Supporting User Adaptation in Adaptive Hypermedia Applications

Hongjing Wu, Geert-Jan Houben, Paul De Bra

Department of Computing Science
Eindhoven University of Technology
PO Box 513, 5600 MB Eindhoven
the Netherlands

phone: +31 40 2472733
fax: +31 40 2463992
email: {hongjing,houben,debra}@win.tue.nl

Abstract: A hypermedia application offers its users a lot of freedom to navigate through a large hyperspace. The rich link structure of the hypermedia application can not only cause users to get lost in the hyperspace, but can also lead to comprehension problems because different users may be interested in different pieces of information or a different level of detail or difficulty. Adaptive hypermedia systems (or AHS for short) aim at overcoming these problems by providing adaptive navigation support and adaptive content. The adaptation is based on a *user model* that represents relevant aspects about the user.

At the Eindhoven University of Technology we developed an AHS, named AHA [DC98]. To describe its functionality and that of future adaptive systems we also developed a reference model for the architecture of adaptive hypermedia applications, named AHAM (for Adaptive Hypermedia Application Model) [DHW99]. In AHAM knowledge is represented through hierarchies of large composite abstract concepts as well as small atomic ones. AHAM also divides the different aspects of an AHS into a *domain model* (DM), a *user model* (UM) and an *adaptation model* (AM). This division provides a clear separation of concerns when developing an adaptive hypermedia application.

In this paper, we concentrate on the *user modeling* aspects of AHAM, but also describe how they relate to the *domain model* and the *adaptation model*. Also, we provide a separation between the adaptation rules an author or system designer writes (as part of the *adaptation model*) and the system's task of executing these rules in the right order. This distinction leads to a simplification of the author's or system designer's task to write adaptation rules. We illustrate authoring and adaptation in by some examples in the AHS AHA.

Keywords: adaptive hypermedia, user modeling, adaptive presentation, adaptive navigation, hypermedia reference model

1. Introduction

Hypermedia systems, and Web-based systems in particular, are becoming increasingly popular as tools for user-driven access to information. Hypermedia applications typically offer users a lot of freedom to navigate through a large hyperspace. Unfortunately, this rich link structure of the hypermedia application causes some serious usability problems:

- A typical hypermedia system presents the same links on a page, regardless the path a user followed to reach this page. When providing navigational help, e.g. through a map (or some fish-eye view) the system does not know which part of the link structure is most important for the user. The map cannot be simplified by filtering (or graying) out links that are less relevant for the user. Not having personalized maps is a typical *navigation problem* of hypermedia applications.
- Navigation in ways the author did not anticipate also causes *comprehension problems*: for every page the author makes an assumption about the foreknowledge the user has when accessing that page. However, there are too many ways to reach a page to make it possible for an author to an-

ticipate all possible variations in foreknowledge when a user visits that page. A page is always presented in the same way. This often results in users visiting pages containing a lot of redundant information and pages that they cannot fully understand because they lack some expected foreknowledge.

Adaptive hypermedia systems (or AHS for short) aim at overcoming these problems by providing adaptive navigation support and adaptive content. Adaptive hypermedia is a recent area of research on the crossroad of hypermedia and the area of user-adaptive systems. The goal of this research is to improve the usability of hypermedia systems by making them personalized. The personalization or adaptation is based on a *user model* that represents relevant aspects about the user. The system gathers information about the user by observing the use of the application, and in particular by observing the *browsing* behavior of the user.

Many adaptive hypermedia systems exist to date. The majority of them are used in educational applications, but some are used for on-line information systems, on-line help systems, information retrieval systems, etc. An overview of systems, methods and techniques for adaptive hypermedia can be found in [B96]. At the Eindhoven University of Technology we developed an AHS system [DC98] out of Web-based courseware for an introductory course on hypermedia. In this system, called AHA, knowledge is represented with the same granularity as content: at the page level. In earlier versions of AHA, the user's knowledge about a given concept was a binary value: *known* or *not known*. The current version supports a more sophisticated representation in the sense that the knowledge level is represented by a *percentage*: reading a page can lead to an increase (or decrease) of the percentage. As part of the redesign process for AHA we have developed a reference model for the architecture of adaptive hypermedia applications: AHAM (for **A**daptive **H**ypermedia **A**pplication **M**odel) [DHW99], which is an extension of the Dexter hypermedia reference model [HS90, HS94]. AHAM acknowledges that doing "useful" and "usable" adaptation in a given application depends on three factors:

