Centrum voor Wiskunde en Informatica
REPORT*RAPPORT*

Integrating Multimedia Characteristics in Web-based Document Languages

J.R. van Ossenbruggen, H.L. Hardman, L.W. Rutledge

Information Systems (INS)

INS-R0024 December 31, 2000

Report INS-R0024
ISSN 1386-3681

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

CWI is the National Research Institute for Mathematics and Computer Science. CWI is part of the Stichting Mathematisch Centrum (SMC), the Dutch foundation for promotion of mathematics and computer science and their applications.
SMC is sponsored by the Netherlands Organization for Scientific Research (NWO). CWI is a member of ERCIM, the European Research Consortium for Informatics and Mathematics.

Copyright © Stichting Mathematisch Centrum
P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

Integrating Multimedia Characteristics in Web-based Document Languages

Jacco van Ossenbruggen, Lynda Hardman and Lloyd Rutledge
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

ABSTRACT
A single multimedia document model needs to include a wide range of different types of information. In particular, information about space and time is essential for determining the spatial and temporal placement of elements within a presentation. Each information type included in a document model requires its own structuring mechanisms. The language used to express the document model has to be able to encapsulate the plurality of required structures. While this is a process that can be carried out relatively easily during the initial design of a language, it is more difficult in the case that a particular document language already exists and extra multimedia characteristics are required. For example, one could consider adding temporal information to an existing "static" document language. We investigate the underlying problems of superimposing a new document feature on an existing language and discuss possible strategies for integrating the required extra information in a modified document description language.

1998 ACM Computing Classification System: H.5.4, H.5.1, I.7
Keywords and Phrases: multimedia document characteristics, time-based, spatial, text-flow, composition structures, Web markup languages.
Note: The research reported here has been carried out under the project "Structured Document Languages for Hypermedia"

1. Introduction

As we gain more understanding about the information types included within complex multimedia presentations, we are able to build models and tools that allow multimedia document descriptions to be declarative. A declarative description of a multimedia document allows the flexible processing of documents in ways already long familiar in the text-based document processing community. With the development of a more mature document processing infrastructure for the World Wide Web (WWW), multimedia documents are already part of the W3C "language kit" since the introduction of SMIL [31]. SMIL, while a good example of a multimedia document language that can be processed and played using Web-compatible tools, is only one particular language. A more useful way to provide support for multimedia on the Web is to allow the creation of application-specific multimedia languages using language "parts" supplied by W3C. (This is equivalent to the current move away from an all-encompassing HTML language to allowing users to define suites of languages specified using XML and related efforts.)

While developing application-specific languages in the "text-based" world is a comparatively solved problem, providing a framework for specifying multimedia languages is more complex and, as yet, a not completely understood problem. The aim of this article is to show that a multimedia language includes a number of different aspects, such as time and space, and that when multimedia characteristics are added to existing languages (historically often text-based and static) problems with combining the existing and additional information are encountered. In this article we explain what these problems are, and suggest ways of solving them.

In essence, a multimedia document includes information about temporal relationships, spatial information and composition information - for example combining shots into continuous scenes, or scenes into related groups. These different aspects need to be incorporated in a single multimedia document

model, where each aspect may have its own structure. For example, temporal information can be recorded as hierarchical information - where groups of elements are played together and groups can be played one after the other. On the other hand, temporal information may be recorded as individual constraints between elements. Whatever the chosen structure for the temporal information, it has to be included, along with structures for other information types, within the document model.

When designing a new document model, the problem that has to be solved is that the document structures have to be chosen to allow the combination of different characteristics within the new model. This is already a non-trivial problem, since there are likely to be multiple aspects that have to be incorporated in the model. If, however, a document model already exists and it needs to be extended with a new characteristic, for example adding temporal information to a text-flow based model, then the existing structures place restrictions on the ways the temporal information can be integrated. In the example, the ideal would be to retain the text-flow-based structure and add extra temporal information to it. This, however, is not a trivial process.

We look at three cases of introducing a new structure to an existing structure. The first case is where the additional structure is similar to the existing structure, in which case extra information needs to be included, but no major structural changes are needed. The second case is where both structures are different, but one is sufficiently simple that it can be flattened and the extra information added to the other structure without losing crucial information. The third case is where both structures are different and complex, in which case some means is needed for full integration of the new structure in the model.

