Generating Individualized Hypermedia Applications

Lori Kettel, Judi Thomson, Jim Greer

ARIES Laboratory
Department of Computer Science
University of Saskatchewan
Saskatoon, SK Canada
lak131@cs.usask.ca

1 Abstract

This paper presents a process for creating adaptive, individualized, instructional hypermedia applications. The process (APHID-2) builds on a prototypical system (APHID) that semi-automatically generates customized, instructional hypermedia applications. APHID ensures that good design principles are followed, both for the hypermedia application and for the interface that presents the hypermedia application. It uses a domain model, constraints and patterns to support partial automation for creating hypermedia applications.

APHID-2 enhances the models of APHID by adding a student model that overlays the domain model with a Baysian Belief Network. APHID-2 codifies the preferences and learning goals of the individual learner in a set of rules that translate preferences and goals into parameters that guide the generation of the hypermedia application. This paper explores the design of the APHID-2 system.

1.1 Keywords

generating hypermedia, student models, Bayes nets, concept maps

2 Introduction

Instructional hypermedia collections are often constructed by teachers and instructional designers to support specific learning needs. These document collections normally consist of some combination of public domain content found on the web and some custom built pages. Building quality hypermedia content is an expensive and time-consuming process. Furthermore, quality content alone does not guarantee pedagogically effective resources. The way content is selected, structured and presented is also important.

Imagine that we have a collection of hypermedia materials that concentrate on the subject of plate tectonics for junior high school science students. The collection might be used for multiple

purposes, including presenting an initial overview of the domain, presenting detailed instruction on some set of topics, providing a review of some previously learned material, or acting as the source for a comprehensive instructional unit to be learned in its entirety. If that same collection were also being used for remedial activities in a college-level geography course, a very different set of instructional requirements might arise. Each use of the hypermedia collection places different requirements on the way the collection is organized or on the content that is highlighted. In fact, each different learner places different requirements on the hypermedia collection. Unfortunately, all to often instructional designers are forced to use a "one-size-fits-all" strategy when building hypermedia collections, and hope that a wide range of learners can make use of the collection in a wide range of learning situations.

Even if we suppose that the learner finds a subset of the hypermedia collection that is relevant and accurate for a particular learning goal, the learner's own preferences and abilities may render several parts of the subset inappropriate. For instance, the content of the pages may vary from being too easy to being too difficult. Or, the format of the pages may not interest the user. Perhaps the pages have many images and the user is not a visual learner. Differences in learning styles and cognitive ability play a large part in selecting appropriate learning materials and appropriate modes of presentation.

Ideally, the instructional designer / hypermedia creator would know the characteristics of the intended users and could design hypermedia applications for some stereotypical user. However, for a hypermedia creator to construct an individualized set of documents for each learner would be time consuming and likely not that successful. The chances of knowing exactly who the learners will be, especially within the context of web-based instruction, are slim. It is especially important that each student's current knowledge and learning goals and style are taken into consideration or the student may not achieve the intended objectives.

Applying a user model and adaptivity to the creation of the hypermedia applications may solve this problem. The goal of this project is to create a set of software tools that will allow a hypermedia designer to automatically create an individualized hypermedia application based on a set of hypermedia elements, a model of the domain, and a model of the learner.

This paper presents a system that can generate adaptive instructional hypermedia applications. The system is built upon a generative hypermedia application environment named APHID (Thomson, 2000). APHID dynamically produces instructional hypermedia applications from primitive data elements and a concept map. The applications generated by APHID are based on a number of fixed user stereotypes. We have extended APHID in this work by making it capable of generating individualized hypermedia applications, taking into account a model of the learner. Throughout this paper we will be using an example of creating hypermedia sites on the topic of plate tectonics to illustrate the process.

3 Individualizing Instructional Web Sites

In order to individualize the web for instructional purposes, one must create individualized instructional plans for each student. Those plans rely on an understanding of both the learner

and the content of the domain. There are generally two types of approaches to instructional planning. The first is delivery planning, which decides the sequence of the explanations, tests, presentation problems, exploration, and decides how to manage the initiative in a tutorial dialogue. The other approach is content planning, which involves generating, ordering and selecting which content goals should be used based on the current state of the student, and monitoring the execution of the content plan in order to determine when to re-plan, or generate a new plan as the state changes (Vassileva & Wasson, 1996). Knowing the student's current knowledge level, goals and preferences is vital to creating an individualized plan that will suit the needs of a particular student.

