AHyCo: a Web-Based Adaptive Hypermedia Courseware System

Natasa Hoic-Bozic[1] and Vedran Mornar[2]

[1] Department of Computer Science, Faculty of Philosophy, University of Rijeka, Croatia
[2] Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia

Adaptive hypermedia courseware systems resolve the problem of users' disorientation in hyperspace through the adaptive navigation and presentation support. We describe the AHyCo (Adaptive Hypermedia Courseware) — an adaptive Web-based educational system for creation and reuse of adaptive courseware with emphasis on adaptive navigation support and lessons sequencing. The proposed model consists of the domain model, the student model, and the adaptive model. The system is composed of two environments: the authoring environment and the learning environment.

Keywords: Web-based adaptive educational systems, adaptive hypermedia, adaptive navigation support, courseware authoring, AHyCo (Adaptive Hypermedia Courseware) system.

1. Introduction

Traditional computer-aided education has been greatly enhanced by utilizing the Web-based hypermedia systems. Despite their advantages, some problems related to the usage of such WWW systems become apparent as well, particularly concerning the disorientation of the users, or the "getting lost in hyperspace" problem (Maurer & Scherbakov 1996). An adaptive hypermedia system (AHS) resolves this problem by adapting the presentation of hypermedia content or links, based on the user model (Brusilovsky 1996).

In this paper we describe the AHyCo — an adaptive educational hypermedia system (AEHS) for creation and reuse of adaptive courseware with emphasis on adaptive navigation support and lessons sequencing. The purpose of lessons or curriculum sequencing technology is to provide a student with the most suitable sequence of knowledge units to learn (Hübscher 2000). It helps the student to find an optimal navigational path through the material to be learned (Hoic-Bozic & Mornar 2001a).

The model we propose marks all the links and suggests which page the student should visit, according to the student's knowledge and some specific attributes of each lesson. A main principle for our approach is to maximize the space that the user may explore and to accomplish a trade-off between free exploration and guided exploration.

The proposed model consists of the domain model, the student model, and the adaptive model. The system is composed of two environments: the authoring environment and the learning environment.

The remainder of this paper is organized as follows: in section II some more recent achievements in the area of adaptive hypermedia systems have been mentioned, section III describes the domain model, student model and adaptation model of the AHyCo system, section IV describes some of the implementation issues, section V presents the authoring and learning environments, section VI describes the use and evaluation of the AHyCo system, section VII presents conclusions and future plans.

2. Background

According to Brusilovsky (1996), with the term *adaptive hypermedia systems* we denote all hypertext and hypermedia systems that reflect some

features of the user in the user model and apply this model to adapt various visible aspects of the system to the user. An adaptive hypermedia system (AHS) adapts the presentation of content or links, based on the user model. Two major technologies in adaptive hypermedia are distinguished: adaptive presentation and adaptive navigation support. Adaptive presentation adapts either the content of a document or the style of the text. Adaptive navigation support concentrates on changing the presentation of links.

The most popular area for adaptive hypermedia research is the educational hypermedia, where the goal of a student is to learn the material on a particular subject (Brusilovsky, 1996). The most important element in educational hypermedia is the user knowledge of the subject that is being taught. Certain student may know almost nothing about the same lesson that may be trivial and boring for another. In both cases the students need navigational help to find their way through the knowledge space.

A number of first generation adaptive hypermedia systems (Carver, Hill, & Pooch, 1999) were built between 1985 and 1993. They were generally standalone PC or Macintosh-based systems with limited adaptability through stereotype-based user models and limited adaptation techniques. *ISIS-Tutor* is a good example of a first-generation adaptive system (Brusilovsky, & Pesin, 1994).

Since 1993 the Web has become the primary platform for developing educational AHS (Brusilovsky, 1999). These second-generation AHS were generally platform independent. They introduced new features such as adaptive multimedia. Some examples are *ELM-ART* (Brusilovsky, Schwarz, & Weber, 1996), *InterBook* (Eklund, & Brusilovsky, 1998), *DCG* (Vassileva, 1997), *AHM* (Da Silva, 1998), *CALAT* (Nakabayashi, 1997), *KBS Hyperbook* (Henze, & Nejdl, 2000), *ALICE* (Kavcic, 2001), *AHA* (De Bra, & Ruiter, 2001) and *AHA!* (De Bra, et al. 2003), *NetCoach* (Weber, et al. 2001), *ALE* (Specht, et al., 2001).

The second-generation AHS mostly use link annotation. A variant of the overlay model for representing the student's knowledge is used, sometimes in combination with stereotypes. The educational state of the concepts from the domain model is updated if user visits the page that presents the concept's content. Some of the AHS (*CALAT, InterBook, ELM-ART, KBS Hyperbook, ALICE, NetCoach*) use tests as additional and more reliable criteria.

The most important shortcoming of an AHS is the authoring part. The process of authoring should include the development of the actual hypermedia content (lessons, tests, etc.) and the definition of the rules for adaptation.

In order to motivate more authors to use the adaptive hypermedia, the authoring process should be made much simpler than in some existing GUI-based authoring tools (*NetCoach, ALE*). The authoring component should enable the straightforward creation of concepts, the linkage of concepts by prerequisite relationships, and easy generation of the test questions. It should be user-friendly enough to enable a person who is not a computer expert to design the courseware. That includes the development of a graphic editor for concept networks, which will enable the authors to define the prerequisite relationships with a drag-and-drop interface.

In our AHS *AHyCo*, particular attention is given to the authoring component of the system. *AHyCo* user interface is form-based. We strictly separate the learning dependencies (prerequisite relationships) from the actual content (multimedia fragments). This separation allows the authors to use the same set of fragments from a domain to build diverse courseware by simply defining new prerequisite relationships and the values for adaptation rules (Hoic-Bozic, & Mornar, 2003).

AHyCo attempts to provide a complete courseware management system for practical courses. *AHyCo* is delivered as an Open Source software (*AHyCo*, 2003). It uses an easy to use graphical drag-and-drop interface for the definition of prerequisite relations between learning objects. *AHyCo* is designed to enable the authors to develop adaptive learning courses without the knowledge of programming.

