Table of contents

Preface

Adaptive and intelligent Web-based educational systems (AIWBES) provide an alternative to the traditional "just-put-it-on-the-Web" approach in the development of Web-based educational. AIWBES attempt to be more adaptive by building a model of the goals, preferences and knowledge of each individual student and using this model throughout the interaction with the student in order to adapt to the needs of that student. They also attempt to be more intelligent by incorporating and performing some activities traditionally executed by a human teacher - such as coaching students or diagnosing their misconceptions. Since the first pioneer AIWBES developed in 1995-1996, many interesting systems have been developed and reported. An interest to provide distance education over the Web has been a strong driving force behind these research efforts. A good help for the research community was provided by a sequence of workshops that get together researchers working on AIWBES, let them learn from each other, and advocate the ideas of this research direction via on-line workshop proceedings (Brusilovsky, Henze & Millán, 2002; Brusilovsky, Nakabayashi & Ritter, 1997; Peylo, 2000; Stern, Woolf & Murray, 1998). A number of interesting AIWBES that were reported on some early stages of their development at these workshops have since achieved the level of maturity.

As long as the field was moving to a more mature state with a good number of developed and evaluated systems, the focus of the leading research teams has gradually moved from creating more and more new AIWBES technologies to the problems of design and authoring. It became clear that the creation of each AIWBES is an endeavor that requires considerable time and expertize even though most of the developed systems supported just one aspect of educational process. Better authoring and design support was required to bring AIWBES technology to the "real life". Several groups started advocating and developing various authoring tools and frameworks. Special attention was devoted to component-based communication architectures that allowed to re-use adaptive and intelligent components in multiple AIWBES. To support this trend and to provide a place for AIWBES researchers to discuss these emerging issues, we decided to focus the new workshop in the series on the problem of authoring tools and reusability. While the workshop was accepting a range of submissions related to AIWBES, we have encouraged the leading AIWBES teams to present their recent work on authoring design frameworks. The workshop has certainly achieved its goal in collecting an excellent set of papers with a strong focus on architectural and design issues.

In total, 12 full papers and 2 posters were submitted to the workshop. From this number, the program committee selected 6 contributions to be presented as full papers and 3 to be presented as short papers. The papers were reviewed by an international committee formed by 14 members. Each paper was reviewed by at least two referees. We do hope that this volume will be another milestone for this research direction and will serve as a source of creative ideas for the research worldwide.

Finally, we would like to express our gratitude to all the researchers that submitted their contributions. Their excellent work warrantees a high scientific level for this workshop.

May, 2005

Peter Brusilovsky
Ricardo Conejo
Eva Millán

Brusilovsky, P., Henze, N., and Millán, E. (eds.) (2002) Proceedings of the workshop on Adaptive Systems for Web-Based Education at the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'2002). Málaga, Spain: University of Malaga, also available at http://www.lcc.uma.es/~eva/WASWBE/main.html.

Brusilovsky, P., Nakabayashi, K., and Ritter, S. (eds.) (1997) Proceedings of the Workshop "Intelligent Educational Systems on the World Wide Web" at AI-ED'97, 8th World Conference on Artificial Intelligence in Education. Kobe, Japan: ISIR, also available at http://www.contrib.andrew.cmu.edu/~plb/AIED97_workshop/.

Peylo, C. (ed.) (2000) Proceedings of the International Workshop on Adaptive and Intelligent Web-based Education Systems held in conjunction with 5th International Conference on Intelligent Tutoring Systems (ITS'2000). Ösnabrück: Institute for Semantic Information Processing, University of Ösnabrück, also available at http://virtcampus.cl-ki.uni-osnabrueck.de/its-2000/.

Stern, M., Woolf, B. P., and Murray, T. (eds.) (1998) Proceedings of the workshop on Intelligent Tutoring Systems on the Web at 4th International Conference on Intelligent Tutoring Systems (ITS'98). San Antonio, TX: St. Mary University, also available at http://www-aml.cs.umass.edu/~stern/webits/itsworkshop/.

Organizing Committee

Peter Brusilovsky	University of Pittsburgh
Ricardo Conejo	University of Málaga
Eva Millán	University of Málaga

Program Committee

Susan Bull	University of Birmingham
Peter Dolog	University of Hannover
Jon Dron	University of Brighton.
Maria Grigoriadou	University of Athens
Nicola Henze	University of Hannover
Judy Kay	University of Sydney
Jaakko Kurhila	University of Helsinki
Judith Masthoff	University of Aberdeen
Tanya Mitrovic	University of Canterbury
Demetrios Sampson	University of Piraeus
Amy Soller	Institute for Defense Analyses,Virginia

The eXtensible Tutor Architecture: A New Foundation for ITS

Goss NUZZO-JONES, Jason A. WALONOSKI, Neil T. HEFFERNAN, Tom LIVAK
Worcester Polytechnic Institute
100 Institute Rd, Worcester, MA 01609
(508) 831-5569
goss@wpi.edu, jwalon@wpi.edu, nth@wpi.edu, tomlivak@alum.wpi.edu

Abstract. The eXtensible Tutor Architecture (XTA) was designed as a platform for creating and deploying many types of Intelligent Tutoring Systems across many different platforms. The XTA presently has support for state graph pseudo-tutors and JESS model-tracing cognitive tutors, in both a client and server context. Supported interfaces are presently Java Swing / WebStart and HTML. The XTA was designed with future development in mind, allowing easy specification of new tutor types, tutoring strategies, and interface layers. It has been used as the foundation of the Assistments Project, a wide scale server based ITS deployment. The Assistments Project is on track to provide ITS content to 100,000 students in the state of Massachusetts.

1. Introduction & Background

This research was conducted to develop a scalable, stable framework for deploying Intelligent Tutoring Systems (ITS) of many types to a variety of platforms. The term Intelligent Tutoring Systems covers a wide range of possible computer-based tutors, from cognitive model tracing tutors [3], constraint-based tutors [10], to pseudo-tutors. Pseudo-tutors are simplified cognitive models based on state graphs. The state graphs of pseudo-tutors are finite graphs with each node representing a state of the interface, and each arc representing an action made by a student. Student actions trigger transitions in the graph, and the current state of the problem is represented by the graph. Pseudo-tutors are behaviorally equivalent to rule-based tutors [1]. Our research attempted to support all these types of tutors, but provide a clear path for future development and customization.

Additionally, our research was dependent on the needs of the Assistments Project. This project required that we be able to support the full range of tutors, provide stability and scalability, and deliver tutoring content to a host of clients – either rich client applications such as Java WebStart, or thin light-weight HTML clients (possibly enriched by scripts and Macromedia Flash™). To accomplish these client interface goals, we were required to follow software engineering practice by cleanly separating the logic and presentation of tutors.

The success of ITS in general is well known, demonstrating useful learning effects [8]. There have been ITS that have been deployed on a wide scale [8], but they suffered from some limitations, such as a lack of centralized logging, upgrade difficulties, and tutor strategy inflexibility. It has been shown that centralized logging of student actions in databases for experimental analysis is valuable [11]. Our research sought to address these issues, as well as provide a rich feature base for future development of all tutor types.

Other projects [7] have sought to provide a rapid development environment and stable runtime for deploying individual tutoring applications. However, this approach also has shortcomings in terms of wide deployment and scalability, as well as separation of logic and presentation. We attempted to resolve these problems by creating an environment that can support many tutoring strategies (including those mentioned above), operate as both a

client and scaleable server application, provide logging capabilities for student analysis, and remain highly extensible for future development. The results of this research were used as the deployment mechanism for the Assistments Project, a mathematics ITS project based at Worcester Polytechnic Institute and Carnegie Mellon University [12].

1.1 The eXtensible Tutor Architecture

The result of our research is a framework that we refer to as the eXtensible Tutor Architecture (XTA). This framework controls the interface and behaviors of our intelligent tutoring system via a collection of modular units. These units conceptually consist of a *curriculum* unit, a *problem* unit, a *strategy* unit, and a *logging* unit. Each conceptual unit has an abstract and extensible implementation allowing for evolving tutor types and content delivery methods.

The XTA is represented by the diagram given in Figure 1, illustrating the actual composition of the units. This diagram shows the relationships between the different units and their hierarchy. Within each unit, the XTA has been designed to be highly flexible in anticipation of future tutoring methods and interface layers. This was accomplished through encapsulation, abstraction, and clearly defined responsibilities for each component. These software engineering practices allowed us to present a clear developmental path for future components. That being said, the current implementation has full functionality in a variety of useful contexts.

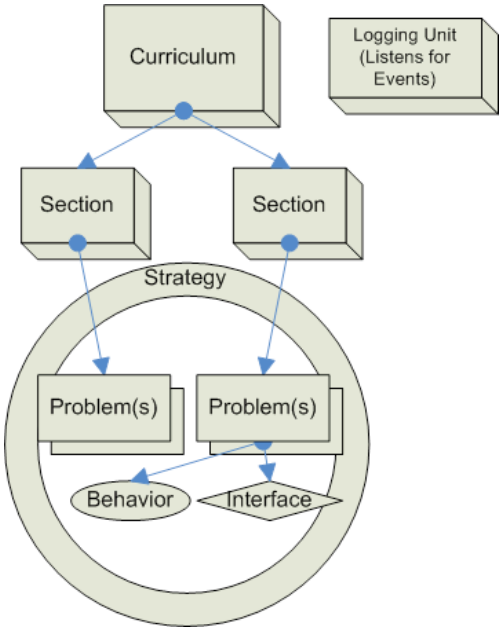


Figure 1 - Abstract Unit Diagram

1.1.1 Curriculum Unit

The *curriculum* unit can be conceptually subdivided into two main pieces: the *curriculum* itself, and *sections*. The *curriculum* is composed of one or more *sections*, with each *section* containing *problems* or other *sections*. This recursive structure allows for a rich hierarchy of different types of *sections* and *problems*.

Progress within a particular *curriculum*, and the *sections* of which is it composed, are stored in a *progress file* – an XML meta-data store that indexes into the *curriculum* and the current *problem* (one progress file per student per curriculum).

The *section* component is an abstraction for a particular listing of problems. This abstraction has been extended to implement our current *section* types, and allows for future expansion of the *curriculum* unit. Currently existing *section* types include "Linear" (*problems* or sub-*sections* are presented in linear order), "Random" (*problems* or sub-*sections* are presented in a pseudo-random order), and "Experiment" (a single *problem* or sub-*section* is selected pseudo-randomly from a list, the others are ignored). Plans for future *sections* types include a "Directed" *section*, where *problem* selection is directed by the student's knowledge model [2].

1.1.2 Problem Unit

The *problem* unit represents a problem to be tutored, including questions, answers, and relevant knowledge-components required to solve the problem. For instance pseudo-tutors are a hierarchy of questions connected by correct and incorrect answers, along with hint messages and other feedback. Each of these questions is represented by a *problem* composed of two main pieces: an *interface* and a *behavior*.

The *interface* definition is interpreted by the runtime and displayed for viewing and interaction to the user. This display follows a two-step process, allowing for easy customization to platform and interface specifications. The *interface* definition consists of "high-level" *interface* elements ("widgets"), which can have complex behavior (multimedia, spell-checking text fields, algebra parsing text fields). These "high-level" widgets have a representation in the runtime composed of "low-level" widgets. "Low-level" widgets are widgets common to many possible platforms of *interface*, and include text labels, text fields, images, radio buttons, etc. These "low-level" widgets are then consumed by an *interface* display application. Such applications consume "low-level" widget XML, and produce an interface on a specific platform. At present we have implemented a Java Swing *interface* display application, and a HTML *interface* display application that runs through a J2EE container. Because of our requirement to support HTML thin clients, our *interface* widget set is somewhat limited compared to another widget kit, such as Java Swing. However, the event model (described below) and relationship of "high-level" to "low-level" widgets allow a significant degree of interface customizability even with the limitations of HTML. Other technologies, such as JavaScript and streaming video are presently being used to supplement our *interface* standard. Future *interface* display applications are under consideration, such as Unreal Tournament for Warrior Tutoring [9] (an entirely different domain, unrelated to our mathematics project), and Macromedia Flash™ for rich content definition.

The *behaviors* for each *problem* define the results of actions on the *interface*. An action might consist of pushing a button or selecting a radio button. Examples of *behavior* definitions are state graphs, cognitive model tracing, or constraint tutoring, defining the interaction that a specific *interface* definition possesses. To date, state graph or pseudo-tutor definitions have been implemented in a simple XML schema, allowing for a rapid development of pseudo tutors [13]. We have also implemented an interface to the JESS (Java Expert System Shell) production system, allowing for full cognitive model *behaviors.* A sample of the type of cognitive models we would wish to support is outlined in Jarvis et al [6]. The abstraction of *behaviors* allows for easy extension of both their functionality and by association their underlying XML definition.

Upon user interaction, a two-tiered event model is used to respond to that interaction. These tiers correspond to the two levels of widgets described above, and thus there are "high-level" actions and "low-level" actions. When the user creates an event in the *interface*, it is encoded as a "low-level" action and passed to the "high-level" *interface* widget. The "high-level" *interface* widget may (or may not) decide that the "low-level"

action is valid, and encode it as a "high-level" action. An example of this is comparing an algebra text field (scripted with algebraic equality rules) with a normal text field by initiating two "low-level" actions such as entering "3+3" and "6" in each one. The algebra text field would consider these to be the same "high-level" action, whereas a generic text field would consider them to be different "high-level" actions. "High-level" actions are processed by the interpreted *behavior* and the *interface* is updated depending on the *behavior's* response to that action. The advantage of "high-level" actions is that they allow an *interface* widget or content developer to think in actions relevant to the widget, and avoid dealing with a large number of trivial events.

1.1.3 Strategy Unit

The *strategy* unit allows for high-level control over *problems* and provides flow control between *problems.* The *strategy* unit consists of *tutor strategies* and the *agenda*. Different *tutor strategies* can make a single *problem* behave in different fashions. For instance, a scaffolding *tutor strategy* arranges a number of *problems* in a tree structure, or scaffold. When the student answers the root *problem* incorrectly, a sequence of other *problems* associated with that incorrect answer is queued for presentation to the student. These scaffolding *problems* can continue to branch as the roots of their own tree. It is important to note that each *problem* is itself a self-contained *behavior*, and may be an entire state graph / pseudo-tutor, or a full cognitive tutor.

Other types of *tutor strategies* already developed include message strategies, explain strategies, and forced scaffolding strategies. The message strategy displays a sequence of messages, such as hints or other feedback or instruction. The explain strategy displays an explanation of the problem, rather than the problem itself. This type of *tutoring strategy* would be used when it is already assumed that the student knew how to solve the problem. The forced scaffolding strategy forces the student into a particular scaffolding branch, displaying but skipping over the root *problem*.

The concept of a *tutor strategy* is implemented in an abstract fashion, to allow for easy extension of the implementation in the future. Such future *tutor strategies* could include dynamic behavior based on knowledge tracing of the student log data. This would allow for continually evolving content selection, without a predetermined sequence of *problems*.

This dynamic content selection is enabled by the *agenda*. The *agenda* is a collection of *problems* arranged in a tree, which have been completed or have been queued up for presentation. The contents of the *agenda* are operated upon by the various *tutor strategies*, selecting new *problems* from *sections* (possibly within *sections*) within a *curriculum* to append and choosing the next *problem* to travel to [4].

1.1.4 Logging Unit

The final conceptual unit of the XTA is the *logging* unit with full-featured relational database connectivity. The benefits of logging in the domain of ITS have been acknowledged, significantly easing data mining efforts, analysis, and reporting [11]. Additionally, judicious logging can record the data required to replay or rerun a user's session.

The *logging* unit receives detailed information from all the other units relating to user actions and component interactions. These messages include notification of events such as starting a new curriculum, starting a new problem, a student answering a question, evaluation of the students' answer, and many other user-level and framework-level events.

Capturing these events has given us an assortment of data to analyze for a variety of needs. User action data captured allows us to examine usage-patterns, including detection of system gaming (superficially going through tutoring-content without actually trying to learn) [4]. This data also enables us to quickly build reports for teachers on their students, as well as giving a complete trace of student work. This trace allows us to replay a user's session, which could be useful for quickly spotting fundamental misunderstandings on the part of the user, as well as debugging the content and the system itself (by attempting to duplicate errors).

The *logging* unit components are appropriately networked to leverage the benefits of distributing our framework over a network and across machines. The obvious advantage this provides is scalability.

1.1.5 System Architecture

The XTA provides a number of levels of scalability. To allow for performance scalability, care was taken to ensure a low memory footprint. It is anticipated, based on simple unit testing, that thousands of copies of the XTA could run on a single machine. More importantly, the individual units described above are separated by network connections (see Figure 2). This allows individual portions of the XTA to be deployed on different computers. Thus, in a server context, additional capacity can be added without software modification, and scalability is assured.

The runtime can also transform with little modification into a client application or a server application instantiated over a web server or other network software launch, such as Java WebStart. Both types of applications allow for pluggable client interfaces due to a simple interface and event model, as described in the *interface* unit. A client side application contains all the network components described above (Figure 2) as well as content files required for tutoring, and has the capacity to contact a remote *logging* unit to record student actions. Running the XTA in a server situation results in a thin client for the user (at present either HTML or Java WebStart), which operates with the interface and event model of the server. Thus the server will run an instance of the XTA for every concurrent user, illustrating the need for a small memory footprint. The XTA instances on the server contact a centralized *logging* unit and thus allow for generated reports available through a similar server [4].

2. Methods and Results

The XTA has been deployed as the foundation of the Assistments Project [12]. This project provides mathematics tutors to Massachusetts students over the web and provides useful reports to teachers based on student performance and learning. The system has been in use for a year, and has had nearly 1000 total users. These users have resulted in over 1.3 million actions for analysis and student reports [4]. To date, we regularly support a live concurrency of approximately 50 users from Massachusetts schools. Additionally, during load testing, a single machine can serve over 500 simulated clients from a single J2EE / database server combination. The primary server used in this test was a Pentium™ 4 with 1 gigabyte of RAM running Gentoo Linux. Our objective is to support 100,000 students across the state of Massachusetts. 100,000 students divided across 5 school days would be 20,000 users a day. Massachusetts's schools have 7 class periods, which would be roughly equivalent to supporting 3,000 users concurrently. This calculation is clearly based on estimations, and it should be noted that we have not load tested to this degree.

Tutors that have been deployed include scaffolding state diagram pseudo-tutors with a variety of strategies (see Figure 3 for a pseudo-tutor in progress). We have also deployed

a small number of JESS cognitive tutors for specialized applications. It should be noted that the tutors used in the scaling test described above were all pseudo-tutors, and it is estimated that a much smaller number of JESS tutors could be supported.

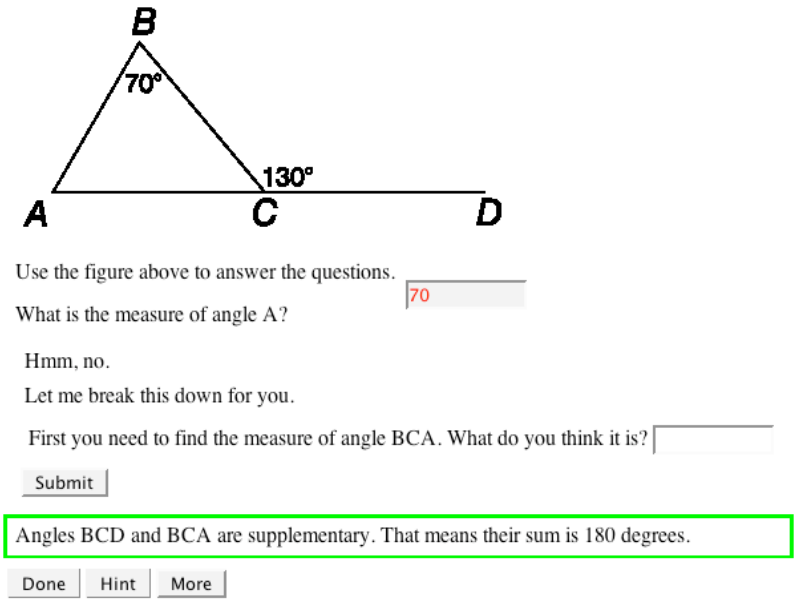


Figure 2 - Pseudo-tutor in Progress

In summary, the launch of the XTA has been successful. The configuration being used in the Assistments project is a central server as described above, where each student uses a thin HTML client and data is logged centrally. The software has been considered stable for several months, and has been enthusiastically reviewed by public school staff. Since September 2004, the software has been in use at least three days a week over the web by a number of schools across central Massachusetts. This deployment is encouraging, as it demonstrates the stability and initial scalability of the XTA, and provides significant room to grow.

3. Conclusions

The larger objective of this research was to build a framework that could support 100,000 students using ITS software across the state of Massachusetts. We are encouraged by our initial results from the Assistments Project, which indicate that the XTA has graduated from conceptual framework into a usable platform (available at http://www.assistment.org). However, this test of the software was primarily limited to pseudo-tutors, though model-tracing tutors are supported. One of the significant drawbacks of model-tracing tutors in a server context is the large amount of resources they consume. This resource consumption could be prohibitive in scaling to the degree that is described in our results without additional measures such as load balancing. Another partial solution to this might be the support of constraint-based tutors [10], which could conceivably take fewer resources, and we are presently exploring this concept. These constraint tutors could take the form of a simple JESS model (not requiring an expensive model trace), or another type of scripting language embedded in the state-graph pseudo-tutors.

