Adaptive Web-Based Educational Hypermedia

Paul De Bra, Lora Aroyo, and Alexandra Cristea
{debra,laroyo,acristea}@win.tue.nl

Eindhoven University of Technology
Eindhoven, The Netherlands

Abstract

This chapter describes recent and ongoing research to automatically personalize a learning experience through *adaptive educational hypermedia.* The Web has made it possible to give a very large audience access to the same learning material. Rather than offering several versions of learning material about a certain subject, for different types of learners, adaptive educational hypermedia offers personalized learning material without the need to know a detailed classification of users before starting the learning process. We describe different approaches to making a learning experience personalized, all using adaptive hypermedia technology. We include research on authoring for adaptive learning material (the AIMS and MOT projects) and research on modeling adaptive educational applications (the LAOS project). We also cover some of our ongoing work on the AHA! system, which has been used mostly for educational hypermedia but has the potential to be used in very different application areas as well.

1 Introduction

Education has been and still is changing dramatically. Until recently education was a well-structured process. Young people went to school (primary school, high school, college or university), following courses taught in classrooms by a physically present teacher. This was a continuous process (interrupted only by vacation) that ended with the highest degree the learner would ever obtain. An evolution that we have seen in this process is a shift from purely oral communication, through note taking by the learner and later note preparation by the teacher, to the publication of textbooks that (when they are very good) are beginning to make the physical presence of teachers and learners in a classroom superfluous. Several additional changes are occurring simultaneously, characterized by the three a's *anyplace*, *anytime*, *anyhow*[1]. Learners no longer want to go through one multi-year course program leading to a "final" degree in the shortest possible time. They want to combine study and work, and as a result are not available for a teaching/learning session in a classroom at scheduled times. The education process must allow learners to study at times when their work allows that. The process must not require the students to physically travel to an institute in order to be able to learn. The process must also accommodate different ways of learning, provided these ways lead to the same knowledge and skills.

Internet provides a technological basis for making the "new education" possible, because it is available *anyplace* and *anytime*. However, Internet does not automatically make the teacher available *anyplace* and *anytime*. We need a means to make the teaching/learning process independent of the physical availability of the teacher. For very interactive teaching processes many researchers have tried to use computer technology to create teaching systems that try to *simulate* the intelligence of the teacher. (See, e. g. the many publications in the field of Intelligent Tutoring Systems, or ITS.) A common problem with this approach is that many ITS fail to give the impression of being intelligent. Nonetheless there have been a number of successful ITS like ELM-ART [30, 29] and SQL-Tutor [25, 26]. A different approach is to rely on the intelligence of the learner to select the information and assignments needed to master the subject of a course. This approach puts the emphasis on making information available to the learner. A first step in this direction was the creation of good textbooks. A second step was the transformation of such textbooks into hypermedia form (and putting them on the Web). This step gave learners the freedom to select information and to choose their own reading order. A third step is to make the hypermedia textbook "dynamic" in the sense that it changes itself (possibly in an invisible way) to accommodate the reading order chosen by the learner. The result of this step is what we call *adaptive educational hypermedia*, the subject of this chapter.

Adaptive educational hypermedia provides adaptation regarding the three c's *connectivity*, *content* and *culture*[2]. Connectivity is what distinguishes hypermedia from (text)books: information items are linked together in many ways, offering learners many possible ways to navigate through the information. Adaptation means that the system can offer guidance by making distinctions between links, based on how "appropriate" or "relevant" links are for a given learner. Content adaptation enables the system to offer additional notes to learners who appear to need them (thus compensating for information the learner missed in his/her navigation path), or to omit information the learner appears not to need (because it was acquired elsewhere). Culture represents that learners have different preferences regarding *how* they are able to learn.

[1] The three a's are mentioned very often in presentations, but we have been unable to find out who first came up with them, hence the missing reference.

[2] Again, we don't know the origin of the three c's.

A single adaptive educational hypermedia application must accommodate different learning styles. Adaptation in the three c's realizes the most important of the three a's: *anyhow*.

In the next section we give a sketch of the architecture of adaptive Web-based educational hypermedia. We specifically focus on what a system needs in order to be able to provide adaptation. We refer to some well-known adaptive educational hypermedia systems (some also predating the Web) to illustrate the use of the concepts covered by the architecture. After that we look at some current developments and research ideas that may shape future Web-based educational systems.

2 Adaptive Educational Hypermedia Architecture and Systems

In this section we describe the main architectural elements of adaptive educational hypermedia applications. We do not intend to give a complete reference architecture like AHAM [9, 31] or the Munich Model [23, 24]. We concentrate on the *user model*, a system representation of how the learner relates to the conceptual structure of the application, the way in which the system *updates* this user model, and the way in which the state of the user model affects how the system *adapts* to the learner.

2.1 Web-based Adaptive Systems

Adaptive educational systems predate the Web. The advantages of Web-based systems (mostly the availability anyplace and anytime) may make us forget the problems of Web-based systems. A non-Web-based system can have a tight integration of the user-interface and the underlying functionality. Every action of the user can be recorded: every mouse movement, window size and placement, scrolling, etc. This is not possible on the Web. Web-based systems assume a client-server approach in which the client is fairly dumb. It presents information it receives from the server, and it sends user requests to the server. These consist of either a GET request for another page (as a result of clicking on a link anchor) or a POST request that results from filling out a form, perhaps consisting of answers to questions of an evaluation test, or perhaps specifying a search for certain terms. The "intelligence" resides completely on the server side. (This has the advantage that the same learner can interact with the educational application from any browser anywhere on the Internet.) When the server receives a request (which it can associate to a particular learner) it performs the following series of actions:

1. The system first retrieves the user model from permanent storage (file system or database). It may keep that user model in main memory in between requests, for performance reasons.

2. The system must have a representation of the conceptual structure of the application's knowledge domain. This structure is normally retrieved from permanent storage (and then kept in memory) and thus not integrated into the system.
3. Part of the conceptual structure is a definition of how the user's request and the current state of the user model must be used to calculate a new state of the user model. The user's request reflects some change in the user's knowledge state, and the system must make the user model reflect that change.
4. The user's request results in a response that has to be sent back to the browser. This can be an information page or a dynamically generated page giving feedback on a submitted form (test or search). Again the conceptual structure contains a definition of how that presentation is influenced by the state of the user model. This *adaptation* process normally uses the *new* state of the user model. (We will explain later why.)
5. When the adapted response is sent to the browser the new (updated) user model is saved. Waiting to save the user model until after sending the response to the browser improves the system's response time and enables the system to undo the user model changes (by not saving them) in case sending the response to the browser fails.

In the following subsections we discuss the different parts of an adaptive educational application. Although existing systems vary wildly, they do share some common parts.

2.2 User Model

An educational application is used to learn about certain subjects. The information domain of the application can be divided into such subjects, topics or concepts. In the sequel we shall always use the term *concept*. A *user model* is used to represent how the user *relates* to these concepts. In adaptive applications this notion of *relates* can mean many things, but in education applications it typically represents the learner's *knowledge* about the concepts. A user model that models the learner's knowledge about concepts through a *knowledge value* (or knowledge level) per concept is called an *overlay model*.

There are two ways of modeling knowledge of concepts: *coarse grained* and *fine grained*, and three ways to model knowledge values: *binary*, *enumerated* and *(pseudo) continuous*.

- With *coarse grained* modeling of knowledge we mean that the knowledge domain is modelled using only a few "broad" concepts. Learner actions (like accessing/reading pages and performing assignments or tests) result in changes to the knowledge value for some concept(s). In order to model the gradual increase of knowledge about a broad concept a value domain is needed with many different values. Real numbers between 0 and 1, or integers between 0 and 100 (to mean a percentage) are examples of value domains that can be used.

