Perspectives for Electronic Books in the World Wide Web Age

François Bry and Michael Kraus

Institute for Computer Science, University of Munich, Germany
http://www.pms.informatik.uni-muenchen.de

Abstract

While the World Wide Web (WWW or Web) is steadily expanding, electronic books (eBooks) remain a niche market. In this article, it is first postulated that (1) specialized contents and device independence can make Web based eBooks compete with paper prints, and that (2) adaptive features that can be implemented by client-side computing are relevant for eBooks, while more complex forms of adaptation requiring server-side computations are not. Then, enhancements of the WWW standards (specifically of XML, XHTML, of the style-sheet languages CSS and XSL, and of the linking language XLink) are proposed for a better support of client-side adaptation and device independent content modeling are proposed. Finally, advanced browsing functionalities desirable for eBooks as well as their implementation in the WWW context are described.

Keywords: Electronic Books, eBooks, Adaptive Hypermedia, World Wide Web, WWW, Style Sheets

1 Introduction

1.1 eBooks: Ubiquitous or Absent?

While the World Wide Web (WWW or Web) is steadily expanding, electronic books (eBooks) remain a niche market: Although all kinds of textual information is nowadays accessible on the WWW, leading eBook stores such as amazon.com (http://amazon.com), Barnes and Noble (http://bn.com), and eBooks.com (http://www.ebooks.com/) currently do not offer more than a handful titles [Knight, 2002]. Thus, while the WWW has made electronic documents ubiquitous, eBooks proper are almost absent from todays market for electronic devices and contents.

The reasons for this rather paradoxical situation have been much debated and the following causes are often given: A diversity of incompatible eBook standards (especially Adobe Acrobat eBook Reader, Gemstar, Microsoft Reader, and Palm Reader), the dependency of most eBooks to specific reading devices, and the prices, small screens and low display and/or rendering quality of reading devices.

1.2 A Future for eBooks: Specialized Contents and Device Independence

An increasing number of Web sites are offering mostly textual contents for special purposes: News such as the Web pages of daily or weekly papers and professional journals, technical documents such

as manuals, teaching material, and directories such as address and telephone directories. Often, the presentation and browsing of such specialized contents with conventional WWW browsers (such as Microsoft Explorer and Netscape Navigator) is not fully satisfying. For example, browsing news might require time dependent primitives, for reading teaching material adaptive functionalities might be needed, using directories might require application specific search tools. Nevertheless, the WWW is an appropriate medium for the distribution of such specialized electronic documents because it make them easily accessible to a wide audience using a wide range of widespread devices.

Arguably, specialized contents and device independence are two features that can make Web based eBooks compete with paper prints. Up till devices relying on new technologies such as eInk [eInk, 2002], [Comiskey et al., 1997], [Sheridon and Berkovitz, 1997] and ePaper [Gibbs, 1998], [Electronic Paper, 1999] and [D'Amico and Ferranti, 1999] provide at low prices with reading areas comparable in size and rendering quality to those of quality paper prints, device independent Web based documents intended for a professional usage (such as professional journals, technical manuals, directories, or teaching material) or aiming at fulfilling specific nonprofessional needs (such as directories and news servers) are likely to be among the most promising eBook applications.

1.3 The Technology Suggested: Client-Side Web Based Adaptation and Advanced Browsers

Web based eBooks are closely related to Web based adaptive systems [lovsky, 1996], [Nejdl and Wolpers, 1998], [Bra and Calvi, 1998], and [Albrecht et al., 2000]. At first it is difficult to distinguish between the two kinds of systems. Undoubtedly, adaptive capabilities such as adaptation to user profiles and/or user performances [lovsky, 1996] (for example, learning performances as far as teaching material is concerned, or skills if technical manuals are considered) should not be precluded for eBooks. Some other forms of adaptation (like adaptive features pertaining to computer supported cooperative work), however, do not belong to a widespread intuitive notion of a (paper or electronic) book. Let draw the border between "eBook relevant" and "eBook alien" Web based adaptation as follows: Forms of adaptation that can be realized by client-side computing will be considered "eBook relevant", while forms of adaptation requiring server-side computations will be considered "eBook alien".

Although this definition refers to the underlying technology, not to a reader's perception, it seems to make sense for the following reason. Adaptive functionalities that can be implemented client-side are by definition restricted to one reader's environment while adaptive features intrinsically requiring server-side computations necessarily refer to an environment wider than that of one reader. Intuitively, reading a (paper or electronic) book is an activity depending on no more than the reader's environment. While including additional reading material into one reader's environment is a common reader's experience, considering the actions or reading environments of other readers goes beyond a common reader's experience. Arguably, the definition of "eBook related" and "eBook alien" Web based adaptation postulated above conveniently extends to eBooks an intuitive notion of paper prints reading experience.

In this article, it is proposed to realize Web based eBook by relying on client-side Web-based adaptation and advanced browsing functionalities. Specifically, the following two complementary issues are investigated: Enhancements of the WWW standards for a better support of client-side adaptation, and advanced browsing functionalities desirable for eBooks and their implementation in the WWW context.

First, it is proposed to distinguish in XML [Bray et al., 2000] and XHTML [Pemberton et al., 2000]

between link model (defining link types based on semantic relationships between data) and browsing model (specifying a hypertext behavior for links). This distinction is used in enhancing XML and XHTML with features desirable for eBooks: hyperlink enabling, hyperlink disabling, and various forms of linkbase [DeRose et al., 2001] processing. Second, a simple extension (consisting in using the path selectors of style sheet languages such as CSS [Berts et al., 1998] and XSL [Clark, 1999]) is proposed for enhancing HTML and XML with adaptation. Third, advanced modeling and browsing functionalities desirable for Web based eBooks are proposed: *Author's views* for grouping semantically related, yet different contents and the tool *Reader's View* is described using which a reader can annotate and restructure an electronic book while exploring it.

