Web-Based Learning Environment using Adapted Sequences of Programming Exercises

Radovan Kostelník*
radok@nextra.sk

Mária Bieliková*
bielik@elf.stuba.sk

Abstract: Adaptive hypermedia (AH) educational systems aim at improving self-learning of individuals. Although the research in this area is very active, only little work is done in the area of educational knowledge representation and their use in various domains. In this paper we present the ALEA (Adaptive LEArning) system, which is a web-based environment targeted on support of learning programming languages using adapted sequences of programming exercises. We describe the design of the ALEA, its architecture and the method of adaptation. The main focus is on design of adaptive behavior, which includes strategy, concept and fragment selection and presentation adaptation. We compare our approach to the existing adaptive hypermedia systems or models.

Key Words: adaptation, adaptive hypermedia, web-based system, learning style, sequences of programming exercises, automatic navigation, rules, XML.

1 Introduction

A support for personalization and adaptation based on various aspects of a user behavior and context of the environment use became an important feature in modern learning environments. Current educational systems often contain large bases of information fragments interconnected by various relations. Without a support of navigation through the information jungle the user may found him/herself lost. Obvious solution is a definition of lessons or predefined paths through the application domain [9]. This is just a partial solution to the problem because the lessons may not fit every user's learning style. Techniques of adaptive hypermedia give us the possibility to change dynamically presented educational material according to the user needs [5].

There are several ways how to perform adaptation. The most visible and therefore most popular is the adaptation of presentation. The presentation of the same content is modified by the adaptation engine, thus a user gets different output according to his/her current needs (represented by a user model). The most widely used technique is content highlighting by means of color and/or pictures. This technique is typically used to help a user to navigate in the information space.

The second approach is the adaptation of the content. The content that is not likely to be seen by a user is hidden or grayed out. Some advanced adaptive hypermedia (AH) systems include also natural language processing facilities that modify the actual text of the presented pages.

The third approach is the adaptation of navigation. Following a link in a standard web application results in displaying the requested page. Adaptation featured systems may include

* Department of Computer Science and Engineering, Slovak University of Technology, Ilkovičova 3, 812 19 Bratislava, Slovakia

additional processing into link resolution. For example, if a user clicks a link to display content for a concept, the system may offer him/her with a series of pages that the user should read before the requested one (in case he/she had not already read them).

Knowledge about the adaptation is crucial in all mentioned approaches. In this paper we describe an approach to design an adaptive educational web-based system targeted on learning programming using solved exercises (programming samples). Our approach is based on representing knowledge about adaptation on several layers (strategy selection layer, concept selection layer, fragment selection layer) represented using the XML documents.

2 Learning using exercises

Most of adaptive educational systems are not constrained by the content delivered to a learner. The content is usually divided into groups based on the media type: text, images, or eventually video or audio. Only text parts are further marked by a user or context of presentation characteristics aimed to the adaptation. However, in existing systems all text content is often considered as an explanatory text.

We consider several types of text fragments in the domain of learning programming using exercises (we are concentrated on beginners, so our exercises are simple assignments; size of resulting programs is in most cases several tens of lines of code). The explanatory text is just one of them. For example, we distinguish an *exercise definition*, *hint* and *solution*. It is obvious that the type of a text fragment plays an important role in such system.

Adaptation is based on considering the cognitive style of learning related to the domain of programming. The application domain of programming gives us the opportunity to lead the learner over different paths through the information base. Some students prefer first to see an explanatory text related to a concept they are learning, then a generalization of the learned programming concept (given by a programming schema [1]) and finally to practice programming by solving given exercises. Hints together with a definition of the programming tasks further improve the process of learning.

Another group of students prefers go straight to solve the programming tasks immediately after seeing the explanatory text (if even), then look at the schema related to the exercises of concepts and compare solutions with presented generalization.

