Towards a Multimedia Formatting Vocabulary

Jacco van Ossenbruggen

Joost Geurts

Lloyd Rutledge

CWI, P.O. Box 94079
1090 GB Amsterdam, The Netherlands

Lynda Hardman
CWI and Technische Universiteit Eindhoven
P.O. Box 513
5600 MB Eindhoven, The Netherlands

Firstname.Lastname@cwi.nl

WWW2003, May 20-24, 2003, Budapest, Hungary.
ACM 1-58113-680-3/03/0005.

Abstract

Time-based, media-centric Web presentations can be described declaratively in the XML world through the development of languages such as SMIL. It is difficult, however, to fully integrate them in a complete document transformation processing chain. In order to achieve the desired processing of data-driven, time-based, media-centric presentations, the text-flow based formatting vocabularies used by style languages such as XSL, CSS and DSSSL need to be extended. The paper presents a selection of use cases which are used to derive a list of requirements for a multimedia style and transformation formatting vocabulary. The boundaries of applicability of existing text-based formatting models for media-centric transformations are analyzed. The paper then discusses the advantages and disadvantages of a fully-fledged time-based multimedia formatting model. Finally, the discussion is illustrated by describing the key properties of the example multimedia formatting vocabulary currently implemented in the back-end of our Cuypers multimedia transformation engine.

Keywords

Document transformation, formatting objects, multimedia, hypermedia, Cuypers

1. Introduction

The large amount of Web content that currently needs to be designed, authored *and* maintained, has made the need for document engineering technology clear to Web developers. While once considered technical jargon used only by SGML-zealots, after the success of HTML [31], CSS [6], XML [7], XSLT [8] and related specifications, terms such as *structured document*, *stylesheet* and *document transformation* have become fundamental and well-known ingredients of everyday Web development.

With the growing diversity of devices and increasing amount of non-text information available on the Web, multimedia content providers also have growing needs for being able to style their Web content and use transformation techniques to adapt their presentations for a variety of delivery contexts. The goal of the work reported here is to extend the current suite of Web document engineering tools to enable their application to time-based multimedia presentations. This would allow the application of style to XML-based multimedia presentations, such as SMIL, and the generation of multimedia presentations using document transformations such as provided by XSLT. Both depend on an underlying *formatting vocabulary*.

Formatting vocabularies are used in many different ways on the Web. When an HTML page is styled using CSS, a style sheet designer uses the CSS formatting vocabulary to describe the intended result. When an XML source document in transformed into a printable document by XSLT, the author of the transformation sheet has to choose a target format that adequately describes the intended result, for example by using XSL's formatting vocabulary or a combination of HTML and CSS. The same applies to database-driven websites, where the server pages use a formatting vocabulary to define the look and feel of, say, a CSS styled HTML template, of which the content is filled in on the fly by querying a database.

A formatting vocabulary provides the set of terms that can be used to describe the intended presentation according to a specific model: this model is called a *formatting model*. A formatting model is designed to *explicitly specify the intended presentation behavior* of a document. Contrast this with, for example, the document model of HTML, that was intentionally designed to abstract from formatting details and to capture only the high-level structures of a document. CSS [6], XSL [32], and, less common, DSSSL [12] all define a specific formatting model with the associated vocabulary. All these models, however, have been originally defined for text-centric documents. To describe the intended behavior of multimedia presentations, designers often need features that are not, or only partly covered by these models.

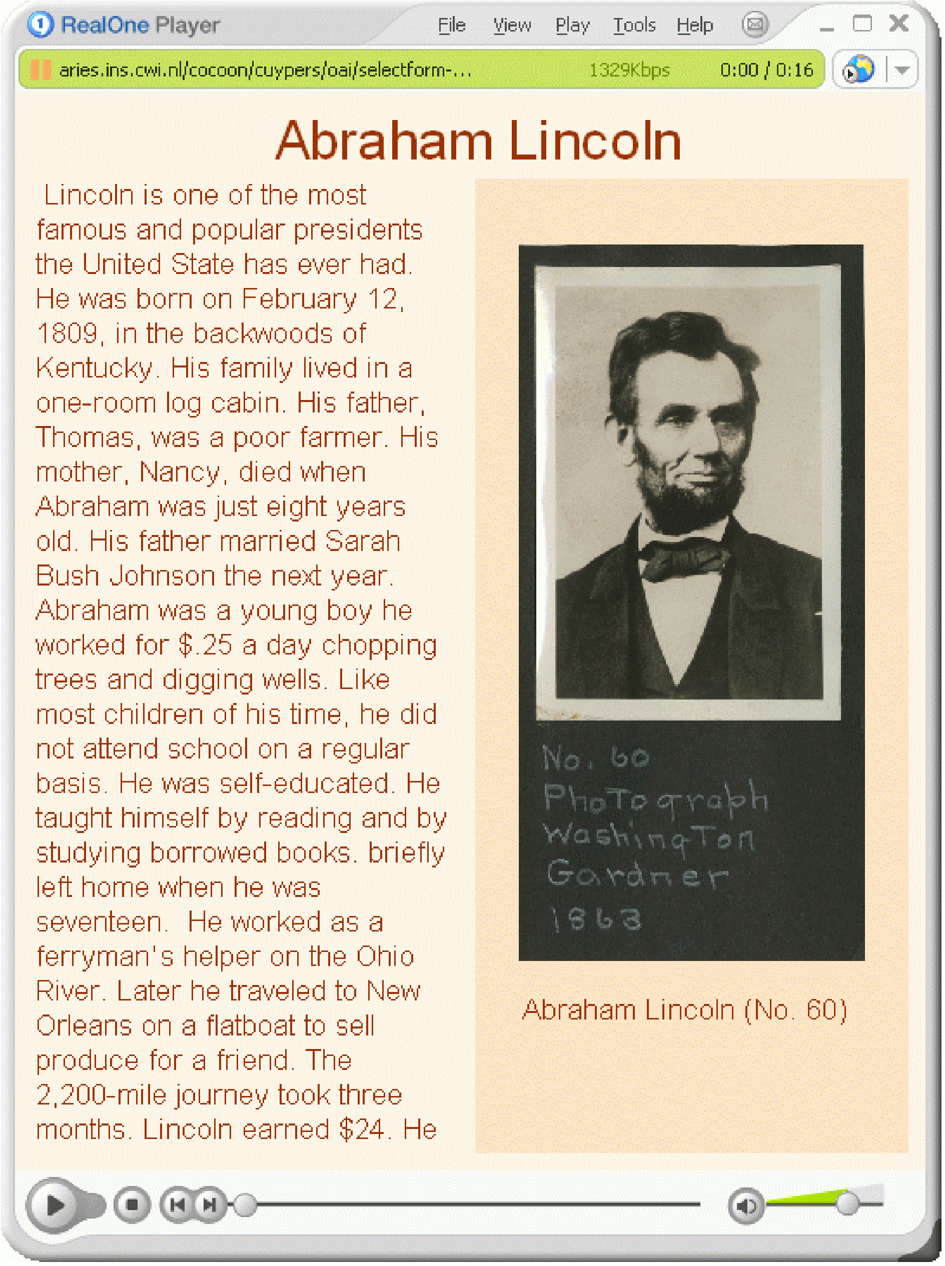
In this paper, we claim that the current Web infrastructure can be extended to allow Web developers to style and transform media-centric documents in a way that is currently only supported for text-centric documents. Section 2 explores a number of use cases for the types of transformations and applications of style that we wish to make to media-centric presentations. Based on these use cases, Section 3 describes the requirements for a multimedia formatting model. Section 4 first discusses the trade-offs for alternative approaches to the problem, and then describes the example multimedia formatting vocabulary implemented by our Cuypers multimedia transformation engine [18,28]. Finally, we discuss the pros and cons of our vocabulary, and discuss the applicability of multimedia formatting vocabularies in general.