1.4 Contents of this Article

This article consists of seven sections, the first of which is this introduction. In Section 2, desiderata for the techniques described are given. In Section 3, a few minimal changes of the hypertext model of XML (and of XHTML) are proposed. Section 4 is devoted to client-side adaptation in the (enhanced) XML/XHTML context. A notion of "semantic views" over a document desirable in modeling eBooks is introduced in Section 5. Advanced browsing functionalities needed for eBooks are discussed in Section 6. Section 7 is a conclusion.

This article builds upon the former reports [Bry and Kraus, 2002b], [Bry and Kraus, 2002c], and [Bry and Kraus, 2002a].

2 Desiderata

2.1 Keeping Standard Diversity Small

The proposals made in this article rely as much as possible on existing and already established WWW standards and technologies, instead of trying to introduce brand new standards and technologies. This means that, apart from small extensions, no new algorithms are needed.

2.2 Conservative Extension of Existing Standards

When changing standards, both upwards and downwards compatibility should be sought for. Upwards compatibility means that already existing data and related standards can still be used with the new, extended standard. This is achieved by extending, not altering the existing standard. Downwards compatibility means that data and related standards based on the new, extended standard can also be used with processors of the original standard.

2.3 Open to Future Standards

Extensions of existing standards that are open to changes of these standards and/or to future standards are desirable. Approaches relying upon essential features of existing standards (like e.g. path selectors as far as style sheet languages are concerned) are likely to remain compatible with future languages.

2.4 Rule-Based Approach to Expressing Adaptation

It is preferable to express context adaptation in a rule-based formalism than in an imperative language. Indeed, rule-based formalism are more declarative than imperative languages that hide actual presentation parameters behind assignments, loops, etc. Rules describe only a logic, no processing. The processing, or algorithm, can be kept the same when rules are refined.

2.5 Privacy and Security Concerns

Privacy and security are essential issues in the Web context. While advanced adaptive and browsing functionalities are desirable for eBooks, eBook readers should not have to disclose to content providers much of their preferences.

3 Enhancing the Hypertext Model of XML

The following proposals are based on the following metaphors: “Hyperspace as further layout dimension” and “Hiding elements in hyperspace”.

According to the first metaphor, adaptation can be seen as “hyperspace layout” where hyperspace is understood as including the time dimension and user and device profiles. Arguably, style sheets are appropriate for layout in the hyperspace as much as for layout in the two-dimensional printing space. A unified approach for layout in both spaces, hyperspace and printing space, is preferable to different approaches for two reasons: Ease of programming (for people), ease of processing (for computers).

In rendering a document, it is often beneficial not to render some of its elements as usual, but instead to make them accessible through hyperlinks, .ie. to “hide them in the hyperspace”. To achieve this without transformations of the document, it is proposed to extend the link languageXLink [DeRose et al., 2001] and the style sheet languages CSS [Berts et al., 1998] and XSL [Clark, 1999] of XML (and XHTML) with “hyperlink enabling”. Complementary to hyperlink enabling, disabling existing hyperlinks is often needed in conceiving adaptive systems.

3.1 Link Model vs. Browsing Model or Semantics vs. Navigation

It is first proposed to enhance XML [Bray et al., 2000] and XHTML [Pemberton et al., 2000] with a distinction between link model and browsing model as follows: A link model defines link types, possibly hyperlink types, based on semantic relationships between data; a browsing model specifies a hypertext behavior for links.

This distinction is analogous to the distinction between a generic markup language such as XML [Bray et al., 2000], whose purpose is to model the logical structure of data items, and a style sheet language such as CSS [Berts et al., 1998], whose purpose is to specify the layout for a given logical structure. It has been discussed for example in [Bra et al., 1992], [Bra et al., 1994], and [Halasz and Schwartz, 1994].

Link models and browsing models are usually not distinguished. For example, the HyTime model [HyTime, 1997], which inspired the link model XLink [DeRose et al., 2001] of XML, defines hyperlinks primarily in terms of the behavior of standard browsers (such as Netscape Navigator or Internet Explorer) to be used with standard size screens. If Web contents are to be rendered not only with standard browsers and standard size screens, then the distinction between link model and browsing

model is essential, for it makes a context-dependent (for example device-dependent) hypertext rendering of the same link model possible. The distinction between hyperlink model and browser model proposed here contributes to both, independence of data modeling from data usage and independence of data modeling from presentation devices.

Consider as an illustration the "behavior attribute" `show` of XLink. For simplicity assume a binary, directed hyperlink between a source data item and a target data item. The attribute show may have the value `show="undefined"` or one of the following three values:

- `show="replace"` means that the presentation of the source data item disappears from the window from which the hyperlink was actuated and instead a presentation of the target data item appears in this window;
- `show="new"` means that the presentation of the source data item remains unchanged and a presentation of the target data item appears in a new window;
- `show="embed"` means that a presentation of the target data item is integrated into the presentation of the source data item at the position where the hyperlink was actuated (in the case of data items consisting of text, this behavior is also known as stretch text).

A link model might define the following, semantically characterized, hypertext-free link types:

- "hyperlink to new information",
- "hyperlink to alternative descriptions of the current information",
- "hyperlink to supplementary information".

A browsing model could then specify that these three types of hyperlinks behave in the three ways described above. Different browsing models may specify the behavior differently, depending on the device or more generally on the context.

3.2 Hyperlink Enabling

In rendering a document it might be beneficial to exchange its "structure dimension" for the "hyperspace dimension", i.e. to make a subelement (for example a section) reachable through a hyperlink. This appear useful for example if the document is to be rendered on a small screen such as a PDA or a mobile phone.

The anchor of the element to be "hidden in the hyperspace" could be specified in the browsing style sheet. The browsing style sheet language could also make it possible to use the first two or three (or generally, n) words of the element, possibly followed with a sign such as an arrow, as an anchor for the element hidden in the hyperspace.

Style sheet languages such as CSS [Berts et al., 1998] and XSL [Clark, 1999] can easily be conservatively enhanced so as to enable a hyperlink on a certain element.