Related research in our lab has yielded web-based systems for teachers to create and assign curriculums to their students, generate numerous reports on their students' strengths and weaknesses [4], as well as an application for rapidly authoring ITS content for use with the XTA [13]. Research and work in all of these areas is ongoing. Overall analysis of the system and its learning effects are covered in [12].

3.1 Future Work

Other planned improvements to the system include dynamic *curriculum* sections, which will select the next problem based on the student's performance (calculated from logged information). Similarly, new *tutor strategies* could alter their behavior based on knowledge tracing of the student log data. Also, new *interface* display applications are under consideration, using the *interface* module API. As mentioned, such interfaces could include Unreal Tournament™ (for applications such as Warrior Tutoring and other domains), Macromedia Flash™, or a Microsoft .NET™ application. All the components of the XTA were implemented in a modular and highly extensible way, so that adding new functionality is programmatically quite easy. We believe the customizable nature of the XTA could make it a valuable tool in the continued evolution of Intelligent Tutoring Systems.

References

[1] Anderson, J. R. (1993). Rules of the mind. Hillsdale, NJ: Erlbaum.

[2] Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences*, 4 (2), 167-207.

[3] Anderson, J.R., & Pelletier, R. (1991). A development system for model-tracing tutors. In *Proceedings of the International Conference of the Learning Sciences,* 1-8.

[4] Feng, Mingyu, Heffernan, N.T. (2005). Informing Teachers Live about Student Learning: Reporting in the Assistment System. *Submitted to the 12th Annual Conference on Artificial Intelligence in Education 2005, Amsterdam*

[5] Heffernan, N. T. & Croteau, E. (2004) Web-Based Evaluations Showing Differential Learning for Tutorial Strategies Employed by the Ms. Lindquist Tutor. *Proceedings of 7th Annual Intelligent Tutoring Systems Conference, Maceio, Brazil*. Pages 491-500.

[6] Jarvis, M., Nuzzo-Jones, G. & Heffernan. N. T. (2004) Applying Machine Learning Techniques to Rule Generation in Intelligent Tutoring Systems. *Proceedings of 7th Annual Intelligent Tutoring Systems Conference, Maceio, Brazil*. Pages 541-553

[7] Koedinger, K. R., Aleven, V., Heffernan. T., McLaren, B. & Hockenberry, M. (2004) Opening the Door to Non-Programmers: Authoring Intelligent Tutor Behavior by Demonstration. *Proceedings of 7th Annual Intelligent Tutoring Systems Conference, Maceio, Brazil*. Pages 162-173

[8] Koedinger, K. R., Anderson, J. R., Hadley, W. H., & Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, *8,* 30-43.

[9] Livak, T., Heffernan, N. T., Moyer, D. (2004) Using Cognitive Models for Computer Generated Forces and Human Tutoring. *13th Annual Conference on (BRIMS) Behavior Representation in Modeling and Simulation. Simulation Interoperability Standards Organization. Arlington, VA. Summer 2004*

[10] Mitrovic, A., & Ohlsson, S. (1999) Evaluation of a Constraint-Based Tutor for a Database Language. *Int. J. on Artificial Intelligence in Education* 10 (3-4), pp. 238-256.

[11] Mostow, J., Beck, J., Chalasani, R., Cuneo, A., & Jia, P. (2002c, October 14-16). Viewing and Analyzing Multimodal Human-computer Tutorial Dialogue: A Database Approach. *Proceedings of the Fourth IEEE International Conference on Multimodal Interfaces (ICMI 2002), Pittsburgh, PA*, 129-134.

[12] Razzaq, L., Feng, M., Nuzzo-Jones, G., Heffernan, N.T., Aniszczyk, C., Choksey, S., Livak, T., Mercado, E., Turner, T.E., Upalekar. R, Walonoski, J.A., Macasek. M.A., Rasmussen, K.P. (2005) The Assistment Project: Blending Assessment and Assisting. *Proceedings of the 12th Annual Conference on Artificial Intelligence in Education 2005, Amsterdam*

[13] Turner, T.E., Macasek, M.A., Nuzzo-Jones, G., Heffernan, N..T, Koedinger, K. (2005). The Assistment Builder: A Rapid Develoment Tool for ITS. *Poster, 12th Annual Conference on Artificial Intelligence in Education 2005, Amsterdam*

Design of adaptive feedback in a web educational system

Vanda LUENGO, Lucile VADCARD
Laboratoire CLIPS-IMAG, 385 rue de la Bibliothèque,
BP 53, 38041 Grenoble cedex 9, FRANCE

Abstract. This paper presents our interdisciplinary approach for the design of a technology-based learning environment for orthopaedic surgery. We present how we designed a part of this environment. This part is in relation to the web environment. The key idea is to produce an intelligent feedback in relation to the learner's activity. The didactical analysis of teaching and learning in the workplace gives a framework to build the computational representation of knowledge. This analysis is shown within the context of the apprenticeship of concepts of anatomy and orthopaedic surgery.

1. Introduction

This paper deals with the design of a technology-based learning environment in the domain of surgical apprenticeship. This work combines theories from artificial intelligence, computer science and didactic in order to model and represent didactic and expert knowledge. The application field is orthopaedic surgery, more precisely the screw placement of pelvic fractures. The technology-based environment we designed deals with the learning of declarative and decisional concepts of this surgical procedure. Gestural apprenticeship will be treated later. In this paper we describe the design of adaptive feedback for declarative knowledge.

Surgical training is usually divided into formal learning (involving the acquisition of declarative knowledge) and practical training. The gap between these two aspects of learning has been shown in some of our previous work [1]. Our aim is thus to create a learning environment which allows the use of formal knowledge before the training step in the operating theatre. It will be used in an intermediate phase of learning, between formal learning and apprenticeship in situation. It will thus provide an operative dimension of knowledge [2, 3] before the real situation, with its stress and time constraint aspects, is encountered.

Our work is inscribed in the research domain on the design, implementation and use of technology-based learning environments [4, 5].

2. Theoretical framework for adaptive feedback architecture

The "milieu" for the apprenticeship [6] must be organized to favour learning. In particular, the system must produce relevant feedback according to the learner's actions on the problem-solving situation interface. We assume that the system can produce relevant feedback if it reacts according to an internal validation of the learner's solution process. This means that we are basing the system feedback on consistency checks of learner's actions rather than on a priori solutions [7, 8], we choose to work in the Interactive problem solving support technology [9].

The architecture of our system is shown below :

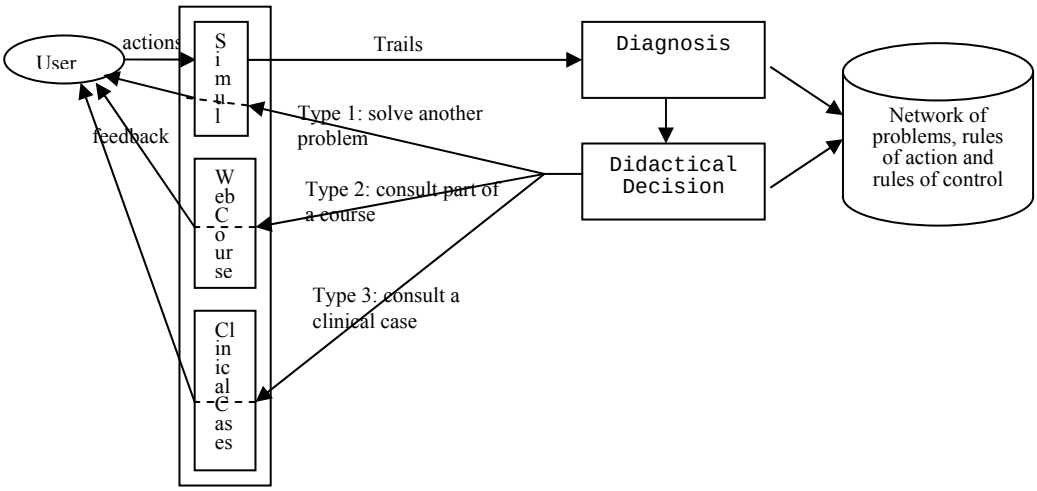


Figure 1 – Global architecture

The user solves a problem using web simulation software. Tracks of the user's actions are analysed in terms of their possible relationship to identified conceptions. A conception is an organised set of problems and pieces of knowledge. This diagnosis allows to take a didactical decision, which determines the feedback to give to the user. This feedback can be a proposition of another problem to solve, a redirection to a precise part of the online associated course, or a clinical case to consult.

Among the required components, the online course already exists; we have developed the tracking of the user's actions, and a module which allows redirection to precise and relevant parts of the course. This module allows not only syntactic links, but also semantic ones. The development of the knowledge and problems database is ongoing and will evolve with usages and future experiments. The development of the diagnosis and didactical decision components is also in progress.

In this paper we describe our methodology to design and produce the second type of feedback, feedback related to declarative knowledge represented by the semantic web. We show the construction of the knowledge model, which takes into account constraints from the knowledge analysis and from the computing aspects. This leads to an internal validation of the user's activity, taking into account his or her own problem-solving process.

3. Simulation tool description

In our learning environment, we separate the simulation component, the online course and the system component dealing with didactical and pedagogical intentions [10], [11]. To illustrate the parts of these specifications described in this paper, we briefly present an on-line training simulation of the sacro-iliac percutaneous screw placement, developed in Grenoble and available at http://www-sante.ujf-grenoble.fr/SANTE/voeu/visfran/vissage.htm (follow the links "protocole fluroscopique", then "exercice"). This exercise is designed to be close to certain aspects of the real activity of the surgeon in situation. The user is shown a 3D pelvis representation, with skin and cutaneous landmarks. He/she has to position a pin and introduce it in the body. His/her actions are unconstrained, the allowed movements are continuous, and he/she can go back (remove the pin, change its entry point, etc.). At any time the user can ask for an X-ray control. The three available orientations (inlet, outlet and lateral) are those used the orientations used by the expert in real situations. At the moment of validating the trajectory,

the user has obtained hints from the system, but not any evaluation. It thus belongs to himself/herself to use knowledge to decide on the validity of his/her trajectory.

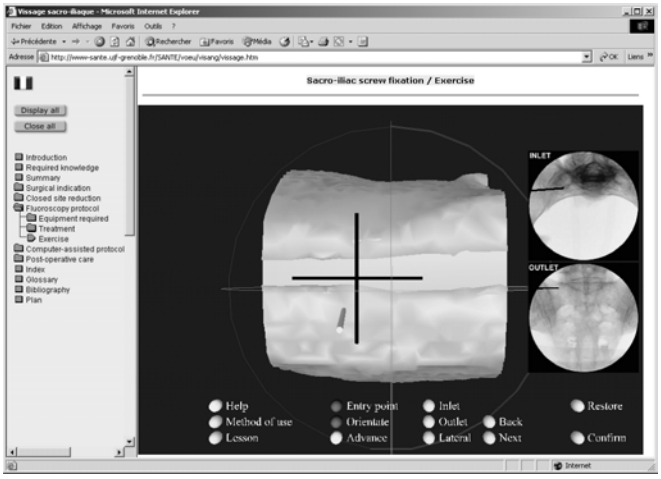


Figure 2, Screenshot of the on-line simulation module

Once the validation is made, by clicking a button, the system supplies him/her with various feedback: a "transparency" cursor which makes the skin disappear and which thus allows the visualisation of the pin's course through the bone; a qualitative judgement on the trajectory (for example "warning: extra-osseous trajectory") and a quantitative feedback on the number of attempts, the number of extra-osseous validated trajectories, the number of performed X-rays.

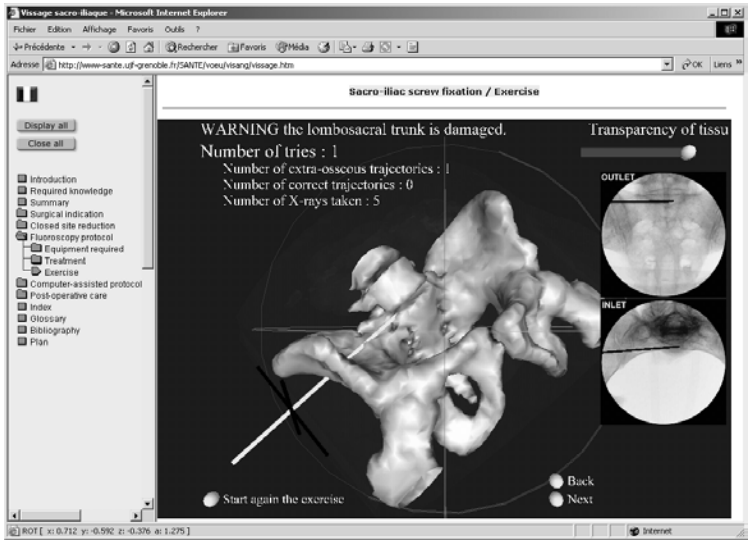


Figure 3, Various provided feedback after validation

The simulation feedback is not necessarily in terms of knowledge. Our system must intervene when it detects a didactical or pedagogical reason, and then generate an interaction. We do not want to constrain "a priori" the student in his/her activity with the simulation. On the other hand, the didactical and pedagogical system has to determine the most adequate feedback in relation to the knowledge that the user manipulates during the learning session [9].

4. Model of declarative knowledge

The aim of this research is to allow the acquisition of declarative knowledge in surgery. The adopted methodology is based on two linked phases. In the first phase, we must identify some declarative components of the surgeon's knowledge. The declarative knowledge model is based on online courses and academic documentations, and is improved by interactions between the didactical expert and surgeons. The identification of errors, for our model, is done by observation of expert and learner interactions during surgical interventions, and by surgeon's interviews. In this phase, we focus on the control component of knowledge, because we assume that control gives us key information for feedback design. This hypothesis is related to the theoretical framework of knowledge modelling, which we will present just after. During the second phase, we must implement this knowledge model in the system, in order to link the proper feedback to the user's actions

We adopt the point of view described by Balacheff to about the notion of conception, which "has been used for years in educational research, but most often as common sense, rather than being explicitly defined" [12]. The cK¢ model [12] gives a framework to didactical research for computational modelling in artificial intelligence. For brevity, we describe here only its structure and main characteristics. The first aspect of this model is inherited from earlier research in psychology and didactic: it defines a conception as a set of related problems (P), a set of rules (R) to act on these problems, and an associated representation system (L). Assuming that validation is a key aspect of conceptualisation, this model also takes into account a control structure, called Σ, which aims at making explicit a meta-level with respect to action. The crucial role of control in problem-solving has been already pointed out (by Schoenfeld for example [13]) : the control elements allow the subject to decide whether an action is relevant or not, or to decide that a problem or sub-problem is solved.

To illustrate the model functioning, we present two case studies related to declarative knowledge. This kind of knowledge comes from reference (academic) knowledge described in the online course. This course presents the planned actions to be carried out. They are schematised below:

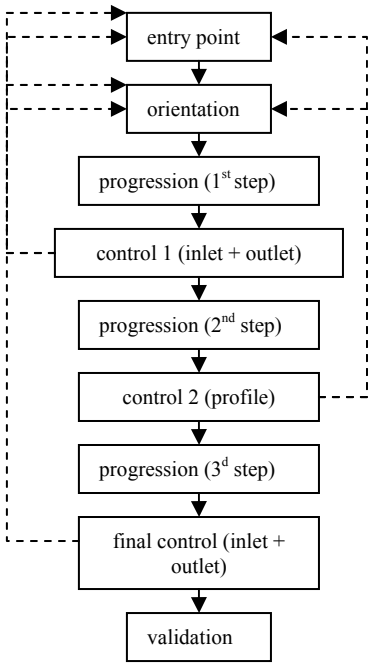


Figure 4, planned actions schema

Case-study n°1:

Let us consider a user who simulates a correct trajectory: entry point and orientation are correct (acceptable values); the three validation controls (inlet, outlet, lateral) are done and correct.

However, even if the trajectory is intra-osseous and doesn't provoke any neurological damage, we consider that the problem-solving process is incorrect in this case, because it lacks different important steps of the prescribed situation. In particular, the progression is done in a single step and thus the first control (inlet + outlet) is missing. The missing X-rays are not only missing actions, but they indicate that the user is missing an important aspect of his/her actions' validation.

The related feedback will thus be both positive: "Congratulations, your trajectory is strictly intra-osseous / number of X-rays: 3" and negative, with a redirection to the web page related to the mid-progression control criteria ("Pin progression"). We would like to produce, with our didactical component, the second kind of feedback. In this case the trajectory is correct but we would like to be sure that the student has the control criteria ("Pin progression").

Case-study n°2:

Let us consider a user who scrupulously respects all the different planned steps. This time, after the validation step, the system diagnoses an extra-osseous trajectory, probably causing neurological damages. The feedback will then be: "Warning: your trajectory is extra-osseous / number of X-rays: 5"; and will redirect the user to the web pages related to the diagnosed problems: entry point position ("Pin introduction") and validation criteria associated to the interpretation of the X-rays taken at the first control step ("Inlet control" and "Outlet control"). In this case we are sure that the student has a lack with declarative knowledge.

5. Computer representation of declarative knowledge and feedback

In this section we describe the computer system used to build our dynamic feedback in this module. The next schema illustrates the steps for producing a web feedback in relation to the online course:

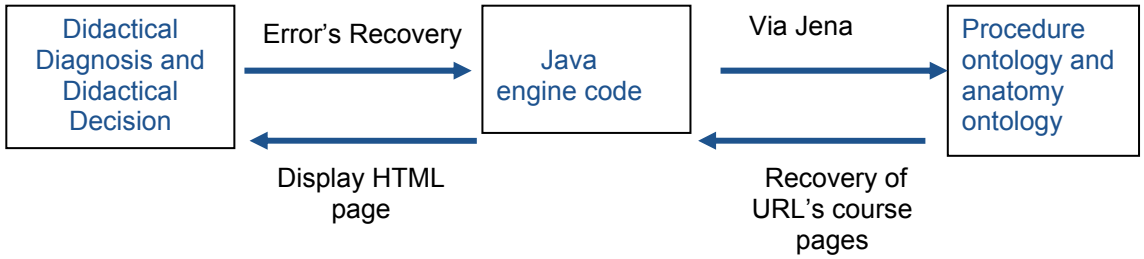


Figure 5, Technological solution for produce feedback to declarative knowledge

Our component, for produce this kind of feedback, receives, from the didactical component, the error(s) that must be considered. The error is analysed by the java program, using the ontology, and finally it produces a web page with a set of links to the online course, this set of links being related to the error(s). The Java Engine code use the open source tool Jena which offers libraries that allow to work with OWL files [14].

Our component is based on two ontologies, one related to the pelvis anatomy which is based on Standford university anatomy ontology [15], and the other one related to the screw placement procedures.