3.1 Student Modelling

In order to create individualized hypermedia applications for a student, information about the student must be gathered into a student model. In the most general sense, a student model is the system's beliefs about the learner's knowledge, interests and goals (Holt, et al., 1994). An adaptive instructional system will use this student model to modify the interaction to suit individual students. A comprehensive student model would contain information about all of the student's knowledge prior to using the tutoring system, the learner's progress, preferences, interests, goals and any other information related to the learner. For obvious reasons, one cannot model all of the potential knowledge of a student and much of it would not even be useful for the given domain. Therefore, many systems attempt to model the student's knowledge as an overlay on the given domain knowledge structure (Holt, et al., 1994).

One might construct such an overlay from a concept map of the domain. The concept map will contain a list of the concepts plus the relationships between them. This concept map can then be used to construct a Bayesian Belief Net (BBN) for updating and making inferences about the student's knowledge. One method for using BBNs to model a student's knowledge involves describing the belief network as a graph with the nodes representing the concepts and arcs representing prerequisite links between the concepts (Reye, 1999). Given two topics A and B where A is a prerequisite for B then the following will describe the relationship:

$$p(\text{student-knows}(B) \mid \neg\text{student-knows}(A)) = 0$$

This reads as, *the probability that the student knows B given that the student does not know A is 0*. The next thing that can be modelled in the prerequisite relationship is how the student's knowledge of A will change the belief in the student's knowledge of B. This is specified by assigning a value to the probability:

$$p(\text{student-knows}(B) \mid \text{student-knows}(A))$$

The closer the relationship between A and B, the higher the probability will be. By using a BBN a model can be created not only of how much the student knows of each concept but also of the student's understanding of how the concepts relate to one another.
However, the student model must consist of more than just the student's current knowledge level. It must also contain information on how the student learns, what his/her goals are, his/her overall level, and preferences on interactions with the material (Bull, 1997).

There are different ways in which this student model could be created. One could imagine a system that begins by asking the student some general questions and perhaps giving a quiz on the topic to determine knowledge level. Then as the student browses through the collection of documents the modelling system could watch the student and update the knowledge level. Using the student browsing behaviour, it could determine whether the student is learning or if the student is lost and randomly clicking on links. The browsing pattern may also give clues on whether the student is trying to learn the material in-depth or just looking at an overview and which concepts seem most interesting or important to the student.

The student model can be collected and maintained using many different approaches. For this research we assume that a model of the student exists, but we do not address the issues of diagnosis or maintenance of the student model. Instead, the work focuses on how to use the student model to guide the selection of instructional material.

3.2 Domain Modelling

Typically the design process for a hypermedia application involves first selecting a topic, then creating a model of the domain to facilitate understanding of the structure of the information, and finally creating a model of the interface and user elements that will be a part of the application. The resulting application provides navigation options for the user based on the structure of the data. The assumption made is that the bulk of the information within the application is well structured with identifiable attributes.

Domain modelling is a valuable method of gaining an understanding of the structure of the information within the instructional domain. From the data model one can determine the relationships among the individual data elements within the instructional domain. These relationships can be used to provide some of the navigation opportunities within the application such as allowing users to navigate from descriptions of classes to the home page of the instructor for the class (using the relationship classes have instructors). Indeed, for casual users who wish to gain a general understanding of the domain, data model navigation may be the only type of navigation required.

Applying Patterns to Hypermedia Instructional Design (APHID) is an approach to instructional design, which allows the author of a hypermedia application to semi-automatically generate different web sites from a collection of data. The approach relies on the creation of several types of models, including a detailed domain model. Each generated site has a different instructional plan, which is represented by a sequence of hyperlinks that traverses the entire site in a particular order. The structure of the domain model, a stereotyped model of users and the instructional goals govern the differences between generated sites (Thomson, 1999).

The APHID process begins with the author creating a domain model in the form of a concept map and then data elements for each concept. The concept map records associations between concepts (presently 4 types of associations are used: aggregation, generalization, prerequisite and uses). A data element consists of the data (text, video, pictures) that the learner will view along with additional meta-data that are used by APHID to create the hypermedia application. Data

elements are also classified by the type of instruction represented by the element (e.g. quiz, explanation, simulation, directions). Web sites are created by selecting data and concepts using parameters and guidelines that differ according to the type of site and type of user.