- With *fine grained* modeling of knowledge we mean that small items like pages or perhaps even fragments of pages or single media objects (like images) are represented as concepts in the user model. When there are many small concepts the learner's knowledge can be represented using simple values about these concepts. A binary value, like *known* or *not known*, can already be sufficient and was used in early versions of the AHA! system for instance [8]. The system becomes more expressive if a few more values are used, like *well known*, *known*, *partly known* and *not known* (used with slightly different terms in Interbook [12] for instance). It does not make much sense to describe the knowledge about small concepts much more in detail (like with a percentage) because the system has very little information to base such a detailed judgement on. (How would you decide whether a user who reads a page has 73 or 74% knowledge of that page for instance?)

Coarse and fine grained user modeling both have advantages. In a system that keeps track of the learner's knowledge of a few broad topics (using a detailed scale of values), one can easily generate an overview of the knowledge that is easy to understand. Also, performing adaptation to the learner's knowledge can be done using these few knowledge variables that have a clear meaning. As a result the adaptive behavior of the educational application is predictable and easy to understand and explain. On the other hand, modeling the learner's knowledge in detail, using many small concepts, perhaps with only a crude representation of knowledge, enables a system to adapt to this detailed knowledge. If the learner does not know a particular term used on a page, the system can add a short explanation of that term, or it can replace the term by a non-technical equivalent. Ideally an adaptive educational system would combine the strong points of both knowledge representations. The system should represent detailed and global knowledge. This implies that (if the same value domain is used for all concepts) a rich value domain is used for small concepts as well as broad concepts, which is in some sense overkill.

In an educational application the user model also contains information about the learner, independent of the specific subject domain of the application. An example of such information is a *learning style*. Some simple aspects of how a learner likes concepts to be presented can be used by an adaptive system. Some learners like to see a few examples before the definition or description of a concept, whereas others prefer to see the definition first and some examples later. Some learners prefer to learn just a bit about a concept and discover the rest through assignments, whereas others prefer to study everything and only perform tests or assignments to verify their knowledge (and not to increase it).

Some educational applications also support additional information about the learner's intensions and goals: the *task model*. Often the learner has to perform tasks only within a specific sub-part of the application and not in the whole subject domain. The task model allows for splitting the subject domain into separate complementing subjects (or topics) and assigns the necessary corresponding tasks in order to achieve a specific learning goal. This way, rather than learning everything, the learner can navigate and focus on a specific subset of the subject domain according to his/her course task. That subset does not necessarily have to correspond to a chapter-structure of a textbook. Within educational systems with concept-oriented domain descriptions often the most natural way of describing a task would be by using the concepts of the user model. It is then also an *overlay model*, and it is typically a fine grained model. The relevance of the concepts for the task is used to determine which parts the learner should study for this task. The task model can also be used to improve a search facility or a graphical concept map, by adding additional terms to a search string or filtering out the non-relevant concepts from the map. Both techniques have been used in AIMS [1, 2] for instance.

2.3 Domain and Adaptation Model

Adaptive educational hypermedia applications deal with a subject domain. At an abstract level that domain can be described using (high or low level) *concepts*, the same concepts that are used in the user model (which explains the term *overlay model*). The application also consists of *pages*, which are often considered as (low level) concepts.

A commonly used modeling approach for the conceptual structure of an educational application is the use of a *concept hierarchy*. This follows the typical structure of a textbook, consisting of chapters, sections, subsections and paragraphs. The system can update the user model by considering the propagation of knowledge from pages to subsections to sections to chapters. By considering high and low level concepts it becomes easier to specify adaptation that depends on details as well as adaptation that depends on the knowledge of a whole chapter.

The concept hierarchy describes the structure of the application domain, but not the way in which the learner should or could navigate through that domain. Considering only the hierarchy the learner could start descending to the page level and start studying in the last as well as the first chapter. (In fact, there need not be a "first" vs. "last" chapter as the hierarchy can be unordered.) Furthermore the *hypermedia* aspect is realized through associative or cross-reference links throughout the application. These links connect pages in completely (structurally) different parts of the application, based on the meaning of the pages. Following arbitrary links may lead to *orientation* and *comprehension* problems:

- Following arbitrary links through any hypermedia structure always causes a risk of not knowing where you are. This problem is independent of

whether the application is of an educational nature or not. Solutions usually involve displaying part of the structure surrounding the "current" page, possibly adapted to the user in some way. This is called "map adaptation" in Brusilovsky's taxonomy [10, 11].
- The comprehension problem is typical for educational hypermedia: it is likely that a user will be able to follow a path that leads to pages using technical terms without passing by pages that explain these terms. This problem can be tackled in two ways: by blocking such paths or by adding the explanation when needed.

Adaptation in educational hypermedia is most often based on what are called *prerequisite relationships* between concepts or pages. "A is a prerequisite for B" means that the learner should "know" A before studying B. Defining prerequisite relationships between high-level concepts may be hard for a teacher. But prerequisite relationships that indicate that a page containing an explanation of a term should be read before going to a page that uses that term are easy to identify. The adaptation offered by the system relieves the author from the task of ensuring that all possible navigation paths obey the prerequisites. The system can enforce the use of valid navigation paths or compensate for the missing knowledge. The way in which the system adapts the presentation of information and links by using the prerequisites is defined in what the AHAM model [9] calls the *Adaptation Model* (AM).

- In Interbook [12] the prerequisite relationships are used to provide *annotation* to the links, in the form of colored balls. When the learner is ready to read a page that offers new (not previously studied) information a green ball is shown next to the links (actually link anchors) to that page. If the learner is not ready, meaning that some prerequisite knowledge is missing, the annotation becomes a red ball. Interbook offers more adaptive features that also use this color metaphor. A "teach me" button on a page (with unsatisfied prerequisites) generates a list of (links to) pages that need to be studied in order to acquire all the prerequisite knowledge for the current page. In this list the colored balls appear again, to indicate which pages to study first (the ones with green balls). There is always a reading order that only contains "recommended" pages, because the structure of prerequisite relationships must be *acyclic*. The entire adaptive behavior of Interbook is "hard-wired" into the system. Interbook thus has a fixed, built-in Adaptation Model.
- In AHA! [8] the standard behavior is that of *link hiding*. Links normally appear in blue (unvisited) or purple (visited). But when a link leads to a page for which the learner is missing some prerequisite knowledge the link anchor is displayed in black, and is not underlined. The anchor is therefore indistinguishable from plain text and effectively "hidden". As we describe below AHA! does not really use prerequisite relationships but a powerful event-condition-action rule language that is capable of expressing many different types of concept relationships, including prerequisites of course.

- Brusilovsky's (original) taxonomy [10] mentions three other types of *adaptive navigation support*: *direct guidance*, in which the learner is shown a "next" button or some similar indication of the "best" next page to read, *adaptive sorting of links*, meaning that a list of links is sorted from most relevant to least relevant, for this particular user, and *map adaptation*, which we explained earlier. All techniques for adaptive navigation support try to point out the recommended link(s) to the user.
- A completely different way to deal with prerequisites is to *compensate* for the missing knowledge, by adding a *prerequisite explanation* to a page for which the learner is not really ready. This technique is used in the AHA!-based hypermedia course at the Eindhoven University of Technology (see http://wwwis.win.tue.nl/2L690/). It is a special case of a more general technique of *conditional inclusion of fragments*, implemented in AHA! as well as in other system, like C-Book [21] for instance. In C-Book the technique is also used to offer *additional* or *comparative* explanations. Learners who are familiar with Pascal are given comparisons between C constructs they are learning and similar Pascal constructs they already know. These comparisons are useless to non-Pascal programmers and therefore only shown to Pascal programmers. In the *stretchtext* technique the recommended fragments are shown to the user, but the hidden fragments are accessible through (small iconic) links or buttons. Using this technique all information is available to everyone.

The techniques mentioned above are the most commonly used techniques, but there are many more possibilities. In the AHAM model [9] one can define arbitrary concept relationship types, and associate them with user model updates and adaptation. An example is the *inhibitor* type, which can be viewed as the opposite of a prerequisite. An on-line course may for instance contain several alternative descriptions of the same concept. Once the learner has studied the concept through one of these descriptions access to the alternative descriptions may be inhibited. Similarly, a page may contain some explanation that is automatically omitted when the learner already has some specific knowledge.