2. Styling and transforming multimedia: Motivating scenarios

In the early days of the Web, most Web content was static and manually authored using plain text editors or special-purpose authoring tools. This was before the advent of style sheets, XML transformations and on-the-fly generated, database-driven HTML pages. While much has changed since, the authoring process of Web-based multimedia presentations still looks remarkably similar to the manual authoring of static HTML pages.

In this section we discuss to what extent the Web's current document engineering infrastructure can be applied to media-centric documents, and where it needs to be extended. In this discussion, we describe a number of use cases based upon W3C's open multimedia document format SMIL [33]. A large part of the discussion, however, could also apply to other (proprietary) multimedia document formats. In several use cases, explicit extensions to the current Web infrastructure are proposed. The only goal of these proposals is to illustrate the possibilities, and none of them should be read as a serious proposal that could, for example, be standardized ``as is'' by W3C.

To illustrate the requirements for styling and formatting vocabularies proposed in this paper, we have implemented some of these extensions in our prototype formatting model for time-based media presentations. While this formatting model will be further explained in section 4, we will motivate the requirements we derive in the following section through the use cases described in this section, many of which are based on the example presentation shown in Figure 1. The example shows a screen shot of a simple multimedia presentation about Abraham Lincoln. The presentation consists of a short biography to the left of a slide show of images of Lincoln, where each image is accompanied by a caption. Essential properties of the presentation include the timing of the presentation (e.g. the tempo of the slide show, the duration of the display of the biography text, etc.), transition effects that are used when moving from one slide to the other, visual alignment of the image box with the text, etc. Also note that the semantics of which caption belongs to which image is also directly communicated by the spatio-temporal layout (e.g. by aligning the caption and the image).



Abraham Lincoln

Lincoln is one of the most famous and popular presidents the United State has ever had. He was born on February 12, 1809, in the backwoods of Kentucky. His family lived in a one-room log cabin. His father, Thomas, was a poor farmer. His mother, Nancy, died when Abraham was just eight years old. His father married Sarah Bush Johnson the next year. Abraham was a young boy he worked for $.25 a day chopping trees and digging wells. Like most children of his time, he did not attend school on a regular basis. He was self-educated. He taught himself by reading and by studying borrowed books. briefly left home when he was seventeen. He worked as a ferryman's helper on the Ohio River. Later he traveled to New Orleans on a flatboat to sell produce for a friend. The 2,200-mile journey took three months. Lincoln earned $24. He

Abraham Lincoln (No. 60)

Figure 1: Screenshot of a SMIL 2.0 presentation about Abraham Lincoln

In the remainder of this section, we use this example to illustrate some use cases related to styling multimedia (section 2.1) and multimedia document transformations (section 2.2).

2.1. Stylable multimedia

The introduction of Cascading Style Sheets (CSS) allowed HTML authors to separate the description of the intended presentation of their pages from the main document structure and its contents. Later, CSS was also applied to other formats, including Scalable Vector Graphics (SVG [10]). While most multimedia formats, including SMIL, are not (yet) CSS ``stylable'', many of the advantages of using CSS for HTML or SVG content also apply to SMIL content.

Use case: stylable color schemes

Consider the screenshot of the small example SMIL presentation shown in Figure 1. The overall background color of the presentation is set in the `background-color` attribute of SMIL's `root-layout` element, and the darker background color for the slide show is set in the corresponding SMIL `region` elements. In a CSS-stylable version of SMIL, however, these background colors could also be specified by the CSS property with the same name. The usual advantages of using CSS would apply: it becomes easier to maintain a large collection of SMIL presentations with a consistent look using a shared style sheet; it becomes easier to change the color scheme without touching the original SMIL files, etc.

In the figure, the background color of the individual text items matches the background color of the presentation. In this particular case, the text items are RealText [21] items, that are explicitly styled to have a matching background color (the same applies to the foreground color of the text). This is a frequently occurring requirement when styling composite multimedia documents consistently [22]: either the constituent media items need to be styled to match the overall style, or, in case the media items are not stylable, the overall style needs to be adapted to match the style of the media items. In the formatting model discussed in section 4, we have implemented the first option and are investigating the second option (see [20]).

Use case: stylable transition effects

The slide show in the presentation of Figure 1 uses SMIL 2.0 transitions to fade from one image to the next. The transition effects used (in this case, a one second fade-in at the start and a ``clock wipe'' at the end) are hard coded in the SMIL file. Again, the transition effect would be a good candidate to move to a separate CSS style sheet. This would make it easier to maintain a large collection of SMIL files in a flexible way, supporting consistency not only among SMIL files but also within an individual SMIL file. While the CSS formatting vocabulary does not currently support transitions, it could be easily extended by, for example, importing SMIL 2.0's transition vocabulary. An example CSS rule (using such an extended vocabulary) to apply the fade-in and clock wipe effects to all images in the presentation could look like:

```
img { transInType:      fade;
      transInDur:       1s;
      transOutType:     clockWipe;
      transOutSubtype:  clockwiseTwelve;
      transOutDur:      1s;
    }
```

Our timed multimedia formatting vocabulary supports these transition properties. They are normally used on individual media items, but when specified on composite objects they are applied as a default value to all children of the composite. For example, to apply the same transition effect to all images in the slide show, one could specify this once by attaching the properties to the slide show object.

Use case: stylable visual layout

Note the vertical alignment at the top and bottom of the biography text on the left of Figure 1 with the dark background of the slide show on the right. This is realized in SMIL by explicitly specifying matching values for the `top` and `height` attributes for the associated `region` elements. In general, moving the specification of the dimensions and position of SMIL's regions to a CSS style sheet is straightforward when using absolute positioning -- CSS would not even need to be extended to make this possible.

In this case, however, one might want a higher level specification for the indented layout. We claim that the text-based alternative to absolute positioning can, in general, not be applied to multimedia. The relative formatting of text is based on the assumption that the text can be broken up in lines, columns and pages without affecting the semantics of the content. While this is often true for text, it is not for multimedia, where the semantic relations among the media items are often communicated via the visual layout [29]. In our example, a text-based formatter might insert a page break between the image and the caption, which would not yield the intended layout because it destroys the perceived relationship between the image and its caption. Fortunately, many other methods for higher level layout of media items have been proposed in the literature, such as the constraint-based multimedia layout of Madeus [14] and the constraint-based layout of SVG graphics described in [4]. In our multimedia formatting vocabulary, it is possible to layout media items using similar techniques (more details will follow in section 4). In our example, these techniques could be used to replace the absolute positioning. Instead, one could just specify that the biography needs to be positioned left of the slide show, and that it needs to be top and bottom aligned, and leave the rest to the application. This would give the formatting more flexibility while still respecting the semantics that need to be communicated. It would, however, require a significant extension of CSS.