The structure of the paper is as follows. We first discuss the ways in which temporal, spatial, composition and text-flow information can be expressed within a document structure. We then look at the three cases of combining two different structures. In the third, most complex, case we discuss the options for introducing the new structural information to the existing document model. Throughout the paper we illustrate our arguments with examples from the W3C set of languages. We end with a summary and a conclusion.

2. Multimedia versus text-based documents

2.1 Multimedia document characteristics

A multimedia document needs to contain a number of, potentially independent, characteristics. These are aspects of a multimedia document model [16, 14, 15]. In essence, the different types of information that may be included within a multimedia document are the following:

- content — the video, audio, text or graphics elements that are presented to the viewer;
- style — information affecting the appearance of content items, such as font styles for text or background colour;
- linking — specification of the beginning and end of links, along with transition information when traversing the link;
- temporal information — start times and durations of elements;
- spatial information — placement information as to where elements are positioned, and how large they are;
- composition — grouping of items to manage complexity when dealing with large numbers of items.

When combining different characteristics within a single model, content, style and linking are relatively easy to incorporate on top of any existing document structure. That is, content is generally contained within a single element in the document and as such has no influence on the rest of the document structure. Style and linking information can be contained within a single element, or can be

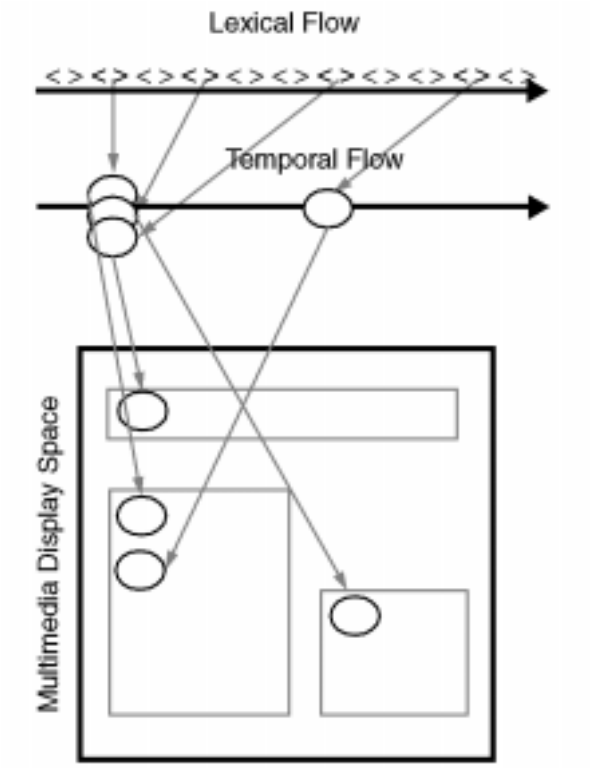


Figure 1: Lexical Flow, temporal and spatial layout for multimedia

applied from outside the main document structure. We thus do not include these types of information in our further discussion.

In this section we explore the ways temporal, spatial and composition can be dealt with in multimedia document models. These types of information tend to be what we call encompassing - that is, they describe relationships among large numbers of elements in the document, and it is these relationships that constitute the multimedia document structure. Temporal information establishes temporal relations between document objects, such as that two play simultaneously, or that one plays after another. Objects can appear next to each other on the screen. General composition groups document objects together as members of one collection. These relationships are typically independent of the position of the objects in the lexical flow of the document format, as illustrated in Figure 1.

Encompassing information cannot be represented in the document encoding by assigning descriptive attributes to individual elements. Encoding such encompassing information would require having elements refer to each other or having multiple elements be contained in the same parent element.

2.2 Temporal information

The inclusion of temporal information in a document model is the determining characteristic of multimedia. Temporal information itself can be modelled in several different ways. For example, timing information may be relative to a timeline, where each element is given a start time and duration in absolute terms [24]. In the multimedia literature, more advanced methods of specifying temporal information have been described. These include duration hierarchies [4], constraint-based systems[9, 20], temporal glue [26, 21] and parallel/sequential hierarchies [1, 26, 31, 30]. A large number of models manage the complexity of the temporal specifications by employing a form of hierarchical composition.