Figure 1 shows an authoring interface developed for the AHA! system. On the left it shows the concept hierarchy. This hierarchy is used to indicate how pages contribute knowledge to higher-level concepts. When (like in the example) there are two pages contributing knowledge to the concept "beer" each page contributes 50%. And since there are two concepts ("beer" and "chocolate") contributing knowledge to the concept "belgium" each contributes half again. As a result reading a page contributes 25% knowledge to "belgium". These values can be altered, much in the same way as in the system described in [19]. On the right the concepts (dragged from the left) can be connected using concept relationships of different types. Here the solid arrows mean "interest relationships" and the dashed arrows are prerequisites. They indicate that one first needs to read about a "plain" beer (Stella) before going on to a more local, special beer (De Koninck), and that one needs to read about a fairly common chocolate (Meurisse) before going to a real Belgian specialty brand (Callebaut).

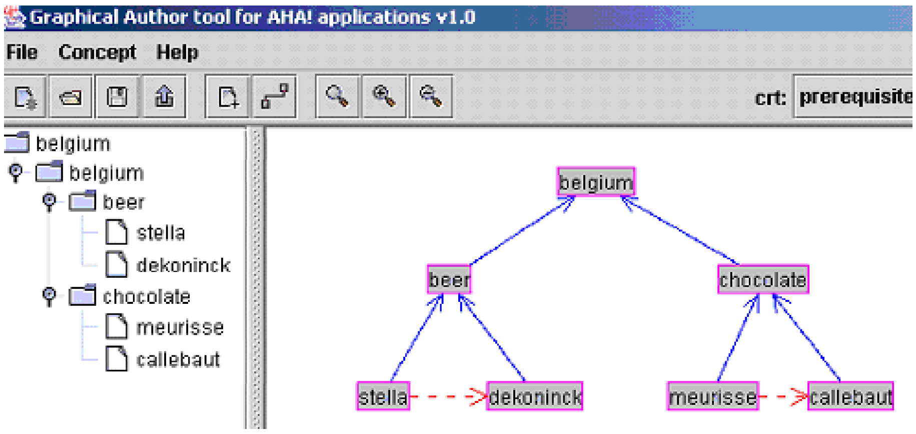


Fig. 1. AHA! authoring tool for concept hierarchy and concept relationships.

Whereas an interface like Figure 1 defines the application domain at an abstract level, and expresses some desired adaptation aspects, it does not define the desired behavior of the system in terms of the user model updates and the link and content adaptation. Indeed, as shown above, there are different ways to perform adaptation based on prerequisite relationships. In most adaptive educational platforms that "performance" is completely hard-coded. But in AHA! it is defined through a set of event-condition-action rules that define this behavior. This approach was taken to reflect the low-level functionality suggested by the AHAM model, and is described in Section 3.

The common behavior of adaptive educational systems is to *first* update the user model, taking into account the knowledge that will be gained by reading the requested page. In a *second* phase the page is generated, conditionally including fragments that depend on prerequisite or other concept relationships, and annotating or hiding links based on these relationships. It is important to note this order of actions. Intuitively one would expect a page to be presented based on the user model state *prior to* presenting the page. This makes perfect sense because if the user is missing some prerequisite knowledge that can be compensated for by inserting an explanation, the system must look at the user model state before generating the page in order to conclude that the prerequisite explanation must be inserted. However, if the page to be presented contains prerequisite knowledge for other pages to which the page also contains links, one wishes these links to be shown as "recommended". Since there is no (easy) way to change the presentation of a page once it is shown to the user these links cannot first be shown as "not recommended" and after some reading time changed to "recommended". Hence it is necessary to show the links as "recommended" as soon as the user retrieves the page. It is clearly a limitation or drawback to first update the user model and then generate the presentation, but for the inclusion of prerequisite explanations it turns out not to be a big problem as the prerequisite explanation normally does not replace the full explanation. In other words, the page with the prerequisite explanation included should not generate enough knowledge about the missing prerequisite concept to make the prerequisite explanation not be included (or at least not the first time this page is accessed). It takes careful system design to ensure that the counterintuitive order (of first updating the user model and then generating the presentation) indeed produces the expected presentation. In Section 3 we show how this is done in AHA! using its event-condition-action rules.

2.4 Assessment

It is an illusion that an adaptive educational hypermedia system can accurately assess a learner's knowledge by observing which pages the learner reads. The technique can be refined, by observing the reading time, by taking into account which fragments are included in the pages the learner reads, and perhaps even by performing eye-tracking to check which parts of a page the learner really reads. All these more or less intrusive ways of monitoring the learner fail to grasp what really goes on in the learner's mind. Therefore many adaptive educational hypermedia applications resort to a different technique for measuring the learner's knowledge: tests.

Multiple-choice tests are most popular because, although they are difficult to create, they are easy to evaluate. In theory a multiple-choice test can be implemented using "plain" adaptive educational hypermedia. A page can be shown that contains a question and several answers. Each answer is a link to another page that explains why the answer is right or wrong. Accessing these pages (by clicking on the answer) performs a user model update just like accessing "normal" pages. The difference is that when reading information pages knowledge can only increase whereas clicking on "answer" pages may cause the knowledge value to decrease. In AHA! [8] for instance every page access can cause knowledge increases as well as decreases.

Most systems (including AHA!) offer a separate module for (multiple-choice) tests. In the AHA!-based hypermedia course the learner is first guided to three introductory chapters. Each chapter ends with a multiple-choice test. Access to (unhiding of the links to) the advanced chapters is granted after completing the three multiple-choice tests, not after reading most or all the pages of these chapters. At the end of the course another test, containing 15 (randomly selected) questions, is included before access is granted to the final assignment. Initially the test contained a fixed sequence of 20 questions. But after having this course available for some time we discovered a drawback of the *anytime* (and perhaps *anyhow*) aspects of Web-based learning: more and more students achieved a perfect score on the difficult final test. It turned out that the correct sequence of answers was being communicated among students. This taught us that in an on-line course the student assessment needs to use a different test for each student (or at least enough randomness that communicating the answers becomes difficult).

Multiple-choice tests can be very useful in initial tests, to determine the entry-level of the learner, and also to determine the level at different points in time. In a (Spanish) Linux course Romero [28] has used a modified AHA! application with initial and final multiple-choice tests. Log files from this course were used (with association rule mining) to find out potential problems in the course. If information pages and followup tests are not well matched this will show up in the test results. (If the test is too straightforward after reading the pages then virtually all the students will answer the questions correctly. If the test is too difficult, e. g. because the information needed to answer the questions cannot be found in the preceding information pages then many students will give incorrect answers. Likewise, if the information is there but either it is wrong or the answer indicated as the correct one is wrong, then many students will get a low score as well.)

2.5 Learning Objects and Learning Standards

Since adaptive educational hypermedia applications are centered around the notion of *concept* one may wonder whether the architecture compares to the use of *learning objects*, as used in different standards or standards initiatives, such as the LOM (Learning Objects Metadata) standard from IEEE Learning Technology Standards Committee (See http://ltsc.ieee.org/), or the SCORM (Shareable Courseware Object Reference Model) from the US Government Advanced Distributed Learning (ADL) Initiative, or IMS (Instructional Management System) from the Global Learning Consortium. A comprehensive page about such initiatives can be found at http://www.learnativity.com/ (under the heading "standards and learning objects", accessed on July 1, 2003). The focus of these initiatives is on distribution and sharing. In order for teachers to construct courses from material gathered from different sources, a clear description is needed of what that material entails. Therefore, each piece of instructional material, called a "learning object" is tagged with a significant amount of metadata. Just like concepts, learning objects can be large or small, entire courses or lectures as well as a single illustration, application or applet, or a paragraph of text.

Although an author can build a course from learning objects that reside on different servers (they need not be copied to a single location in order to use them, and in the case of server applications they often even cannot be copied), it is difficult to use the learning objects and their metadata to perform *adaptation*.

