Design issues in adaptive web-site development

De Bra, P.M.E.

Published in:

Published: 01/01/1999

Document Version
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 18. Dec. 2018
Design Issues in Adaptive Web-Site Development

Paul De Bra [*]
Eindhoven University of Technology
The Netherlands
debra@win.tue.nl

Abstract: For almost a decade people have been developing hypertext or hypermedia applications
that adapt to some "features" of their users, like knowledge or preferences [Brusilovsky, 1996].
Recently some adaptive application environments have become available that use World Wide Web
technology. Examples of such systems are Interbook [Brusilovsky et al., 1998] and AHA [De Bra &
Calvi, 1998]. The adaptation can range from a simple (automatic) selection between different versions
of some information pages to the completely dynamic generation of all pages from atomic information
units and the automatic generation of all hypertext links. This paper sketches a general architecture for
adaptive Web-sites by building on existing models such as Dexter [Halasz & Schwartz, 1994] and
IMMPS [Bordegoni et al., 1997].

More importantly, this paper identifies issues in adaptive Web-site design for which no general
approach or solution appears to exist (yet). These include (but are not limited to): the separation of a
conceptual representation of an application domain from the content of the actual Web-site, the
separation of content from adaptation issues, the structure and granularity of user models, the role of
a user and application context, and the communication between different adaptive Web-site "engines".

Introduction

Hypermedia systems in general, and Web-based systems in particular, are becoming increasingly popular as tools
for user-driven access to information. The linking mechanism of hypermedia offers users a large amount of
navigational freedom. Unfortunately, because of this freedom it becomes impossible for authors to anticipate all
possible navigation paths a user can take. When a user (re-)visits a Web-site, she will often find links to
information that is either not relevant for her current task or that is hard or impossible to understand. The user
may be missing some important background knowledge or information that is available on the Web-site, but that
is not necessarily visited first.

During the past decade different types of hypermedia systems and Web-sites were built that are able to perform
some kind of personalization. There are different names for such systems or applications:

- In adaptable hypermedia the user can provide some profile (through a dialog or questionnaire). The system
  provides a version of the hypermedia application that corresponds to the selected profile. Settings may
  include certain presentation preferences (colors, media type, learning style, etc.) and user background
  (qualifications, knowledge about concepts, etc.) On the Web there are several such sites that use a
  questionnaire to tailor some part of the presentation to the user (usually the advertisement part...)
- In adaptive hypermedia the system monitors the user's behavior and adapts the presentation accordingly.
  The evolution of the user's preferences and knowledge can be (partly) deduced from page accesses.
  Sometimes the system may need questionnaires or tests to get a more accurate impression of the user's state
  of mind. Most of the adaptation however is based on the user's browsing actions, and possibly also on the
  behavior of other users (although we don't describe the latter feature in this paper).
- In dynamic hypermedia the user's behavior is monitored just like with adaptive hypermedia. However,
  instead of changing (adapting) a predefined presentation, dynamic hypermedia systems generate a
  presentation from "atomic" information items, often through natural language generation.

This paper focusses on Web-based adaptive hypermedia applications. This means that we assume that a
hyperdocument exists that consists of (HTML) pages and links. An Adaptive Hypermedia System (AHS) or
engine may change the content and presentation of nodes and may alter the link structure or annotate links, based
on a user model. Such functionality can be achieved through some "standard" Web technology such as
CGI-scripts, Java Servlets, or Active Server Pages. The aim of an adaptive Web-site is twofold:
• The AHS tries to guide the user towards relevant, interesting information and away from irrelevant information or pages the user cannot (yet) understand. This is done by manipulating the link structure or link presentation. We call these manipulations link adaptation.
• The AHS provides additional or alternative information (on a page) to ensure that the (most) relevant information is shown and that the user can understand the information as it is presented. (Some technical terms may need to be explained or avoided for instance.) We call these manipulations content adaptation.

(Brusilovsky [Brusilovsky, 1996] talks about adaptive navigation support and adaptive presentation. However, we avoid these terms because they are confusing. Often the way in which navigation support is made adaptive is by means of link (anchor) annotation. In that case the “presentation” of the link is changed, but in [Brusilovsky, 1996] this is not considered to be adaptive presentation.)