2.3 Spatial information
Visual elements making up a presentation need to be presented on the screen, so that spatial placement information is also essential. Again, spatial information can be expressed in a number of ways. For example, with respect to a set of window coordinates where each element is given a start position, height, width, and for determining overlapping, a z-index. This can be done directly in the definition of each element, or indirectly, which allows multiple elements to reuse the same placement specification. Examples of indirect spatial positioning include CCS2 style sheets [6] and SMIL regions. More advanced methods can be used where relative spatial positioning is specified using hierarchies of regions. Examples of such methods include the use of channels in CMIF [30], nested vertical and horizontal boxes in TeX [22], and the widget hierarchies found in many GUI toolkits [23, 25].

2.4 Composition
While the temporal or spatial structures mentioned above can be used for grouping and structuring the media items in a multimedia document, in practice it is often convenient to be able to group media items independently from the temporal and spatial layout. For example, definitions of words can be grouped together in a glossary, where nothing is specified beforehand about when or where the definitions will appear; items in a video library can be grouped by theme.

2.5 Text-based document characteristics
Text-based documents historically contain structuring information on top of the underlying linear text-flow. The structuring information can describe the appearance of the document or can be based on a more abstract decomposition, such as a chapter/section hierarchy. In the latter case, the information contained in the document needs to be processed at some later point in order to display the document to the reader. This requires the layout information to be specified as part of the process. In early systems this information tended to be embedded in the formatting software, whereas more recent approaches use style sheets.

Text-based documents are intrinsically linear, and the only constraint for displaying them on a two-dimensional page or screen is that the reader should be able to reconstruct the linear ordering. For example, in Indo-European languages the reader assumes the linear flow is left to right and top to bottom, see Figure 2. As long as the layout produced by the formatting software conforms to the conventions then the details are a matter of aesthetics and the meaning of a document remains the same regardless of, e.g., the column width or the font size.

When we compare text-based document characteristics with those for multimedia, then current text-based document models tend not to address temporal issues. Spatial issues are addressed by visual markup and page description languages, such as PostScript[3], PDF [2] and style sheet languages such as CSS [6], XSL [10] and DSSSL [18]. Composition independent of the spatial layout is supported by generic markup languages such as XML [8] and SGML [17]. Current multimedia document models, however, tend not to provide support for text-flow other than within a single text media item.

3. Combinations of document characteristics
When developing a new document model that needs to support both text-flow and multimedia-specific features then the task is to ensure that all aspects are included and well integrated with one another. While this is by no means a trivial task, it is even more difficult to take an existing document model and add a new feature to it. This is because certain trade-offs have already been made to combine the set of original features included in the model, and these trade-offs may be inappropriate for the addition of a new feature. Additional problems arise when the document model has to be encoded in a document markup language. These problems are discussed in the following section. In the remainder of this section we investigate the problems of combining document characteristics in a single document model. When adding the information structure associated with a new document feature to an existing document model, then three cases can be distinguished.

In the first case, the new structure is similar to one of the existing information structures. In this

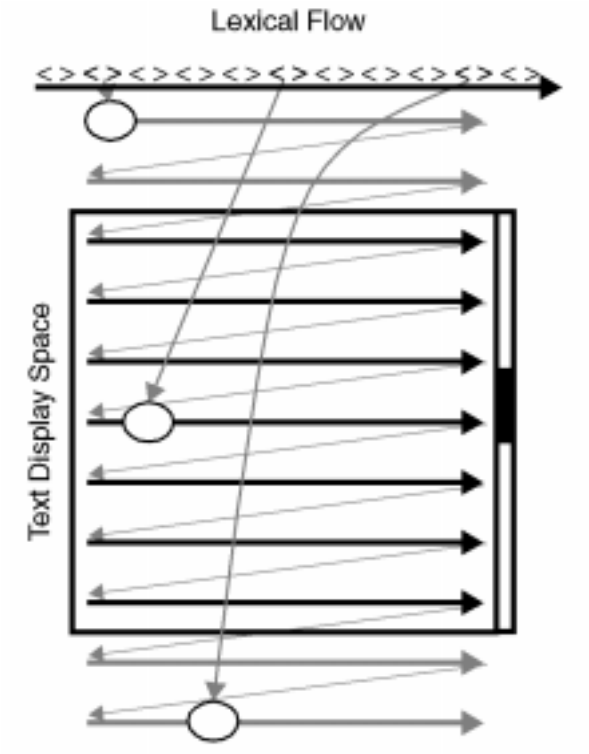


Figure 2: Lexical flow and spatial layout for text

case, the new information needed can be included within one of the already existing structures. For example, in the case of markup languages if the new structure matches the main document hierarchy, then information corresponding to the new structure can be included as attributes of the elements in the main document hierarchy.

