Journal of Web Engineering, Vol. 2, No.1&2 (2003) 074-089

LINK-INDEPENDENT NAVIGATION SUPPORT IN WEB-BASED ADAPTIVE HYPERMEDIA

PAUL DE BRA
Eindhoven University of Technology
Department of Mathematics and Computing Science
PO Box 513, 5600 MB Eindhoven, the Netherlands
debra@win.tue.nl

Received December 29, 2002
Revised July 24, 2003

Many websites offer their users a lot of freedom to navigate through a large hyperspace. Some sites offer navigation or orientation support in the form of (complete or partial) site maps or guided tours. Some sites also use *adaptive* hypermedia techniques such as *link annotation* to help users find their way, based on an individual or group user model. In such systems the navigation support is often tied to the existing link structure. In this paper we discuss how websites can also offer adaptive navigation and orientation support like site maps and guided tours that are independent of the underlying link structure of the website. In particular, we show how the AHAM model, introduced in [6], can represent such adaptive global or local orientation support. To this end we define Link-Independent Navigation Support (LINS) that provides the user a better understandable navigation environment and a strong connection among pages at different abstraction levels in hyperspace. AHAM provides a design platform to define all kinds of relationship graphs, called abstract views in this paper. Abstract views describe connectivity among concepts independently from the basic link structure of the underlying hyperspace, and LINS is based on abstract views.

Key words: Adaptive hypermedia, User modeling, Navigation support, Hypermedia reference model, Adaptation rules
Communicated by: M Gaedke & G Rossi

1 Introduction

The introduction of World Wide Web has made hypermedia the preferred paradigm for user-driven access to information. Websites typically offer users a lot of freedom to navigate through a large hyperspace. Unfortunately, this rich link structure of hypermedia applications may cause some usability problems:

- Users may be interested in one or more topics on which the website offers information (or services or products). A typical website tries to offer access to all its information, through a rich link structure that is the same for all users. Many company websites try to offer information for employees as well as visitors. A university website has information for students, for faculty and for visitors. As a result, a user looking for information on a specific topic is confronted with links to pages that either do not deal with that topic or that deal with the topic but are intended for a different audience. It is difficult for an author (or website designer) to design the link structure in such a way that links to irrelevant information do not appear throughout the browsing experience. Our own university's website (www.tue.nl, accessed July, 2003) for instance distinguishes students, faculty and visitors for some information, and only on the "central" server. After a choice for student, faculty member or visitor at the top level, the user never sees a page intended for another audience. To achieve this result on the static website a lot of redundancy was introduced. There is also general information for everyone, which again has no overlap with the group-specific information, thereby introducing even more redundancy. Also, when linking through to another server the distinction between user groups is dropped. Suddenly the student may reach pages intended for faculty members or visitors, or vice versa. No indication of the intended audience is given on any of the pages. Some sites try to compensate for the navigation difficulties by offering a sitemap. However, the sitemap just displays the entire site structure, and thus includes all the irrelevant pages, thereby perhaps making it even more difficult to find the pages that deal with the topic a user is interested in. In order to guide users with a specific information need through a site (without introducing a lot of redundancy) the link structure must be adapted to the user's needs. And in order to provide a useful overview of the website's structure, a personalized, partial sitemap is needed as well.
- Navigation in ways the author did not anticipate also causes *comprehension problems* for the user: for every page the author must take into account what foreknowledge the user has when accessing that page. In order to do so the author must at least consider all possible paths that lead to the current page. This is clearly an impossible authoring task because there are more ways to reach a page than any (human) author can foresee. In a traditional hypermedia application a page is always presented in the same way. This may result in users visiting pages containing redundant information and pages that they cannot fully understand because they lack some expected foreknowledge. The website should really select the pieces of information that are shown on a page, based on a history of which pages the user has seen before. As an example of how not to do it, on our university's website a "facts and figures" page, is only two clicks away from the top, in the general information for everyone, and it claims that "the TU/e supervises two of the six top research schools and one of the four leading technological institutes". It claims that "the TU/e participates in European university networks CESAER, Santander and CLUSTER", etc. None of these terms are explained (and not even put between quotes to indicate that they are defined terms and not plain English words), and for none of the terms a link to a page with an explanation is provided. Clearly, for a casual visitor this page simply contains meaningless promotional blabber. We are confident that our university intended for this page to provide valuable information indicating the high quality of our research and the national and international recognition, but the page fails miserably at conveying this message.

Adaptive hypermedia systems (AHS) in general, and adaptive websites in particular aim at overcoming the navigation and comprehension problems by providing *adaptive navigation support* and *adaptive content*. The adaptation (or personalization) is based on a user model that represents relevant aspects of the user such as preferences, knowledge and interests. We focus on simple Web-based systems that can only gather information about the user by observing the *browsing* behavior of that user. Each time the user "clicks" on a link the system uses the selected link to update the user model and to adapt the presentation accordingly. Also, in this paper we concentrate on adaptive navigation support, not on adaptive content. More specifically we study navigation support that is independent of the website's fixed link structure. (The topics we do not describe can be found in earlier papers, like [6] and [12].)

