AHAM: A Reference Model to Support Adaptive Hypermedia Authoring

Hongjing Wu, Geert-Jan Houben, Paul De Bra
Department of Mathematics and Computing Science
Eindhoven University of Technology
PO Box 513, 5600 MB Eindhoven
the Netherlands
email: {hongjing,houben,debra}@win.tue.nl

Abstract:

Hypermedia applications generally contain a rich (hyper)link structure. For users of a hypermedia application the richness of this link structure can generate comprehension and orientation problems. Adaptive techniques have been used by a number of researchers [see e.g. 1,2,4,5,8,11,12,15,16,17] in an attempt to offer guidance through and orientation support for rich link structures. The majority of these adaptive hypermedia systems (AHS) have been used in educational applications, although some on-line information systems (or "kiosk"-systems) and information retrieval systems have been made adaptive as well.

In this paper we describe a reference model for adaptive hypermedia applications, called AHAM, which encompasses most adaptive features supported by adaptive systems that exist today or that are being developed (and have been published about). The model uses "educational terminology", but this does not imply that we would preclude other types of applications. Key aspects in AHAM are:

- *The adaptation is based on a* domain model*, a* user model *and a* teaching model *which consists of* pedagogical rules*. We give a formal definition of each of these (sub)models.*
- *We distinguish the notions of* concept *and* page*. In several AHS these notions are confused.*
- *We provide a formalism which lets authors write pedagogical rules (about concepts) in such a way that they can be applied automatically.*
- *The model is specifically designed to make it easy to develop authoring tools for it.*
- *AHAM supports constructing the user model not only from the user's reading and navigation, but also from external sources, such as tests. This feature is useful for exchanging parts of a user model between different AHS.*

We illustrate various aspects of AHAM by means of the AHA system, used in distance learning applications and a kiosk information system at the Eindhoven University of Technology.

1. Introduction

Hypermedia systems in general, and Web-based systems in particular, are becoming increasingly popular as tools for user-driven access to information. Adaptive hypermedia is a new direction 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 applications by making them personalized. An adaptive hypermedia system (or AHS for short) builds a model of the goals, preferences and knowledge of each individual user. It uses this model to adapt the application to the needs of the user.

In order to clarify the goal of this research, we recall the following definition from Brusilovsky [1]:

> *By adaptive hypermedia systems we mean all hypertext and hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user. In other words, the system should satisfy three criteria: it should be a hypertext or hypermedia system; it should have a user model; it should be able to adapt the hypermedia using this model.*

In this paper we start by providing some background information on adaptive hypermedia. We recall commonly used methods and techniques in this research field. (Readers who are intimately familiar with this field can safely skip Section 2, which briefly summarizes definitions and comments from [1] and [11].) In Section 3 we show the shortcomings of typical AHS that are based on these methods and techniques. Most of these problems are in the area of authoring and maintenance. In Section 4 we describe AHAM, the **A**daptive **H**ypermedia **A**pplication **M**odel, in which adaptive applications can be described in a way which is easy for the author, yet at least as powerful as the systems that exist today or that are under development (and that we know about from preliminary publications). We also describe the existing AHA system [11] in terms of the AHAM model. (AHA is the **A**daptive **H**ypermedia **A**rchitecture, used for distance learning applications and for a "kiosk" information system at the Eindhoven University of Technology. [See URL references]) From this description we derive directions in which AHA can be further developed. AHAM suggests how to make authoring for AHA much easier, and how to enhance the possibilities for adaptation as well.

2. Adaptive hypermedia methods and techniques

Following Brusilovsky [1] we distinguish between high level methods for adaptive hypermedia support and lower level techniques that are used to realize or implement that support. By a method we mean a notion of adaptation that can be presented at the conceptual level. A technique is then a way to implement a specific method. Techniques operate on actual information content and on the presentation of hypertext links. It may be possible to implement the same method through different techniques and to use the same technique for different methods.

In this paper we distinguish between *content-adaptation* and *link-adaptation*. (Brusilovsky [1] calls these *adaptive presentation* and *adaptive navigation*.) In Section 2.1 we present different methods and techniques for content-adaptation. In Section 2.2 we do the same for link-adaptation.

2.1 Content-adaptation

It may be desirable to present information on a certain topic in different ways, depending on the user's (fore)knowledge, goals, preferences or other characteristic properties of the user. For example, a user with experience in the domain can be shown more specific, more detailed information, while a novice receives additional (and maybe introductory) explanations. In a hypermedia system the content of a "page" may not only be text, but it may also contain various

multimedia items. In AHAM, which we present in Section 4, we do not distinguish between text and multimedia since we treat content at an abstract (media independent) level. We must remark however that the current generation of AHS only offers content-adaptation for text, and just media selection for multimedia items.

2.1.1 Content-adaptation methods

In [1] three main content-adaptation methods are considered:

- additional, prerequisite, and comparative explanations
- explanation variants
- sorting

The most popular method appears to be the *additional explanations*. Its goal is to provide additional information, explanations, illustrations, examples, etc., to those users who appear to need or want them. At the same time the system hides such explanations from users who do not want them, for instance because of a mismatch in level of difficulty, or because they prefer terse and basic information only. *Prerequisite explanations* are to some extent a special case: an explanation is added because the system decides that without this explanation the user will (or may) not understand the remainder of the page. Prerequisite explanations are thus used to compensate for the lack of prerequisite knowledge. *Comparative explanations* are added when there is information about other concepts that are similar to the one described in the "current" page (or dissimilar in a specific way, or otherwise related in some "interesting" way). Showing this additional information about how the current topic relates to these other concepts only makes sense when these other concepts are known to the user.

Explanation variants are used in cases where all users need roughly the same information (or explanation), but they need a different presentation of it. Some variants may be more or less verbose, and may use or avoid specific technical terms for instance.

With *sorting* the goal is to present the same information items to all users, but to order them from most relevant to least relevant or according to some other criterion, which not only depends on an explicit goal (such as a search request) but also on a user's background and foreknowledge.