- The application must be based on a *domain model*, describing how the information content of the application (or "hyperdocument") is structured. This model must indicate what the relationship is between the high (and low) level concepts the application deals with, and it must indicate how concepts are tied to information fragments and pages.
- The system must construct and maintain a fine-grained *user model* that represents a user's preferences, knowledge, goals, navigation history and possibly other relevant aspects. The system can learn more about the user by observing the user's behavior. The user's knowledge is represented using the concepts from the domain model.
- The system must be able to adapt the presentation (of both content and link structure) to the reading and navigation style the user prefers and to the user's knowledge level. In order to do so the author must provide an *adaptation model* consisting of *adaptation rules*, for instance indicating how relations between concepts influence whether it will be desirable to guide the user towards or away from pages about certain concepts. Most AHS will offer a default adaptation model, relieving the author from explicitly writing these rules. In the original definition of AHAM [DHW99] we used the terms *teaching model* (TM) and *pedagogical rules*. These terms stem from the primary application of AHS's which is in education.

The key elements in AHAM are thus the *domain model* (DM), *user model* (UM) and *adaptation model* (AM). This division of adaptive hypermedia applications provides a clear separation of concerns when developing an adaptive hypermedia application. The main shortcoming in many current AHS is that these three factors or components are not clearly separated:

- The relationship between pages and concepts is sometimes too vague (e.g. in [PDS98]). When an author decides that two pages each represent 30% of the same concept, there is no way of inferring whether together they represent 30%, 60% of the concept or any value in between. On the other hand systems like AHA [DC98] the relation between pages and concepts is strictly one-to-one, which leads to a very fragmented user model without high-level concepts.
- The adaptation rules can often not be defined at the conceptual level but only at the page level. In AHA [DC98], ELM-ART [BSW96a] and Interbook [BSW96b] for instance the destination of a link is (in almost all cases) a fixed page, described through a plain HTML anchor tag. (The "teach me" button in Interbook is an exception.)

- There may be a mismatch between the high level of detail in the user model and the low reliability of the information on which an AHS must update that user model. The basic information available to most AHS is the time at which a user requests a page (through a WWW-browser). Many educational AHS compensate for the unreliable event information by offering (multiple-choice) tests. A few systems, including AHA [DC98], capture reading time by logging both requests for pages and the time at which the user leaves a page (even when jumping to a different Web-site).

In this paper we focus on the user modeling aspects of AHAM and the use of adaptation rules to generate adaptive presentations and to update the user model. We extend the results of [WHD99b] by separating adaptation rules from the specification of the execution of these rules.

This paper is organized as follows. In Section 2 we describe the AHAM reference model for adaptive hypermedia applications. In Section 3 we elaborate on user modeling and on the use of adaptation rules in AHAM, that is how to construct the user model, update the user model by observing the user's behavior, and how to make content adaptation and link adaptation depending on the user model. In Section 4 we use AHAM to describe the user modeling and adaptation features of the AHA system, before we conclude in Section 5.

2. AHAM, a Dexter-based Reference Model

In hypermedia applications the emphasis is always on the information nodes and on the link structure connecting these nodes. The Dexter model captures this in what it calls the Storage Layer. It represents a *domain model* DM, i.e. the author's view on the application domain expressed in terms of concepts.

In adaptive hypermedia applications the central role of DM is shared with a *user model* UM. UM represents the relationship between the user and the domain model by keeping track of how much the user knows about each of the concepts in the application domain.

In order to perform adaptation based on DM and UM an author needs to specify how the user's knowledge influences the presentation of the information from DM. In AHAM this is done by means of a *teaching model* TM consisting of *pedagogical rules*. In this paper we use the terms *adaptation model* (AM) and *adaptation rules* to avoid the association with educational applications. An adaptive engine uses these rules to manipulate link anchors (from the Dexter model's *anchoring*) and to generate what the Dexter model calls the *presentation specifications*. Like the Dexter model, AHAM focuses on the Storage Layer, the anchoring and the presentation specifications. Figure 1 shows the structure of adaptive hypermedia applications in the AHAM model.

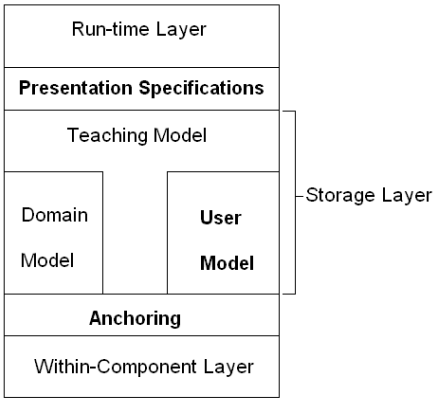

Figure 1: global structure of adaptive hypermedia applications.