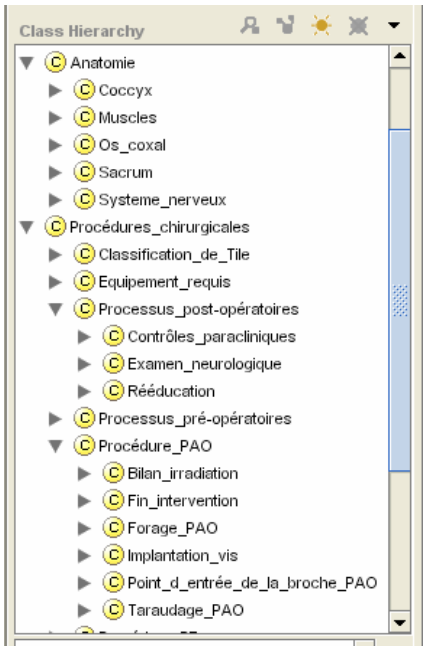


Figure 6, ontology for orthopaedic surgery learning system

In order to show how our component works, we propose to consider the two case studies presented above. In the first case study, if we give the error "pin progression", which is in relation with the procedural ontology, the java engine produces this set of links related to the procedure:

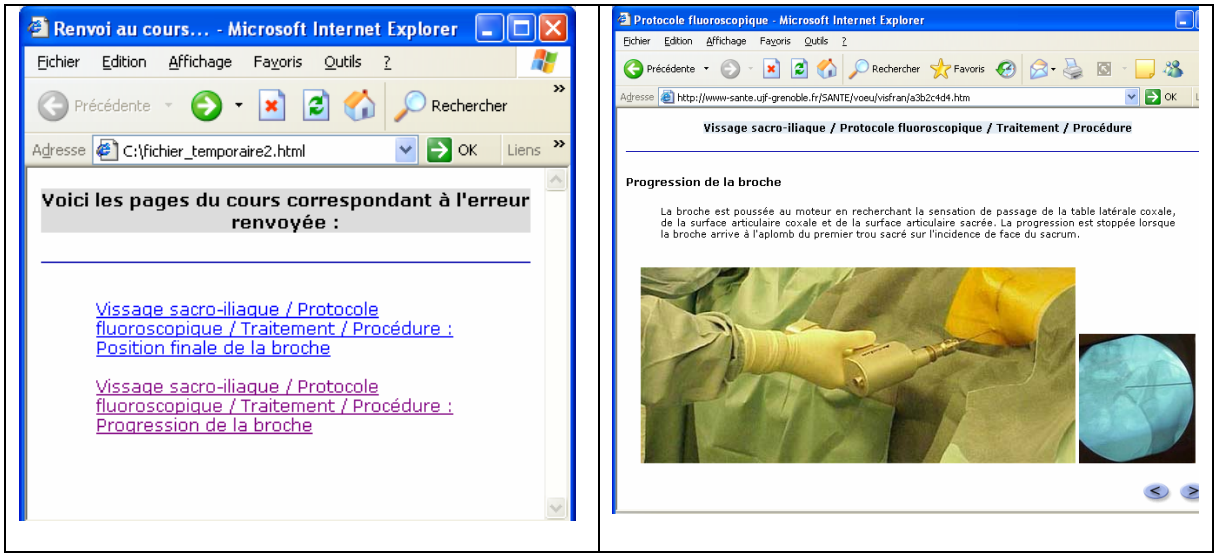


Figure 7, production of dynamic feedback in relation to the error "pin progression"

In the second case study, if we give the error "Inlet control and Outlet control", which is in relation with the anatomical ontology, the java engine finds the classes related to this error and produces this set of links (left image) which allow to go to the related links (right image):

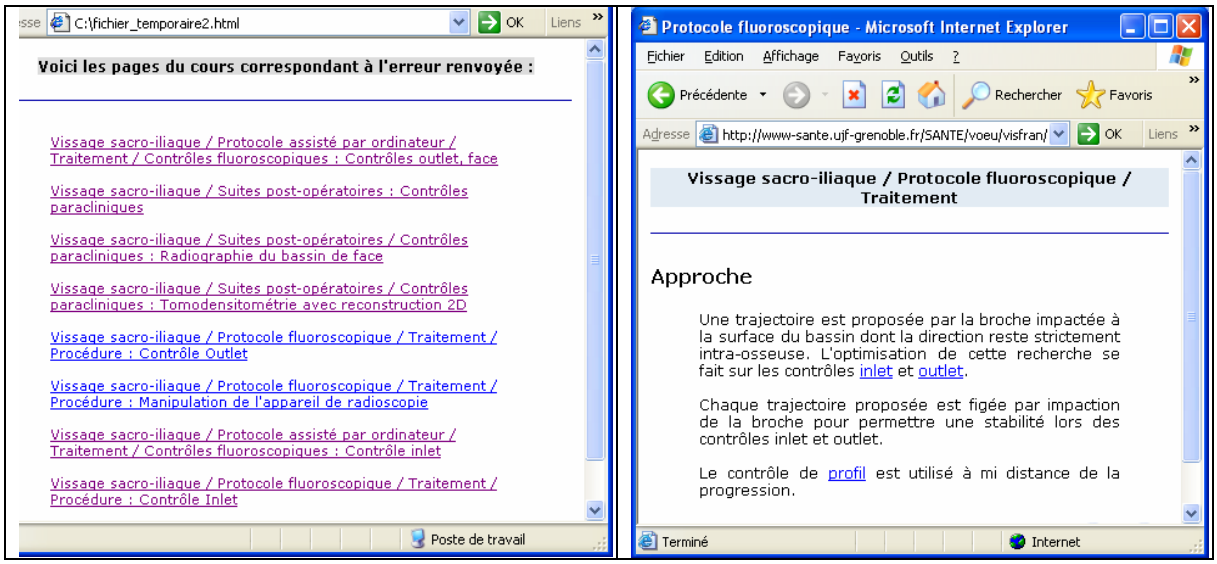


Figure 8, production of dynamic feedback in relation to the error "Inlet control and Outlet control"

We choose this methodology and architecture for interoperability, maintainability and reusability reasons.

- The interoperability reason is in relation to the integration with the others modules, i.e. the others kind of knowledge and feedbacks (see figure 1).
- Maintainability reason is in relation to the progress in the medical domain and the work with the expert for the knowledge representation. In our project the online course will be modified, particularly in relation to the procedure protocols which are continuously evolving. Also for an educational aspect, because we would like to propose an open environment where the teacher can add clinical cases and problem situations.
- Reusability reason is about the possibility to apply the same methodology to others systems or to use another validate knowledge modelling and to integrate it in the system. For example, in our case we integrate one ontology produce in Standford University (see § 4).

Protégé [16] is a good tool for edition and maintainability of the knowledge. With this tool we can edit, visualize and verify the knowledge used by our java engine. Moreover, it is designed for hierarchical structure knowledge.

The declarative knowledge is an academic knowledge validated and shared by the expert community. The ontology representation gives a good frame to describe these kinds of knowledge. Also, this knowledge can be described by hierarchical classification. The ontologies are expressed in OWL because it adds more semantic vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties, etc.

With this architecture we can generate an adapted user interface related to the error diagnosis and the learning situation. We can also respect our cK¢ knowledge model [12] based on the student's control actions. For us this last aspect is a key aspect because the others parts of the systems are modelled with cK¢, because we build the feedback in relation to the controls actions, identified by the systems.

6. Conclusion and future work

Our research deals with the design of a technology-based system for the learning of some concepts of orthopaedic surgery and pelvis anatomy. This environment provides feedback

related to the knowledge used by the user during the problem-solving activity. In other words the knowledge, in the learning situation, is the object of feedback. In this article we have presented our declarative knowledge component in a web platform. It allows integrating a model of knowledge in the system, in order to link didactical decision-making to the user's diagnosed state of knowledge.

The component system is developed in the framework of the TELEOS project [11] whose goal is a complete learning system for orthopaedic surgery. A part of this system is dedicated to the declarative knowledge and is constituted of a user interface, medical protocols and anatomy expressed in an object-based representation formalism, and one solver to find the best feedback in relation to a given error. This part of the system is accessible for the knowledge engineers and contains several modules embedded in the PROTÉGÉ architecture for the editing, visualisation and maintenance of declarative knowledge. One goal of this paper is to show that the technologies of knowledge representation are useful for maintenance of this kind of knowledge.

The current research in computer science within the TELEOS project follows two main directions. The objective of the first one is to embed this component system in a multi-agent architecture with a web server relying on the principles and technologies of the semantic web, in order to provide an intelligent access to knowledge and services that are useful for the learning situations. One of the main issues of the semantic web relies on interoperability for knowledge and applications. Thus, building a semantic web system implies a standardisation for knowledge and software components of the TELEOS system. For the knowledge bases, standardisation relies on a sharable domain model and leads to the definition of general ontologies in surgery. This kind of knowledge base takes into account the knowledge representation formalism of TELEOS with knowledge representation formalisms for the semantic web, such as OWL. The second direction is in relation to decisional knowledge: for this kind of knowledge we work with bayesian networks, which we have to integrate in the same multi-agent platform.

The first validations are planned for September 2005 and will use two complementary approaches, a qualitative evaluation of the learning situations, with a classical didactical methodology (called didactical engineering); and a quantitative evaluation of the educational added value of the technology-based learning, with a classical methodology of "with" versus "without" sets of users, measurement of progress, etc. Concerning the computational aspects of the declarative knowledge component, our evaluation will be in relation to maintainability, completeness, and flexibility of the system.

References

[1] Vadcard L. (2003), Pedagogical strategy for VOEU, Final Deliverable, European Union ProjectIST-1999-13079, http://vou-caos.vitamib.com/

[2] Vergnaud G. (1996), Au fond de l'action la conceptualisation in Savoirs théoriques et savoirs d'action, J.M Barbier (ed.), Paris, PUF

[3] Pastré P. (2002), L'analyse du travail en didactique professionnelle, Revue Française de Pédagogie, n°138, 2002, pp. 9-17.

[4] Tchounikine P, Baker M., Balacheff N., Baron M., Derycke A., Guin D., Nicaud J-F., Rabardel P. (2004), Platon-1: quelques dimensions pour l'analyse des travaux de recherche en conception d'EIAH, Rapport de l'action spécifique « Fondements théoriques et méthodologiques de la conception des EIAH », département STIC du CNRS, http://hal.ccsd.cnrs.fr/docs/00/02/69/65/PDF/Platon-1.pdf

[5] Vosniadou S., De Corte E, Mandl H. (1991), Technology-Based Learning Environments, psychological and educational fundations, NATO ASI Series F, computer and systems sciences vol. 137, Springer Verlag.

[6] Brousseau G. (1997). Theory of Didactical Situations. Dordrecht : Kluwer Academic Publishers edition and translation by Balacheff N., Cooper M., Sutherland R. and Warfield V.

[7] Ohlson S. (1994), Constraint-based student modelling, in Greer J.E. & Mac Caller I. (eds), Student Modelling: the key to individualized knowledge-based instruction, NATO ASI Series F vol. 125, Springer Verlag
[8] Luengo V. (1999), A Semi-Empirical Agent for learning mathematical proof. Proceedings of Artificial Intelligence in education (AIED 99), Amsterdam : IO Press.
[9] Brusilovsky, P. (1999) Adaptive and Intelligent Technologies for Web-based Education. In C. Rollinger and C. Peylo (eds.), *Künstliche Intelligenz* (4), Special Issue on Intelligent Systems and Teleteaching, 19-25, http://www2.sis.pitt.edu/~peterb/papers/KI-review.html.
[10] de Jong T. (1991), Learning and instruction with computer simulations. Education & Computing, 6, 217-229.
[11] Luengo V., Mufti Alchawafa D., Vadcard L. (2004), The knowledge like the object of interaction in an orthopaedic surgery-learning environment, ITS'2004 (Intelligent Tutoring Systems) (Brazil) Springer Verlag eds.
[12] Balacheff N., Gaudin N. (2002), Students' conceptions: an introduction to a formal characterization, http://www-leibniz.imag.fr/NEWLEIBNIZ/LesCahiers/2002/Cahier65/ResumCahier65.html
[13] Schoenfeld A. (1985). Mathematical Problem Solving. New York: Academic Press.
[14] Web-Ontology (WebOnt) Working Group. Web Ontology Language (OWL) Reference Version 1.0. W3C Candidate Recommendation, August 2003.
[15] *http://www.smi.stanford.edu/* (*Research Projects*, *Foundational Model of Anatomy*)
[16] N. Noy, R. Fergerson, and M. Musen. The knowledge model of Protégé-2000: Combining interoperability and flexibility. In Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management, EKAW 2000, October 2000.

Exploiting user models to automate the harvesting of metadata for Learning Objects[1]

Simon Goldrei [a], Judy Kay [a] and Bob Kummerfeld [a,2]
[a] *School of Information Technologies, University of Sydney*

Abstract. Metadata on learning objects has a valuable role to play in supporting long term reuse and adaptive selection of learning objects. Unfortunately, there are serious problems when acquiring metadata. This paper explores a new approach to these problems, by exploiting user models. We make use of the PersonisLite user modeling approach to represent both a scrutable user model of the learning object author as well as a scrutable model of the learning object itself. We harvest information available from the environment and from the learning objects themselves and, combining this with the author's user model, we build a model of the learning object. From this, we generate LOM, Learning Object Metadata. The approach has the promise of reducing the effort required to produce learning object metadata as well as providing scrutable, explainable conclusions about metadata values, an issue of particular importance in the case of subjective elements. We describe the application of this approach in Seminar, a system which makes it easy for presenters to capture their presentations and lectures so that these are readily available for viewing on the web.

Keywords. Automated metadata creation, LOM, User models

1. Introduction

Learning environments rely on content authors to provide metadata relating to the educational materials the system stores. This paper details the design and ongoing implementation of the Seminar system developed at the University of Sydney. We describe how extensions to Seminar will provide the necessary metadata for, facilitating future, relevant matches from complex user queries. Our approach is in developing effective techniques to automate the creation and harvesting of presentation metadata.

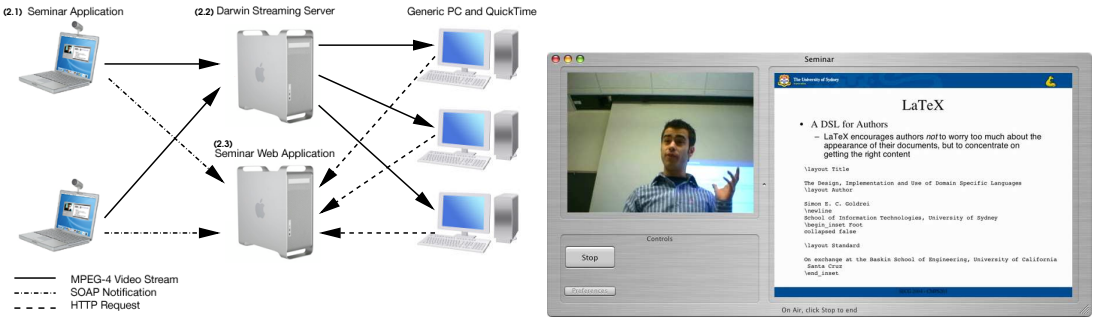
The architecture of the Seminar system involves exploiting a user model of the Learning Object presenter or facilitator. To manage user models, and resolve between multiple candidate metadata values, the Seminar system utilises the PersonisLite user modeling toolkit [16]. Metadata which is correctly, and most importantly, reliably associated with a captured presentation provides for an indexable and retrievable Learning Object. We use IEEE 1484.12.1 Learning Object Metadata (LOM) as the adopted standard for learning technology present in the Seminar System [12,10]. LOM is a cataloging scheme, consisting of nine categories, used to describe the content of a learning object and its use [13].

In developing the Seminar system, we demonstrate a tightly integrated solution where a presenter need only run a background application that surreptitiously harvests metadata with little or no user intervention. Harvested metadata is complemented with information from a user model to provide both relevant and accurate metadata.

In Section 2 we introduce the components of the Seminar system in its capacity as a Learning Object authoring tool. We also show in this section how the Seminar system provides convenient access

[1]This research is jointly sponsored by Apple Computer Australia, the Apple University Consortium under a grant from the Apple University Development Fund.

[2]Authors' addresses: S. Goldrei, J. Kay and R. J. Kummerfeld, School of Information Technologies, University of Sydney, NSW 2006, Australia.; Email: {simon, judy, bob}@cs.usyd.edu.au



(a) System overview

(b) The Seminar Application as seen by the author (a presenter)

Figure 1. Seminar System

and re-usability from an end user's perspective. In Section 3 we discuss, in an architectural sense, our approach in applying the user model of an *author* in determining reliably the metadata for the Learning Objects that they create. In Section 4 we discuss some of the techniques that we have either already employed or are planning to explore to build such a user model. Finally, in Section 5 we analyse the potential validity of our harvesting techniques and their amenability to elements of the Learning Object Metadata standard [10]. We conclude in Section 6.

2. The Seminar system

Seminar is a three part system for the real-time broadcasting and archival of Learning Objects that can be used in a range of situations such as conference presentations, research group meetings, lectures, or even segments of presentations. Seminar provides a simple, uniform method to capture a computer-based presentation regardless of the software application used in the presentation. Seminar captures the video and audio of the presenter, as well as a motion capture of the computer's main display.

The Seminar System consists of an application written for Mac OS X [1], a streaming media server and a web-accessible database. Presentations captured with Seminar can be viewed on any platform that Apple's QuickTime Player [2] supports. Figure 1(a) shows an overview of nodes that comprise the Seminar system. Also shown are the types of messages, requests and data streams that are transmitted between each of the system components. In the following subsections we describe each of these components.

2.1. Seminar Application

The basic functionality of the Seminar application is the capture of video and audio of the presenter along with a video stream of the presenter's computer display. Shown in Figure 1(b) is the interface of the Seminar Application as seen by the seminar presenter. The resulting three media streams (two MPEG-4 video [9] and audio) conform to Apple's QuickTime format. As a video source, Seminar supports any video capture device that has a corresponding VDIG (video digitiser) [3] component for QuickTime, often referred to as a device driver. Supported devices include FireWire (IEEE 1394 [4]) DV Cameras, some USB web cameras and IIDC [4] devices such as an Apple iSight [5]. Seminar's screen capture is directed at the system's main display; this is not user controllable. The Seminar Application relies on the Seminar Screen Capture component. The Seminar Screen Capture component is installed as a QuickTime VDIG pluggable system component. The Seminar Screen Capture component, responsible for the motion display capture, provides the Seminar Application a video loop-back

(a) Seminar Web Application, showing archive contents

(b) A user view of a Seminar Learning Object as displayed in a browser.

Figure 2. Seminar Web Application.

of the system display. To QuickTime, the Seminar Application, and indeed other video software on OS X, this software-only device driver appears like a system video source like any other.

2.2. Streaming Server

As shown at the top of Figure 1(a), Seminar interacts with the Darwin Streaming Server [6] from the open source Darwin project [7]. The streaming server is used to reflect (rebroadcast) the three streams to requesting QuickTime players on a network. Typically, the Seminar application is configured to uni-cast directly to this server. The node providing the Darwin Streaming Server, and maintaining the Seminar archive, is also responsible for servicing on-demand streams from the archive. This is shown as the three solid arrows at the right of Figure 1(a), representing delivery of separate uni-casts to the three systems.

2.3. Seminar Web Application

The Seminar Web Application, shown in the lower middle of Figure 1(a) is accessible through a web-browser, and provides a publicly accessible interface for listing live, scheduled and archived seminars. It facilitates users joining live seminar broadcasts or watching archived seminars on demand. The interface for the Seminar Web Application is shown in Figure 2(a). In this screen-shot we have listed five archived seminars (each being a Learning Object). To view a seminar, the user, a learner selects the seminar title which links to the on-demand playback, as indicated by the three broken lines at the right of Figure 1(a), each representing a connection to the Seminar Web Application.

The Seminar Web Application, dynamically generates an appropriate SMIL [8] document when clients request to join a broadcast, or when clients request a seminar on demand. The layout of the resulting SMIL document can be altered by modifying a system template, the default template when rendered is shown in Figure 2(b). This is the view of the learning object from the user's perspective.

It should be noted that typically one would not need to deploy the Seminar Web Application on a machine that is used solely for seminar capture.

3. Harvesting metadata for Seminar Learning Objects

Seminar's initial goal was as the authoring tool described in Section 2. To support categorisation and searching within the archive, the system needed to also harvest metadata about each seminar. It was

our goal to explore techniques for automatic generation while also overcoming the issues of metadata relevance and correctness as identified, for example, by Duval and Hodgins [12]. The LOM document generated by Seminar describes the educational content of the Learning Object as well as the aspects of the object's creation, such as author, times of modifications, revisions and technical requirements. Before detailing our architectural design for harvesting Learning Object metadata automatically, we discuss the key motivational factors for our approach.

Element relevance

To provide effective searching of Learning Objects in a repository, and return to the user relevant materials, Learning Object authoring tools need to provide metadata that is not only reliable but also *well suited* to the particular Learning Object and its content domain. Many implementations of the LOM standard [15] have identified that some elements are more directly applicable than other elements. These implementations therefore provide only a subset of the standard which can be thought of as a limitation on the description scope. Mohan and Brooks in [13] contend that the varying scope of individual elements "threaten to cause considerable interoperability problems". We propose to explore an automated metadata approach that operates based on the relevance of an individual metadata element for a particular domain of learning. Once aware of the level of relevance, the system can determine if the element's inclusion in the description is necessary and worthwhile. An automated approach that ignores this approach may describe the Learning Object with superfluous, irrelevant or in a worst case inaccurate metadata.

We believe that the use of an author's user model will support scrutable processes for inferring metadata. If sufficient evidence exists in a user model in support of an individual LOM element then the inclusion of that element in the Learning Object description is indeed relevant.

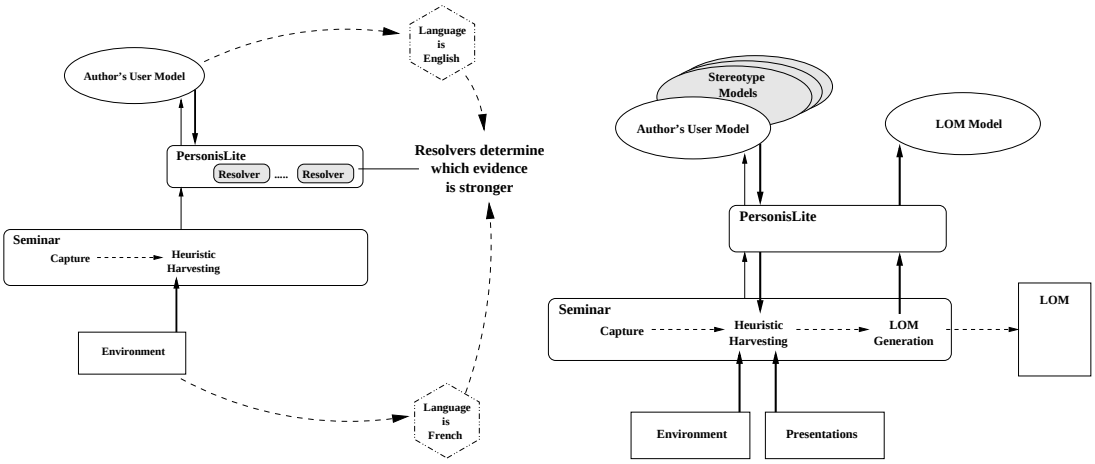
Consider a simple example, where our author's user model provides a lot of evidence that our author has a background in "16th Century France" (perhaps they have previously authored a paper with that in the title). Yet there is only superficial, heuristically harvested, evidence from the environment that this Learning Object is on "16th Century France". Then, regardless, the system can with generally good reliability include automatically, say, the General.Coverage element with the value "16th Century France".

Element subjectivity

We agree with Duval and Hodgins in [12] that the subjectivity of metadata elements is a feature and not a problem. A LOM modeling approach, enables for the model's evidence to be considered or weighted differently under different circumstances. Consider the scenario where students in computer science recommend "Introduction to programming in Python" whereas students from the business school may not find that particular learning object as relevant. In this case, we would like to consider this subjective evidence differently.