This paper describes several problem areas related to adaptive hypermedia in general, and to adaptive Web-sites in particular. It is aimed at spinning some discussion on these issues, not on describing solutions that are widely accepted and used. Section 2 describes how a domain model can be designed such that information content and link structure can be described on a conceptual level and on a concrete (information) level. Section 3 focusses on user modeling and on how to provide adaptation (based on a user model) in such a way that the desired adaptation is easy to describe. It also introduces the notion of context, and shows how context relates to the domain model and the user model. Section 4 shows how adaptive hypermedia applications can be built using some “standard” Web-technology. Section 5 focusses on the communication between adaptive hypermedia systems. Before the use of adaptive hypermedia on the Web can become widespread we need an easy and flexible way to initialize an adaptive Web-site by importing user models from other sites (or other parts of the same Web-site). Different (sub-)applications may work together to build a common user model.

2. Modeling an Application Domain

An adaptive hypermedia application or Web-site deals with a certain subject domain. The description of this domain can be viewed (and described) at three levels:

• At the lowest level there are information fragments. These are considered as atomic units as far as the AHS is concerned. A fragment can be a paragraph (or other piece) of text, an image, a video clip, etc. The AHS is not concerned with the internal structure of a fragment. Fragments may be static (stored) units of text or may be generated by an application-specific piece of software (like a natural language generation module).
• The “unit of presentation” is called a (Web-)page. In hypermedia terminology the term node is often used instead. A page is constructed out of fragments. Which constructors are possible depends on the AHS. In the AHA system [De Bra and Calvi, 1998] for instance (see also Section 4) every page is a linear sequence of static fragments. These fragments are conditionally included. (The adaptive "engine" determines which fragments are shown and which are not.) When fragments are stored in separate files the technique of server-side includes can be used to assemble pages. AHA uses HTML pages that contain all fragments of that page, and "conditionals" to determine which fragments to show (see Section 4). Interbook [Brusilovsky et al., 1998] uses MS-Word files from which several (separate) HTML pages are generated.
• The application domain can also be described in terms of high level concepts. (In Interbook the MS-Word files from which several Web-pages are generated can be regarded as such concepts.) Relationships between concepts can be used to suggest desirable navigation paths. Since this description is at a high level, the navigation paths do not necessarily translate directly to hypertext links between pages. Each high-level link to a concept must be translated or resolved (by the adaptive engine) to an actual link to a Web-page (because only Web-pages can be shown). Some concepts may be part of a "bigger", composite concept. The composite concept "hierarchy" must be a directed acyclic graph, meaning that no concept may contain itself (either directly or indirectly).

When designing an adaptive application one has to decide which kinds of concept relationships need to be supported, and how these will be used. Most AHS support a fixed set of concept relationship types, such as hypertext links and prerequisite relationships.

• Links become an interesting challenge when they are allowed to point to composite concepts. The AHS then has to resolve a link destination to one of the Web-pages that correspond to the composite concept. This idea of links to concepts instead of to pages is present in the Dexter reference model [Halasz & Schwartz, 1994] and the recently developed AHAM model [De Bra et al., 1999] (a Dexter-based model for adaptive hypermedia). In Dexter (and AHAM) a link points to a specifier of its destination. A specifier can
be (somewhat) compared to a URL. When a user follows the link the URL is interpreted by software on the server side in order to translate it into a server-specific address of a (static or dynamic) object. Dexter (and AHAM) further require the existence of an accessor function to actually generate or retrieve the object. Dexter (and AHAM) use an architecture consisting of 5 layers. They concentrate on the storage layer that deals with nodes (concepts and pages) and links. The actual interaction with the user is achieved through the runtime layer. When a user "clicks" on a link it is the runtime layer that passes a URL to the storage layer. The actual access to objects is implemented in the within-component layer that is system-dependent and therefore not described within the model. These three main layers are connected through anchoring and presentation specifications. In this paper we go further than Dexter and AHAM because we want to describe adaptive applications not only conceptually, but also within the context of the WWW architecture. Figure 1 shows the AHAM model.

