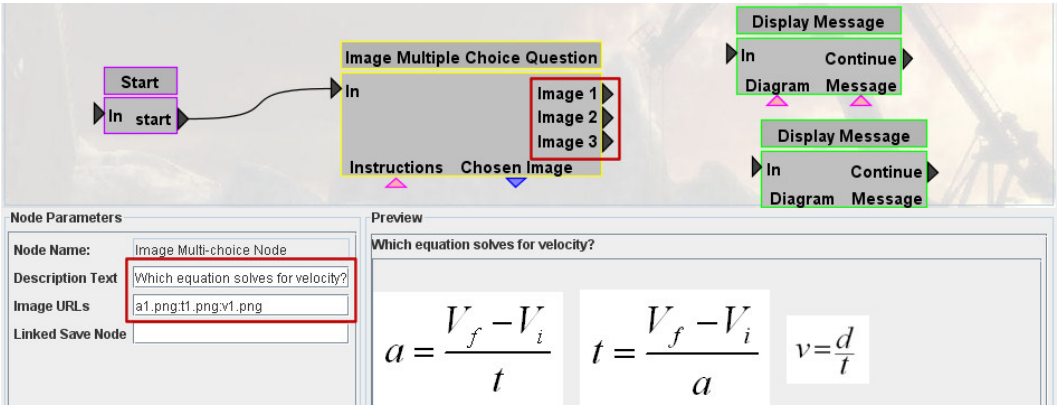

Figure 4.5: Configuring the Image Multiple Choice Question node.

As soon as the 3 options appear, the preview panel is updated immediately and the node itself suddenly has 3 new connectors, each representing what path to take next when the student chooses the corresponding image.

Next, the two Display Message nodes are configured so that the bottom one displays the message “Correct!” and the top one displays a message indicating the selected answer was not correct, and then provides a hint (after presenting an obvious big red X).

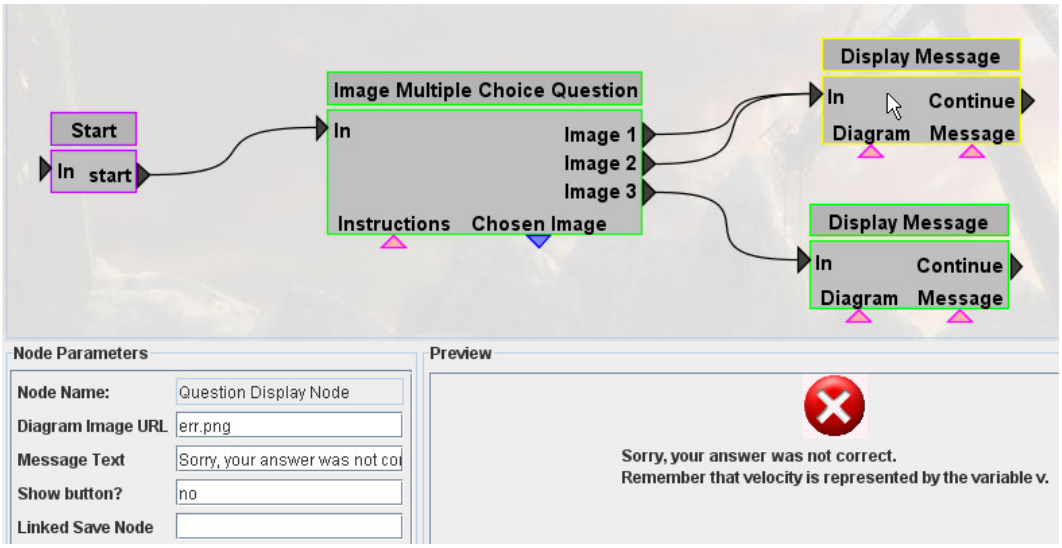

Figure 4.6: The two Display Message Nodes are configured and connected to the Image Multiple Choice Question Node.

Given that the first 2 options in the Image Multiple Choice Question were the incorrect ones, we would connect them to the top Display Message node to inform the student of this. The 3rd option was the correct one, so if the student selects it, we will instead branch off to the bottom Display Message node to congratulate the student. Finally, the user can test this system by clicking on the "Run" button in the toolbar. The results of two separate runs are shown in Figure 4.7.