3. The Model of the AHyCo

Our model of an adaptive educational hypermedia system consists of the domain model, which describes the structure of the learning

domain as a set of reusable concepts linked together with prerequisite relationships, the student model encompassing the student's knowledge of the learning concepts, and the adaptive model which contains rules for adaptation (Hoic-Bozic & Mornar, 2001b).

3.1. Domain Model

A domain model of the proposed AHS has a two-level structure and consists of concepts as elementary pieces of knowledge for the given learning domain (Fig. 1).

For the first domain level the AHS use the form of a graph $(\mathcal{C}_k, \mathcal{LC}_k)$, where $\mathcal{C}_k$ is the set of concepts and $\mathcal{LC}_k$ is the set of arcs, $\mathcal{LC}_k \subseteq \mathcal{C}_k \times \mathcal{C}_k$. Links represent the prerequisite relationships $\prec$ that denote pedagogical constraints, for example $C_i \prec C_j$ means "concept C_i should be learned before concept C_j". The AHS distinguish between the lessons C_i and tests T_j as the concepts or graph nodes. Tests contain the questions about the domain lessons.

A lesson C_i is defined as a

$$(\mathcal{FC}_i, \mathcal{PC}_i, \mathcal{Q}_i, R_i, wc_i, lMy_i)$$

where:

$\mathcal{FC}_i$ is a set of multimedia fragments (small building blocks, e.g. a piece of text, graphics, sound, video clip...).

$\mathcal{PC}_i$ is a set of prerequisite concepts, which are essential for the student to understand the lesson C_i.

$\mathcal{Q}_i$ is a set of questions related to lesson C_i. All questions are multiple-choice/multiple-or-single-answer, so each question is defined as $(\mathcal{FQ}, q, \mathcal{A}, a, \mathcal{B})$ where:

$\mathcal{FQ}$ is the set of multimedia fragments that form the question.

q is the weight of the question or the confidence level of the fact that student either knows the lesson if he/she answers the question correctly, or does not know the lesson if he/she answers incorrectly, $q \in (0, 1]$.

$\mathcal{A}$ is the set of offered answers. Each offered answer A_j is an ordered pair. The first element is a set of multimedia fragments, or a function $f_j(p_1, p_2, \ldots, p_n)$ that evaluates a candidate's answer on the basis of parameters $p_1, p_2, \ldots, p_n$. Function f_j is defined in a scripting language and will be evaluated after the random generation of the parameters. The second element is v_i, the value of the i^{th} answer, which can be positive (for correct answers) or negative. More obvious answers generally should have a smaller value.

$\mathcal{B}$ is the set of lower and upper bounds for parameters $p_1, p_2, \ldots, p_n$.

R_i is the rank of the lesson C_i calculated as

$$R_0 = 0$$
$$R_k = \max R_j + 1, \quad j \mid C_j \in \mathcal{PC}_k.$$

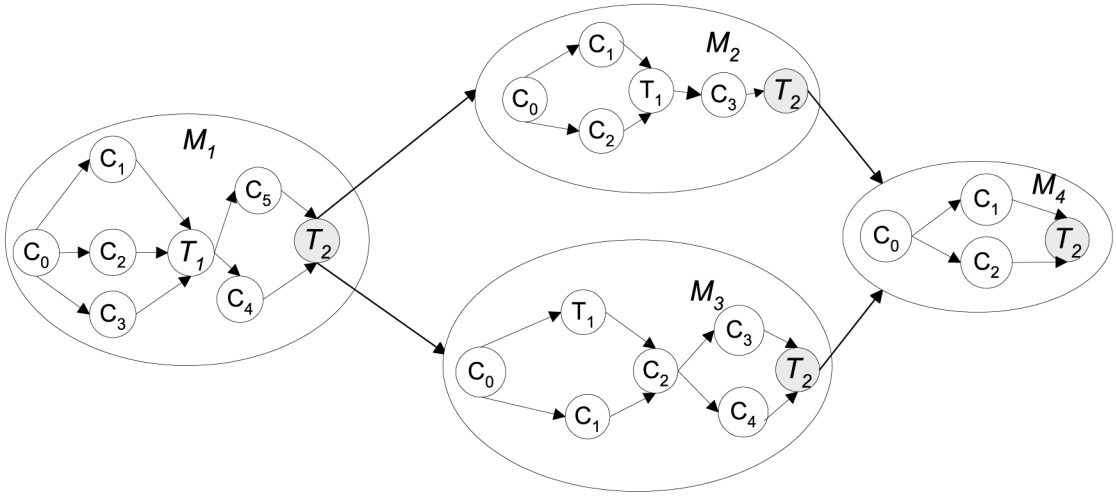

Fig. 1. An example of a domain model.

wc_i is the weight of the lesson, $wc_i \in (0, 1)$.

lMy_i is the minimum acceptable knowledge level for MYCIN model, $lMy_i \in (-1, 1)$.

A test T_j is defined as a $(\mathcal{PT}_j, n_j, \mathcal{N}_j, R_j)$ where:

$\mathcal{PT}_j$ is a set of prerequisite concepts-lessons, $\mathcal{PT}_j = \{C_i \mid C_i \prec T_j\}$. T_j will contain the questions related to lessons C_i.

n_j is the total number of questions in T_j.

$\mathcal{N}_j$ is the set of configuration rules that specify how many questions for each concept $C_i \in \mathcal{PT}_j$ are placed into the test T_j.

R_j is the rank of the test T_j.

To split the domain into more manageable units, concepts are grouped into modules M_k. The second level of the domain model is a directed graph $\mathcal{D} = (\mathcal{M}, \mathcal{LM})$, where $\mathcal{M}$ is the set of modules and $\mathcal{LM}$ is the set of arcs, $\mathcal{LM} \subseteq \mathcal{M} \times \mathcal{M}$. The arc connecting modules M_k and M_l exists if $M_k \prec M_l$. The directed graph $\mathcal{D}$ represents the class the student has enrolled.

A module M_k is defined as a $(\mathcal{C}_k, \mathcal{PM}_k, lm_k, R_k)$ where:

$\mathcal{C}_k$ is a set of concepts that create the module.

$\mathcal{PM}_k$ is a set of prerequisite modules for module M_k.

lm_k is the minimum acceptable knowledge for module, $lm_k \in (-1, 1)$.

R_k is the rank of the module M_k.