In the second case, the new structure does not match any of the existing information structures, but the information contained in the structure can be flattened and incorporated within one of the existing structures. For example, when adding a temporal structure to a text-flow model the information can be flattened (e.g. by deriving absolute begin and end times from the temporal structure) and then added to the elements in the text-flow.

In the third case, the new structure does not match any of the existing information structures and the information contained in the structure cannot be flattened without losing essential information. In the example above, all the information about relative timing that could have been specified in a constraint-based or hierarchical temporal structure is lost. To prevent this problem, we need to fully integrate the new structure into the existing document model. For example, to truly integrate complex temporal information within text-flow based markup, we need to wholly incorporate the temporal constraints or hierarchical temporal structures with the text-flow document markup.

In this last case, there are a number of ways of combining the different structures. In the next section we discuss how document characteristics are expressed in current markup languages on the Web.

4. Expressing document characteristics in markup languages on the Web

Historically, markup languages have been the standard way of encoding text-based document models. On the Web, the standard markup language has, until now, been HTML [27]. With the emergence

of XML on the Web, markup is also becoming more and more common for non-textual information, such as SMIL for multimedia [31] and Scalable Vector Graphics (SVG) for graphics [13]. All of these markup languages use a single hierarchical structure to encode their documents. In practice, most of these markup languages use this hierarchy to encode one of the document characteristics mentioned above. The choice of characteristic normally reflects what is considered to be the most important characteristic of the document model. For instance, in SMIL the main hierarchy is temporal based and in HTML it is text-flow based. Other characteristics of the document also need to be encoded, and current languages on the Web use three methods to do this: adding new elements and attributes to the main hierarchy, adding information to the head, attaching information by using style sheets.

These other document characteristics can be expressed by specifying additional attributes and elements and embedding these within the main document hierarchy. For example, in HTML, link information is embedded within the main document hierarchy by using the a element. The advantage of this approach is that the information can directly be associated with the appropriate elements without using additional pointer mechanisms. A drawback, however, is that the various document characteristics have to share the same document structure. For example, grouping in HTML can be specified by using div or span elements, but this method does not allow hierarchical composition orthogonal to the main document hierarchy. Another drawback is that additional document characteristics can be neither maintained without modifying the document itself nor shared across multiple documents. For example, in HTML one cannot add outgoing links without modifying the document itself.

A second method is to add additional information in a secondary hierarchy outside the main document hierarchy, typically within the head section of a document. This method is currently used, for example, to add meta-data and, in SMIL, to define the spatial characteristics of the presentation. The advantages are that the additional document characteristics can be hierarchically structured, and defined orthogonally to the main document hierarchy. A disadvantage is that both structures need to refer to the same information, for example, elements in the head need to point to elements in the main document hierarchy, or vice versa. If elements already have a unique ID, XML's built-in reference mechanisms can be used (ID /IDREF), otherwise more flexible addressing mechanisms are required, such as those provided by XPointer [11].

A third method is to define additional information externally to the document. This method is currently primarily used for the specification of style and spatial layout information. Advantages are that this information can be maintained without modifying the document itself and can be shared across multiple documents. A disadvantage is the same as the previous case, in that some means is needed of referring to the elements in the document. For example, in CSS [6], this is carried out using selectors. Note that elements in the document cannot refer to information contained in the style sheet.

Existing Web languages express one or more document characteristics in the main document hierarchy and by using one or more of the three methods described above. When designing a new language, the trade-offs are made as to which characteristics are expressed using which method. In the next section we investigate the problem of adding new characteristics to existing languages.

4.1 Combining multiple structures within web-based documents

All of the multimedia and text-based document characteristics described earlier in the paper can be found in different existing markup languages on the Web, although none currently fully supports all characteristics. In Table 1 we summarize the document characteristics of a number of existing web languages and the way these are represented within the document format.