2.1.2 Content-adaptation techniques

There are five kinds of techniques to implement the above mentioned methods:

- conditional text
- strechtext
- fragment variants
- page variants
- frame-based technique(s)

With *conditional text*, all available information about a concept is divided into several chunks of texts (or multimedia content). With each chunk a (Boolean) condition is associated on elements of the user model. When displaying a page about the concept, the system only presents the chunks for which the condition is true. This is the lowest-level technique, but also the most flexible one. It can be used to "simulate" the four other techniques. With conditional text it is

easy to implement the methods of additional, prerequisite and comparative explanations. However, because it is such a low-level technique writing conditional text makes authoring look like programming. The method of sorting cannot easily be implemented using this technique. Conditional text is the technique used in the current AHA system [11].

The Guide hypertext system [3] offered *replacement links*. Clicking on a word or phrase would open up a content fragment (paragraph) on that term. Parts could be opened and closed as desired. The technique of *stretchtext* is very similar. But while in Guide all fragments would initially be closed, in stretchtext the AHS determines which fragments to open (or stretch) when displaying a page. Stretchtext is useful for implementing additional, prerequisite or comparative explanations, but less for explanation variants and not at all for sorting.

With *fragment variants* one can easily implement explanation variants. The AHS stores several variants of the same information fragment, and selects the variant to display based on the user model. On one information page there may be many fragments, each with several variants. The author needs to keep an overview of all possible combinations of fragment variants the system may select and present on one page.

The use of *page variants* is simpler: variants of a whole page are prepared by the author, and the variant which is most appropriate for a given user is selected by the AHS. It is easier for the author to keep an overview of all possible presentations, but there may be a lot of overlap between different variants of the same page.

Frame based techniques are somewhat like fine-grained fragment variants. In addition to selecting which fragments to select for assembling a page, the AHS also decides in which order to present these fragments. Often natural language techniques are needed to convert the generated sequence of fragments into naturally sounding text. The fragments are really atomic units of information, usually individual words or short phrases. The only content-adaptation method that requires the use of frame based techniques is sorting.

2.2 Link-adaptation

The basic idea with link-adaptation is to adapt the rich link structure in such a way that the user is guided towards interesting, relevant information, and kept away from non-relevant information. Link-adaptation tries to simplify the rich link structure to reduce orientation problems, while maintaining a lot of navigational freedom, which is typical of hypermedia systems.

2.2.1 Link-adaptation methods

Brusilovsky [1] mentions five link-adaptation methods:

- global guidance
- local guidance
- local orientation support
- global orientation support
- managing personalized views

Guidance can be provided in hypermedia applications where users have some goal in terms of information they want (they need information which is contained in one or several pages somewhere in the hyperspace) and where browsing is the preferred or only way to find the

required information. *Global guidance* means that the system suggests navigation paths on a global scale. This is especially useful in educational hypermedia. When a user wishes to learn about a certain topic, the system may suggest a set of pages to read, along with an indication of a desired or at least meaningful reading order. Guidance is needed because whether a reading order is sensible or not depends on "prerequisite" relationships between concepts. *Local guidance* means that the system suggests the next step to take, for instance through a "next" or "continue" button.

Orientation support means that the system presents an overview of the whole (link) structure of the hyperspace (*global orientation support*), or of a part thereof (*local orientation support*). The system also indicates the position of the user (of the "current" page) in this structure. When the orientation support is *adaptive* it means that the structure which is shown indicates relevant parts to go to, parts that were visited before, and parts that are (still) to be avoided. Global orientation support is sometimes offered through a kind of "table of contents" column in Web-based presentations. Local orientation support shows a small part thereof, like one or two (link) levels up or down from the "current" page.

Another way to protect users from the complexity of the overall hyperspace is to let them organize *personalized goal-oriented views*. Each view may be a list of links to all pages or sub-parts of the whole hyperspace which are relevant for a particular working goal. Simple forms of personalized views are bookmark lists. Advanced forms of views have been introduced in the literature, including Coollists (as opposed to hotlists).

2.2.2. Link-adaptation techniques

In [1] and [11] (together) we find seven techniques for link-adaptation:

- direct guidance
- link sorting
- link hiding
- link annotation
- link disabling
- link removal
- map adaptation

Direct guidance means that the "next best" node for the user to visit is shown, e.g. through a "next" or "continue" button. (The destination for this link is determined by the AHS, based on the user's goal and other elements from the user model. It is not like a static "next page" button.) Direct guidance is an obvious choice to implement local guidance.

The basic idea of *adaptive link sorting* is to sort all the links on a particular page according to the user model and to some goal-oriented criteria: the more towards the top of the page, the more relevant the link is. Adaptive sorting can be used for global guidance: all links from the node are sorted according to their relevance according to the global goal. Special care needs to be taken to ensure that the order of the links does not change while the user is trying to reach the global goal. Sorting is typical for information retrieval systems. However, the "teach me" feature of Interbook [7] for instance illustrates that sorting can be used in an educational AHS as well.

With the technique of *link hiding* the navigation space is simplified by hiding links to "non-

relevant" pages. With contextual links (links occurring as "hot words" or "anchors" within the text) hiding can be realized by changing the color of the anchors to that of normal text. For non-contextual this technique doesn't work. In an index or map hiding can only be achieved by making the anchors transparent (or giving them the "background color"). However, this looks more like *link removal* than like hiding (see below).

The idea of the adaptive *link annotation* is to augment the link with some form of comment which tells the user more about the current state of the pages to which the annotated links refer. Adaptive annotation can be used to realize each of the first four possible link-adaptation methods. It provides for a stable order of links. It is more powerful than hiding (as hiding is in essence also a form of annotation), but it does not reduce the cognitive overhead as much as hiding does. (The navigation space is not simplified by showing fewer links.) Link annotation informs the user about the current "state" of the pages the links point to. We have found three methods for deciding how to annotate links:

- The annotation may indicate the relevance of a link. Colors may be used to distinguish for instance "highly relevant", "somewhat relevant", "not relevant".
- The annotation may indicate whether the user already knows the concepts described in the page a link points to. Several degrees of knowledge may be distinguished, for instance "not known", "learned", "well-known".
- The annotation may also indicate whether a user is able to understand the information contained in the destination page ("ready to read" vs. "not ready to read").

