Adaptive Hypermedia made simple using HTML/XML Style Sheet Selectors

François Bry and Michael Kraus

Francois.Bry@informatik.uni-muenchen.de
Michael.Kraus@informatik.uni-muenchen.de
Institute of Computer Science, University of Munich
Oettingenstrasse 67, 80538 Munich, Germany
http://www.pms.informatik.uni-muenchen.de/

Abstract. This paper addresses enhancing HTML and XML with adaptation functionalities. The approach consists in using the path selectors of the HTML and XML style sheet languages CSS and XSLT for expressing content and navigation adaptation. Thus, the necessary extensions of the selector languages are minimal (a few additional constructs suffice), the processors of these languages can be kept almost unchanged, and no new algorithms are needed. In addition, XML is used for expressing the user model data like browsing history, browsing environment (such as device, location, time, etc.), and application data (such as user performances on exercises). The goal of the research presented here is not to propose novel forms or applications of adaptation, but instead to extend widespread web standards with adaptation functionalities. Essential features of the proposed approach are its simplicity and both the upwards and downwards compatibility of the extension.

1 Introduction

Recently, much attention has been paid to adaptive computing in business and industry: "contextual computing", i.e. context dependent intranet services is considered a fast growing market with tremendous promises [18]. However, although intranets nowadays mostly rely on web techniques, the discrepancy between current web standards and the state of the art in adaptive hypermedia is noticeable: Basically, adaptation in current HTML and XML is limited to a device dependent choice of style sheets.

In this paper, a rather simple extension is described for enhancing HTML and XML with adaptation. The approach consists in using the path selectors of the HTML and XML style sheet languages CSS [4] and XSLT [11] for expressing content and navigation adaptation [7]. The necessary extension of a selector language like Selectors [14] (that of CSS) or like XPath [12] (that of XSLT) are minimal, a few additional constructs suffice. The processors of these languages can be kept almost unchanged, no new algorithms are needed. Furthermore, XML is used for expressing the user model data like browsing history, browsing

environment (such as device, location, time, etc.), and application data (such as user performances on exercises).

In existing systems, extending HTML and XML with simple adaptive hypermedia functionalities is done using a combination of cookies, that is, client-side user identification, server-side scripting languages like PHP [17], and URIs. This has several drawbacks. With this approach, information about the user has to be stored and processed on the server. Due to the nature of the web's HTTP protocol, this information is limited as compared to the information (possibly) available on the client side. For example, it is not possible to track navigation using the back and forward buttons, navigation in different windows, or navigation on more than one server. This prevents non-trivial adaptive hypermedia systems being implemented using this technique. In contrast, the approach proposed in this paper does not suffer from the above-mentioned drawbacks, as it works on the client side.

The paper is organized as follows. This introduction is followed by section 2, which introduces a data structure called *browsing context*. Browsing contexts can be used for expressing user models. Section 3 describes an extension to style sheet selectors that allows HTML and XML web pages to adapt to the information represented in a browsing context. Next, section 4 briefly addresses a few possible extensions to this approach. Finally, section 5 discusses advantages and limitations of the approach proposed in this paper.

2 Browsing Context: A Data Structure for expressing User Models

HTML and XML have no means to express a user model, that is, information about the user like browsing history, user preferences, performances on excercises, answers to questionnaires, etc. This paper first proposes a data structure called *browsing context*, which allows such information to be stored by the browser, that is, on the client side, to be accessed through style sheets, and to be updated through web applications using scripting languages like Javascript [13]. These features make the data structure "browsing context" convenient for an adaptive presentation of web pages.

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. Note that there are several different notions of location, like geographical location and virtual location, which is discussed in section 4.

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 excercises, like the numbers of correct and wrong answers. In an e-commerce application, 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 HTML and XML. Therefore it has to be provided by the application itself. Section 4 briefly describes how scripting languages like Javascript can be used to update application data.