Effort to add metadata

A major criticism of the LOM standard is that content authors are unwilling to invest the considerable effort to first determine which LOM elements are relevant and then add all the metadata to their Learning Object [14]. The amount of effort that is required is a real impedance for authoring tools like Seminar, which are designed to be used in what Duval and Hodgins [12] describe as an "artisanal" setting. To illustrate this, consider where Seminar is used in day-to-day teaching, and the effort required for an end user to verify that all the description elements are correct. The solution that Duval and Hodgins proposed is for automatic techniques for harvesting metadata. Also proposed is the use of a template of re-usable metadata. Building on this previous proposal, of a heuristic approach combined with a re-usable template, the Seminar system first exploits a *user model* of the author as an initial source of evidence to infer values for metadata. We now detail the architecture of such an approach.



(a) The role of Resolvers in Personis

(b) Exploiting a user model to harvest metadata

Figure 3. User model architectures

3.1. Architecture

In the Seminar system we exploit a, possibly pre-existing, user model to accurately populate those LOM elements that are relevant to the Learning Object authors' typical teaching patterns. The premise is that a user model defines not only the characteristics of, say, the author's language and perhaps biographical details but more significantly the user model also reveals relevant metadata about, for example, the author's field of expertise.

Currently the LOM standard does not allow for the *sources* of LOM elements values to be stored. We are not suggesting here that such sources should be stored in a metadata collection. Rather the relative reliability of such sources should be somehow scrutable. When authoring a Learning Object we would like to employ a mechanism, that if a particular source is unavailable, unsuitable or unreliable it is omitted from the LOM. This omission can be for one of two reasons; either the data is simply unattainable, or it doesn't meet some level of confidence.

To achieve this goal of reliable harvesting of metadata, the Seminar Application utilises the PersonisLite [16] user modeling toolkit. For each element defined in the LOM, PersonisLite is instructed to determine if the user model can satisfactorily *resolve* some attribute of the user. Resolvers are application definable components in the PersonisLite toolkit. Their responsibility is to interpret multiple, possibly conflicting and candidate, pieces of *evidence*. Evidence in this context are candidate LOM element values. Seminar provides PersonisLite with a set of predetermined and appropriate Resolvers for each element in the LOM standard. Figure 3(a) shows an example scenario, where harvested metadata from the environment or presentation software provides evidence that the Learning Object is in French, where the user model suggests the the Learning Object is in English. Such a scenario can arise if say an English speaking teacher is presenting in France.

4. Building the user model

In order to provide reliable and suitable metadata for Learning Objects, using the mechanism described in Section 3.1, the Seminar system exploits a user model. The purpose of the user model, as previously mentioned, is to provide accurate LOM element values from multiple sources of candidate evidence. From a user model Seminar can resolve metadata about many LOM elements, such as the Learning Object author, affiliations and area of expertise.

Alternatively, should no user model exist, the Seminar system defaults to a *stereotypical user model* which provides general evidence to resolve LOM element values. For example we illustrate the stereotypical model as follows: the Seminar system is deployed in the Music department at the University of Sydney. The stereotypical user model would provide evidence that Learning Objects authored in this environment are:

- Aimed at a tertiary level audience:
  Education.Context : "higher education"
  Education.TypicalAgeRange : "18-"
- Authored by the Music department:
  LifeCycle.Contribute.Role : "content provider"
  LifeCycle.Contribute.Entity : VCard for Music department, University of Sydney
- The location is the University of Sydney and
  Technical.Location : "http://www.sydneyseminars.com"
- The subject discipline is Music:
  General.Coverage : "Music Theory"

To supplement, either a default user model or a pre-existing user-model, Seminar harvests a number of additional sources in a heuristic manner. The metadata sources that have been used in the development of Seminar, producing element values that seem promising, along with other possible sources are:

- **Presentation software** (the Learning Object itself): Slide content; Presenter's notes; Keywords: format, authors, title as recorded when editing
- **Operating System**: Default language; Time zone cities; Full name of user
- **Author's web page**: Author's biographical background; Presentation abstract(s)
- **Seminars** (the captured media): Media format details; Duration

Resolvers supplied to PersonisLite determine whether metadata harvested heuristically by Seminar or pre-existing in the user model are to be used for the LOM element values. By combining harvested metadata with either a stereotypical or a pre-existing user model PersonisLite and Seminar are able to generate automatically LOM in addition to providing a more comprehensive and re-usable user model. Relevancy and accuracy is governed by the quality of the user model. This process, resulting in LOM document generation, is presented in Figure 3(b). In this diagram, we see that Seminar harvests candidate metadata from the seminar capture, the environment and the presentation software shown in the bottom left. Harvested data is contributed to the author's user model, shown in the top left, as candidate evidence allowing PersonisLite to resolve each element value. Resolvers in the Personis system deem which candidate evidence is more reliable. Once each element has been resolved Seminar can generate the LOM document, shown middle right. PersonisLite maintains a persistent LOM model, shown in the top right, with all the candidate evidence for future revisions of the Learning Object. The LOM model can be use to explain the process used in gathering the metadata.

With Figure 3(b) now in mind, consider the scenario, where Seminar heuristically determines who the author of the Learning Object is. One such method is to harvest from the environment (operating system) the currently logged in user, and locate their user model. The author's user model provides evidence that our author has a background in "France during the middle ages." In addition, the presentation software provides, more substantial evidence (than provided by the user model) that the title is "16th Century France". This introduces some contention, as to which is the more reliable value for **General.Title**, which a Resolver will need to evaluate. One of the stereotypical user models provides evidence the author is employed as an educator at a tertiary level and at the "University of Sydney" providing values for **Educational.TypicalAgeRange** and a **LifeCycle.Contribute** entry. Finally, the system's (say, Mac OS X) Address Book facility provides a VCard [11] suitable for a separate **LifeCycle.Contribute** entry for the currently logged in user. By combining all this information using the Personis user model approach, the system is able to resolve the Learning Object title as "16th Century France" (title in this example was the only element in contention). In addition Seminar using Personis is able to generate a more complete LOM model, with supporting evidence for each element, that can be used to generate the final LOM document.

Src	Element	Amenability
SA	1.1 Catalog & Entry	The Seminar Application always maintains the URI for the Seminar Web Application archive.
PS	1.2 Title	The presentation software can offer this value simply as title metadata.
PS	1.2 Title	More reliable than the title metadata, is any heading styled text on the first slide.
Env	1.3 Language	The operating system can return the current locale default language.
UM	1.3 Language	Language evidence from the User Model is preferable over the operating system locale.
PS	1.3 Language	The most reliable source of language evidence is the metadata from the presentation software.
UM	1.4 Description	Author's bio usually provides areas of expertise, which relate to the Learning Object description.
PS	1.4 Description	Heading text from each slide, providing an 'overview'
UM	1.6 Coverage	Combining expertise from the UM and the resolved Title (1.2) to derive common terms. example: UM expertise: farming, agriculture, sustainable development & Title: "Farming in 16th Century France", Then coverage should be: "Farming".
Env	1.6 Coverage	The time-zone city such as "Sydney/Australia" is least preferable.

Table 1. Effective sources of metadata harvesting: General category.

Key: SA - Seminar Application, PS - Presentation Software, Env - Operating System Environment, UM - User Model(s)

5. Amenability of metadata to LOM elements

In this section we provide some analysis of the potential validity of harvesting metadata from the sources previously presented in Section 4. We summarise in Table 1 how amenable each of the sources is to the appropriate element in the LOM standard. In this table we present only the analysis of one category, the General category, from the LOM standard due to space limitations. The General category is provided here as it was thought to be the most widely familiar and therefore useful for discussion. A technical report will, by the time of publication of this paper, detail our approach to the full standard. As an example of how to interpret Table 1 consider the last example, the Coverage element. If the Resolver for the **LifeCycle.Coverage** element, in Personis, has evidence of:

- The author's experience: "farming, agriculture, sustainable development" yielded from the user model as well as
- The title, as harvested heuristically by Seminar (possibly from the presentation software): "Farming in 16th Century France"

Then it can resolve (by deriving common terms used) the coverage as simply "Farming". The Resolvers in the Personis system implement the order of preference logic that is detailed in the rightmost column for each element.

6. Conclusion

This paper proposes a novel approach to the harvesting of Learning Object metadata by initially exploiting a user model of a Learning Object's author. We have described the architecture of the Seminar system which combines heuristic metadata harvesting techniques with either a pre-existing or a stereotypical user model. The paper describes how the Personis user modeling toolkit uses a resolver mechanism to choose amongst candidate LOM element values available in a user model.

The user model approach is both promising in its effectiveness and suitable to harvesting Learning Object metadata because of the strong relationship between a Learning Object's author and the content. The Personis user model approach is particularly advantageous as it is able to capture over time element values that would otherwise be difficult to harvest only at the point of Learning Object creation. Overall our preliminary work seems to indicate that as the quality of the author's user model increases, the accuracy of metadata increases while the scope of description narrows to those elements that are most relevant.

Acknowledgements

The following people have played active roles in the development of Seminar: Sam Bushell, Alan Fekete, Shirley Goldrei, Judy Kay, Bob Kummerfeld, Chris Mattia and Daniel Steffen.

References

[1] Apple Computer Inc, Mac OS X, http://www.apple.com/macosx
[2] Apple Computer Inc, QuickTime, http://developer.apple.com/quicktime
[3] Apple Computer Inc, QuickTime Component Manager, http://developer.apple.com/documentation/QuickTime/RM/Fundamentals/ComponentMgr/rmComponentMgr
[4] 1394 Trade Association, http://www.1394ta.org/Technology/index.htm
[5] Apple Computer, iSight, http://www.apple.com/isight
[6] Apple Computer, Darwin Streaming Server, http://developer.apple.com/darwin/projects/streaming
[7] Internet Systems Consortium Inc, Darwin Project, http://www.opendarwin.org
[8] World WideWeb Consortium, Synchronized Multimedia Integration Language (SMIL 2.0), W3C Recommendation, http://www.w3.org/TR/smil20
[9] Apple Computer Inc, MPEG-4 fact sheet, http://www.apple.com/mpeg4/pdf/MPEG4_v3.pdf
[10] Draft Standard for Learning Object Metadata, IEEE 1484, Learning Technology Standards Committee, July 2002.
[11] vCard MIME Directory Profile, http://www.ietf.org/rfc/rfc2426.txt
[12] E. Duval, W. Hodgins, A LOM Research Agenda. *WWW2003 Twelfth International WorldWideWeb Conference*, Budapest, Hungary, pp 20-24, 2003.
[13] P. Mohan, C. Brooks, Engineering a Future for Web-based Learning Objects, *International conference on Web Engineering*, Oviedo, Spain July, pp 120-123, 2003.
[14] G. Marchionini, The costs of educational technology: A framework for assessing change, *Ed-Media 95, World conference of educational multimedia and hypermedia*, Graz, Austria, 1995
[15] CanCore, http://www.cancore.ca
[16] J. Kay, B. Kummerfeld, P. Lauder. Personis: A server for User Models, *Adaptive Hypertext 2002*, pp 203 - 212, 2002

MEDEA: an Open Service-Based Learning Platform for Developing Intelligent Educational Systems for the Web

Mónica TRELLA, Cristina CARMONA, Ricardo CONEJO
Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga.
{trella, cristina, conejo}@lcc.uma.es

Abstract. As a consequence of the increasing importance of distance education there is a growing interest in the application of intelligent techniques to existing web-based educational systems. Many researchers are focusing their efforts on reusing high quality educational software and material to take advantage of their sound theoretical foundations and effective teaching strategies. In this paper, an open learning platform for the development of intelligent web-based educational systems is presented. MEDEA[1] is a service-based learning platform that allows using intelligently different integrated learning systems for instruction purposes. To achieve this task, MEDEA provides a common student model and an instructional planner. Learning resources are integrated as web services. Systems developed with MEDEA guide students in their learning process but permit free navigation to better suit their learning needs. The learning platform has been instantiated and evaluated by developing with satisfactory results two intelligent web-based systems for the study of Logic and Agrarian Economy

Introduction

Most of existing adaptive and intelligent web-based educational systems (AIES) are the result of research efforts focused on a particular pedagogical task, learning domain or teaching strategy, such as, for example, ELM-ART [16], or AlgeBrain Tutor [1]. Although less common, we can also find educational web-based tools for generic domains, like CATGlobal [12] or DCG [15]. Given the availability of all these tools, educators and course designers might wish to integrate some of them as resources of their own systems. For example, a teacher might use DCG to develop the instructional material of a LISP AIES, ELM-ART as a drill-and-practice environment for acquiring problem-solving skills, and CATGlobal as a sound evaluation of a student's knowledge. In most cases the systems' modularity is limited, and therefore reusability is almost impossible unless the system is used as a whole.

There are several examples of systems that try to integrate external software components as resources of their own systems. The first attempts can be found in Koendinger [12], Brusilovsky [6], or DeBra [10], who connected two independent systems and defined standard protocols to communicate them. More recently, systems like ActiveMath [11] or KnowledgeTree [5], based on the integration of web-educative services, are being developed.

Therefore, it seems that the next step in the integration of AIES is the definition of learning platforms that allow the creation of intelligent systems offering domain-independence, extensibility and resource reusability and interoperability. Our contribution in this direction is

[1] This project has been partially financed by Spanish MCYT projects IFD97-1286 and TIC2003-04480

MEDEA[2] which is an open learning platform for the development of AIES, based on existing tutorial web systems. MEDEA allows teachers to develop their domain specific web-based systems, and to this end provides curriculum sequencing, student diagnosis techniques, and selection of teaching tasks, whereas the execution of these pedagogical tasks (examples, exercises, tests, etc.) is performed by external tutorial systems (MEDEA *instructional resources*). Unlike AHA, ELM-ART sequels and other architectures, the MEDEA platform is defined to allow the integration, at runtime and even at the cognitive level, of any web-based tool that could be encapsulated as a web service that accomplishes a communication protocol. MEDEA is not based on a fixed set of components; and it is not a particular solution for the integration of two given systems.

1. How MEDEA works

This section is devoted to the description of MEDEA from the student's point of view. Perhaps the best way to illustrate how MEDEA works is to compare it with the behaviour of a personal educational advisor. Imagine that a disoriented student arrives at an educational institution where personal advisors are available. In a personal interview, the student and his/her personal advisor discuss his/her background, interests, preferred learning styles and other relevant educational information. As a result, the advisor can make an informed recommendation of the most suitable courses and teachers available. The personal advisor, who also stays in permanent contact with the student's teacher, keeps a record of his/her results. MEDEA performs this process of educational advising and coaching. Table 1 compares the procedures used for such a process in both learning contexts, and shows that for each action on the part of the educational advisor there is a corresponding action in MEDEA. Useful conclusions (for MEDEA's design) can be drawn from this table: actors of each action are identified, and its inputs and outputs are established.

Table 1. Educational advising in traditional learning and in MEDEA.

	Traditional learning			**MEDEA**		
Instructional step	**ACTOR**	**INPUT**	**OUTPUT**	**ACTOR**	**INPUT**	**OUTPUT**
Asking for advice to study a subject	Student	Subject		Student	Domain model	
Selecting the most adequate topic	Advisor	Student's academic records Student's personal features	Next topic to study	Instructional planner	Student knowledge model Student attitude model	Next topic to study
Selecting the most adequate teaching style	Advisor	Preferred learning style School staff	A teacher	Instructional planner	Student profile Instructional resources	An instructional resource
Performing the instruction	Teacher	Message from the advisor, including target topic and other relevant information Teacher's learning material Teacher's syllabus Student's background (previous sessions)	Student's results	Instructional resource	Message from the planner, including target topic and other relevant information Instructional resources' learning material Instructional resources' domain model Instructional resources' student model	Student model of the instructional resource
Updating student knowledge state	Advisor	Student's results from the teacher	Updated student's academic records	Student model	Partial student model from the instructional resources	Updated student model

[2] MEDEA (*Methodology and Tools for the Development of Intelligent Teaching and Learning Environments*)

2. The Architecture of MEDEA

MEDEA is composed of a domain-independent *kernel* and a set of *instructional resources* (IR). Fig. 1 illustrates the architecture and its modules, which will be described in more detail in the following subsections.

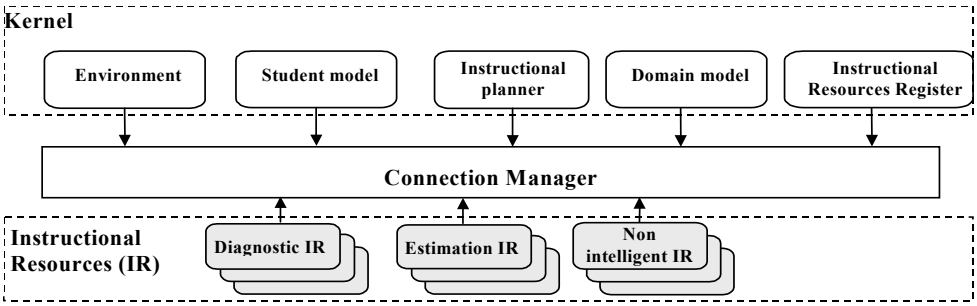


Fig. 1. The Architecture of MEDEA

2.1 Instructional Resources

These are external educational systems for concrete pedagogical tasks (electronic books, simulation systems, assessment tools, etc.). From the point of view of MEDEA, an instructional resource has 1) its own domain model, 2) a development interface for content authoring, 3) a student interface, and 4) its own student model, (which contains the relevant information to be passed to MEDEA).

MEDEA defines three types of instructional resources (IR): 1) *Diagnostic Intelligent IR*, which use some diagnosis process to establish the student knowledge level (i.e. a test system), 2) *Estimation Intelligent IR*, which use heuristics (mainly based on student interaction observation) to estimate the student knowledge level (i.e. an electronic book that stores the percentage of visited pages) and 3) *Non Intelligent IR*, which are systems that do not have a student model and consequently have no need of inferring it (for example a simulation).

Instructional resources offer the following services to the kernel system modules: 1) *init*, initiates the execution of an IR session, 2) *end*, concludes the pedagogical task and requests the *student model updating* service (2.2.3), 3) *new user*, creates a new student model in the IR, 4) *student model*, returns the IR student model.

2.2 Kernel

The MEDEA kernel includes those resources needed for a) guiding student through the curriculum, and b) selecting additional educational tasks for domain learning. These resources are described below.

2.2.1 Environment

It represents the student interface and includes a set of controls (links and buttons) that allow students to a) execute a pedagogical task, b) request an instructional plan or c) consult and modify the student model.

2.2.2 Domain Model

It stores the knowledge about the subject. The standard domain representation in web-based educational systems is the semantic network model [4]. The domain is defined by: a) a semantic network of concepts and the relations between them, and b) the pedagogical knowledge required for the instruction. In order to allow data interchange, OXML (Ontology eXtensible Markup Language) has been used. In this context, it is important to note that the domain is being modelled for pedagogical purposes, and therefore it is not strictly necessary to represent all the possible relations [2]. Thus, only four relations have been included: two of them are pedagogical (*prerequisite_of* and *subtopic_of*) and the other two are classical relations in an ontology definition (*subconcept_of* and *part_of*).

2.2.3 Student Model

MEDEA's student model composed of a *Student Knowledge Model* (which represents what the student knows about the subject) and a *Student Attitude Model* (which represents other student features that are relevant for the instructional process).

MEDEA's student knowledge model is an overlay model divided into four layers: (1) the *estimated model*, which represents what the system guesses about the student's knowledge based on his/her interaction; (2) the *verified model*, containing data obtained from evaluative components; (3) the *inferred model* from the *prerequisite* relationships; and (4) the *inferred model* from *part-of* relationships. The inferred layers are currently supported by two independent Bayesian networks (see [8] for a more complete description and an empirical evaluation with simulated students).

The student model includes the *student model updating* service, that updates the student model whenever the student performs a pedagogical task, that is, every time that an *instructional resource* is executed. *Diagnostic IR* can accurately evaluate student's knowledge about some domain topics. This information then goes to the verified knowledge model. The evaluation of *Estimate IR* is mainly based on student observation. In this case, the available information is used to update the estimated knowledge model. *Non intelligent IR* do not evaluate students in any way, so the student model updating service just informs that the student has executed that task.

2.2.4 Instructional Planner

This module provides guidance during the learning process. The adaptation during instruction is divided into two sub-processes: *microadaptation* and *macroadaptation* [14]. Microadaptation refers to which knowledge unit should be selected next, and macroadaptation determines how to present the selected knowledge.

In MEDEA, microadaptation is carried out by the instructional planner, and consists in selecting not only the concept to be studied but also the most adequate instructional resource to teach said concept. These tasks represent, respectively, the *focus concept* and the *focus pedagogical task* services. The implementation of specific instructional theories (macroadaptation) related to a concrete domain or to a particular pedagogical strategy/task (adaptive assessment, example-based learning, dialog-based reasoning, etc.) is the responsibility of the instructional resources.

When designing a planning procedure valid for any domain, it is difficult to draw conclusions from related work because many ITS sequencing techniques are strongly domain dependent and are based on heuristic rules obtained from the experience of teachers. However,

our study of related work has allowed us to identify a set of generic pedagogical facts which will be considered as design principles in MEDEA:

- An adaptive system must reduce the negative effects of a hypermedia system: cognitive overload and disorientation. MEDEA satisfies this need by means of a planner algorithm to help students choose an instructional path.
- The system must allow students to build their own knowledge structures by freely choosing the instruction sequence (constructivism). MEDEA offers curriculum sequencing, although it also allows free navigation.
- The teacher must be able to fix some sequences of the instructional path by defining prerequisites.
- Too many interruptions in the learning process can provoke discouragement and boredom[3]. MEDEA will only assess students when an important difference between estimated and verified student models is detected.
- There are two kinds of strategies to select the next concept: a) selecting the concept studied more recently until the student has learned it, or b) choosing concepts that have not been visited recently. The first one can cause boredom and consequently make the student to quit. The second one can provoke dispersion of the instructional focus and disorientation. An equilibrium between these two strategies has been reached in MEDEA by means of applying the following rules: a) the concept selected as instructional focus must belong to the topic being studied; b) the system must insist on a concept only if the student is about to learn it; and c) when selecting a concept, the interest shown by the student in learning this concept is taken into account.