All three types of annotation can be used simultaneously (provided that the system is able to distinguish many link types in a visual way). For instance, links to non-relevant pages may be hidden, while links to relevant pages are colored differently depending on whether the user is "not ready to read" them, "ready to read them for the first time", or "has already read" them.

The basic idea of *link removal* [11] is that link anchors for undesired links (non-relevant or not yet ready to read) are removed. This technique works well for non-contextual links, like in lists or maps, but cannot be used in running text, because the words that make up the link anchor cannot be omitted.

The *link disabling* [11] technique is based on the idea that the "link functionality" of a link is removed. This technique is not usable by itself. The disabled links must be either hidden or annotated so that the user does not expect the links to work.

Map adaptation is mentioned by Brusilovsky [1] as a separate technique, in which the content and presentation of a map of the link structure of the hyperspace is adapted. But in fact, the adaptation to the map can be seen as a combination of link removal operations, annotations and (maybe two-dimensional) sorting.

3. Shortcomings in adaptive hypermedia methods and techniques

The possibilities an AHS offers for doing "useful" and "usable" adaptation in a given application depend on three factors:

- The application must provide a *domain model*, describing how the information content of the application (or "hyperdocument") is structured. This model must indicate what the relationship is between 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. At the same time the system should infer how the relevant aspects of the user change while the user is using the application.

- 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 a *teaching model* consisting of *pedagogical 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.

The main shortcoming in the current generation of AHS is that these three factors or components are not clearly separated:

- The relationship between pages and concepts is sometimes too vague [17]. When 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. In other systems the relation is strictly one-to-one, which leads to a very fragmented user model without high-level concepts.

- The pedagogical rules can often not be defined at the conceptual level, because they are tied to the links or the (conditional) text fragments, and because as a consequence they must often be specified within the actual information pages.

- The user model is often unreliable because the AHS only receives unreliable input. Especially in many Web-based AHS most information in the user model is generated from access patterns that are logged by the Web-server (or a server-side script). The system does not know how long a user has been reading a page, because there is no user surveillance. The system does not even know how long the page was up on the user's computer screen. (The AHA system [11] is a notable exception: it does log the time a page is being displayed.) Tests (multiple-choice questions or other tests) can be used to verify the user's knowledge in a more reliable way. However, integrating tests into a system in a meaningful way that does not just require extra effort from the user but that actually adds to the learning process, is a non-trivial task. Also, tests are not commonly accepted in non-educational applications. We do not address the unreliability of user models in this paper.

In the next section we provide a framework for developing adaptive hypermedia applications in a way that makes it possible to separate *domain model*, *user model* and *teaching model*.

4. The Adaptive Hypermedia Application Model (AHAM)

In this section we describe a model, called AHAM, by which an adaptive hypermedia application is divided into four parts:

- A *domain model* describes how the application domain is structured. It does this both at the conceptual level and at the level of information fragments and pages.

- A *user model* describes the user's knowledge of the (concepts of) the application domain.

This is a so called *overlay model*: for each concept the user's knowledge of that concept is stored in the user model. The user's knowledge is thus a vector in a high-dimensional space. A user's knowledge can be compared to predefined stereotypes by calculating the distance (in the high-dimensional space) between the user's vector and the vector that corresponds to a stereotype. But the knowledge of each concept can be used separately as well, for adaptation to the fine-grained representation of a user's knowledge.

- A *teaching model* describes *pedagogical rules* that indicate under which circumstances it becomes desirable or undesirable for the user to be guided towards certain parts of the application domain. Many rules follow directly from "structural" relationships between concepts, thus from the domain model, but the author can also write additional rules. While we call this part a teaching model it does not mean that AHAM is intended specifically for educational applications.

- An *adaptive engine* is the actual software environment that is used for constructing and adapting content and links. The engine offers a library of functions for constructing information pages from fragments, based on elements from the domain model, user model and teaching model. A (usually very simple) language is used to select the "constructor" to use. Some engines may offer a way to define new constructors or extend existing ones. However, a sufficiently powerful adaptive engine should offer enough standard functionality to alleviate the need for authors to explicitly specify new constructors in most applications. The engine also updates the user model by observing the user's reading behavior and thus taking notice of how the user's knowledge changes.

Adaptive hypermedia applications are first of all *hypermedia* applications. Information elements are called *nodes*. There are *atomic* nodes (of which the internal structure is of no concern of the hypermedia system) and *composite* nodes, which consist of smaller parts (*nodes* and possibly also hypertext *links*). An information page as seen by the user may be an atomic node, but in most cases it will be a composite node, consisting of several parts, some of which are text fragments and some are images or possibly objects in other media.

While reading and browsing through the information nodes a user gains knowledge about concepts. Relationships between concepts are used to guide the user towards (not previously read) information about relevant concepts, and to adapt the content of nodes to the knowledge level of the user. Each node may contribute towards the knowledge about one or more concepts. The user's knowledge about concepts can thus "gradually" increase. An AHS may also take into account that a user's memory isn't perfect and that she thus gradually forgets what she read. (We don't describe this "forget-function" in this paper. It can be implemented as an automatic background function performed by the adaptive engine.) The AHS may also offer the possibility for external applications to modify the user's knowledge of certain concepts. A typical example of such external applications is the use of multiple-choice tests. (We don't describe these external user-model update functions of the adaptive engine in this paper either.)

Definition 1: A *concept* is an abstract representation of an information item from the application domain. Every concept has an object identity. For simplicity we shall call this the *concept name*. We consider *low-level* or *atomic* concepts, which correspond to a single information fragment (see Definition 4), and *high-level* or *composite* concepts which consist of a set of other (high- or low-level) concepts.

Definition 2: The AHS associates a number of (system or author defined) attributes to each concept of the domain model. For each user the system maintains a table (a relation in database terminology) in which for each concept the attribute values for that concept are stored. The structure of this table is the *user model scheme*. The table for a specific user is a *user model*

instance. If there is no confusion between scheme and instance we just use the term *user model*. The "first" attribute of the user model is always the *concept name*. Other typical attributes are:

- The *knowledge value* (or *value* for short) indicates how much the user knows about the concept. The <concept, value> pairs together form an *overlay model*. It represents the "knowledge" of the user.