3.3 Hyperlink Disabling

Disabling a hyperlink is to be understood here as rendering the link anchor (text or image) while removing all hyperlink functionalities. Hyperlink disabling makes it possible to prevent a user from

traversing a link which is not relevant to his current context. For example, hyperlinks to advanced information can be disabled for novice users, hyperlinks to large graphics can be disabled for personal digital assistant (PDA) users, and a hyperlink to a patient's health insurance details can be disabled when, while making a diagnosis, a physician is consulting the patient's medical record. Note that currently none of HTML, XLink, CSS, and XSL/XSL support hyperlink disabling.

Hyperlink disabling is one possible hyperlink presentation that can be expressed using an XLink [DeRose et al., 2001] attribute. New attribute/value pairs can be introduced into CSS to express such a hyperlink presentation.

The following presentations facets are desirable for versatile adaptive hypermedia systems:

- *Normal:* traversable, anchor rendered as hyperlink
- *Untraversable:* not traversable, anchor rendered as hyperlink
- *Hidden:* not traversable, anchor rendered as normal text or image
- *Invisible:* not traversable, anchor not rendered

Note that only the first of these facets is currently specified by the XLink [DeRose et al., 2001] or XHTML [Pemberton et al., 2000] standard. Note also that enhancing XLink or XHTML with these facets would require neither significant changes of the standard, nor to enhance browsers with any complex processing.

3.4 Linkbase Processing

Context adaptation is likely to be especially useful in rendering eBooks having a complex structure of links, involving not only simple links à la HTML but possibly also complex links (i.e. links associating more than two resources, inbound and third-party links [DeRose et al., 2001]). Complex link structures are better specified using linkbases [DeRose et al., 2001]. Therefore, the processing of linkbases is likely to become soon an essential practical aspect of the development of Web based eBooks.

There are several manners for a browser to process linkbases, a topic which is not covered by the current XLink standard [DeRose et al., 2001]. It is worth to distinguish between the set of hyperlinks that are defined in each document (or in the linkbases it refers to) and the hyperlinks that the browser has knowledge about, what we refer to as a *browser's internal linkbase.* A browser's internal linkbase contains not only those links defined in the document (or in the linkbases this document refers to) it currently renders. A browser's internal linkbase also contains those links collected from previously loaded linkbases, for example from those linkbases referred to by formerly rendered eBooks. The issue is to specify what the browser's internal linkbase should be at each point of time.

A *closed world linkbase processing scenario* assumes that it is possible to have knowledge about all the hyperlinks relevant to the application, even if not all of them are referenced by the document currently being displayed. If this scenario is applicable, then the browser's internal linkbase might consist at any time of all the hyperlinks relevant to the application. While this scenario is surely not applicable to the open World Wide Web, it makes sense for eBooks, electronic libraries, and electronic encyclopediae.

In contrast, an *open world linkbase processing scenario* assumes that it is not possible to have knowledge of all the hyperlinks relevant to the application. This assumption applies for example to

the WWW. It applies also to eBooks with contents referring to resources outside the eBook, e.g. a textbook referring to a (electronic) user manual.

Under the open world linkbase processing assumption, two different approaches can be retained: *inflationary linkbase management*, in which case the browser's internal link base continously grows during a session, and *conservative linkbase management.* The first approach has the drawback of possibly yielding different renderings of the same web page at different times of a same session (if a page visited after this page has added new hyperlinks or hyperlink arcs to or from this page). A possible drawback of the second approach is to preclude forward hyperlink references, thus confining applications to hierarchically definable hyperlink structures.

The scenarii defined above, closed world linkbase processing, open world linkbase processing with inflationary linkbase management, and open world linkbase processing with conservative linkbase management, specify the border cases of a large space of mixed possibilities: it surely makes sense for some eBooks to treat some links according to the closed world assumption, other links according to the open world assumption under inflationary management, and links of a third kind according to the open world assumption under conservative management. Also, the linkbase processing retained might be context-dependent. Clearly, it could make sense for some eBook to choose a link restrictive linkbase processing if the rendering is to be done on a small screen.

4 Client-Side Adaptation

4.1 Browsing Contexts: A Data Structure for Expressing User Models

XML [Bray et al., 2000] and XHTML [Pemberton et al., 2000] have no means to express a user model, i.e. information about the user like browsing history, user preferences, performances on exercises, answers to questionnaires, etc. This section first proposes a data structure called *browsing context*, which allows such information to be stored by the browser, i.e. on the client side, to be accessed through style sheets, and to be updated through web applications using scripting languages like Javascript [ECMA-262, 1999]. A browsing context consists of three kinds of information that can be distinguished according to its acquisition: *browsing history data*, *browsing environment data* and *application data*.

Browsing history data contains information about the browsing actions the user has performed in the past, that is, visiting web pages, traversing hyperlinks, opening and closing windows, etc. This information is automatically generated by the browser and is updated each time the user performs a browsing action. Although current web browsers do not offer adaptive hypermedia functionalities, they already store some of the information that is typically part of user models in adaptive hypermedia systems, for example browsing history or user preferences. However, the storage of this information is browser-specific, and therefore it is not possible with current Web standards to access this information through style sheets or scripting languages.

Browsing environment data contains information about the device (hardware), browser (software), location, time, language, etc. Like browsing history data, this information is automatically generated and updated by the browser, if necessary. In contrast to browsing history data, browsing environment data is not affected by browsing actions. Some of the information does not change during a browsing session, for example device and browser information. Other information, for example location and time, may change during a browsing session.

Application data contains information specific to the web application being browsed by the user.

In the case of an electronic tutor system, for example, this can be the user performances on exercises, like the numbers of correct and wrong answers. In an e-commerce catalogue, application data can comprise information about number and type of items in a shopping cart. Another possibility are the user's answers to questionnaires. This information cannot be automatically generated by a browser, because it lies beyond the markup languages XHTML [Pemberton et al., 2000] and XML [Bray et al., 2000]. Therefore it has to be provided by the application itself.