![AHAM reference model](image)

- **Prerequisites** are used to help the user in selecting meaningful paths through the information. When concept A is a prerequisite for B it means that the user should visit (pages about) A before B. However, this does not mean that there should be a link from A to B. (A may be studied "long" before B.) By describing prerequisite relationships in the domain model an author does not prescribe a specific way in which the AHS must deal with prerequisite relationships. In Section 3 we come back to this issue when discussing how link-adaptation can be achieved (on the Web). When A is a prerequisite for B the AHS will use link-adaptation to guide the user towards A before showing or emphasizing the way to B.

An AHS should offer authors a tool to verify whether the generated concept relationship structure is sound. In an AHS with links and prerequisites this means that it must be possible to reach every page (from an author-defined starting point) without "violating" any prerequisite relationships. Note that this check involves links and prerequisites together. It implies the following rules (but is stronger than both of them):

- The link structure should be connected. (It must be possible to reach every page from the author-defined starting page.)
- There should be no cycles in the prerequisite relationships.

One can think of other types of concept relationships as well. An example are **inhibitor relationships**: when A inhibits B this means that after a user has studied concept A she should not (or no longer) visit B. There are two approaches towards providing more flexibility in defining new kinds of relationships:

- An advanced AHS may offer a tool that lets designers define new concept relationship types. In order for these new types to be useful the designer must be able to specify how relationships of a new type influence the adaptation. The AHAM reference model suggests that the AHS may offer a rule-based language for expressing the semantics of concept relationship types. Such rules are called **generic rules**. It will often be
possible to develop efficient algorithms to verify the soundness of a concept relationship structure consisting of relationships of user-defined types.

- The AHS may offer a language in which an author can express how specific concepts relate to each other and how (knowledge about) concepts influence the adaptation. The AHAM model suggests the use of specific rules for this. Such specific rules offer the greatest possible flexibility. However, they have the disadvantage that the rules must be "repeated" for each instance of a relationship of a new type. Also, it is not feasible to develop efficient algorithms to verify the soundness of a structure of such arbitrary concept relationships and their associated effect on the adaptation. The AHA system [De Bra & Calvi, 1998] offers specific rules. The (Web-based) course 2L690: Hypermedia Structures and Systems (offered by the Eindhoven University of Technology to students of many universities via Internet) uses the AHA system extensively. The specific concept relationship types in this course include prerequisites and inhibitors. Section 4 describes some aspects of the AHA system.

3. User Modeling and Adaptation

In order to create a Web-site that adapts itself to each individual user the server must register each user's action and deduce from that how the user's "state of mind" evolves. Based on this abstraction of the user's state the system can decide how to perform some adaptation. The representation of the user's state of mind is called user model. It contains aspects that are controlled explicitly by the user, such as color or media preferences, learning style, background knowledge, job situation and other items that can be entered through a questionnaire. The more interesting part of a user model is the information the system maintains about the user's "relation" to the domain concepts. Furthermore, the system gathers this information by observing the user's browsing behavior.

The AHAM model [De Bra et al., 1999] describes the structure of a user model as a table that contains for each (domain model) concept a set of attribute/value pairs. The table below shows a very small example of such a user model.

<table>
<thead>
<tr>
<th>name</th>
<th>knowledge</th>
<th>read</th>
<th>ready</th>
</tr>
</thead>
<tbody>
<tr>
<td>WWW</td>
<td>learnt</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>HTML</td>
<td>well-learnt</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>HTTP</td>
<td>not-known</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Each AHS may provide different attributes and different attribute types. An advanced AHS might even let the author or designer declare new attributes, provide rules to generate values for these attributes, and rules to perform adaptation based on these values. Some attributes typically found in AHS are:

- **Knowledge value**: by reading pages (or taking tests) the user gains or confirms knowledge about concepts. The possible values for this attribute can differ depending on the AHS used. AHA [De Bra & Calvi, 1998] currently supports only Boolean values. Interbook [Brusilovsky et al., 1998] supports a few "discrete" values, such as "not known", "learned", "known". Other systems (see e.g. [Pilar da Silva et al., 1998]) may support more values. While having more different values enables more different ways to adapt to the difference in knowledge about a concept, it also becomes more difficult to accurately determine (guess?) what the appropriate knowledge value for a certain user and concept should be.

- **Read**: For concepts that are associated with (Web-)pages the AHS may register whether the concept was read. This can be a Boolean value (enough to accomplish the typical blue/purple link color change Web-users are accustomed to), or a complete history of access times. The AHA system even registers both the time when a user requests a page and the time when she leaves that page.