APHID allows the generation of highly customized web sites, but it does not use a detailed model of individual students to create the instructional plan and consequently does not take into account the individual student's knowledge level, learning style and learning goals. By adding a student model one could create an individualized view of the hypermedia application that will hopefully allow the student to better learn the information or at least find it easier to navigate the site.

4 Individualized Instructional Planning with APHID-2

Individualized hypermedia applications can be created by extending APHID with a student model and a system of rules to govern how the student model affects the choices made about presenting content in the application. We are working on the development of a set of rules that can change the settings of the parameters that exist in the APHID software. APHID-2 is a software system that uses the APHID hypermedia generation process and adds the ability for the system to react to a detailed user model. To accomplish this, a user model is maintained for each individual learner and that model is user to trigger rules and tweak parameters that APHID uses to create hypermedia applications.

4.1 The APHID-2 Student Model

The APHID-2 student model consists of an overlay of the concept map. In addition to the level of knowledge the student has about the various domain concepts, the student model also includes a measure of the student's general knowledge level (novice, intermediate or expert). Other information about student preferences such as the student's learning goal, the student's preferred learning style, preferences for textual versus visual presentations, degree of social interaction, eagerness to learn, etc. are also important. The student's learning goal may be to learn a specific concept in-depth, be given an overview of a topic, or follow a review of a concept or group of concepts. The student may want a highly structured tutorial or an open-ended opportunity to browse and explore. The learning style information consists of two aspects: 1. whether the student prefers explanatory descriptions and examples versus interactive activities and 2. whether the student prefers diagrams, pictures or images versus more text.

The student model for APHID-2 is represented as an XML document. The DTD for the user model (figure 1) describes the components of the student model. It lists the parts of the student model and the possible values for the various attributes. The knowledge overlay of the concept map is described by the concept elements. Each concept element has three attributes: an id that uniquely identifies it, the student's knowledge level, and a prior probability that the Bayes Net will use. The concept also has 3 elements: a name, an inlist and an outlist. The inlist is a list of all the concepts that are pointing to this concept and the outlist is a list of all the concepts that this concept points to in the Bayes Net. The inlist has a full-joint probability distribution to characterise the degree of influence that predecessor nodes have on the current node, Bayesian inference is used to update the student model. The results of the updated student model are used to make decisions for individualizing the hypermedia application.

4.2 Extending APHID

```
<!ELEMENT studentmodel (name, concept+)>
<!ELEMENT name (#PCDATA)>
<!ATTLIST studentmodel
   goal (in-depth | overview | review),
   level (novice | intermediate | expert),
   styel (explanatory | interactive),
   visual (1 | 2 | 3),
   linkType (inline | side)
   linkCoding (colour | text)>
<!ELEMENT concept (name, inlinks*, outlinks*)>
<!ATTLIST concept
  id ID #REQUIRED
  knowledgeLevel  (novice | intermediate | expert)
  prior CDATA #REQUIRED >
<!ELEMENT inlinks (concept, fullJointProbabilityDistribution)>
<!ELEMENT fullJointProbabilityDistribution (#PCDATA)>
<!ELEMENT outlinks (concept)>
```

Figure 1: Document Type for Student Model

The APHID system contains the model of the domain as it is understood by the instructional designer and has the capability to dynamically create hypermedia applications from that model given a set of parameters to guide the construction. The parameter values are preset within the APHID software and are derived from APHID's knowledge about stereotypical users and the type of application desired. It is a relatively simple matter to change the preset values of individual parameters to something that more closely matches a detailed model of the user.

Figure 2 illustrates the process of creating documents using APHID. The domain model is created and the parameters for the application (type and user) are selected. These parameters cause the rules to be given values that are used as constraints on the application generation process. The document generator then creates the hypermedia application.

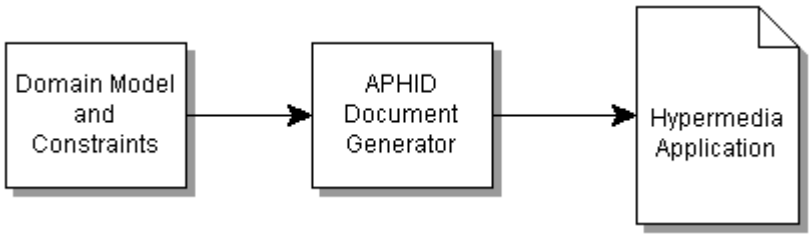

Figure 2: Architecture for APHID system

The parameters within APHID affect the appearance of the individual pages, the site as a whole, and which concepts and data elements are chosen. Parameters that affect the individual pages are the number of data elements per page, the maximum number of images per page, the maximum or minimum length of text on each page, and whether the links should be inline or to the side. Parameters that affect the site as a whole are the maximum and minimum depth of the hypermedia and the maximum number of pages. The choices for parameter settings made may be based on the student's knowledge, learning style, or learning goals.