Using style sheets and scripting languages in conjunction with a browsing context offers the possibility to quite easily implement an adaptive eBook. This is described in detail below in section 4.2. For accessing the browsing context with style sheets and scripting languages in a way that is natural to these approaches, it is necessary to store the browsing context in XML format. The following is an example of a browsing context represented as an XML document:

```
<browsingcontext xmlns="http://www.browsingcontext.com">
  <!-- browsing history data -->
  <webpage uri="www.cdshop.com/index.html">
    <webpage uri="www.cdshop.com/search.html">
      <webpage uri="www.cdshop.com/rockpop-25.html">
        <webpage uri="www.cdshop.com/search.html"/>
        <webpage uri="www.cdshop.com/cart.html"
                 target="_new"/>
     </webpage>
    </webpage>
  </webpage>

  <!-- browsing environment data -->
  <device>Desktop</device>
  <browser>IE</browser>
  <os>Windows</os>
  <country>Germany</country>
  <virtuallocation>Home</virtuallocation>
  <time>23:23</time>
  <language>German</language>

  <!-- application data -->

  <shoppingcart>
    <item artist="Kate Bush" title="Lionheart" price="9.99"/>
    <item artist="Steve Hackett" title="Live Archive" price="29.95"/>
    <item artist="Nik Kershaw" title="to be Frank" price="14.99"/>
  </shoppingcart>
</browsingcontext>
```

In this example, the user has started his browsing session visiting the online catalogue of a CD shop (`www.cdshop.com/index.html`). From there, he has navigated to a search page (`www.cdshop.com/search.html`). The search has resulted in a certain page about rock and pop music (`www.cdshop.com/rockpop-25.html`). Then the user has navigated again to the search page and has opened a Web page with his shopping cart in a new browser window (`www.cdshop.com/cart.html`). The browsing environment data contains information about the user's type of device (desktop), location (Germany/home), current time (23:23), etc. Finally, the online cataolgue stored data about the user's shopping cart (three items) in the application data part of the browsing context.

In common adaptive hypermedia systems, the structure of this information, the information itself, and the way of information acquisition together form a user model. This paper does not propose

a specific user model, but a framework relying upon HTML and XML that allows a simple implementation of user models. The main advantage of this framework is to make adaptive hypermedia techniques available in the Web context at low cost, that is, with minimal changes of the existing standards. As HTML and XML are *the* standards for the World Wide Web, a slight modification of these standards providing adaptation functionalities gives chance to a widespread, non-proprietary way of implementing adaptive eBooks as Web applications.

Web browsers store an internal representation of the document currently being displayed, for example as a DOM [Apparao et al., 1998] tree. This document is referred to in the following as *"naked document"* because it does not contain any browsing context information. In a similar way as this "naked document" is stored by the browser, a *browsing context document* like the one presented above can also be stored by the browser. The information contained in the browsing context document is somewhat similar to the information contained in the `head` element of an HTML document: The content of the `head` element is not rendered directly by the browser (as opposed to the content of the `body` element), but it is meta information about the document like description, keywords, character encoding, etc. Similarly, the content of the browsing context document, that is, browsing history data, browsing environment data and application data, can also be seen as meta information. This meta information can be used by the browser to adapt the rendering of the "naked document", as it is described in section 4.2.

Both the "naked document" and the browsing context document can be considered as the two parts of one (virtual) *context enriched document* stored within the browser. In the case where the "naked document" is an HTML document, a context enriched document might look like this:

```
<!-- context enriched document -->
<root>
  <!-- original browsing context document -->
  <browsingcontext xmlns="http://www.browsingcontext.com">
    <!-- browsing history data -->
    <!-- browsing environment data -->
    <!-- application data -->
  </browsingcontext>

  <!-- original "naked document" -->
  <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
      <!-- meta information -->
    </head>
    <body>
      <!-- content to be rendered -->
    </body>
  </html>
</root>
```

The context enriched document takes over the role of the original "naked document" within the browser, that is, style sheets are applied to the context enriched document instead of the "naked document", scripting languages have access to the DOM tree of the context enriched document instead of the DOM tree of the "naked document", etc. In fact, the context enriched document is a virtual document combining a browsing context (using which adaptation is expressed) with a standard HTML or XML document. Note that the approach described below does not require the materialization of this virtual document.

4.2 Style Sheets for Content and Navigation Adaptation

This section describes a simple extension to style sheet selectors making it simple to implement adaptive hypermedia functionalities with HTML [Pemberton et al., 2000] and XML [Bray et al., 2000]. The path expression of a style sheet selector is not to be matched against the original "naked document" tree, but against the new *context enriched document tree*, as it has been introduced in the last section.

Typical Web style sheet languages like CSS [Berts et al., 1998] and XSL [Clark, 1999] have constructs of two kinds: style rules and selectors. Style rules define certain presentation parameters for elements in the document tree (like fonts, colors and margins), and transformations of the document tree (like insertion and sorting of elements). Selectors are path expressions that determine which style rule is applied to which element in the document tree. The following CSS rule specifies that all `strong` elements inside a `h2` element shall be rendered with red color:

```
h2 strong { color: red }
```

If the path expression of a style sheet selector is not matched against the original "naked document" tree, but against the new *context enriched document tree*, as it is proposed in this paper, then path expressions can be built that depend on the content and structure of both, the "naked document" and the browsing context document. The following (extended) CSS rule specifies that if the user has previously visited the Web page with URI `http://www.cdshop.com/index.html`, all `em` elements inside a `h1` element shall be rendered with a sans-serif font, otherwise the style rule does not apply:

```
bc:webpage[uri="http://www.cdshop.com/index.html"] html:h1 html:em
{ font-family: sans-serif }
```

This style rule is processed as follows. The path expression matching algorithm (that of CSS in this case) has to recognize that the first component of the expression (`bc:webpage[...]`) refers to the browsing context (which is part of the context enriched document). This can be achieved by using XML namespaces [Bray et al., 1999], as it is shown here. The matching algorithm for the expression component itself remains unchanged, that is, the example matches if there is an element with name `webpage` that has an attribute with name `uri` and value `http://www.cdshop.com/index.html`. The next two components of the expression (`html:h1` and `html:em`) refer to the "naked document" (which is also part of the context enriched document), which again can be deduced from the namespace. These components can be matched like conventional CSS selector expressions. The whole matching algorithm works as if it is applied to the context enriched document instead of the "naked document", as it is in conventional style sheet processors. Note that it is not necessary to actually materialize the context enriched document for the processing as it is shown here.