- **Ready to be read**: When all prerequisite knowledge for a concept is acquired, all pages about that concept become ready to be read. The AHS may present links to such "ready" pages differently from links to pages for which the user is not ready. It may make a difference whether an AHS maintains the readiness state in the user model or whether it determines the readiness each time a page is requested.

Many other attributes can be thought of, including a knowledge decay value (how much the user has forgotten about this concept), an expiration time value (for pages with a dynamic content that is updated at certain time intervals, such as news bulletins or a weather forecast). Again, while having more attributes increases the
possibilities for adaptation, it becomes more difficult to determine meaningful values for these attributes, and also to come up with meaningful rules to determine how the attributes must be used in the adaptation process.

There is a close tie between the value domain for the knowledge value attribute and the granularity of concepts. In the AHA system every page is considered a concept. Such fine-grained approach is needed because the knowledge of a concept can only be true or false. In Pilar Da Silva’s system the knowledge value ranges between 0 and 100. Several pages can contribute towards the knowledge of a single (higher-level) concept. Thus, a richer value domain for knowledge values enables a coarser granularity in the concept space. Such a simpler concept space is easier to design, and concept relationships can be simpler as well. (In AHA a concept relationship often involves a large number of concepts, as is exemplified in the hypermedia course 2L690.

Figure 2 shows the architecture of the IMMPS reference model for Intelligent Multimedia Presentations [Bordegoni et al., 1997]. The left-hand side shows the processes involved in handling a simple user interaction such as clicking on a link (which is a primitive way of formulating a goal). The right-hand side shows functional parts of the AHS that are involved in maintaining different models (domain model, user model, context model) and in designing presentations. We only describe the role of the processes on the left-hand side of the figure.

1. First there is the Control Layer, in which the link (URL) is resolved. (This is part of the run-time layer of the Dexter [Halasz & Schwartz, 1994] and AHAM [De Bra et al., 1999] models.) This means that when the link does not identify a single Web-page but rather a composite concept, the AHS must deduce (from the domain model and the user model) which Web-page to show.

2. Next the appropriate content is selected and retrieved in the Content Layer. (This corresponds to the accessor function in the Dexter (AHAM) model. The content is retrieved from the within-component layer.) In a Web-based system the content is a Web-page. The Web-page may be generated or assembled from fragments. The selection of the appropriate fragments is performed in this layer.

3. The Design Layer is responsible for assembling a (Web-)page from the selected fragments. Fragments may need to be sorted, and links may be assigned to different link classes in order to make it possible to do link-annotation. The IMMPS model distinguishes three possible ways to achieve a properly designed Web-page that satisfies user-specific requirements as well as platform-specific requirements: layout after production, layout before production and layout interleaved with production. Apart from adaptation based on the user model the AHS must also adapt the Web-page to the user-interface (Web-browser) that will be used to view the page. The actual HTML code sent to a computer with a high resolution screen will be different from that sent to a Web-TV or to a PDA for instance.

4. The Realization Layer is responsible for finalizing the Web-page so that it can be displayed by the browser. One of the tasks of this layer is to add the appropriate style sheet. This determines page-layout aspects for the content (fonts, alignment, etc.) as well as for the links (link color to class association for instance). (The result of this layer is what Dexter and AHAM call a presentation specification.)

5. The Presentation Display Layer represents the rendering of the actual page, as performed by the Web-browser. This corresponds to the run-time layer of the Dexter (and AHAM) model.

The techniques for content-adaptation may be straightforward, and will not be elaborated upon in this section. We briefly describe different ways to perform link adaptation, and how they fit in with the model(s). (In Section 4 we describe how to implement content- and link-adaptation on the Web.)

- **Direct guidance:** This technique involves the use of link anchors that do not always lead to the same Web-page. A typical example would be a “Next” button, leading to the best page to be read next. The AHS determines the best next page based on the user model and the user’s goal. It can either associate a server-side program with the button (in which case the URL is the same for every user, but the AHS generates a different page), or the URL of the best page can be calculated when generating the current page. (One can safely assume that the decision which page is best to be read next does not change while the user is reading the current page.)