The domain model distinguishes between the two kinds of tests: the mini-tests or quizzes T_k for modification of the navigation within the module, and one final test T_f for the navigation between the modules. The final test is the node with the highest rank in the graph $(\mathcal{C}_k, \mathcal{LC}_k)$.

The proposed representation for the domain structure is suitable for storing learning materials from different areas (computer science, mathematics, medicine, art, etc.). The structure of the knowledge is not necessarily hierarchical (chapters, subchapters, pages) but rather concept-oriented (Hoic-Bozic & Mornar, 2001a).

The pedagogical constraints between the lessons depend on the subject area. For example, for math lessons, $C_i \prec C_j$ usually means that it is impossible for a student to start learning C_j if he/she has not gained the knowledge of C_i. For some other subject areas, the prerequisite relationships simply denote the sequence of learning, proposed by the author. For the same set of lessons, another teacher may choose a completely different learning sequence, i.e. prerequisite relationships.

3.2. Student Model

For the AHyCo system, a two-level student model as a variant of the overlay model (Brusilovsky, 1996) for representing the student's knowledge is proposed. The first level represents the estimate of students' knowledge about the lessons. The second level represents the knowledge about the modules. For every lesson C_i the main attributes recorded for each student are r_i and k_i. For every M_k the AHS is recorded the knowledge value about module km_k.

r_i is the estimate about whether the student has read the lesson C_i or not. k_i is the estimate about the student's knowledge of the lesson C_i and is calculated using a variant of the MYCIN model, a widely used expert systems' model (Anjaneyulu 1997, Ng & Abramson, 1990).

The knowledge value of a lesson is set by testing and can range from -1 (student does not know the lesson) to 1 (student knows the lesson). Before the student takes any of the tests, all lessons in the student model have an initial value of $k_i = 0$.

After answering each test question related to lesson C_i, the new knowledge value k_i' for the lesson C_i is calculated according to (1). The new value k_i' is based on the previous knowledge value k_i and the factor q, the question weight or the confidence level of the fact that the student knows or does not know the lesson. If the student answers the question correctly, $f = q$, otherwise $f = -q$.

$$k_i' = \begin{cases} k_i + (1 - k_i) \cdot f, & k_i > 0,\ f > 0 \\ k_i + (1 + k_i) \cdot f, & k_i < 0,\ f < 0 \\ (k_i + f)/(1 - \min(|k_i|, |f|)), & \text{otherwise} \end{cases} \quad (1)$$

The model asymptotically increases/decreases the knowledge value with each correct/incorrect answer according to the previous knowledge value k_i and the question weight q from the domain model (Hoic-Bozic & Mornar, 2001b).

The knowledge value km_k of a module M_k is set according to (2).

$$km_k = \sum_{j|C_1 \in \mathcal{C}_k} k_j * wc_j \qquad (2)$$

According to the formula, more important lessons for the module (with higher weight wc_j) have greater influence when calculating the knowledge level km_k.

3.3. Adaptation Model

The adaptation model consists of adaptation rules that define how the domain model and the student model are combined to perform adaptive navigation support.

In our system, we employed the adaptive navigation, which is a combination of free (open) and guided (forced) navigation. The student can freely follow any hyperlink within a module or graph $(\mathcal{C}_k, \mathcal{LC}_k)$, but a list of hyperlinks is offered that suit him best, according to the navigation plan generated for him. The AHyCo use the combination of link sorting and link annotation adaptive techniques. The navigation within a graph $(\mathcal{M}, \mathcal{LM})$ is restricted and depends on the student's knowledge value km_k.

According to the student model and currently displayed lesson C_a, the concepts from module M_k are classified into several subsets: learned concepts $\mathcal{CL}_a$, recommended concepts where all prerequisite concepts have been visited $\mathcal{CC}_a$, and not recommended concepts $\mathcal{CN}_a$. There are also completely recommended concepts $\mathcal{CP}_a$ or recommended concepts that are in direct prerequisite relationship with C_a.

Concepts C_i from the $\mathcal{C}_k$ of the module M_k are classified according to the algorithm 1:

Input: graph $(\mathcal{C}_k, \mathcal{LC}_k)$, active lesson $C_a \in \mathcal{C}_k$

Output: sets $\mathcal{CL}_a, \mathcal{CP}_a, \mathcal{CC}_a, \mathcal{CN}_a$

$\mathcal{CL}_a = \emptyset, \mathcal{CP}_a = \emptyset, \mathcal{CC}_a = \emptyset, \mathcal{CN}_a = \emptyset$
for each $C_i \in \mathcal{C}_k \backslash \{C_a\}$
  if $r_i = true$
    /* C_i is visited */
    $\mathcal{CL}_a = \mathcal{CL}_a \cup \{C_i\}$
  else if $C_a \prec C_i$
    if $r_j = true, \forall C_j \in \mathcal{PC}_i \backslash \{C_a\}$
      /* all prerequisites for C_i are visited except C_a –
      C_i is completely recommended */
      $\mathcal{CP}_a = \mathcal{CP}_a \cup \{C_i\}$
    else
      /* C_i is not recommended */
      $\mathcal{CN}_a = \mathcal{CN}_a \cup \{C_i\}$
  else if $r_j = true, \forall C_j \in \mathcal{PC}_i$
    /* all prerequisites for C_i are visited
    – C_i is recommended */
    $\mathcal{CC}_a = \mathcal{CC}_a \cup \{C_i\}$
  else
    /* C_i is not recommended */
    $\mathcal{CN}_a = \mathcal{CN}_a \cup \{C_i\}$

Algorithm 1. Classifying the concepts $C_i \in \mathcal{C}_k$.