The approach shown here uses XML namespaces [Bray et al., 1999] (`bc` in the example above) to distinguish between those components of a selector referring to the "naked document" and those components referring to the browsing context. It would also be possible, for example, to introduce new tokens to the CSS selector language that allow for such a distinction. The advantage of using namespaces, however, is that namespaces are already part of the CSS selector language and therefore the selector language itself can be left unchanged, only the processing of selector expressions is slightly modified.

Note that if the path expression of a selector contains no parts referring to a browsing context, the semantics of a style rule remains unchanged. That is, it is not affected by the introduction of the concept of browsing contexts as described in this paper. For example:

```
h1 em { font-family: serif }
```

Similarly to the first example in this section, this CSS rule specifies that all `em` elements inside a `h1` element shall be rendered with a serif font. Assuming that the default namespace is depicting the "naked document" (and not the browsing context), the semantics of this style rule is the same whether it is defined in a system that is capable of handling browsing contexts, or in a conventional, non-browsing context system. This means that the approach described here ensures upwards compatibility of existing CSS style sheets with browsing context aware systems.

The proposed extension makes also possible to mix style rules with both, browsing context and conventional selectors in a single style sheet, for example:

```
h1 em { color: red }
bc:webpage[uri="http://www.cdshop.com/search.html"] html:h1 html:em
{ color: blue }
```

In this example, all `em` elements inside a `h1` element shall be rendered with red color (first rule). But, if the user has previously visited the Web page with URI `http://www.cdshop.com/search.html`, all `em` elements inside a `h1` element shall be rendered with blue color (second rule). The conflict resolution necessary in this case is already built-in in CSS: According to CSS semantics, the second rule has a higher priority than the first rule, because it is more specific, thus resulting in blue text in the second case.

Note that browsing context style sheets are also downwards compatible with non-browsing context capable processors. Style rule components referring to the browsing context part of the context enriched document will never match using the conventional CSS selector matching algorithm, because the "naked document" (which is the only document in the case of conventional CSS processors) does not contain elements of the browsing context namespace.

All examples that have been given so far are (extended) CSS rules. In fact, all style sheet languages that build on path selectors can be extended in a similar way as proposed above. In case of XSL, for example, the selector language is XPath 1.0 [Clark and DeRose, 1999]. XPath 1.0 or XPath 2.0 [Clark and DeRose, 1999] is extended in a similar way as it has been described for CSS selectors. In the following example, `info` elements from the XML source are transformed into an HTML `h2` heading (first template rule). If the user has previously visited the Web page with URI `http://www.cdshop.com/cart.html`, a hyperlink to that Web page is added after the heading. In this example, the assumed intention is to prevent novice users, i.e. those that have not visited the CD shop's catalogue so far in this case, from information overloading by presenting too much links to them:

```
<xsl:template match="//info">
  <html:h2><xsl:value-of select="title"/></html:h2>
</xsl:template>
<xsl:template
match="//bc:webpage[uri="http://www.cdshop.com/cart.html"]/info">
  <html:h2><xsl:value-of select="title"/></html:h2>
  Click <a href="http://www.cdshop.com/cart.html">here</a> to go
  directly to your shopping cart.
</xsl:template>
```

Note that the browsing context-enhanced XPath expression from the `match` attribute is strictly speaking not a path expression any more: The XPath processor has to recognize that the first part of the expression (`bc:webpage[...]`) refers to the browsing context part of the context enriched document (with its own root element), whereas the second part of the expression (`info`) refers to the "naked document" part (with its own root element). The matching algorithm for each of the two parts remains unchanged, however. Assuming the XML representation of the context enriched document from the previous section, the XPath expression could be rewritten like this:

```
//bc:webpage[uri="http://www.cdshop.com/cart.html"]/following::info
```

This expression would also work with conventional XPath processors, but it is clumsy compared to the previous one. Therefore, the expression from the above example could be considered as a short form of this conventionally valid XPath expression. Both, a fully XPath conform implementation of browsing contexts, and a more intuitive approach, although requiring a slight modification of XPath processing, are possible. Are more detailed discussion of this topic is out of the scope of this paper, however.

4.3 Possible Extensions

4.3.1 Updating Application Data using Scripting Languages

Using style sheet selectors to express content and navigation adaption, as it is described in this article, is not sufficient for modeling certain complex aspects of adpative hypermedia systems. Still missing is the possibility for such systems to store data in the browsing context, which then could be used by style sheets as a source of adaptation. As previously mentioned, scripting languages like Javascript can be used to achieve this. In a similar way as Javascript code contained in web pages can change the ("naked") document tree, Javascript code contained in web pages can change the content of a browsing context's application data. However, an in-depth discussion of this topic is out of the scope of this paper.

4.3.2 Modeling Locations

As stated in Section 4.1 above, there are several different notions of location. First, a location can be information about the country or region where the user is, like Germany or France. This information is available in desktop computer systems and does not change during a browsing session. Second, a location can be information about the geographical position of the user, expressed for example as longitude and latitude. This information is available in mobile devices like cellular phones or PDAs with special positioning equipment such as GPS (Global Positioning System). Third, a location can be information about *virtual locations* like home, car, office, meeting, etc. However, virtual locations are not represented in today's computer devices and with current web standards. All of these notions, for example geographical position and virtual locations, can be represented simultaneously as browsing environment data in a browsing context.