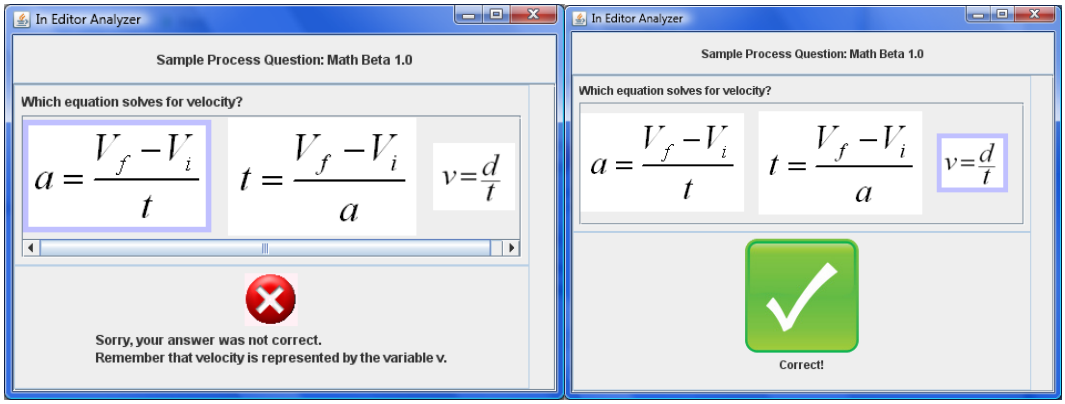

Figure 4.7: The results of two separate test runs are shown. Note that the in-editor version of the Process Analyzer does not have the Process Tree visualizer plug-in installed, but it will be available again when this process is run in applet mode.

4.3 System Architecture

In its most simplest form, the Process Analyzer is basically a platform that executes a series of Process Nodes one after the other where each Process Node performs some kind of task; be that to display a question, display a message, or perhaps something even as simple as adding two numbers together. Once a Process Node is done it simply moves onto the next one that it is connected to. If it is connected to multiple nodes (i.e. has multiple output connectors) it will choose which one to go to next based on some internal decision mechanism. On top of this basic structure, Data Nodes also allow for data to be saved from a Process Node for use later on, or to pass data from one Process Node to another. Like typical web technologies, all data is simply expressed as strings of text and it is up to individual process nodes to interpret them appropriately. For example, consider the following part of the logic section of a Process Analyzer educational game that teaches students to play the card game "21" (also sometimes referred to as "Blackjack").

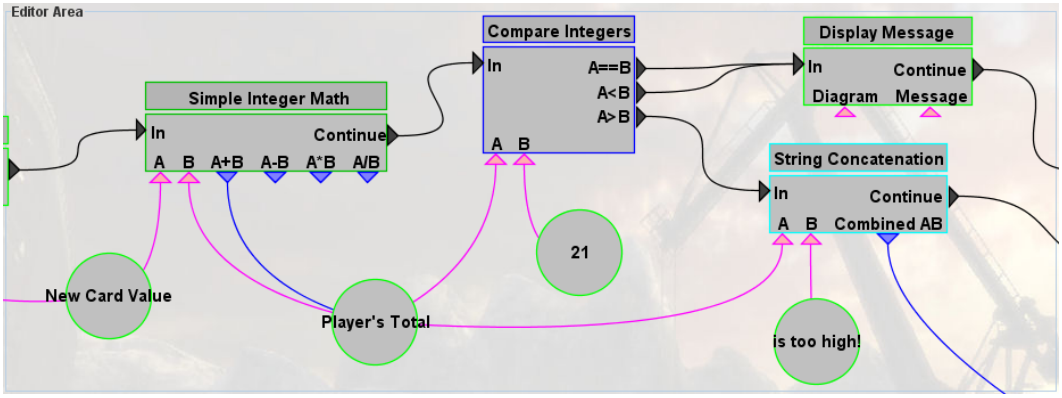

Figure 4.8: The section of the Process Analyzer Game "21" that decides whether or not the student has just lost the game.