Navigation within the module M_k goes on before the student solves the final test T_f. This navigation is actually the traversing of a directed graph $(\mathcal{C}_k, \mathcal{LC}_k)$, following the hyperlinks suggested by the system on the bottom of the page. The model uses mini-tests or quizzes T_j to check the students' knowledge and to correct the student model while he/she navigates within the module. Transition to another module is possible after the successful completion of the final test T_f (Hoic-Bozic & Mornar, 2001a). There are three possible outcomes of the test T_f belonging to the module M_k:

> T_f is completely passed — the M_k is learned: knowledge value $km_k > lm_k$ and $k_i > lMy_i$, $\forall C_i \in \mathcal{C}_k$. The student can proceed to another module according to the directed graph $(\mathcal{M}, \mathcal{LM})$;
>
> T_f is partially passed — the concepts C_i with $k_i \leq lMy_i$ are offered for repetition but the student can proceed to another module since the $km_k > lm_k$; M_k is partially learned;
>
> T_k is not passed — $km_k \leq lm_k$ the concepts C_i with $k_i \leq lMy_i$ from M_k are offered for repetition and the student should retake the test T_f in order to proceed to another module.

The student model is updated after the test T_j has been solved according to the algorithm 2:

Input: the student model, lm_k and lMy_i from the domain model, results of the test T_j, $C_i \in \mathcal{PT}_j$

Output: updated values of k_i, rp_i, km_i

use MYCIN for calculating k_i
if $\exists C_i \in \mathcal{PT}_j, k_i \leq lMy_i$
    set $rp_i = 1$
    M_k is partially learned
if $T_j = T_f$
    /* T_j is final test for M_k */
    calculate value km_k of M_k
    if $km_k > lm_k$
        M_k is learned
    else M_k is partially learned

Algorithm 2. Updating the student model after T_j.

4. Implementation Issues

In the development of the AHyCo system we rely on the Microsoft .NET technology. ASP .NET provides a new server-side control architecture which facilitates the development of highly interactive web pages. It has an event-based programming model, making web development much more like traditional VB forms programming. An average ASP.NET page requires less code than an equivalent ASP page, which leads to greater developer productivity and better maintainability. ASP.NET pages are also compiled, so web servers running ASP.NET applications significantly exceed the performance and scalability levels of previous ASP applications (Anderson et al. 2001).

The AHyCo network application is based on the relational database management system. The system components (Fig. 2) are:

a) relational database

b) authoring interface

c) midle-tier component for communication between Web application and database

d) ASP.NET Web application.

4.1. Relational Database

All information about the subject matter and the students are stored in a Microsoft SQL Server 2000 database.

The main part of the database model is the learning and testing subschema. The formulas and smaller multimedia fragments of lessons and questions are stored in the database as binary objects. Larger multimedia objects, such as video or audio clips, are stored in the file system, and are connected to the lessons by hyperlinks.

For security reasons, the set of IP addresses that are allowed to access the system can be specified.

4.2. Authoring Interface

AHyCo system uses Microsoft Access forms as an interface for the authors. Because the lessons and the questions are stored in the database as Word objects, the author can use the familiar Microsoft Word application for updating. Word

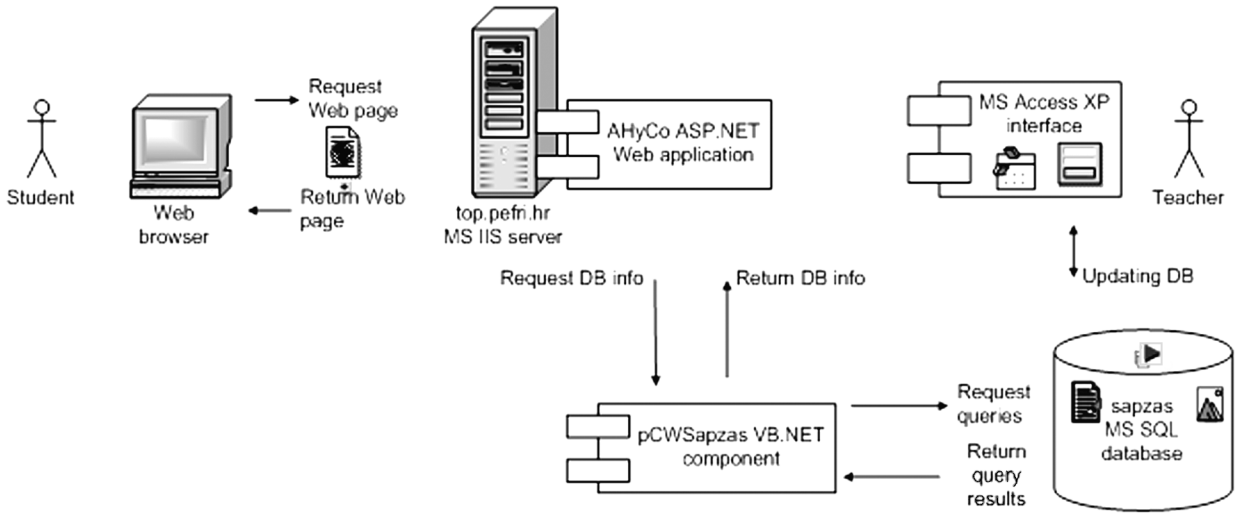

Fig. 2. The components of the AHyCo network application.

is chosen because of its programming features: the fragments $\mathcal{FC}_i$ can be easily connected, updated and converted to HTML or PDF format. To enhance performance of the system and ensure better scalability, the lessons and questions are also stored on the Web server as sets of HTML documents during the courseware preparation phase.

The author names the lessons C_i and groups them into modules M_k. The modules are grouped into subjects. The prerequisite relationships beetween concepts and modules are defined.

The author should also determine the weight of the questions (q) and lessons (wc_j) and the minimum acceptable knowledge level for each concept in the module (lMy_i) and for modules themselves (lm_k). This elements form the variable part of the adaptation rules.

In our system we separate the learning dependencies (prerequisite relationships) and the variable part of the adaptation rules from the actual content (multimedia fragments). This separation allows the authors to use the same set of fragments from a domain to build diverse courseware by simply defining new prerequisite relationships and values for the adaptation rules (Hoic-Bozic & Mornar 2001a).

4.3. Middle-tier Component

The midle-tier component is responsible for communication between the Web application and the database. It contains the logic of the system — the rules for adaptation (Anderson et al. 2001).

To generate the pages, the system consults the middle-tier component. This component determines which content will be shown next, according to the adaptation rules and information from the domain model and the student model. Based on the information returned from the middle-tier component, appropriate HTML fragments are composed together by the Web application that is responsible for displaying the lesson or test question simultaneously with the rest of the page (hyperlinks and buttons for navigation).