- The *read* attribute indicates whether the user read something (a fragment, a page or a set of pages) about the concept. The *read* attribute is often used to generate a different presentation for links to pages that were read before than for links to unread pages.

- The *ready-to-read* attribute indicates whether the AHS considers that the user is ready to read about this concept.

While most AHS to date provide a fixed set of attributes, future AHS may offer the possibility for authors to invent new attributes. Also, the "table" representation only exists at the conceptual level. An actual AHS may use an implementation of the user model which is different. The *read* attribute for instance may be "derived" from a server log file which is a list of <user, page, timestamp> tuples.

For convenience we shall use the notation C.*attr* to indicate the value for attribute *attr* in the tuple which has the value *C* for the attribute *concept name*. We also assume that there is no confusion about the user's identity.

Definition 3: A *concept relationship* is a tuple $<C_1,C_2,T,A>$ where C_1 and C_2 are concepts, T is the *type* of the relationship and A is an attribute value (corresponding to type T).

Some examples of possible relation types and associated attributes are:

- The type *prerequisite* means that C_1 should be studied before C_2. The attribute is a real number between 0 and 1, indicating how much knowledge a user should have about concept C_1 for information about C_2 to become desirable.

- The type *inhibitor* means that C_1 should not be studied before C_2. The attribute is also a real number between 0 and 1. It indicates that when the user's knowledge about C_1 exceeds the attribute's value then it becomes undesirable to read about C_2.

- The type *part of* represents a *compositional* relationship: C_1 is a part of C_2. The attribute is a real number between 0 and 1, indicating what fraction of C_2 is represented by C_1.

- The type *link* represents that there is a (hypertext) link from C_1 to C_2. The attribute is used to denote whether this is a link that points to an external source (not interesting for adaptation), whether it is a static link to another concept of the application, or whether it is an adaptive link. In the sequel (Definitions 4 and 5) we explain that we associate concepts to content fragments and pages. In this way we can translate links between concepts to links between fragments and pages.

Each AHS (or the "adaptive engine" in particular) will offer support for a standard set of relationship types, most likely including *prerequisite*, *inhibitor*, *part-of* and *link*, but possibly some others as well. Some systems may allow authors to "invent" new types as desired, and specify *pedagogical rules* (see Definition 7) to indicate how to handle these types.

Definition 4: A *fragment* is an atomic content unit (as far as the adaptation is concerned). A fragment is a tuple <ID, VAL, CN>. ID is the object identity of the fragment, VAL is the

(content) value and CN is the name of an atomic concept associated with the fragment.

While a fragment is an atomic unit as far as adaptation is concerned, it may have an internal structure which has meaning for the hypermedia system. For instance, it may be a piece of HTML source consisting of several paragraphs, and possibly including images or other objects as well.

Definition 5: A *page* is a tuple <SF, C, CT> where SF is a set of fragment ids, C is a (usually composite) concept associated with the page and CT is a (page) constructor. The constructor is a software function (part of the adaptive engine) that builds the page from its fragments. It does so by conditionally including (and sorting) fragments based on the user model and on the concept relationships. Each AHS offers a set of standard constructors, which should be sufficient for most authors. However, the system might also offer a (hopefully very simple) kind of programming language to let authors create new page constructors. The constructor may use rules from the *teaching model* which we describe in Definition 8. These rules make the meaning of concept relationships explicit.

Associating a (possibly new) concept to every page makes it easier to express concept relationships that indicate that how the relevance of pages depends on the user having read other pages.

- A typical standard constructor is a "sequence" constructor, which puts fragments in a fixed order, and includes each fragment only if the prerequisite concept relationships for that fragment indicate that the user is ready to read the fragment and if none of the inhibitors indicate that the user is not (or no longer) ready to read the fragment.

- Another typical constructor is the "identity" constructor, which assumes that the page consists of only one fragment. The user is ready to read the whole page if the prerequisite concept relationships and the inhibitors for the concept associated with the page indicate so.

In the case of a one fragment page we normally use the same concept for the fragment and for the page. We assume there is a predefined concept relationship of type *on page*. This relationship exists between all pairs of concepts. It has a Boolean attribute which has the value *true* when the first concept is associated with a fragment, the second concept with a page, and when the fragment belongs to that page.

Definition 6: The *domain model* is a tuple <SP, SC, SCR>, where SP is the set of pages, SC is the set of concepts and SCR is the set of concept relationships.

Note that since the pages contain their fragments, and the fragments are associated with (atomic) concepts, the domain model contains all the information about how the information content is linked to the information concepts. The concept relationships indicate how composite concepts are composed of (other composite and/or) atomic concepts, and which prerequisite (or other) relationships there are between concepts.

The actual adaptation which takes place is based on so called *pedagogical rules*. These rules define how attribute values of concept relationships are interpreted, both in general and in specific cases.

Definition 7: A *generic pedagogical rule* is an association <CT, S> between a concept relationship type CT and a statement S over the concepts and attribute value(s) of the concept relationships of that type. A *specific pedagogical rule* is just a statement over a set of (concrete) concepts. Each statement assigns a value to some attribute "of" a concept, or a "predicate" over

several concepts. The syntax of the permissible statements depends on the language supported by the AHS. Each AHS offers some predefined attributes that together form the *user model*. The AHS also offers some implicit (default) generic pedagogical rules for these attributes.

Let us assume $<C_1,C_2,T,A>$ is a generic notation for a concept relationship. We give a few examples of pedagogical rules. Let us assume that *read* is a predefined attribute, stored in the user model, and meaning that the user has read about the associated concept C. Let us also assume that *reading* is a pseudo-attribute, not in the user model, which represents the event of a user requesting to read about the associated concept C. Through these two attributes we can do adaptation just before showing information about C for the first time in a different way than when revisiting that information later. The attributes *read* and *reading* are probably only defined for concepts that are associated with pages and fragments. The adaptive engine is responsible for triggering pedagogical rules when a user requests a page (by following a link or otherwise). What happens is the following:

The AHS maintains the value for a number of attributes in the user model. There are events that trigger the adaptive engine into updating the user model and generating a presentation of information about some concept(s). Following a link (and then reading a page) is such an event, but there may also be other events to link the AHS to external applications like (multiple-choice) tests or tools for importing parts of a user model from another application. In this paper we concentrate only on the "reading" event.