In the card game 21, the objective is to get the total of your cards as close to 21 as possible without going over. As we can see, this section of the process starts off by using the Simple Integer Math node to take the integer value stored in the node

labeled New Card Value and add it to the existing node called Player's total. Next, a Compare Integers Node is used to compare the Player's total with the value 21. If the Player's total is less than or equal to 21, a message is shown indicating that the player is still in the game. However, if the player's total is greater than 21, the player's total is concatenated onto a message indicating that they have lost because their new total is too high, and this message is later displayed to the student before ending the game.

The flexibility this offers to instructors is enormous. For example, if a student is asked to compute a numerical answer as part of a problem and gets it wrong, the instructor could use data nodes to see if the answer is within certain ranges, and provide a hint based on how close the student was to the correct answer, or perhaps how close they were to an answer that is a common mistake (i.e. like getting a final answer that is negative instead of positive).

Formally, we might say that there is a many-to-one mapping from Process Node data connectors and data nodes because a data input or data output connector (as seen on the bottoms of the process nodes) can only be connected to single data node, whereas the data nodes themselves can be connected to an arbitrarily large number of connectors. In this context we might also say that there is a many to many mapping between the actual Process nodes because any process node can be linked to an arbitrarily large number of other Process nodes, and the number of nodes it can link to on its output side is limited only by the number of output connectors that particular node has. Process nodes can even link directly to themselves if desired.

4.3.1 Process Tree Implementation

If an instructor wishes for their process question to use the Process Tree, they simply need to indicate inside the process sequences where they wish to make updates or branches to the Process Tree. This is accomplished through the use of the special "Save State" nodes as shown in Figure 4.9.

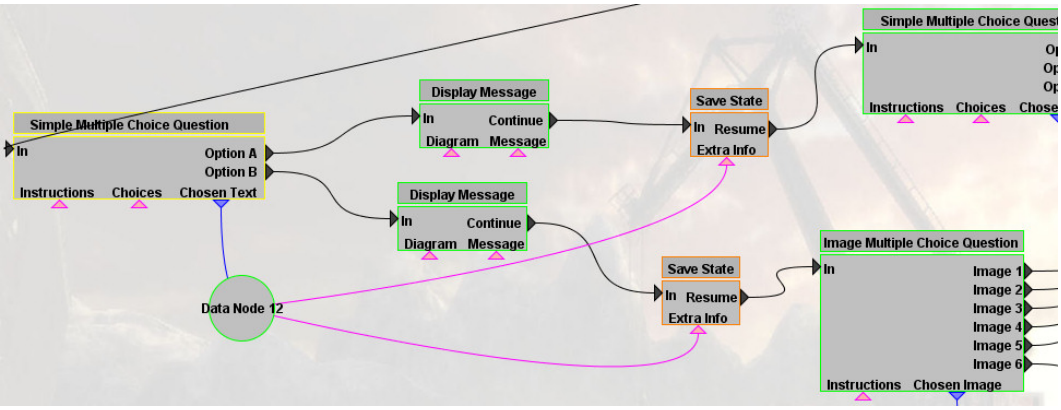

Figure 4.9: Save state nodes are placed just before question nodes so that if the student returns to the state by clicking on the corresponding node in the Process Tree, they can attempt the question item again.

Whenever the Process Analyzer encounters a Save Sate node, it adds a new leaf node onto the Process Tree representing the state the system was in at the time. If the student later clicks on that node, the Process Analyzer halts whatever action it was previously doing, unconditionally jumps back the corresponding Save Sate node, and then resumes the process question from there so that the student may attempt another approach. The reader friendly name for the process node is entered either as a parameter in the save state node, or passed in from a data node on its "Extra Info" data input connector.

4.3.2 Process Question XML Format

As mentioned before, the Process Editor saves user-created process questions in XML format. To be used with the online applet version of the Process Analyzer, the URL to this XML file simply needs to be passed in as an applet parameter. The basic structure of the XML save file is outlined in Figure 4.12.