In this section we present the elements of AHAM that we will use in Section 3 to illustrate the user modeling and adaptation.

2.1 The domain model

A *component* is an abstract notion in an AHS. It is a pair (uid, cinfo) where uid is a globally unique (object) identifier for the component and cinfo represents the component's information. A *component's information* consists of:

- A set of attribute-value pairs;
- A sequence of anchors (for attaching links);
- A presentation specification.

We distinguish two "kinds" of components: *concepts* and *concept relationships*. A *concept* is a component representing an abstract information item from the application domain. It can be either an *atomic concept* or a *composite concept*. An *atomic concept* corresponds to a fragment of information. It is primitive in the model (and can thus not be adapted). Its attribute and anchor values belong to the "Within-component layer" and are thus implementation dependent and not described in the model. A *composite component* has two "special" attributes:

- A sequence of children (concepts);
- A constructor function (to denote how the children belong together).

The children of a composite concept are all atomic concepts (then we call it a *page* or in typical hypertext terms a *node*) or all composite concepts. The composite concept component hierarchy must be a DAG (directed acyclic graph). Also, every atomic concept must be included in some composite concept. Figure 2 illustrates a part of a concept hierarchy.

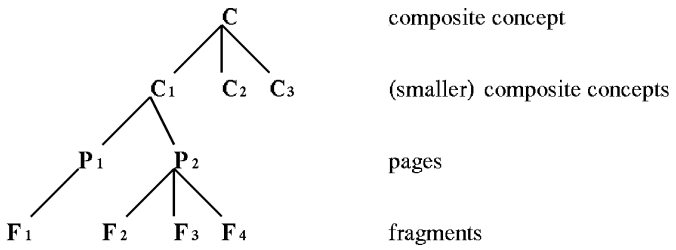

Figure 2: Example concept hierarchy.

An *anchor* is a pair (aid, avalue), where aid is a unique (object) identifier for the anchor within the scope of its component and avalue is an arbitrary value that specifies some location, region, item or substructure within a concept component. Anchor values of atomic concepts belong to the (implementation dependent) Within-Component layer. Anchor values of composite concepts are identifiers of concepts that belong to that composite.

A *specifier* is a tuple (uid, aid, dir, pres), where uid is the identifier of a concept, aid is the identifier of an anchor, dir is a direction (FROM, TO, BIDIRECT, or NONE), and pres is a presentation specification.

A *concept relationship* is a component, with two additional attributes:

- A sequence of specifiers;
- A concept relationship type.

The most common type of concept relationship is the type **link**. This corresponds to the link components in the Dexter model, or links in most hypermedia systems. (Links typically have at least one FROM element and one TO or BIDIRECT element.) In AHAM we consider other types of relationships as well, which play a role in the adaptation. A common type of concept relationship is **prerequisite**. When a concept C_1 is a prerequisite for C_2 it means that the user should read C_1 before C_2. It does not mean that there must be a link from C_1 to C_2. It only means that the system somehow takes into account that reading about C_2 is not desired before some (enough) knowledge about C_1 has been acquired. Every prerequisite must have at least one FROM element and one TO element. Figure 3 shows a small set of (only binary) relationships, both prerequisites and links.

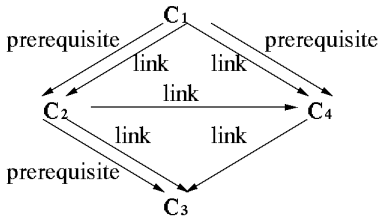

Figure 3: Example concept relationship structure.

The atomic concepts, composite concepts and concept relationships together form the *domain model* DM of an adaptive hypermedia application.

2.2 The user model

An AHS associates a number of *user model attributes* with each concept component of DM. For each user the AHS maintains a *table-like structure*, in which for each concept the attribute values for that concept are stored. Section 3 describes the user model in detail. For now it suffices to know that because of the relationships between *abstract* concepts and *concrete* content elements like fragments and pages a user model may contain other attributes than simply a *knowledge level*. For instance, the user model may also store information about what a user has actually read about a concept or whether a concept is considered relevant for the user.

Since the user model consists of "named entities" for which we store a number of attribute/value pairs, there is no reason to limit these "entities" to *concepts* about which the *knowledge level* is stored and updated. Concepts can be used (some might say abused) to represent other user features, such as preferences, goals, background and hyperspace experience. For the AHS (or the AHAM model) the actual meaning of concepts is irrelevant.