Use case: stylable temporal behavior

The temporal behavior of the Lincoln presentation is also ``hard wired'' in the SMIL markup. For example, the ``speed'' of the slide show that is determined by the duration of each slide is specified using SMIL `dur` attributes. These durations are also good candidates for presentation aspects that could be subject to consistent styling in a separate CSS style sheet. One could go even further, and describe the complete temporal behavior in a style sheet. Such style sheets are sometimes referred to as *timesheets* [24,26,30].

2.2. Multimedia transformations

While CSS allows HTML authors to separate the presentation details from the main document hierarchy, it is not able to change the document hierarchy itself. This issue was addressed by the development of XSLT, which allowed authors to specify, in a standardized way, functional transformations from one XML tree structure to another. With XSLT, developers are able to reorder and regroup the content of an XML document in the way that is best suited for a particular presentation. Since SMIL has an XML syntax, the SMIL presentation in Figure 1 could have been generated from a source XML document using XSLT. Note that, since this approach would use SMIL 2.0 itself as the formatting vocabulary, there would be no need to extend existing, or invent new, vocabularies.

Use case: content selection and filtering

An obvious advantage of using a transformation would be the potential to tailor the SMIL presentation to the needs of the current application by filtering out all information that is irrelevant in the current context. Our example presentation could have been generated from an XML file that contains not only a section with information related to Lincoln, but also sections with information about other U.S. presidents. When one knows in advance that, for this particular presentation, only the Lincoln section is relevant, XSLT selectors could be used to select only this material and transform it to SMIL.

Use case: content grouping and ordering

XSLT allows the structure of the target SMIL file to be completely independent of that of the input XML file. One can thus use XSLT to reorder and regroup the content in the way that best suits the current context. For example, the source XML file may have ordered the Lincoln images based on ownership, while for the presentation, a temporal ordering that matches the events described in the biography would be more appropriate.

Use case: template-driven transformations

Presentations can be either template-driven or data-driven. A template-driven approach is common when the structure of the presentation is known in advance, and can be adequately described using a single template. XSLT (see section 2.3 of [8]) provides an abbreviated syntax for this special case.

In our example, when we know that the presentation is going to consist of a short biography text and a slideshow of images with captions, we can easily define a SMIL template for this structure, and then fill in the actual content later. When XSLT is used, the content can be selected from the XML source using XSLT selectors. One could also use many of the currently available database-driven approaches, including ASP, JSP and XSP [19,25,27]. These are also based on template techniques, where the template is filled with content selected using database queries.

Specifying the intended formatting based on a single template is relatively easy and well suited to achieving a consistent layout at the global level. In formatting languages such as XSL and DSSSL, and also in other typesetting software, the concept of a ``master page'' often functions as a template for achieving the same goal.

Use case: data-driven transformations

The main drawback of using a single template is that this approach is less suited in situations where the structure of the target presentation depends on highly variable input. This is often the case when the input itself is selected at runtime or generated automatically, as is the case in database-driven websites. For such websites, the input of the transformation process is the result of an on-line database query, and the characteristics of the results are likely to be different for each query. It is unlikely that one could specify an adequate ``one size fits all'' template that would match all different query results. XSLT addresses this by allowing the transformation to be decomposed into many smaller template rules instead of a single large template for the entire presentation. Each template rule contains a selector that matches a particular part of the input, and is responsible for transforming that part to the target output format. The main advantage of this approach is that input is no longer explicitly selected in the main template, but that the appropriate template rule is implicitly invoked by the transformation engine based on the input. In our example, the transformation sheet would no longer assume that the input contained a list of images that need to be transformed into a slide show. Instead, it would contain a template rule that matches a list of images and converts them into a slide show, *but this rule would only be invoked when the actual input contains a list of images*. Similar rules would be provided for other types of input.

Note that formatting vocabularies, such as those provided by CSS, XSL and also our own multimedia formatting vocabulary, typically support both template-driven and data-driven presentations.

Use case: media items with size constraints

An obvious advantage of a purely data-driven approach is that transformations are able to handle a wide variety of inputs, where the resulting presentation reflects these varieties. A serious drawback, however, is that it becomes harder to predict what the results would look like: unexpected input could result in highly unwanted, or even incorrect, presentations. Consider the slide show of our presentation which contains a number of unscaled images of Lincoln. These images just happen to be roughly the same size. Imagine what would happen, however, if the input unexpectedly contained an image that was, say, 20 times larger than the others. It would be very unlikely that adding this image to the slideshow unscaled would yield the desired result. Fortunately, languages such as SMIL, CSS and HTML allow authors to constrain the amount of screen resource an image is allowed to consume, and also how to achieve this (e.g. by scaling with, or without, preserving the aspect ratio, cropping, etc.).

Unfortunately, constraining dimensions of text boxes is more complicated than for images. CSS allows authors to specify the width and height of text boxes, but this often leads to unwanted behavior for overflow (i.e. the specified box is too small for its content). Currently, CSS options include displaying the content outside the specified box (`overflow = visible`), displaying only the content that happens to be inside the box (`overflow = hidden`) or inserting a scroll bar (`overflow = scroll`). It is not possible to specify alternative strategies, for example, the rendering engine could reduce the fontsize, kerning or line spacing until the content fits the prescribed box.

Another feature of XSL that is pertinent to multimedia is the ability to specify an explicit aspect ratio for an area (this is currently not supported in CSS). In our prototype multimedia formatting vocabulary, we have found it convenient to be able to use explicit aspect ratio properties, not only for images, but also for other media types, including text.

Use case: global resource constraints

The issues related to overflow strategies apply not only to individual boxes, but also to more global size constraints. For example, the presentation might be required to fit within a maximum screen size. In many multimedia presentations, one wants to ensure that such constraints are met without resorting to clipping or scrolling. Sometimes, the same constraints can be met by finding alternative layouts, for example by ``tweaking'' other style properties such as margins, font sizes, padding and border sizes, etc. Specifying these alternative strategies declaratively is often difficult.

Most readers will be familiar with the situation when preparing a slide show presentation and the last point of a bulleted list does not fit on the slide. There are, of course, several options to resolve this conflict: one could enlarge the list's text area, or decrease the font size, or decrease the vertical spacing, or even remove some content. Stating in the formatting vocabulary that a bulleted list should fit on one slide is one thing, but stating what combination of style properties should be tweaked to produce the best result, however, often depends on the situation and is hard to specify in advance. Similar problems apply to a page limit for conference papers. Explicitly stating that this paper should be formatted in 10 pages is relatively easy, but what measures the style sheet should take when the page limit is exceeded is significantly harder. Other examples include poster design, where all content needs to fit a fixed, single area; (vector) graphics, where subparts need to fit within areas already defined by higher-level objects; and TV-like media presentations, where multiple media items that play with potentially overlapping durations need to fit within the available amount of screen space.