The parameters also determine the tutorial order of the concepts by determining the traversal algorithm that will be used for each concept, or whether the concept will be included or not. Tutorial order is the term used to describe the order in which the pages would be viewed if they were presented as a strict tutorial with only next and back links.

From this linear tutorial order the application is then expanded to a full hypermedia application by adding links to related pages. These links are selected based on constraints in the concept map and local and global constraints derived from the student model. For example, in figure 3 one can see the links (on the side) that represent “next” and “back” plus additional “related” links. “Effects of Plate Techtonics” links to a more detailed page and the other two link to more general pages. Colour coding is used to help guide the user. Alternatively, a textual representation could have been used instead of colour. How the links are coded and how many are allowed on each page are constraints that can be determined by the student model.

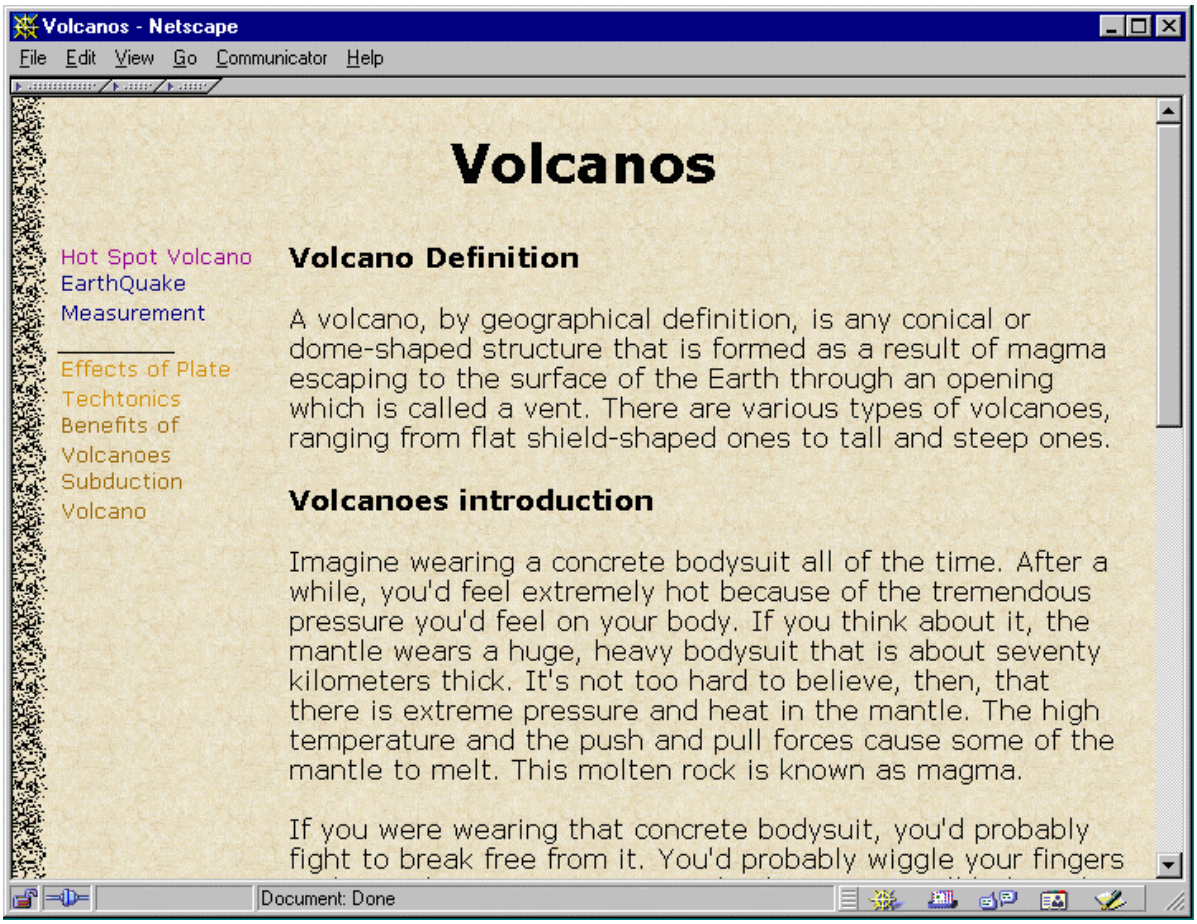

Figure 3: Sample Aphid-Generated Page

APHID uses XSLT (XSL Transformations (XSLT) 1.0, 1999) to control the presentation of the final hypermedia sites. XSLT is a collection of template rules. Each rule consists of a pattern and a template. The pattern determines which XML elements from the input document will be processed using the template. An XSLT document usually consists of at least one template for every type of element in the XML document. Every input element is processed with a single template.

Figure 4 shows the parameters used within APHID. APHID-2 will set those parameters based on the student model rather than relying on the stereotype models currently used by APHID.
XSLT is flexible enough to allow APHID-2 to create sites that are adaptive as well as individualized. For instance, by adding rules, additional XSLT templates can be activated to change how a page is displayed. The change would be based on the student model and would allow the learner to easily be shown which material is yet to be covered. Possible presentation level changes could include a change in background colour, highlighting of key words or the addition of extra hyperlinks to the page. Style sheets are used for modifying the presentation. Figure 3 uses one style sheet but by changing the style sheet used, the presentation can be altered. The choice of style sheet is another attribute that will be determined by the student model.