2.3 The adaptation (teaching) model

The adaptation of the information content of a hyperdocument and of the link structure is based on a set of *rules*. These rules form the connection between DM, UM and the presentation (specification) to be generated [WHD99a].

We partition the rules into four groups according to the adaptation "steps" to which they belong. These steps are IU, UU-Pre, GA, and UU-Post. An algorithm applies rules in each group. IU is to initialize the user model, under control of *Initialize-UM*; UU-Pre is to update UM before generating the page, under control of *Update-UM-pre*; GA is to generate adaptation, under control of *Adaptation*; UU-Post is to update UM after generating the page, under control of *Update-UM-post*. The four algorithms control how the rules in each group work together. By this we mean that an algorithm will trigger applicable rules (in some order) until no more rules can be applied or until the application of rules would no longer incur any change to UM.

A *generic adaptation rule* is a rule in which (bound) variables are used that represent concepts and concept relationships. A *specific adaptation rule* uses concrete concepts from DM instead of variables. Other than that both types of rules look the same. The syntax of the permissible rules depends on the AHS. In Section 3 we give examples of adaptation rules, using an arbitrarily chosen syntax. In Section 4 we give examples of adaptation rules as they are implemented in the AHA system [DC98]. Generic adaptation rules are often system-defined, meaning that an author need not specify them. Such a rule may for instance define how the knowledge level of an arbitrary concept C_1 influences the relevance of other concepts for which C_1 is a prerequisite. Author-defined rules always take precedence over (conflicting) system-defined rules. (Some AHS do not provide the possibility for authors to define their own generic adaptation rules.) Specific rules always take precedence over generic rules.

While specific rules are typically used to create exceptions to generic rules they can also be used to perform some ad-hoc adaptation based on concepts for which DM does not provide a relationship. Specific

adaptation rules must always be defined by the author.

The *adaptation model* AM of an AHS is the set of (generic and specific) adaptation rules.

An AHS does not only have a domain model, user model en adaptation model, but also an *adaptive engine*, which is a software environment that performs the following functions:

- It offers generic page selectors and constructors. For each composite concept the constructor is used to determine which page to display when the user follows a link to that composite concept. For each page the constructor is used for building the adaptive presentation of that page.
- It optionally offers a (very simple programming) language for describing new page selectors and -constructors. Generic and specific adaptation rules (from UU-pre and GA) are used during page selection and construction.
- It performs adaptation by executing the page selectors and constructors. This means selecting a page, selecting fragments, sorting them, maybe presenting them in a specific way, etc. It also means performing adaptation to links by manipulating link anchors depending on the state of the link (like enabled, disabled, hidden, etc.).
- It updates the user model (instance) each time the user visits a page. It does so by triggering the necessary adaptation rules in UU-post. The engine will thus set the knowledge value for each atomic concept of displayed fragments of the page to a value that depends on a configurable amount (this could be 1 by default but possibly overridden by the author). It determines the influence on the knowledge value for page- and composite concepts. It also maintains other attribute values for each concept.

The adaptive engine thus provides the implementation dependent aspects while DM, UM and AM describe the information and adaptation at the conceptual, implementation independent level. An *adaptive hypermedia application* is a 4-tuple (DM, UM, AM, AE), where DM is a domain model. UM is a user model, AM is a adaptation model, and AE is an adaptive engine.

3. User Modeling and Adaptation in AHAM

According to AHAM the AHS maintains a fine-grained user model that represents the state of the user's features not only at the page level but also at the abstract conceptual level. It offers the ability to consider navigation history and other relevant user aspects as part of the user model UM. The maintenance of the relevant user aspects in UM is achieved by the application of the adaptation rules that are part of the adaptation model AM.

3.1 Representation of user features using (attribute/value) pairs

By definition adaptive hypermedia applications reflect some features of the user in the user model. This model is used to express various visible aspects of the system that depend on the user and that are visible to that user. Brusilovsky [B96] states which aspects of the user can be taken into account when providing adaptation. Generally, there are five user features that are used by existing AHS:

- knowledge
- user goals
- background
- hyperspace experience
- preferences

Almost every adaptive presentation technique relies on the user's knowledge as a source of adaptation. The system has to recognize the changes in the user's knowledge state and update its user model accordingly. Often the user's knowledge is represented by an overlay model. This overlay model is based on a conceptual structure of the subject domain. Sometimes a simpler stereotype user model is used to represent the user's knowledge: this means that the user is classified according to some stereotype. As many adaptation techniques require a rather fine-grained approach, stereotype models are often too simple to provide adequate personalization and adaptation. Overlay models on the other hand are generally hard to initialize. Acceptable results are often achieved by combining stereotype and overlay modeling: stereo-

type modeling is used in the beginning to classify a new user and to set initial values for the overlay model; later a more fine-grained overlay model is used. Using the AHAM definition for user model, it is fairly straightforward how a user's knowledge state can be represented by associating a *knowledge value* attribute to each concept.

