Style Sheet Languages for Hypertext

Jacco van Ossenbruggen (*), Lynda Hardman (**),
Lloyd Rutledge (**), Anton Eliëns (*)
(*) Vrije Universiteit, Fac. of Mathematics and Computer Sciences
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
(**) CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
email: jrvosse@cs.vu.nl

1 INTRODUCTION

There are several advantages in being able to separate the structural information of a hypermedia document from the specification of the actual presentation of the document [17]. For example, one style can be applied to multiple documents, so that changing the style for all documents requires only a single change to the style information. Alternatively, a single document may have multiple styles tailored to different presentation platforms such as large screens, small screens or paper.

Separation of structural and style information has long been commonplace for text, and can also be found in many hypertext models. For example, the Dexter hypertext reference model [7] separates the structural information of the storage layer from the style and layout information encoded in the presentation specifications. In most hypermedia design models, including RMM [12] and HDM [5], the two types of information are designed during different phases of the design process.

Style sheets are frequently used to specify style information separate from the structural information. For example, on the Web Cascading Style Sheets (CSS [13, 2]) are used to describe the appearance of HTML documents. While CSS is designed primarily for describing how HTML documents should be formatted by Web browsers, other applications require style sheets to be applicable to more than one class of documents. The electronic publishing community, for example, uses many SGML document types other than HTML. Instead of developing a style language for every single document type, DSSSL defines a language which can be applied to any SGML-defined document type, including HTML.

Given the number of different hypermedia models and applications that exist today, being able to define style sheets for different document types is an important feature of a hypermedia style sheet language. However, it is not sufficient to fully qualify as a hypermedia style sheet language. For example, a DSSSL style sheet defines a mapping from the source document to a common abstract output model. This output model is page-based, that is, it describes the output of the formatting process as a sequence

of pages, columns, footers, etc. Thus it is not suited as a style sheet language for documents that require the synchronized presentation of hyperlinked media items — for which we use the term hypermedia.

In this article we describe current style sheet languages, identify the areas in which these languages are too limited to be applied to hypermedia, and introduce a solution to overcome these limitations. To illustrate the discussion and our proposed solution, we use a running example of a hyperlinked multimedia presentation on the World Wide Web.

2 CURRENT STYLE SHEET LANGUAGES

Currently, three style sheet languages are important in respect to their use in hypermedia. These languages are CSS, DSSSL and XSL.

2.1 CSS

The World Wide Web Consortium defined a number of formats based on the idea of Cascading Style Sheets (CSS). CSS level 1 (CSS1 [13]) is an official W3C Recommendation and defines how multiple style sheets can be applied to determine how an HTML document is to be displayed on the screen. A number of extensions to CSS1 are currently being developed. These include constructs which allow high-quality printing of HTML documents, positioning of HTML elements on the screen and speech-synthesis of web documents (e.g. for visually impaired users). All of these extensions are part of CSS level 2 (CSS2 [2]).

As its name suggests, one of the advantages of CSS is the concept of cascading, which allows multiple style sheets to be applied to the same document. Other advantages of CSS are:

- CSS uses a straightforward syntax which can be used by most HTML authors.
- A large part of CSS has already been implemented by the major browser vendors. While some compatibility problems still exist, CSS is used by a large number of Web-sites. Literature, documentation and example material is readily available on the World Wide Web.

The main limitation of CSS, that it handles only HTML documents, is due to its simplicity. While CSS is in principle applicable to document formats other than HTML, in practice it is only used in combination with HTML documents. A CSS style sheet is unable to change the structure of, or the elements within, the document tree — it only modifies their associated style properties. For example, a horizontal rule in the input document remains a horizontal rule in the output document. A CSS style sheet cannot, for instance, map it to a page break in a printed version of the document. To be applicable to a wide variety of document formats, a style sheet needs to be able to transform the elements of a foreign document type, i.e. those not recognized by the browser, to those elements which are supported by the browser. Given the wide variety of hypermedia document models, a generally applicable hypermedia style sheet language will have to support this type of transformations. An example of such a language is DSSSL.

2.2 DSSSL

The Document Style Semantics and Specification Language (DSSSL [11]), is an international standard developed to provide a style sheet language for SGML documents. DSSSL consists of two parts, a tree transformation process that can be used to manipulate the tree structure of documents prior to presentation, and a formatting process that associates the elements in the source document with specific nodes in the target representation — the *flow object tree*. DSSSL specifications are device-independent pieces of information that can be interchanged between different platforms. The back-end formatters needed to generate the final form of the document (e.g. PostScript or Rich Text Format, or a presentation on a computer display) are not standardized by DSSSL. Advantages of DSSSL include:

- A DSSSL style sheet may map elements from its input document to arbitrary elements in the output model. In this way, DSSSL makes no assumptions about the structure of the input document and, in contrast to CSS, is applicable to (SGML) document types other than HTML.
- The DSSSL query language gives the style sheet full access to both the content and the structure of the document my means of a powerful query language.
- DSSSL style sheets are typically device and platform independent. For example, a PDF version on a Unix back-end and an RTF version on a Windows back-end can be generated from the same SGML document and the same DSSSL style sheet.