2.2.5 Connection Manager

This is the module that manages all the requests and responses between MEDEA's services. This communication has been considered from two different points of view: as a distributed software problem and as a conceptual semantic issue.

From the implementation point of view, the communication problem has been achieved by using Web Services (WS). WS are software components located somewhere on the Internet that are accessible through standard protocols. The syntax to call a web service can be described using a standard XML-based language: WSDL (Web Service Description Language). A WSDL document describes what a service does, how it can be accessed, and where it is located, so that different clients can understand automatically how to interact with it. The connection manager is able to communicate with the instructional resources using the description provided in the WSDL file by the IR developer.

As mentioned above, there is also a semantic perspective of the communication problem, beyond implementation. The integration of different systems must be transparent for the student; the platform must show a unique view of the curriculum and a unified student model that summarises all the data from the component student models. To this end, relationships between the concepts of MEDEA's domain models and its components (Fig. 2) must be established. Different ways of representing student knowledge on each concept (binary, discrete, and real) and procedures for automatic type conversions have been implemented.

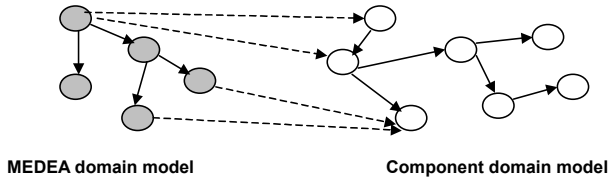


Fig. 2. Establishing the relationships between the domain models

3. Developing an AIES with MEDEA

MEDEA provides tools for the development process of an AIES. The first step consists in building the model for the subject domain. For each concept in the domain, the teacher must provide information about it, such as a name and a difficulty degree (high, medium, low), and also some data regarding its assessment: type (binary, discrete, real) and minimum value for the concept to be considered as learned. The learning material for the subject must be developed using the authoring tools of each instructional resource. This allows teachers to reuse material previously created or to develop material using their preferred tools.

The next step consists in selecting instructional resources that will be used in the AIES. Registration of an IR is performed by the resource developer, who provides the technical aspects needed to establish the connection with MEDEA. These data are used to generate a WSDL file. Once a resource has been registered, any interested teacher can use it.

As each instructional resource is also an AIES, it has its own domain model. For each selected resource, the teacher has to establish the relationships between both domain models. Once this step has been completed, the AIES is available for students through MEDEA's interface.

4. The evaluation of MEDEA

The first step in MEDEA's assessment platform consists in instantiating it to test resource interoperability. The aim is to obtain an AIES where all the resources work properly to achieve a common goal.

This formative assessment process has been accomplished in two phases. In the first phase, two educational tools (SIGUE[3] [7] and SIETTE[4] [9]) have been registered as MEDEA's instructional resources (Section 4.1). The second phase consisted in the development of two AIES: one for the study of Logic and another for Agrarian Economies. Due to limited space, section 4.2 only describes the Logic AIES, but the two courses are available through MEDEA'S interface. Independently, the accuracy of the student model in MEDEA has been tested using simulated students with satisfactory results (reported in [8]).

4.1 The integration of SIGUE and SIETTE as instructional resources in MEDEA

Most web AIES have at least an electronic book and an assessment tool, and therefore two domain independent tools were selected in order to cover this minimum functionality in MEDEA: SIGUE and SIETTE.

[3] http://www.lcc.uma.es/sigue
[4] http://www.lcc.uma.es/siette

SIGUE is an authoring tool for developing and deploying adaptive courses using existing web pages. SIGUE makes estimations of their knowledge based on the percentage of pages visited for each.

SIETTE is an adaptive web-based assessment system. It implements *Computerized Adaptive Tests* (CATs) based on a psychometric theory called Item Response Theory (IRT).

The developers of SIGUE and SIETTE cooperated in the component integration task. The steps needed to carry out this integration task are:

1. Implementing the *instructional resource services* listed in Section 2.1. These procedures were deployed as web services. (This step will require most of the integration effort, depending on the modularity and accessibility of the resource).
2. Providing (through a web form supplied by MEDEA) location and execution parameters of the resource web services to automatically generate the WSDL file.

According to the resource developers involved in this experiment, the main advantages of the MEDEA platform are: a) easiness of use; and b) modularity, which allows adding/removing resources without affecting the rest of the system.

4.2 The development of an AIES on Logic within the MEDEA framework

Our collaborators in this phase were teachers of Logic at the University of Malaga. The work of these teachers consisted in performing the steps described in Section 7.

The result of all this process is a Logic AIES accessible through MEDEA's student interface. The domain model has 37 concepts and 70 relations. The syllabus is composed of six main topics: *Introduction*, *Formal syntax*, *Formal semantics*, *Symbolization*, *Arguments and Calculus*. Each of them contains diverse subtopics.

The domain models in SIGUE and SIETTE (used to develop the course web pages and the test items, respectively) were defined by different teachers. These models were quite similar, but did not fully coincide with MEDEA's domain model; therefore teachers had to establish the correspondence between them.

The results of this experiment showed that the tools available for the development of AIES in MEDEA were considered useful for the corresponding tasks: SIGUE allowed the reuse of pre-existing material and SIETTE provided the AIES generated with a powerful and sound assessment tool. From the teacher's point of view, MEDEA helped in the time-consuming task of developing an AIES from scratch and was considered especially useful if pre-existing material exists and is to be reused.

5. Conclusions

MEDEA is an open learning platform for the development and deployment of adaptive and intelligent web-based educational systems. It is composed of the traditional modules of AIES architecture plus a set of external web-based educational systems (instructional resources).

MEDEA's main goals were to provide teachers with a tool to develop an AIES by taking advantage of other existing material and software, and to give students a learning environment in which they have a personal educational advisor that guides and supports their learning process. Experimental results indicate that both goals have been achieved.

MEDEA offers educators curriculum sequencing, teaching task selection and student model management. The instructional process is adapted by the instructional planner (which decides which knowledge should be selected next and how it should be taught), while the teaching tasks are left to the instructional resources.

This paper presents a general vision of the MEDEA platform. In order to evaluate its suitability, two components were integrated and then two AIES have been developed.

However, this formative assessment process should only be considered as a first step in the evaluation of the system. Once the architecture has been validated, the next step is to evaluate the system from the student's point of view. Though we are aware that the results of this evaluation will strongly depend on aspects that lie much beyond the MEDEA project (such as the quality of teaching material, strategies, evaluation processes, behaviour of the components, etc.), such results will be very useful for the process of refining and improving MEDEA.

Apart from these very important assessment issues, further research work in MEDEA is planned in different directions, such as a) implementing different instructional planners with teaching strategies adapted to particular subject domains, b) semantic description of the behaviour of each instructional resource, c) exploring how the use of intelligent agents can improve how MEDEA works, and c) applying automated reasoning techniques to use the log files to improve the system's behaviour.

A prototype of MEDEA (in Spanish) is operating at http://www.lcc.uma.es/medea

References

[1] Alpert, S. R., Singley, M. K., and Fairweather, P. G., Deploying Intelligent Tutors on the Web: An Architecture and a Example *International Journal of Artificial Intelligence in Education*, vol. 10, pp. 183-197, 1999.

[2] Anderson, J. R. The Expert Module. In: *Foundations of Intelligent Tutoring Sytems*, eds. Polson, M. C. and Richardson, J. J. Lawrence Elrbaum, 1988.

[3] Barr, A. and Feigenbaum E.A. Planning and Problem Solving. In: *The Handbook of Artificial Intelligence*, Anonymous Pitman, 1982.

[4] Brusilovsky, P. Developing adaptive educational hypermedia systems: From design models to authoring tools. In: *Authoring Tools for Advanced Technology Learning Environment.*, eds. Murray, T., Blessing, S., and Ainsworth, S. Dordrecht: Kluwer Academic Publishers, 2003.

[5] Brusilovsky, P. and Nijhavan, H., "A Framework for Adaptive E-Learning Based on Distributed Re-usable Learning Activities.," *Proceedings of World Conference on E-Learning,* 2002.

[6] Brusilovsky, P., Ritter, S., and Schwarz, E., "Distributed intelligent tutoring on the Web," *Proceedings of the 8th World Conference of the AIED Society,* 1997.

[7] Carmona, C., Bueno, D., Guzmán, E., and Conejo, R. SIGUE: Making Web Courses Adaptive . In: *Proceedings of the AH2002 Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems.*, Anonymous Springer-Verlag, 2002.pp. 376-379.

[8] Carmona, C., Millan, E., Perez-de-la-Cruz, J. L., Trella, M., and Conejo, R., "Introducing prerequisite relations in a multi-layered Bayesian student model," *UM2005 User Modeling: Proceedings of the Tenth International Conference,* (to appear July 2005).

[9] Conejo, R., Guzmán, E., Millan, E., Perez-de-la-Cruz, J. L., and Trella, M., SIETTE: A Web-Based Tool for Adaptive Testing *International Journal of Artificial Intelligence in Education*, vol. 14, pp. 29-62, 2004.

[10] De Bra, P., Santic, T. and Brusilovsky, B. (2003) AHA meets Interbook and more. In: *Proceedings of World Conference on E-Learning (ELEARN-2003),* 2003. pp. 57-64.

[11] Melis, E., Andres, E., Franke, A., Frischauf, A., Goguadse, G., Libbrecht, P., Pollet, M., and Ullrich, C., ActiveMath: A web-based learning environment *International Journal of Artificial Intelligence in Education*, vol. 12, pp. 385-407, 2001.

[12] Promissor. CatGlobal. 2003.

[13] Ritter, S. and Koedinger, K. R., An architecture for plug-in tutor agents *International Journal of Artificial Intelligence in Education*, vol. 7, pp. 315-347, 1996.

[14] Shute, V. and Towle, B., Adaptive E-Learning *Educational Psychologist*, vol. 38, pp. 105-114, 2003.

[15] Vassileva, J. and Deters, R., Dynamic courseware generation on the WWW *British Journal of Educational Technology*, vol. 29, pp. 5-14, 1998.

[16] Weber, G. and Brusilovsky, P., ELM-ART: An adaptive versatile system for Web-based instruction. Special Issue on Adaptive and Intelligent Web-based Educational Systems. *International Journal of Artificial Intelligence in Education*, vol. 12, pp. 351-384, 2001.

Rule-Based Adaptive Problem Generation in Programming Tutors and its Evaluation

Amruth KUMAR
Ramapo College of New Jersey,
Mahwah, NJ 07430, USA

Abstract. We will propose a rule-based mechanism for adaptive generation of problems in intelligent tutors. We will present the domain model, student model, and the algorithms for rule-based adaptation in the context of web-based programming tutors. Finally, we will present the web-based protocol we used to evaluate rule-based adaptation and discuss the results. Our evaluation shows that rule-based adaptation helps students learn with fewer practice problems. Rule-based adaptation has several advantages – it is domain-independent, flexible and scalable.

Keywords: Programming tutor, Problem generation, Rule-based adaptation, Evaluation

1. Introduction

We have been developing web-based tutors to help students learn programming language concepts by solving problems. To date, we have developed tutors on expression evaluation, pointers in C++, counter-controlled loops, parameter passing mechanisms, scope concepts and their implementation, and classes. The tutors present programming problems to the learner, grade the learner's answer, provide a detailed explanation of the correct answer, log the student's performance, and determine whether the student has learned the material. Our tutors address application (predicting the behaviour of a program) and analysis (debugging a program) in Bloom's taxonomy [6], as opposed to program synthesis (writing a program), which has been the focus of many earlier works (e.g., LISP Tutor [19], PROUST [10], BRIDGE [7], ELM-ART [21] and Assert [3]). Our tutors are designed to be used as supplements to traditional programming projects, as recommended by the whole language approach [15].

In this paper, we will propose a rule-based mechanism for adaptive generation of problems in intelligent tutors. It applies the traditional rule-based reasoning to adaptively generate problems in web-based tutors. We will first describe our domain and student models, followed by a description of the rule-based adaptation algorithm in the context of our programming tutors. We evaluated the rule-based adaptation in fall 2004. We will present the web-based evaluation protocol and discuss the results of our evaluation.

2. The Domain Model

We have identified a set of learning objectives for each programming topic. Learning objectives for a topic are concepts that must be understood in order to learn the topic. Preferably, these concepts are at a fine level of granularity so that problems can be designed to teach or assess them individually. For instance, the learning objectives for arithmetic expressions are:

- Correct evaluation, precedence, associativity and coercion of addition, subtraction and multiplication operators;
- Correct evaluation of integer and real division, precedence, associativity and coercion of division operator and divide by zero error;
- Correct evaluation, precedence and associativity of the remainder operator, divide by zero error and the inapplicability of remainder operator to real operands.

Typically, we identify 20-30 learning objectives per topic. Note that we also include typical errors associated with a topic as learning objectives for the topic. These are not errors in the student's application of procedures, as described in the theory of bugs [8], but rather, syntax, semantic and run-time errors that are an inherent part of the programming domain – understanding of the programming domain would not be complete without knowledge of these bugs.

We use a single unified domain model for all our programming tutors. This domain model is the concept map of the programming domain, enhanced with learning objectives. The concept map is a taxonomic map of the domain, with domain topics as nodes, and is-a and part-of relationships as arcs. The learning objectives for a topic serve as the children of the node for that topic. The domain model is a hierarchical tree, with domain topics as intermediate nodes and learning objectives as leaf nodes.

For each topic, we list the learning objectives in increasing order of complexity. Often, learning objectives are independent of each other, and can be listed in any order. E.g., precedence and associativity are two independent learning objectives for an arithmetic operator – the student can learn about one independently of the other. However, when a learning objective is dependent on another learning objective, it is listed after that learning objective. E.g., it is necessary for a student to learn nested independent loops before nested dependent loops. So, we list dependent loops after independent loops.

In the domain model, for each learning objective, we identify the level of expected proficiency. We represent the level of proficiency in terms of two measures:

- M_1 - The minimum number of problems the learner must solve on that learning objective. Some considerations for setting the value of M_1 are:
  - M_1 should be set high enough for novices to be able to learn the concept necessary to satisfy a learning objective. For instance, $M_1 = 1$ does not provide for reinforcement of learning.
  - M_1 should be set low enough that advanced students who have learned the concepts corresponding to a learning objective are not encumbered with unnecessary problems. $M_1 \geq 4$ could result in students solving redundant problems on a learning objective well after they have learned the corresponding concepts.

  Typically, we set $M_1 = 2$. For harder learning objectives, we set $M_1 = 3$.
- M_2 - The percentage of problems that the learner must solve correctly on the learning objective to be considered proficient in it. Some considerations for setting the value of M_2 are:
  - M_2 should not be set so low that students meet it without learning the concepts corresponding to the learning objective. M_2 should be greater than the greatest probability of guessing the correct answer to *any* problem for that learning objective.
  - M_2 should not be set so high that students are forced to solve additional problems on a learning objective even after they have learned the associated

concepts. Given a student who already knows the concepts associated with a learning objective, if n is the maximum number of problems the tutor might tolerate having the student solve, $M_2 = n / M_1$.

Typically, when $M_1 \geq 2$, we set $M_2 = 60\%$ - the learner must solve at least 2 problems correctly in order to satisfy a learning objective. For harder learning objectives, we set it lower (e.g., 50%).

If $M_1 = 0$, the tutor does not generate any problems for the learning objective. If $M_1 \neq 0$, but $M_2 = 0$, the tutor generates exactly M_1 problem(s) on the learning objective. Our tutors use these proficiency measures to determine whether the learner has "satisfied" each learning objective.

3. The Student Model

We use an overlay of the above domain model as our cognitive student model. But, instead of saving M_1 and M_2 with each learning objective, we save five terms that record the student's progress - the number of problems generated (G), attempted (A), correctly solved (C), incorrectly solved (W) and missed (M) by the student on that learning objective. Maintaining student progress in this raw form enables us to be flexible about how we interpret it. Currently, our tutors use the following two inequalities to interpret this data and determine whether a student has "satisfied" a learning objective:

- $A \geq M_1$ - Ensures that the student has attempted a minimum number of problems for the learning objective;
- $C / A \geq M_2$ - Ensures that the student has solved a minimum number of problems correctly for the learning objective.

Several researchers have proposed using a pre-test to initialize the student model in adaptive tutors (e.g., [1, 9]). Recently, researchers have proposed various improvements to the pre-test:

- Some researchers have proposed using adaptive pre-tests to minimize the number of problems the learner must solve (e.g., [2, 18]).
- Other researchers have proposed using stereotypes, using a shortened pre-test to stereotype the learner and initializing the student model according to the selected stereotype [1, 11].
- Another recent proposal is to use schema-based assessment of learner's knowledge to quickly initialize the student model [12]. If acquisition of solution schemas is a characteristic of expertise in a domain, express tests can be devised for the domain wherein the learner fills in incomplete intermediate stages in a solution rather than come up with the entire solution.

In our web-based tutors, we use a pre-test to initialize the student model. We chose not to use adaptive pre-tests because we wanted to compare the pre-test score with the score on a similarly constructed post-test to evaluate the effectiveness of the adaptive tutor.

4. Problem Templates

Limited problem set has been recognized as a potential drawback of encoding a finite number of problems into a tutor [16]. In our web-based tutors, we generate problems as instances of parameterized templates, a scheme similar to that found in [4, 13]. Every instance of a template is a new problem and no two problems are identical. This enables our tutors to present different instances of a template to different users at a given time (to prevent plagiarism), or to the same user at different times (for test-re-test). Whereas Belmont et al [4] have proposed templates to automatically generate problems such as true/false and fill-blanks,

we focus on the generation of debugging problems, problems on predicting the output of programs and problems on evaluating expressions.

On each topic, i.e., for each tutor, we have coded a repository of problem templates. These templates are indexed by learning objectives. Each template T_j and its associated learning objective(s) L_i constitute a rule of the following form in our template knowledge base:
If tutoring is desired for the learning objective L_i, then use template T_j.
E.g., the following is a template on arithmetic expressions:

Template No. 120
Learning Objective: /.Real.Correct
Template: 24 / <R1#integer;2<=R1<=8;#>
Type: expression

The learning objective associated with the above template is the correct evaluation of real division. The template contains a meta-variable R1, which is instantiated during problem generation to an integer value between 2 and 8, inclusive. So, the tutor may generate any of the following problems from the above template: 24 / 2, 24 / 3, 24 / 4, 24 / 5, 24 / 6, 24 / 7 or 24 / 8. Typically, we have encoded 20-25 templates per learning objective in our template knowledge base.

5. Rule-Based Adaptation of Problem Generation

We will now present the algorithm for rule-based adaptation of problem generation. This algorithm assumes that the problem templates are indexed by learning objectives and the student model is represented in terms of learning objectives.

The Algorithm:

1. Let the set of all the learning objectives on the topic be AL = $\{L_1, L_2, \ldots, L_m\}$, where L_1, L_2, …, L_m are individual learning objectives.
2. For each learning objective L_i, extract from the template knowledge base, all the templates that match the objective. Let the resulting set of templates be $T_i = \{T_{i1}, T_{i2}, \ldots, T_{ip}\}$, where T_{i1}, T_{i2}, …, T_{ip} are individual templates that match L_i.
3. Identify the list of learning objectives that the learner has not yet satisfied. Let this set be $L = \{L_1, L_2, \ldots, L_n\}$, $n \leq m$. If the set L is empty, the student has mastered this topic, exit.
4. Select the next learning objective L_j from the set L.
5. Select the next template T_{jk} from the set of templates T_j corresponding to the learning objective L_j and generate the next problem as an instance of the template.
6. After the learner has attempted the problem, update G,A,C,W and M for the learning objective L_j in the student model, as well as any other learning objective affected by the template T_{jk}. Repeat from Step 3.

Sub-algorithm for Step 4: First, we define **persistence** p as the maximum number of problems a tutor generates back to back on a learning objective before moving on to the next learning objective. Given the last learning objective was L_i, the algorithm to select the next learning objective is as follows:

1. If L_i has been satisfied, return the next learning objective in the list L_{i+1}. If $i + 1 > n$, the number of learning objectives not yet satisfied, set $i = 1$, and return L_1
2. If p problems have been generated back to back on the learning objective L_i, return L_{i+1}. If $i + 1 > n$, set $i = 1$, and return L_1
3. Else, return L_i.

Since the learning objectives are listed in increasing order of complexity in the domain model (of which the student model is an overlay), the tutor generates problems on a learning objective only after generating problems for all of its pre-requisite learning objectives. As to the value of persistence p, the limit that we introduced on the number of problems the tutor would present back to back on a learning objective:

- p = 1 means that the learning objective is changed from one problem to the next. This may not reinforce learning due to rapid switching of the learning objective.
- p = 2 or 3 helps reinforce learning since the tutor presents 2-3 problems back to back on a learning objective. However, if the student satisfies the learning objective with fewer than p problems, the above algorithm moves the student to the next learning objective.
- p > 3 may make the tutor predictable and boring. The student may begin guessing the correct answer to problems, which would negatively affect learning.