In our example, if the Lincoln images in the slide show presentation were all landscape, it would be more appropriate to position the biography text under the slide show. For a smaller screen, it might even be necessary to schedule the biography before the slide show, or to connect the biography text with the slide show using a hyperlink. While such options are theoretically possible in XSLT, finding the most appropriate strategy depends to a great extent on the number and sizes of the media items in the input. Additionally, once an appropriate strategy has been selected, this potentially influences every rule in the transformation.

Current formatting models are not very well suited for formatting documents with such constrained designs. They often require designers to run their documents and style sheets through their rendering engine multiple times, each time tweaking the font sizes, margins, spacing and other style properties, until the style sheet produces the desired result (but only for that particular document). Alternatively, they are forced to alter the source document to reduce, for example, the amount of content so it will fit. In both approaches, most of the advantages of structured documents and style sheets are lost. In fact, the designer may even be better off without the structured-document paradigm and revert to using a direct manipulation, WYSIWYG (what you see is what you get) tool.

Since these types of resource constraints are very common in multimedia presentations, we have experimented with a multimedia formatting vocabulary that allows designers to explicitly state the global resource constraints. Creating a formatting object fails when it would generate a part of the presentation that violates the constraints. The transformation rule that created the object is notified of the failure and can thus try to create another formatting object, using a different layout strategy. Note that the formatting objects are sufficiently ``intelligent'' to be aware of the amount of resources they consume, and whether this consumption is within the limits specified. They are not, however, sufficiently intelligent to find an alternative solution when the constraints are violated -- this is left to the transformation rules.

Use case: re-evaluation of transformation results

In the example of the bullet point that doesn't fit on a slide, the main problem is that if one generates the slide from a structured document using, for example CSS or XSLT, one cannot know in advance whether the bullet will fit or not. This is because the actual space required to render the complete bulleted list depends on too many parameters. Only some of these parameters are known at the time the style sheet is applied (including the amount of text, font sizes, margins, etc.), but others are too low level and implementation specific (including the hyphenation, justification and kerning algorithms used to layout the text). In fact, the bullet point might not fit when using one CSS implementation, but may just fit when using another, because of these implementation differences.

In theory, it could be convenient if the text-rendering back-end could provide feedback to the transformation process about the resources consumed when the text is rendered, so that the transformation process could take appropriate action if certain resource constraints are violated.

In practice, however, this would require too close an integration of the transformation and rendering engines. The rendering of text is sufficiently complicated that it makes sense to implement the transformation engine and the rendering engine as two independent applications. Even stronger, many developers would like to keep the transformation and rendering processes completely independent, that is, they would like to be able to execute one process without the need to communicate with the other.

The layout of timed multimedia presentations, however, does not typically depend on complex, back-end specific formatting algorithms. Rather, it involves calculating the positions and dimensions of a dozen relatively large media clips instead of calculating the positions of (tens of) thousands of individual character glyphs. In many multimedia formats, including SMIL, the size and position of the constituent media items are specified explicitly in the document. The calculations required for the layout are thus, in the multimedia case, traditionally done by the ``front-end'' authoring application, and not, as in text, by the ``back-end'' rendering engine.

In our multimedia formatting model, every time a transformation rule generates its part of the presentation, the resource consumption of the partial result is automatically evaluated, so that the transformation rule can take appropriate action when certain constraints are violated.

Use case: Supporting multiple target formats

One of the advantages of current document engineering technology is that because the input is independent of the target document, one can support multiple output formats. Our example presentation was generated by a style sheet that produced SMIL 2.0, but, by adding other style sheets, one could generate similar presentations in other timed Web formats, for example, in HTML+TIME [23].

A drawback of this approach is that many design decisions have to be duplicated across the various style sheets. This is a well known problem, which is caused by the fact that our transformation does not separate the concerns of specifying what the presentation should look like from how it should be realized in a particular output format. This problem is addressed by formatting vocabularies such as XSL, that allow all design decisions to be made in a single transformation from the source XML to the intermediate format described by the formatting vocabulary, which can then be further transformed by a back-end engine into a number of other formats. Unfortunately, the current Web infrastructure does not provide such an intermediate format for timed multimedia presentations. CSS and XSL are not able to describe the timing-related features of presentations. Note that to be able to produce annotated presentations, such an intermediary format also needs to be able to model metadata on the various levels of the presentation hierarchy. XSL's `role` property, for example, allows each formatting object to refer to RDF metadata for this purposes.

Our multimedia formatting vocabulary is designed, on the one hand, to be able to describe in sufficient detail the intended presentation behavior of timed multimedia presentations, and, on the other hand, to be sufficiently abstract to be able to generate final form presentations in multiple timed formats. It also provides support for RDF metadata. Objects may refer to an external RDF source (similar to XSL's `role` property), but RDF metadata can also be included directly on all formating objects. This significantly simplifies the creation of RDF metadata during the transformation and allows, for example, to reuse information in the source document as metadata in the presentation. It can also be used to ``log'' information related to the transformation process as metadata in the presentation. Metadata can also be used to provide hints to browsers on the relationships among media items presented for increasing the accessibility of a document. For example, whether an audio item is (non-essential) background music, or whether it is an audible alternative to a text item. This functionality is provided in XSL via the `role` property.

3. Requirements for a multimedia formatting vocabulary

Our goal is to provide for time-based multimedia presentations equivalent functionality that we are used to for text transformations. This requires the ability to style XML-based multimedia presentations, such as SMIL, and the ability to generate multimedia presentations using document transformations such as provided by XSLT. Given this goal, we now derive a number of specific requirements for a multimedia styling and formatting vocabulary, based on the use cases discussed in the previous section.

- **Visual layout**

  A multimedia formatting model needs to be able to specify the *sizes and positions* of the areas in which the content is to be rendered. Unlike text, however, these calculations cannot be based on the position of the content in the text-flow. Inserting line, column and page breaks in a stream of multimedia items will, in general, not result in a coherent multimedia presentation. *Absolute positioning* of media items should be supported for authors that want full control over their layout, just as in text formatting. This is, however, too low level to be the only mechanism provided. In addition, explicit and high level *relative positioning* should also be supported, both among media items (e.g. ``this text label needs to be horizontally centered under this image'') as well as between parents and their children (e.g. ``align all slide objects with the bottom of their slide show object'').

  Consequently, we consider text-flow based layout formats (including timed formats such as HTML+TIME) unsuitable for modeling visual layout in an abstract multimedia formatting model, because text-based formatting often does not respect the semantics communicated by the visual layout. In addition, formats that provide only absolute positioning (such as SMIL 1.0) or only low-level relative positioning (such as SMIL 2.0) do not provide sufficiently high-level support to provide the basis for an abstract multimedia formatting model.

- **Visual style**

  In addition to the position and sizes of the areas, the model also needs to describe the *visual style* of presentations, including colors, fonts, paddings, borders, background images etc. Note that, in general, it does not suffice to specify these style properties at the overall presentation level: the constituent media items often need to be (re)styled to get the result. For example, in Figure 1, a consistent background color through-out the presentation can be achieved by enforcing the generation of correctly colored text items. In this particular case, the text content itself can be styled since it is generated. For applications that also include ``non-stylable'' content, it may be necessary to use only colors and other style properties that match the included content. A formatting model thus needs to be able to account for both dependency directions.

