Towards Generic Layouts for Multi-Channel Publishing: "XF—Extensible Formatting"

Heli Tervo, Anne Honkaranta, Paula Leinonen

Department of Computer Science
University of Kuopio
PO Box 1627
FI-70211 Kuopio, Finland
{heli.tervo, paula.leinonen}@cs.uku.fi

Department of Computer Science
University of Jyväskylä
PO Box 35
FI-40014 Jyväskylä, Finland
anne.honkaranta@it.jyu.fi

Abstract: XML (Extensible Markup Language) has become the *lingua franca* for information integration, e-Business and metadata. It is also invading to office documents and hence becoming the format for organizational and end-user documents. XML documents themselves do not contain information about styling, i.e., how they should be rendered for print, screen, or other channels of publication. One document type may need numerous styling definitions as CSS (Cascading Style Sheets) or XSL (Extensible Stylesheet Language) stylesheets for differing multi-channel publishing purposes. Producing style definitions with differing styling languages may be quite tricky for end-users. The management of a myriad of formatting definitions may pose challenges for organizations as well. Tools for defining CSS style definitions are available but the support for differing versions of CSS on different end-user devices may be varying. Use of XSL requires programming skills or applications providing graphical user interfaces for XSL. Although there are a few commercial products available for producing XSL from easy-to-use graphical interface, there is a need for tools allowing generic styling for XML documents to be defined. The generic styling may support multiple differing formatting outputs. This paper presents a possible solution for XML document formatting. XF (Extensible Formatting language) is a generic high-level formatting language serving as a mediator for contemporary, powerful formatting languages such as CSS and XSL. A prototype implementation of XF with an easy-to-use visual interface is also described. XF is based on XML and bridges the gap between end-users and existing powerful formatting languages.

1 Introduction

XML [Br04] has become widely used for data processing and exchange, especially on the Web. One of the premises of XML technologies is universal interoperability and independency from operating systems and software-producers' proprietary formats such as versions of Microsoft Word. Each XML document may be formatted for the purpose of specific user device such as a PDA, a web browser, or a printer. The styling properties may be attached to the document at the advent of "publication", for example, when a user opens a document on her or his web browser or by formatting groups of documents for static delivery as a part of publication process. Separation of content and styling is a relief for document maintenance; it relieves the publishers from the pain of manipulating the documents with embedded styling features one by one. Yet defining styling instructions for XML documents may become a stumbling block for organizations and end-users. Formatting languages or applications are either difficult to use or do not really support the production of multiple style formats, or both. In the worst case distributing data in multiple formats means defining, or even programming, multiple style sheets separately for each of the organization's document types. As time goes by the management of differing versions of style sheets may become an issue to be dealt with, too.

The most popular contemporary formatting languages are CSS [Cascading Style Sheets; Bo98] for HTML and XML documents and XSL [Extensible Stylesheet Language; Ad01] for XML documents, both by the World Wide Web Consortium (W3C) [Wo06]. In CSS and XSL the formatting properties are based on the form of the result. CSS is designed for Web display while XSL has a strong support for printed pages. XSL is much more powerful and complex language than CSS, demanding programming skills, hence making it difficult to adopt and use without specific software tools. XSL has the potential to become the standard style language for XML documents, yet the support for XSL in browsers is still rare. It is also too difficult for an ordinary user to write XSL scripts manually. CSS has its own syntax consisting of selectors and style properties while XSL style sheet is an XML document itself and adopts a specific vocabulary for defining page-oriented as well as other formatting properties.

Research on style languages and style systems for structured documents such as XML has so far focused on multimedia documents [BR02, Vi01, KST04, QV04, Si04, Bu05]. Also solutions for more flexible Web displays [KL01, AMP04] and multiple layouts for one document have been studied. XML editors [PVQ01, Ya04, Va05, GNB03] or form editing tools [HV04] have not been able to tackle with the problems related to styling in multi-channel publishing. Main stream of XML research has focused on transformations or utilization of XML on a large scale while the research focused solely on the styling of XML documents has been quite scarce. Proposed solutions for multi-channel publication and styling have hence remained modest by their number. Munson [Mu95, MM98] has developed a style language called PSL, and Richy [Ri98] has considered generic style sheets. Yet there is a need for a lightweight front-end language that supports truly multiple outputs and provides an easy-to-use graphical user interface (GUI) for end-users. Some efficient commercial applications realize, at least partly, the problem of multi-channel publishing [Al05, Re06, An06, St06, Ar06] but being complete publishing solutions they are extensive and suffer from quite complicated UIs.