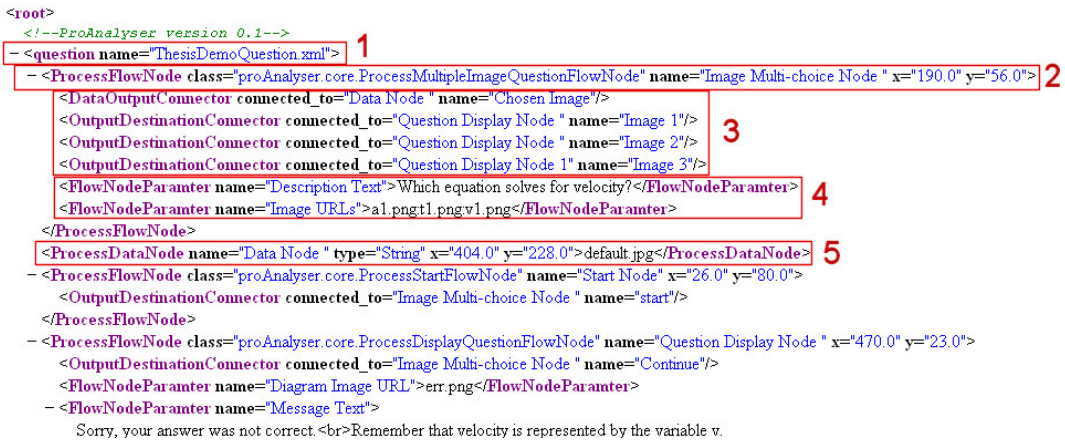

Figure 4.10: A snippet of a saved XML file that defines a Process Question.

In the simplest sense, the file is essentially just a listing of all the Process Nodes (2) and Process data nodes (5) that make up the question (1). Inside each Process Node there is a listing of all the node's connectors that are connected to another node (3). Output Destination Connectors identify which Process Node they are connected to by name (as denoted by their "name" attribute) and as such, the Process Editor ensures that all Process Nodes and Data Nodes have a unique name assigned to them (this is a requirement for correct operation). At startup these string values are replaced with actual pointers to the node objects themselves so that access can occur in constant time. Finally, each Process Node also contains a listing of all parameters that have values set which are different than their default values (4). Note that the "x" and "y" attributes for the Process Nodes and Data Nodes are only used by the editor to remember where the objects were placed on the canvas. When the question is ready for deployment online with the Process Analyzer, these values could be stripped out if file size is a concern. The XML save files are constructed using the java built-in DOM document builder and are therefore fully compliant with currently established XML standards.

Chapter 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this thesis, I covered some of the latest challenges in the field of Computer Based Assessment and e-Learning in general, and discussed the issue regarding the lack of focus on education Multimedia systems that attempt to present and assess large, complex, multi-step problems that are central to several subject areas; questions that require a process involving several intermediate thinking steps in order to solve. I then proposed the new Process Analyzer and its ProAnalyser Framework as a possible means by which to address this problem. By modeling such process-based questions as workflows within the Process Editor authoring tool, and by creating an interactive visual feedback mechanism with students via the Process Tree visualization, users of the system can visualize their own path(s) thus far, and jump backwards or forwards through their Process Trees in order to try to approach the problem from a different angle. After the student finishes, instructors can also view student Process Trees in order to learn where a student or a group of students are struggling. The future potential exists for a continuous assessment approach where students are assessed solely on the Processes they use to solve problems rather than the final answer.

To summarize, the main contributions of the proposed Process Analyzer system include:

-The ability to assess students based on the process used to solve a problem rather than simply the final answer.

-A mechanism that allows students to visualize their process trees in real-time as they solve a problem, and use this visualization to backtrack to a previous state if needed.

-The ability to passively and continuously assess students even when the system is being used for tutoring.

-An innovative multimedia authoring tool that allows instructors to intuitively model long multi-step process-based problems as workflows.

5.2 Future Work

Even though the Process Analyzer and Process Editor systems are already fully implemented, there is still a great deal of work to be done to make the system more viable on a larger scale. Plans for future work include making the system easier to use and an expanded library of Process nodes with more innovative multimedia item types. Other parts of the ProAnalyser Framework such as the Process Tree Visualization will be implemented as well.

Finally, integration with an LMS such as Moodle, LAMS, SCORM or KnowledgeTree is also a possible future direction.

Bibliography

[1] T. Volery and D. Lord, “Critical Success Factors in Online Education,” Int. Journal of Educational Management, Vol. 14, No. 5, pp. 216-223, 2000.