Using style sheets and scripting languages in conjunction with a browsing context offers the possibility to quite easily implement an adaptive hypermedia system. This is described in detail in section 3. 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 an online 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 CD shop web application 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, nonproprietary way of implementing adaptive hypermedia systems as web applications.

Web browsers store an internal representation of the document currently being displayed, for example as a DOM [2] tree. This document is referred to in the following as *"nude document"* because it does not contain any browsing context information. In a similar way as this "nude 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 "nude document", as it is described in section 3.

Both the "nude 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 "nude 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 "nude 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 "nude document" within the browser, that is, style sheets are applied to the context enriched document instead of the "nude document", scripting languages have access to the DOM tree of the context enriched document instead of the DOM tree of the "nude 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.

3 Implementing Adaptation using Style Sheet Selectors

This section describes a simple extension to style sheet selectors making it simple to implement adaptive hypermedia functionalities with HTML and XML. The path expression of a style sheet selector is not to be matched against the original "nude 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 and XSLT 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 "nude 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 "nude 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 [6], 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 "nude 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 "nude 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 (`bc` in the example above) to distinguish between those components of a selector referring to the "nude 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 "nude 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 "nude 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 XSLT, for example, the selector language is XPath 1.0 [12]. XPath 1.0 or XPath 2.0 [3] 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, that is, those that have not visited the CD shop's homepage 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 "nude 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 Possible Extensions

Updating Application Data using Scripting Languages. Using style sheet selectors to express content and navigation adaption, as it is described in this paper, 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 ("nude") 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 outside the scope of this paper.

Modeling Locations. As stated in section 2, there are several different notions of location. First, location can be information about the country or region where the user is, like Germany or France. This information is typically available in desktop computer systems and does not change during a browsing session. Second, location can be information about the geographical position of the user, expressed for example as longitude and latitude. This information is typically available in mobile devices like cellular phones or PDAs with special positioning equipment, for example a GPS (Global Positioning System) device. Geographical location information can change during a browsing session as the user moves around. Third, 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. Information about virtual locations can change during a browsing session. All of these notions, for example geographical position and virtual location, can be represented simultaneously as browsing environment data in a browsing context.

5 Discussion and Concluding Remarks

The approach proposed in this paper has several advantages and limitations. First, the approach is quite simple. It introduces a wide range of adaptation features into existing HTML and XML standards at the cost of very limited extensions to these standards. The extensions to these standards are as follows:

First, information like browsing history data and browsing environment data, most of which is already stored by conventional browsers, is to be stored as a standardized *browsing context* in an internal XML representation like DOM.

Second, the style sheet processor(s), for example those of CSS or XSLT, match the selector part of a style rule not against the original "nude document", but against the (virtual) context enriched document (consisting of the "nude document" enriched with a browsing context). The style sheet processor must recognize those selector components referring to the "nude document" and those referring to the browsing context. This is conveniently achieved using namespaces, for example, as described above.

Apart from these, no further changes are needed, especially, no new algorithms have to be conceived and implemented. Note that only the processing of style sheet rules is extended, the style sheet languages remain otherwise unchanged (because of the use of namespaces, see section 3). This ensures upward compatibility with already existing style sheets. Also, style sheets that make use of browsing context selectors are downwards compatible with non-browsing context enabled browsers. With such browsers the data can be accessed, only the adaptation features are missing. (Note that these compatibilities are essential when extensions to existing web standards are proposed.) Thus, the approach proposed in this paper is a conservative extension of the already existing and well-established web standards.

The approach described in this paper is not specific to CSS or XSLT. It relies only on path selectors, which play a central role in web standards. The same approach can easily be applied to other or future style sheet languages or to other web 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, which have introduced a considerably more complex type system, a set of relational operators, and certain kinds of variables into the language. This flexibility, however, makes the approach limited to the scope of the underlying standards, in this case CSS or XSLT. Features common to conventional adaptive hypermedia systems, for example the disabling of hyperlinks to prevent novice users from information overloading, cannot be implemented using CSS. Such extensions, however, could easily be introduced into CSS and future style sheet languages may already offer more powerful features that allow to implement complex adaptive hypermedia systems. This, however, remains to be done and is outside the scope of this paper.