When a page is requested, the following happens:

1. The adaptive engine retrieves the stored user model. Thus, for some attributes the "previous" values are retrieved.

2. For the attributes that are not part of the user model, but that appear in pedagogical rules, the values are initialized to a predefined default value.

3. The *knowledge value* of the concept associated with the requested page will increase, and the *read* attribute also changes. (It may for instance become *true* if this attribute is of type Boolean.)

4. The pedagogical rules are applied to deduce updates to attribute values that are a consequence of the event that occurred. It is important that the rules are defined in such a way that when the updates are propagated no infinite loops occur, and that the updates are independent of the order in which the rules are applied.

5. Using the updated attribute values a page constructor is used to generate the presentation of the requested page.

6. The updated user model is saved.

The procedure described above assumes that each event results in the adaptive engine being started, the user model being read and updated, a page being generated and the updated model being saved. When CGI-scripts are used as the technology on which to base an AHS this scenario will closely resemble the actual way in which the engine works. But an AHS may of course have optimizations to avoid some of the storage, retrieval and recalculation work, as long as the observable behavior remains the same.

A first example of a possible standard generic pedagogical rule (written in a C-like syntax) is:

<on page, C_2.value = C_2.reading ?1: C_2.value>

The rule means that when a page is requested the knowledge about the concept associated with that page becomes 1, and the knowledge about concepts associated with other pages remains unchanged. This would typically be a predefined rule. It only operates on pages, because the concept C_2 must be the second concept of an "on page" relationship. Also, while the rule specifies what happens for pages that are not the one being read, the adaptive engine will of course not execute these instantiations of the rule that have no effect. An author could override this rule, for instance to indicate that the value should only become 0.8 instead of 1.

Next we show some typical standard generic rules for an attribute "ready-to-read". This could be a predefined Boolean attribute that is part of the user model. However, here we shall assume that it is a Boolean attribute that is not part of the user model. We also assume this attribute is initially true for all concepts. Note that "initially" means that this default value is used again each time an event occurs. This contrasts with attributes of the user model which retain their (previous) value in between events.

<prerequisite, C_2.ready-to-read &= C_1. read || C_1.value>A>

This rule says that for all "prerequisite" relationships the concept C_2 is not "ready-to-read" if the knowledge of C_1 does not (yet) exceed the value of A, and if C_2 has not yet been read. The standard constructors for a page (like the sequence constructor) will include a fragment only if its corresponding concept is marked as "ready-to-read" (meaning that "ready-to-read" is *true*). This leads to a somewhat complicated predefined generic rule:

<on page, C_1.value = (A && C_2.reading && C_1.ready-to-read)?1: C_1.value>

This rule says that if C_1 corresponds to a fragment of the page with which C_2 is associated (meaning that the attribute value A is *true*) then the knowledge value of C_1 becomes 1 when the page is being read and the fragment is "ready-to-read" and will thus be shown as part of the page. The knowledge value for the fragments that are not shown (that are not "ready-to-read") remains unchanged. Similarly one can develop a rule that says that the knowledge value for a (concept associated with a) page is only set to a small amount like 0.5 if the page is ready while its concept is not marked as "ready-to-read".

Ordering can be implied by a (predefined, generic) pedagogical rule as well:

<additional, precedes(C_2,C_1) = true>

means that if C_1 offers additional information to C_2 then information (content) corresponding to C_2 must be displayed before (above) the information about C_1. The standard sequence constructor will use the "precedes" predicate to sort fragments.

An example of a specific pedagogical rule over two concepts *pre* and *post* is:

post.ready-to-read &= *pre*.value<0.8

This rule specifies that when the user knows more than 0.8 about concept *pre* then she is not (or no longer) ready to read about *post*. All specific pedagogical rules must be written by the author. An AHS only offers predefined generic rules.

Specific pedagogical rules take precedence over generic rules, and author-defined pedagogical rules take precedence over predefined standard rules. It is thus possible to override or define a generic rule for all concept relationships of a certain type, and then to make exceptions for a few specific relationships of the same type and apply a different rule for them. Although the above examples do not show it, pedagogical rules may deal with more than one concept relationship or more then just a pair of concepts. For instance, a specific rule over concepts A, B and C may

indicate that:

A.value = 0.8 * (B.value + C.value)

This rule means that the knowledge (of a user) about (composite) concept A is 80% of the sum of the knowledge about B and C. It could be used to override a (predefined) generic rule like:

<part of, C_2.value += C_1.value * A>

It is thus possible to define specific ways in which knowledge about sub-concepts adds up towards knowledge about a composite concept. Not being able to do so is one of the problems of the AHS described in [17].

Pedagogical rules are also used to prepare link-adaptation. They are used to set a "status" for a link. If <C_1,C_2,link,conditional> is a generic representation of an adaptive link, then we can specify a generic rule:

<link, hidden(C_1,C_2) = ! C_2.ready-to-read>

This rule says that links to concepts that are not "ready-to-read" will be hidden. Note that according to this rule the fact whether a link is hidden or not depends only on the status of the destination of the link, not on the source. One can of course also specify generic or specific rules for a link status that depends on both the source and the destination.

Definition 8: A *teaching model* is a tuple <PGR,SGR,SSR> which consists of a set of predefined generic pedagogical rules, a set of author-defined generic pedagogical rules SGR and a set of specific pedagogical rules SSR.

The teaching model provides a translation from relationships and relationship types between concepts to system- or author-defined attributes and predicates, such as the "ready-to-read" attribute and the "precedes" predicate we described above. When only using predefined attributes and predicates the author can use standard page constructors offered by the adaptive engine (see below). In order to handle author-defined attributes or predicates it will be necessary to write new page constructors. Still, even in that case the authoring is simplified by first writing (many) rules that all contribute towards determining the value of a single attribute or predicate, and then only having to use that attribute or predicate in the page constructor.