[2] A. Basu, I. Cheng, G. Rao and M. Prasad, “Multimedia Adaptive Computer Based Education: An Overview,” IEEE ICME Special Session, pp. 1850-1853, 2007.

[3] I. Cheng, R. Shen and A. Basu, “An Algorithm for Automatic Difficulty Level Estimation of Multimedia Mathematical Test Items,” Advanced Learning Technologies, IEEE International Conference on Advanced Learning Technologies, pp. 175-179, 2008.

[4] C. Parshall, T. Davey and P.J. Pashley, “Innovative item types for computerized testing,” in Computerized Adaptive Testing: Theory and Practice. W. van der Linden & C. Glas (Editors), pp. 129-148, 2005.

[5] WebCT WebCT Course Management System, Lynnfield, MA, WebCT, Inc., 2002, available online at http://www.webct.com

[6] Blackboard Inc. Blackboard Course Management System, Blackboard Inc., 2002, available online at http://www.blackboard.com/

[7] Moodle available online at http://moodle.org

[8] J. Dalziel, “Implementing Learning Design: The Learning Activity Management System (LAMS),” In G. Crisp, et al (Eds), Interact, Integrate, Impact: Proceedings of the 20th ASCILITE Conference Adelaide, 2003.

[9] D.H. Leo, J.I. Perez, and Y.A. Dimitriadis, “IMS Learning Design Support for the Formalization of Collaborative Learning Patterns,” ICALT '04: Proceedings of the IEEE International Conference on Advanced Learning Technologies, IEEE Computer Society, pp. 350-354, 2004.

[10] Moodle Roadmap available online at http://docs.moodle.org/en/Roadmap

[11] D. Parsons and P. Haden, "Parson's Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses," Proceeding of the 8th Australian Conference on Computing Education, Vol. 52, pp. 157-163, 2006.

[12] E. Guzman, R. Conejo and J Perez-de-la-Cruz, "Improving Student Performance Using Self-Assessment Tests," IEEE Intelligent Systems Magazine, Vol. 22, No 4, pp. 46-52, Jul/Aug 2007.

[13] S. D'Mello, R. Picard and A. Graesser, "Toward an Affect- Sensitive AutoTutor," IEEE Intelligent Systems Magazine, Vol. 22, No 4, pp. 53-61, Jul/Aug 2007.

[14] F. Culwin, "Learning beans: design, implementation & evaluation," In Proceedings of the 21st British HCI Group Annual Conference on HCI 2008, British Computer Society, University of Lancaster, United Kingdom, pp. 23-26, 2007.

[15] P. Brusilovsky, "KnowledgeTree: A Distributed Architecture for Adaptive E-Learning," In WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, ACM, New York, NY, USA, pp. 104-113, 2004.

[16] SCORM:Advanced Distributed Learning, Sharable Content Object Reference Model: Dodds, P.: Advanced Distributed Learning, Sharable Content Object Reference Model Version 1.2, Release 2001. http://www.adlnet.org, 2002-03-25.

[17] O. Bohl, J. Schellhase, R. Sengler, and U. Winand, "The Sharable Content Object Reference Model (SCORM)—A Critical Review," In Proceedings of the international Conference on Computers in Education (December 03 - 06, 2002), ICCE, IEEE Computer Society, Washington, DC, pp. 950, 2002.

[18] I. Cheng, N. Rossol and R. Goebel, "Self-Tutoring, Teaching and Testing: An Intelligent Process Analyzer," IEEE ICALT, pp. 746-750, 2008.

[19] I. Cheng, A. Basu and R. Goebel, “Interactive Multimedia for Adaptive Online Education,” IEEE MultiMedia, Vol. 16, No. 1, pp. 16-25, 2009.

[20] S Callahan, J. Freire, E. Santos, C. Scheidegger, C. Silva and H. Vo, “Managing the Evolution of Dataflows with VisTrails,” IEEE Workshop on Workflow & Data Flow for Scientific Applications 2006.

[21] VisTrails available online at http://www.vistrails.org

[22] R.E. Bennett, “Reinventing Assessment: Speculations on the Future of Large-Scale Educational Testing”, Educational Testing Service, Princeton, N.J., 1998.