5 Semantic Modeling: Author's Views

5.1 Modeling Author's Views

Author's views are inspired from database views. In many databases, views ensure the conceptual independence between the primary data stored in the database and the various interpretations needed for applications using the database. Common database views are partial look-ups of the data stored. Most database applications make use of views of some sort.

Views also make sense in eBooks, especially in eBooks with complex structures and/or contents. Such eBooks, for example, technical manuals and textbooks, are often read at different *semantic levels*. This is for example the case of maintenance manuals that can be read by both, an engineer for estimating the time needed for performing some operation, and a worker for learning how to perform the operation. It is also the case with textbooks that can be read by a teacher preparing a lecture, or by a student learning for an examination. The two readers will probably focus on different parts. A teacher might, for example, focus on the arguments of the introductory sections and skip proofs or explanations he is already familiar with, or restrict his attention to overviews of these proofs and explanations, if such overviews are available. A student, in contrast, is likely to skip the introductory parts and to devote more attention to detailed expositions of proofs and explanations.

Modeling each view over a text as a separate document, i.e. modeling one document for each view – such as a *teacher-view* document and a *student-view* document in the above-mentioned example – would have several drawbacks. This would induce redundancies in cases where distinct views coincide on some text items. Because of these redundancies, updating errors could happen. Hence, the consistency between the different texts giving complementary *views* of the same content would be difficult to maintain.

It is preferable to group together the various views of each text item into a single document. I.e. a document with views should not be structured as a sequence of views, but instead retain the structure common to all the view fragments and group together the views corresponding to the same text items. Doing so, the resulting document consists of a sequence of `views-groups` elements, each collecting the items related to the various views.

This approach is illustrated on an example as follows. In this example, three possible views over this very article are considered:

- a *detailed* view which consists in this article,
- a *digest* view which is a shortened digest of this article,
- a *slide* view to be formated into slides for a screen presentation of this article.

```
<article>
 <title>Perspectives for Electronic Books in the World Wide Web Age</title>
 <abstract>
  <views-group>
   <view type="detailed">
      While the World Wide Web (WWW or Web) is steadily expanding,
      electronic books (eBooks) remain a niche market: Although all
      kinds of textual information [...] the dependency of most
      eBooks to specific reading devices, and the prices, small
      screens and low display and/or rendering quality of reading
      devices.
   </view>
```

```
   <view type="slide">Perspectives for Electronic Books in the World
      Wide Web Age
      <list>
        <item>Author's views</item>
        <item>Browsing style sheets</item>
        <item>Reader's views</item>
      </list></view>
   </views-group>
  </abstract>
  <section>
    <title>Introduction</title>
    <views-group>
      <view type="detailed">[...]</view>
      <view type="digest">[...]</view>
      <view type="slide">[...]</view>
    </views-group>
  </section>
 [...]
</article>
```

The elements of type `views-group` are grouping constructs for collecting together various views over a text item – such as the abstract of this article. The content of a `views-group` element consists of `view` elements. The values of the `type` attribute of a `view` element serve to name the views involved: these values can be freely defined by the XML application designers.

The overall structure of the document in the example above is that of the initial article, i.e. the present article. This structure is shared by the three views over the document. Sharing this structure contributes to facilitate the maintenance of the document. Indeed, structure changes do not have to be repeated in different view documents, what would be error prone, and a content change in one view item suggests to perform the corresponding changes in the associated other view items, a feature which could be supported by *view-aware* XML tools or editors. Such an editor could for example display all views in parallel, each in a separate window, and prompt the author after an update of one view item for possible updates of the corresponding items of the other views.

The idea of adaptive presentation in hypertexts has already been investigated in e.g. [Halasz and Schwartz, 1994], [Bra et al., 1992], and [Bra et al., 1999]. The main advantage of the approach described above is the simplicity of its integration into the XML/HTML world.

5.2 Related Notions: Ruby and Footnotes

The `views-group` and `view` constructs described above in Section 5.1 remind of the ruby elements [M. Sawicki et al., eds., 2001] recently added to the latest specification of XHTML [Pemberton et al., 2000] and CSS3 [Berts et al., 1998]. This is not by chance: Views and ruby annotations are closely related notions.

Ruby annotations, short *rubys*, are short annotations to a base text used in Japanese and Chinese documents to indicate pronunciation or for some other purpose. Ruby annotations are usually formatted alongside the base text using a smaller typeface (the name *ruby* stems from a typeface frequently used for such annotations).

A first difference between views, as considered in this paper, and ruby, as specified in [M. Sawicki et al., eds., 2001], is that views do not assume a *base text* while the specification [M. Sawicki et al., eds., 2001] does. This difference is not significant, however, since the semantic

labeling provided by the values of the type attribute can be used for marking a view as *base text*. Another difference is that ruby annotations are in general very short, typically a few words long, while views often extend over several sentences or paragraphs. This difference, however, is a difference of usage, not of principle. Thus, views as considered in the present article can be seen as a generalization of structural ruby annotations [M. Sawicki et al., eds., 2001].

Author's views, as defined above in Section 5.1, can also be applied to specify *structural footnotes*. While classical footnotes are connected to a position in the text they refer to, we call a *structural footnote* a text extension attached to a portion of a base text. In modeling texts with complex semantics, it is often preferable to characterize precisely the portion of text a *footnote* refers to. This increases the accuracy of the *footnote*. Structural footnotes can be supported by view-aware XML browsers in various manners: For example, the request to display the footnote – as a stretch or pop-up text – might be accompanied by a highlighting of its base text. In addition, views as defined above, provide with a framework for a semantic typing of *footnotes*. Both notions, structural footnotes and semantic typing of footnotes contribute to a better modeling and reading of eBooks.

