Learning Programming with Adaptive Web-Based Hypermedia System AHA!

Mária Bieliková, Jaroslav Kuruc and Anton Andrejko
Institute of Informatics and Software Engineering
Faculty of Informatics and Information Technologies, Slovak University of Technology
Ilkovičova 3, 842 16 Bratislava, Slovakia
{andrejko,bielik,kuruc}@fiit.stuba.sk

Abstract

In this paper we describe an application of open source adaptive hypermedia system called AHA! for learning programming. We present models for adaptive hypermedia systems targeted at the AHA! architecture. Our approach to learning programming is based on adaptive presentation of program examples that serve as program exercises related to particular programming language and programming paradigm. Each program exercise in the domain model is defined as a concept consisting of three basic parts: definition, hint and solution. The domain model also defines several relations typical for educational systems such as a prerequisite together with relations specific to domain of learning programming, e.g., the relation linking a program exercise and a program schema that generalizes the program exercise. Proposed relations allow effective adaptation on several levels of abstraction. We also present extensions devised for purpose of accommodating specific features related to the application domain, such as considering time for adaptation, refined granularity of the concept suitability, or accommodating students' view of understanding particular concept while learning.

1. Introduction

Learning programming requires not only well prepared textbooks (currently often presented using web-based teaching/learning environments) but also a lot of practicing in order to develop a programming skill. For the practice, an idea of using program examples and provide students with solutions to problems is advisable approach taken by several learning systems [9, 3].

Our approach to learning programming is based on developing a programming skill that can only be learned through practice. Program examples that serve as program exercises related to particular programming language are presented to the student together with appropriate guideline for navigation within the space of program exercises. We further improve learning by considering goals and knowledge level together with a context of learning for each student individually. We identified time as an important context attribute in this environment. Taking the time into account can indicate for example that a program exercise is strongly recommended in particular week of semester, or a sequence of exercises recommended for particular student can be influenced by his time available or planned for learning.

Nowadays several web-based adaptive educational hypermedia systems that enhance e-learning by the effective personalized content presentation or navigation exist. Such systems build a model of goals, preferences and knowledge of each individual student, and use this model throughout the interaction with the student, in order to adapt to the needs of that student [2, 5].

In this paper we present models necessary for adaptive hypermedia systems aimed at support of learning programming by providing program exercises. Models are developed within the framework of Adaptive Hypermedia Architecture and realized by the general purpose adaptive system AHA! [7]. Adaptation mechanisms are employed for personalization of navigation within an information space of program exercises. In such a way the educational system guides a student during learning programming. This work is based on our previous experiences with design, development and usage of adaptive web-based educational system ALEA, which serves for purposes of learning programming in the Functional and logic programming course at the Slovak University of Technology in Bratislava since academic year 2002/2003 [8].

The rest of the paper is as follows. In Section 2 we describe the AHA! system concepts. We concentrate on issues related to proposed models. Next, we present structure of the domain model for learning programming. In Section 4 we describe proposed extensions considering time. Section 5 discusses learning approaches and support of models for their realization. The paper concludes with summary and future directions for this research.

2. AHA! Concepts

AHA! is an open source general-purpose adaptive hypermedia system developed and maintained at the Eindhoven University of Technology [7]. Currently the AHA! 3.0 prerelease is available on the AHA! web site

(`http://aha.win.tue.nl/`). AHA! exploits a model of a user characteristics such as goals, preferences, learning style etc. for adaptive content presentation and navigation. It uses combined domain/adaptation model related to the user model through defined concepts in the overlay user model.

Domain model consists of concepts and their relationships [6]. Every information fragment presented to a user as the web page is related to the corresponding concept. Concepts are arranged in a hierarchy and can be shared among several information fragments.

The definition of a concept in the AHA! domain model includes above all *name, resource* (reference to the corresponding information fragment (page) in the information space) and several predefined attributes:

- *access* – the attribute serves the purpose of starting the adaptation engine when the concept resource is accessed;
- *suitability* – the attribute is used for expressing whether the page assigned to the concept is suitable for presentation (it serves for decision about the style of the link to the resource presentation annotation);
- *knowledge* – the attribute stores an integer value which corresponds to the user's level of knowledge of particular concept;
- *visited* – the attribute stores information whether the user visited the resource of the concept or not.