The basic idea with adaptive navigation support is to adapt the link structure in such a way that the user is guided towards interesting, relevant information, and kept away from non-relevant information. Typical link-adaptation tries to simplify the rich link structure to reduce orientation problems, while maintaining a sufficient level of navigational freedom, typical of hypermedia systems. Brusilovsky [1] mentions the following link-adaptation methods:

- Global guidance
- Local guidance
- Global orientation support
- Local orientation support

Guidance can be provided in hypermedia applications where users have some goal in terms of information they want 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, like with *guided tours*. This is for instance useful in educational websites. 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 a meaningful reading order. *Local guidance* means that the system suggests the next step to take, for instance through a "next" or "continue" button. When the user deviates from the suggested path, new suggested continuations are generated on the fly.

Orientation support means that the system presents an overview of the global (link) structure of the hyperspace (*global orientation support*), or of a part thereof that is near the user's "current" position (*local orientation support*). The system also indicates that "current" position in this structure. Brusilovsky in fact also mentions "*personalized goal-oriented views*" as an adaptation method. We regard this as an extension of global orientation support: Each "view" may be a map or list of links to all pages or sub-parts of the whole hyperspace that are relevant for a particular working goal. Like with guided tours, the link structure presented in these views may be generated for the specific user, and not part of the "fixed" link structure of the website. In this paper we are particularly interested in such adaptively generated link structures, which we shall call *abstract views*.

In previous research we have shown that our AHAM model [6] can be used to describe link adaptation, as used in systems that guide users towards interesting information by changing the presentation of link anchors. In general however, adaptation of the existing link structure alone is not enough to solve all users' navigation and orientation problems:

- It may not be possible to select a guided tour consisting of existing links, and of only pages that are interested for a given user, because the interesting pages may not form a connected subgraph in the whole link structure.

- It may take too many steps (possibly going through uninteresting pages) to guide the user to the page(s) that deal with the topic the user is interested in.
- The user (may wish to but) cannot visit the interesting pages in any order that the user prefers.

To improve the basic adaptive navigation support in AHS, we propose *link-independent navigation support* to supplement the above cases. Link-independent navigation support (or LINS) aims to guide the user through hyperspace through an abstract view based on the user's preferences. It uses extra connections among concepts through certain relationships. Link adaptation techniques, e.g. *adaptive link annotation* can be used in addition, to provide guidance even within the *abstract view*. LINS can of course also be combined with the existing link structure, possibly also made adaptive, to provide an even richer personalized navigation environment.

While LINS could be investigated at a more general level, we concentrate on its use in AHS that can be described in the AHAM model. Such systems maintain a user model by tracing the user's browsing behavior and perform adaptation based on that user model. AHAM is well suited to describe many Web-based AHS.

This paper is structured as follows: in Section 2 we briefly review the AHAM reference model, thereby concentrating on the parts that are needed to describe adaptation functionality at an abstract level. Section 3 discusses how to provide LINS through the concept of *abstract views*. It also describes the generation and adaptation of navigation and orientation support (based on views) in the database-like language we introduced in [12]. Section 4 draws conclusions and provides an outlook into our future work.

2 AHAM, a Dexter-based reference model

Many adaptive hypermedia systems share parts of their architecture. Just like the Dexter model [8][9] tried to capture the facilities offered by hypermedia systems of its time (and of potential future systems), AHAM [6], (for Adaptive Hypermedia Application Model) describes the common architecture of adaptive hypermedia systems. Part of this common architecture is typical for Web applications: their event-driven nature, where each page access results in a user-model update and an adaptive presentation. AHAM's overall structure is an extension of that Dexter model. According to AHAM each adaptive hypermedia application is based on three main parts:

- The application must be based on a *domain model*, describing how the information content of the application or "hyper-document" is structured (using concepts and pages).
- The system must construct and maintain a fine-grained *user model* that represents a user's preferences, knowledge, goals, navigation history and other relevant aspects.
- 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*. An AHS itself may offer built-in rules for common adaptation aspects. This reduces the author's task of providing such rules. In fact, many AHS do not offer an adaptation rule language; the way in which the user model is updated and the presentation adapted is then completely predefined.

The division into a *domain model* (DM), *user model* (UM) and *adaptation model* (AM) 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.

Modeling an existing AHS in AHAM may not be straightforward because AHAM requires these parts to be made explicit, and the adaptive behavior to be described using adaptation rules. However, using AHAM enables us to clearly describe how an AHS works, how different AHS compare, and also how to design new and more powerful AHS.