6 Advanced Browsing: Reader's View

6.1 Restructuring While Reading

Reader's View is a tool developped at the Institute for Computer Science of the University of Munich with which a reader can restructure a document (within certain limitations given by the document's author or the browser). *Reader's View* is somewhat similar to an XML editor included within a browser, but with one difference: *Reader's View* leaves the original source document unchanged. For example, when the reader decides to leave out a certain paragraph of text (cf. below), this paragraph will not be erased from the source document, but *Reader's View* stores the information not to display this paragraph anymore.

Let call "reader's views" the views which can be built using the tools *Reader's View*. Despite of their names, reader's views are fundamentally different from author's views (cf. above Section 5.1). Author's views are created by the *author* of the document at writing time. A reader's view, in contrast, is created by the *reader* of the document while browsing.

6.2 Features

Using a browsing tool such as *Reader's View*, the nodes of the source document tree may be selected or rearranged. For example, a certain node can be left out or put in front of another node. Restrictions to such operations can be expressed. For example, one can forbid restructuring violating parent-child relationships or the selection of parts (rather than the whole) of nodes.

Explicit construction of a reader's view allows the reader to *pick up* certain nodes from the source document and add them in any order to the reader's view under construction. The nodes are not copied from the source document to the reader's view, but referenced via links.

Thus, the tool *Reader's View* can be seen as the generalization of today's major browsers' bookmark functionality. Traditional bookmark tools make it possible to store a set of links. With Reader's View one can in addition store arbitrary nodes, in particular text nodes written by the reader, as well. Such nodes can be linked to nodes of the document(s) from which the reader's view was constructed, thus yielding a powerful annotation mechanism.

Moreover, the nodes added by a reader can contain their own links. These links may point to nodes contained in the original source documents or to nodes that are only contained in the reader's view under construction. Thus, using *Reader's View* for browsing an eBook a reader can build up his own structured document containing new text and links as well as text, links and structure from the original source document.

6.3 Implementation with XLink

A reader's view is simply an XML document. This means that it can be viewed with any browser, even if the browser is not reader's view-aware. Specific to reader's views is how they are created. This is a major difference to ordinary annotation systems like [Annotea, 2001], which rely on special annotation servers.

If a reader selects a certain node from the original document, an outbound link with the selected node as its ending resource is created. This link is inserted into the reader's view. If the user wants a certain node not to be included in the reader's view, no link is created for that node.

The behavior attributes of XLink [DeRose et al., 2001] or of a browsing context (cf. Section 4.1) can be used to implement the features of a reader's view. They allow a reader to see a reader's view document as if its nodes were copied from the original document, even though they are only referenced by links.

Nodes created by the reader, i.e. "annotations", cannot be referred to in a reader's view document just as links, as the nodes' content has to be stored as well. The content of such nodes is copied into the reader's view document as it is entered by the user.

7 Conclusion

In this article, an approach to Web based eBooks has been proposed based on the postulates that (1) specialized contents and device independence can make Web based eBooks compete with paper prints, and that (2) adaptive features that can be implemented by client-side computing are relevant for eBooks, while forms of adaptation requiring server-side computations are not. This approach has several interesting properties.

First, the changes to Web standards proposed in this article are conservative extensions of existing Web standards, i.e. they ensure both, upwards and downwards compatibility (cf. Section 2.2). The proposal for realizing context adaptation, for example, is not specific to CSS [Berts et al., 1998] or XSL [Clark, 1999], it relies only on path selectors, which play a central role in WWW standards. The same approach can easily be applied to other or future style sheet languages or to other WWW standards like XML query languages, as long as they build on path selectors. Note also that this approach is stable against the changes from XPath 1.0 to XPath 2.0 [Clark and DeRose, 1999].

Second, the use of style sheet rules for expressing context adaptation makes it possible to formulate context-sensitive presentation in a declarative way, as opposed to Javascript, for example. Using Javascript, it is also possible to implement certain kinds of adaptation functionalities, but this is done using an imperative programming language.

Third, because the style sheet based approach proposed in this article relies on client-side computations only and does not require any processing on the server side, it does not face privacy and/or security problems as conventional approaches do, for example the combination of cookies, i.e. client-side user identification, PHP [PHP, 2002] and URIs. With the approach advocated for in this paper,

the whole processing takes place within the browser. Therefore no private or sensible data has to be transferred to and from the server, what would compromize privacy or security.

Acknowledgments

The authors are thankful to Norbert Eisinger and to anonymous referees for useful suggestions.

References

[Albrecht et al., 2000] Albrecht, F., Koch, N., et al. (2000). Smexweb: an Adaptive Web-based Hypermedia Teaching System. *International Journal of Continuing Engineering Education and Life-Long Learning*.

[Annotea, 2001] Annotea (2001). Annotea Project. http://www.w3.org/2001/Annotea/.

[Apparao et al., 1998] Apparao, V., Byrne, S., et al. (1998). Document Object Model (DOM) Level 1 Specification Version 1.0. W3C Recommendation. http://www.w3.org/TR/REC-DOM-Level-1.

[Berts et al., 1998] Berts, Lie, H. W., et al. (1998). Cascading Style Sheets, level 2. W3C Recommendation. http://www.w3.org/TR/REC-CSS2.

[Bra and Calvi, 1998] Bra, P. D. and Calvi, L. (1998). AHA: A Generic Adaptive Hypermedia System. In *2nd Workshop on Adaptive Hypertext and Hypermedia, HYPERTEXT'98*.

[Bra et al., 1992] Bra, P. D., Houben, G., and Kornatzky, Y. (1992). An Extensible Data Model for Hyperdocuments. In *4th ACM Conference on Hypertext*, pages 222–231. http://wwwis.win.tue.nl/~debra/echt92/final.ps.

[Bra et al., 1994] Bra, P. D., Houben, G., and Kornatzky, Y. (1994). A Formal Approach to Analyzing the Browsing Semantics of Hypertext. In *Proc. CSN-94*. http://wwwis.win.tue.nl/~debra/csn94/csn94.ps.