Apart from the concept's identifier (which may be just a name) a typical AHS will store not only a *knowledge value* for each concept, but also a *read* value which indicates whether (and how much) information about the concept has been read by the user, and possibly some other attribute values as well. While the model uses a table representation, implementations of AHS may use different data structures. For instance, a logfile can be used for the *read* attribute. Table 1 illustrates the (conceptual) structure of a user model for a course on hypermedia: the concepts Xanadu and KMS were at least partially learnt. The concept WWW, consisting of two sub-parts, is partially learnt because WWW-page1 has been read but WWW-page2 has not been read. One can see that WWW must be a composite concept that is not a page, because it is already partially learnt while it has not been read at all.

concept name (uid)	Knowledge value	read	...
Xanadu	well learned	true	...
KMS	learned	true	...
WWW-page1	well learned	true	...
WWW-page2	not known	false	...
WWW	learned	false	...
...	...	...	...

Table 1: Example user model (instance).

The second kind of user feature is the user's goal. The user's goal or task is a feature that is related with the context of the user's working activities rather than with the user as an individual. The user's goal is the most volatile of all user features. It can be considered as a very important user feature for AHS. One representation of possible user goals uses a hierarchy (a tree) of tasks. Another representation of the user's current goal uses a set of pairs (Goal,Value), where Value is the probability that Goal is the current goal of the user. The latter representation perfectly matches the way in which AHAM models the user's state.

Two features of the user that are similar to the user's knowledge of the subject but that functionally differ from it, are the user's background and the user's experience in the given hyperspace. By background we mean all the information related to the user's previous experience *outside* the subject of the hypermedia system. By user's experience in the given hyperspace we mean how familiar is the user with the structure of the hyperspace and how easy can the user navigate in it. Again, these features can be modeled in AHAM using concepts' attribute/value pairs.

For different possible reasons the user can prefer some nodes and links over others or some parts of a page over others. This is used most heavily in information retrieval hypermedia applications. In fact in most adaptive information retrieval hypermedia applications preferences are the only information that is stored about the user. User preferences differ from other user model components, since in most cases they cannot be deduced by the system. The user has to inform the system directly or indirectly about the preferences. AHAM's attribute/value pairs can again be used to model the user's preferences.

From the above descriptions we can conclude that although a user model needs to represent (five) very different aspects of a user, all of these kinds of aspects can be implemented as sets of *concepts with associated attribute/value pairs*. For presentation purposes it is not necessary to treat these different kinds of aspects in a different way, but for implementation purposes it is often needed to treat these in different ways in adaptive hypermedia applications.

The knowledge value of a concept can be a Boolean, discrete or continuous value depending on the choice of the author (or the properties of the AHS). By using a Boolean value, the knowledge about the concept can be either *known* or *unknown*. By using a discrete value the knowledge about the concept can be one of a small set of values, like *unknown*, *learnt*, *well learnt* or *well known*. By using continuous values from the range of [0..1], the value can more precisely describe the user's knowledge, and even describe the loss or decay of knowledge over time. In conclusion, AHAM's user model UM has enough expressive power to model all user features that current AHS take into account.

3.2 Changes in user features

In the previous subsection we discussed features that describe the user's state in the browsing process. Usually in adaptive hypermedia applications (as opposed to adaptable hypermedia applications, see [DHW99]), only the browsing behavior is observed in order to influence the adaptation. Basically, there are five ways in which the user features can change in an adaptive hypermedia application:

1. the user clicks on an anchor (and follows a link);
2. the user performs a test (explicitly);
3. information (about the user) is imported from an external testing system;
4. a user preference is (explicitly) set or declared by the user (initially);
5. a user preference is (automatically) inferred from the user's behavior.

Besides observing the browser behavior, the application can change the user features based on information that is explicitly imported from its environment or that is explicitly declared or implicitly inferred about the user's preferences.