Important disadvantages of DSSSL are:

- DSSSL uses a side-effect free Scheme dialect to specify the layout of a document, which is very different from the point-and-click approach most authors are familiar with. Furthermore, a thorough understanding of DSSSL's interface to the SGML document structure requires expert knowledge of SGML. Because DSSSL is a relatively young standard, there is still very little literature, documentation or examples about the practical application of DSSSL. The transformation process is not implemented by publicly available tools, and applications of the formatting process on an industrial scale are scarce.
- DSSSL formatting can be device independent because style sheets convert to the same output mode — DSSSL's *flow object tree* (FOT). The FOT defines an abstract, page-based output model which needs to be converted to a concrete output format by the back-end of the DSSSL engine. As a consequence, the formats which can be generated on a particular platform depend on the back-ends available for that specific platform. Documents formats which cannot be modeled by the flow object tree need to define (and implement) application specific extensions to the flow object tree.
- The use of DSSSL to generate HTML documents is rather cumbersome. Since DSSSL's transformation language is not widely implemented, people use the DSSSL formatting language to generate HTML from other SGML documents. Because HTML's structural elements are not included in the standard DSSSL

flow object tree (which contains only layout objects and no structural elements as titles and ordered lists), this process requires implementation-specific extensions to the standard set of flow objects. As a result, it is unlikely that these style sheets can be processed by other DSSSL engines.

2.3 XSL

The eXtensible Style sheet Language (XSL, [1]) is a proposed style sheet language for XML documents. Because the tags used in an XML document may vary from application to application, a style sheet should provide a mapping from the set of elements used in the XML document to the set of elements known to the browser. CSS is not sufficiently powerful to provide such a mapping. While DSSSL can provide these mappings, it is considered to be too complex, especially in environment where most XML applications will originate from an HTML background.

XSL will therefore be based on DSSSL, but with some important differences:

- XSL will use only that part of DSSSL that is needed to specify the online display of and a minimum set of printing utilities for XML documents on the World Wide Web.
- XSL will provide an XML-based syntax as an alternative to the Scheme-based syntax of DSSSL.
- To simplify the targeting of HTML as the output format, XSL will extend the set of flow objects offered by DSSSL with a set of HTML flow objects. These objects will have characteristics based on the attributes of the HTML objects, extended with their properties as defined by CSS.

With these extensions, XSL will have several advantages over the current DSSSL standard:

- The translation of existing CSS style sheets to XSL can be done automatically. This will reduce initial barriers to adoption, especially for Web sites with a large installed base of CSS style sheets. Since the translation from XSL's tagged syntax to DSSSL's Scheme syntax (and vice versa) can also be automated, XSL style sheets can easily be reused by current DSSSL implementations.
- XSL's tagged syntax looks more familiar to HTML users than DSSSL's Scheme syntax. Additionally, since XSL style sheets will be proper XML documents, they can be readily parsed by current XML parsers.

The extensions XSL makes to DSSSL are expected to be incorporated into DSSSL during the next corrigendum, which would make XSL a proper subset of DSSSL.

2.4 Evaluation

When we compare the three style sheet languages above, we see important differences in their processing models. First of all, the CSS output model implicitly contains

HTML elements, extended with the layout properties defined by CSS. Conceptually, CSS has a single processing model, where both the input and the output consists of (extended) HTML documents. When applying this processing model to the example SMIL document, a CSS-like style sheet would add style information to an already existing SMIL document.

In contrast, in the processing model of XSL, the input can be any (XML) document, which is mapped to a single abstract output model, called the flow object tree. The XSL processing model can be applied to our SMIL example if the output model is capable of expressing the desired hypermedia presentation. In terms of our example, the style of a SMIL document could be specified by a style sheet which maps the SMIL document to a presentation which is defined in terms of the given output model. However, the use of such an output model would not be restricted to SMIL, and could be used to define the style of hypermedia document formats other than SMIL.

Finally, DSSSL has two different processing models. The first, called the formatting process, is comparable to the processing model of XSL. The second, called the transformation process, does not produce a final-form document which can be readily displayed. Instead, it transforms one (SGML) document type into another. We can apply this transformation model to our SMIL example in two ways. First, we can use the transformation to convert a SMIL document to another document format (e.g. a format in which the higher level concepts of SMIL are replaced by lower level concepts that are easier to process by the play-out environment). Secondly, we can use the transformation process to generate SMIL presentations from other, more abstract hypermedia document formats such as richly structured HyTime documents for which no general play-out environment is available [15].

3 CONCLUSIONS

We discussed the aspects of developing a style sheet language for hypermedia in general, and used the newly introduced SMIL hypermedia description language for the Web as an example. A future hypermedia style sheet language would require extensions to the current style sheet languages to include temporal aspects — in particular parallel, sequential and relative synchronization constraints. While CSS2 already contains a sufficiently rich spatial layout model for multimedia layout, the inheritance mechanism of CSS properties is not suited for hypermedia documents using temporal composition. Linking support in SMIL is currently limited, so that the hyperlinking aspects in CSS2 are sufficient, but would have to be extended for more complex hyperlinking in future versions of SMIL.