[Bra et al., 1999] Bra, P. D., Houben, G.-J., and Aham, H. W. (1999). Aham: A Dexter-based Reference Model for Adaptive Hypermedia. In *Proceedings of ACM Hypertext '99*, pages 147–156, Darmstadt, Germany.

[Bray et al., 1999] Bray, T., Hollander, D., et al. (1999). Namespaces in XML. World Wide Web Consortium. http://www.w3.org/TR/REC-xml-names.

[Bray et al., 2000] Bray, T., Paoli, J., Sperberg-McQueen, C. M., and Maler, E. (2000). Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation. http://www.w3.org/TR/REC-xml.

[Bry and Kraus, 2002a] Bry, F. and Kraus, M. (2002a). Adaptive Hypermedia made simple using HTML/XML Style Sheet Selectors. In *2nd Int. Conf. on Adaptive Hypermedia and Adaptive Web Based Systems (AH'2002)*. http://www.pms.informatik.uni-muenchen.de/publikationen/#PMS-FB-2002-1.

[Bry and Kraus, 2002b] Bry, F. and Kraus, M. (2002b). Advanced Modeling and Browsing of Technical Documents. In *17th ACM Symposium on Applied Computing (SAC 2002)*. http://www.pms.informatik.uni-muenchen.de/publikationen/#PMS-FB-2001-11.

[Bry and Kraus, 2002c] Bry, F. and Kraus, M. (2002c). Style Sheets for Context Adaptation. W3C Workshop on Delivery Context. http://www.pms.informatik.uni-muenchen.de/publikationen/#PMS-FB-2002-3.

[Clark, 1999] Clark, J. (1999). XSL Transformations (XSLT) Version 1.0. W3C Recommendation. http://www.w3.org/TR/xslt.

[Clark and DeRose, 1999] Clark, J. and DeRose, S. (1999). XML Path Language (XPath) Version 1.0. W3C Recommendation. http://www.w3.org/TR/xpath.

[Comiskey et al., 1997] Comiskey, B., Albert, J. D., Polito, B., and Jacobson, J. (1997). Electrophoretic Ink: A Printable Display Material. In *Proceedings of the Society for Information Display*, Boston, MA.

[D'Amico and Ferranti, 1999] D'Amico, M. L. and Ferranti, M. (1999). Xerox, 3M Collaborate on Electronic Paper. http://www.windowstechedge.com/wte/wte-1999-07/wte-07-paper_p.html.

[DeRose et al., 2001] DeRose, S., Maler, E., and Orchard, D. (2001). XML Linking Language (XLink) Version 1.0. W3C Recommendation. http://www.w3.org/TR/xlink/.

[ECMA-262, 1999] ECMA-262 (1999). Standard ECMA-262. ECMAScript Language Specification. ftp://ftp.ecma.ch/ecma-st/Ecma-262.pdf.

[eInk, 2002] eInk (2002). eink.com. http://www.eink.com/.

[Electronic Paper, 1999] Electronic Paper (1999). Electronic Paper. http://www.parc.xerox.com/dhl/projects/epaper/.

[Gibbs, 1998] Gibbs, W. W. (1998). The Reinvention of Paper. Scientific American. http://www.sciam.com/1998/0998issue/0998techbus1.html.

[Glazman et al., 2001] Glazman, D., Çelik, T., et al. (2001). Selectors. W3C Candidate Recommendation. http://www.w3.org/TR/css3-selectors.

[Halasz and Schwartz, 1994] Halasz, F. and Schwartz, M. (1994). The Dexter Hypertext Reference Model. *Communications of the ACM*, 37(22):30–39.

[HyTime, 1997] HyTime (1997). *Information technology – Hypermedia/Time-based Structuring Language (HyTime)*. International Standard ISO/IEC 10744, second edition edition. http://www.ornl.gov/sgml/wg8/docs/n1920/html/n1920.html.

[Knight, 2002] Knight, J. (2002). Who's Winning the Format War? *Electronic Book Web*. http://12.108.175.91/ebookweb/stories/storyReader$1205.

[lovsky, 1996] lovsky, P. (1996). Methods and Techniques of Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, 6(2-3):87–129.

[M. Sawicki et al., eds., 2001] M. Sawicki et al., eds. (2001). Ruby Annotation. W3C Recommendation. http://www.w3.org/TR/ruby/.

[Nejdl and Wolpers, 1998] Nejdl, W. and Wolpers, M. (1998). KBS Hyperbook – A Data-driven Information System on the Web. Technical report, University of Hannover, KBS Institute.

[Pemberton et al., 2000] Pemberton, S. et al. (2000). XHTML[tm] 1.0: The Extensible HyperText Markup Language. W3C Recommendation. http://www.w3.org/TR/xhtml1.

[PHP, 2002] PHP (2002). PHP – Hypertext Preprocessor. http://www.php.net/.

[Sheridon and Berkovitz, 1997] Sheridon, N. K. and Berkovitz, M. A. (1997). The Gyricon – A Twisting Ball Display. In *Proceedings of the Society for Information Display*, page 289, Boston, MA.

Biography

François Bry, born 1956, is currently investigating database methods and applications emphasizing semistructured data, document modeling, and query answering. Formerly, he worked on deductive databases, logic programming, and automated theorem proving. Since 1994, he is a full professor at the Institute for Computer Science of the University of Munich, Germany. Before joining the University of Munich, he has been working with the European Computer-Industry Research Centre (ECRC), Munich, and a few other companies in Paris, France. In 1981 he received a PhD from the University of Paris. He has been visiting at several university and research centers, including ICOT in Tokyo. He contributed to scientific conferences as an author, program committee member or chairman. He enjoys biking, hiking, and traveling.

Michael Kraus, born 1975, is working on his PhD in the field of web application modeling at the Institute for Computer Science of the University of Munich, Germany. Currently, he is spending two years as a visiting researcher at the W3C office at Keio University, Japan. In his diploma thesis, he developed a toolkit for advanced XML browsing functionalities.