HTML's main document hierarchy represents the text-flow; temporal characteristics are not supported; spatial positioning overriding the default layout derived from the text-flow can be specified by CSS, external to the main hierarchy; composition is partially supported by allowing grouping of elements using the class attribute, and by inserting div elements within the constraints of the main hierarchy. Hierarchical composition that is orthogonal to the main document hierarchy is not supported in HTML. SMIL's main document hierarchy represents the hierarchical temporal structure;

	HTML	SMIL	SVG	HTML+TIME
Time	None	Primary Hierarchy	None	Attributes
Space (2D)	External (CSS)	Secondary hierarchy	Attributes	External (CSS)
Composition	Weak (`div/class`)	None	Primary Hierachy	Weak (`div/class`)
Text-flow	Primary hierarchy	None	External (`textflow`)	Primary hierarchy

Table 1: Characteristics of Document Languages on the Web

spatial positioning is determined using region elements defined in the head of the document, external to the main document hierarchy; composition, other than spatial and temporal, cannot be expressed; text-flow based composition is also not possible. SVG's main document hierarchy represents the composition structure of the graphical elements, which is typically independent of the spatial structure; temporal information is currently supported using attributes; spatial information is specified using element attributes. Text-flow composition is discussed in the section titled Combining document characteristics in SVG. HTML+TIME [29] is a Note to W3C that proposes an extension to HTML by adding temporal attributes to elements in the main document hierarchy. This functionality is also supported by the SMIL 2.0 Working Draft [5].

While each of these languages encodes a number of different document characteristics, there is a need to combine, for example, temporal information within text-flow and graphics based languages. Rather than producing ad hoc solutions for each combination of characteristics, we discuss a number of problems with and generic solutions for integrating multiple characteristics in a single document language.

4.2 Adding a new document characteristic to an existing document language

Above, we describe ways of combining multiple document characteristics in a document language. When adding a new document characteristic to an existing document language, integration at multiple levels has to be considered. First, the new characteristic needs to be integrated within the document model. On this level, three options - similar, flattened and full integration - are discussed in Combinations of document characteristics. Second, the new characteristic needs to be integrated on the level of the existing document language. Again there are three options: integration within the main document hierarchy, integration in the document but separate from the main hierarchy and integration external to the document, as discussed in Expressing document characteristics in markup languages on the Web. Finally, the document model and language should not be seen in isolation from the rest of the document processing environment - other models and languages, such as the rendering model, style sheet language and transport protocol, need to be considered. In this section we discuss the problems related to encoding a new document characteristic in an existing document language where the integration has already been carried out at the model level.

When integrating information within the document by adding attributes or elements, it should be clear which of these have been introduced. There may also be problems with clashes between the new and existing names. In both cases, these problems can be avoided by using an XML Namespace prefix [7]. The advantages of this approach is that it has a minimal impact on the existing document language, and allows for a certain level of backward compatibility because Web browsers usually ignore unknown attributes and elements. This approach does not guarantee full compatibility, because browsers do not ignore the contents of new elements. Another minor drawback is that the extensions further complicate the existing document language.

When integrating information within the document by encoding it separately from the main document hierarchy, a sufficiently powerful addressing mechanism is needed to point to the same information in both structures. For example, in style sheets the style rules use the selector mechanism to identify the portions of the document to which the rule applies. In SMIL, media object elements refer to their layout region using a standard XML ID /IDREF construct. In SVG, graphic elements can be reused by referring to their definitions by using links conforming to XLink [11]. A general way of

```
<html>
  <body>
    <h1>Heading A</h1>
    <ol>
      <li t:begin="3s">Point A1</li>
      <li t:begin="6s">Point A2</li>
    </ol>
  </body>
</html>
```

Figure 3: Embedding a flattened timeline within the text-flow using HTML+TIME

identifying portions of an XML document is provided by XPointer [12].

Below, we describe examples that are interesting from a multimedia perspective: adding time to HTML, and integrating document characteristics within SVG.

Adding time to HTML Adding a temporal model to HTML can be useful for the declarative specification of temporal behavior of text-flow based applications, such as synchronized slide shows, or the scheduling of short text messages to mobile phones. On the document model level, one needs to determine which of the cases in Combinations of document characteristics are appropriate for a particular application. One has to determine whether the temporal structure is similar to the already-existing text-flow structure, sufficiently simple to be flattened, or requires full integration. Given this, one needs to determine which of the three methods discussed in Expressing document characteristics in markup languages on the Web will be used to integrate the new structure at the document markup level. One can choose to add new, time-oriented elements and attributes to the main HTML document hierarchy, encode the temporal structure in the head of the HTML document, or encode the temporal structure in an external file. In the following, we discuss an example for each of these methods. The example is based on a simple application with one or two lists, each with a header and two bullet points. The timing of the elements varies in the examples.