We expect that sometime in the near future the worlds of adaptive educational applications and of distributed learning environments will come together, but at the moment of this writing they are still too far apart to consider the use of these learning standards in *adaptive* educational systems.

3 AHA! The Adaptive Hypermedia Architecture

In 1994 we started offering a course on hypermedia at the Eindhoven University of Technology. This course went through several generations and is now available as http://wwwis.win.tue.nl/2L690/. In 1996 we started adding adaptive functionality to the course, hence AHA! was born. The first version was based on CGI-scripts written in C. The browsers used at that time lacked the ability to "play" with link colors through style sheets, a feature we currently use to create the *good*, *neutral* and *bad* link colors (which are blue, purple and black by default). Therefore, instead of simply "hiding" unrecommended links, the anchor (<A>) tags were removed, thus disabling the use of unrecommended links as well as hiding them. Conditional inclusion of fragments was done through C-preprocessor constructs (#if statements). See [13] for details.

In the next version [7] the C-preprocessor constructs were replaced by HTML comments. Link removal/disabling was still used as in the initial version. The user model in these early versions consists of one concept per page, with a binary value, indicating *known* or *unknown*.

3.1 AHA! Version 1.0

The first version of AHA! used outside the Eindhoven University of Technology is called version 1.0. This is the version used by Cini et. al. [14] to measure the "Presentation Adaptivity Degree" in adaptive applications, and by Romero et. al. [28] to perform association rule mining to help authors of adaptive courses spot anomalies in their courses. AHA! 1.0 supported the main features described in Section 2 as necessary in adaptive educational applications. The main features of AHA! 1.0 are:

- The user model in AHA! 1.0 consists of one *concept* for every page, and possibly many more *abstract concepts*, not associated with a page. For every concept the learner's knowledge is stored as a percentage (an integer value between 0 and 100). This allows more values than needed for pages, but is mostly useful for keeping track of the user's knowledge about larger concepts (like whole sections or chapters of a course).
- For every concept there is a *rule* that defines how a change in the knowledge of the concept influences the knowledge of other concepts. As an example, when the rule for a concept A contains:

```
        B:+40 C:30 D:-50
```

it means that when the knowledge of A is increased by X the knowledge of B will be increased by 40% of X, the knowledge of C will be set to 30 and the knowledge of D will be decreased by 50% of X. Increases or decreases however can never make a value lower than 0 or higher than 100. The change X to the knowledge value of A can be positive or negative.

- For every page there is also a *requirement* that defines the prerequisites. This requirement may look like:

  ```
  E>50 and F<30
  ```

  meaning that this page is “recommended” when the knowledge of concept E exceeds 50 and the knowledge of F is lower than 30. It is thus possible to express *prerequisite relationships* as well as *inhibitors*. When the learner visits a page the knowledge of that page becomes 100 if the page was recommended. If the page was not recommended the knowledge becomes 35 or the previous value, whichever is higher. AHA! thus expresses that reading a page generates partial or full knowledge of that page depending on the satisfaction of prerequisites.
- Because (like in the “beer” example of Figure 1) knowledge often needs to be propagated to a high-level concept the knowledge update rules may trigger each other. The rule for stella would read:

  ```
  beer:+50
  ```

  and the rule for beer would read

  ```
  belgium:+50
  ```

  When reading about “stella” (for the first time) the knowledge of stella jumps from 0 to 100. This causes the knowledge of beer to be incremented by 50 (50% of 100) and that of belgium by 25% (50% of 50). Through these rules one can easily describe the knowledge propagation through a concept hierarchy. One should keep in mind that because of the use of integer values there may be rounding (truncation) errors, causing the knowledge value of a high-level concept to not reach 100 when one would expect it to. (In [31] an example is given to illustrate that the user model updates can depend on the execution order of the rules, because of the truncation in integer arithmetic.)
- An inherent danger of rule propagation is the occurrence of infinite loops. Indeed, rules can interact with each other, generating an infinite sequence of knowledge value updates that go up and down. In order to prevent such loops AHA! 1.0 only performs propagation on monotonic relative updates (the ones with a + sign in the rules). Should there still be a loop in the rules, the effect can only be that several knowledge values are repeatedly updated in the same direction (all increased or all decreased). Such a loop must end when all values become 100 or 0. Because there are only a finite number of concepts and only a finite (101) number of possible values for each concept the loop must end after a finite number of steps.

- AHA! uses the (cascading) style sheet mechanism to tell the browser how to display links. AHA! tells the browser not to underline links, so that black link text is indistinguishable from plain text (without a link anchor). The default link adaptation mechanism in AHA! is that of *link hiding*: recommended links are blue (unvisited) or purple (visited) and unrecommended links are black. The user can change this color scheme, making the unrecommended links more visible. According to Brusilovsky's taxonomy [10] AHA! then supports *link annotation*.
- A special "if" tag is used in pages to indicate conditionally included fragments. The conditions for fragment inclusion are the same as for page recommendation, and can therefore express prerequisites as well as inhibitors.
- AHA! contains some special features, like a module for generating and evaluating multiple-choice tests, an optional page header including the number of pages read or still to be read, links to a list of read pages and a list of unread pages, and an invisible Java applet that makes the server record the reading time for each page.

Modeling the knowledge structure of an educational application is easy in AHA!. Adaptation based on prerequisite relationships is also fairly straightforward.

3.2 AHA! version 2.0: More Adaptation Flexibility

In AHA! 1.0 for every user model concept there is a single integer value between 0 and 100. In the previous section we have implicitly assumed that a concept means a concept from the application domain and that the value means the (system's idea of the) user's knowledge about that concept. However, there is nothing in the AHA! system to support that assumption. It is all in the eyes of the beholder.

Concepts in AHA! are just variables that can be manipulated through rules that are triggered by page accesses. This can be convenient to add adaptation to parts of the information presentation that are not related to knowledge. AHA! does not enforce a specific presentation style or look and feel and is in that respect very different from most other adaptive educational systems. We have created an adaptive course with HTML frames, using a "left" frame to show navigation support in the form of a Windows-Explorer-like hierarchical menu, and a "right" frame to show the information pages. We have used adaptation to make submenus open and close automatically depending on the location of the "current" page in the concept hierarchy. When following a cross-reference link the learner jumps to a completely different position in the concept hierarchy. The Explorer-like navigation aid would follow this jump and show the context of the new page.

AHA! 2.0 [6] makes it a lot easier to create ad-hoc adaptative behavior. Concepts can have arbitrarily many (named) attributes with Boolean, integer or string values. Typical attributes are:

- *access*: This is a Boolean attribute with a system-defined meaning. When a page is accessed, the access attribute of the concept(s) associated with this page becomes true. This triggers the execution of a set of *event-condition-action rules* associated with the attribute. A typical rule for the access attribute would be to increment the *knowledge* of the concept with an amount that depends on the *suitability* of the concept. The access attribute is volatile: its value is not stored in the user model.
- *knowledge*: This attribute represents the user's knowledge about the concept. There are typically some rules associated with this attribute, to "propagate" knowledge to higher-level concepts. This is similar to the way the updates worked in AHA! 1.0. Changes to attribute values (like knowledge) trigger the rules associated with these attributes (of the changed concepts). This "propagation" can be turned on or off for each rule.
- *interest*: This attribute represents the user's interest in the concept. It may depend for instance on the user's task or goal. AHA! can use the attribute value to guide the link annotation or hiding, in order to provide the user with links to pages that correspond to his/her interest.
- *suitability*: This attribute can be used to remember whether the concept is suitable for the user. This can, e. g. mean that the user has all the required prerequisite knowledge. Using the suitability attribute expressions for the conditional inclusion of fragments can also be simplified.
- *visited*: This attribute remembers whether the user visited this concept (usually a page). It is used by AHA! to choose between the *good* and *neutral* link colors, which are blue and purple by default, to mimic the standard browser behavior.