Defining the formatting for a class of XML documents for some specific target formats is possible with several languages and applications. However, the number of style definitions and their versions to be managed may become high by time. There is a need for an easy-to-use *generic formatting language* defining a generic layout for a class of XML documents independently from the resulting style specification syntax and version. Such a language would act as a mediator between the user interface and the specific style definitions created in the formatting task. The generic formatting language and an application for using it may therefore allow a sort of plug-in approach for generating a number of result formats from the easy-to-use interface. The XSL language is a powerful styling language and there is no reason to try to override it or other practical styling languages. Yet an easy-to-use interface for XSL and other style definitions would be convenient for end-users. A generic styling along with user-friendly interface may also save time spent on the style definition tasks.

We can imagine this kind of need with an organization that wants to have a generic layout in all documents that are produced; documents on intranet and Internet, paper printout documents, letters, et cetera. In this situation, the XML data has to be formatted with several style sheets, depending on the form of result. It would be advantageous if the generic layout could be defined with a single style sheet.

This paper contributes by describing the features and the syntax of an XML-based generic formatting language called XF—Extensible Formatting. A prototype application developed for utilizing XF via graphical user interface for generic formatting is also described. XF utilizes XML as its syntax hence making use of XML technology and contemporary formatting solutions rather than ignoring them.

The rest of the paper is organized as follows. Section 2 outlines the requirements and properties for a generic formatting language and introduces the Extensible Formatting language XF. Section 3 describes the phases of the styling process and the XF prototype architecture. Section 4 illustrates the prototype interface for XF and provides an example of XF use-case. Section 5 concludes the paper.

2 XF—Extensible Formatting Language

The XF language was developed by using an explorative prototyping approach. After the requirements for a generic formatting language were defined, the schema for XF styling was developed. The XF prototype was developed side by side with gradual experiments on the XF language. This section outlines the requirements for generic styling language and describes the properties of XF language.

2.1 Requirements for Generic Formatting Properties

A generic description language for formatting data to multiple result formats has to have formatting objects suitable for every result form, independently of the nature of the resulting output style definition format. The requirements for multi-channel publication are varying: one should take page properties into account for paper print, it is commonly known that font face for Web text needs to be different from that for print, and that differing output devices such as PDAs and computer screens vary a lot on their capabilities on rendering and positioning textual and multimedia objects. Regardless of diverse requirements a number of generic styling features that are common to most forms of styling output results may be identified.

The common formatting features for XF were extracted from the most commonly used style definition formats CSS and XSL. The layout features and properties were also studied by inspecting the layout of paper prints and Web screens. The common formatting features identified as required formatting properties were: classification, position, alignment, table, background, border, font, and page.

Classification property defines how each element should be displayed; as a block, inline, list or table element. With position properties block-level elements may be positioned with absolute coordinate values on the result page. Alignment includes vertical and horizontal alignments of elements. As many users involved with multi-channel publishing already know, the tables may require complex formatting properties to be used for defining the positioning of columns and rows and their numbers. Background property allows defining a background colour or image to an element, and with border properties simple border styles can be defined. Font properties include style, weight, size and colour properties. Page properties define dimensions and margins of a printable page.

It may be noticed that these features are not purely common for every unknown result format. The page category is not common in a sense we discussed earlier, but we came to a conclusion that this feature is fundamental for paper prints, because formatting the page is impossible without values of page dimensions. On the other hand defining fixed default values will restrict the idea of a generic formatting language inconveniently.

2.2 Syntax of XF Language

An XF style definition document is a well-formed XML document composed of rules for selecting objects to be formatted, formatting properties assigned to objects, and global collected properties called property-sets. An example of a part of an XF script is shown in Figure 1.