4.4. Web Application

The learning environment is implemented as a Microsoft ASP.NET C# web application. This is the only part of the system with which the students interact. They use the Web browser-based interface to the learning environment of the AHyCo courseware.

The web application needs to perform the following tasks: login and validate the student, list subjects related to the student, display lessons' content and navigational elements, display tests' questions and submit the students' answers, display the test results.

Web pages presented to a student are generated adaptively (on the fly) when the student requests them, based on the contents extracted from the database. There are three kinds of pages: lessons, questions and special pages (e.g. login page, help, and test results page).

5. Authoring and Learning Environments

The system is composed of two environments: the authoring environment and the learning environment. The authoring environment is used by teachers to define adaptive courseware materials in various learning domains. The web-based learning environment allows the student to log in and study automatically generated courseware, which dynamically adapts according to his/her success in acquiring knowledge.

5.1. Authoring

Authoring is the crucial task in adaptive hypermedia design. It includes the development of both the network of lessons and the tests. Before the courseware construction has started, the author should divide the subject matter into smaller parts (concepts or lessons). The lessons should be connected by prerequisite relations and grouped into modules.

The first authoring step includes creation of the set of hypermedia fragments $\mathcal{FC}_i$ (text, image, ...) that represent the content of the lesson C_i (Fig. 3). The lessons are stored in the database as Word, Excel or PowerPoint objects, so the author can use the familiar Microsoft Office applications for preparation and updating.

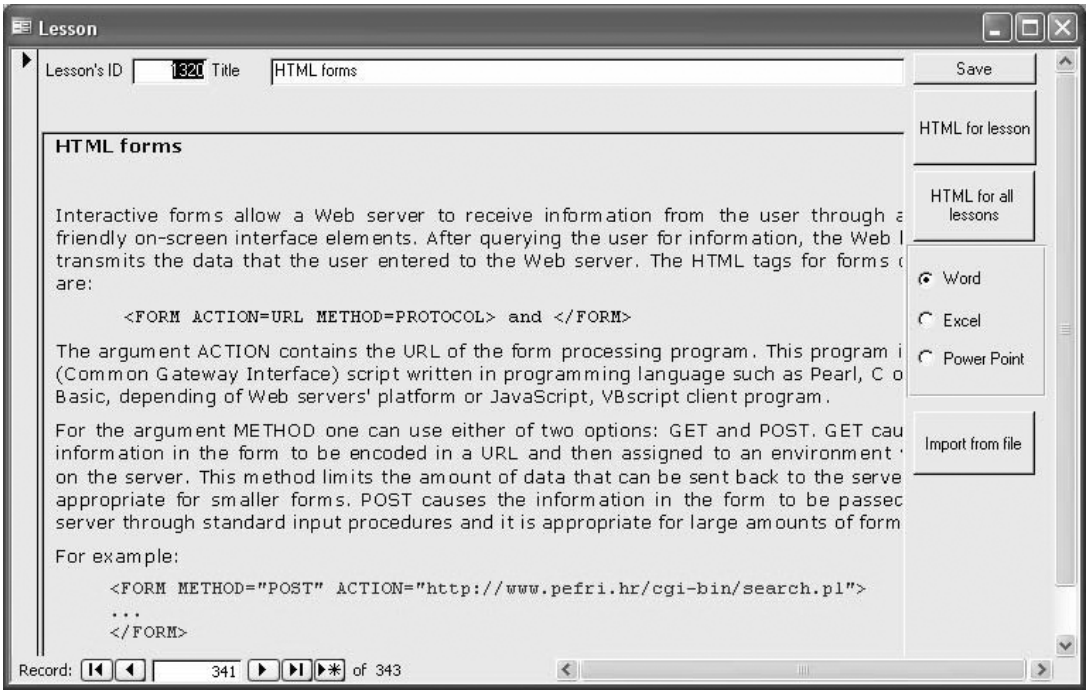

Fig. 3. MS Access form with Word OLE object that contains the text of a lesson.

During the authoring phase, the author sets the rules for adaptation by defining the prerequisite graphs of concepts $(\mathcal{C}, \mathcal{LC})$ and modules $(\mathcal{M}, \mathcal{LM})$. The graphs are created and connected by using a graphic editor with a drag-and-drop user interface (Fig 4.).

When defining the lessons in a module, the author determines the weight wc_i of the lesson for the particular module, as well as the minimum acceptable knowledge level for each lesson in the module lMy_i. When defining the test questions, the author defines the weight q of the questions for the associated lesson. Tests will be created randomly, so explicit enumeration of questions is not necessary, only the structure of the test should be specified. For each module in a domain, the minimum knowledge level lm_k is set. Hence non-technical educators are able to use the adaptive features in a simple manner. The authors do not need to take into account any other adaptation rules, or to use any other tool. The authoring process is thus concentrated on the definition of prerequisites and on the establishment of the difficulty level for lessons and tests. The responsibility for adaptive navigation remains on the *AHyCo* system (Hoic-Bozic & Mornar, 2001b).

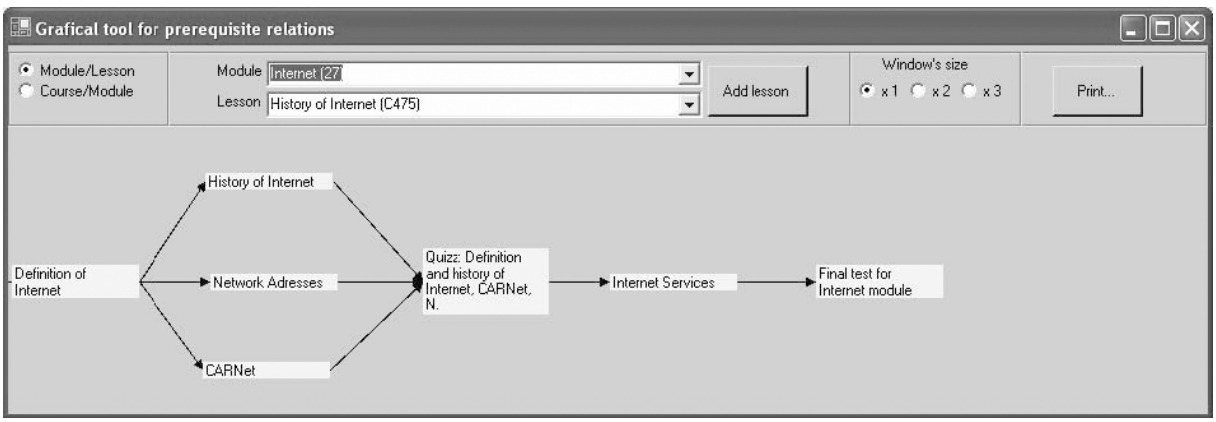

Fig. 4. Creating a graph of concepts.