These five different kinds of changes lead to five kinds of "rules" how to maintain the user features. The system can be made more *author centered* by including rules of types 2 and 3 (besides rules of type 1), while the application can become more *user centered* by including rules of types 4 and 5. It is also possible to choose a combination that suits the application.

3.3 Adaptation based on the user model

By maintaining the user model the system can infer how relevant aspects of the user change while the user is using the application and thus is using the adaptation. The adaptive engine realizes adaptive presentation and adaptive navigation (or link adaptation) according to the (adaptation) rules that are system-defined or written by the author and that depend on the user model. Below we give a number of examples to show how adaptation rules are used to do adaptation. The syntax used for the rules is arbitrary and only exemplary. AHAM does not prescribe any specific syntax. Normally every AHS will provide its own syntax for defining adaptation rules.

Example 1 *For atomic concepts (fragments) let us assume that the presentation specification is a two-valued (almost Boolean) field, which is either "show" or "hide". When a page is being accessed, the following rule sets the visibility for fragments that belong to a "page" concept, depending on their "relevance" attribute-value.*

< access(C) and F IN C.children and F.relevance = true => F.pres := show >

Here we simplified things, by assuming that we can treat C.children as if it were a set, whereas it really is a sequence. It is common to execute rules for generating presentation specifications before generate the page, so it is in GA.

Example 2 *The following rules set the presentation specification for a specifier that denotes a link (source) anchor depending on whether the destination of the link is considered relevant and whether the destination has been read before. For simplicity we consider a link with just one source and one destination.*

< CR.type = link and CR.cinfo.dir[1] = FROM and CR.cinfo.dir[2] = TO and CR.ss[2].uid.relevant = true and CR.ss[2].uid.read = false => CR.ss[1].pres = GOOD >

< CR.type = link and CR.cinfo.dir[1] = FROM and CR.cinfo.dir[2] = TO and CR.ss[2].uid.relevant = true and CR.ss[2].uid.read = true => CR.ss[1].pres = NEUTRAL >

< CR.type = link and CR.cinfo.dir[1] = FROM and CR.cinfo.dir[2] = TO and CR.ss[2].uid.relevant = false => CR.ss[1].pres = BAD>

These rules say that links to previously unread but "relevant" pages are "GOOD". Links to previously read and "relevant" pages are "NEUTRAL" and links to pages that are not "relevant" are "BAD". In the AHA system [DC98] this results in the link anchors being colored blue, purple or black respectively. In ELM-ART [BSW96a] and Interbook [BSW96b] the links would be annotated with a green, yellow or red ball. We can consider the actual presentation (the coloring of the anchors) as belonging to the Run-time Layer and thus outside the scope of AHAM. However, should we opt to include the color preferences for GOOD, NEUTRAL and BAD links in the user model then the translation of the presentation specification to the color could still be described using a adaptation rule. These rules are in GA also.

3.4 Maintenance of user model

To record the reading history of the user and the evolution of the user's knowledge, the system updates the user model based on the observation of the user's browsing process. The rules that the author has defined in AM describe how to keep track of the evolution of the user's knowledge. For the application of adaptation rules we assume that the *FollowLink* operation from the Dexter (and thus AHAM) model's Run-time Layer results in a call to a *resolver function* for a given specifier. In AHAM the resolver translates the given specifier to the uid of a composite concept component that corresponds to a page, or to a set of such uid's. Which page exactly is selected depends on DM and UM. For the selected page an *accessor function* is called, according to the Dexter model, which returns the (page) concept component that corresponds to the resolved uid. Then the rules for presentation are executed, as shown in Subsection 3.3.

Example 3 *The following rule expresses that when a page is accessed the "read" user-model attribute for the corresponding concept is set to true:*

< access(C) => C.read := true>

This rule is in *UU-post*. It is the *Update-UM-post* that will trigger other rules that have *read* on their left-hand side in the same group.

Example 4 *The following rule expresses that when a page is "relevant" and it is accessed, the knowledge value of the corresponding concept becomes "well-learnt". This is somewhat like the behavior of Interbook [BSW96b].*

< access(C) and C.relevant = true => C.knowledge := well-learnt>

In Interbook, as well as in AHA [DC98], knowledge is actually updated before the page is generated. These rules thus are in *UU-pre*. At the end of Section 4 we shall describe why this option is chosen, and which problems it creates. In general one wishes to have the option to base some adaptation on the knowledge state *before* accessing a page and some adaptation on the knowledge state *after* reading the page.