Sub-algorithm for Step 5: We use the round-robin algorithm for selecting the next template for a learning objective. If the last template used by the tutor for a learning objective is T_{ij}, the next time it revisits the learning objective, it uses the template T_{ij+1}.

This rule-based algorithm is independent of the domain: it can be used for any domain wherein 1) appropriate learning objectives can be identified; 2) the student model is maintained in terms of learning objectives; and 3) problem templates are indexed by learning objectives. This rule-based adaptation algorithm has several advantages over vector spaces [20] and learning spaces [14] that have been popularly used to implement adaptation:

- The rule-based system is easier to build - there is no need to place all the problem templates in an exhaustive vector or learning space.
- The rule-based system is easily scalable - in order to add a new learning objective, we simply insert it in the domain model of which the student model is an overlay, and add additional problem templates to the template knowledge base, indexed by the new learning objective. This will not affect any existing learning objectives or their templates.

The learning path of individual learners is determined by the matching of the templates in the template knowledge base with the unsatisfied learning objectives in the student model. A rule-based system automatically supports all the learning paths - even those that may not have been explicitly modelled in a vector or learning space. Therefore, the resulting adaptation is more flexible. Our rule-based adaptation is similar to the adaptation mechanism used in ActiveMath [17] to determine the information, exercises, and examples presented to the learner, and the order in which they are presented.

5.1 An Example

Consider the tutor on arithmetic expressions. For this example, we will consider only the following learning objectives: correct evaluation and precedence of +, * and / operators. Let the following table represent the initial student model, where m / n denotes that the student has correctly solved m out of the n problems (s)he has attempted on the learning objective:

Student Model	+	*	/
Correct Evaluation	2/ 2	½	0/ 2
Precedence	0/ 2	2/ 2	1/ 2

Assuming $M_1 = 2$ and $M_2 = 60\%$, the student has not yet satisfied the following learning objectives: correct evaluation of * and /, and precedence of + and /. Assume that the next

template for the correct evaluation of * yields the expression 3 + 4 * 5, and the student correctly solves the entire expression. Since the expression includes the correct evaluation and precedence of + and * operators, the student gets credit for all four learning objectives:

Student Model	+	*	/
Correct Evaluation	3/ 3	2/ 3	0/ 2
Precedence	1/ 3	3/ 3	1/ 2

Since the student just satisfied the learning objective of the correct evaluation of *, the tutor considers the next unsatisfied learning objective, viz., correct evaluation of /. Assume that the next template for the correct evaluation of / yields the expression 5 + 10 / 4, and the student correctly solves the entire expression. Since the expression includes the correct evaluation and precedence of + and / operators, the student gets credit for all four learning objectives:

Student Model	+	*	/
Correct Evaluation	4/ 4	2/ 3	1/ 3
Precedence	2/ 4	3/ 3	2/ 3

If persistence p = 2, the tutor generates a second problem on the correct evaluation of /. Note that even if the student solves the second problem correctly, the learning objective of correct evaluation of / will remain unsatisfied (2/4 < 60%). All the same, since persistence p = 2, the tutor will pick the next learning objective for the subsequent problem.

Following are highlights of our adaptive algorithm:

- A student may satisfy a learning objective without attempting any problem on it. Note that the student satisfied the precedence of / operator while attempting problems on the other learning objectives. However, in order for this to occur, the tutor must be capable of automatically allocating (partial) credit. Our tutors on expression evaluation are capable of doing so.
- It is possible for a student who has already satisfied a learning objective to revert to the unsatisfied state. For instance, if the student had incorrectly solved the last two problems, correct evaluation of + would have reverted from satisfied (2/2) to unsatisfied state (2/4).

6. Evaluation of the Adaptive Tutor

In spring 2005, we conducted a web-based evaluation [5] of the rule-based adaptation in our tutor on arithmetic expressions. We used a between-subjects design: students were randomly assigned to either the control or the experimental group by the tutor. The control group used the non-adaptive version of the tutor and the experimental group used the adaptive version. Students used the tutor asynchronously, as part of a mandatory non-credit course assignment.

Protocol: We used the pre-test-practice-post-test protocol for evaluation of both the versions of the tutor:

- **Pre-test** – We used this stage to assess the prior knowledge of the students. The tutors used the pre-test to initialize the student model. The pre-test consisted of 21 problems covering over 20 different learning objectives for arithmetic expressions. Students were allowed 7 minutes for the pre-test. The tutor did not provide any feedback during the test.

- **Practice** – This stage was designed to help students learn from the tutor. The tutor provided detailed feedback for each problem.
    - **Non-Adaptive** tutor: This tutor presented 3 practice problems per learning objective, in the same order of learning objectives as on the pre-test. All the students were presented the same sequence of problems, regardless of how well they did on the pre-test. In other words, the tutor did not adapt to the learner's needs. The practice session lasted 15 minutes.
    - **Adaptive** tutor: This tutor adapted to the student's needs in two ways:
        - It presented problems on only those learning objectives that the student did not satisfy on the pre-test.
        - For each learning objective that the student did not satisfy, it presented 3 problems at a time or until the student satisfied the learning objective, whichever came first, before continuing with the next learning objective not yet satisfied by the student.

      The practice session lasted 15 minutes or until the student satisfied all the learning objectives, whichever came first. Therefore, students who satisfied all the learning objectives on the pre-test were presented no problems during practice. Those who did not satisfy any learning objective, and worse, solved all the problems incorrectly on the pre-test were presented problems in the same sequence as the non-adaptive version of the tutor.
- **Post-test** – We used this stage to assess the effect of practicing with the tutor, on the learning of the students. The post-test consisted of 21 problems, in the same order of learning objectives as on the pre-test. Students were allowed 7 minutes for the post-test. The tutor did not provide any feedback during the test.

The three stages: pre-test, practice and post-test were administered by the tutor back-to-back, with no break in between. The students did not have access to the tutor before the experiment.

Analysis: We calculated the percentage correctness of each answer, and calculated the average of these percentages for each student. Since this is per-problem average correctness, it eliminates practice effect that usually leads to students solving more problems on the post-test than on the pre-test. Table 1 lists the class average of these student averages on the pre-test and post-test for the non-adaptive and adaptive versions of the tutor. The improvement from the pre-test to the post-test was statistically significant (paired t-test 2-tailed p value < 0.05) for both the versions of the tutor. One-way ANOVA analysis showed that the difference from the pre-test to the post-test was statistically significant in both the groups. The only other statistically significant difference was between non-adaptive pre-test and adaptive post-test groups.

Table 1. Non-adaptive versus Adaptive Tutor

Average correctness of answers	Pre-Test	Post-Test	Change	Significance
Without adaptation (N = 15)				
Average	0.47	0.65	0.17	p = 0.014
Standard Deviation	0.24	0.20	0.24	
With adaptation (N = 25)				
Average	0.55	0.69	0.14	p = 0.0002
Standard Deviation	0.21	0.20	0.16	

However, the difference in the number of problems solved by the adaptive and non-adaptive groups was statistically significant (independent 2-tailed t-test p-value < 0.05). The minimum, maximum and average number of problems solved by the two groups during the practice session is listed in Table 2. Given that the improvement in learning was similar for both the groups, and that there was a statistically significant difference between the numbers of problems solved by the two groups during practice, our results are in accordance with the earlier result that adaptive problem sequencing helps students learn with fewer problems. For this evaluation, we did not consider the time spent by the students on practice since all the students on the control group were required to practice for 15 minutes.

Table 2. Problems Solved by the Control and Experimental Groups during 15-minute Practice

	Control Group (Non-Adaptive Tutor)	Experimental Group (Adaptive Tutor)	Statistical Significance
Minimum problems solved	28	1	
Maximum problems solved	86	60	
Average Problems solved	45.80	24.22	$p = 0.00017$
Standard Deviation	15.44	14.56	

7. Conclusions

We proposed a rule-based mechanism for adaptive generation of problems in web-based intelligent tutors. We described the domain and student models in our programming tutors, and presented an algorithm for rule-based adaptation of problem generation. We presented the protocol and results of a web-based within-subjects evaluation of the adaptation. The improvement in student learning from the pre-test to the post-test was statistically significant for both the versions of the tutor. However, there was a statistically significant difference in the number of problems solved by the two groups during practice – on the average, students using the non-adaptive tutor solved nearly twice as many problems during practice than those who used the adaptive tutor. Therefore, rule-based adaptive problem generation in web-based tutors helps students learn with fewer practice problems.

8. Acknowledgements

Partial support for this work was provided by the National Science Foundation's Course, Educational Innovation Program under grant CNS-0426021.

References

[1] Aimeur, E., Brassard, G., Dufort, H., and Gambs, S. CLARISSE: A Machine Learning Tool to Initialize Student Models. S. Cerri, G. Gouarderes, F. Paraguacu (eds.), Proc. of ITS 2002, Springer (2002). 718-728.

[2] Arroyo, I., Conejo, R., Guzman, E., & Woolf, B.P. An Adaptive Web-Based Component for Cognitive Ability Estimation., Proc. of AI-ED 2001, IOS Press (2001). 456-466.

[3] Baffes, P. and Mooney, R. J.: A Novel Application of Theory Refinement to Student Modeling. Proc. of AAAI 1996, Portland, OR, (1996) 403-408.

[4] Belmont, M.V., Guzman, E., Mandow, L., Millan, E., and Perez-de-la-Cruz, J.I. Automatic generation of problems in web-based tutors. In Virtual Environments for Teaching & Learning, L.C. Jain, R.J. Howlett, N.S. Ichalkaranje and G. Tonfoni (ed.), World Scientific, 2002.

[5] Birnbaum, M.H. (Ed.) Psychological Experiments on the Internet. San Diego, Academic Publishers, 2000. http://psych.fullerton.edu/mbirnbaum/web/IntroWeb.htm

[6] Bloom, B.S. and Krathwohl, D.R.: Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook I: Cognitive Domain, New York, Longmans, Green (1956).

[7] Bonar, J. and Cunningham, R.: BRIDGE: Tutoring the programming process, in Intelligent tutoring systems: Lessons learned. J. Psotka, L. Massey, S. Mutter (Eds.), Lawrence Erlbaum Associates, Hillsdale, NJ (1988).

[8] Brown, J.S. and Burton, R.R. Diagnostic models for procedural bugs in basic mathematical skills. Cognitive Science, Vol 2 (1978). 155-191.

[9] Czarkowski, M. and Kay, J. Challenges of Scrutable Adaptivity. U. Hoppe, F. Verdejo and J. Kay (eds.), Proc. of AI-ED 2003, IOS Press (2003). 404-406.

[10] Johnson, W.L. Intention-based diagnosis of novice programming errors. Morgan Kaufman, CA 1986.

[11] Kay, J.: Stereotypes, Student Models and Scrutability. Proc. of ITS 2000. G. Gauthier, C. Frasson and K. VanLehn (eds.). Springer (2000). 19-30.

[12] Kalyuga, S. Rapid Assessment of Learner's Knowledge in Adaptive Learning Environments, Proc. of AI-ED 2003, IOS Press, (2003). 167-174.

[13] Koffman, E.B. and Perry, J.M.: A Model for Generative CAI and Concept Selection. International Journal of Man Machine Studies. Vol. 8 (1976) 397-410.

[14] Kurhila, J., Lattu, M., and Pietila, A. Using Vector Space Model in Adaptive Hypermedia for Learning. Proc. of ITS 2002, Springer (2002). 129-138.

[15] Mann, P., Suiter, P., & McClung, R.: A Guide for Educating Mainstream Students. Allyn and Bacon, 1992.

[16] Martin, B. and Mitrovic, A. Tailoring Feedback by Correcting Student Answers. Proc. of ITS 2000. Springer (2000). 383-392.

[17] Melis, E., Andres, E., Budenbender, J., Frischauf, A., Goguadze, G., Libbrecht, P., Pollet, M. and Ullrich, C. ActiveMath: A Generic and Adaptive Web-Based Learning Environment. International Journal of Artificial Intelligence in Education, Vol 12 (2001). 385-407.

[18] Millan, E., Perez-de-la-Cruz, J.L., and Svazer, E. Adaptive Bayesian Networks for Multilevel Student Modeling. Proc. of ITS 2000. Springer (2000), 534-543.

[19] Reiser, B., Anderson, J. and Farrell, R.: Dynamic student modelling in an intelligent tutor for LISP programming, Proc. of IJCAI 1985. Los Altos CA (1985).

[20] Salton, G., Wong, A. and Yang, C.S. A Vector Space Model for Automatic Indexing. Communications of the ACM, Vol. 18(11), (1975). 613-620.

[21] Weber, G. and Brusilovsky, P. ELM-ART: An Adaptive Versatile System for Web-Based Instruction. International Journal of Artificial Intelligence in Education, Vol 12 (2001). 351-384.

Loosely Coupling Web-Applications

Paul Libbbrecht [a1] Enrique Machuca [b] Mark Spanbroek [c]
[a] *DFKI GmbH, Saarbrücken, Germany*
[b] *Universidad Malaga, Malaga, Spain*
[c] *Eindhoven University of Technology, The Netherlands*

Abstract.
When developers of two web-based environments wish to share activities, they need to negotiate the ways to do so. It may boil down to an a simple authorization, or URL exchange.
Generally, however, the exchange is more convoluted and measures have to be taken to guarantee the highest quality in the browsing experience. We address this problem in an attempt to generalize the action of following a hyperlink into a *browser delegation*. Report on the experience gained in implementing the delegation between ACTIVEMATH and SIETTE is provided.

Keywords. world-wide-web, browser, delegation, decentralized, integration,

Introduction

The World Wide Web has brought thousands of knowledge libraries and hundreds of educational activities within two clicks of a mouse. Inviting a reader from a web-page to another site is thus natural. This is generally materialized by a hyperlink, an ingredient which has made the world-wide-web so powerful.

An author who writes a link assumes the target of the link will stay what he expects. When the client browser follows the link, it will browse the link's target site, but will have no way to *get back* except by following browser history.

When the two sites are more elaborate web-applications, such links are typically generated instead of authored and are often only valid during a specific session of user activity. For example the fact that the user is logged in, or the relation to its user-model. Some web-applications provide relatively transparent links which could be used as in the hand-authored HTML case. But the problem of getting back is still not solved.

As a result a coupling is needed between the two web-applications. This coupling should allow the managed delegation of the web-browser between one server and the other and should allow an experience close to the one that would happen on a single server. To achieve such experience, currently, developers of the web-applications need to be involved, they need to exchange and agree on protocols and information to be exchanged during the delegation. The results are rarely (ex)portable and will only be maintained as far as the collaboration between the two sites' owners remains.

An integration of several systems is the objective of several general purpose architectures:

- Knowledge-Tree [2] uses the proxy-approach where a central server tunnels the content and requests which allows them to be adapted and tracked and which allows a central learner-model to be built from it.
- MEDEA [11] is a project whose aim is to integrate several existing learning-systems in a single one based on a shared ontology of the domain knowledge and of the user modelling information.
- Through the authoring of web-service composition scenarios, APeLs [4] has a potential to integrate heterogeneous services to provide a complete learning experience. It needs, however, a centralized service composition framework.

We describe an approach to generalize the task of *loosely coupling* two web-applications so that it should be possible for two authors to decide to loosely couple the web-applications which host their content.

A typical example is that of two collaborating researchers in the same domain. They will, typically, have complementary course-material and want their students to use each others course material.
Typically, these two researchers will, at most, be authors of content. Most probably they will not even have administrative rights on the e-learning system. Should it prevent them to share their course material?

This paper starts with an analysis of the requirements. We present the three systems that are planned to be loosely coupled within the LEACTIVEMATH EU project. The browser delegation scenario is, then, described in some detail. Finally, we describe the prototypes and current development, and present future work.

1. Loose coupling requirements

The servers that take part in the process of delegating a user-and-his-browser from one server to the other are as follow:

- the *guide-server* is the server where our browser is before this scenario comes into play. This guide server has some reasons to wish to send the user's browser to the following server. These reasons are probably the result of some previous exchange of knowledge such as the browsing and writing of authors or such as exchanges between agents using a shared ontology.
- the *activity-server* which is the server that will serve the activity to the browser that it is delegated from the guide-server. The term activity is used quite loosely to describe any sequence of web-interactions. In the setting of two communicating learning environments such as ours, it will be refined.

From the end-user perspective it should be perceived that the two servers are more or less the same servers. We can note the following requirements:

- single-sign-on: the user with his client-browser should not need to re-authenticate when passing from one server to the other. Based on the trust between the servers and on authentication at the guide-server, authentication should be deduced on the activity server.
  Note that it is unreasonable to expect that the first contact of a user with the web-application on the activity server will always directly start the activity. For

example, several web-applications will ask registration questions in order to seed their user-models.
- interface homogeneity: one could expect a similarity of textual and graphical language between the two applications without requiring full homogeneity.
- keep-going: a dead-end in the navigation course will be taking the user away from his current goal. Therefore the guide server should continue guiding after an activity is finished or indicate the achievement of a milestone.
- privacy: since we are talking about the process of exchanging information related to the user experience, personal-history, and/or credentials to enter domains, the user should be sure that his data is kept safely. This safety is generally guaranteed at each servers site but needs to be cared about in the exchange between the two servers. We expect encrypted and authenticated communication to be sufficient and manageable for this purpose.

It is important to note that a request for coupling is very far away from an integration at the development level. We hope to make possible the coupling of two web-applications on the impulse of two content-authors and under the auspices of system administrators.

2. Intended Usage

In this section we take the time to describe the web-applications that we intend to loosely couple and the scenarios where delegation is to happen.

2.1. The SIETTE *adaptive assessment tool*

SIETTE [3] is a web implementation of *Computerized Adaptive Tests*. The system is based upon a well-founded theory, *Item Response Theory* which explains how to diagnose learner's knowledge from answers to questions (items). The main advantage of using this approach is that we do not depend on number of items used to obtain the knowledge level of a student. With few items this theory ensures the correct knowledge level estimation.

SIETTE provides a web interface to pose questions to students, but also can be integrated within external systems. SIETTE stores the items in a *knowledge base*. These items can be authored by a teacher using the *test editor* but he can also analyze the behavior of students in tests using the *analyzer*, and correct difficulty on items thanks to the *item calibration tool*

Recent features [7] include *external items*, so called because these are exercises that are presented in a different system than SIETTE. The evaluation of the user's input in these items is then received as input from some external site. A first attempt of the delegation process actually happened as SIETTE delegating the browser to ACTIVEMATH for an exercise, using this approach.

Further work is being made to achieve delegation in both directions. Currently SIETTE is being adapted to be accessed using the delegation process, in order to let ACTIVEMATH delegate the evaluation of a topic to SIETTE. In the same way, ACTIVEMATH will let SIETTE discover the resources in the learning environment, so as content can be stored into SIETTE. Then, an assessment session from a topic on ACTIVEMATH can be delegated, but also an assessment on SIETTE can delegate the exercise's presentation to

ACTIVEMATH. In this way, SIETTE is used just as a sequencer of items to provide the adaptive behavior and the posterior statistical analysis of results.

2.2. The ACTIVEMATH *learning environment*

ACTIVEMATH is a web-based learning environment. It uses a semantic representation of the content, assorted with metadata annotations, to present learning material to learners and to offer interactive mathematical exercises. ACTIVEMATH maintains a learner-model describing the estimated *knowledge*, *comprehension*, and *application-capacity* of the learner for each concept. Using it, ACTIVEMATH selects books of content to be learned to achieve learning objectives and suggests further reading. For more details about the learning environment see [13], [10] and the references therein. Every user-interaction in ACTIVEMATH is on the web and ACTIVEMATH uses several web-services (see [9]).

The ACTIVEMATH learning environment is at the heart of the FP6 EU project named LEACTIVEMATH. Among others, this projects intends to provide an integrated platform including ACTIVEMATH, the SIETTE assessement tool, and an exercise repository. Moreover, it should combine exercises with tutorial dialogues, prototype a course-generator based on a reactive planner (see [12]) and a learner-model based on the competency model of the Pisa study [1].

2.3. The LEACTIVEMATH *exercise repository*

As described in the previous section, the need for interactive exercises is fundamental for an experience in the ACTIVEMATH learning environment. It was proposed to make in the LEACTIVEMATH EU project a collection of re-usable exercises that could be browsed, searched, experienced, and re-used by the public in the domain of calculus. The LEACTIVEMATH repository is being realized at the Eindhoven University of Technology.

2.4. Delegations in LEACTIVEMATH

Maintaining a learner-model with values that approximate the estimated mastery of the learner is a delicate task which relies, in ACTIVEMATH, on tracking the learner's reading actions and receiving exercise diagnoses. Both of these methods, however, can only be achieved after some time using the learning environment whereas one wishes to offer guidance as early as possible.

A first attempt in this direction was a form of *self-assessment* which invited the learner to estimate her mastery of each of the topics in the current course. Little positive experience was gained with this self-assessment approach and more efficient bootstrapping mechanisms of the learner model should be proposed.

One of them can be to invite the learner for an assessment session which should discover her knowledge of the domain as in SIETTE. This delegation will be integrated.

The decision to do so shall be taken by the tutorial component, which is the component responsible for the content selection and further-reading advise, or the open-learner-model, which is the component responsible to present to the learner the estimated mastery. During the interactions with the tutorial component or the open-learner-model, hyperlinks inviting the learner for an assessment session will be presented. The latter, when clicked, will take the browser for an assessment session in SIETTE which upon comple-

tion returns the learner's browser to the component it was before. After such a session, the learner-model will be updated as a result of the tests. The tutorial component will be able to provide better advice on further content to read or exercises to achieve. The open learner-model will present these links, among others, within dialogues with the student about the learner-model when doubts are emitted about the learner-model estimates. After the assessment, it will be able to present estimates with a greater evidence which should enhance the learners' trust in the capabilities of the system.