- **Temporal structure**

  A multimedia formatting language should be able to model the full temporal behavior of today's multimedia presentation formats. This includes not only the basic orchestration (i.e. specification of when media items appear on and disappear from the screen), but also which synchronization relations need to be maintained (e.g. for audio streams that need to play lip sync with a video stream) and for controlling the speed and other basic behavior of animated content. Also note that interactive multimedia presentations do not necessarily have a linear temporal structure. The exclusive element in SMIL 2.0 (`excl`), for example, allows multiple timelines to be selected interactively.

- **Temporal style**

  In multimedia it often makes sense to speak of the ``temporal style'' of the presentation, in addition to the visual style. Part of that temporal style is already determined by the orchestration of the document (e.g. the fast cuts of a rock video-clip style versus the long scenes of a documentary style). Other temporal aspects, such as the type and duration of the transitions used between scenes, are also an essential part of the style of the document and need to be expressible in the formatting vocabulary. Furthermore, in some cases it is convenient to model background music not as one of the main media items of the presentation, but as an optional, and easily changeable style property.

- **Support for top-down transformations**

  A template approach is required to allow the extraction of media items from a known collection to be placed in a consistent layout. Such a template specified at the top level needs to be sufficiently flexible that it can be applied to all possible inputs. This is more difficult for multimedia than for text (since texts will fit in the template of a reasonably designed master page, independently of the number of columns, margin sizes etc.).

- **Support for bottom-up transformations**

  In data-driven multimedia applications, where the number, size and type of the media items that need to be displayed is not known until runtime, template design is much more difficult. Bottom-up transformation rules allow the choice of a particular composite formatting object to depend on the layout of its children. For example, two portrait images might fit together on the screen only when stacked horizontally, while their landscape counterparts might fit only in a vertical arrangement. A combination of both would fit only when displayed one after the other in time. For data driven applications, it might not be known in advance whether the images are landscape and portrait, so the style sheet would need to be able to make the (data-driven) decision at runtime. One can also imagine a hybrid approach, where a top level template specifies, for example, a small bar on the top used for logos and other branding purposes, on one of the sides a sidebar with links for the main navigation, with the remaining screen space filled in by a bottom-up, data-driven approach.

- **Abstract from target output format**

  A formatting model should not only allow detailed specifications of intended presentation behavior, but should enable a designer to specify these independently from low level implementation issues that relate to *how* the intended formatting is realized in the final rendering of the document. In addition, it should support (RDF) metadata annotations. These requirements are no different from the text case. XSL's formatting vocabulary, for example, can be rendered to a wide variety of (annotated) final form output formats. For multimedia, it requires that the formatting model be able to express spatial, temporal and multimedia linking aspects, detailed in previous requirements, independently of the target output format. A higher abstraction level and a declarative model also makes, in general, writing style sheets easier. At the same time, the (declarative) model should not prevent the author from using (procedural) approaches for the specification of behavior that is not covered by the model. For example, one might allow scripting to specify complicated animations that go beyond the scope of the formatting model.

- **Support for resource-constrained presentations**

  Multimedia presentations are immediately confronted with the spatial layout resource constraints (screens have a fixed size) and temporal resource constraints are also common. In a text-centric flow object model, a basically linear sequence of glyph objects can in principle be split at virtually any arbitrary point. As long as the original order is respected, the semantics will also remain the same. The order of multimedia objects, in contrast, can be reshuffled to a much greater degree, in both space and time. The system, however, needs to have more information about possible layout configurations in order to be able to propose acceptable solutions that do not violate the semantics.

- **Support for generating consistent presentations**

  A multimedia formatting model, just as for text, needs to support designers in creating a consistent presentation. First, similar content should be formatted in a similar way, where exceptions can be explicitly stated by the designer. Second, design decisions made at the beginning of the formatting process should remain valid until a new decision is made explicitly. Both selectors and property inheritance from CSS can be reused to provide style consistency in multimedia presentations, but, as detailed in previous requirements, extensions are required for consistent styling in composite documents.

4. The Cuypers Multimedia Formatting Vocabulary

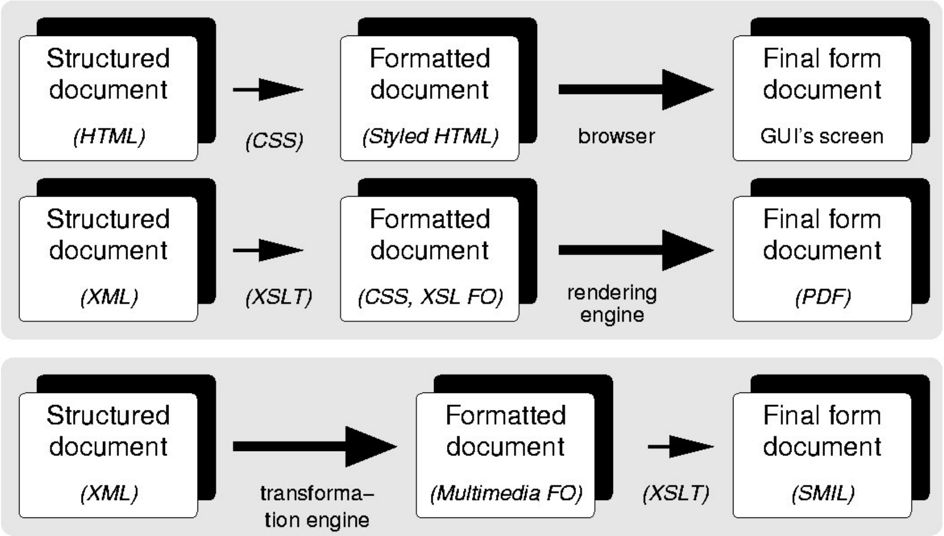


Figure 2: Computational differences between CSS/XSL FO versus multimedia formatting

For styling multimedia presentations, we do not intend to reinvent CSS's styling vocabulary, but rather to extend it and provide extra functionality for multimedia-specific style properties such as timing and transitions. Adding the extra vocabulary needed could be part of the currently ongoing work within W3C on new levels of CSS. The real challenge would be to convince the implementors of multimedia authoring and player software to support CSS-stylable versions of SMIL and related document formats.

The requirements discussed in the previous section have a major impact on the transformation process. As such, a fully-fledged multimedia formatting vocabulary, suitable as a target output for multimedia document transformations, would require a more significant extension of the current infrastructure. An approach to satisfying these requirements is to develop a highly adaptive multimedia formatting vocabulary. The presentation language described by Bes and Roisin [5] can be regarded as a first attempt towards such a model. This language not only allows designers to define some resource constraints that need to be satisfied, but also provides the formatting engine with hints that could help choose the most appropriate resolution strategy in case the constraints are violated. Examples of such hints include assigning priorities to media items and fall-back rules. This approach has two main advantages. First, a transformation to such a language can be, as in the case of text, a relatively simple, functional specification, which can be specified using standard tools such as XSLT. Second, selecting the most appropriate adaptation strategy can be deferred to a back-end application, for example a player at the client-side. This would both relieve the server and make optimal use of the delivery context information available at the client side (e.g. device characteristics, etc.).