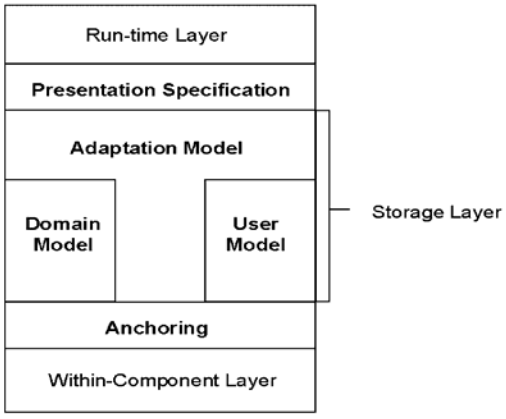

Figure 1. Structure of adaptive hypermedia applications.

The AHS consists not only of the three sub-models (mentioned above) but also of an adaptation engine (AE). The AE describes implementation dependent aspects of the AHS. In previous work [12] we described design issues for a general-purpose AE, and defined AHAM CA-rules (condition-action rules) to illustrate how sets of rules work together. We will use these rules to describe the generation of and browsing through LINS in Section 3.

Figure 1 shows the overall structure of an adaptive hypermedia application in the AHAM model. The figure has been made to resemble the architecture of a hypermedia application as expressed in the Dexter Model [8][9].

2.1 The Domain Model

The domain model of an adaptive hypermedia application consists of *concepts* and *concept relationships*. Concepts are objects with a unique object identifier, and a structure that includes attribute-value pairs and a sequence of anchors.

A concept represents 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 thus cannot be adapted). Its attribute and anchor values belong to the "Within-Component Layer" and are therefore considered implementation dependent and not described in the model.
- A *composite concept* has a sequence of children (sub-concepts) and a constructor function that describes how the children belong together. The children of a composite concept are either all atomic concepts or all composite concepts. A composite concept with (only) atomic children is called a *page*. The other (higher-level) concepts are called *abstract concepts*.

The composite concept hierarchy must form a DAG (directed acyclic graph). Also, every atomic concept must be included in one or more composite concepts. Figure 2 illustrates a part of a concept hierarchy.

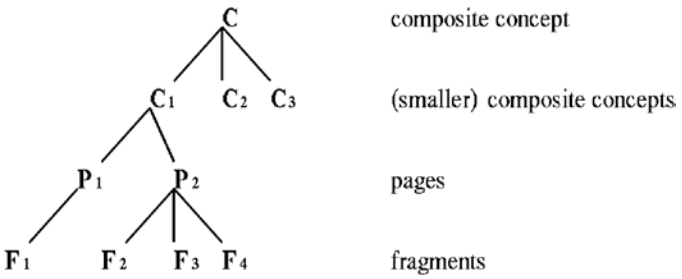

Figure 2: Part of a concept hierarchy.

A concept relationship is an object (with a unique identifier and attribute-value pairs) that relates a sequence of two or more concepts. Each concept relationship has a type. The most common type is the hypertext link. In AHAM we consider other types of relationships as well, which play a role in the adaptation, e.g. the type prerequisite. When a concept C_1 is a prerequisite for C_2 it means that the user "should" know C_1 before reading about C_2. This does not imply 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. Through link adaptation the "desirability" of a link will be made clear to the user. Figure 3 shows a small set of concepts associated to one another by three types of (binary) concept relationships: prerequisite, inhibit, and link.

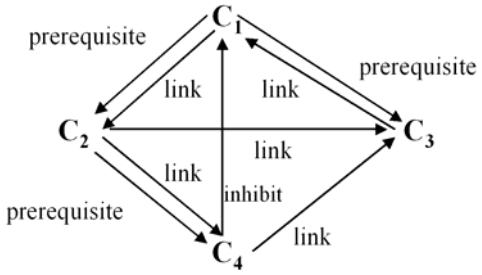

Figure 3: Example concept relationship structure.

Apart from the "implicit" relationship type and set of relationships that form the concept hierarchy an AHS need not contain or support any other relationships. AHAM can thus also represent applications without traditional hypertext links (like e.g. in spatial hypertext [11]). A relationship graph defines a certain view to reflect a relationship among the set of pages. In Section 3 we will use relationship graphs to define abstract views to provide link-independent navigation support to supplement the basic adaptive navigation support in AHS.

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

2.2 The User Model

A user model consists of named entities for which we store a number of attribute-value pairs. For each user the AHS maintains a *table-like* structure, in which for each concept in the DM the attribute values for that concept are stored. Because there can be many relationships between *abstract concepts* and *concrete* content elements like fragments and pages, a user model may contain many attributes per concept to indicate how the user relates to the concept. Typical attributes would be *knowledge level* (e.g. in educational applications) and *interest* (e.g. in encyclopedia, museum sites, or on-line mail-order catalogs). The user model may also store information about what a user has *read* about a concept, and for *how long* or *how long ago*, the *quantity* of an item in a shopping basket, etc. Concepts can furthermore be used (some might say abused) to represent "global" user aspects such as preferences, goals, background, hyperspace experience, or a (stereotypical) classification like student, employee, visitor, etc. In the AHA! system for instance [5] a concept "personal" is used to store information about the user that is unrelated to the actual adaptive application. For the AHS or the AHAM model the true meaning of concepts is irrelevant.