A requirement for the link inviting the session to be displayed is the discovery of the assessment sessions available. For such a session to be useful to the LEACTIVEMATH learner-model, the assessment measures should handle the same topic-names and do so using the same domain knowledge. A simple web-service call was agreed upon that should take, as parameter, the name of a domain knowledge node and return a list of resource-identifiers for each activities that are *for* this domain knowledge node. This allows the discovery of exercises in the repository and of assessment sessions in the assessment tool.

For the integration of SIETTE and ACTIVEMATH, more tuning is needed as more of the domain knowledge is needed for adaptive testing. From the set of concepts in ACTIVEMATH, and based on a table-of-contents typical of the domain, an export of the domain knowledge can be done. This exported domain knowledge can be enriched with the exercises attached to each exercise in ACTIVEMATH.

In the first version of this export, the only exercises supported where multiple-choice-questions and could be exported to SIETTE. Since then, however, richer interactivity exercise-types are supported by ACTIVEMATH. Therefore we use the delegation process again: SIETTE can delegate the browser back to ACTIVEMATH for exercises done there. The export is then limited to export *external exercises*.

3. The Browser Delegation Scenario

In this section we describe in detail the steps of the delegation scenario as a sequence of remote procedure calls. This sequence can be followed in picture 1.

The browser-delegation scenario starts with our user using the browser currently interacting with the *guide-server*. The latter, through authored content or discovery, offers a link that should lead the browser to the *activity-server* for the time of the activity and *bring it back* when finished.

The scenario described here is a sequence of web-service calls to the guide or activity servers interleaved with browser actions. It is not clear wether any web-service sequencing or choreography language can be used for this description.

check-availability a call should be made to check that the delegation is possible. It should include:

- a *resource-identifier*, a string describing the activity
- optionally, an *interaction type* that represents a *verb* describing what is expected the browser will have as interaction with the activity server. Examples include "run an exercise", or "see a piece of content"
- a user-identifier: a string to identify the user on both sides. This string may be the result of a translation to allow mapping between learners.

Faults may happen as the result of this request. Most of the HTTP error-codes [6] can apply here. The guide-server, receiving such a fault should update its knowledge about the availability of the activity for the given user and should let it proceed to a *further step*. No other result except a possible fault is expected from this call.

wish-to-start the guide-server notifies the activity-server, through a web-service call, that it intends to send a browser to interact with a given resource. It provides at least the following arguments:

- the *resource-identifier*, *interaction-type*, and *user-identifier*
- an amount of other optional information to allow the interaction to be best suited to the user. Provide a handle to the learner model may be a solution if the activity-server supports it. More realistically, we expect this to be a space for a small set of values computed from the learner model and encoded within a shared ontology or such information as the expectation about the duration of the activity.
- a URL-to-return-the-result-to which is a resource-locator to the web-service where to invoke the *activity-finished* web-service called later
- optionally, a URL-to-send-the-events-to can be provided so that events can flow between the servers while the interaction happens (see [9] about the usage of events).

In response to the notification received from the guide-server the activity-server should provide a URL-to-lead-the-browser-to, the guide-server now directs the browser to this URL. This URL should contain enough information so that the interaction can start right away. Among others, this means that this URL contains extra *tickets* that would automatically log-in the browser on the activity-server when it first requests this URL.

The request response may contain extra information such as URLs where events can be sent to or where the learner actions can be tracked. If the *check-availability* call has been done shortly before, no fault should be raised.

activity-cancelled In some cases, the call to *wish-to-start* will not be followed by the actual activity. The guide-server should then call this method with the parameters of the

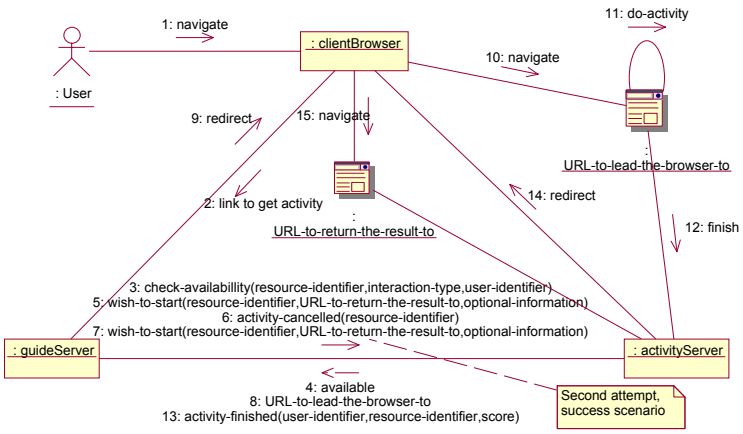


Figure 1. Loose-coupling steps.

resource and user-identifiers which should de-allocate any resource allocated for activities for this user- and resource-identifiers.

Now the learner interacts with the resources on the activity server.

activity-finished when the interaction is finished, the activity-server should direct the browser to future interactions that the guide-server should indicate.

It contacts the URL-to-return-the-result-to and invoke an activity-finished method with parameters including the user- and resource-identifiers as well as a numerical score, a floating number between 0 and 1. This should be promptly answered by the guide-server which will provide a URL-to-come-back which will be sent to the browser as the next place to go to. Faults should only occur in exceptional cases here as there would be no other option but to forward the error to the user, who can then only go back using his browser history.

4. Implemented Prototypes

The integration of SIETTE and ACTIVEMATH within the LEACTIVEMATH learning-environment is currently under work. The delegation process seems to be sufficient aside of the discovery activities. Among the steps for this, we have implemented the delegation of ACTIVEMATH exercises within an assessment session in SIETTE using XML-RPC web-service calls. It would be too verbose to present the method-names and parameters here, but it suffices to say that the method-names were the ones of the delegation scenario and parameters were them as well with space for extra information exchanged during the delegation (such as the URL to a learner-model web-service).

Based on this experience, the generalization of the delegation process has been under-work and will be used in the integrations planned in LEACTIVEMATH.

The prototype that we realized delivered activity-links which point to the guide-server and, only when requested, trigger the *wish-to-start* web-service call. This, in turn, resulted in a redirect of the browser to the activity-server. The same was true when invoking *activity-finished*. This setting made several HTTP requests (between various hosts thousand of kilometers apart) for a single page download and turned out to appear quite unresponsive. Calling the *wish-to-start* method earlier as link presentation seems to be a better approach.

Conclusion

We have presented a practical approach to couple two web-applications together with preliminary experiments. Reviewing the litterature seems to show little efforts in the direction of *loosely* coupling web-applications where, in principle, only content-authors and system-administrators are involved by writing content and providing authorizations. This seems, however, to be a fundamental ingredient to allow a quality user-experience while not requiring centralized systems such as Microsoft Passport[1] or the Central Authentication Service initiative.[2]

[1] MicroSoft Passport is a centralized single-sign-on solution, `http://www.passport.net/`.

[2] CAS is an open-source centralized authentication system based on cookies, see `http://www.yale.edu/tp/auth/`.

The implementation realized thus far is partial. Among the problems left to be solved for a completely portable approach to browser-delegation are the seals of mutual trust between the servers, the mapping between user-names, and the possible translation of resource identifiers. We have found in [8] several advice that might help in this direction.

This integration effort has tackled little the problem of discovering the resource of an activity. We do expect the semantic web technologies to be applicable in the future to solve this discovery problem. Similarly, exchange of learner information has not been considered here even though work has been done in this direction such as [5]. Our research, however, attempts the qualification of the delegation of activities with limited human contribution and in a decentralized way. It is probable that, once discovery and user-model exchange is achieved, a protocol similar to the browser-delegation will be used between two peers of such discovery, thereby avoiding the manual contribution of a link of the content author or the system administrator.

References

[1] *Measuring Student Knowledge and Skills – A New Framework for Assessment*. Organisation for Economic Co-operation and Development (OECD), 1999.

[2] P. Brusilovsky. Knowledgetree: A distributed architecture for adaptive e-learning. In *Proceedings of The Thirteenth International World Wide Web Conference, WWW 2004*, pages 104–113, New York, NY, May 02–05 2004. ACM Press. See `http://www2.sis.pitt.edu/~peterb/papers/p641-brusilovsky.pdf`.

[3] R. Conejo, E. Guzman, E. Millan, M. Trella, J. L. Perez de-la Cruz, and A. Rios. SIETTE: A Web-Based Tool for Adaptive Teaching. *International Journal of Artificial Intelligence in Education (IJAIED 2004)*, 14:29–61, 2004.

[4] Owen Conlan, David Lewis, Steffen Higel, Declan O'Sullivan, and Vincent Wade. Applying adaptive hypermedia techniques to semantic web service composition. In Paul de Bra, editor, *Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems, AH2003*, 2003. See also about `http://wwwis.win.tue.nl/ah2003/`.

[5] Peter Dolog. Identifying relevant fragments of learner profile on the semantic web. In *Proceeding of Workshop on Applications of Semantic Web Technologies for E-learning SW-EL'04*, Hiroshima, Japan, 2004.

[6] Roy Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. RFC 2616: HTTP/1.1, HyperText Transfer Protocol, 1999. See `http://www.w3.org/Protocols/`.

[7] E. Guzman and R. Conejo. A Brief Introduction to the New Architecture of SIETTE. In P. de Bra and W. Nejdl, editors, *Proceedings of the third International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 405–408. Springer, 2004.

[8] Norman Walsh Ian acbos. Architecture of the world wide web, volume one, Dec 2004. `http://www.w3.org/TR/webarch/`.

[9] P. Libbrecht and S. Winterstein. The service architecture in the activemath learning environment. In N. Capuano, P. Ritrovato, and F. Murtagh, editors, *First International Kaleidoscope Learning Grid SIG Workshop on Distributed e-Learning Environments*. British Computer Society, 2005. See `http://ewic.bcs.org`.

[10] E. Melis and E. Andres. Global feedback in ACTIVEMATH. *Journal of Conputers in Mathematics and Science Teaching*, 24:2, 2005. Accepted.

[11] M.Trella, R. Conejo, D.Bueno, and E. Guzman. An autonomous component architecture to develop WWW-ITS. In *Proceedings of workshop: Adaptive Systems for web based education*, pages 69–80, Malaga, Spain, 2002. See `http://www.lcc.uma.es/~eva/WASWBE/trella.pdf`.

[12] C. Ullrich. Tutorial planning: Adapting course generation to today's needs. In *Proceeding of 12th International Conference on Artificial Intelligence in Education*, 2005.

[13] C. Ullrich, P. Libbrecht, S. Winterstein, and M. Mühlenbrock. A flexible and efficient presentation-architecture for adaptive hypermedia: Description and technical evaluation. In Kinshuk, C. Looi, E. Sutinen, D. Sampson, I. Aedo, L. Uden, and E. Kähkönen, editors, *Proceedings of the 4th IEEE International Conference on Advanced Learning Technologies (ICALT 2004)*, pages 21–25, 2004.

Personalization Services for e-Learning in the Semantic Web

Nicola Henze [a,1]
[a] *ISI – Semantic Web Group,*
University of Hannover & Research Center L3S

Abstract. The Personal Reader framework implements a service-based architecture for developing and maintaining personalization functionalities on the Semantic Web, stemming from disciplines like e.g. adaptive hypermedia systems or collaborative filtering systems. A modular framework of components / services - for providing the user interface, for mediating between user requests and available personalization services, for user modeling, for providing personal recommendations and context information, et cetera, is the core of the Personal Reader framework. When a user is viewing some Web Content (the "Reader" part of the Personal Reader) s/he receives additional, personal information on the context of this particular Web content (the "Personal" part of the Personal Reader). Personal Readers have been developed for the area of e-Learning (Java, Semantic Web), and for browsing scientific publications.

Keywords. Personalization Services, Personalization Architectures, Semantic Web

1. Introduction

With the idea of a Semantic Web [2] in which machines can understand, process and reason about resources to provide better and more comfortable support for humans in interacting with the World Wide Web, the question of personalizing the interaction with web content is at hand. In the area of adaptive hypermedia, research has been carried out to understand how personalization and adaptation strategies can be successfully applied in hypertext systems and hypertext like environments. It has been stated that in the area of adaptive hypermedia and of adaptive web–based systems, the focus of developed systems has been so far on *closed world* settings. This means that these systems work on a fixed set of resources which are normally known to the system designers at design time (see the discussion on closed corpus adaptive hypermedia [4]). This observation also relates to the fact that the issue of authoring adaptive hypermedia systems is still one of the most important research questions in this area, see e. g. [3]. A generalization of adaptive hypermedia to an *Adaptive Web* depends therefore on a solution of the closed corpus problem in adaptive hypermedia. Within the Personal Reader project, we propose an architecture for applying *some* of the techniques developed in adaptive hypermedia to an open corpus. A modular framework of components / services - for providing the user

[1] Correspondence to: Nicola Henze, ISI - Semantic Web Group, University of Hannover & Research Center L3S, Appelstr.4, D-30167 Hannover Tel.: +49 511 762 19716; Fax: +49 511 762 19712; E-mail: henze@l3s.de

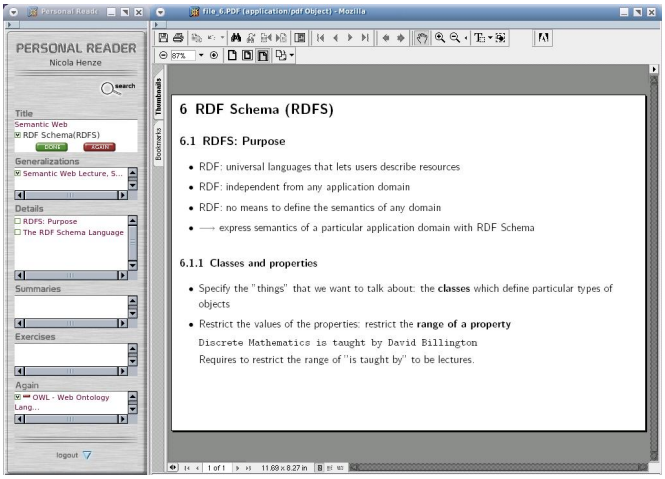


Figure 1. Screenshot of the Personal Reader for learning about the Semantic Web. The Personal Reader consists of a browser for learning resources *the reader part*, and a side-bar or remote, which displays the results of the personalization services, e.g. individual recommendations for learning resources, contextual information, pointers to further learning resources, quizzes, examples, etc. *the personal part*.

interface, for mediating between user requests and available personalization services, for user modeling, for providing personal recommendations and context information, et cetera, is the core of the Personal Reader framework [7]. The communications between all components / services is syntactically based on RDF descriptions. E.g. the request for getting personal recommendations for a learning resource for a certain user is provided by an RDF description which is exchanged between the components mediator and personal recommendations. Thus each component is a service, which is usually independent from the others and which can interact with them by "understanding" the RDF notifications they send. The common "understanding" is realized by referring to semantics in the ontologies used in the RDF descriptions which provide the valid vocabulary (see [6,7]). Prototypes of Personal Readers have been developed for the area of e-Learning (Java, Semantic Web), and for browsing scientific publications.

2. Proof-of-Concept: Personal Readers for e-Learning and for Browsing Scientific Publications

2.1. Personal Readers for e-Learning

The Personal Readers for e-Learning [5] (see Figure 1) provide a learner with a personal interface for regarding learning resources: the Personal Annotation Service recommends the learner next learning steps to take, points to examples, summary pages, more detailed information, etc., and always recommends the most appropriate of these information according to the learner's current knowledge, his/her learning style, learning goal, background, etc. The Personal search service extracts information from the actually regarded learning resource and checks for related information in other e-Learning corpora, and recommends retrieved results. If you want to set up your own Personal Reader instance

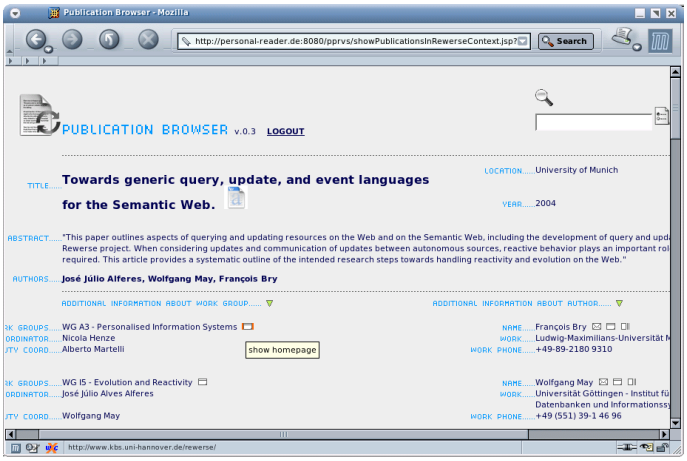


Figure 2. Screenshot of the Personal Publication Reader. When a user is viewing some publication, s/he receives additional, personal information on the context of this publication within the REWERSE project: background information about the persons and working groups carrying out this kind of or related research, additional information about the authors, etc.

for a course you are running, you need to provide RDF description on the learning resources of this course (examples of such RDF descriptions can be found following the link Resources on this project page, and a link to some domain ontology describing the application domain of your course, which you also use to annotate your resources.

Highlights:

- easy creation of personalized Readers for learning objects annotated according to LOM standard;
- demonstrates: re-usable personalization functionality for e-Learning courses;
- reasoning for the personalization services is realized using TRIPLE [9]

2.2. The Personal Publication Reader

The Personal Publication Reader [1] (see Figure 2) has been developed for the Network of Excellence REWERSEfor providing a personal interface to the publications developed in the project: All web-pages containing information about publications of the REWERSE network are periodically crawled and new information is automatically detected, extracted and indexed in the repository of semantic descriptions of the REWERSE network. This information, with extracted information on the project REWERSE, on people involved in the project, their research interests, etc., is used to provide more information on each publication: who has authored it, which research groups are related to this kind of research, which other publications are published by the research group or by this author, which other publications are on the similar research, etc.

Highlights:

- automatized annotation of Web data: automatic extraction of Web data, and automatized annotation of extracted data with meaningful semantic information (powered by the Lixto Suite, www.lixto.com) ;

- demonstrates: personalized content syndication;
- reasoning for the personalization service is realized Jena's RDQL language [8].

3. Conclusion

We have presented a framework for designing, implementing and maintaining adaptive *Reader* applications for the Semantic Web. The Personal Reader framework is based on the idea of establishing personalization functionality as services on the Semantic Web. The realization of personalization functionality is done on the logic layer of the Semantic Web tower, making use of description and rule language recently developed in the context of the Semantic Web. We have tested the framework with example readers in the area of e-Learning (Java programming, Semantic Web), and for browsing scientific publications of the REWERSE project. The current state of the project can be followed at `www.personal-reader.de`, where all the realized prototypes are available, too.

Acknowledgments

This work has partially been supported by the European Network of Excellence REWERSE - Reasoning on the Web with Rules and Semantics. We would like to thank Fabian Abel, Robert Baumgartner, Tobias Buchloh, Stefan Decker, Peter Dolog, Lilia Cheniti-Belcadhi, Christian Enzi, Marc Herrlich, Dennis Kohlmetz, Matthias Kriesell, Kashif Mushtaq, Wolfgang Nejdl, Michael Sintek, Sascha Tönnies, and Kai Tomaschewski for their support in getting the idea of Personal Readers into reality.

References

[1] Robert Baumgartner, Nicola Henze, and Marcus Herzog. The Personal Publication Reader: Illustrating Web Data Extraction, Personalization and Reasoning for the Semantic Web. In *European Semantic Web Conference ESWC 2005*, Heraklion, Greece, May 29 - June 1 2005.

[2] Tim Berners-Lee, Jim Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, May 2001.

[3] P. De Bra, A. Aerts, D. Smits, and N. Stash. AHA! version 2.0: More adaptation flexibility for authors. In *Proceedings of the AACE ELearn'2002 conference*, October 2002.

[4] Peter Brusilovsky. Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, 11:87–110, 2001.

[5] Nicola Henze. Personal Readers: Personalized Learning Object Readers for the Semantic Web. In *12th International Conference on Artificial Intelligence in Education, AIED'05*, Amsterdam, The Netherlands, July 2005.

[6] Nicola Henze, Peter Dolog, and Wolfgang Nejdl. Reasoning and ontologies for personalized e-learning. *Educational Technology & Society*, 7(4), 2004.

[7] Nicola Henze and Matthias Kriesell. Personalization Functionality for the Semantic Web: Architectural Outline and First Sample Implementation. In *1st International Workshop on Engineering the Adaptive Web (EAW 2004)*, Eindhoven, The Netherlands, 2004.

[8] RDQL - query language for RDF, Jena, 2005. http://jena.sourceforge.net/RDQL/.

[9] Michael Sintek and Stefan Decker. TRIPLE - an RDF Query, Inference, and Transformation Language. In *International Semantic Web Conference (ISWC)*, Sardinia, Italy, 2002.

A Web-based ITS for OO Design

Glenn Blank, Shahida Parvez, Fang Wei and Sally Moritz

Computer Science and Engineering Department
Lehigh University
Bethlehem, PA 18015 USA
001-610-758-4085, 001-610-758-4867
{gdb0, smp9, faw2, sgh2}@lehigh.edu

Abstract. Learning object-oriented design and programming is a challenging task for many beginning students. CIMEL ITS coordinates student learning in two client programs: web-based multimedia courseware (CIMEL) and the Eclipse IDE, each of which post student interactions to a server-based CIMEL ITS. The Expert Evaluator analyzes student work in Eclipse, comparing novice with expert solutions. The Student Model combines knowledge from the expert evaluator and the multimedia. Finally, the Pedagogical Agent, guided by updates from the student model as well as a learning styles inventory, interacts with the learner by selecting from several tutorial strategies. All three components share knowledge from a Curriculum Information Network, which represents a new "design-first" introduction to software development in Java.