Example 5 *The following rule expresses that after a user has taken a test about a concept C, his knowledge about concept C is changed (a rule of "type 2" from Subsection 3.2). Here, an action "test" is used that represents that a test has been taken. It is in UU-pre*

< test(C) and C.test > 60 => C.knowledge := known>

4. User Modeling and Adaptation in the AHA system

AHA [DC98] is a simple adaptive hypermedia system. We describe the properties of the version that is currently being used for two on-line courses and one on-line information kiosk, plus some features of the next version that is currently being developed.

- In AHA the *domain model* consists of three types of concepts: *abstract concepts*, *fragments* and *pages*. Concepts are loosely associated with (HTML) pages, not with fragments.
- The *user model* consists of:
  - Color preferences for link anchors which the user can customize. (These preferences result in "non-relevant" link anchors to be hidden if their color is set to black, or visibly "annotated" if their color is set to a non-black color, different from that of "relevant" link anchors.)
  - For each abstract concept, a *knowledge* attribute with percentage values. (100 means the concept is fully known). For pages and fragments there is no *knowledge* attribute value.
  - For each page, a Boolean *read* attribute. (*True* means the page was read, *false* means it was not read.) AHA actually logs access and reading times, but they cannot be used in a more sophisticated way in the current version. For abstract concepts and fragments there is no *read* attribute value.
- AHA comes with an *adaptation model* containing system-defined generic adaptation rules. It offers a simple language for creating author-defined specific adaptation rules (but no author-defined generic rules).

The *domain model* can only contain concept relationships of the types that are shown below. An author cannot define new types. The influence of these relationships on the adaptation and the user model updates is defined by system-defined generic adaptation rules. In AHA all rules are executed before generate the page and are triggered directly by a page access, thus eliminating the need for propagation.

- When a page is accessed, its *read* attribute in the user model is updated as follows (it is in *UU-pre*):

  < access(P) => P.read := true >

- The relationship type *generates* links a page to an abstract concept. A *generates* relationship between P and C means that reading page P generates knowledge about C (it is in *UU-pre*):

  < access(P) => C.knowledge := 100>

  This "generation" of knowledge in AHA is controlled by a structured comment in an HTML page:

  ```
  <!-- generates readme -->
  ```

  This example *generates* comment denotes that the concept `readme` becomes known when the page is accessed.

- The relationship type *requires* links a concept to a virtual composite concept that is defined by a (constructor which is a) Boolean expression of concepts. Although in principle this composite concept is unnamed, we shall use a "predicate" or "pseudo attribute of the page" to refer to it. *P.requires* is used as a Boolean attribute of which the value is always that of the corresponding Boolean expression. It is not a user model attribute as its value is always computed on the fly and not stored in the user model. A *requires* relationship is implemented using a structured comment at the top of an HTML page, e.g.:

  ```
  <!-- requires ( readme and intro ) -->
  ```

  This example expresses that this page is only considered relevant when the concepts `readme` and `intro` are both known (100%). In AHA, links to a page for which *requires* is *false* are considered BAD, and reading such a page generates less knowledge than reading a GOOD page. Below we give the rules in *GA* that determines how the link anchors will be presented. They are very similar to the rules in Example 2 (Subsection 3.3):

  < CR.type = link and CR.cinfo.dir[1] = FROM and CR.cinfo.dir[2] = TO and CR.ss[2].uid.requires = true and CR.ss[2].uid.read = false => CR.ss[1].pres = GOOD >

  < CR.type = link and CR.cinfo.dir[1] = FROM and CR.cinfo.dir[2] = TO and CR.ss[2].uid.requires = true and CR.ss[2].uid.read = true => CR.ss[1].pres = NEUTRAL >

  < CR.type = link and CR.cinfo.dir[1] = FROM and CR.cinfo.dir[2] = TO and CR.ss[2].uid.requires = false => CR.ss[1].pres = BAD>

- The relationship type *link* only applies to pairs of pages in AHA. "Page selectors" that exist in

AHAM in general are thus not needed (or possible) in AHA.

AHAM allows author-defined specific adaptation rules only for the conditional inclusion of fragments in HTML pages. Structured HTML comments are used for specifying these rules. With a fragment F we can associate a "pseudo attribute" *requires* to indicate the condition, just like for whole pages. The syntax is illustrated by the following example:

```
<!-- if ( readme and not intro ) -->
... here comes the content of the fragment ...
<!-- else -->
... here is an alternative fragment ...
<!-- endif -->
```

AHA only includes fragments when their *requires* "attribute" is *true*.