Above mentioned learning styles represent the common approach to learn programming (we do not consider here the cognitive style of learning in terms of didactic resources preferences or presentation forms preferences [11]). The first is known as “from general to concrete”, the second one “from concrete to general”. We call them *strategies*. A strategy itself is realized by series of rules that are used to select the content for adapted for the learner according preferred strategy. The strategy typically determines the order of concepts that are presented to the learner.

Strategies are defined during the course authoring. Two or more typical patterns that fit the user’s behavior are defined. The task is to automatically deduce which strategy is the optimal for the current user. This process is called strategy selection or automatic strategy switch. Each strategy contains a set of rules that specify the right strategy. Decisions are based on the user movement through the application domain and his/her “click-patterns”.

3 The ALEA Design

In this section we present design of the ALEA (Adaptive LEArning) system, which is a web-based environment targeted on supporting teaching of programming languages using adapted

sequences of programming exercises. The ALEA system consists of the following parts (see Figure 1): application domain model, information fragment base, user model, adaptation knowledge rule base and the adaptation engine.

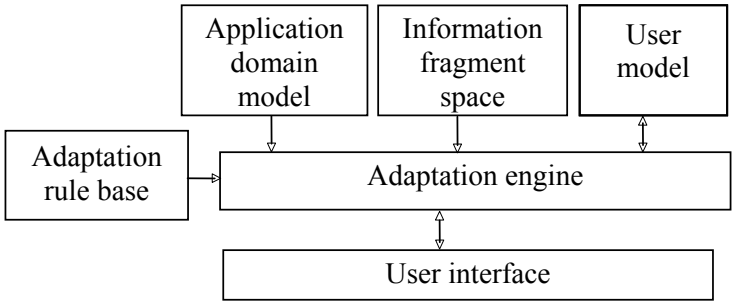

Figure 1. Components of the ALEA system.

3.1 Application domain model and information fragment base

Application domain model is usually represented by a graph, in which each node represents a concept and edges represent relations among them. Concepts and relations typically have additional attributes (e.g., type). Concepts are atomic or composite, the first representing individual pages, the latter abstract groups of lower level concepts. Concepts usually do not contain the material that is presented to a user. The material (e.g., text, pictures) is stored in the information fragment base. Mapping between a concept and its presentation is driven by various ways: some AH systems use concepts that are one-to-one mapped to physical pages [6], others allow one to many relation between a concept and information fragments [9].

The ALEA application domain model is based on the concept space idea. Individual concepts are organized in an interconnected network. Nodes of the network represent concepts down to the level of single exercises. Edges form relations among concepts.

We have developed an application domain for programming languages Lisp and Prolog. Our information base at the moment contains more than 80 solved simple problems (programming exercises). Each exercise consists of at least three parts: a definition, one or more hints and a solution. We created an ontology that describes the application domain with the following relations:

subconcept: relation between two non-specialized concepts (typically explanatory texts),

instance: relation between a concept and an example,

schema: relation between a concept and a schema,

similarity: relation between two or more similar concepts,

prerequisite: relation determining the order in which the concepts are to be learned.

Application domain represented by a graph of concepts is related to the information fragment base, which stores actual content presented to a learner. The information fragment base consists of a set of text chunks stored in files. The relations among concepts and fragments are stored in a database. As we mentioned above, each fragment has a type (e.g., exercise definition, hint, solution or text).

3.2 User model

The user model contains data about each individual user, his knowledge, preferred learning style, etc. The two most widely used approaches to the user modeling are the overlay model and the stereotype model. The overlay model provides a more detailed information about the estimated user knowledge because it stores individual value for each concept, page or information fragment. The result is the copy of the domain model filled with the user's data.

In stereotype model each user is assigned a stereotype, which designates that he/she is a member of a group that shares some characteristics. Typically the stereotypes are based on

the knowledge level. There may exist stereotypes in other dimensions. It is useful to consider a combination of both approaches depending on the amount of data about the user (start with a stereotype model and later switch to overlay model).