1. Introduction

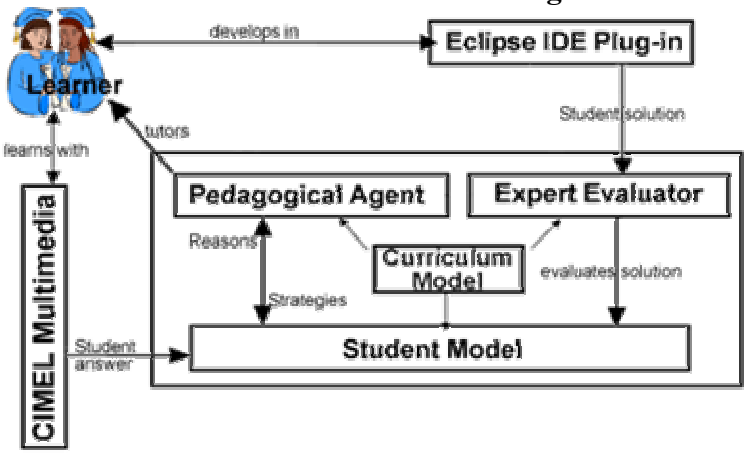
The application of ITS to programming has been limited to specific procedural aspects of programming languages such as LISP [2, 4], Pascal [20], C++ [11] and Java [17], not keeping up with "objects-first" pedagogy let alone recent trends in object-oriented design. Object-oriented design and programming are challenging for beginners, with large percentages of students struggling to understand basic concepts [12, 16]. We created web-based interactive multimedia courseware supporting an "objects-first" approach using BlueJ [10]. Experimental evaluation demonstrates that CIMEL multimedia improves conceptual understanding, but does not necessarily improve performance on a programming task [14]. These results motivate the development of an adaptive tutoring system, CIMEL ITS, observing student work with both the multimedia and the Eclipse programming environment.

CIMEL ITS is designed to interact with students through two client programs. First, multimedia courseware called CIMEL (Collaborative, constructive, Inquiry-based Multimedia E-Learning) introduces concepts in object-oriented design and Java [3]. CIMEL includes its own interactive activities, introduces small design/programming exercises, and gives quizzes based both on its own content and on programming exercises. CIMEL posts all student interactions to a database on a server, which the Student Model of CIMEL ITS observes. Experimental evaluation demonstrates that this multimedia improves conceptual understanding, but does not necessarily improve performance on a programming task [14]. These results motivate the development of an adaptive tutoring system, CIMEL ITS, which observes student work with both the multimedia and the Eclipse programming environment.

Second, students solve problems in the Eclipse integrated development environment (IDE), configured with plug-ins for UML design and interactive programming. Our UML plug-in also sends student work to an Expert Evaluator, which compares it with expert solutions. The CIMEL ITS, running on a server, will thus integrate observation of each student's progress from two different client-based perspectives. It will then offer assistance based on adaptive pedagogical strategies, within either the multimedia (such as reviewing prerequisite knowledge) or the IDE (such as improving flawed design).

The rest of this paper presents the design of CIMEL ITS as follows. Section 2 describes the flow of control from the multimedia through the ITS, then section 3 describes the flow of control from Eclipse through the ITS, using concrete examples. Section 4 summarizes innovations of our architecture and planned future work.

2. Flow of control from multimedia through the ITS



The figure on the left shows the architecture of CIMEL ITS. In its heart is the Curriculum Model, which represents knowledge about a novel curriculum. Before getting bogged down in the details of coding, we believe students should get a glimpse of the big picture of software development. We therefore propose a "design first" approach to learning software development, in which students learn object-oriented analysis and design as problem-solving skills [13, 14]. In our curriculum, students learn how to develop use cases to analyze problems, then design solutions using UML, before implementing much code in Java.

The Curriculum Model [19, 21] represents the knowledge that students must learn, design-first, in a Curriculum Information Network (CIN). We use the CIN to provide core domain knowledge for all three active components of the ITS, simplifying their design. The CIN links concepts together to show relationships between them. A concept may be identified as having one or more prerequisite concepts, and it may also be a component of another, higher-level concept. For example, in order to understand the concept of a method, one must know the concept of class, parameter, return value, etc. On the other hand, a method could be considered a component of an object. A difficulty measure is also assigned to each concept within the CIN. The three active parts of the ITS refer to the CIN to tie the student's learning activities to concepts in the curriculum.

The CIMEL ITS can interact with students either through CIMEL multimedia (on the left) or the Eclipse IDE (upper right), each of which initiate different flows of control through the ITS architecture. One possibility is that a student begins viewing the multimedia chapter "Objects and Classes". This chapter introduces basic object concepts, including classes, attributes, methods and instances. Interspersed throughout the content are interactive exercises and quizzes, which reinforce concepts and measure the student's understanding.

Suppose a student is learning how to identify the attributes for a class. The multimedia shows several examples (such as a "Tree" class which has attributes height and color). In a drag-and-drop exercise, the student is presented with things that represent a class, an object and an attribute. The student must classify these things by dragging and dropping them in containers labelled class, object and attribute. Later she completes a quiz with 4 multiple-choice questions relating to understanding attributes. For example: "Characteristics of an object (such as size and color of a dress) are called _____?" [learner selects from: attributes, methods, instances or classes]."

The CIMEL client records each screen the student visits, each interaction in exercises, and each quiz result to a server-side database, which the Student Model (SM) monitors. The SM represents student understanding of concepts as Bayesian networks corresponding to the concept structure in the CIN. The SM has three layers. First, the problem-domain layer is a set of graphs, each of which models the probability that a student understands a target concept and

its prerequisites, given their performance on a target exercise. Suppose an exercise asks a student about the concept on a UML attribute. There is a link from the node for the attribute concept to a node for the student's answer. There are also links from the prerequisite concepts, the concepts object and class to the attribute concept. The attribute concept also appears as root nodes in other networks for which it is a prerequisite. Once the SM determines the probability of the attribute concept, all other concept networks for which attribute is a prerequisite are updated.

Second, the historical knowledge layer updates a model of the student's knowledge state for student solutions to the same problem or multiple problems. If there are many wrong answers, this layer identifies possible reasons, such as that the student doesn't understand the target concept (attribute), or the prerequisite concepts (object and class), or the student may have forgotten these concepts over time (say, more than three days).

Finally, the cognitive layer [8] infers whether a student exhibits general problem solving patterns (such as decomposition or analogy) or antipatterns (such as blind hacking). For instance, if the student skipped the drag-and-drop question, the cognitive layer infers that the student may not be paying attention. The SM packages its diagnosis, including all possible reasons with probability values, as a packet, which it sends to the PA.

The Pedagogical Agent (PA) provides feedback and tutoring to the student. It consists of a feedback network and tutoring strategies. The feedback network is similar to the CIN, adding feedback constructs for each concept in the domain knowledge. Feedback is assigned a numerical level indicating if the feedback is basic or advanced. For example, the feedback consisting of concept definitions will be assigned level 1. The tutoring strategies, which may be represented by distinct agents, include a traditional tutoring strategy in which an agent plays the role of a tutor, and variations of cooperative learning strategies [5]. The "learning by disturbing" strategy employs a traditional tutor agent and a companion agent that attempts to test the student's knowledge by misleading him [1]. The "learning by teaching" strategy also has a traditional tutor agent and a companion agent that learns along with the student [15, 16]. In addition to the diagnosis from the SM, the PA considers the student's preferred learning style of the student. When the student uses CIMEL ITS for the very first time, she fills out a learning styles questionnaire [7], which categorizes individual learning styles based on the Felder-Silverman learning style model [6]. This model categorizes individual learning styles on a sliding scale of five dimensions: sensing-intuitive, visual-verbal, active-reflective, sequential-global, and inductive-deductive. The PA selects its tutorial strategy from the feedback network by combining the SM's diagnosis and the student's learning style.

For example, if a student has already reviewed a concept in CIMEL but the SM reports several mistakes and inattention, the PA will consult the student's preferred learning style to provide guidance about how the student could get more out of the multimedia. If the student's learning style indicates that she learns better with concrete facts, the PA will recommend that the student review a section of the HTML-based "Just The Facts" material, then follow the links to the quizzes. If the PA judges that the student cannot learn more from the multimedia and is still performing weakly, it can send a diagnostic report to a human instructor via e-mail.

3. Flow of Control from Eclipse through CIMEL ITS

The multimedia also directs the student to begin work on an assigned problem in Eclipse. The student then starts building a simple movie ticket vending machine in a student-oriented Eclipse plug-in supporting the design of classes in Unified Modeling Language (UML). As the student enters each piece of her solution (class name, each attribute, each method), the plug-in sends data to the Expert Evaluator (EE) for analysis.

The EE tries to match the student's entry with a corresponding part of an acceptable solution, which it generates from an instructor's textual description of a problem. Whenever the student completes an action, the EE will evaluate the student solution and generate a

data packet with the evaluation results. When the EE is able to match the component entered by the student, it will generate a packet containing the student's action, a packet count (how many packets were generated for this step) and the concept in the Curriculum Information Network to which the action relates. If the student correctly creates a class called TicketMachine and adds two attributes, movieTitle and ticketPrice, three packets will be sent to the SM.

Like quiz results for the multimedia, each Eclipse action is a leaf node in a problem-domain model. For the target concept class-name, class and object are prerequisite concept nodes, and the Eclipse action naming the class TicketMachine is the leaf node. Successfully performing this action updates the probability of the class-name concept. This probability will in turn be passed to the historical knowledge layer, which updates the student profile with the student's solution. The SM appends the probability that the student understands the target concept to the packet it received from the EE, then passes it to the PA.

The PA will play the role of an experienced tutor who encourages the student when he does well and offers help when it is needed. Determining the timing and frequency of positive feedback is tricky. The agent should try not to distract or annoy the student; on the other hand it does not want the student to feel abandoned. This is particularly important for female students who may lack confidence about their coding ability or by peers who have a head start [9]. The agent will use each student's preferred learning style and gender to determine the frequency of positive feedback. If the student prefers active learning, then the pedagogical agent will provide encouraging phrases more often than if the student prefers reflective behavior.

Suppose the student enters an attribute called "ticketsRequested." The EE will attempt to match the attribute on possible acceptable attributes for the TicketMachine class. When it doesn't find a match, it will search through other parts of its solutions: other class names, attributes in other classes, methods, and method parameters. A common student error is misidentifying a valid element of the solution: in this case, the student has identified a data item as an attribute that is more appropriate as a parameter to a method (the "printTickets" method). The EE infers valuable information about the student's reasoning.

Student errors are tied to CIN concepts. The CIN concept "attribute" is tied to the action of adding an attribute, but the definition of an attribute has more than one component: an attribute represents a characteristic, or data item describing or used by the class, and it is a value that must persist through the life of the object. These two parts of the definition are separate nodes in the CIN, and represent components of the CIN node for attribute. This particular error is tied to the "persistence" portion of the attribute definition.

The EE now generates an error packet which contains the student's action, the error (misidentified a parameter as an attribute), the CIN concept tied to the error, the correct action for the step (if any), and the actual text from the problem description that applies to this step. It then sends this error packet to the SM.

Since this example leads to multiple target concepts in the CIN, the SM will generate a corresponding structure of multiple target nodes in the problem-domain layer. The problem-domain layer calculates the probabilities for each target concept. After the historical knowledge layer updates itself, it may infer possible reasons for the error, such as the student doesn't understand an attribute is a property or an object-life value, or she does not understand the prerequisite concepts. Depending on the student's solution history, the cognitive layer may discern that the student is using preconceived notions to put the "ticketsRequested" as an attribute, because she did a similar action in another design exercise in a different context. The SM appends its diagnosis of possible reasons and probability values to the EE packet, sending it along to the PA.

When the PA receives an error packet, the first thing it will do is to check if the student had difficulty with the same concept before by looking at its feedback history.

Suppose there is no record that the student had any problem with this concept before. Next, the PA chooses the reason with the highest probability and uses it to get the associated concept from the CIN. Then, the PA uses the reason, concept and preferred learning style to retrieve tutoring content from the Feedback Information Network. In this case, the concept that the student is having a problem with is "Attribute". The feedback information for this concept is multileveled; first it is the problem specific feedback which is tied to the current problem. For example, it could say "Remember, you don't need to keep track of tickets requested." The next level feedback is domain specific level 1 feedback which could say "The attributes are values that need to be kept track of for the life of the object." Feedback for each concept will have many different components, such as verbal, visual, global, sequential, intuitive, and sensing components. Each of these components maps to a type of learning style. Each component will contain feedback that matches with that learning style. For example the visual component could have a picture of an object that specifies its attributes while the intuitive component could describe what an attribute actually means.

4. Conclusions and Future Work

The CIMEL ITS architecture offers several innovations. First, it integrates knowledge about what the student is learning from two sources: web-based multimedia and the Eclipse IDE. Both tools are designed to support a novel "design-first" approach to learning object-oriented software development. Second, the EE performs structured matches of student UML class designs with corresponding snippets of expert solutions, which it generates from instructor problem descriptions. Its step-by-step analysis reduces the complexity of the match and improves responsiveness. Third, the SM makes inferences about reasons for a student's correct or erroneous behaviours at three levels: the current student work, the student's history, and general problem-solving patterns and antipatterns. Fourth, the PA selects pedagogical strategies by combining the SM diagnoses and a learning styles inventory. Finally, it abstracts conceptual knowledge about the domain in a common Curriculum Information Network, facilitating the synergy of the three active components (each the topic of a Ph.D. dissertation). Separating much of the domain knowledge into a common repository simplifies the design of the three active components: the EE accounts for student behaviours in terms of CIN concepts; the SM generates estimates of student knowledge about these concepts; and the PA indexes a Feedback Information Network that parallels the structure of the CIN. While our work is still in the design and early implementation stage, we believe that our architecture makes useful improvements on the standard tri-partite model in terms of reusability.

We plan to demonstrate and discuss the first release of CIMEL ITS at the workshop. After further development and testing, we plan to evaluate CIMEL ITS in both university and high school settings.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grants No. EIA-0087977 and 0231768 and the Pennsylvania Infrastructure Technology Association. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF or PITA. The first author also gives thanks to God for inspiration and blessing.

References

[1] Aïmeur, E. and Frasson, C. Analyzing a New Learning Strategy According to Different Knowledge Levels. *Computers in Education*, vol. 27, no. 2, 1996, pp. 115-127.

[2] Anderson, J.R. and Skwarecki, E. (1986) The automated tutoring of Introductory Computer Programming. *Communications of the ACM*, vol. 29, September 1986, pp 842-849, ACM Press.

[3] Blank, G. D., Barnes, R. F. and Kay, E. J.. *The Universal Computer: Introducing Computer Science with Multimedia* (McGraw-Hill/Primis, 2003/2004). Sample material at www.cse.lehigh.edu/~glennb/um/ and cimel.cse.lehigh.edu.

[4] Brusilovsky, P., Schwarz, E., & Weber, G. (1996). ELM-ART: An Intelligent Tutoring System on World Wide Web. In Frasson, C., Gauthier, G., & Lesgold, A. (Ed.), *Intelligent Tutoring Systems* (Lecture Notes in Computer Science, Vol. 1086). Berlin: Springer Verlag. 261-269.

[5] Chan T. W. and Baskin A. B. Learning Companion Systems. In *Intelligent Tutoring Systems: At the crossroads of Artificial Intelligence and Education* (Edited by Frasson, C. and Gauthier, G.), Chap 1. Ablex, N.J., 1990.

[6] Felder, R. M. and Silverman, L. K. *Learning Styles and Teaching Styles in Engineering Education*, Engr. Education, 78, 7, 674-681 (1988).

[7] Felder, R. M. and Soloman, B. A. *Index of Learning Styles Questionnaire*, Available online at June (2003) in:http://www.ncsu.edu/felder-public/ILSdir/ilsweb.html

[8] Gürer, W. D. A Bi-level Physics Student Diagnostic Utilizing Cognitive Models for an Intelligent Tutoring System, PhD Dissertation, Lehigh University, 1993.

[9] Irani, L. Understanding Gender and Confidence in CS Course Culture, *Proceedings of the Thirty-Fifth SIGCSE Technical Symposium on Computer Science Education*, Norfolk, VA, March 2004, 195-199.

[10] Kölling, M., Quig, B., Patterson, A. and Rosenberg, J., The BlueJ System and its Pedagogy, *Journal of Computer Science Education,* vol. 13, no. 4, Dec 2003.

[11] Kumar, A. Model-Based Reasoning for Domain Modeling in a Web-Based Intelligent Tutoring System to Help Students Learn to Debug C++ Programs, *6th International ITS Conference*, Biarritz, France and San Sebastian, Spain, June 2002.

[12] McCracken M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Yifat Kolikant, Y., Laxer, C., Thomas, L., Utting, I., Wilusz, T., A Multi-National, Multi-Institutional Study of Assessment of Programming Skills of First-Year CS Students. In *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, December 01, 2001, Canterbury, UK

[13] Moritz, S. and Blank, G. A Design-First Curriculum for Teaching Java in a CS1 Course, to appear in *SIGCSE Bulletin (inroads)***,** June, 2005.

[14] Moritz, S., Wei, F., Parvez, S., and Blank, G. D. (2005), From Objects-First to Design-First with Multimedia and Intelligent Tutoring, *The Tenth Annual Conference on Innovation and Technology in Computer Science Education*, Monte da Caparica, Portugal, June, 2005.

[15] Palthepu, S., Greer, J., and McCalla, G. Learning by Teaching. *The Proceedings of the International Conference on the Learning Sciences*, AACE, 1991.

[16] Ratcliffe, M. B.. Improving the Teaching of Introductory Programming by Assisting the Strugglers. *The 33rd ACM Technical Symposium on Computer Science Education*, Cincinnati, USA, March, 2002.

[17] Sykes, E.R. and Franek, F.. A Prototype for an Intelligent Tutoring System for Students Learning to Program in Java. In *Advanced Technology for Learning,* vol. 1, no. 1, 2004.

[18] Wei, F., Moritz, S., Parvez, S., and Blank, G. D. (2005), A Student Model for Object-Oriented Design and Programming, *The Tenth Annual Consortium for Computing Sciences in Colleges Northeastern Conference*, Providence, RI, April 2005.

[19] Wescourt, K., Beard, M. & Gould, L. Knowledge-based adaptive curriculum sequencing for CAI: Application of a network representation, *Proceedings of the National ACM Conference*, Seattle, Washington, 1977.

[20] Woolf, B. and McDonald, D., Human-Computer Discourse in the Design of a PASCAL Tutor, *Proceedings of Conference on Human Factors in Computing Systems,* Boston, MA, 1983.

[21] Zhou, G. Tsong-Li, J. Wang, P., and Ng, A., Curriculum Knowledge Representation and Manipulation in Knowledge-Based Tutoring Systems. *IEEE Transactions in Knowledge Data Engineering* 8(5): 679-689, 1996.

The Assistment Builder: A Rapid Development Tool for ITS

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

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

1. Introduction

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

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

1.1 Overview of the Assistments Project

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

1.2 Purpose of the Assistment Builder

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

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

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

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

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

2 Methods

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

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

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

3 Results & Analysis

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

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

Table 1: Full Item Creation

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

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

4 Conclusions

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

References

1. Anderson, J. R. (1993). Rules of the mind. Hillsdale, NJ: Erlbaum.
2. Jackson, G.T., Person, N.K., and Graesser, A.C. (2004) Adaptive Tutorial Dialogue in AutoTutor. *Proceedings of the workshop on Dialog-based Intelligent Tutoring Systems at the 7th International conference on Intelligent Tutoring Systems. Universidade Federal de Alagoas, Brazil, 9-13*.
3. Jarvis, M., Nuzzo-Jones, G. & Heffernan. N. T. (2004) Applying Machine Learning Techniques to Rule Generation in Intelligent Tutoring Systems. Proceedings of 7th Annual Intelligent Tutoring Systems Conference, Maceio, Brazil. Pages 541-553
4. Koedinger, K. R., Aleven, V., Heffernan. T., McLaren, B. & Hockenberry, M. (2004) Opening the Door to Non-Programmers: Authoring Intelligent Tutor Behavior by Demonstration. *Proceedings of 7th Annual Intelligent Tutoring Systems Conference, Maceio, Brazil*. Page 162-173
5. Koedinger, K. R., Anderson, J. R., Hadley, W. H., & Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, *8*, 30-43.
6. Murray, T. (1999). Authoring intelligent tutoring systems: An analysis of the state of the art. International Journal of Artificial Intelligence in Education, 10, pp. 98-129.
7. Razzaq, L., Feng, M., Nuzzo-Jones, G., Heffernan, N.T., Aniszczyk, C., Choksey, S., Livak, T., Mercado, E., Turner, T.E., Upalekar. R, Walonoski, J.A., Macasek. M.A., Rasmussen, K.P. (2005) The Assistment Project: Blending Assessment and Assisting. *12th Annual Conference on Artificial Intelligence in Education 2005, Amsterdam*
8. Rose, C. P. Gaydos, , A., Hall, B. S., Roque, A., K. VanLehn, (2003), *Overcoming the Knowledge Engineering Bottleneck for Understanding Student Language Input , Proceedings of AI in Education.*