Element properties defines the formatting properties for certain data elements. Obligatory attribute for (line 5) gives an XPath [Cl99a] expression for selecting data elements to be formatted. The value can be, for instance, a single node or a set of nodes. Element properties includes formatting properties as its child elements and may also include references to property-sets.

```
1 <xf:property-set name="text-font">
2         <xf:property name="font-family" value="helvetica"/>
3         <xf:property name="font-size" value="10"/>
4 </xf:property-set>
5 <xf:properties for="head">
6         <xf:use-property-sets names="text-font"/>
7         <xf:property name="font-size" value="16"/>
8         <xf:property name="font-weight" value="bold"/>
9 </xf:properties>
```

Figure 1: An example of XF script

Element property defines one formatting property (lines 7 and 8). It has an obligatory attribute name that defines the property name and value that defines the property value. The generic form of the property elements is

<xf:property name="*property-name*" value="*property-value*"/>

The generic form supports the extensibility of XF with new properties; the user may define a whole new property with a name and a value, in an XF script. It, though, means that the properties must be implemented in the architecture.

Element property-set defines a named set of properties, see lines 1-4. Element use-property-sets is for including the properties defined in a property-set element (line 6). The property-set can be included as whole in any properties element (or property-set element), providing a way of defining commonly used sets of properties in a single place. Multiple definitions are treated like in CSS: there are now two values for font-size specified for each element head in the example above, but the last value will be chosen. This feature allows us to use global, collected definitions flexibly and to override some properties later if needed.

It is possible to use *short-hand notations* in place of a property. Short-hand notations are defined for *built-in properties*. These are properties that are defined in XF, thus they are not defined for the user's own properties. A *built-in property group* is a set of properties that belong together somehow, for example, all properties concerning font, like font-size and font-family. In short-hand notations all properties belonging to a built-in property group can be specified in the same element. This idea is borrowed from CSS, where properties can be grouped together into one rule. For example, lines 7 to 8 could be written as

<xf:font font-size="16" font-weight="bold"/>

XF has eight built-in property groups that were derived from the common formatting features discussed above. The built-in properties with built-in property groups are defined in Table 1. The requirements for formatting tables were solved with a rough categorization of table properties: rows and cells are actual data and columns are metadata. The metadata for table has a specific property group element table containing also an attribute column-widths.

Table 1: Built-in properties and property groups in XF

BUILT-IN PROPERTY GROUPS	ATTRIBUTES	ATTRIBUTE VALUES
For 'property' elements		
classification	display	block,inline,list-item,table,table-row,table-cell,none
table	columns column-widths	<number> <length>+
position	Top, left, height, width	<length>
alignment	text-align vertical-align	left, center, right, justify top, middle, bottom
background	background-color background-image	<color>, transparent <uri>, none
border	border-style border-color	none, dotted, dashed, solid, double <color>
font	font-family font-style font-weight font-size color	serif, sans-serif, cursive, fantasy, monospace normal, italic normal, bold <length> <color>
For 'page-property' elements		
page	page-width page-height margin	<length> <length> <width>

3 Styling Process and Architecture of XF

Design of the XF architecture makes formatting of XML documents semi-automatic; end-users specify only the XF definition with a graphical UI and choose the preferred output formats. Data processing from XF definition and XML source document to resulting formats is carried out automatically.

XML syntax makes it possible to process both the source document and the formatting specification using common XML tools. This brings compatibility and easier usability with XML technology and makes the language more generic. XF provides true compatibility with contemporary multi-channel distribution and allows new style formats to be generated when needed; more sophisticated users or experts may build plug-ins for result document formats not implemented in the current XF application.

Figure 2 illustrates the styling process in XF, which is managed in two phases. In the first phase, the user defines formatting for the source document in a graphical UI and the system produces the formatting document XF. The source XML document and the formatting document XF are processed and combined together to a "presentation document". This resulting document has all the content of the source document, but also the formatting instructions from XF alongside with calculated values which are needed, for example, for tables. In the second phase, the resulting style formats are generated from the presentation document with stand-alone applications. The system produces these output formats automatically. The front-end (Figure 2: I, building the presentation document) and the back-end (II, generating the result format) are distinct parts of the XF architecture.