```
class Crules : public Cobject
{
        private:
                UserType m_eUser;
                OrgPatternType m_ePattern;
/***    site limitations***/
                int m_iMaxDepth;
                int m_iMinDepth;
                int m_iMaxPages;
/***    page limitations        ***/
                int m_iMaxLinks;
                int m_iMinLinks;
                int m_iMaxPics;
                int m_iMaxText;
                int m_iMinText;
/***    rules about concepts   ***/
                int m_iRootMaxExp;
                int m_iSeminalMaxExp;
                int m_iSecondaryMaxExp;
                int m_iTargetMaxExp;
                TraversalType m_eRootTraversal;
                TraversalType m_eSeminalTraversal;
                TraversalType m_eSecondaryTraversal;
                TraversalType m_eTargetTraversal;
                AlgorithmType m_eRootAlgorithm;
                AlgorithmType m_eSeminalAlgorithm;
                AlgorithmType m_eSecondaryAlgorithm;
                AlgorithmType m_eTargetAlgorithm;
                bool m_bRootMarked;
                bool m_bSeminalMarked;
                bool m_bSecondaryMarked;
                bool m_bTargetMarked;
/***    rules for data elements         ***/
                int m_iMaxElements;   //should be normalized
}
```

Figure 4: Possible Parameters within APHID

4.3 APHID-2 System Architecture

In order to determine how best to present the material to the student, a comparison must be made between the student model and the stereotype model. Where the student model differs from the stereotypical case, parameters must be changed. In order to do this, a set of rules tell APHID-2 when and how to change a parameter. The Java Expert System Shell (Jess) (Friedman-Hill, 2000) is a CLIPS-like rule engine and scripting environment written in Java, which will be used for implementing the rule base.

Deciding what the APHID parameters should be is a two-step process. The first step involves selecting a stereotype from APHID based on the student's knowledge level and learning goal. This gives defaults for all the parameters and set the traversal pattern to be used for concepts. The next step is to adapt the stereotype and thereby tweak the settings based on the rules and the student model. Figure 5 illustrates the architecture of APHID-2, focusing on the additions made to APHID. A student model is added and certain attributes and constraints are potentially changed from those normally set by APHID's stereotypes before generating the hypermedia application.