In the sequel we will always consider UM as being the user model for a single user. In this paper we do not discuss adaptation to group behavior.

2.3 The Adaptation Model

The AHAM model targets adaptive hypermedia applications that follow *the request-response* paradigm that is typical for the Web. The interaction with the system is described through events generated by the user (or by an external source). Each event triggers user model updates and results in an adaptive presentation. In [12] we introduced a database-like language to express the effects of user actions as *condition-action rules* (AHAM CA-rules). This implies that we do not explicitly model events as events, but as updates to attributes that trigger rules. Accessing a web-page for instance will result in a Boolean "access" attribute of the page (in the user model) to become *true*. The small example below illustrates the structure of these rules. A syntax description can be found in [12].

C: **select** P.access
A: **update** F.pres := "show"
where F **in** P.fragments **and** F.relevance = "recommended"

In this example we first see that the condition for this rule is that P.access has become true for some page P. When this happens (and because there is no additional **where** clause in the condition) the *action* is executed. In the action we look at fragments F of page P. If a fragment is marked as "recommended" then that fragment will be shown. This is indicated as a presentation specification and represented as a "pres" attribute of the fragment. Note that because abstract concepts, pages and fragments appear in the domain model as well as the user model (but with different attributes) we take the liberty of mixing both in the adaptation rules. E.g. our notation F **in** P.fragments in fact refers to the concept hierarchy from the domain model, whereas P.access and F.relevance for instance refer to attributes in the user model.

Note that the AHAM CA-rule language is just a vehicle for describing how an AHS should perform user model updates and adaptation. It does not imply that we require an AHS to use such a language. Even when an AHS has only a built-in behavior, we can still describe this using the AHAM CA-rule language. Also, we partition adaptation rules into *phases* to indicate that certain rules must always be executed before certain other rules. The phases include IU, the initialization of the user model, UU-pre, the user model updates that are performed before generating the presentation, GA, the

generation of the adaptation, and UU-post, the user model updates that come after the presentation. The phases are a convenience for ensuring that the execution of the rules has desirable properties such as *termination* and *confluence*, as discussed in [12].

2.4 The Adaptation Engine

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

- It offers generic page selectors and constructors. For each composite concept the corresponding selector is used to determine which page(s) to display when the user follows a link to that composite concept. For each page a constructor is used for building the adaptive presentation of that page (out of its fragments). Page constructors allow for dynamic content like a ranked list of links.
- It optionally offers a (very simple programming) language for describing new page selectors and constructors. For instance, in AHA! [5] a page constructor consists of simple commands for the conditional inclusion of fragments (in version 2.0). Alternatively the adaptation rule language of AHA! can be used to do conditional selection of objects to include in a page (in version 3.0).
- It performs adaptation by executing the page selectors and constructors. This means selecting a page, selecting fragments, organizing and presenting them in a specific way, etc. Adaptation also involves manipulating link anchors depending on the state of the link (like enabled, disabled and hidden).
- It updates the user model (instance) each time the user visits a page. The engine will change some attribute values for each atomic concept of displayed fragments in a page, of the page as a whole and possibly of some other (composite) concepts as well (all depending on the adaptation rules).

The adaptation engine thus provides the implementation-dependent aspects, while DM, UM, and AM describe the information and adaptation at the conceptual, implementation independent level. Note that DM, UM and AM together thus do not describe the complete behavior of an AHS. The same set of *adaptation rules* may result in a different presentation depending on the *execution model* of the adaptation engine.

3 Link-independent navigation

In adaptive hypermedia applications, an author defines abstract relations between concepts and between concepts and pages, in order to enable the system to automatically generate user-guidance. This abstract information is normally hidden from the user. Users benefit from these (invisible) relationships through adaptive navigation support based on these relationships. A typical example is adaptive link annotation based on prerequisites, as exemplified by applications of the Interbook system [2] or AHA! [5]. In such applications the navigation support augments the existing link structure. In this paper we describe *link-independent navigation support* (LINS):

- It provides navigation support based on abstract views: LINS based on abstract views is a supplement to navigation support based on the basic link structure. It adds extra ways for users to navigate through hyperspace more efficiently.