5.2. Learning Environment

To use the AHyCo learning environment, a student has to log in.

After the process of authorization, the student has to choose the subject for learning. For the selected subject $\mathcal{D}$, the Web page containing the lesson C_i is generated. This lesson is chosen in accordance with adaptation rules and the data stored in the student model, to correspond with the students' previous knowledge. The upper part of the page (Fig. 5) is static and represents the content of a lesson.

At the bottom of the page, hyperlinks to the continuation lessons or tests T_j proposed by the system are enumerated. The suggested hyperlinks are automatically generated before the page is shown and are annotated with various colors corresponding to concept types.

The concepts are listed in the following order:

- Completely recommended or main concepts — green color annotates the concepts where all prerequisite concepts have been visited and these concepts are the best continuation for C_i according to the directed graph $(\mathcal{C}_k, \mathcal{LC}_k)$.
- Recommended concepts — orange color annotates all other concepts where all prerequisite concepts have been read.
- Not recommended concepts – red color annotates the concepts where some of the prerequisite concepts have not been read or the knowledge level of some prerequisite $k_i \leq lMy_i$.
- Visited concepts — blue color annotates the lessons with $r_i = true$ or $k_i > lMy_i$.

All the hyperlinks within the hypertext network are functional, so the student can follow any hyperlink. The idea of free navigation is only to support and aid students. It is up to individual student if he/she will follow the system's suggestions. After the student has finished learning of the module's lessons, he/she should choose the final test button. Test questions (Fig. 6) and the sequence of the offered answers are generated randomly.

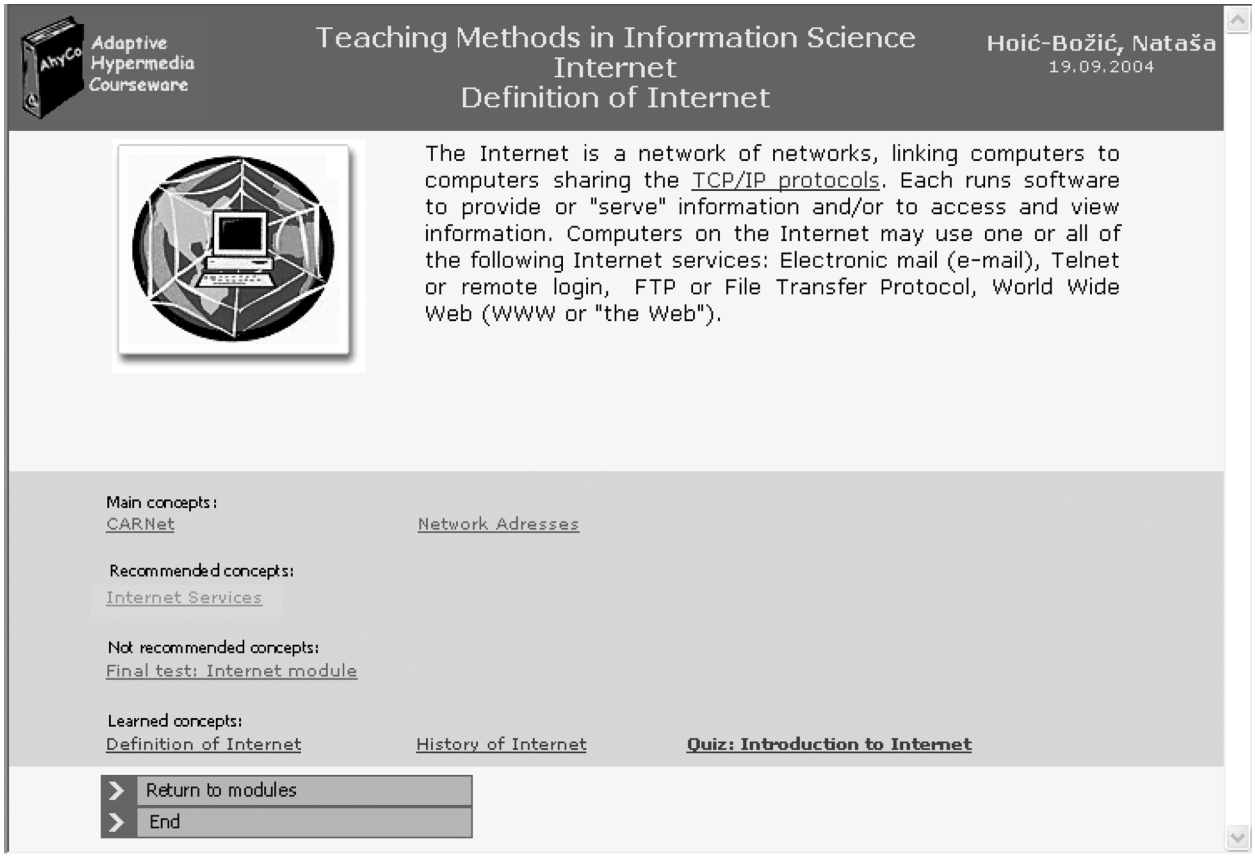

Fig. 5. The page with the content of a lesson.