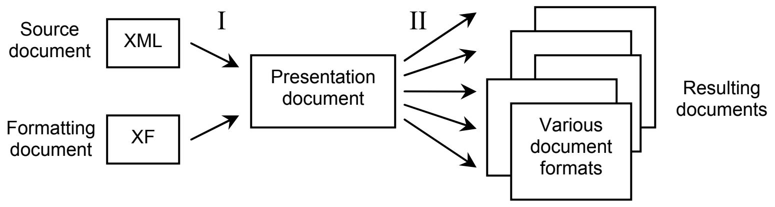


Figure 2: The styling process for XML documents in XF: building the presentation document (I) and generating the result formats (II)

The first prototype of the XF front-end (Figure 3) is implemented with Java and DOM4J [DO06] which is an interface to Document Object Model [DO00] for Java application developers. DOM4J contains support for XPath [Cl99a]—a powerful language for addressing and filtering nodes on XML documents. The use of XPath language provides an easy way for generating the presentation document.

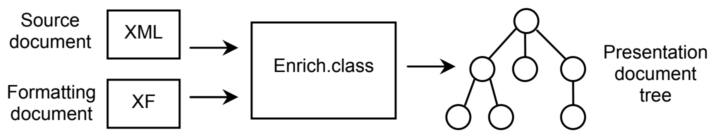


Figure 3: The front-end in the styling process of XF: Building the presentation document tree with Java and DOM4J

In the first part of the XF styling process the XML source document and the XF formatting document defined by a user are both read to DOM trees. The "Enrich" application reads the XF DOM tree node by node, and appends formatting properties for the source document tree. In case a user does not provide formatting instructions for a source document node there are built-in default formatting properties in XF. The enriched tree is used as a source for producing layout for chosen output format automatically. The presentation document tree includes all information that the back-end applications need to build output formats.

XF prototype contains several back-end applications; one for each different output style definition as illustrated in Figure 4. A back-end application maps the properties of the presentation document tree to features of the selected output format such as to XSL formatting objects (XSL-FO). Back-end applications are capable of producing the output formatting syntax directly from the presentation document tree. Two back-end applications are currently implemented on the XF prototype with XSLT [Cl99b]: one for producing HTML and the second one for XSL-FO from which paper prints can be produced as PDF. Back-end applications are independent from the rest of the architecture, so sophisticated users can easily build new back-end applications with XSLT, or other suitable languages such as Java.

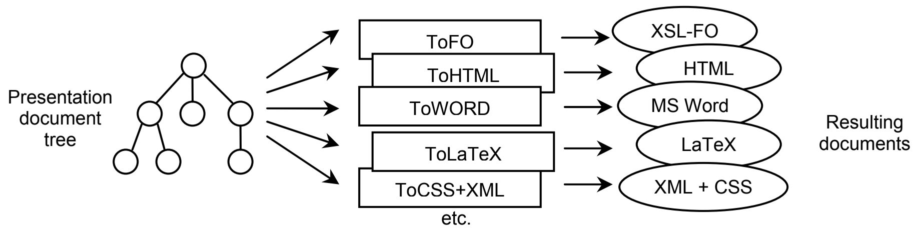


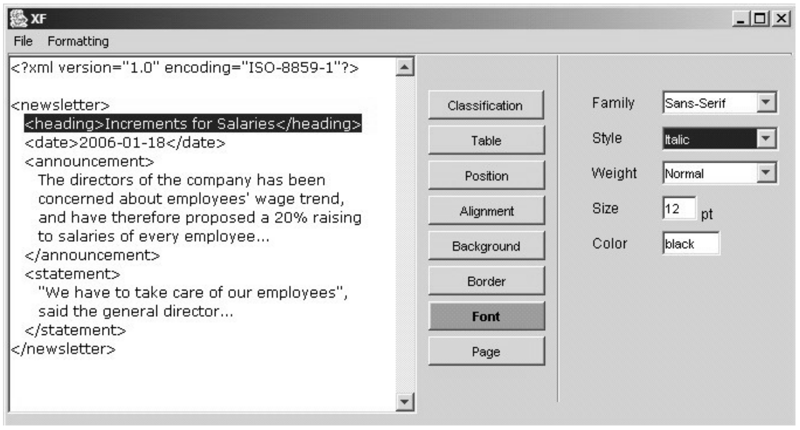
Figure 4: The back-ends in the styling process of XF: Generating result formats

4 XF Interface and XF Use-Case

From the end-users' aspect a graphical UI is a mandatory requirement. End-users on the Web may not have enough skills to handle raw XML documents; a GUI makes the formatting language easily available for such users. Among users on the Web, for example, the formatting task should be as easy as creating a short web page. If the GUI of a styling language can provide functions that are easily adopted with earlier knowledge of using computer (i.e., functions similar to Word application or in InternetExplorer) then the language has a potential to become commonly used on the Web and adopted as a part of handling documents on the fly.