Using the above mentioned attributes and carefully chosen event-condition-action rules the user modeling and adaptation possibilities in AHA! are very powerful. As an example, it is easy to increment a knowledge value by a small amount each time a page is visited. The conditional inclusion of fragment (containing a prerequisite explanation) can be done based on that small amount. If knowledge is initially 0 and incremented by 10 on each visit, a fragment with expression "`knowledge`< 20" will be shown the first time, and hidden from the second visit on. On the first visit the knowledge value goes from 0 to 10 *before* the page generated. The conditional inclusion of fragments thus uses the updated user model values. On the second visit the knowledge value becomes 20, and the fragment is no longer included. The knowledge value can be updated through different pages as well, so a prerequisite explanation can be shown on the first out of a set of many pages on a certain concept, etc.

An Concept Editor (Java applet based) hides the underlying XML syntax of the event-condition-action rules from the author. Two examples of rules (in an arbitrary syntax not used by AHA!) are given below.

E: `stella.access`
C: `stella.suitability`> 50
A: `stella.knowledge`$:= 100$

This rule is triggered when the `access` attribute of `stella` changes, which is the case when the page about stella is accessed. The rule is executed when the `suitability` of stella is high enough, and the action is to set the `knowledge` attribute to 100. A second rule could then be:
E: `stella.knowledge`
C: `beer.knowledge` < 100
A: `beer.knowledge` $+= 0.5*$`_stella.knowledge`
This rule is triggered by a knowledge change of stella and adds 50% of that change to the knowledge of beer if that knowledge is still less than 100. (The underscore _ is used to indicate that then change of the attribute value is used instead of the value itself. It is easy to say that the following rule then performs the 25% increase in knowledge about Belgium, as described earlier:
E: `beer.knowledge`
C: `belgium.knowledge` < 100
A: `belgium.knowledge` $+= 0.5*$`_beer.knowledge`

3.3 Beyond AHA! 2.0: Better Authoring Support

The user modeling and adaptation flexibility of AHA! 2.0 comes at the price: the complexity of authoring. AHA! does not know how to deal with prerequisite relationships. These have to be encoded using attributes and event-condition-action rules. AHA! also does not know about learning styles. They too have to be translated to rules.

Figure 1 shows a first attempt to enable higher-level authoring in AHA!. The available *types* of concept relationships are defined by a system designer, as well as their translation to the AHA! event-condition-action rules. The author simply chooses concept relationship types from a list.

The conditional inclusion of fragments is also being changed in AHA!. In version 2.0 the fragments are parts of the (xhtml) pages, whereas in the upcoming version they will be separate objects. This has the advantage that a fragment that may appear on different pages can be stored just once.

What AHA! continues to lack is support for yet higher-level aspects such as task support and adaptation to learning styles. In the next section we present some new research into adaptive educational hypermedia authoring, and the support for tasks, styles and goals.

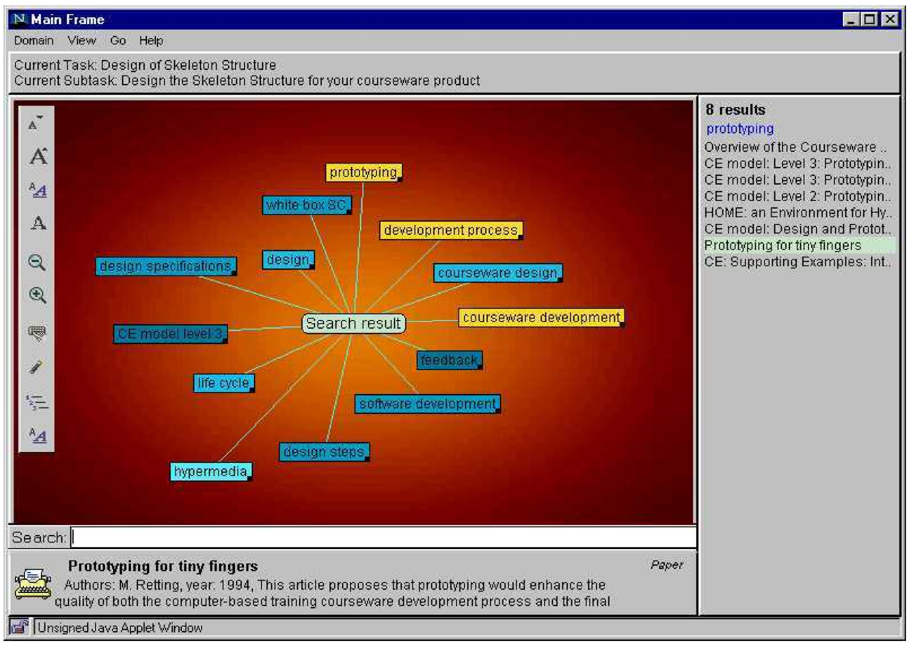
4 New adaptive hypermedia authoring techniques and implementations

The potential benefits of adaptive educational hypermedia for educational applications should be clear from the previous sections. However, surprisingly, we do not find adaptive hypermedia so wide-spread and widely accepted as we would expect. To find an explanation for this fact, we start by looking at the weak points in present adaptive hypermedia systems (AHS): the difficulties in authoring and the lack of support for partial learning, i. e. learning just one subject of a course, or searching for the information needed for one specific task. We proceed by searching for methods for solving the authoring problems and for task-based searching. Sections 4.1 and 4.2 describe ongoing research in task support, considering both the learner's tasks and the authoring tasks. Section 4.3 describes an extension of the AHAM model, considering high-level adaptation to goals and constraints as wel as low level ad-hoc adaptation. Section 4.4 shows how these ideas are being experimented with in an experimental authoring tool.

4.1 Authoring Support Framework for Adaptive (Knowledge-based) Courseware

In the previous sections we have seen examples of AH authoring activities as they are performed within the AHA! system. As concluded above different AHS share common modules with respect to defining domain, adaptation, user and task models. Subsequently, the authoring activities for instantiating these models within various AHS systems also share common functionality. AIMS is an example of an intelligent courseware tool that follows an instructional model (analogous to the adaptation model in AHA!) over a concept domain definition and overlayed user and task models. This way it provides support for instructors to build on-line courses as well as for learners to identify information necessary for performing course tasks (e. g. course assignments) [2]. The system can be used standalone (for example, as an extension to a traditional or on-line distance course) or integrated in a larger electronic learning/training/work environment that allows the users to perform open learning tasks in a specific subject domain. In both cases it provides the user with immediate, on-line access to a broad range of structured information and with domain-related help in the context of work, thus supporting more efficient task performance. In figure 2 the student task-oriented search and browsing environment is presented, which employs the above approach.

To insure this we employ the AHS approach for User and Domain Models and introduce the Resource Model for knowledge classification and indexing based on conceptualization of the course material and subject domain. The Instructional Model here provides the main set of rules in order to link the above models and produce the desired instructional result. In the line of recent advances in the research related to Semantic Web and ontologies [4] AIMS focuses on using ontologies for defining rules and structures for the subject domain, user model construction, course structure building and resource managements. The goal here is twofold. On one hand, an ontology offers means to define vocabulary, structure and constraints for expressing metadata about learning resources. It also offers formal semantics for the primitives defined in the ontological structures and thus, more efficient reasoning useful for the support of both the learning and the authoring processes. On the other hand



Current Task: Design of Skeleton Structure
Current Subtask: Design the Skeleton Structure for your courseware product

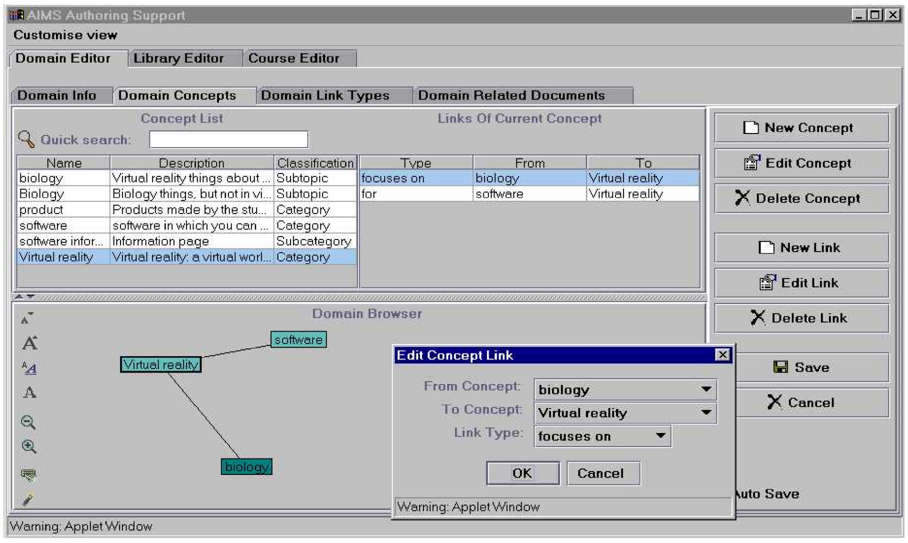
8 results
prototyping
Overview of the Courseware ..
CE model: Level 3: Prototypin..
CE model: Level 3: Prototypin..
CE model: Level 2: Prototypin..
HOME: an Environment for Hy..
CE model: Design and Protot..
Prototyping for tiny fingers
CE: Supporting Examples: Int..