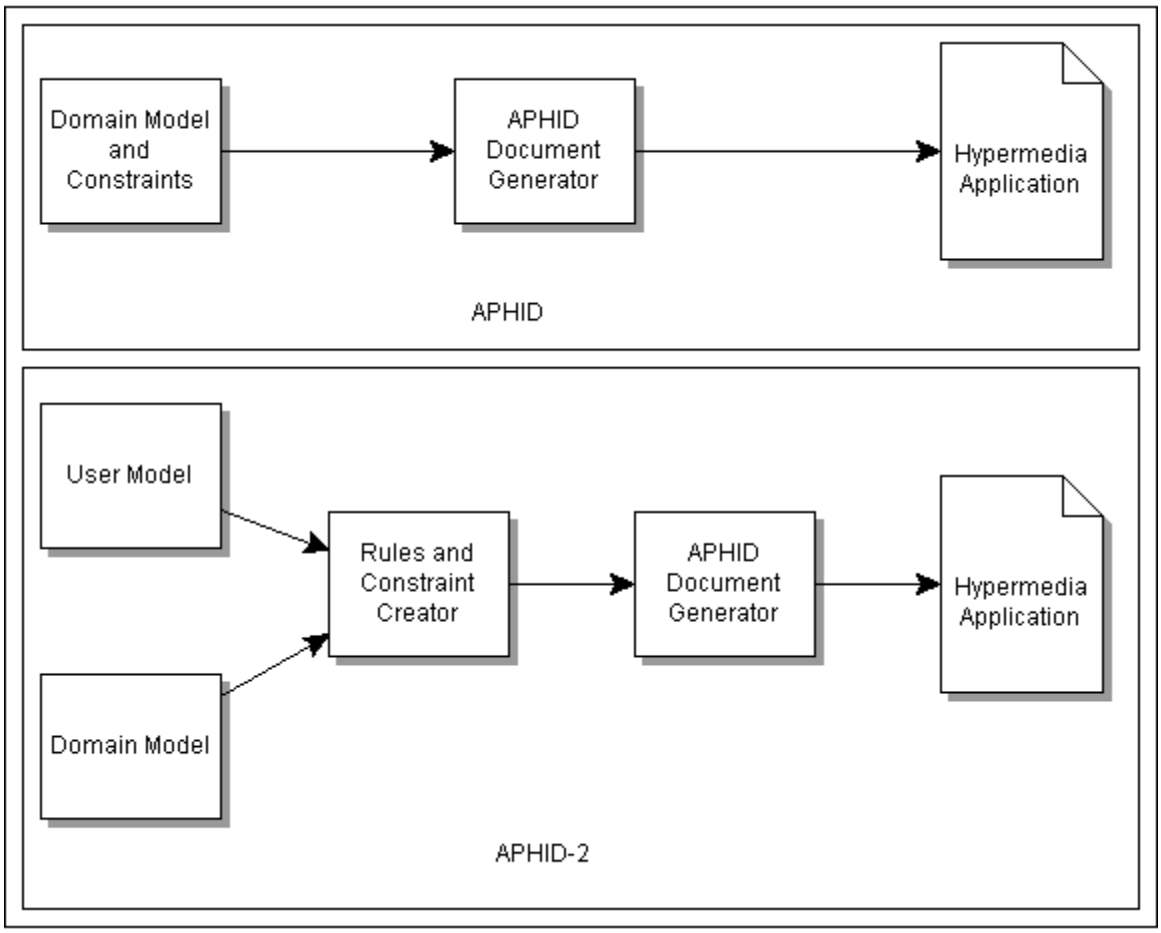

Figure 5: System Architecture

5 Encoding knowledge in Jess

Once the student model and domain models have been created, APHID-2 converts the user model into a set of Jess facts. These facts look like the following:
(student (name "Bob Smith") (goal "inDepth") (level "novice") (style "examples") (visual 2) (linkType "inline")(linkCoding "colour"))

The facts also include information pertaining to the student's knowledge level of the concepts. A direct mapping from the XML version of the student model to the Jess facts is envisioned.
After the facts have been encoded, APHID-2 must decide which rules are to be used. These rules are based on the domain model. The domain model describes the best way to display the information and what constraints to set given the various characteristics of its intended audience. These constraint settings can be mapped in a fairly straightforward manner to Jess rules.

For example the constraints describe which traversal algorithms to use to explore the domain model given a certain learning goal and knowledge level. A set of rules (written in Jess) could be created to express each of the conditions described by the constraints. For example, one rule set would select the stereotype model from APHID:

- if the goal is to learn in-depth and the knowledge level for a concept of type seminal is expert then use a spiral traversal algorithm.
- if the goal is to learn in-depth and the knowledge level for a concept of type seminal is novice then use a depth first algorithm.

A second example illustrates how the composition of pages can be adapted using the student model. The preferred learning style portion of the student model tells whether the student learns best by visualizing the information or reading text. The parameters for the hypermedia application include a description of the minimum and maximum images per page should be used depending on the preference of the user. This information would map into a rule of the following form:

- if the student's visual level is 1 use a minimum of 0 and maximum of 4 images per page
- if the student's visual level is 2 use a minimum of 2 and maximum of 6 images per page

The rules must be engineered for each domain giving some degree of domain dependence to the rules. The rules don't refer to the contents of the domain model but rather to its structure.