4.1 Graphical User Interface for XF

The first prototype of the GUI consists of two parts: the XML document to be formatted is downloaded to the left pane (Figure 5) and the formatting properties are shown on the right side. A user selects an element by clicking it. Then she or he may define the desired formatting properties for it by using the buttons and drop-down menus on the right.



```
<?xml version="1.0" encoding="ISO-8859-1"?>

<newsletter>
  <heading>Increments for Salaries</heading>
  <date>2006-01-18</date>
  <announcement>
    The directors of the company has been
    concerned about employees' wage trend,
    and have therefore proposed a 20% raising
    to salaries of every employee...
  </announcement>
  <statement>
    "We have to take care of our employees",
    said the general director...
  </statement>
</newsletter>
```

Family: Sans-Serif
Style: Italic
Weight: Normal
Size: 12 pt
Color: black

Figure 5: The graphical user interface of XF

4.2 Case: Newsletter on the Fly

As an example we consider a fictional company that delivers a newsletter as a web page and a paper print. All the documentation on the company is handled in XML form, so the newsletter is also an XML document. The newsletters have to be published promptly, but publishing the short daily messages is not the responsibility of any specific employee or unit. It means that the author should take care of the publication and delivery as well. The authors could produce their daily news to preferred output formats with XF in a following manner.

The newsletter in XML form is loaded to the XF GUI as shown in Figure 4. The heading will use a red italic font of size 16. The element of the heading is selected by clicking on the top of it. Then the properties for the font are selected by clicking the "font" button, and after that on the right side of the window we can see the possible properties (attributes) for the font. From the drop-down menu the user chooses the style 'italic' for the font. After all the preferred formatting properties have been specified to all elements, the document can be produced to chosen output format.

In this example the outputs are HTML for Web display and PDF for paper print. As soon as an author chooses the preferred output the XF styling process automatically generates formatting and transformation scripts hidden from the user. The first generated script is an XF script (Figure 6). It acts as a mediator for general formatting. The source XML document and the automatically generated XF script are combined together. The resulting "presentation document" is shown in Figure 7 (see Figure 3, also).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xf:formatting xmlns:xf="http://www.cs.uku.fi/XF/2003">
   <xf:properties for="heading">
            <xf:property name="font-style" value="italic"/>
            <xf:property name="font-color" value="red"/>
   </xf:properties>
</xf:formatting>
```

Figure 6: Automatically generated XF script

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<newsletter xf:page-width="21cm" xf:page-height="29.7cm"      xf:margin="2cm" xf:text-
align="left" xf:font-size="10pt"...>
   <heading xf:font-style="italic" xf:font-color="red" xf:font-
   size="16pt">Increments for Salaries</heading>
   <date>2006-01-18</date>...
</newsletter>
```

Figure 7: A part of the presentation document

The outputs are generated with XSLT from the presentation document tree. Figures 8 and 9 represent portions of the outputs of HTML and XSL-FO scripts, respectively (see Figure 4, also).

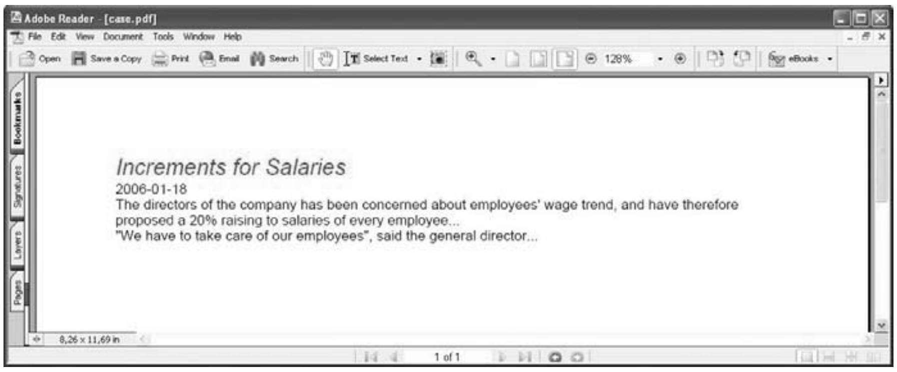
```
<html xmlns:xf="http://www.cs.uku.fi/XF/2003">
   <body>
            <div style="align: left; family: 'Helvetica';
            font-style: italic; font-weight: normal; size: 16pt;
            color: red; ">Increments for Salaries</div>...
   </body>
</html>
```