Except above mentioned (predefined) attributes, user defined attributes are allowed.

Concepts can be connected to each other through concept relationships defined in the form of adaptation rules. They can be predefined using templates that represent relationship types. The *prerequisite*, *knowledge propagation* and *knowledge update* are examples of AHA! (predefined) relationship types:

- *prerequisite* relationship is typical for educational hypermedia. It serves for defining appropriate sequences of concepts for effective learning, i.e. the user is advised to visit particular concept only if its prerequisites have been visited;
- *knowledge propagation* relationship serves for automatic propagating knowledge from lower levels to the concepts that are at the higher level according the concept hierarchy;
- *knowledge update* relationship is a unary relationship defined for every page. When the page is read and its *suitability* attribute is true then the *knowledge* of the page is set to the value of 100 (representing situation where the concept is understood).

The *adaptation model* consists of *event-condition-action* rules. It is responsible for performing adaptation based on the knowledge represented in the adaptation model (an update of the user model and consecutive influence of the presentation or navigation according the updated user model). Frequently used event defines situations when the user accesses the page. The condition is expressed as a Boolean expression using attributes of concepts, and the action consists of one or more assignments of values or expressions to attributes of concepts.

AHA! uses an overlay *user model*. Every concept in the domain model is defined also in the user model. The value of each attribute for particular user is stored in the user model. The attribute can be defined as persistent, i.e. its value is stored between several sessions. Except domain model concepts, additional concepts can be defined in the user model. For instance, the concept *personal* is stored only in the user model and stores values of attributes such as the login or password.

The definition of the attribute in the domain model includes also *event-condition-action* rules affecting the way the values of the attributes in the user model are updated. When the event occurs (e.g., a page is shown), rules associated with the built-in *access* attribute are triggered. If the condition associated with the rule is fulfilled, a set of defined actions is fired. Action defines which attribute will be changed together with its new value. The rules can also recursively trigger another rules. By this way, the user model is maintained during the work with the AHA! system.

3. Structure of domain model for learning programming

AHA! system offers two software tools supporting definition of a domain model/adaptation model [6]. Graph Author is a high-level tool for defining conceptual structure using predefined templates of concepts and concept relationships by the graph-based approach. Concept Editor is lower level tool which allows editing native XML representation of the domain and adaptation models using form-based user interface.

The knowledge, which is the aim of the learning, is usually expressed using a text composed in a textbook. Textbook is logically divided into smaller parts – chapters. To make learning more effective, the learning material often contains also supporting tasks and tests, so the student is able to examine newly acquired knowledge. The structure of the textbook is shown in Figure 1. In our implementation of the AHA! models we do not consider tests, external system for testing is supposed.

Learning material (knowledge grouped in a hierarchy of chapters that form the textbook) in a university course is presented during defined period (semester) in several lectures. The most frequent case is to provide students with one lecture per week, but it may vary. The relation between the lecture and the timetable of the university course is shown in Figure 2. Every week of semester can

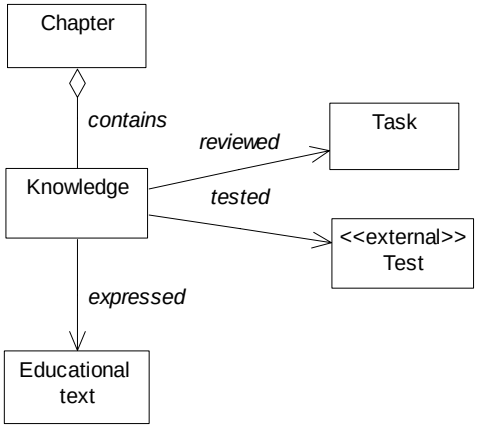

Fig. 1. The structure of the textbook.

have associated one or more lectures. Each lecture is related to the knowledge concept from Figure 1.

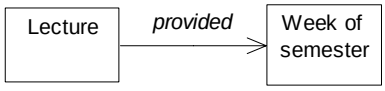

Fig. 2. The relation between the lecture and timetable of course.