Prototyping for tiny fingers
Authors: M. Retting, year: 1994, This article proposes that prototyping would enhance the quality of both the computer-based training courseware development process and the final

Fig. 2. AIMS student information search and domain navigation environment.

ontologies offer a solution for shareable and reusable knowledge bases among different courses and authors (instructors), which is imperative in educational environments. By reaching these goals we provide a basis for more efficient task-oriented information retrieval and adaptation to the learner status [3], as well as methods to fill the deep conceptual gap between authoring systems and the authors.

As we have seen in AHA! as well as in AIMS, building instructional structures, adaptive navigation and content presentation appears to be an extremely difficult and time-consuming task. The successful implementation of the suggested approaches turns out to depend crucially on the ontological engineering for providing means for sharing and reusing instructional knowledge. Thus, the key focus here becomes the authoring process and the formalization of the knowledge about it which can be employed within an intelligent authoring tool. In this context we explore the construction of and the interoperability between the various ontological structures for domain and instructional modeling and the modeling of the entire authoring process. In this we decompose the authoring process in (1) a set of generic atomic tasks underlying (2) higher-level authoring activities related to the subject domain ontology, dynamic course structure, and resource management metadata. This is the first step towards building a meta ontology of courseware authoring functional concepts (generic authoring tasks), in the same sense as introduced by [22]. The intention is to use it as a basis for defining a corresponding set of tools to support the main phases of ontology engineering for courseware authoring.



AIMS Authoring Support

Customise view

Domain Editor | Library Editor | Course Editor

Domain Info | Domain Concepts | Domain Link Types | Domain Related Documents

Concept List

Quick search:

Name	Description	Classification
biology	Virtual reality things about ...	Subtopic
Biology	Biology things, but not in vi...	Subtopic
product	Products made by the stu...	Category
software	software in which you can ...	Category
software infor...	Information page	Subcategory
Virtual reality	Virtual reality: a virtual worl...	Category

Links Of Current Concept

Type	From	To
focuses on	biology	Virtual reality
for	software	Virtual reality

Edit Concept Link

From Concept: biology

To Concept: Virtual reality

Link Type: focuses on

Warning: Applet Window

Warning: Applet Window

Fig. 3. AIMS authoring environment.

4.2 Authoring Task Ontology

The Authoring Task Ontology (ATO) is based on the notion of "task ontology" defined by Mizoguchi et. al. [27]. It provides operational semantics to the authoring process by specifying, in a domain independent way, all authoring activities, sub-activities, goals and stages based on a common system of vocabulary. This way ATO enables the authoring tools to guide, support, navigate and check the consistency and completeness of the authoring activities at all levels of complexity. In the construction of ATO we consider specification of top-level authoring activities, low-level authoring functions, decomposition of the authoring process into a sequence of phases and decomposition of the authoring process in main activity modules (related to the educational system components). The authoring activities are independent of the system's domain, the educational strategy and the educational goal. The authoring task ontology describes the relations among the authoring tasks and the roles of domain objects, which they play in a particular task [5]. Each authoring task is defined by: (1) a sequence of activities with their activity type, constraints and input/output resources, with their resource type and constraints; (2) a goal; (3) requirements; (4) constraints, which can be flexible or hard constraints. Following this we define a set of:

- generic nouns reflecting the roles of the objects in the authoring process (e. g. CONCEPT, LEARNING ACTIVITY, STRUCTURE, RESOURCE, COURSE, LESSON, DOMAIN, AUTHOR, STUDENT, TEXT, RELATIONSHIP, GOAL, CONSTRAINTS, etc.).
- generic verbs representing authoring activities over the objects and applied in a combination or sequence of activities with specific objects or concepts and their modifications. (e. g. MODIFY, EDIT, ASSIGN RETURN, UPDATE, SELECT, CHECK, etc.).
- generic adjectives representing the modifications of the objects and applied for modification and identification of these objects' attributes (e. g. SHARED, FINISHED, REQUIRED, IDLE, IN-USE, UPDATED, etc.).
- other authoring task specific concepts (e. g. CONCEPT PREREQUISITE, LESSON CONSTRAINT, CONSTRAINT SATISFACTION, STATE, ATTRIBUTE, PREDICATE, KNOWLEDGE, etc.).

The primitive functions are defined on objects (e. g. concepts, documents, course topics and tasks, etc.) within a specific structure such as course structure, goal hierarchy, teaching action sequence, etc. They express a simple functional formalism, where the object changes the structure, or the structure is manipulated. Examples of atomic authoring functions include:

CREATE (Structure), CREATE (Object, Structure), ADD (Object, Structure), DELETE (Structure), DELETE (Object, Structure), VIEW (Objects), EDIT (Object, Structure), LIST (Objects, Structure), UPDATE (Object, Structure), UPDATE (Structure),

where $Object \in Domain\_Concepts$, $Course\_Topics$, $Course\_Tasks$, $Resources$ and $Structure \in Domain\_Model$, $Course\_Model$, $Resource\_Base$.

The hierarchy of higher-level tasks represents conceptual categories of relationships (interdependence) between primitive functions. These present certain aggregation criteria (including causal and other relations among components) that are used for grouping primitive functional concepts into higher-level authoring functions (classes). This way we can construct/identify functional groups of authoring tasks. The higher-level functions represent a role of one base function for another base-function. They are concerned not with the actual change in the objects, but with their actual function in the process of authoring. Examples of links realizing it include: *'is-a-prerequisite-for', 'is-assigned-to', 'is-achieved-by', 'follows-from', 'is-preceded-by', 'requires', 'is-followed-by', 'if-<goal>-then-<action>'*. (Note that the authoring tool shown in Figure 1 already provides a means for creating such relationships in the limited context of the AHA! system.) Examples of some higher-level authoring tasks include: *DeleteLinks, DeletePath, DeleteAll, LinkDocTask, LinkDocConcept, CopyCourse, CompareObjectsStructure, ExistObjectStructure.* All the authoring tasks are grouped in relation to domain model, user model resource management and instructional-task model construction, in order to build a hierarchical organization of concepts linked by the ontological link types *'is-a', 'part-of' and 'attribute'*. With the above description we have formed a functional basis for the authoring support in adaptive educational systems and have illustrated some of the benefits of an ontology-based approach for an authoring framework. In the next section we will present extension of the AHAM reference model in order to achieve author-friendly primitives in terms of which the author can easily describe the goals and constraints of their instruction.

4.3 The LAOS authoring model

LAOS [17] is a general model for adaptive hypermedia authoring, that tries to respond to the problems listed above. It is built to express this multitude of alternatives in an easy-to-edit manner, allowing *collaborative* (many authors working together on an adaptive hypermedia design task) or *incremental authoring* (same author working at different moments in time on the same adaptive hypermedia design task, e. g., after having some first students' feedback; or introducing new requirements form the head office, etc.). LAOS also allows automatic authoring techniques. It is based on the AHAM model [31], which in turn extends the Dexter reference model [20] for the specific field of adaptive hypermedia.