- It is independent from the basic link structure: The hyperspace link structure may be well connected if the author designs it carefully, but in most cases the hyperspace is large, making this hard to control. Every link structure is a compromise between the aim to offer the user a small set of interesting links to choose from and the aim to offer short paths to all the information, thus requiring many links on each page. If navigation support is built on the link structure of the hyperspace, it is obvious that it is limited to the defined connectivity. No additional information or links can be offered than those that are already visible. Certain pages cannot be reached (directly) because no link exists to those pages. We suggest that adaptive navigation support should at least partially be independent of the link structure, so that it can offer additional connectivity. This additional connection is based on semantic relationships among concepts described in the DM. Decoupling the information content from the link structure is not a new idea: it has been studied extensively in the Open Hypermedia community. Recently the adaptive selection of linkbases has also been studied [4][7]. Open hypermedia uses virtual links and virtual anchors, it has a linkbase storing the context dependent links. It extracts and analyzes a document's spatial context as a view from a user's perspective. Based on the context, it augments the virtual link.
- An abstract view is chosen based on user preferences: Navigation support is more effective if the user has the possibility to personalize it. In this way the LINS is generated towards a user's goal or interest, or based directly on prerequisite relationships for instance.
- It is well linked so that users can easily go where they want to go: We argue that adaptive navigation support should provide not only a suggested path and orientation map, but also a *connection* from the navigation support to the real pages, so that users can go where they want to from the navigation support. For guidance it is suitable to have a list of suggested page links, with or without order, or even direct guidance through a next or continue button. For orientation support, the system should have something like a graph in which nodes represent a page link or a concept link. The graph may be a tree if the relationship between nodes has a tree structure. The graph may be a list if the relationship between the nodes is linearly ordered. Lists and trees are commonly used index methods in our daily lives and are also common in structure-based navigation support in Websites. (Typically a list or a tree structure is presented in a navigation column on the left side of a webpage.)
- The LINS itself is adaptive: For global (or local) guidance, the suggestion list should have content adaptation by including links to parts of pages selected according to the user's global (or local) goal. For different goals, global (or local) guidance can provide a different list or navigation buttons. It should also have link adaptation by showing how relevant the links are. Annotation and order are commonly used techniques to show relevance. For global (or local) orientation support, the graph needs link adaptation, to indicate the relevance of links. This can be done using link annotation, but also through link hiding to hide or remove links to irrelevant concepts or pages. Adaptation in navigation support should be consistent with adaptation in a basic page. This means that the same page link should have the same link adaptation suggestion, whether it appears in a map or on a page.

Section 3.1 explains how to apply these ideas within the AHAM framework. Section 3.2 explains how to update the user model for generating the LINS. Section 3.3 explains how to generate adaptive navigation support based on abstract views. Section 3.4 explains how to infer cross-references for each page that has implied connections to other abstract concepts. Section 3.5 explains how to show the content of the nodes for link-independent navigation support.

3.1 Defining abstract views of hyperspace

The link structure is not the only information that can be used to generate navigation support. In fact in some systems the author must define some abstract relational requirement among pages to provide the navigation support. AHAM provides a platform to define these relational requirements as relationship graphs, called abstract views in this paper. Different users may want to use different views to navigate through hyperspace. In figure 4, e.g., user A visits the hyperspace through views P or Q, and user C visits the hyperspace through views R or S. User B, however, visits the hyperspace using no view, i.e. she uses the basic link structure. Link-independent navigation support can be tailored to the user's preferences through the selection of an appropriate view. This corresponds to the selection of linkbases as done in [4] and [7].

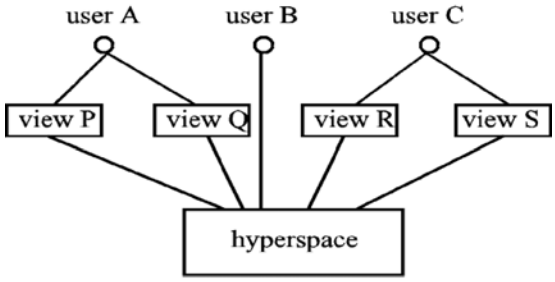

Figure 4: Abstract views of hyperspace.

We consider views separately when we provide link-independent navigation support. We don't consider the combination of several views in this paper. Each view corresponds to the relationship graph of a specific relationship type. A view gives a certain perspective on a set of concepts connected to a set of pages. It consists of two parts: one consists of concept nodes that represent abstract concepts, and edges that represent the same type of relationships among concepts; another part consist of page nodes and links from the parent concepts in the concept hierarchy. Figure 5, e.g., could be an index view describing a (hyper)book, or a structure of prerequisite relationships. $R(C_1, C_2)$ describes that relationship between C_1 and C_2. $L(C_4, P_1)$ describes a link from C_4 to P_1.