- **Link sorting:** This technique is typically used in applications where it is necessary to generate a list of links to pages. Each link (anchor) can be presented as an item in a list, and be considered as a fragment. Presenting the list of links then becomes a matter of selecting fragments from a larger list (maybe of links to every page) and sorting them. Typical applications are information retrieval systems that sort (links to) pages according to a (personal) relevance criterion, and educational applications where the user selects a learning goal and the AHS generates a list of pages to study, in an order that is based on prerequisite relationships.
• **Link annotation:** When the AHS determines that some pages are more relevant or appropriate than others, and the links to such pages appear in running text, the system may wish to indicate the different status of these links in a visual way. Interbook (Brusilovsky et al., 1998) uses colored balls and arrows to indicate the "status" of links. AHA (De Bra & Calvi, 1998) uses link classes and style sheets to adapt the color of the link anchors.

• **Link hiding:** This is a special case of link annotation: links that are considered not desirable (at the moment) are presented as normal (uncolored) text. In the AHA system the color scheme can be configured, leading to link annotation or link hiding (when one of the colors is set to "black", the color of running text).

• **Link removal:** In lists of links it is possible to simply remove non-relevant links. This technique is often combined with link sorting: only the first few (most relevant) links are shown, the others are removed.

• **Link disabling:** Links are accessible through "link anchors" (The &lt;A&gt; tag in HTML). When the anchor is removed, but the text is presented just like a link the user can see that there is a link, but clicking on the link has no effect. This technique can be used with link hiding (the text remains visible but the user cannot see that their might ever be a link here), or with link annotation (the inaccessible links are presented in a visually different way than the accessible links).

In Section 4 we show how all these techniques can be realized using Web technology. Unfortunately we know of no current AHS that supports all these techniques (although each technique is supported in some existing AHS).

### 4. Realizing Adaptive Hypermedia on the Web

This section describes techniques that are available to develop adaptive Web-sites. It focusses on the use of Web-related technology, not on proprietary architectures. We separate issues related to domain-modeling, user-modeling, and the performing the actual adaptation. We illustrate how these aspects are handled in the AHA system [De Bra & Calvi, 1998] which is used for some Web-based courseware and a "kiosk" system at the Eindhoven University of Technology.

#### 4.1 User-Modeling on the Web

In order to adapt to each individual user the AHS must maintain a model (representation) of the user’s "state of mind". We are interested mainly in adaptation to individual users, not to groups. This suggests that the user...
model could well be stored on the user's (client) computer, and not on the Web-server. The concept of cookies was invented specifically for maintaining some user-dependent data on the client site. (See http://www.netscape.com/newsref/std/cookie_spec.html for details on cookies.) However there are some good reasons for maintaining the user model on the server side (or for not keeping it on the client side):

- A single user may not always connect from the same machine. PCs in labs or libraries on a university campus may not allow a user to share a single set of cookies. And especially in environments where users sometimes connect through a Unix workstation and sometimes through a PC they may have at least two separate sets of cookies.
- Cookies are limited in number and size. A single cookie cannot exceed 4Kbyte. No more than 20 cookies are allowed per domain and no more than 300 in total (on a client machine). When a cookie becomes too long it is truncated. When the 20 or 300 limit is exceeded the least recently used cookie is discarded. These restrictions imply that one cannot consider cookies to be a means of maintaining a permanent record of a user's state of mind.
- Cookies are sent back and forth between client and server. This is necessary because the server needs to have access to the user model for doing some of the adaptation, especially the selection of fragments. (It would be extremely inefficient to send all the fragments of a page to the client and let the browser figure out what to show and what to hide.) If a user-model-cookie is 2Kbyte large the exchange of this cookie may require more network bandwidth than the transmission of a whole Web-page.

For the above (and possibly other) reasons existing Web-based AHS like AHA [De Bra & Calvi, 1998] and Interbook [Brusilovsky et al., 1998] store the user-model on the server-side.

Updates to the user-model can only be the result of a user action that involves an interaction with the Web-server. In systems like Interbook and AHA this happens whenever the user requests a page (by clicking on a link anchor) or when the user completes a form, like in the case of a (multiple choice) test. In AHA a special invisible "stop" applet is inserted in every page. This applet sends a request to the server each time a page is unloaded (because another page is accessed). AHA thus registers how long a user has been reading a page (or actually, how long the user's browser has been displaying the page). Using Dynamic HTML (and scripting languages like JavaScript) it is becoming possible to have the browser send requests to the server as a result of user actions without changing the page that is being displayed. This is interesting for user actions that cause additional information to become visible or that cause information to disappear. (Unfortunately it is not (yet) possible to associate scripting code to the scrolling event.

The user model in the AHA system consists of the following:

- Some registration info (name, email, id, password, etc.)
- Color preferences, used to perform link adaptation.
- The set of known concepts.
- A complete browsing history, with for each page all access as well as deactivation times.
- All results for all multiple-choice tests.

In the current version quite a bit of this information is not (yet) used: the browsing history is used to decide whether to color links blue or purple (or other colors if the user changes the preferences). Exact access and reading times are not used. Also, the knowledge of a concept is a Boolean value. In a future version we will be using a "percentage".

As the IMMPS model shows (see Figure 2) some designers wish to define a context model (or context expert) as well as a user model. Such a context model could maintain state information not directly related to the user, but related to the environment in which the application is being used. Examples of context information include properties of the computing environment of the user (e.g. screen resolution, network bandwidth) or of the application domain (e.g. the status of some machinery in a factory, the date or time, the location of the user, the current weather conditions, etc. Such contextual elements can be taken into account much in the same way as color preferences, knowledge about concepts or the outcome of multiple-choice tests. We therefore do not describe how to handle such a context model in a way that would differ from how the user model is handled.

4.2 Concepts and Content on an adaptive Web-site

In Section 3 we have proposed an architecture in which an application domain is described at three levels: concept, page and fragment. Since on the Web the "unit of presentation" is a (Web-)page, these three levels have
to somehow be converted into one. Some ways in which this can be done are:

- In [Pilar da Silva et al., 1998] an AHS is described in which a graphical tool is used to describe how different (Web-)pages are associated with a single domain concept. Each page contributes a fraction (percentage) of the knowledge of the concept. Although this is the most promising and powerful approach it is also difficult to implement correctly. The following cases need to be handled correctly:
  - When a page is read more than once, its contribution to a concept should not be counted twice.
  - When a page is read when prerequisites are not yet satisfied the contribution to the knowledge of a concept should probably be lower than when the user is ready to read the page. When the page is first read when the user is not ready, and later when she is, calculating the contribution to the knowledge becomes tricky.
  - Two pages may have overlapping information, and thus an overlapping contribution to the knowledge about a concept. It is thus not clear that contributions should simply be added.
  - The sum of the contributions in pages about a concept may exceed 100% but the knowledge about the concept cannot exceed 100%.
- In systems like Interbook the application domain is described by a structured document. Sections of the document are associated with domain concepts. A "make" procedure translates the document into a set of HTML pages, and associates concepts to pages. Each page has background and outcome concepts. Background concepts represent prerequisite knowledge. Knowledge about outcome concepts is generated by reading the page.
- Instead of decomposing a large piece of information (like a structured Interbook document) into a set of Web-pages on the server side the decomposition can also be achieved by selectively making portions of the document visible in the browser. In HTML version 4.0 (or "Dynamic HTML") it is possible to open and close or show and hide parts of a Web-page, based on user-generated events (such as mouse clicks). This approach is not used in current Web-based AHS, possibly for one or more of the following reasons:
  - The definition of Dynamic HTML has only become stable very recently and complete implementations in Web-browsers are not yet readily available.
  - The Web-page that must be transferred from server to client is much larger than a normal Web-page (because it contains several virtual pages).
  - The browser must notify the server each time a different part of the large page is viewed by the user, in order to keep track of what the user has read.
- The (non-Web-based) AHS MetaDoc [Boyle & Encarnacion] uses a technique called streichtext that is similar to what we described above. Different parts of a page can be opened up and closed. The system adaptively decides which parts to open up when a page is first displayed, and it takes into account the parts opened up or closed by the user. Dynamic HTML is making it possible to implement this functionality on the Web.
- The AHA system associates zero or more concepts (but typically one) to a page. In the current version knowledge is only "generated" if a page is read when the prerequisites are satisfied. (The next version will deal with "partial knowledge"). Instead of using a set of concepts as prerequisite knowledge, each page depends on a requirement that is a Boolean expression on concepts. Through the use of and, or, not and arbitrary parentheses a rich collection of requirements can be formulated. However, it is difficult to maintain a clear picture of the "concept map" when complicated requirement-expressions are being used.
- The AHA system offers the conditional inclusion of fragments. A Web-page may contain an arbitrary number of (possibly nested) fragments that are included if a Boolean expression on concepts is satisfied. The order of the fragments on the page is fixed.
- An alternative to the AHA approach would be to use a separate file for each fragment and server side includes to assemble pages from fragments. This would make it easier to put fragments in a different order for different users, or to include a fragment in several pages. A disadvantage is that this approach involves more overhead, especially when the fragments are very small. (In the course 2L690 some fragments consist of only a few words or a link anchor.)

4.3 Techniques used to achieve Adaptive Content and Linking

There are two areas in which adaptive content is being used in AHS:

- When a user wishes to move to a "major" concept, the AHS has to determine how to guide the user through the different pages that describe the concept. The Interbook system automatically generates a partial table of contents (of the structured source document). Each item in this table of contents is a link to the corresponding section, and is annotated to indicate whether the user is advised to go to that section or not.
- Depending on the user's knowledge the AHS may decide to insert some additional information (fragments)
in a page, or to remove some information the user does not need. The AHA system uses structured HTML
comments to conditionally include fragments of text, as shown in the example below:

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

The presentation of a page, as seen by the user, is always the result of a filter operation performed by some
server-side program. Such a program can be a CGI-script, a server plug-in or a Java Servlet. In case the
presentation consists of several frames or "sub-windows", the content of each frame is generated by a separate
request to the server and thus by a separate (run of the) program. Interbook uses a fixed presentation that consists
of five frames. The overhead of executing five CGI-scripts is likely to be noticeable. The presentation in AHA is
not fixed. An author can create any desired frames structure. AHA has been used for a course without the use of
frames and for another course and a kiosk system, both of which use frames. The overhead in AHA is minimal
thanks to the use of a Java-based Web-server together with Java Servlets.

Some kinds of link adaptation can be realized through adaptive content. In AHA, a link can be easily disabled
and hidden through the inclusion of a conditional fragment:

```html
This course ends with an
<!-- if ready-for-assignment -->
<A HREF="assignment.html">assignment</A>
<!-- else -->
assignment
<!-- endif -- >
that is accessible when you have studied all concepts.
```

However, most link-adaptation is performed by visually annotating link anchors to indicate whether the user is
advised to follow the link or not.

- In Interbook all link anchors are followed by a graphical icon indicating the status of the link. The color of
the icon indicates whether the link leads to interesting new information (green), no new information (yellow)
or information that is not desirable at this time (red). The icon can be a ball or an arrow indicating that the link goes up or down the (section) hierarchy. A ball can be overlaid with a checkmark to indicate
that an outcome of a page is already (partially) known.
- In AHA the link anchors themselves are colored to indicate their status. This has the advantage that a
sentence containing a link is not interrupted by some graphical icon, but it has the disadvantage that the
possibilities for annotation are more limited (i.e. no balls, arrows and checkmarks). The way AHA achieves
link annotation is through the use of (cascading) style sheets. A link in an HTML page is given the class
"conditional":

```
This is a
<A HREF="link.html" CLASS="conditional">conditional link</A>.
```

The AHA Servlet verifies whether the destination of this link is desirable or not, and whether it has been
visited or not. The link class is then transformed into good, neutral or bad:

```
This is a
<A HREF="link.html" CLASS="GOOD">conditional link</A>.
```

The AHA Servlet also inserts a stylesheet definition into the header of each HTML page:

```html
<STYLE TYPE="text/css">
A.Good { color: 0000ff }
A.Bad { color: 202020 }
A.Neutral { color: 7c007c }
</STYLE>
```

The above stylesheet shows the default color scheme of AHA. The "bad" links are shown in a dark shade of
gray, which makes the link adaptation behave very closely to link hiding.

In order to motivate more authors to use adaptive hypermedia the authoring process should be made much simpler than it is today. For Interbook the author must write MS-Word files that are structured in a specific way and that use (hidden) comments to provide information about the underlying concepts. For AHA the author must write HTML files with structured comments to provide information about concepts and to conditionally include fragments. We are currently working on a redesign of AHA that includes tools to facilitate authors in designing the concept space separately from the content space.

5. Communication Between Adaptive (Sub-)Applications

In order to perform adaptation to each user in a proper way an AHS needs to observe a user a long time. Currently each adaptive Web-site must start its observations from scratch. When adaptive sites can exchange information about the same user they can adapt to a user more quickly and in a better way. Therefore adaptive Web-sites should be able to exchange (parts of) user models.

Technically it is not difficult to enable adaptive Web-sites to exchange user model information. The AHA system offers a forms-based interface to the user model. Through that interface a user can view her user model, adjust it if desired, and submit it back to the server. It is easy to add a similar interface and API to enable remote systems to request a user model from AHA and to update it. (There is of course an issue of authorization that we ignore in this paper.) The still unsolved issue in communication between AHS in general (and adaptive Web-sites as well) is the semantics of the data in a user model:

- If two adaptive Web-sites share an application domain concept (with the same name), how can they be sure that they both mean the same "real world" thing? For instance, one Web-site may deal with Unix commands and describe the concept "cat", while another Web-site may deal with pets and also describe the concept "cat". A user who knows the concept "cat" in one of these sites should not be treated as knowing that concept in the other site.
- If two adaptive Web-sites share an application domain concept (with the same name and meaning), how do the ways in which they measure knowledge about that concept compare? AHA uses Boolean values, Interbook uses a few discrete (named) values and Pilar da Silva uses percentages. It is impossible to automatically determine whether "true" (in AHA) translates to "learned" or "known" in Interbook, or to 70%, 80% or perhaps 100% in Pilar da Silva's system.
- If two adaptive Web-sites share an application domain concept (with the same name and meaning), and they agree on the meaning of the knowledge values, how do they decide whether the information that one system offers about the concept is equivalent to the information the other systems offers about the same concept?

It is easy to see that a general resolution for these issues cannot exist. Different universities (especially in different countries) do not agree on the issue which course from one institute is equivalent (or even better) than a course from the other institute. Different governments do not agree on the equivalence of university degrees.

Because of the ambiguities described above some conversion tools need to be built that are easily configured to translate concept names and to convert knowledge values. An area where standardization may help is in domain-independent aspects of a user model. This includes aspects like

- User and platform dependent media preferences. Some users prefer text, some video and some audio. Some platforms only allow certain data types and some network connections require reduced quality (and thus bandwidth consumption).
- Navigation or learning-style preferences. For instance, some users prefer to study definitions before seeing examples while others prefer to first see some examples and then study the definitions.
- Link adaptation technique. Systems like AHA can be configured to support more than one link adaptation technique. (A user can configure AHA to use link hiding or link annotation. An author can also force AHA to use link disabling.)
- Content adaptation technique. AHA determines whether to include a fragment or not. Other options would be to enable users to "expose" hidden fragments, or to gray out fragments instead of removing them. The SaD system [Hothi & Hall, 1998] for instance grays out fragments that are deemed not to be suitable for the user.
6. Conclusions

Adaptive hypermedia has been around for about a decade. The Web offers the technological base for implementing most of the adaptive technology that has been implemented on other platforms. Some Web-based systems, including Interbook [Brusilovsky et al., 1998] and AHA [De Bra & Calvi, 1998] are being used, mostly in educational environments.

Authoring adaptive hypermedia remains a problem area, whether we are considering Web-based systems or not. Systems like Interbook and AHA were designed and implemented by computer scientists and the first applications of these systems were developed by (the same) computer scientists as well. Authoring is probably still too complicated for “average” authors from non-computer-related fields.

Web-based systems are expected to take advantage of the global nature of the Web. However, sharing user models still proves to be difficult because application domains are difficult to compare. Also, the internal representation of knowledge about an application domain in one system is difficult to translate to the internal representation of the same application domain in another adaptive hypermedia system.

7. References

[Bordegoni et al., 1997]

[Boyle & Encarnacion]

[Brusilovsky, 1996]

[Brusilovsky et al., 1998]

[De Bra & Calvi, 1998]

[De Bra et al., 1999]

[Halasz & Schwartz, 1994]

[Hothi & Hall, 1998]

[Pilar da Silva et al., 1998]

[*]
Paul De Bra is also affiliated with the University of Antwerp and the “Centrum voor Wiskunde en Informatica” in Amsterdam.