The ALEA user model is the *overlay* type. To save space and increase extensibility the user model is realized as a database, which contains only data related to the concepts and fragments already visited. The list of visited fragments includes number of visits to the page displaying the fragment and the date of last access. The concept-visit list includes also the estimated level of knowledge the user has reached about each concept. The user model also contains recorded user behavior and the user preferences. The recording of a user behavior produces a sequence of predefined actions (e.g. Login, Logout, ContextClick) with additional data, which is used for the adaptation.

4 Adaptation

Delivery of the content to a user is done by the content-selection and display mechanism. Adaptive engine lies in the heart of the system and glues all parts together. Its main tasks are to select the content to be presented to a user and to prepare it to suit the user's needs.

The knowledge about the process of content selection is divided into three layers:

- strategy selection layer,
- concept selection layer and
- fragment selection layer.

Each layer is represented separately using rules written in the XML language.

4.1 Strategy selection layer

The strategy is represented by a set of rules that determine the order in which the sequences of individual concepts would be presented. The system infers which strategy is the most suitable for the user in particular context. The strategy selection is driven by rules attached to each strategy. The result of rules evaluation is a numeric score. The strategy with the highest numeric score is selected as the most suitable.

It is not effective to revise and alter the strategy after each user's action. This would likely cause a confusion of the user. The amount of time (of perhaps number of user actions) after which the strategy is to be reconsidered should be set carefully.

Figure 2 depicts a typical strategy selection rule set. The result of the rule set evaluation is the variable *StrategyScore* that contains the numeric evaluation of the suitability of the strategy. The strategy with the highest score is selected.

```
Ruleset default()
If   um:AverageUserKnowledge >= um:intermediate
Then var:StrategyScore += 10

If   um:AverageUserKnowledge < um:intermediate
Then var:StrategyScore -= 10

If   um:AverageSolutionTime <= 2
Then var:StrategyScore += 8

If   um:AverageSolutionTime > 2
Then var:StrategyScore -= 8
```

Figure 2. An example of strategy selection rule set.

The strategy selection is one of the most difficult decisions in the adaptation. The inputs on which the system may set the decision are almost exclusively computed values based on the observation of user's

behavior. The selection of positive and negative values that determine the score of the strategy is also a matter of experience and experimentation.

4.2 Concept selection layer

The second layer of content selection is the *concept selection* layer. Selection of concepts is driven by the actual strategy, which is represented by a set of rules. The rules determine which links would be displayed, order and type of the links and the next concept the user would see after he/she presses the *Next* button or navigates a context link.

Concept selection process is started each time a user requests a concept, which is each time he/she clicks on a link. The system loads the rules from the actual selected strategy. The result of rule set interpretation is an information structure containing a list of links related to the requested concept.

However, when a user requests guidance by clicking the *Next* button, the process of link selection described above is preceded by the process of selection of the "next" concept to display. The system loads the appropriate set of rules and the result of interpretation is a sequence of one or more concepts that the system suggests the user to visit.

4.3 Fragment selection layer

Main task of the *fragment selection* layer is to select actual fragments from the information fragments base and produce the resulting layout of the concept presentation. The fragment selection is also based on interpretation of rules, which are part of a strategy.

The main criterion for selecting fragments, which are going to be displayed, is a type of the selected concept. For example, the type of the concept *exercise* is usually related to the fragments of three types: definition, hints and a solution. It is obvious that the solution may not be displayed as the first one.

4.4 Adaptation

The most visible effect of adaptation visible to the user is the presentation adaptation. By the presentation adaptation we mean altering the appearance of the content, mainly by means of various colors and fonts. Presentation adaptation is also directed by rules. The rules work with prepared presentation description: an internal document containing information about the page that is to be displayed. This information includes the links to other concepts and their difficulty level, types of the links, etc. The data from the user model are also available. The presentation description is altered according to these data by interpreting the rules. This results mainly in changing the link background color based on whether the concept was previously visited or not. Figure 3 depicts a screenshot where links to schemas or exercises (left part of the screen) are displayed in various colors.