Such a highly adaptive language, however, also has serious drawbacks. First, the language would need to support a wide variety of different adaptation strategies that do not change the semantics of the presentation in an unacceptable way. It should do this in a single, declarative, application-independent model. This requires a thorough understanding of the adaptation needs of Web-based multimedia presentations. Second, to achieve a sufficient level of interoperability on the Web, such a language needs to be standardized and supported by players on multiple platforms. To address these problems, we take a different approach that leaves more room for experimenting with different strategies and does not need a standardized multimedia formatting language.

Figure 2 compares the processing chains of HTML, with CSS, and XML with XSLT/XSL FO, with our approach to multimedia formatting. At first sight, these processes may look alike. In all three cases, the major design decisions that define the look and feel of the final presentation are declaratively specified in the first step. The task of the second step is merely to realize the declarative specification in terms of the specific target format. Note that the intermediate output should be sufficiently detailed to give the author full control over the look and feel of the presentation, while sufficiently abstract to allow the look and feel to be realized in different target formats. For example, CSS styled HTML can be rendered by browsers on different platforms and different GUI toolkits, and also to paper. The same applies to CSS styled XML an XSL FO and a multimedia formatting language. The multimedia formatting objects discussed below can be rendered in SMIL 1.0, SMIL 2.0 and HTML+TIME with absolute positioning in CSS.

From a computational perspective, however, the multimedia case differs from both text cases. In the text cases, the first step is computationally relatively simple (e.g. attaching style properties to the HTML document tree or performing a simple functional transformation from the source XML tree to the target XML tree), and can be adequately specified in a CSS or XSLT style sheet. The hard work is done in the second phase, where a browser or other dedicated rendering engine needs to render the declarative, high-level specification in terms of the much lower-level facilities provided by the target format (for example, in terms of the primitives provided by the GUI toolkit upon which the browser has been built, or in terms of a final form document format such as PDF). For text, rendering typically involves the intricate text rendering algorithms discussed before, such as kerning, hyphenation, pagination, etc. These algorithms are too computationally complex and too low level to be practically implemented on the style sheet level.

In our multimedia case, we allow the transformation engine to try a transformation rule, verify the results of applying the rule against the specified resource constraints and choose an alternative rule when the constraints are violated. While this makes the first step computationally complex, it has the advantage that there is no need to predefine the possibly application-dependent adaptation strategies in the formatting vocabulary, since such strategies are part of the (application-dependent) transformation process. Once this hard work is done, however, the second step is relatively easy: transforming the multimedia formatting object tree into a concrete delivery format (e.g. SMIL) can be specified by a simple XSLT transformation. This second step is straightforward because the calculations for the visual layout and timing of the media items are performed during the first step, which has produced a formatting tree that is known to meet the specified requirements.

Note that this subtle difference has a major impact on the standardization requirements. For text, standardization of the intermediate result (e.g. the CSS-styled HTML or XSL formatting object model) is crucial, because the processing chain depends on (commonly available) browsers or rendering engines to implement them. The same applies to the approach of Bes and Roisin, where the adaptation is carried out at the back-end. In our approach, the intermediate result does not need to be standardized as long as it has an XML serialization and is accompanied by a suitable transformation sheet that transforms the multimedia formatting tree to a standardized delivery format (for example, SMIL).

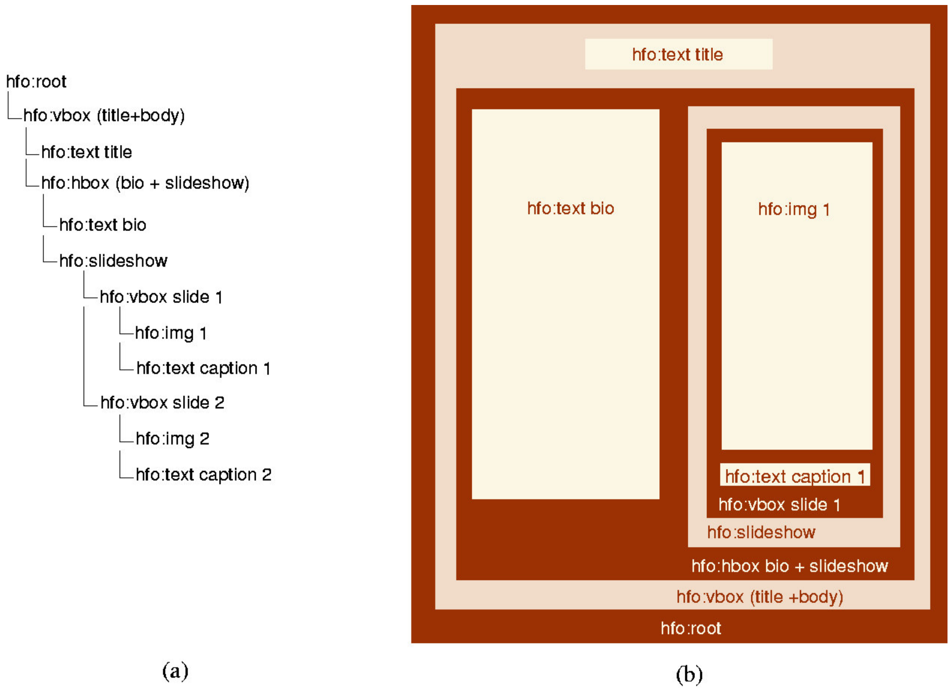


**Figure 3: (a) Cuypers' Hypermedia Formatting Object (hfo) tree corresponding to the Lincoln presentation in Figure 1;
(b) Visual box representation of the presentation (second slide not shown)**

To explore the use of application-specific adaptation in resource constrained multimedia presentations, we implemented a small set of multimedia formatting objects in our Cuypers multimedia transformation engine . We focused on the formatting aspects related to constrained layouts and temporal behavior.

The example presentation in Figure fig:lincoln-screen has been generated by our Cuypers system, using this vocabulary. The presentation is generated as a response to a query to an on-line database . The database has retrieved six relevant media items of which four (title, text, caption, image) are visible in the screendump. The two not shown are the image with its caption of the second slide. In Figure fig:lincoln-fo (a) the hierarchical structure of the formatting objects is shown. Figure 3 (b) shows the visual area structure of this hierarchy explicitly. Note that to avoid transformations having to keep track of a separate temporal and spatial hierarchy, the formatting object tree structure represents both the temporal and spatial containment relations. That is, a child object is displayed within the visual area of its parent and during the temporal duration of its parent.

The following are examples of formatting objects that were used in the creation of the Lincoln presentation:

- **hfo:image**

  This atomic object models an image and contains contextual information about the image. This includes the usual information relevant to the presentation rendering, such as height, width, URL etc., but can also include preferred, minimum and maximum duration and meta data information such as copyright notifications. Note that the original Dublin Core metadata from the database is preserved by all atomic objects, and can thus be preserved for inclusion in the final presentation.