To support domain modeling for learning programming, we defined several new templates for concepts, concept fragments and concept relationships for the Graph Author tool. We mention here four basic types of concepts:

- *chapter* – represents consistent logic unit of the learning material, integrates other types of concepts;
- *text* – represents information fragment (page) containing explanatory text (related to program exercises) that corresponds to the text in a textbook or other learning material;
- *program exercise* – represents an exercise related to the given concept in the field of programming (e.g., programming technique);
- *program schema* – represents generalization of solutions of a group of program examples that is related to particular programming technique.

Since personalization of the learning environment is advisable according to the student's knowledge and ability to solve problems, the information fragment representing a program exercise is divided to several parts. Every part is assigned to the concept fragment and can be presented in various ways to the student (e.g., using annotations or hiding fragments). We define three types of the program exercise concept fragments [8]:

- *specification* – an information fragment where the specification of given problem is presented;
- *hint* – an information fragment where a hint to the solution for the student is given;
- *solution* – an information fragment where the solution to given specification is presented, usually associated with annotated the source code.

Proposed structure of the learning material in the field of learning programming is shown in Figure 3.

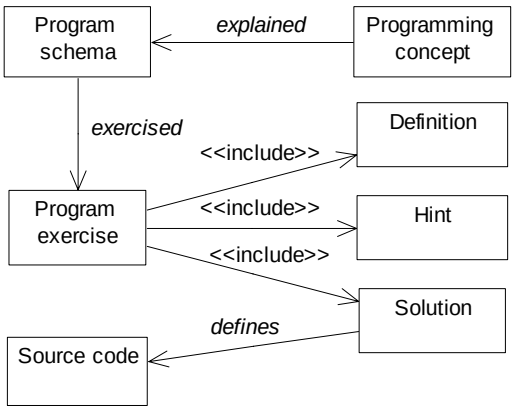

Fig. 3. The structure of the learning material in the field of programming.

According to the level of knowledge of particular student, some of the fragments are not shown. For beginners, a hint would be helpful, whereas for students with advanced programming skills this information may be redundant.

For efficient adaptation it is necessary properly estimate the student knowledge level related to particular concept representing knowledge. Estimating the student knowledge level is a complex task. It is usually performed using various on-line tests. To improve reflection of the student's knowledge of the domain in the user model, we proposed an extension to the standard knowledge update mechanism.

Our solution is to support feedback mechanism from the student, so she is able to "tell" the system, whether she understands presented information. In AHA! appropriate way to achieve this is to extend every concept with the user changeable attribute, in our case attribute called *understood*, with an initial value `false` [10].

Realization of described approach requires extending the user interface with a button (or other kind of the form element) to provide a student with the feedback mechanism. Corresponding knowledge update mechanism is as follows:

- if the visited concept is suitable, but not understood by the student, the value of the *knowledge* attribute is set to definite value in the range 0-100 (we have used in the first prototype

the value of 65, which corresponds to common understanding of suitability in the AHA! system),

- if the visited concept is suitable and understood, the value of the *knowledge* attribute becomes the highest possible (100).

The *understood* attribute definition in AHA! is as follows:

```
<attribute>
  <name>understood</name>
  <desc>Have you understood this
        page?</desc>
  <default>false</default>
  <type>boolean</type>
  <isPersistent>true</isPersistent>
  <isChangable>true</isChangable>
</attribute>
```

To support decisions related to suitability of the concept resource being presented to the particular student in given context, we extended the *suitability* attribute. Predefined *suitability* attribute is Boolean type. We extended it to integer, which allows having several levels of suitability.

4. Considering time

Educational courses are usually provided in particular time period, which is divided into smaller parts. For university study typical period for the course is semester divided into the number of weeks. In this environment, time-based adaptation of the information related to the study is effective [1]. Current version of the AHA! system does not support notion of time implicitly. There does not exist any mechanism for considering time during adaptation.

We proposed a solution in the form of adding a new attribute called *timepos* to the user model representing the student's actual position in the course timetable. The value of the attribute could be interpreted relatively (e.g., the week of the semester), or absolutely (e.g., actual date). In the first case, a manual update of this attribute by the user is possible. Thus, the student is able to select study advancement as needed. In the second case, automatic update by a background process is defined.

To affect the presentation considering time, mapping of the relevant timetable position to the concepts in the domain model is required. This information could affect user's goals and concepts suitability, where combination of actual timetable position relative to the desired timetable position assigned to the concept and concept knowledge may estimate the suitability of the concept. Presented approach to considering time is somehow limited. We plan extend it by possibility of assigning more timetable positions and by using time intervals for expressing suitability of the concepts according time.