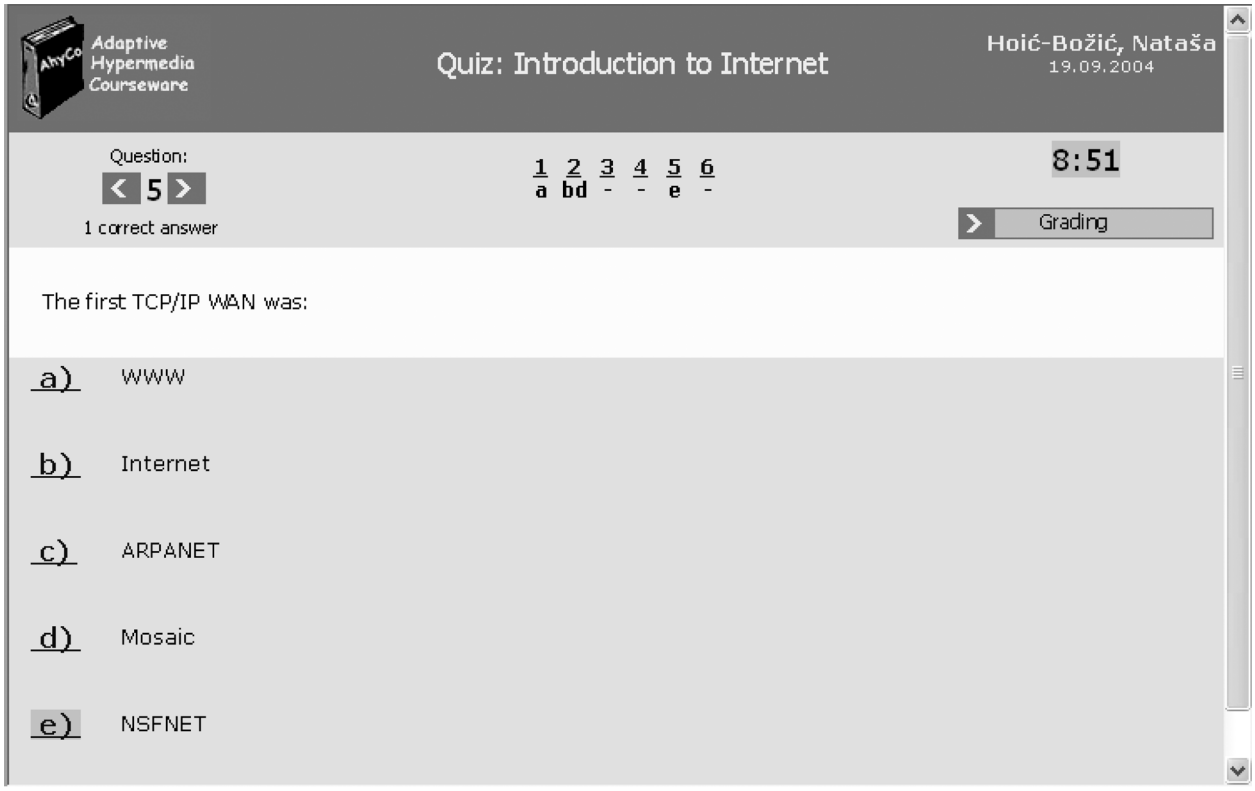

Fig. 6. The question page.

Transition to another module is possible after the successful completion of a test T_f. Outcomes of the tests by individual lessons are displayed separately.

6. Using and Evaluating the AHyCo System

The AHyCo system is currently being used in teaching the students at the Faculty of Philosophy, University of Rijeka.

The purpose of this research was partly to explore how the use of adaptive hypermedia system improves students' learning and to find out the students' attitude concerning the AHyCo usage.

However, the main goal of the AHyCo system evaluation was to find out how the adaptation rules could be corrected according to the students' results in gathering the knowledge. The system enables the diagnostic of the success in the adaptive courseware authoring. According to the students' knowledge results, the teacher finds out possible shortcomings in the developed learning materials. After that, the author should correct the adaptation rules. In that way, the teachers – authors learn how to create and structure learning materials for use with AHyCo system in a better way.

A research has been conducted at the Department of Information Science, Faculty of Philosophy, Rijeka on 19 senior students of Information Science in the context of the class "Seminar on Teaching Methods in Information Science". Part of the class's subject matter has been presented as six AHyCo modules. Each module has various number of concepts linked together with prerequisite relationships.

In the context of the class, the students, as future teachers in schools, learn how to use information and communication technology in education. So AHyCo system has been useful not only as a learning tool, but also as an example of using computer technology and new teaching methods.

The students learned how to use AHyCo system during the summer semester of the academic year 2001/2002. They used computers in PC-room with LAN connected with modem

connection to Croatian Academic Research Network (CARNet) or to their own home computers.

After the students had finished learning with AHyCo, the evaluation of the system was performed according to their knowledge levels about the concepts (k_i) and modules (km_k).

Despite of the fact that the main purpose of the evaluation was not to show the level of students' success in learning, we could mention that all the students accomplished the goal and successfully solved the final test T_f of the last module M_6.

Actually, the main goal was to find out how to correct the variable part of the adaptation rules (the question weight q, the lesson weight wc_i, the MYCIN level lMy_i, the module level lm_k) and to verify if the prerequisite relationships were correctly defined (prerequisite graphs of modules ($\mathcal{M}$, $\mathcal{LM}$) and concepts ($\mathcal{C}$, $\mathcal{LC}$)).

Analyzing the basic statistic data (mean, median, standard deviation, quartile values) and the correlation coefficients for k_i and km_k, as well as by cluster analysis, recommendations for the authors were proposed about the corrections of the adaptation rules.

In order to explore the students' attitude concerning the AHyCo usage, the questionnaire about the effectiveness and quality of AHyCo and the level of students' acceptance of AHyCo as a teaching resource was developed. According to the questionnaire results, the students accepted the new way of learning with AHyCo system. The main difficulty regarding the learning conditions was in gaining access to AHyCo courseware since computers at the Faculty were not available all the time.

7. Conclusion and Future Work

In this paper we have discussed the model and implementation of AHyCo system for development and distribution of the adaptive Web-based courseware. The navigation model is based on adaptive lessons sequencing and adaptive navigation support techniques. To accomplish a trade-off between free and guided navigation, the list of suggested hyperlinks is offered at the bottom of the page, in the optimal order generated by the system. However, a student is allowed to follow any of them.

In order to verify the results, the sample courseware was generated at the Department of Computer Science, Faculty of Philosophy, Rijeka. The main purpose of the evaluation was to find out how adaptation rules could be corrected according to the students' knowledge levels about the concepts and modules.

According to the results of statistical analysis, recommendations for authors have been proposed with regard to correction of adaptation rules and improvement of prerequisite graphs.