- **hfo:text**

  Text objects are also atomic and used for displaying all textual items in the presentation. These include titles, captions and longer text items such as the biographical text presented in the Lincoln presentation. Finding the right dimensions of the area associated with the text is of crucial importance, especially when the length of the content is not known in advance and scrollbars are not allowed. Cuypers' text object possesses a wide range of features to help the style sheet author establish appropriate dimensions, which may depend on the number of characters in the text, preferred font sizes, aspect ratios (a relatively large aspect ratio can be used for titles and captions; longer text, in contrast, can be assigned a smaller aspect ratio). Similar features are available for establishing a suitable duration that allows sufficient time for end-users to read the text. Most numerical values, such as preferred width, height, duration, fontsize and aspect ratio, can be specified using a single value, but also by specifying allowed intervals. The latter is often preferred because it permits a higher degree of adaptation when necessary.

Note that creation of all atomics fails immediately when their dimensions exceed the global resource limitations. The final dimensions in the presentation are the result of interplay between the properties of an individual object, and those of other related objects, e.g., through alignment. Such alignments are modeled as properties of composite objects, such as the ones described below.

- **hfo:vbox**

  A vertical box presents its children in a top-to-bottom order, and is neutral from a timing perspective (by default, it plays all its children in parallel, starting at the same time). Properties such as visual alignment define the characteristic features of the vbox object. In the example presentation, a `center` alignment is set on the first child (the title) of the outer vbox. This ensures that the title is centered above the content of the second child (the body of the presentation). In addition to alignment properties, which each relate to a single child, the `equalHeight`, `equalWidth` and `equalDuration` properties describe relations among children. For example, an `equalDuration` relation is established between the children of the outer vbox in the presentation to ensure that the title is visible during the whole presentation.

- **hfo:hbox**

  A horizontal box is similar to the hfo:vbox with the difference that the children are ordered in a left-to-right fashion. In the presentation, both `equalWidth` and `equalHeight` relations are defined on the biography text and the slideshow. Note that in some cases the functionality of alignment properties and equal height, width and duration properties can overlap. For example, the `equalHeight` on the biography text and slideshow can also be achieved by setting `topAlign` and `bottomAlign` properties. This, however, only holds if the two objects overlap on the y-axis, which is the case in the presentation. Likewise, `leftAlign` and `rightAlign` requires overlap on the x-axis. This is *not* the case in the Lincoln presentation: the biography text and slideshow have no overlap on the x-axis. To ensure equal widths, heights and duration, independent of the positions of the media items involved, the `equalWidth`, `equalHeight` and `equalDur` are needed.

- **hfo:slideshow**

  A slideshow presents its children one after the other. It is neutral to visual formatting, but by default a `center` alignment property is set to center all its children within the slideshow. In addition, a default `equalDuration` property ensures that all children are shown with equal durations. Properties to specify transitions can also be applied to slideshows. This is realized by setting the properties `transIn` and `transOut` to one of the defined values. Although transition effect properties are set on the slideshow, the effect is that all of its children should have the specified transition. Transition effects set on the parent are inherited by its children and thus trickle down to the actual media items showing the effect. It should be noted that most output formats, such as SMIL, define transitions only on the actual media items (c.f. CSS where inheritable style properties set on a parent are inherited by their children). Other inheritable properties include padding and border values, foreground and background color, etc.

- **hfo:root**

  The root object forms the base of every presentation. It represents the outer window that contains the complete presentation and also stores the delivery context of the presentation, including the maximum screen size, bandwidth and other relevant CC/PP [15] parameters. Note that multi-window presentations can be modeled by creating multiple root objects.

Our formatting implementation uses constraint solving techniques to calculate the value of the coordinates. Many descriptions of constraint-based multimedia presentations have been reported in the literature, including André's work on constraints generated by a planning system [1,2,3] and the constraints generated by the Madeus authoring system [14]. In our approach, a large majority of the constraints remain hidden from the transformation rules because they are automatically generated by the implementation of the formatting objects [11,18].

5. Conclusion

Multimedia content providers need to publish their content for a wide variety of Web devices and to facilitate the creation of on-line presentations from content stored in structured XML documents or multimedia databases. To do this effectively, the well-known advantages of document engineering techniques need to be made applicable to multimedia content. The current Web infrastructure, however, is not yet able to provide the same level of support as for text. Style sheet and document transformation technology can only be successfully applied to media-centric presentations when multimedia style sheet authors can use a multimedia formatting vocabulary that is

1. sufficiently powerful to express the intended presentation behavior of their documents, and
2. sufficiently abstract to protect them from the low level details and syntactic idiosyncrasies of the final form multimedia presentation language.

This article describes a number of use cases for stylable multimedia and multimedia transformations that were used to derive the requirements for a multimedia formatting vocabulary. We discussed our implementation of a small prototype vocabulary that allows documents to be transformed to a formatting object tree that describes the intended behavior of timed multimedia presentations. Such presentations are often difficult to describe in terms of page characteristics and text-flow. Instead, their behavior is better described in terms of the temporal orchestration and synchronization of the media items, transition effects and styling of both the global presentation and its constituent media content. Our formatting vocabulary focuses on these temporal aspects. Because the vocabulary abstracts from the delivery format's syntax, various delivery formats can be supported (SMIL 1.0, SMIL 2.0, HTML+TIME are currently implemented). We are aware, however, that by concentrating on these aspects and treating text objects as ``black boxes'' we lose the power to do our own text-formatting. This problem is also reflected in the SMIL and SVG [10] delivery formats.

While the layout model of multimedia formats such as SMIL are based on explicit specification of the size and position of all media items, the required calculations can be delegated to the formatting objects. If violations of the specified resource constraints are detected during these calculations, alternative transformation rules can be selected to resolve the conflict. By moving the selection of adaptation alternatives from the rendering process to the transformation process, the transformation can make application-dependent trade-offs during this selection, or implement new, application-dependent adaptation strategies. While this requires a more complex, and non-standard, transformation process, it provides a greater level of flexibility when exploring adaptation strategies that cannot be provided when all adaptation strategies need to be predefined by a generic, standardized formatting vocabulary. It also simplifies converting the formatting object tree to a (standard) target format, which can be realized using a simple XSLT style sheet.

In future work, we plan on extending our multimedia vocabulary for more complex presentations that go beyond the TEX-like box-oriented model presented here. Perhaps more importantly, we will focus on an architecture for the transformation rules that allows a more intelligent way to employ this vocabulary. In particular, we want to be able to use application-specific and design-specific knowledge on the Semantic Web to allow transformations to make informed choices when selecting one layout or style strategy above the other. See [29] for details.

Acknowledgments

Part of the research described here was funded by the Dutch national Token2000/IRP and NWO/NASH projects. Oscar Rosell Martinez implemented the first version of the Cuypers HFO library. The screen shots are taken from a Cuypers-based demonstrator. The first version of the demonstrator was developed in the context of the MAENAD project by Joost Geurts and Suzanne Little in cooperation with Jane Hunter at DSTC, Queensland, Australia. The demonstrator uses Dublin Core metadata provided by the Open Archives Initiative. Media content has been adapted from the Web site of the Abraham Lincoln Presidential Library and Museum, and the Rare Book and Special Collections Library at the University of Illinois.

Bibliography

1

E. André, W. Finkler, W. Graf, T. Rist, A. Schauder, and W.Wahlster.
WIP: The Automatic Synthesis of Multimodal Presentations.
In *Intelligent Multimedia Interfaces* [9], pages 75-93.

2

E. André, J. Müller, and T. Rist.
WIP/PPP: Knowledge-Based Methods for Fully Automated Multimedia Authoring.
London, UK, 1996.

3

E. André and T. Rist.
The Design of Illustrated Documents as a Planning Task.

In *Intelligent Multimedia Interfaces* [9], pages 94-116.

4
G. J. Badros, J. J. Tirtowidjojo, K. Marriott, B. Meyer, W. Portnoy, and A. Borning.
A Constraint Extension to Scalable Vector Graphics.
In *The Tenth International World Wide Web Conference* [13], pages 489-498.

5
F. Bes and C. Roisin.
A Presentation Language for Controlling the Formatting Process in Multimedia Presentations.
In *Proceedings of Document Engineering 2002*, 2002.

6
B. Bos, H. W. Lie, C. Lilley, and I. Jacobs.
Cascading Style Sheets, level 2 CSS2 Specification.
W3C Recommendations are available at http://www.w3.org/TR, May 12, 1998.

7
T. Bray, J. Paoli, and C. M. Sperberg-McQueen.
Extensible Markup Language (XML) 1.0 Specification, February 10, 1998.
W3C Recommendations are available at http://www.w3.org/TR.

8
J. Clark.
XSL Transformations (XSLT) Version 1.0.
W3C Recommendation, 16 November 1999.

9
M. T. M. (Editor).
Intelligent Multimedia Interfaces.
AAAI Press, 1993.

10
J. Ferraiolo.
Scalable Vector Graphics (SVG) 1.0 Specification.
W3C Recommendation, 4 September 2001.

11
J. Geurts, J. van Ossenbruggen, and L. Hardman.
Application-Specific Constraints for Multimedia Presentation Geneneration.
Technical Report INS-R0107, CWI, May 31, 2001.

12
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.

13
IW3C2.
The Tenth International World Wide Web Conference, Hong Kong, May 1-5, 2001. ACM Press.

14
M. Jourdan, N. Layaïda, C. Roisin, L. Sabry-Ismaïl, and L. Tardif.
Madeus, an Authoring Environment for Interactive Multimedia Documents.
In *Proceedings of ACM Multimedia '98*, Bristol, UK, 1998.

15
G. Klyne.
CC/PP Attribute Vocabularies.

Work in progress. W3C Working Drafts are available at http://www.w3.org/TR, 21 July 2000.

16

C. Lagoze and H. V. de Sompel.
The Open Archives Initiative: Building a low-barrier interoperability framework.
JCDL2001, 2001.

17

S. Little, J. Geurts, and J. Hunter.
Dynamic Generation of Intelligent Multimedia Presentations through Semantic Inferencing.
In *6th European Conference on Research and Advanced Technology for Digital Libraries*, pages 158-189. Springer, September 2002.

18

O. R. Martinez.
Design dependencies within the automatic generation of hypermedia presentations.
Master's thesis, Technical University of Catalonia, June 30, 2002.
Published as CWI technical report INS-R0205.

19

Microsoft Corporation.
ASP.NET Web: The Official Microsoft ASP.NET Site.
See http://www.asp.net/.

20

F. Nack, M. Windhouwer, L. Hardman, E. Pauwels, and M. Huijberts.
The Role of High-level and Low-level Features in Style-based Retrieval and Generation of Multimedia Presentations.
New Review of Hypermedia and Multimedia, 7:39-65, 2001.

21

RealNetworks, Inc.
RealText Authoring Guide.
See http://service.real.com/help/library/guides/realtext/realtext.htm.

22

L. Rutledge and P. Schmitz.
Improving Media Fragment Integration in Emerging Web Formats.
In *Proceedings of the International Conference on Multimedia Modeling 2001 (MMM01)*, pages 147-166, CWI, Amsterdam, The Netherlands, November 5-7, 2001.

23

P. Schmitz, J. Yu, and P. Santangeli.
Timed Interactive Multimedia Extensions for HTML (HTML+TIME): Extending SMIL into the Web Browser.
W3C Note are available at http://www.w3.org/TR, September 1998.

24

P. L. Schmitz.
A Unified Model for Representing Timing in XML Documents.
WWW9 Workshop: Multimedia on the Web, 2000.

25

Sun Microsystems, Inc.
JavaServer Pages.
See http://java.sun.com/products/jsp/.

26

W. ten Kate, P. Deunhouwer, and R. Clout.
Timesheets - Integrating Timing in XML.

WWW9 Workshop: Multimedia on the Web, 2000.

27
The Apache Software Foundation.
XSP Logicsheet Guide.
See http://xml.apache.org/cocoon/userdocs/xsp/logicsheet.html.

28
J. van Ossenbruggen, J. Geurts, F. Cornelissen, L. Rutledge, and L. Hardman.
Towards Second and Third Generation Web-Based Multimedia.
In *The Tenth International World Wide Web Conference* [13], pages 479-488.

29
J. van Ossenbruggen and L. Hardman.
Smart Style on the Semantic Web.
In *Semantic Web Workshop, WWW2002*, May 2002.

30
J. van Ossenbruggen, L. Hardman, and L. Rutledge.
Integrating Multimedia Characteristics in Web-based Document Languages.
Technical Report INS-R0024, CWI, December 2000.

31
W3C.
XHTML 1.0: The Extensible HyperText Markup Language: A Reformulation of HTML 4.0 in XML 1.0.
W3C Recommendation, January 26, 2000.

32
W3C.
Extensible Stylesheet Language (XSL) Version 1.0.
W3C Recommendation, 15 October 2001, 2001.

33
W3C.
Synchronized Multimedia Integration Language (SMIL 2.0) Specification.
W3C Recommendation, August 7, 2001.
Edited by Aaron Cohen.

Footnotes

... option
We have made the RealText stylable by using a parameterized XSP page that generates the RealTexts. The background color of the global SMIL or HTML presentation is then passed as a parameter on the URL that refers to the XSP page.

... absolute
Note that the term ``absolute'' and ``relative'' positioning often means different things in different worlds. In text-centric layout models such as CSS, the ``absolute'' refers to positioning that is not relative to the text-flow, while in media-centric models such as SMIL, the term refers to positioning that is not expressed relatively to (i.e. as a percentage of) the parent window or other media items.

... engine
The XML serialization of the example presentation's HFO tree is available from `http://www.cwi.nl/~media/cuypers/`. An on-line demo of the system and XSLT transformation sheets to transform HFOs to SMIL 2.0, SMIL 1.0 and HTML+TIME are also available from this location.

... database

In this case a database that is based on the Dublin Core metadata provided by the Open Archives Initiative [16]. The metadata is used not only to find relevant texts and images in the database, but also to infer semantic relationships among the texts and images that can be used as a basis for a coherent multimedia presentation (see [17] for a more elaborate description of the use of Dublin Core for presentation generation).

... values

The effects supported are dependent on the presentation back-end, in our case the SMIL player or HTML+TIME browser.