To support modeling a concept suitability in dependence on time, we have modified *prerequisite* concept relationship template for the Graph Author tool as shown in Figure 4.

The *prerequisite* relationship defines the *suitability* attribute update. When the page associated with the

```
<aha_relation_type>
  <name>prerequisite</name>
  <listitems>
    <setdefault location="destination.suitability" combination="AND">
      source.knowledge &gt; var:50 &amp;&amp;
      destination.timepos == personal.actual_timepos
    </setdefault>

    <generateListItem isPropagating="true" location="source.access">
      <requirement>destination.timepos == personal.actual_timepos</requirement>
      <trueActions>
        <action>
          <conceptName>destination</conceptName>
          <attributeName>suitability_ex</attributeName>
          <expression>source.knowledge + var: 50</expression>
        </action>
      </trueActions>

      <falseActions>
        <action>
          <conceptName>destination</conceptName>
          <attributeName>suitability_ex</attributeName>
          <expression>source.knowledge</expression>
        </action>
      </falseActions>
    </generateListItem>
  </listitems>
</aha_relation_type>
```

Fig. 4. Definition of the prerequisite relation.

source concept of the relationship is accessed, the *suitability* attribute value of destination concept of the relationship is set to a given value. When the concept suitable time position is equal to the time position in the user model, the value is updated to the value of the source concept knowledge increased by definite value (we have used in correspondence of usual interpretation of suitability in AHA! the value of 50). When the concept suitable time position and the time position from user model are not equal, the value is set as the source concept knowledge attribute value. Thus, in case of time match, the destination concept is considered as more suitable.

5. Learning approaches

Considering learning approach in the field of learning programming is important issue. We provide a mechanism for selecting desired learning approach that is based on similar principle as the time-based extension presented in the previous section. The user model is extended by the attribute representing desired learning approach. This attribute should be changeable by the user and should affect suitability of the concepts. The concepts related to the desired learning approach are preferred. Regarding to the concept types for learning origramming presented in the Section 2, when the approach "from general to concrete" is selected, the program schema concepts are preferred against the program exercise concepts. When the approach "from concrete to general", opposite preference is applied.

An example of using learning strategy for adaptation of links in the resource regarding the suitability of the concept associated with the destination page is shown in Figure 5. When a link is marked as conditional, it is annotated considering the suitability of the concept related to the destination page. In our example, the link to the `recursion.xhtml` page will be apparently annotated as suitable, as the presented pattern uses recursion mechanism.

The example also shows the conditional inclusion of fragments. When the "from general to concrete" learning approach is desired, a link to the particular program schema is generated. Following the link, information fragment including schema is showed.

```
<h1>Top-level list mapping schema</h1>
<pre>
  (defun <mapping> (list)
    (cond
      ((null list) NIL)
      (T (cons
           (<transf-function> (first list))
           (<mapping> (rest list)))  )))
</pre>
```

In case of "from concrete to general" learning approach, a link to the example is generated.

```
<h1>Top-level list mapping example</h1>
<p>Numbers in list get squared.
    (1 2 3) -> (1 4 9)</p>
<pre>
  (defun map-square (list)
    (cond
      ((null list) NIL)
      (T (cons
           (* (first list) (first list))
           (map-square (rest list))))))
</pre>
```

Described mechanism allows adaptation based on considering cognitive style of learning related to the domain of programming. The adaptive educational systems leads the student over different paths through the information base. We distinguish that 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 program schema in our approach [8]) and finally to practice programming by solving exercises. Another group of students prefers go straight to solve the programming tasks immediately after seeing the explanatory text (if even), then look at the corresponding program schemata and compare solutions with presented generalizations.

```
<h1>Top-level list mapping</h1>
<p>The top-level list mapping program schema maps items according
   specified function ...</p>
<p>Back to the <a href="recursion.xhtml" class="conditional">description
   of recursion</a>.</p>

<if expr="personal.learning_approach == 'top-down'">
   <block>
      <p>Look at the <a href="tl-schema.xhtml">schema</a>.</p>
   </block>
</if>

<if expr="personal.learning_approach == "'bottom-up'">
   <block>
      <p>Look at the <a href="tl-example.xhtml">example</a>. </p>
   </block>
</if>
```

Fig. 5. Information fragment with conditional links.