Definition 9: An *adaptive engine* is a software environment that performs the following functions:

- It offers generic page constructors. For each page the author selects a constructor to use for building the adaptive presentation of the page.
- It optionally offers a (very simple programming) language for describing new page constructors.
- It performs adaptation by executing the page constructors. This means 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 pedagogical rules. The engine will thus set the knowledge value for each atomic concept of displayed fragments of the page to a value which depends on a configurable amount (this could be 1 by default but possibly overridden by the author). It

also maintains a log of visited (concepts associated with) pages.

We are now ready to define a complete adaptive hypermedia application:

Definition 10: An adaptive hypermedia application is a 4-tuple <UM,DM,TM,AE> where UM is a *user model*, DM is a *domain model*, TM is a *teaching model*, and AE is an *adaptive engine*.

In the next subsection we show how the different content and link adaptation methods and techniques can be expressed in AHAM. Subsection 4.2 shows how to represent current and planned features of the AHA system correspond to elements of AHAM. (AHA is being developed and used at the Eindhoven University of Technology.) Section 5 shows how AHAM can form a basis for building authoring tools that separate (to a large extent) the definition of the user model, the domain model and the teaching model. We argue that the separation of these models makes adaptive hypermedia application authoring easier.

4.1 Adaptation expressed in AHAM

AHAM is an adaptive hypermedia *model*. It describes features of adaptive hypermedia applications at a conceptual level, not at an implementation level. The actual construction of pages, and the actual techniques used to manipulate links, depend on the language and features offered by the adaptive engine. The model does not prescribe how the (page) constructors are built, but it does describe the kinds of rules that can be used by these constructors.

The content-adaptation methods *additional*, *prerequisite* and *comparative explanations* are represented as follows:

- For each method a concept relationship type with the same name is introduced (by the author or the system).
- For each method a generic pedagogical rule is introduced that describes when information about a concept is "ready-to-read".

For example, the *prerequisite* relationships result in the rule:

<prerequisite, C_2.ready-to-read = C_1.value>A>

There can be rules that say that a page or fragment is "ready-to-read" when the user has a certain knowledge, but also to say that the fragment is needed when the user does not (yet) have a certain knowledge.

Likewise, *Explanation variants* can be implemented as fragments such that only one of them (has an associated concept that) is "ready-to-read". For example:

before-A.ready-to-read = *A*.value<=0.8

after-A.ready-to-read = *A*.value>0.8

Sorting relies on the ability of the adaptive hypermedia engine to perform sorting of concepts based on concept relationships and/or pedagogical rules. Apart from rules that determine whether a fragment is "ready-to-read" some ordering rules need to be formulated resulting in attributes like "precedes(A,B)". The adaptive engine must offer the possibility to define a constructor which sorts fragments depending on the order that results from sorting the concepts associated with the fragments. The adaptive engine of the AHA system [11] for instance does not (yet) offer such functionality. Hence, AHA does not support sorting as a content-adaptation method.

As can be seen from the above explanation, AHAM strongly suggests that the low-level technique that should be used is that of *conditional text*, with some sorting thrown in. However, this is only one suggestion. Depending on the availability of functionality for other kinds of page constructors one could also model *stretchtext*, *fragment variants* and *page variants* directly.

Link-adaptation relies on the use of some generic and specific concept relationships, including *link* relationships and *prerequisite* relationships, as well as pedagogical rules for these types of relationships. It also implicitly relies on the user model, as the knowledge values for concepts are used in the pedagogical rules. A rule can for instance express that a link to a concept (which still has a low knowledge value) is "desired" when certain prerequisite knowledge is acquired by the user, and that it is "hidden" otherwise. The adaptive engine needs to translate these attribute values for the links into the actual corresponding manipulation of the link anchors. In this way one can realize *global* and *local guidance*. Providing *global* or *local orientation support* cannot be described in a similar way: this orientation support is offered by displaying links to related parts of the application. This requires generating some form of content, and thus requires a page and corresponding constructor. The same is true for adaptive *personalized views*.

The inability of AHAM to express orientation support as a pure form of link-adaptation stems from a basic underlying assumption that links have (source) anchors that are constructed and presented in the same way as fragments and pages are, and that techniques like link hiding and annotation are realized by the adaptive engine by manipulating page contents. This is not a restriction on the usability of AHAM: all hypermedia systems need a presentation for link anchors in order for the user to be able to "follow" a link. In a number of "open" hypermedia systems in which the "link service" is separate from the textual content the link anchors do not occur in the source text of content pages, but they do appear as anchors in the presented text that is generated by the system. So in the context of AHAM even such systems generate link anchors as content elements.

4.2 Example: modeling the AHA system

AHA (for "Adaptive Hypermedia Architecture") is a simple adaptive engine which is being used at the Eindhoven University of Technology for serving two adaptive course texts and one kiosk system. (See URL references) In this section we briefly explain how AHA works by describing AHA in terms of the AHAM model. The current version of AHA is 1.0. We are working on a newer version with more features and better authoring support. We indicate extensions that are under development where appropriate.

AHA is Web-based. It uses HTML as the language in which to write content as well as page constructors. *Fragments* are pieces of HTML source. (They may include images, applets and other objects, and they may be missing opening tags at the beginning or closing tags at the end.) Fragments are delimited by HTML comments that indicate the beginning and end of a fragment and that are used in constructing pages. In AHA the atomic concepts associated with fragments are not named. Also, in AHA each fragment appears in only one page.