4.5 Additional features

Many educational systems support only one way communication: from the teacher to the learners. We decided to augment the capabilities of the ALEA with some additions (ALEA has been used in the autumn 2002/2003 course Functional and Logic Programming at the Slovak University of Technology [10]).

The first one is the possibility to append custom comments to each displayed concept. These comments are visible to other users of the system. This introduces a mechanism for users to cooperate on the learning process. Supervisors‘ (typically teachers) comments are displayed using different color that the comments written by students. The supervisor has the ability to

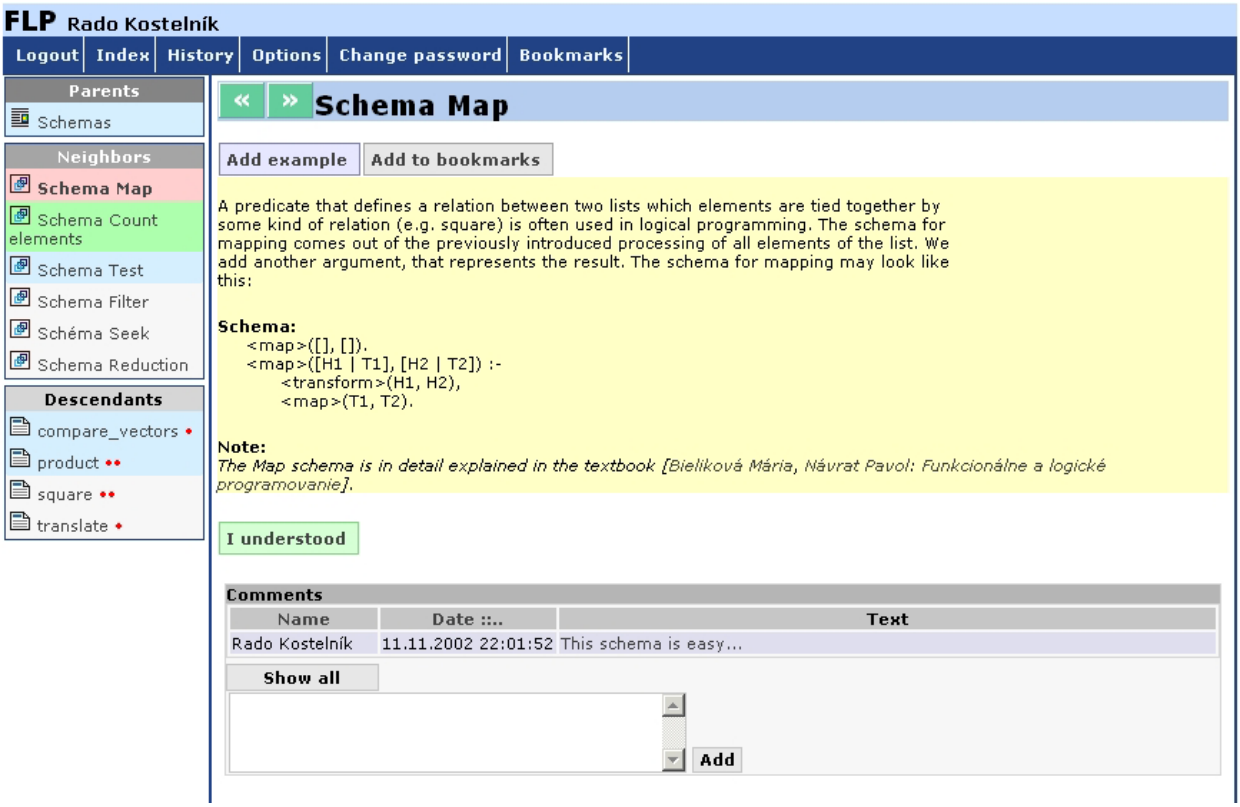

Figure 3. A screen shot of the ALEA system.

remove comments. Comments may serve also as a means for evaluating student's comprehension or their activity.