First we discuss an example approach that uses the first method, that is adding attributes and elements to the main document hierarchy. When the temporal structure can be flattened, it can be expressed by begin and end times as attributes on existing elements. When the temporal structure is similar to the structure of the text-flow, it can be expressed by introducing new elements to embed a temporal sequential/parallel hierarchy within the text-flow hierarchy. These are the two approaches taken by HTML+TIME [29], where the temporal model is based on a timeline combined with an event model. Figure 3 shows how a simple temporal structure, which is flattened to a set of begin times, is expressed in HTML+TIME using the begin attribute. The list heading appears at time zero, the default, and is followed by the first bullet points at time three seconds and the second at time six seconds. Note the use of an XML Namespace prefix (in this case t:) that discriminates the temporal markup from the standard HTML markup. If the temporal model were more complex, for example by using the nested par and seq structures of SMIL, then the composition structure would need to be integrated in some other way. Introducing par or seqbehavior directly within the HTML document makes sense when the textflow structure is similar to the temporal structure. This is the case in Figure 4, where elements can be grouped in a single composite that functions both as the body element in the text-flow and as a par in the temporal structure.

As an alternative for integrating the temporal hierarchy within the text-flow, the temporal information can be expressed externally to the document using a style sheet. This is the approach taken by bHTML [32], a variant of HTML designed for broadcast applications. Figure 5 illustrates this by adding SMIL-based temporal style properties to CSS. The example extends the previous example by adding an extra bulleted list to the document. Note that the temporal ordering of the elements does not match their lexical ordering in the text-flow. Since CSS is only able to define properties of

```
<html>
  <body t:par="true" id="TL1" t:dur="10">
    <h1>Heading A</h1>
    <ol>
      <li t:begin="3s">Point A1</li>
      <li t:begin="6s">Point A2</li>
    </ol>
  </body>
</html>
```

Figure 4: Embedding a temporal hierarchy within the text-flow using HTML+TIME

File stylesheet.css:

```
body { synctype: par; dur: 10s}
.first { begin: 3s }
.second { begin: 6s }
```

HTML file:

```
<html>
  <head>
    <link rel="stylesheet" type="text/x-css"
      href="stylesheet.css" />
  </head>
  <body>
    <h1>Heading A</h1>
    <ol>
      <li class="first">Point A1</li>
      <li class="second">Point A2</li></ol>
    <h1>Heading B</h1>
    <ol>
      <li class="first">Point B1</li>
      <li class="second">Point B2</li></ol>
  </body>
```

Figure 5: Temporal information in style sheet

elements in the existing document structure, it cannot be used to define new hierarchical structures orthogonal to the document hierarchy.

The examples above work only where the temporal hierarchy can be flattened, or can be embedded within the existing text-flow hierarchy. In general this is not the case, so that for applications where the text-flow and temporal structures are orthogonal the approaches discussed above cannot be used. A potential solution is to describe the temporal information in a separate hierarchy and to include it in the head of the HTML document. For example, in Figure 6, the body specifies the same elements as in the previous example, with the same intended behavior, only in this case the temporal structure is fully preserved. That is, not only is the text-flow composition of the bulleted lists explicit, but also the temporal composition, which groups together the elements that are played simultaneously. The temporal hierarchy is contained within the head of the HTML document (the s: prefix is used to indicate elements from the SMIL namespace), orthogonal to the text-flow hierarchy specified in the body. Note that this is not possible in HTML+TIME, where the temporal markup is attached directly to the main document hierarchy.