With every (HTML) page in AHA we associate zero or more concepts. Usually there is exactly one concept, and it has the same name as the page (minus the ".html" filename suffix). Although strictly speaking this is just a convention, we shall assume this one-to-one mapping between pages and concepts in the sequel. The association of a concept to a page is indicated by an HTML comment near the top of the page, like for a readme page:

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

Every page starts with a "requires" HTML comment that looks like:

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

This comment actually indicates a pedagogical rule. If the comment appears in the page "this-concept.html", the rule would be:

this-concept .ready-to-read = readme.value==1 && intro.value==1

Initially the user's knowledge about all concepts is 0. When a user requests a page which is "ready-to-read", the knowledge value for the concept associated with this page becomes 1. This is described by the system-supplied rule:

<on page, C_2.value = (C_2.reading && C_2.ready-to-read)?1: C_2.value>

The order in which fragments appear in AHA documents is fixed. It is the order in which they appear in the HTML page that contains all the fragments from which the page is to be constructed. The default and only page constructor offered by AHA is thus the "sequence constructor". The author can define rules to determine whether a fragment will be displayed. This is again done through HTML comments:

```
<!-- if ( readme and not intro ) -->

... here comes the content of the fragment ...

<!-- else -->

... here is an alternative fragment ...

<!-- endif -- >
```

If the two fragments correspond to concepts F_1 and F_2 then this AHA construction corresponds to the following specific pedagogical rules:

F_1.ready-to-read = readme.value==1 && intro.value==0

F_2.ready-to-read = !(readme.value==1 && intro.value==0)

AHA offers link-adaptation through annotation and hiding. We only show the adaptation for internal conditional links. There are three cases:

- A link points to a page that is "ready-to-read" and has not been read before. (The link is then considered *good* and shown in blue.)
- A link points to a page that is "ready-to-read" but that was read before. (The link is then considered *neutral* and shown in purple.)
- A link points to a page that is not "ready-to-read". (The link is then considered *bad* and shown in black.)

The pedagogical rules for conditional links are shown below:

<link, good(C_1,C_2) = C_2.ready-to-read && ! C_2.read>

<link, neutral(C_1,C_2) = C_2.ready-to-read && C_2.read>

<link, bad(C_1,C_2) = ! C_2.ready-to-read>

The AHA adaptive engine translates these link attributes to colors for link anchors. The HTML header of every page generated by AHA contains a definition of a style sheet that ties colors to

link classes.

The above examples do not only show that AHA constructs can be represented by the AHAM reference model, but also that only a relatively small fraction of AHAM is needed to express the limited functionality of AHA version 1.0. AHAM suggests many possible future extensions, including:

- In AHA it is not yet possible to define composite concepts that do not correspond to a single page. A composite concept can represent the union of a number of other concepts and thus greatly simplify the rules that depend on a commonly used set of (prerequisite) pages.

- In AHA a user's knowledge about a concept is 0 or 1. When a user reads a page for which she is not ready, her knowledge of the corresponding concept remains 0. By using real numbers between 0 and 1 we could register that her knowledge about a still undesired concept is already 0.5 for instance.

The representation of AHA in AHAM also uncovers the main reason why authoring adaptive hypermedia applications for the AHA system is difficult: all different conceptual parts of the AHAM representation are embedded inside the HTML source for the pages:

- The fragments that make up a page are all put together in one HTML file. This makes it hard or impossible to implement sorting as a content-adaptation method.

- The association between a page and its concept(s) is embedded in the HTML file (as an HTML comment).

- Concept relationships are not explicitly available, except for relationships of the type *link*. These relationships are embedded in the HTML file (as anchor tags). As a result it is difficult for the author to get an overview of how concepts are linked together in a "navigational" way.

- There are some kind of pedagogical rules, but these again are stored within the HTML files. The "assignment" of a Boolean value to the "ready-to-read" predicate for instance is based upon the "requires" HTML-comment inside the HTML file.

- The explicit pedagogical rules that are used to determine whether fragments are "ready-to-read" are also embedded in the HTML file as comments.

- AHA does not offer a way to override the default, predefined pedagogical rules. For instance, it is not possible to write a rule that may set the status of a certain link to "good" depending on another condition than that of the "requires" comment of the destination of the link.

- AHA does not offer a difference between "read" and "reading". Knowledge about concepts becomes 1 when requesting a page and "C.read" becomes true at the same time as "C.reading". It is not possible to specify a rule that can be used for doing content-adaptation differently when a page is revisited.

It is clear that future work on AHA will first and foremost be aimed at separating the elements of the domain model from each other (like separating concepts and concept relationships from pages) and at separating the teaching model from the domain model. The next section draws up a sketch of how to build and structure an authoring system that closely follows the decomposition of adaptive hypermedia application into the four components of AHAM.

5. Authoring

One of the most important purposes of using AHAM is to improve the support for authoring. AHAM is designed to make it easy to develop tools for authoring support. On the other hand AHAM supports the exchange of user models between different AHS. In this section we concentrate on the support for authoring. We look at the way in which the concepts from AHAM help an author to design an adaptive hypermedia application.

5.1 Modular authoring-tool design

Authoring an adaptive hypermedia application starts by describing the *concept space*. Concept relationships like *part of* and *prerequisite* suggest a graphical tool for designing this part of the domain model. A DAG (directed acyclic graph) representation for *part of* relationships can be used, because cycles in these relationships do not make sense. *Prerequisite*, *inhibitor* and similar relationships can be drawn using a graph-drawing tool as well. *Link* relationships are also drawn using a graph-drawing tool. So different types of relationships all result in separate graph representations.

A table can be used to represent the relationships between fragments and concepts. Another table may represent how fragments are combined into pages, and name the concepts tied to pages.

Creating an authoring tool for the *teaching model* is more difficult. An AHS may offer a rich set of generic pedagogical rules. A forms interface can be used to configure these rules, for instance to indicate how much of a concept must be known in order to consider it sufficiently well known to satisfy a *prerequisite* relationship, or to indicate how knowledge of concepts adds up towards knowledge about a composite concept. As such a teaching model authoring tool is simple enough. The difficult part is the tool for creating specific pedagogical rules. Such rules represent ad-hoc deviations from generic rules. Experience with the applications realized using the AHA system [11] has shown that specific pedagogical rules are rare. In AHA such rules are implemented as "if statements" embedded inside HTML files as structured comments. Separating these rules from the HTML source files requires that the fragments inside the page be given an object id, an id which is then used in the specific rules. It is not yet clear whether this separation will actually make authoring easier. The design of authoring tools for the teaching model remains a subject for future research.