Figure 8: A portion of the resulting HTML document

```
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"
xmlns:xf="http://www.cs.uku.fi/XF/2003">
   <fo:layout-master-set>
            <fo:simple-page-master master-name="page"
                     page-width="21cm" page-height="29.7cm">
                     <fo:region-body margin="2cm" />
            </fo:simple-page-master>
   </fo:layout-master-set>
   <fo:page-sequence master-reference="page">
            <fo:flow flow-name="xsl-region-body">
                     <fo:block text-align="left" font-
                              family="'Helvetica'" font-style="italic"
                              font-weight="normal" font-size="16pt"
                              color="red">Increments for Salaries
                     </fo:block>...
            </fo:flow>
   </fo:page-sequence>
</fo:root>
```

Figure 9: A part of the generated XSL-FO script

Formatting instructions defined by an XSL-FO script serve as an input to XSL Formatting Object Processor, such as Apache FOP [Ap06]. The FO processor produces a PDF output document such as the one illustrated in Figure 10.



Adobe Reader - [case.pdf]
File Edit View Document Tools Window Help
Open Save a Copy Print Email Search Select Text 128% eBooks
Bookmarks Signatures Layers Pages
8,26 x 11,69 in
1 of 1

Increments for Salaries
2006-01-18
The directors of the company has been concerned about employees' wage trend, and have therefore proposed a 20% raising to salaries of every employee...
"We have to take care of our employees", said the general director...

Figure 10: The resulting output document in PDF format

5 Conclusion

While existing research on XML has focused on information and application integration and transformations, the attention on XML document styling mechanisms and their usability has been scarce. This may be due to the fact that XML has not reached the offices and main stream document production yet. Along with new office products, XML is going to invade to offices and become the format for common documents. It becomes available for everyone using novel office applications.

There are both powerful styling languages such as XSL and more simple and flexible ones like CSS available for styling. There is no need to replace them. Instead, new solutions are needed for automating creation of style definitions. Multi-target formatting is a difficult task not only for end-users but in a general level, also; there is no widely accepted standard way to define styling that would directly support the generation of multiple formats. There is a need to distribute data to several formats, like disparate Web displays, hand-held devices or printable pages. With existing technology this usually means describing multiple formatting scripts with multiple style sheet languages or applications. There is a need for easy-to-use, generic formatting tools allowing easy creation of numerous style formats such as HTML with CSS as well as XSL-FO.

This paper presented XF—a high-level description language for formatting XML documents by their generic layout features. The prototype XF styling application allows definition of styles for multiple different result formats from one generic XF style sheet. At the moment, XF produces HTML and XSL-FO for PDF but new style formats may be easily plugged in due to the separation of the front- and back-end sides of the XF architecture. XF provides a prototype example of an easy to use interface to existing styling technologies. The graphical UI of XF thus bridges the gap between end-users and powerful styling technologies. XF or a similar application may also resolve the tedious task of defining multiple style sheets for different distribution channels, as well as save time and effort needed on maintaining the style sheets.

Acknowledgements

Professor Pekka Kilpeläinen has inspired this work and provided support for the development of XF and the XF application.

References

[Ad01] S. Adler et al, Extensible Stylesheet Language (XSL) Version 1.0, W3C Recommendation, October 2001, http://www.w3.org/TR/xsl.

[Al05] Altova StyleVision 2005, http://www.altova.com/ (visited 8.2.2006).

[AMP04] A.R. de Andrade, E.V. Munson, M. da G. Pimentel, A Document-based Approach to the Generation of Web Applications, DocEng'04, Wisconsin, USA, October 2004, pp. 45-47.