Another feature is the possibility of uploading material – and therefore changing the application domain by adding a new concept and its information content. Currently the system supports only adding new exercises. A student can add an exercise concept related to any other non-exercise concept (e.g., programming schema). The only information he/she must provide is a description of the exercise and its definition. Neither hints nor solution are required. Such uploaded concept is displayed using different icon to signalize that a supervisor did not yet approve it. The supervisor has the option to approve or remove the example and edit the contents of the newly uploaded concept. After successful approval by the supervisor, the exercise is no longer displayed using a different icon and it acts like other concepts defined in the course preparation phase.

Moreover, to increase user's comfort, each user can modify the color scheme the system uses to distinct different links. The basic are the background colors of visited/not visited concepts, color of links to concepts that do not have prerequisites met yet.

5 Related works

In this section we provide a comparison of our system with AHAM, XAHAM models and WHRULE system, which inspired our research (more related systems exist but these can be considered in some sense as referential).

AHAM is a model developed by De Bra and his colleagues [2]. The AHA! system was built using the AHAM model [3]. The model itself has the aim to define a basis on which new adaptive hypermedia system can be based. The ALEA system satisfies the conditions to be

AHAM compliant. But there are some distinctions. In the ALEA, every concept can contain references to fragments or so called atomic components, which enables modeling of different knowledge levels.

In the ALEA, there is no concept of phase in the evaluation of the rules. Our system rather provides the teacher or the authoring person with events that correspond to the phases of the evaluation. Also the propagation switch is not used because every update is immediately visible either in the user model or in the actual presentation description.

User profiles in XAHM [8, 7] stand for a similar concept to the strategy selection in the ALEA. A strategy defines how the system would adapt the presentation. The strategy assigned to a user associates the user with a stereotype. The way to determine the suitable user profile is the main distinction between the XAHM and the ALEA. The ALEA uses rules to infer the most feasible strategy. Current implementation supports only one dimension of the adaptivity: the user's behavior including the path of concepts the user went through.

The WHURLE system [4] uses a concept of chunks to store data about information fragments. Each chunk description contains the media type of the chunk, the actual text content with links to non-text resources and some other information. Chunks are similar to fragments in the ALEA. There is one important difference: chunks in the WHURLE are self-contained, which means that the metadata and data of the chunk are stored together whereas the ALEA uses different means to store these two.

The chunks are organized into lessons using *lesson plans*. A lesson plan is an XML document containing a hierarchy of *level*s. Levels produce similar abstraction like concepts in the ALEA's application domain model. The difference is that whereas in the WHURLE a lesson plan defines a top-down hierarchy of „concepts" an author proposed path through individual chunks, in the ALEA domain model order of concepts and grouping by a parent-child relations is not explicitly specified.

Adaptation in the WHURLE is realized through conditional inclusion of chunks and levels. Each lesson plan may include dependencies. Dependencies may involve such aspects as the user's prior experience (e.g. previous mandatory levels) and time (a material may be available only during a limited period of time). Adaptation driven by dependencies is also included in the ALEA where the prerequisite relations among concepts represent dependencies. The time limitations may be incorporated in the rules defining the strategy. However, the WHURLE system does not mention link adaptation (at least at the time of publication of [4]) and presentation adaptation in terms of altering the user interface.

6 Conclusions

Today's educational applications with static content, ordered to lessons are not sufficient. They need a background that makes them looking "alive", adapting to a user needs and preferences. We described in this paper the ALEA system, which is targeted on constructing a system enabling the author the opportunity to adapt the adaptation process and a user the comfort of adaptive learning environment.

The system has been implemented and experimentally examined in the autumn 2002/2003 course Functional and Logic Programming. The information base consists of a set of solved programming exercises from Lisp and Prolog programming languages together with related programming schemas. This environment formed the base for defining the abstractions used in design and development of the system.