More advanced style sheet languages such as DSSSL allow the specification of structural transformations and style specifications for text-based and HTML documents. While DSSSL can be used for the structural transformation of hypermedia documents, the output model of DSSSL's formatting process supports only page-based output. For presentations that include synchronized hypermedia elements, a style sheet language is required that also accounts for the following aspects: temporal layout, spatial layout that is not text-flow based, hyperlinking to and from continuous media objects, hyperlinking to and within a temporal presentation, hyperlinking to a subset

of a presentation, and adaptive environments. We discussed two alternatives of using DSSSL for hypermedia, since the language features both a powerful document query language and a language for expressing structural transformations. Additionally, DSSSL makes fewer assumptions about the structure of the source documents than CSS, which is an important characteristic given the wide variety of existing hypermedia applications and document models.

We showed that while using the formatting process with a standardized hypermedia extension of the DSSSL output model allows for platform independent style sheets, this approach may be both too ambitious and too rigid. DSSSL's structural transformation process avoids the use of a fixed intermediate model, where in addition the lower level details of a specific output format can be hidden from the style sheet by employing dedicated DSSSL libraries.

References

[1] Sharon Adler, Anders Berglund, James Clark, Istvan Cseri, Paul Grosso, Jonathan Marsh, Gavin Nicol, Jean Paoli, David Schach, Henry S. Thompson, and Chris Wilson. A Proposal for XSL, August 1997. W3C Note for discussion only. Available at http://www.w3.org/Submission/1997/13/Overview.html.

[2] Bert Bos, Håkon Wium Lie, Chris Lilley, and Ian Jacobs. Cascading Style Sheets, level 2, November 1997. Work in progress. W3C Working Drafts are available at http://www.w3.org/TR.

[3] D.C.A. Bulterman, L. Hardman, J. Jansen, K.S. Mullender, and L. Rutledge. GRiNS: A GRaphical INterface for Creating and Playing SMIL Documents. In *Seventh International World Wide Web Conference*, April 1998. Submitted.

[4] James Clark. Jade — James' DSSSL Engine. Available at http://www.jclark.com/jade/, 1997.

[5] Franca Garzotto, Luca Mainetti, and Paolo Paolini. Hypermedia Design, Analysis, and Evaluation Issues. *Communications of the ACM*, 38(8):74–86, August 1995.

[6] W3C SYMM Working Group. W3C Activity on Synchronized Multimedia. More information at http://www.w3.org/AudioVideo/.

[7] F. Halasz and M. Schwarz. The Dexter Hypertext Reference Model. *Communications of the ACM*, 37(2):30–39, February 1994. Edited by K. Grønbæck and R. Trigg.

[8] L. Hardman, D. C. A. Bulterman, and G.van Rossum. The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model. *Communications of the ACM*, 37(2):50–62, February 1994.

[9] L. Hardman, D.C.A. Bulterman, and G. van Rossum. Links in hypermedia: the requirement for context. In *Proceedings of ACM Hypertext '93 (Seattle)*, pages 183–191. ACM, November 1993.

[10] Lynda Hardman. *Modelling and Authoring Hypermedia Documents*. PhD thesis, Universty of Amsterdam, 1998. ISBN: 90-74795-93-5.

[11] International Organization for Standardization/International Electrotechnical Commission. Information technology — Processing languages — Document Style Semantics and Specification Language (DSSSL), 1996. International Standard ISO/IEC 10179:1996.

[12] Tomás Isakowitz, Edward A. Stohr, and P. Balasubramanian. RMM: A Methodology for Structured Hypermedia Design. *Communications of the ACM*, 38(8):34–44, August 1995.

[13] Håkon W. Lie and Bert Bos. Cascading Style Sheets, level 1, December 1996. W3C Recommendation; Available at http://www.w3.org/TR/REC-CSS1-961217.

[14] Lloyd Rutledge, Jacco van Ossenbruggen, Lynda Hardman, and Dick C.A. Bulterman. A Framework for Generating Adaptable Hypermedia Documents. In *Proceedings of ACM Multimedia*, pages 121–130. ACM Press, November 1997.

[15] Lloyd Rutledge, Jacco van Ossenbruggen, Lynda Hardman, and Dick C.A. Bulterman. Getting Around Amsterdam with Standards, Tools and Libraries. In *ACM Digital Libraries*, April 1998. To be published.

[16] W.R.T ten Kate, P.J. Deunhouwer, D.C.A. Bulterman, L. Hardman, and L. Rutledge. Presenting Multimedia on the Web and in TV broadcast. In *3rd European Conference on Multimedia Applications, Services and Techniques*, May 1998. Submitted.

[17] Jacco van Ossenbruggen, Lynda Hardman, Lloyd Rutledge, and Anton Eliëns. Style Sheet Support for Hypermedia Documents. In *Hypertext'97 — The Eighth ACM Conference on Hypertext*, pages 216–217, April 1997.

[18] W3C. Synchronized Multimedia Integration Language (SMIL), April 1998. Edited by Philipp Hoschka. Work in progress. W3C Proposed Recommendations are available at http://www.w3.org/TR.