6. Conclusions

In this paper we presented models for the open source adaptive hypermedia system AHA! aimed at learning programming. Our approach is based on adaptive presentation of program examples, which serve as program exercises related to particular programming language and programming paradigm together with program schemata, which serve as generalizations of learned programming concepts.

The main contribution of this paper is in defining the domain model for learning programming. Each program exercise is specified as a concept consisting of three parts: definition, hint and solution. The domain model includes several relations typical for educational systems such as a prerequisite, but also domain specific relations such as the relation linking a program exercise and a program schema generalizing the exercise. Defined relations allow effective adaptation on several levels of abstraction (based on defined hierarchy of concepts or using different learning strategies such as from general to concrete, or from concrete to general by means of program schemata). We also presented several extensions to the AHA! system that are suitable not only for domain of learning programming. All extensions are incorporated into open source adaptive hypermedia system AHA!, version 3.0. Learning environment can be further improved by incorporating annotations to the program exercises that serve as program examples [4].

Our future work will focus on experimental evaluation of the proposed models, which includes also adjusting constants that occur in definitions of relations, for example increase of the suitability attribute in case of recognizing the concept comprehension.

Acknowledgements

This work has been supported by the Scientific Grant Agency of Slovak Republic grant No. VG1/ 0162/03 and the Cultural and Educational Grant Agency of Slovak Republic grant No. KEGA 3/2069/04.

References

[1] Bieliková, M., R. Habala, "University Course Support by Web-Based Adaptive e-Board", In *Proc. of ICETA 2004*, F. Jakab, L. Samuelis, I. Sivý, M. Bučko (Eds.), Košice, Slovakia, September 2004, pp. 395-402.

[2] Brusilovsky, P., "Methods and techniques of adaptive hypermedia", *User Modeling and User-Adapted Interaction*, Vol. 6, No. 2-3, pp. 87-129, 1996.

[3] Brusilovsky, P., "WebEx: Learning from examples in a programming course", In *Proc. of WebNet'2001, World Conf. of the WWW and Internet*, W. Fowler, J. Hasebrook (Eds.), Orlando, FL, USA, AACE, October 2001, pp. 124-129.

[4] Brusilovsky, P., M. Yudelson, S. Sosnovsky, "An adaptive E-learning service for accessing Interactive examples", In: *Proc. of World Conf. on E-Learning – E-Learn 2004*, J. Nall, R. Robson (Eds.), Washington, DC, USA, November 2004, AACE, pp. 2556-2561.

[5] Bureš, M., I. Jelínek, "Description of the Adaptive Web System for E-learning", In *Proc. of the IADIS Int. Conf. e-Society 2004*, Lisboa: IADIS Press, July 2004, vol. 2, pp. 988-991.

[6] De Bra, P., N. Stash, D. Smits, "Creating Adaptive Applications with AHA!", Tutorial for AHA! version 3.0. In *AH 2004 Tutorials,* L. Aroyo and C. Tasso (Eds.), Eindhoven, The Netherlands, August 2004, pp. 1-29. Available at: http://aha.win.tue.nl/aha3-tutorial.pdf

[7] De Bra P., A. Aerts, B. Berden, B. De Lange, B. Rousseau, T. Santic, S. Smits, N. Stash, "AHA! The adaptive hypermedia architecture", In *Proc. of the ACM Conf. on Hypertext and Hypermedia*, Nottingham, UK, August 2003, pp.81-84.

[8] Kostelník, R., M. Bieliková, "Web-Based Environment using Adapted Sequences of Programming Exercises", In *Proc. of Int. Conf. on Information Systems Implementation and Modelling – ISIM 2003*, M. Beneš (Ed.). Brno, Czech Republic, April 2003, pp. 33-40.

[9] Redmiles, D. F., "Reducing the variability of programmers' performance through explained examples", In *Proc. of INTERCHI'93 – Conf. on Human Factors in Computing Systems*, Amsterdam, The Netherlands, April 1993, ACM, pp. 67-73.

[10] Szöcs, V., "*Adaptive hypermedia for teaching programming*", Final Bachelor theses supervised by M. Bieliková, Slovak University of Technology in Bratislava, 2005.