The first difference between LAOS and AHAM is that LAOS has an extra layer: the *Goals and Constraints Model*. AHAM is best suited for applications in which the user needs to explore the whole information space. Specifying goals enables the system to give a more focused presentation. Through constraints the search space can be reduced. Figure 4 shows LAOS as an extension of AHAM [31]. It consists of five layers: *Domain Model*; *Goal and Constraints Model*; *User Model*; *Adaptation Model*; *Presentation Model*.

The other difference between AHAM and LAOS is contained in the granularity and exact composition of elements in the top layers, as well as in the introduced operators. These are a result of the aim of adding automatic authoring techniques, which imply the authoring result to contain the necessary information for the system to process in an expressive manner, in other words, to be based on a good ontology. Therefore we looked for a better expression for both the static and the dynamic components of the AHS. In other words, we considered where the best place (in which layer) would be to store certain information and how. For example, concept relations should be in the domain layer, as in previous adaptive hypermedia systems, just as long as they express a real conceptual link between two or more concepts (content-driven relations). However, relations like the *prerequisite relationship type* described previously are related to the pedagogical intentions of the author (teacher) and not to the actual content of the concept. Therefore, such a relation should be in a different layer (in LAOS, on the goal and constraints layer).

Another example of separation of concerns is the presentation. AHA! and AHAM as well mix presentation related issues, such as link colors, for instance, with concept content related issues, such as concept relations. The presentation-related issues are very often client dependent, and should be kept separately, so they can be dealt with directly. Therefore the need of the presentation layer (model). More details on the full LAOS model can be found in [17].

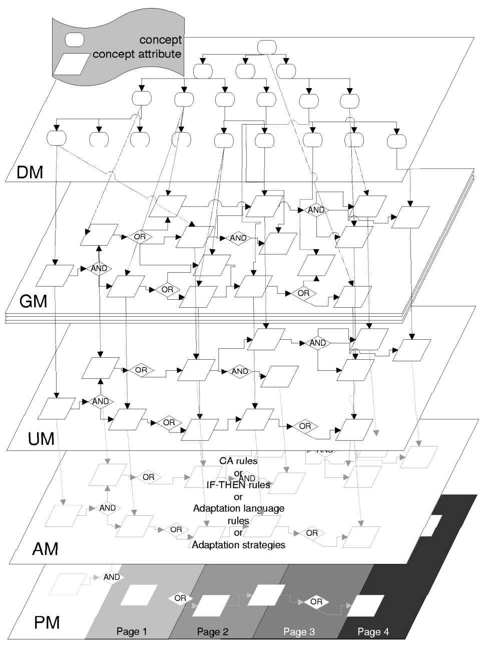


Fig. 4. LAOS: the five layer authoring model of adaptive hypermedia.

The authoring steps of adaptive hypermedia with the LAOS model are presented in [17], where also a more detailed description about the components and functionality of each layer and the respective operators that implement this functionality can be found.

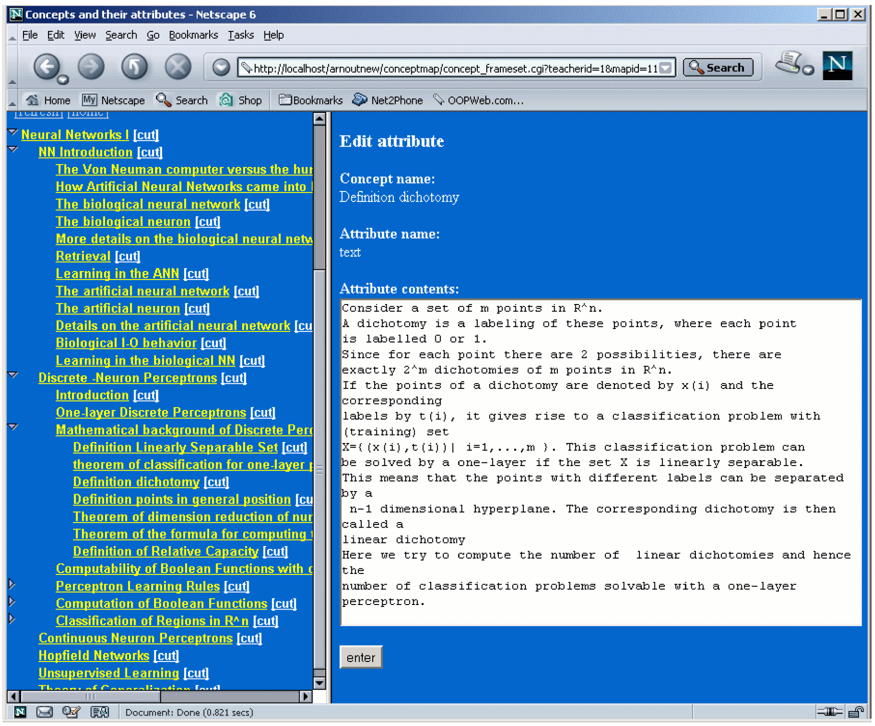
An important feature of LAOS is that it is developed to work with a new adaptation model, LAG [15], that allows views over the different levels of adaptation, responding to the problem of authoring complexity from the capital adaptive hypermedia feature: the adaptation feature.

4.4 MOT: My Online Teacher

LAOS and LAG are more general models, that are applicable for general adaptive hypermedia analysis and authoring. MOT [16] however was designed specifically for distance learning. Its predecessor is MyET (My English Teacher) [18] developed at the University of Electro-Communications, Tokyo, Japan. MOT is gradually implementing the ideas and definitions introduced by LAOS and LAG.

MOT ("My Online Teacher") is a hypermedia tool, developed and extended at the Eindhoven University of Technology, that can be used for authoring adaptive hypermedia courses. With this tool, the subject-matter of the course to be designed can be modeled by means of concept maps (Fig. 5). Based on these concept maps lessons can be constructed. Concept maps and lessons form the two levels of pre-adaptive content, and they are stored in a database. This structure lays the basis for various types of adaptation, as it uses both the expressivity of metadata annotation and the flexibility of the database structure (on which different queries can be performed). MOT is being extended so it can interface with AHA!. This will then allow us to implement LAOS and LAG flexibility in the AHA! system.

MOT can demonstrate the ideas of separating domain model and goal and constraints model, as well as some of the ideas on automatic authoring (specifically, automatic generation of links within the concept domain model and automatic generation of an instance of the goal and constraints model from an instance of the domain model).


Concepts and their attributes - Netscape 6
File Edit View Search Go Bookmarks Tasks Help
http://localhost/arnoutnew/conceptmap/concept_frameset.cgi?teacherid=1&mapid=11
Search
Home
My Netscape
Search
Shop
Bookmarks
Net2Phone
OOPWeb.com...
Document: Done (0.821 secs)

Neural Networks I [cut]
NN Introduction [cut]
The Von Neuman computer versus the hu
How Artificial Neural Networks came into
The biological neural network [cut]
The biological neuron [cut]
More details on the biological neural netw
Retrieval [cut]
Learning in the ANN [cut]
The artificial neural network [cut]
The artificial neuron [cut]
Details on the artificial neural network [cu
Biological I-O behavior [cut]
Learning in the biological NN [cut]
Discrete -Neuron Perceptrons [cut]
Introduction [cut]
One-layer Discrete Perceptrons [cut]
Mathematical background of Discrete Per
Definition Linearly Separable Set [cut]
theorem of classification for one-layer p
Definition dichotomy [cut]
Definition points in general position [cu
Theorem of dimension reduction of nu
Theorem of the formula for computing t
Definition of Relative Capacity [cut]
Computability of Boolean Functions with c
Perceptron Learning Rules [cut]
Computation of Boolean Functions [cut]
Classification of Regions in R^n [cut]
Continuous Neuron Perceptrons [cut]
Hopfield Networks [cut]
Unsupervised Learning [cut]

Edit attribute

Concept name:
Definition dichotomy

Attribute name:
text

Attribute contents:

```
Consider a set of m points in R^n.
A dichotomy is a labeling of these points, where each point
is labelled 0 or 1.
Since for each point there are 2 possibilities, there are
exactly 2^m dichotomies of m points in R^n.
If the points of a dichotomy are denoted by x(i) and the
corresponding
labels by t(i), it gives rise to a classification problem with
(training) set
X={(x(i),t(i))| i=1,...,m }. This classification problem can
be solved by a one-layer if the set X is linearly separable.
This means that the points with different labels can be separated
by a
 n-1 dimensional hyperplane. The corresponding dichotomy is then
called a
linear dichotomy
Here we try to compute the number of  linear dichotomies and hence
the
number of classification problems solvable with a one-layer
perceptron.
```

enter

Fig. 5. MOT: My Online Teacher.