In the further development of this research, the developed AHyCo system will be improved based on the results obtained during the development and evaluation. We will modify the sample AHyCo courseware according to the proposed rules and verify the new version on the next generation of students.

References

[1] AHyCo Home Page (in Croatian), (2003), `http://AHyCo.zpm.fer.hr/`

[2] R. ANDERSON ET AL. *ASP.NET*, Wrox Press, Birmingham, 2001.

[3] K. ANJANEYULU, Concept Level Modelling on the WWW", *Proceedings of the AI-ED 97 8th World Conference on Artificial Intelligence in Education, Knowledge and Media in Learning Systems*, (1997), Kobe, Japan.

[4] P. BRUSILOVSKY, L. PESIN, ISIS-Tutor: an intelligent learning environment for CDS/ISIS users, *Proceedings of the interdisciplinary workshop on complex learning in computer environments* (CLCE'94), (1994), Joensuu, Finland.

[5] P. BRUSILOVSKY, Methods and Techniques of Adaptive Hypermedia, *User Modeling and User-Adapted Interaction*, 6 (1996), pp. 87–129.

[6] P. BRUSILOVSKY, E. SCHWARZ, G. WEBER, ELM-ART: an intelligent tutoring system on World Wide Web, *Proceedings of the Third International Conference on Intelligent Tutoring Systems ITS'96*, (1996), Montreal, Canada.

[7] P. BRUSILOVSKY, Adaptive and intelligent technologies for Web-based Education, *Künstliche Intelligenz, Special issue on intelligent systems and teleteaching*, 4 (1999), pp. 19–25.

[8] A.C. CARVER, J.M.D. HILL, U.W. POOCH, Third generation adaptive hypermedia systems, *Proceedings of WebNet 99, World Conference on the WWW and Internet*, (1999), Honolulu, Hawaii.

[9] D.P. DA SILVA, Concepts and documents for adaptive educational hypermedia: a model and a prototype, *Proceedings of the 2nd Workshop on Adaptive Hypertext and Hypermedia HYPERTEXT'98*, (1998), Pittsburgh, USA.

[10] P. DE BRA, J.P. RUITER, AHA! Adaptive hypermedia for all, *Proceedings of the WebNet Conference*, (2001), Orlando, Florida.

[11] P. DE BRA ET AL. AHA! The Adaptive Hypermedia Architecture, *Proceedings of the ACM Hypertext Conference*, (2003), Nottingham, UK.

[12] J. EKLUND, P. BRUSILOVSKY, Individualising interaction in Web-based instructional systems in higher education, Presented at *The Apple University Consortium's Academic Conference*, (1998), Melbourne, Australia.

[13] N. HENZE, W. NEJDL, Extendible adaptive hypermedia courseware: Integrating different courses and Web material, *Proceedings of Adaptive Hypermedia and Adaptive Web-Based Systems AH 2000*, (2000), Trento, Italy.

[14] N. HOIĆ-BOŽIĆ, V. MORNAR, An Approach to Navigation Support in Adaptive Educational Hypermedia System, *Proceedings of Multimedia and Hypermedia Systems Conference MIPRO 2001*, (2001), Opatija, Croatia.

[15] N. HOIĆ-BOŽIĆ, V. MORNAR, Navigation Support in a Web-Based Adaptive Educational Hypermedia System, *Proceedings of AMCIS 2001*, (2001), Boston, Massachusetts.

[16] N. HOIĆ-BOŽIĆ, V. MORNAR, An approach to adaptive hypermedia courseware authoring, *Proceedings of Hypermedia and Grid Systems*, MIPRO 2003, (2003), Opatija, Croatia.

[17] R. HÜBSCHER, Logically Optimal Curriculum Sequences for Adaptive Hypermedia Systems, *Proceedings of Adaptive Hypermedia and Adaptive Web-Based Systems AH 2000*, (2000), Trento, Italy.

[18] A. KAVCIC, *Adaptation in Web-Based Educational Hypermedia With Regard to the Uncertainty of User Knowledge* (Ph.D. Thesis, in Slovenian), Ljubljana, Slovenia: Faculty of computer and information science, University of Ljubljana, 2001.

[19] H. MAURER, N. SCHERBAKOV, *Multimedia authoring for presentation and education: the official guide to HM-Card*, Addison-Wesley, Bonn, 1996.

[20] K. NAKABAYASHI, Architecture of an intelligent tutoring system on the WWW, *Proceedings of the 8th World Conference of the AIED Society*, (1997), Kobe, Japan.

[21] K. NG, B. ABRAMSON, Uncertainty Management in Expert Systems, *IEEE Expert*, **2** (1990), pp. 29–47.

[22] M. SPECHT, M. KRAVCIK, L. PESIN, R. KLEMKE, Authoring Adaptive Educational Hypermedia in WINDS, *Proceedings of ABIS2001*, (2001), Dortmund, Germany.

[23] J. VASSILEVA, Dynamic course generation on the WWW, *Proceedings of AI-ED 97- 8th World Conference on Artificial Intelligence in Education*, (1997), Kobe, Japan.

[24] G. WEBER, H.C. KUHL, S. WEIBELZAHL, Developing adaptive internet-based courses with the authoring system NetCoach, *Proceedings of Third Workshop on Adaptive Hypertext and Hypermedia*, (2001), Sonthofen, Germany.

Received: October, 2002
Revised: February, 2005
Accepted: May, 2005

Contact address:
Natasa Hoic-Bozic
Department of Information Science
Faculty of Philosophy
University of Rijeka
Omladinska 14
HR-51000 Rijeka
Croatia
e-mail: natasa.hoic@ri.t-com.hr

Vedran Mornar
Department of Applied Mathematics
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3
HR-10000 Zagreb
Croatia
e-mail: vedran.mornar@fer.hr

NATASA HOIC-BOZIC is an assistant professor at the Department of Information Science, Faculty of Philosophy, University of Rijeka. She received Ph.D. degree in computer science from the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia. Her main research interests include information and communication technology (especially hypermedia and Internet) in education. She participated in several research projects in the field of e-learning.

VEDRAN MORNAR is a professor of computer science at the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia, where he currently teaches several graduate and undergraduate computing courses. He graduated and received his PhD degree in computer science from the same university. His professional interest is in application of operational research in the real world information systems, database design, development and implementation.