Finally, APHID-2 will run the rules associated with the domain model and the application parameters using the facts created from the student model. The resulting inference should come up with the proper settings for the constraints. Once these constraints have been set, the new constraint and parameter values are set in APHID and the hypermedia application can be generated.

APHID assumes that there is no student model and that the student has stereotypical behaviour. Therefore APHID generates the entire site at one time. Obviously, in APHID-2, the student model changes as the student progresses through the hypermedia application, therefore, after a while the application may no longer suit the student. There are two possible ways to handle this problem. The first would be to generate the pages on demand. As the student browses, a specialized browser could watch and update the student model as it perceives new evidence and the next page in the application would be generated dynamically. The difficulty with this approach comes in retaining context while generating the next page. The second way to handle the problem of student model changes during browsing would be to generate the entire site and then allow the student at any point to update the model and re-generate the remainder of the site. Retaining context and consistency are still a concern in this instance, but at least it does not have to be done for every page viewed.

6 Conclusion

This project will combine an existing hypermedia generation system (APHID) with student models to create highly individualized hypermedia applications. The idea of individualized, adaptive hypermedia applications for a learner is appealing to most educators. The drawback to individualization is the time required to make multiple versions of a hypermedia application. Hypermedia customization becomes even more appealing when the applications can be individualized semi-automatically (or even automatically).

A study was performed using the existing APHID system in which a subject matter expert compared web sites generated using APHID and web sites created by instructional technologists. All of the web sites contained subsets of the same content and had the same layout. The comparison was done blind; the expert knew nothing about how the different sites had been created until after the evaluation was complete. The results of this study demonstrated that the customization possible using APHID is one of its most appealing features.

The evaluator felt customization of web sites for learners would be beneficial, but indicated that he would never have considered actually creating individualized sites for particular students because of the time involved in creating multiple sites. His initial evaluation of the web sites showed that, when using the default settings for application parameters provided with APHID, the resulting web sites were acceptable, but not spectacular. The evaluator had specific complaints about how the sites had been constructed but felt that they would be useable.

After the initial evaluation, the evaluator was asked to suggest alternate settings for the application parameters based on his personal preference for teaching and learning. He then re-evaluated those web sites and found the sites to be well constructed and intuitively laid out. He was extremely positive in all regards about the sites that were customized with his preferences.

The difficulty in producing such hypermedia with APHID lies in determining exactly what the appropriate settings for the application parameters should be. Most users cannot articulate precisely what their preferences are, and yet they often have strong preferences. Even when users do understand their own preferences, the mapping from those preferences to the application parameters used by APHID is non-trivial. The system proposed in this paper will allow the creation of adaptive, customized hypermedia applications without necessitating detailed knowledge on the part of the user about how those applications are created or about how APHID uses information to change application structure.

7 References

Bull, S. (1997). A Multiple Student and User Modelling System for Peer Interaction. In R.

Schafer & M.Buaer (Eds.), *ABIS-97: 5 GI-Workshop, Adaptivitat und Bunutzermodellierung in interaktiven Softwaresystemen* (pp. 61-71). Universitat des Saarlandes, Saarbrucken.

Friedman-Hill, E. (2000, 31/01). Jess, the Java Expert System Shell.
Available: http://herzberg.ca.sandia.gov/jess/ (Accessed 01/03/2000).

Holt, P., Dubbs, S., Jones, M., & Greer, J. (1994). The State of Student Modelling. In J. Greer and G. McCalla (Ed.), *Studnet Models: The Key to Individualized Education Systems* (pp. 1-35). New York: Springer Verlag.

Reye, J. (1999). Student Modelling based on Belief Networks. International Journal of Artificial Intelligence in Education. Volume 11. To appear.

Thomson, J. (2000). *Applying Patterns to Hypermedia Instructional Design (APHID).*
PhD Thesis , Department of Computer Science, University of Saskatchewan, Canada.

Vassileva, J., & Wasson, B. (1996). *Instructional Planning Approaches: From Tutoring towards Free Learning.* Presented at the Proceedings of EuroAIED'96. Lisbon, Portugal. XSL Transformations (XSLT) 1.0. (1999, November).
Available: http://www.w3.org/TR/xslt (Accessed 06/02/2000).