[An06] Antenna House XSL Formatter, http://www.antennahouse.com/ (visited 8.2.2006).

[Ap06] Apache FOP, http://xmlgraphics.apache.org/fop/ (visited 8.2.2006).

[Ar06] Arbortext Styler, http://www.arbortext.com/html/styler.html (visited 8.2.2006).

[Bo98] B. Bos et al, Cascading Style Sheets, level 2, W3C Recommendation, May 1998, http://www.w3.org/TR/CSS2.

[BR02] F. Bes, C. Roisin, A Presentation Language for Controlling the Formatting Process in Multimedia Presentations, DocEng'02, Virginia, USA, November 2002, pp. 2-9.

[Br04] T. Bray et al, Extensible Markup Language (XML) 1.0 (Third Edition), W3C Recommendation, February 2004, http://www.w3.org/TR/REC-xml.

[Bu05] D.Bulterman et al, Synchronized Multimedia Integration Language (SMIL 2.1), W3C Recommendation, December 2005, http://www.w3.org/TR/SMIL/.

[Cl99a] J. Clark et al, XML Path Language (XPath) Version 1.0, W3C Recommendation, November 1999, http://www.w3.org/TR/xpath.

[Cl99b] J. Clark, XSL Transformations (XSLT) version 1.0. W3C Recommendation, November 1999, http://www.w3.org/TR/xslt.

[DO00] W3C DOM Working Group, Document Object Model (DOM) Level 2, W3C Recommendation, November 2000, http://www.w3.org/ DOM/.

[DO06] DOM4J Homepage, http://www.dom4j.org (visited 8.2.2006).

[GNB03] M. Grossniklaus, M.C. Norrie, P. Buchler, Metatemplate Driven Multi-Channel Presentation, WISEW'03, IEEE 2003.

[HV04] M. Honkala, P. Vuorimaa, A Configurable XForms Implementation, ISMSE'04, IEEE 2004.

[KL01] A. Kaplan, J. Lunn, FlexXML: Engineering a More Flexible and Adaptable Web, ITCC'01, April 2001, p. 405.

[KST04] P. King, P. Schmitz, S. Thompson, Behavioral Reactivity and Real Time Programming in XML, DocEng'04, Wisconsin, USA, October 2004, pp. 57-66.

[MM98] P.M. Jr. Marden, E.V. Munson, PSL: An Alternate Approach to Style Sheet Languages for the World Wide Web, J. of Universal Comp. Sci., 4(10), 1998, 792-806.

[Mu95] E.V. Munson, A New Presentation Language for Structured Documents, Electronic Publishing, 8(2&3), 1995, 125-138.

[PVQ01] E. Pietriga, J.-Y. Vion-Dury, V. Quint, VXT: A Visual Approach to XML Transformations, DocEng'01, Georgia, USA, November 2001, pp. 1-10.

[QV04] V. Quint, I. Vatton, Techniques for Authoring Complex XML Documents, DocEng'04, Wisconsin, USA, October 2004, pp. 115-123.

[Re06] RenderX XEP, http://www.renderx.com/ (visited 8.2.2006).

[Ri98] H. Richy, Document Style Design by Direct Manipulation, Electronic Publishing, Artistic Imaging, and Digital Typography, Saint-Malo, France, 1998, 331-342.

[Si04] H.V.O. Silva et al, NCL 2.0: Intergrating New Components to XML Modular Languages, DocEng'04, Wisconsin, USA, October 2004, pp. 188-197.

[St06] Stylus Studio 2006, http://www.stylusstudio.com/ (visited 8.2.2006).

[Va05] I. Vatton et al, Amaya W3C's Editor/Browser, December 2005 http://www.w3.org/Amaya/ (visited 8.2.2006).

[Vi01] L. Villard, Authoring Transformations by Direct Manipulation for Adaptable Multimedia Presentations, DocEng'01, Georgia, USA, November 2001, pp. 125-134.

[Wo06] World Wide Web Consortium (W3C), http://www.w3.org (visited 8.2.2006).

[Ya04] H. Yao et al, WebTop XML Edito Supporting Operations on Views Generated by User-Defined Styles, AINA'04, IEEE 2004.