5 Concluding Remarks and Outlook into the Future

Adaptive educational hypermedia applications have the potential to give on-line course material a "personal touch". Several successful applications and application frameworks exist, but mass employment of adaptive hypermedia in education is still lacking. We believe that authoring difficulties are the main problem that remains. We have shown new approaches to make authoring easier, and to allow adaptation to other aspects besides the systems's idea of the user's knowledge of concepts.

A second issue that needs to be tackled in order to make adaptive educational hypermedia popular is the current "closed" nature of the applications that exist. Many infrastructures exist to integrate learning material from different sources into large information sources. Yet these infrastructures lack the ability to handle adaptive sources of learning material. The definition of standards in the area of adaptive educational hypermedia, in collaboration with the ongoing learning technology standards developments, is needed to enable the exchange of course material and user model information between adaptive applications.

Acknowledgements

Work on AHA! 1.0 was supported by a grant of the Eindhoven University of Technology. The development of AHA! 2.0 and 3.0 is supported by a grant of the NLnet Foundation (see http://www.nlnet.nl/).

Work on LAOS, LAG and MOT is linked to the European Community Socrates Minerva project "Adaptivity and adaptability in ODL based on ICT" (project reference number 101144-CP-1-2002-NL-MINERVA-MPP). Work on AIMS and ATO is supported by the University of Twente and MizLab, at the Osaka University.

References

1. Aroyo, L. *Task-based approach to information handling support with web-based education.* PhD thesis, University of Twente, 2001.
2. Aroyo, L., and Dicheva, D. Aims: Learning and teaching support for www-based education. *Int. Journal for Continuing Engineering Education and Life-long Learning 11(1/2)* (2001), 152–164.
3. Aroyo, L., Dicheva, D., and Cristea, A. Ontological support for web courseware authoring. *Proceedings of the 6th International Conference on Intelligent Tutoring Systems LNCS 2363* (2002), 270–280.
4. Aroyo, L., Dicheva, D., and Velev, I. A concept-based approach to support learning in a web-based course environment. *AI in Education: Frontiers of AI and Applications 68* (2001), 1–10.
5. Aroyo, L., and Mizoguchi, R. Authoring support framework for intelligent educational systems. *Proceedings of International Conference on Artificial Intelligence in Education(AIED'03)* (2003).

6. Bra, P. D., Aerts, A., Smits, D., and Stash, N. Aha! version 2.0, ore adaptation flexibility for author. *Proceedings of the AACE ELearn Conference* (2002), 240–246.
7. Bra, P. D., and Calvi, L. Creating adaptive hyperdocuments for and on the web. *Proceedings of the AACE WebNet Conference* (1997), 149–154.
8. Bra, P. D., and Calvi, L. Aha! an open adaptive hypermedia architecture. *The New Review of Hypermedia and Multimedia 4* (1998), 115–139.
9. Bra, P. D., Houben, G.-J., and Wu, H. Aham: A dexter-based reference model for adaptive hypermedia. *Proceedings of the ACM Conference on Hypertext and Hypermedia* (1999), 147–156.
10. Brusilovsky, P. Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction 6* (1996), 87–129.
11. Brusilovsky, P. Adaptive hypermedia. *User Modeling and User-Adapted Interaction 11* (2001), 87–110.
12. Brusilovsky, P., Eklund, J., and Schwarz, E. Web-based education for all: A tool for developing adaptive courseware. *Computer Networks and ISDN Systems (Proceedings of the International World Wide Web Conference) 30*, 1-7 (1998), 291–300.
13. Calvi, L., and Bra, P. D. Using dynamic hypertext to create multi-purpose textbooks. *Proceedings of the AACE ED-MEDIA Conference* (1997), 130–135.
14. Cini, A., and de Lima, J. V. Adaptivity conditions evaluation for the user of hypermedia presentations built with aha! *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems* (2002), 497–500.
15. Cristea, A., and Calvi, L. The three layers of adaptation granularity. *Proceedings of the ninth International Conference on User Modelling (UM'03),Pittsburg, US, Springer* (2003), to appear.
16. Cristea, A., and Mooij, A. D. Adaptive course authoring: Mot, my online teacher. *Proceedings of ICT-2003, IEEE LTTF International Conference on Telecommunications,"Telecommunications + Education" Workshop (Feb 23 - March 1)* (2003), CD–ROM.
17. Cristea, A., and Mooij, A. D. Laos : Layered www ahs authoring model and their corresponding algebraic operators. *Proceedings of the Twelfth International World Wide Web Conference (WWW'03),Budapest,Hungary W3C, ACM* (2003), to appear.
18. Cristea, A., Okamoto, T., and Belkada, S. Concept mapping for subject linking in a www authoring tool: Myenglishteacher: Teachers' site. *Proceedings of ANNIE'00, Smart Engineering System Design: Neural Networks, Fuzzy Logic, Evolutionary Programming and Complex Systems, Eds. Drs. Dagli et al, ASME Press* (2000).
19. da Silva, D. P. Comcepts and documents for adaptive hypermedia: A model and a prototype. *Proceedings of the Workshop on Adaptive Hypertext and Hypermedia* (1998), 33–40.
20. Halasz, F., and Schwartz, M. The dexter hypertext reference model. *Communications of the ACM, Vol. 37, nr. 2* (1994), 30–39.
21. Kay, J., and Kummerfeld, B. An individualised course for the c programming language. *(On-line) Proceedings of the Second International WWW Conference* (1994).

22. Kitamura, Y., Sano, T., and Mizoguchi, R. Functional understanding based on an ontology of functional concepts. *Proceedings of PRICAI'00 Conference* (2000), 723–733.
23. Koch, N. *Software Engineering for Adaptive Hypermedia Systems: Reference Model, Modeling Techniques and Development Process.* PhD thesis, Ludwig-Maximilians-Universität München, 2001.
24. Koch, N., and Wirsing, M. The munich reference model for adaptive hypermedia applications. *Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web-based Systems* (2002), 213–222.
25. Mitrovic, A. Experiences in implementing constraint-based modeling in sql-tutor. *Proceedings of the 4th International Conference on Intelligent Tutoring Systems* (1998), 414–423.
26. Mitrovic, A. Using evaluation to shape its design: Results and experiences with sql-tutor. *User Modeling and User-Adapted Interaction 12*, 2-3 (2002), 243–279.
27. Mizoguchi, R., Sinitsa, K., and Ikeda, M. Task ontology design for intelligent educational/training systems. *Proceedings of the 3rd International Conference on Intelligent Tutoring Systems* (1996).
28. Romero, C., Ventura, S., Bra, P. D., and de Castro, C. Discovering prediction rules in aha! courses. *Proceedings of the International Conference on User Modeling* (2003), 25–34.
29. Weber, G., and Brusilovsky, P. Elm-art: An adaptive versatile system for web-based instruction. *International Journal of Artificial Intelligence in Education 12* (2001), 351–384.
30. Weber, G., and Specht, M. User modeling and adaptive navigation support in www-based tutoring systems. *Proceedings of the International Conference on User Modeling* (1997), 289–300.
31. Wu, H. *A Reference Architecture for Adaptive Hypermedia Applications.* PhD thesis, Eindhoven University of Technology, 2002.