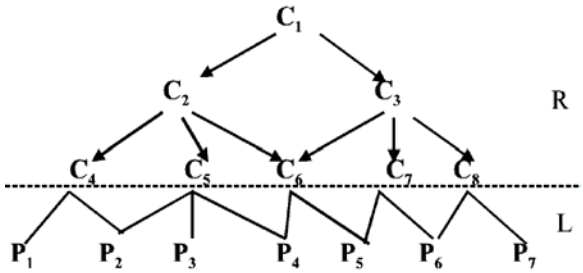

Figure 5: An example of an abstract view of hyperspace.

The author can build navigation support based on views to suggest to the user (possibly several) ways to explore the subject. While AHAM provides a mechanism to define relationships and a rule-

system to make use of them, no assumptions are made as to the semantics of any relationship. This is the author's responsibility. We summarize three steps to define a view:

- Define a concept to represent the view graph as a whole. For convenience (as we shall see later) it is assumed to have an attribute that we shall call "children", to store all nodes in the view, both concept nodes and page nodes.
- Define each concept node in the view. E.g., in Figure 5 the nodes above the dotted line represent abstract concepts, while the nodes below represent physical pages. We assume page nodes have been defined already. We assume that every page has an attribute "fragments" to store its fragments. The fragments are not shown in Figure 5. A concept node has attributes "description" to represent its description, "children" to represent sub-concepts, and "type" to represent the type of relationship graph it is in. Every concept in a view is defined as a page that consists of its description, and links to either its sub-concepts for non-terminal concepts or to the actual pages for terminal concepts.
- Define a relationship for each edge in the view. All relationships above the dotted line in the view have the same "type". All relationships bellow the dot line have the same "type", namely "link".

Relationship graphs are not required to be *directed acyclic graphs* (DAG) as in Figure 5. However, for many types of relationships cycles are a sign of bad design, e.g., with prerequisite or inhibit relationships. Views play an important role in maintaining the user model and generating basic adaptation in AHS [12]. In Interbook [2] for instance there is a "teach me" button that is used to generate a set of links to all pages that contribute knowledge about a certain concept. This is a nice way to use views, and in fact offers link-independent navigation support. In subsection 3.3 we focus on how to generate LINS based on views and the user model.

3.2 Updating the User Model

We briefly illustrate how the AHAM CA-rules are used to perform user model updates, in which the concept relationships play a role. Adaptation is normally based on *relevance* of or *recommendations* for concepts or pages. This is in turn deduced from aspects like *interest* or *knowledge* (which must exceed some *threshold* to be considered sufficient to be taken into account). Below we give a possible rule that decides whether a concept should be recommended based on whether the user has enough knowledge about all prerequisite concepts.

C: **select** C_2.knowledge
  where C_2.knowledge >= C_2.threshold
A: **update** C_1.relevance := "recommended"
  where Prerequisite(C_2, C_1) **and**
    not exists (**select** C_3
      where Prerequisite(C_3, C_1) **and** C_3.knowledge < C_3.threshold)

The rule is triggered when the knowledge of concept C_2 is changed and when that knowledge then matches or exceeds the required knowledge threshold for C_2. The action of the rule sets the relevance value for C_1 to "recommended" if there are no unsatisfied prerequisites left for C_1. (For efficiency reasons only concepts C_1 are considered for which C_2 is a prerequisite. Other concepts cannot be influenced by the knowledge change for C_2.)

In order to describe the user model updates, changes to relevance of pages, and also the resulting adaptation one needs many rules. We don't describe them here, not just because of the size restrictions for this paper, but also because every AHS has a different behavior and thus also a different description using AHAM CA-rules.

3.3 Generating link-independent navigation support

With several examples we will illustrate how AHAM CA-rules can be used to perform adaptation for navigation support pages. In Section 2.3 we indicated that rules are divided into phases to simplify scheduling their execution. The rules we describe here all belong in the GA (Generate Adaptation) phase. This means that when we reach this phase user model updates, like deciding which concepts or pages should be recommended, have already been performed (in the UU-pre phase). Assume G represents a view (for a relationship type G). We can use this view to provide four types of navigation support: "global orientation", "local orientation", "global guidance" and "local guidance". For simplicity we assume that attributes "g-o", "l-o", "g-g" and "l-g" of G are used to trigger the generation of the different types of navigation support. We shall ignore the user model updates needed to select which type of navigation support the user wants.

"Global orientation support" based on view G means that the system presents an overview of the whole (relationship) structure of the view G. It is generated through the rules below. In the view we use colors as an annotation technique for showing the relevance of page nodes. We have taken the "traffic light metaphor" in which green represents *recommended*, red represents *not-recommended*, and yellow represents *not-interesting*.