Even when the temporal information has been combined within the HTML document there is no guarantee that a browser will be able to play the result. Not only the document model and document

```
<html>
  <head>
    <s:par xmlns:s="http://www.w3.org/TR/REC-smil"
      dur="10s">
      <s:par>
        <s:ref s:href="#HA" />
        <s:ref s:href="#HB" /></s:par>
      <s:par begin="3s">
        <s:ref s:href="#PA1"/>
        <s:ref s:href="#PB1"/></s:par>
      <s:par begin="6s">
        <s:ref s:href="#PA2"/>
        <s:ref s:href="#PB2"/></s:par>
    </s:par>
  </head>
  <body>
    <h1 id="HA">Heading A</h1>
    <ol>
      <li id="PA1">Point A1</li>
      <li id="PA2">Point A2</li></ol>
    <h1 id="HB">Heading B</h1>
    <ol>
      <li id="PB1">Point B1</li>
      <li id="PB2">Point B2</li></ol>
  </body>
</html>
```

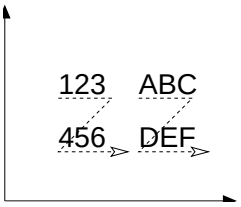
Figure 6: Combining two orthogonal hierarchies

language need to understand the semantics of time, but the processing software needs to as well. Since HTML is text-flow based, some convention has to be established as to how multiple text-flow elements are displayed together when one or more of them are added to the display or removed from it. For example, is screen space reserved for every element that will appear, so that when it is time to display it is presented in the reserved space, or is the complete text-flow redrawn each time an element is added or removed? (The latter is the solution presented in earlier work [28]).

Combining document characteristics in SVG Scalable vector graphics (SVG [13]) is currently being developed by W3C to allow the specification of vector-based graphics within Web document formats. The main document hierarchy of the language is based on general composition - that is, it provides grouping without predefined semantics. Spatial characteristics of graphic elements are typically defined using attributes on the elements themselves. We discuss the integration of two other document characteristics, namely, text-flow and time.

SVG includes text elements, which are positioned in a manner similar to graphic elements. To support selections of multiple text elements, an ordering needs to be specified. The ordering, in fact the text-flow, is orthogonal to the spatial positioning of the elements, and as such needs to be specified independently. Support for text-flow, included in an earlier working draft [19], is illustrated in Figure 7. The example defines four text elements (T1 -T4) which are positioned on the screen using x,y coordinates, as illustrated in the upper part of the figure. The text-flow order of the elements is defined by the two textflow elements - that is, T1 and T3 belong to the first textflow, T2 and T4 to the second. Note that in SVG, all elements defined within a defs element are processed as definitions, so the textflow elements themselves are not directly drawn on the screen. This example also illustrates the need for a pointing mechanism, here the text and textflow elements point to each other using links which are based on the XLink draft [11].

Combining temporal behavior with structured graphics provides animation. When a temporal




```
<defs>
  <textflow>
    <tref href="#T1"/><tref href="#T3"/>
  </textflow>
  <textflow>
    <tref href="#T2"/><tref href="#T4"/>
  </textflow>
</defs>
<textblock id="T1"><text x="100" y="100">123</text></textblock>
<textblock id="T2"><text x="100" y="200">ABC</text></textblock>
<textblock id="T3"><text x="150" y="100">456</text></textblock>
<textblock id="T4"><text x="150" y="200">DEF</text></textblock>
```

Figure 7: Text-flow composition in SVG [19]

model is applied to vector graphics there are two main issues: turning the display of vector graphics on and off over time, and the changing the visual properties of elements over time. The latter is addressed in SMIL Animation [5]. The former case, adding temporal behavior to SVG, is relatively straightforward, since the spatial layout is independent of the text-flow. In particular, the advantage is that an SVG browser will not need to re-render the text-flow depending on which text items are to be displayed at any particular moment in time, whereas this is the case in HTML. A language developer is free to choose how to combine the temporal information at both the modeling level - similar structures, flattened and full integration - and at the language level - embedding within the main hierarchy, adding a secondary hierarchy to the document or as a specification external to the document. Examples of the different approaches to adding temporal markup to SVG are similar to those given in the previous sub-section on HTML.

Note that SVG graphics are expected to be incorporated in other Web documents, including HTML, SMIL and other XML languages. This means that not only does time have to be incorporated in SVG itself, but that the language environment in which the SVG graphics are embedded needs to be able to handle time. For instance, in HTML where long documents cannot be completely displayed in the window, changes may take place over time that cannot be perceived by the reader. The browser may incorporate some automatic scrolling mechanism, but even this may not be sufficient where multiple parts of the document change simultaneously. For example, the window may currently display the middle of an HTML document, while SVG graphic objects are simultaneously being scheduled at the beginning and end of the document.

5. