Creating Adaptive Hyperdocuments for and on the Web

Paul De Bra[1]
Department of Computing Science
Eindhoven University of Technology
The Netherlands
debra@win.tue.nl

Licia Calvi
Department of Roman Languages
University of Antwerp
Belgium
calvi@uia.ua.ac.be

Abstract: The course "2L690: Hypermedia Structures and Systems" is taught entirely through World Wide Web, and offered at six different universities in the Netherlands and Belgium. Different approaches have been taken towards adding adaptivity to the course text. This paper reviews these development steps and presents the final design, which results in adaptive hyperdocuments that can be written in standard HTML 3.2, possibly by using off the shelf HTML editors. We also present a simple but powerful representation of user (student) knowledge, as used to adapt the link structure and textual contents of the course text.

1. Introduction

Many different definitions of and techniques for adaptive hypermedia exist. A good overview is presented in [Brusilovsky 1996]. In this paper we follow the terminology introduced in that overview to characterize the kinds of adaptation introduced in the course "2L690: Hypermedia Structures and Systems". We describe not only the adaptation techniques used for this course, but also compare them with some other initiatives for using adaptive hypertext in courseware, such as a C Programming course [Kay & Kummerfeld 1994a, Kay & Kummerfeld 1994b] and the ELM-ART Lisp course [Brusilovsky et al. 1996a].

World Wide Web was not designed with highly dynamic applications in mind. A typical characteristic of adaptive hypertext is that during the reading process the presentation of an information item (e. g. a page) may be different each time that item is revisited. The way some WWW browsers deal with their history mechanism makes it difficult to ensure that pages are reloaded each time they are (virtually) modified on the server. This problem can be resolved with some browsers (like Netscape Navigator) but on some others (like NCSA Mosaic for X) it cannot. Unfortunately the browsers cannot be blamed for their behavior because the way they implement their history mechanism satisfies the requirements set out by the HyperText Transfer Protocol (HTTP) standard.

Authoring adaptive hypermedia is also difficult in a WWW environment, because the HyperText Markup Language (HTML) has no provision for "conditional text". It is not possible to create a single HTML document of which only selected parts are presented to the user, based on some kind of environment variables controlled by an agent that monitors the user's knowledge state or preferences. The Interbook tool [Brusilovsky et al. 1996b] for instance uses *concept-based indexing* to provide access to (non-adaptive) HTML pages from dynamically generated index pages. Displaying index and "content" pages simultaneously is done through *frames*, a technique introduced by Netscape but which is not available in all browsers, and which is not part of the latest HTML-3.2 standard.

Brusilovsky [Brusilovsky 1996] distinguishes two main categories of adaptivity:

- **Adaptive presentation**: these techniques consist of both the selection of different content or media depending on user preferences and the adaptation of a document's content based on a user's knowledge state.

[1] Paul De Bra is also affiliated to the University of Antwerp, and with the "Centrum voor Wiskunde en Informatica" in Amsterdam.

- **Adaptive navigation support**: these techniques change the (apparent or effective) link-structure between the pages that together make up a hyperdocument. Brusilovsky distinguishes between *direct guidance* (suggest a next best link), *adaptive sorting of links* (displaying a list of links from best to worst), *adaptive hiding of links* (hiding inappropriate links), *adaptive annotation of links* (marking links as appropriate or inappropriate but not deleting any) or *map adaptation* (changing graphical overviews of the link structure).

The courseware for "2L690: Hypermedia Structures and Systems" contains both forms of adaptivity. The textual contents of pages is adapted to the knowledge state of the student. This paper describes a new way of encoding conditional text in (standard) HTML documents. This new way supersedes a first attempt at providing adaptive content, described in [De Bra & Calvi 1997]. The course offers adaptive navigation support by means of adaptive hiding of links. Only links to pages which are "interesting" for the student to read next are shown. The technique used for this navigation support is described in [Calvi & De Bra 1997].

We propose a simple authoring environment for adaptive hypertext courseware, which offers the following features:

- The content of the pages of the course text is adaptive, as well as the link structure.
- The adaptive documents are written in standard HTML, and can be authored using (some of the existing) HTML editors.
- Pages of the course text, as well as tests and assignments that may be embedded in the course text, generate knowledge about *concepts*.
- Information items (ranging from words to large parts of documents) and links can be made dependent on Boolean combinations of concepts (using *and*, *or*, *not* and arbitrary parentheses).
- A verification tool lets authors check whether all the information in the course can be reached by a student, independent of the order in which the student decides to view pages. This problem is non-trivial since knowledge about certain concepts can make pages inaccessible.

2. Why Adaptive Hypertext on World Wide Web is Difficult

World Wide Web was not designed with highly dynamic applications in mind. This is not simply a matter of oversight in the standards definitions, but more of the programmers and companies who first created WWW-browsers and servers, as well as of document authors who only used a fraction of what the HTML and HTTP standards have to offer. Here are a few examples of problems with the current standards and practice:

- In an adaptive hypermedia system following a number of links forwards and then backtracking may result in changes to the previously visited documents. The HTTP standard does not require backtracking to request the documents from the server again, not even when the documents are expired. This means that there is no guaranteed way for a server to tell the browser to reload a document when the user uses the "back" button to revisit that page.

  > *By default, the Expires field does not apply to history mechanisms. If the entity is still in storage, a history mechanism should display it even if the entity has expired, unless the user has specifically configured the agent to refresh expired history documents.*
  > (quoted from RFC's 1945 and 2068, which define HTTP/1.0 resp. 1.1)

  The HTTP standard acknowledges that some browsers may let users configure the history mechanism to verify whether a document is modified even when going through the history mechanism. Unfortunately the standard does not *require* browsers to have such a feature, and specifies that the default behavior should be *not* to verify whether the document has changed.

- Although HTTP offers at least the possibility to suggest that pages should be reloaded by declaring them to be expired, many authors of HTML documents never indicate that a document may expire at some given date, not even when they know exactly when a new version will replace the current one. A possible reason for this is that HTML (either version 2.0 as defined by RFC 1866 or the newer version 3.2) does not offer an <EXPIRES> tag to indicate an expiry date. Authors have to use the <META> tag to force the server to generate an HTTP Expires field, using the following syntax:

```
<META HTTP-EQUIV="Expires"
    CONTENT="Tue, 04 Dec 1993 21:29:02 GMT">
```

The HTTP/1.0 standard encourages the expiry mechanism to the point that an invalid date of 0 should be interpreted as "expires immediately". (Sadly, this "encouragement" has been dropped in the HTTP/1.1 standard.) Still, expiry has not yet become sufficiently popular to warrant that all browsers interpret expire-fields correctly. Furthermore, apart from browsers there are also a number of proxy-caches that do not yet understand the HTTP/1.1 caching directives that have been introduced to avoid caching expired or rapidly changing documents.

> *Note: Applications are encouraged to be tolerant of bad or misinformed implementations of the Expires header. A value of zero (0) or an invalid date format should be considered equivalent to an "expires immediately." Although these values are not legitimate for HTTP/1.0, a robust implementation is always desirable.*
> (quoted from RFC 1945 which defines HTTP/1.0)

- HTML does not offer a possibility to conditionally include text or multimedia objects. There is no such thing as an "<IF>" tag.
  - Several attempts have been made to use the Unix C preprocessor for conditional pieces of content, but mixing C preprocessor commands (like **#ifdef** clauses) with HTML encoded documents generates source text that is difficult to write and read. The previous edition of the course "Hypermedia Structures and Systems" (with code 2L670) used a mix of **#if** and **#ifdef** constructs to achieve adaptive content [Calvi & De Bra 1997]. The newer edition, course 2L690 which started in the fall trimester of 1997, uses the approach described in this paper.
  - Pim Lemmens [Lemmens 1996] has proposed and implemented a mechanism for parameterizing Web pages, by means of &...; constructs. This suggestion works well for small textual variations, like including **&date;** in a document to generate the current date, or to offer alternative wordings for technical terms. Referring to a page as <**A HREF="test.html?node=page**> would result in all occurrences of **&node;** being replaced by the word "page", while <**A HREF="test.html?node=node**> would generate a document in which every **&node;** is displayed as "node" (which would be done after explaining what a "node" is). Although this construct looks like valid HTML it is not, and cannot be generated through a strict HTML editor.
  - In HTML tags are case insensitive. A "smart" preprocessor can therefore interpret tags written in lowercase differently from tags in uppercase. Course 2L690 uses this possibility to distinguish between *conditional links* (authored as <**a href=**... >) and *unconditional links* (authored as <**a HREF=**... >) [De Bra & Calvi 1997].
- An interesting and promising possibility is offered by scripting languages such as JavaScript (developed by Netscape Communications). Using JavaScript one can embed different variations of a document's content in a single file, and make the browser present the appropriate elements based on the values of some variables that can be generated by the agent which monitors a user's knowledge state or preferences. Unfortunately JavaScript is still heavily under development. Only a few browsers offer scripting and their definitions and implementations of JavaScript are incompatibly different. The HTML-3.2 definition is still only partially "script-aware", meaning that a <**SCRIPT**> tag has been defined as a placeholder, but the current JavaScript practice to include method calls in anchor and button tags is not (yet) allowed.

3. Encoding Knowledge and Conditional Text in HTML

Hypertext techniques and World Wide Web technology have been used in educational settings, mostly for computer-science courses where students have to master certain skills such as programming (in C [Kay & Kummerfeld 1994a, Kay & Kummerfeld 1994b], or in Lisp [Brusilovsky et al. 1996a]). The hypertext (link) structure in such courses is fairly simple, since learning a programming language is a mostly linear process. Indicating which chapters are still to be avoided and which pages to read first is easy.

In the course "2L690: Hypermedia Structures and Systems" the link structure is made complex on purpose: students learn about the concepts of hypertext, and the best way to do so is by experiencing hypertext. Not all link structures are equally easy to navigate through. In course 2L690 the "chapters" which appear to exist when the student looks at the first page are actually overlapping sets of pages. For a number of information nodes it is impossible to tell to which (unique) chapter they belong. The course contains some introductory chapters, giving definitions and a historical overview, and advanced chapters, describing reference models, navigation and retrieval problems, authoring issues and multi-user aspects. It is desirable for students to first read the introductory chapters, and therefore to advise or force them to do so (by dimming, hiding or removing links to the advanced chapters at first). Nonetheless an introductory chapter and an advanced chapter may share common pages. This makes enabling or disabling access to pages more complicated than simply enabling or disabling whole chapters, and it may also suggest using "simple" wording when a page is read as part of an introductory chapter, and more technical wording when that same page is read as part of an advanced chapter.

In order to monitor the student's progress and knowledge state *concepts* are associated with pages from the course text. Each concept is denoted by means of a single word. (Multiple words can be simulated by joining them using underscores instead of spaces.) Much like in [Rosis et al. 1994] the concepts are collected in a *Dictionary of Concepts*. While for programming language and similar courses the user-model needs to consist of both "KNOW-ABOUT" and "PRACTICE-IN" facts, we currently make no such distinction. The knowledge state of a student is simply a set of concepts the student has read about (or successfully taken a test about).

For each page of the course text a number of concepts may be prerequisite knowledge, and a number of (other) concepts may make the page superfluous. In [De Bra & Calvi 1997] this prerequisite and/or forbidden knowledge is used to determine whether to enable or disable links to a page. More complex Boolean combinations were not possible in that proposal.

Depending on the knowledge state of the student not only links to pages but also the contents of pages may need to be adapted. In [Calvi & De Bra 1997] we proposed to use C-preprocessor (**#ifdef**) constructs to achieve this goal. However, mixing HTML with C-preprocessor statements makes authoring unnecessarily complicated. Since we aim to provide an authoring environment which is also suited for the development of non computer related courses, authoring needs to be simple and intuitive. In our new proposal both whole pages, links to pages and pieces of HTML text (possibly including images), can be enabled or hidden depending on a Boolean combination of concepts. We use HTML comments to mix "if-statements" with HTML text, as illustrated by the following example:

```
<!-- requires true -->
<!-- generates index -->
  <H1>Hypermedia structures and systems</H1>
  Welcome to course 2L690 at the Eindhoven University of Technology.<p>
<!-- if not readme -->
  Since you are just beginning to browse through this course, you should first read
  <a href="readme.html">the instructions</A>.<p>
  The items below indicate (not necessarily disjoint) parts of the course text, which will
  become accessible after you have read <a href="readme.html">the instructions</a>.
<!-- else -->
  This course contains the following (not necessarily disjoint) parts:
<!-- endif -->
  <ul><li><a href="intro.html">Introduction</a> (please read this before the other items)
  <li><a href="definition.html">Definition of hypertext and hypermedia</a>
  <li>The <a href="history.html">history</a> of hypertext and hypermedia
<!-- if readme but not (introduction and definition and history) -->
  </ul>
  The following parts will become available later (when you are ready for them):
  <ul>
<!-- endif -->
  <li>The <a href="architecture.html">architecture</a> of hypertext systems
  <li><a href="navigation.html">Navigation</a> (and browsing semantics) in hypertext
  <li>etc. (other topics removed to save space)
  </ul>
```

Each page of the course text starts with a comment that indicates which Boolean combination of concepts is *required* to allow access to the page. (In the example, "true" means that nothing is required.) The second comment indicates which concept(s) are *generated* by visiting the page. The latter concepts are added to the student's knowledge **after** reading the page. This implies that text fragments that depend on the concepts generated by a page are not displayed the first time the page is visited. It also implies that a page may forbid the same concept(s) it generates, in which case it will be accessible only once. Section 4 explains that one needs to be careful with the selected combinations of required and generated knowledge. Should a page require a concept it generates, the page will never be accessible unless another accessible page generates the same concept.

Note that although the links to all chapters are always present in the source text, the software described in [De Bra & Calvi 1997] will hide (remove) these links until the student has gained sufficient knowledge. We could have used more "<**!-- if --**>" commands (actually comments in HTML) to include these links conditionally, but that would make the document source much harder to write (and read). The software of [De Bra & Calvi 1997] is kept in place because it is also needed for maintaining each student's individual log file. (The use of log files is described in [De Bra 1996].)

Note also that the keyword *but* in the second "<**!-- if --**>" statement is simply used as a synonym for *and*, but is closer to natural language.

When the student first looks at this page, no concepts are known yet, so the text fragment pointing to the "instructions" (readme.html) will be displayed and none of the sections from the list will be reachable. After reading the instructions this part will be omitted because the concept **readme** will be known. The first three sections become available, and a short explanation is displayed as to why the remaining sections remain unavailable. When the student has read some parts of the introduction, the definition and the history section, the concepts **intro**, **definition** and **history** will be known, and the list of sections will be shown as a whole again, with links to all the sections. (The cd-rom version of the paper contains the three possible versions of the page. These examples have been omitted here for lack of space.)

Conditional content need not necessarily be tied to "real" knowledge gained by the user. In the courseware for 2L690 the user can manually set knowledge on or off through a setup page. By switching knowledge of a **verbose** "concept" on or off one can give the user the option of selecting or deselecting optional additional content. It is thus possible to give the user a choice between different presentations of the same course text.

4. Validating Adaptive Hypertext Link Structures

In a static hypertext analyzing whether all nodes can be reached (from a given "root" node) is a matter of simple graph traversal. The Boolean conditions however complicate the link structure, because links may or may not be present depending on the student's knowledge state. As long as no negation (*not* operator) is used the *reachability problem* is still easy: one can repeat the process of trying to follow links (forwards) each time some knowledge is gained. It is not only important to find whether there are pages that are inaccessible because the knowledge required to read them can never be acquired. One should also check whether the conditions imply additional navigation steps (going back to acquire more knowledge elsewhere before being able to return and resume).

The cd-rom version of the paper contains a few examples that show undesirable adaptive link structures, both with and without negation. (They have been left out here for lack of space.)

We are currently building a simple tool that verifies the link consistency of a course text. In particular the tool analyzes the following properties:

- Are there navigation paths that make it impossible to visit some page(s)? If so, which pages may not be reachable? (This can be a consequence of using negation, but also of requiring concepts which are never generated before they are needed.)
- Are there (conditional) parts of pages that can never be viewed (no matter which navigation path is used)?
- Is it possible to navigate through the whole course text without ever following a forward link to a page that was visited before? (Given a course text like 2L690 this implies that it must be possible to read the text chapter by chapter.) If not, which pages must be revisited in order to gain access to which pages?

5. Conclusions and Future Work

Creating adaptive hypermedia documents, that have a complex (non-hierarchical) structure, is difficult in general. Analysis tools may be needed to help authors verify that their adaptive hyperdocuments are easy to navigate through. We are currently building such a tool that will be used not only for the next version of course "2L690: Hypermedia structures and systems", but also for a course on Italian economy and a course on Graphical User-Interfaces. Course 2L690 is updated about every 6 months. The first version with adaptive content was installed in January 1997. The version using the technology described in this paper has become operational the fall of 1997. A student is currently investigating how the adaptive linking introduced in the fall of 1996 [De Bra & Calvi 1997] has influenced the browsing behavior of student, as compared to the previous (non-adaptive) version described in [De Bra 1996]. Informal interviews with students have already confirmed that adaptive linking alone is insufficient, because hiding links without any additional explanation (which could be conditionally included) is frustrating for the reader.

Our first attempt to use the (Unix) C-preprocessor for the creation of adaptive content resulted in an awkward authoring environment in which two completely different syntaxes had to be mixed. This resulted in source texts that were difficult to write and read. The approach proposed in this paper uses standard HTML, which enables authors to use HTML editors (or generators) for writing adaptive hyperdocuments for and on the Web. Besides conditional constructs in HTML we rely on the software described in [De Bra & Calvi 1997] for conditionally hiding links. This significantly reduces the number of conditionals authors have to include in the source text of their documents. All the (current) software is written in Java. For performance reasons the Web server for the courseware had to be upgraded from a 486-66 to a Pentium-Pro 200 (both running Solaris x86 2.5.1).

6. References

[De Bra 1996] De Bra, P., Teaching Hypertext and Hypermedia through the Web. In: Proceedings of the WebNet'96 Conference, pp. 130-135, San Francisco, 1996. (URL: http://wwwis.win.tue.nl/ debra/webnet96/index.html)

[De Bra & Calvi 1997] De Bra, P., Calvi, L., Improving the Usability of Hypertext Courseware through Adaptive Linking. Proceedings of the Flexible Hypertext Workshop, Southampton, 1997. (URL: http://wwwis.win.tue.nl/ debra/flex97/)

[Brusilovsky 1996] Brusilovsky, P., Methods and Techniques of Adaptive Hypermedia. In: User Modeling and User-Adapted Interaction **6**: 87-129, Kluwer academic publishers, 1996. (URL: http://www.contrib.andrew.cmu.edu/ plb/UMUAI.ps)

[Brusilovsky et al. 1996a] Brusilovsky, P., Schwarz, E., Weber, G., ELM-ART: An intelligent tutoring system on World Wide Web. Proceedings of the Third International Conference on Intelligent Tutoring Systems, ITS-96, Montreal, 1996. (Lecture Notes in Computing Science, vol. 1086, pp. 261–269). (URL: http://www.contrib.andrew.cmu.edu/ plb/ITS96.html)

[Brusilovsky et al. 1996b] Brusilovsky, P., Schwarz, E., Weber, G., A Tool for Developing Adaptive Electronic Textbooks on WWW. Proceedings of the WebNet'96 conference, pp. 64-69, San Francisco, 1996.
(URL: http://www.contrib.andrew.cmu.edu/ plb/WebNet96.html)

[Calvi & De Bra 1997] Calvi, L., De Bra, P., Using Dynamic Hypertext to create Multi-Purpose Textbooks. (to appear) In: Proceedings of ED-MEDIA'97, Calgary, 1997. (URL: http://wwwis.win.tue.nl/ debra/ed-media97/)

[Kay & Kummerfeld 1094a] Kay, J., Kummerfeld, B., An Individualised Course for the C Programming Language. In: Proceedings of the Second International WWW Conference "Mosaic and the Web", Chicago, 1994.
(URL: http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/Educ/kummerfeld/kummerfeld.html)

[Kay & Kummerfeld 1994b] Kay, J., Kummerfeld, B., Adaptive Hypertext for Individualised Instruction. Workshop on Adaptive Hypertext and Hypermedia at the Fourth International Conference on User Modeling, 1994.
(URL: http://www.cs.bgsu.edu/hypertext/adaptive/Kay.html)

[Lemmens 1996] Lemmens, W.J.M., Use of Parameterised Hypertext Pages. Eindhoven Univ. of Technology, 1996.
(URL: http://wwwis.win.tue.nl/ wsinpim/parmpage.html)

[Rosis et al. 1994] de Rosis, F., De Carolis, B., Pizzutilo, S., User-Tailored Hypermedia Explanations. Workshop on Adaptive Hypertext and Hypermedia at the Fourth International Conference on User Modeling, 1994.
(URL: http://www.cs.bgsu.edu/hypertext/adaptive/deRosis.html)