C: **select** G.g-o
A: **update** F.pres := "green"
where F **in** G.children **and** F.relevance = "recommended"

C: **select** G.g-o
A: **update** F.pres := "red"
where F **in** G.children **and** F.relevance = "not-recommended"

C: **select** G.g-o
A: **update** F.pres := "yellow"
where F **in** G.children **and** F.relevance = "not-interesting"

Sometimes it is useful to show the user's current page in "global orientation support". This rule shows the "current page" in another color, e.g. black, in the orientation support. We can create an attribute "current" of G (or a global variable), and update it in the UU-pre phase with the rule:

C: **select** P.access
A: **update** G.current := P

The above rules then need to be modified so they check that F is not the current page, and we add a rule:

C: **select** G.g-o
A: **update** F.pres := "black"
where F **in** G.children **and** G.current = F

"Local orientation support" based on view G means that the system presents only a part of the (relationship) structure of the view G. The scope for local orientation support may be all ancestors of the current page (up to the root of the view if that exists, i.e. if the view is acyclic and thus has a root) or only concepts and pages that are a fixed number of steps (relationships or links) away from the current page. Ancestors, parents, children, brothers, etc., can all be easily expressed using rules. We omit this to keep the paper short (and readable). A possible rule for "local orientation support" based on view G could be:

C: **select** G.l-o
A: **update** F.pres := "green"
  where F **in** G.children **and** (Brother(F, G.current) **or** Ancestor(F, G.current))
    and F.relevance = "recommended"

Rules for other levels of relevance are similar of course. The above rule will present sibling pages of the current page and ancestor concepts of the current page using the color "green" if these pages or concepts are "recommended".

"Global guidance" based on view G means that the system presents a list of pages to read, in a certain order. (Abstract concepts are not pages one can read and thus can be part of an overview but not of some kind of guided tour.) The suggested order of the pages depends on relevance. Expressing this is easiest using an "external" function, although it could be done through a series of rules that together perform sorting based on relevance. (If relevance has only three values the function is simple, but some AHS may have a numerical relevance value requiring complete sorting.) We will ignore the sorting issue here.

C: **select** G.g-g
A: **update** F.pres := "green"
  where F **in** G.children **and** F.relevance = "recommended"
    and exists (**select** C
      where L(C, F))

Note that L(C, F) expresses that F is a page. Rules for other levels of relevance are similar.

"Local guidance" based on view G means that the system presents only the next step or steps to take, based on the view. We give a simplified rule below that assumes the concepts linked to pages provide a sequence through relationships. In general however one may need to go higher up in the view.

C: **select** G.l-g
A: **update** F.pres := "green"
  where F **in** G.children **and** F.relevance = "recommended"
    and exists (**select** C_1
      where L(C_1, F) **and** (L(C_1, G.current) **or**
        exists (**select** C_2
          where G(C_2, C_1) **and** L(C_2, G.current))))

This rule shows that recommended pages that are linked to the same concept as the current page, or to a concept that is one step away (in the current view) will be shown as recommended (with a green link).

Once more, rules for other levels of relevance are similar.

3.4 Inferring cross-references for each page

The system can infer a *cross-reference* relationship for each page to all its connected abstract concepts defined in all views. These cross-references can be shown as a part of original page or in a separate page depending on the system. Cross-references can be very useful in IR systems when users do not know what they want before they find something interesting in the current page. A page may connect with several abstract concepts in one view; it may also connect with several abstract concepts in different views. A cross-reference contains a list of links that can be grouped according to the type of view, for example, so that the user can easily find related concepts without getting disoriented in the hyperspace. The cross-references can be adaptive also in the sense that they reflect the knowledge state for every referenced concept. The author gets these cross-references for free because the system can infer them from the abstract views defined in the DM.

3.5 Generating content presentation for nodes in LINS

In the previous sections we described how to generate link-independent navigation support through AHAM CA-rules. Now we briefly discuss what the system should present when a user "clicks" on a concept link in LINS, e.g. in the view presented as "global orientation support". (These rules also belong in the GA phase.)

When users "click" on a node of a concept link in LINS, the system can generate a page that contains some description about the concept, and a list of links to its sub-concepts down to the page level. The links in this page will be presented using color metaphors as used for adaptive annotation to show the knowledge state for all links to pages and nodes.

Here is a rule example that says when a user "clicks" on a concept node in global orientation support or local orientation support, the system shows the fragment of description of this node in the page. Such support only works if a "description" attribute is available for abstract concepts.

C: **select** C.access
A: **update** F.pres := "show"
  where C.description = F

The following rule describe that when a user "clicks" on a concept node, the system shows all the sub-concept links of this concept using the color metaphor as for adaptation annotation.

C: **select** C.access
A: **update** F.pres := "green"
  where (G(C, F) **or** L(C, F)) **and** F.relevance = "recommended"

Again, the rules for other relevance levels are similar. What the above two rules do not yet express is how the description and the links to sub-concepts will be combined into a single page to present to the user.

4 Conclusions and future work

This paper proposed the idea of providing link-independent navigation support (LINS) based on abstract views in AHS. LINS provides users with extra ways to browsing through a hyperspace by using abstract relationships. It is a supplement to adaptive navigation support based on the basic link structure of hyperspace. We described how to generate and use LINS in AHAM, a general reference model for AHS. This also shows that AHAM is capable of expressing LINS without the need for

extensions to the model. By using LINS users can browse hyperspace in a better understandable navigation environment.