The above examples illustrate that representing the actual functionality of an existing AHS in the AHAM model is fairly straightforward. The main reasons for using such a representation are to be able to compare different AHS, to possibly translate an adaptive hypermedia application from one AHS to another, and to identify potential problems or shortcomings in existing AHS.

We conclude this Section with an illustration of one specific shortcoming that we have found in both AHA [DC98] and Interbook [BSW96b]: the "new" knowledge values are calculated before generating the page (and in fact these systems do not support calculating knowledge values after generating a page at all). When a user requests a page, the knowledge generated by reading this page is already taken into account during the generation of the page. This has desirable as well as undesirable side-effects:

- When links to other pages become relevant *after* reading the current page it makes sense to already annotate the link anchors as relevant when presenting the page. Once a page is generated its presentation remains static while the user is reading it (and rightfully so). The new knowledge thus needs to be taken into account before the page is actually read.

- Pages contain information that becomes relevant or non-relevant depending on the user's knowledge. In some cases the relevance of a fragment may depend on the user having read the page that contains this fragment. This means that a fragment may be relevant the first time a page is visited and non-relevant thereafter, or just the other way round. By already taking into account the knowledge before the page is generated for the first time a different "first time version" becomes impossible to create. (Some readers may argue that having content that changes in this way may not be desirable in any case, but not having this possibility limits the general applicability of the AHS.)

5. Conclusions and Future Work

Over the past few years we have developed an AHS, mainly for use in courseware. We have come across a number of other AHS, with different interesting properties. As part of the redesign of AHA [DC98] we developed a reference model for AHS, named AHAM. The description of adaptive hypermedia applications in terms of this model has provided us with valuable redesign issues. The three most important ones are:

- The division of an adaptive hypermedia application into a *domain model*, *user model*, and *adaptation model* provides a clear separation of concerns and will lead to a better separation of orthogonal parts of the AHS functionality in the implementation of the next version of AHA. We believe that a system which supports this separation of concerns will not only result in a cleaner implementation, but also in a more usable authoring environment [WHD99a].
- In this paper we have described the *adaptation rules* in such a way that the rule definition is independent of the rule execution. This makes authoring easier.
- By representing AHA in the AHAM model we have identified another shortcoming: the lack of a two-phase application of rules. We found that this shortcoming is present in other AHS as well.

We deliberately based the AHAM model on the Dexter hypermedia reference model [HS90, HS94], to show that AHS are "true" hypermedia systems. In this paper we have concentrated on user modeling and

adaptation. The description of these aspects at an abstract level sets AHAM apart from other descriptions of AHS that are too closely related to the actual implementation of these AHS.

In the near future we will develop a new version of the AHA system, in which the separation of *domain model*, *user model* and *adaptation model* will be more complete. We also plan an extended paper with a complete formal definition of AHAM, including a formal specification of a language for specifying adaptation rules.

References

[B96] Brusilovsky, P., "Methods and Techniques of Adaptive Hypermedia". User Modeling and User-Adapted Interaction, 6, pp. 87-129, 1996. (Reprinted in Adaptive Hypertext and Hypermedia, Kluwer Academic Publishers, pp. 1-43, 1998.)

[DC98] De Bra, P., Calvi, L., "AHA: a Generic Adaptive Hypermedia System". Proceedings of the Second Workshop on Adaptive Hypertext and Hypermedia, Pittsburgh, pp. 5-11, 1998.

[DHW99] De Bra, P., Houben, G.J., Wu, H., "AHAM: A Dexter-based Reference Model for Adaptive Hypermedia". Proceedings of ACM Hypertext'99, Darmstadt, pp. 147-156, 1999.

[HS90] Halasz, F., Schwartz, M., "The Dexter Reference Model". Proceedings of the NIST Hypertext Standardization Workshop, pp. 95-133, 1990.

[HS94] Halasz, F., Schwartz, M., "The Dexter Hypertext Reference Model". Communications of the ACM, Vol. 37, nr. 2, pp. 30-39, 1994.

[PDS99] Pilar da Silva, D., "Concepts and documents for adaptive educational hypermedia: a model and a prototype", Proceedings of the Second Workshop on Adaptive Hypertext and Hypermedia, Pittsburgh, pp. 33-40, 1998.

[WHD99a] Wu, H., Houben, G.J., De Bra, P., "Authoring Support for Adaptive Hypermedia", Proceedings ED-MEDIA'99, Seattle, pp. 364-369, 1999.

[WHD99b] Wu, H., Houben, G.J., De Bra, P., "User Modeling in Adaptive Hypermedia Applications", Proceedings InfWet99, Amsterdam, 1999.