The presentation of the content to the user is driven by the content processing component of the system. It is implemented as a web server running an application that delivers actual

content. The other components of the system described earlier stand in background behind the web application.

The main idea of adaptation in the ALEA is definition of various layers at which the system behavior can be adapted: the strategy, concept and fragment selection levels together with presentation adaptation. Future development on ALEA would be focused on enhancing the adaptation capabilities, experimenting with adaptation techniques and designing a user friendly authoring environment. We plan also consider different constraints of students (e.g., time available to learning) to improve concept selection according real user needs. This research will be based on collected data about users' behavior during the experiment.

This work has been supported by the Grant Agency of Slovak Republic grant No. VG1/ 0162/03 "Collaborative accessing, analysis and presentation of documents in internet environment using modern software tools".

References

1. Bieliková, M. and Návrat, P.: Use of program schemata in Lisp programming: an evaluation of its impact on learning. *Informatica*, Vol. 9, No. 1, 5-20. 1998.
2. De Bra, P., G.J. Houben, and Wu, H.: AHAM: A Dexter-based Reference Model for Adaptive Hypermedia. In Proc. of the ACM Conference on Hypertext and Hypermedia, Darmstadt, Germany, 1999, pp. 147-156.
3. De Bra, P.: AHA! Meets AHAM. In Proc. of 2nd Conf. on Adaptive Hypermedia and Adaptive Web-based Systems, P. De Bra, P. Brusilovsky, and R. Conejo (Eds.), Malaga, Spain, May 2002. Springer LNCS 2347, pp. 388-391.
4. Brailsford, T.J. et al.: Autonavigation, Links and Narrative in an Adaptive–Web Based Integrated Learning Environment, In Proc. of WWW 2002, May 7-11, 2002, Honolulu, Hawaii, USA. http://www2002.org/CDROM/alternate/738/index.html
5. Brusilovsky, P.: Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, Kluwer academic publishers, 11 (1-2), 87-110, 2001.
6. Calvi, L. and Cristea, A.: Towards Generic Adaptive Systems: Analysis of a Case Study. In Proc. of 2nd Conf. on Adaptive Hypermedia and Adaptive Web-based Systems, P. De Bra, P. Brusilovsky, and R. Conejo (Eds.), Malaga, Spain, May 2002. Springer LNCS 2347, pp. 79-89.
7. Cannataro, M., Cuzzocrea, A., Mastroianni, C., Ortale, R., and Pugliese, A.: Modeling Adaptive Hypermedia with an Object-Oriented Approach and XML. In Proc. of 2nd Int. Workshop on Web Dynamics, 2002. Available at ttp://www.dcs.bbk.ac.uk/webDyn2/proceedings/.
8. Cannataro, M. and Pugliese, A.: XAHM: an XML-based Adaptive Hypermedia Model and its Implementation. In Proc. of 3rd Workshop on Adaptive Hypertext and Hypermedia, Arhus, Denmark, August 14-18, 2001. Available at http://wwwis.win.tue.nl/ah2001
9. Fischer, S. and Steinmetz, R.: Automatic Creation of Exercises in Adaptive Hypermedia Learning Systems, In Proc. of Hypertext 2000. San Antonio, TX. ACM. pp.49-55.
10. Kostelník, R.: An approach to web-based support of learning programming by means of exercises. Diploma thesis. Slovak University of Technology in Bratislava, 2002.
11. Souto, M.A. et al.: Towards an Adaptive Web Training Environment Based on Cognitive Style of Learning: An Empirical Approach. In Proc. of 2nd Conf. on Adaptive Hypermedia and Adaptive Web-based Systems, P. De Bra, P. Brusilovsky, and R. Conejo (Eds.), Malaga, Spain, May 2002. Springer LNCS 2347, pp. 338-347.