Because the style sheet based approach proposed in this paper works on the client side, it does not face security problems as server-side approaches do, for example the combination of cookies, PHP and URIs, that has been described in the introduction. The whole processing is done within the browser and therefore

no possibly private data has to be transferred to and from the server, what would become a security issue. On the other hand, the client-side approach forces all data that possibly might be needed to be transferred to the client before it can be processed. This might cause bandwidth problems as compared to a server-side approach, where only data that is transferred that is really needed on the client.

Finally, this approach has not yet been fully implemented. The implementation of a system similar to traditional adaptive hypermedia systems like [15, 5, 1] is in process. The prototype described in [9] and the implementation of XPath in [16] provide with a framework for this implementation. Field tests with users and various user models remain to be done.

This work has been submitted to the W3C CSS and XSL working groups.

References

1. F. Albrecht, N. Koch, et al. Smexweb: an Adaptive Web-based Hypermedia Teaching System. *International Journal of Continuing Engineering Education and Life-Long Learning*, 2000.
2. V. Apparao, S. Byrne, et al. Document Object Model (DOM) Level 1 Specification Version 1.0. W3C Recommendation, 1998. http://www.w3.org/TR/REC-DOM-Level-1.
3. A. Berglund, S. Boag, et al. XML Path Language (XPath) 2.0. W3C Working Draft, 2001. http://www.w3.org/TR/xpath20.
4. B. Bos, H. W. Lie, et al. Cascading Style Sheets, level 2. W3C Recommendation, 1998. http://www.w3.org/TR/REC-CSS2.
5. P. D. Bra and L. Calvi. AHA: A Generic Adaptive Hypermedia System. In *2nd Workshop on Adaptive Hypertext and Hypermedia, HYPERTEXT'98.*
6. T. Bray, D. Hollander, et al. Namespaces in XML. World Wide Web Consortium, 1999. http://www.w3.org/TR/REC-xml-names.
7. P. Brusilovsky. Methods and Techniques of Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, 6(2-3):87–129, 1996.
8. F. Bry and M. Kraus. Adaptive Hypermedia made simple using HTML/XML Style Sheet Selectors. In *2nd Int. Conf. on Adaptive Hypermedia and Adaptive Web Based Systems (AH2002).*
9. F. Bry and M. Kraus. 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.
10. F. Bry and M. Kraus. Style sheets for context adaptation. W3C Workshop on Delivery Context, 2002.
11. J. Clark. XSL Transformations (XSLT) Version 1.0. W3C Recommendation, 1999. http://www.w3.org/TR/xslt.
12. J. Clark and S. DeRose. XML Path Language (XPath) Version 1.0. W3C Recommendation, 1999. http://www.w3.org/TR/xpath.
13. Standard ECMA-262. ECMAScript Language Specification, 1999. ftp://ftp.ecma.ch/ecma-st/Ecma-262.pdf.
14. D. Glazman, T. elik, et al. Selectors. W3C Candidate Recommendation, 2001. http://www.w3.org/TR/css3-selectors.
15. W. Nejdl and M. Wolpers. KBS Hyperbook - a data-driven information system on the web. Technical report, University of Hannover, KBS Institute, 1998.

16. D. Olteanu, H. Meuss, et al. Symmetry in XPath. http://www.pms.informatik.uni-muenchen.de/publikationen/#PMS-FB-2001-16, 2001.
17. PHP - Hypertext Preprocessor. http://www.php.net/.
18. J. Wrolstad. IDC Identifies New Opportunity in Collaborative Computing. *www.CRMDaily.com*, 2001. http://www.newsfactor.com/perl/story/9789.html.