5.2 User modeling in adaptive hypermedia

The definition of adaptive hypermedia states which reflect some features of the user are reflected in the user model. This model is applied to adapt various visible aspects of the system to the user. Brusilovsky [1] states which aspects of the user can be taken into account when providing adaptation. Generally, there are five features which are used by existing AHS:

- knowledge
- user goals
- background
- hyperspace experience
- preference

Almost all adaptive presentation techniques rely on the user's knowledge as a source of adaptation. A system has to recognize the changes in the user's knowledge state and update the user model accordingly. The user's knowledge is often represented by an overlay model which is based on a conceptual structure of the subject domain. Sometimes a simpler stereotype user model is used to represent the user's knowledge. Many efficient adaptation techniques require a rather fine-grained approach. Stereotype models are often not fine-grained enough. Overlay models on the other hand are generally hard to initialize. Acceptable results are achieved by combining stereotype and overlay modeling: stereotype 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 AHAM's definition of user model, it is fairly straightforward how the knowledge state can be represented.

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 the concept-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 stored about the user. User's 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 concept-value pairs can 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 *concept-value pairs*. It is thus not necessary to treat these different kinds of aspects in a different way in adaptive hypermedia applications.

6. Conclusions

Adaptive hypermedia applications use different methods and techniques to implement adaptation of the hypermedia contents and links to the users. In order to design usable adaptive hypermedia applications it is important that authors can model the relevant aspects of an application effectively. In our opinion one of the more important things in an adaptive process is the mechanism for calculating the relevance of linked pages and the contents included in them. This mechanism is the critical factor in improving the quality of adaptive processes. In educational adaptive hypermedia applications, this mechanism is realized using pedagogical rules. In the reference model that we proposed in this paper, we take into account the important role of pedagogical rules in teaching courses through adaptive hypermedia. We proposed a framework

that is built on a strong notion of *concept* in order to provide the author with a more powerful way to express adaptation through the calculation of relevance. Also, it is our belief that the separation of the notions of concept, concept relationship, fragment and page make authoring and understanding adaptive hypermedia applications easier. Experiments to verify this claim will be conducted in the near future.

References

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

[2] Boyle, C. and A.O. Encarnacion. MetaDoc: An Adaptive Hypertext Reading System. User Modeling and User-Adapted Interaction, 4(1), 1994, 1-19. (Reprinted in Adaptive Hypertext and Hypermedia, Kluwer Academic Publishers, 1998, 71-89.)

[3] Brown, P.J. Turning ideas into products: The Guide system. Proceedings of the ACM Hypertext'87 Conference, 1987, 33-40.

[4] Brusilovsky, P., J. Eklund. A Study of User Model Based Link Annotation in Educational Hypermedia. Journal of Universal Computer Science. 4(4), 429-448.

[5] Beaumont, I. User Modeling in the Interactive Anatomy Tutoring System ANATOM-TUTOR. User Modeling and User-Adapted Interaction, 4(1), 1994, 21-45.

[6] Brusilovsky, P., L. Pesin. ISIS-Tutor: An intelligent learning environment for CDS/ISIS users. Proc. of the interdisciplinary workshop on complex learning in computer environments (CLCE94), 1994, 29-33.

[7] Brusilovsky, P., E. Schwarz and T. Weber. A tool for developing adaptive electronic textbooks on WWW. Proceedings of the WebNet'96 Conference, 1996, 64-69.

[8] Calvi, C. A Proficiency-Adapted Framework for Browsing and Information Filtering in Web-Based Educational Systems. Methodological Implications for Language Learning on the WWW. Doctoral thesis, University of Antwerp (UIA), 1998.

[9] Calvi, C., P. De Bra. Proficiency-Adapted Information Browsing and Filtering in Hypermedia Educational Systems. User Modeling and User-Adapted Interaction 7(4), 1997, 257-277.

[10] Conklin, J. Hypertext: An introduction and survey. IEEE Computer, 20(9), 1987, 17-40.

[11] De Bra, P., L. Calvi. AHA: a Generic Adaptive Hypermedia System. 2nd Workshop on Adaptive Hypertext and Hypermedia, 1998, 1-10.

[12] de Rosis, F., N. De Carolis and S. Pizzutilo. User Tailored Hypermedia Explanations. INTERCHI'93 Adjunct Proceedings, 1993, 169-170.

[13] De Young, L. Linking Considered Harmful. Designing and Reading Hyperdocuments. 1st European Conference on Hypertext, 1990, Cambridge University Press, 238-249.

[14] Fisher, G., T. Mastaglio, B. Reeves, and J. Rieman. Minimalist Explanations in Knowledge-Based Systems. 23th Annual Hawaii International Conference on System Sciences, 1990, 309-317.

[15] Hohl, H., H.-D. Böcker, R. Gunzenhäuser. Hypadapter: An Adaptive Hypertext System for Exploratory Learning and Programming. User Modeling and User-Adapted Interaction, 6(2-3), 1996, 131-156.. (Reprinted in Adaptive Hypertext and Hypermedia, Kluwer Academic Publishers, 1998, 117-142.)

[16] Kobsa, A., D. Müller and A. Nill. KN-AHS: An Adaptive Hypertext Client of the User Modeling System BGP-MS. Fourth International Conference on User Modeling, 1994, 31-36.

[17] Pilar da Silva, D. Concepts and documents for adaptive educational hypermedia: a model and a prototype. 2nd Workshop on Adaptive Hypertext and Hypermedia, 1998, 33-40.

[18] Thüring, M., J. Haake, J., Hannemann. What's ELIZA doing in the Chinese room? Incoherent hyperdocuments - and how to avoid them. ACM Conference on Hypertext, 1991, 161-177.

URL References

IShype is a "kiosk" information system at the TUE: http://wwwis.win.tue.nl/IShype/

2L690 is the course on "Hypermedia Structures and Systems", offered only through distance education: http://wwwis.win.tue.nl/2L690/

2M350 is the course on "Graphical User-Interfaces", notes to use with lectures and assignments: http://wwwis.win.tue.nl/2M350/