Our method assumes that authors define the abstract views before the system generates the LINS. The LINS is very useful in educational hypermedia for instance because teachers would provide the abstract views for the courses. LINS is also useful in information sources like corporate websites. Returning to the TU/e website and its meaningless "facts and figures" page, in a *visitor view* a set of relationships can be used to generate links to descriptions of the concepts "top research school" and "leading technological institute" (which have a defined meaning in the Netherlands and are not just English expressions), whereas in a *faculty view* different relationships can be considered, perhaps leading to the generation of links to a list of these schools and institutes. The former view can be based on *prerequisite* relationships (indicating the need to study the definitions of the terms) whereas the latter can be based on *interest* relationships (indicating that getting to the facts and figures page means the user may be interested in more information about these schools and institutes).

We think that LINS will turn out to be less practical in large information systems (or information sources that extract data from large databases), because these sources do not have a clear notion of an "author". At the TU/e we have a different line of research (the HERA project [10]) that considers the generation of navigation support in large database-based Web information systems. In that research we study the possibilities to automatic generate abstract views.

Our future work is twofold: first we wish to continue to show that AHAM can express the functionality of many existing AHS and that it is a useful framework to compare adaptive applications and to describe their functionality in a unified way. In [13] we already have a description of Interbook [2] and AHA! [5]. These descriptions have taught us which features AHA! was missing, compared to Interbook. In [3] we have presented the necessary extensions to AHA! and a compiler from Interbook to AHA!. Thanks to a grant of the NLnet Foundation we are extending AHA! further, to incorporate more ideas from AHAM. The conditional selection of information fragments (based on concept relationships) is already present in version 3.0, but the automatic generation of links (and more specifically of link anchors) is still future work. Recent work in the area of open hypermedia systems [4][7] already comes closer to providing LINS.

Acknowledgement

This paper is based on the phd research of Hongjing Wu at the Eindhoven University of Technology, and a paper presented at the WWW2002 conference. For legal reasons she could not be named as co-author of this JWE paper. Part of this work was funded through a grant of the NLnet Foundation, and part by the Minerva project "ADAPT", project 101144-CP-1-2002-NL-MINERVA-MPP.

References

[1]. Brusilovsky, P., "Methods and Techniques of Adaptive Hypermedia". User Modeling and User-Adapted Interaction, 6, pp. 87-129, 1996.

[2]. Brusilovsky, P., Eklund, J., and Schwarz, E., "Web-based education for all: A tool for developing adaptive courseware". Computer Networks and ISDN Systems (Proceedings of Seventh International World Wide Web Conference, pp. 14-18 April 1998) 30 (1-7), pp.291-300, 1998.

[3]. Brusilovsky, P., Santic, T. and De Bra, P., "A Flexible Layout Model for a Web-Based Adaptive Hypermedia Architecture". Proceedings of the Adaptive Hypermedia 2003 Workshop, TU/e CSN 03/04, Budapest, Hungary, pp. 77-86, 2003.

[4]. Bailey, C., EI-Beltagy, S. R. and Hall, W., "Link Augmentation: A Context-Based Approach to Support Adaptive Hypermedia". Proceedings of the 3rd Workshop on Adaptive Hypertext and Hypermedia, pp. 55-62, 2001.

[5]. De Bra, P., Aerts, A., Berden, B., de Lange, B., Rousseau, B., Santic, T., Smits, D., and Stash, N., "AHA! The Adaptive Hypermedia Architecture". Proceedings of ACM Hypertext'03, Nottingham, 2003.

[6]. 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.

[7]. EI-Beltagy S. R., Hall, W., De Roure, D. and Carr, L. "Linking in Context". Proceedings of The 12th ACM Conference on Hypertext and Hypermedia, pp. 151-160, 2001.

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

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

[10]. Houben, G.J., Barna, P., Frasincar, F. and Vdovjak, R., "Hera: Development of Semantic Web Information Systems". International Conference on Web Engineering, (ICWE 2003), Springer Verlag, LNCS 2722, pp. 529-538, 2003.

[11]. Marshall, C.C., Shipman, F.M., "Spatial Hypertext: Designing for Change". Communications of the ACM, Vol. 38, nr. 8, pp. 186-191, 1995.

[12]. Wu, H., De Kort, E., De Bra, P., "Design Issues for General Purpose Adaptive Hypermedia Systems". Proceedings of the 12th ACM Conference on Hypertext and Hypermedia, pp.141-150, 2001.

[13]. Wu, H., "A Reference Architecture for Adaptive Hypermedia Applications", Phd thesis, Eindhoven University of Technology, ISBN 90-386-0572-2, 2002.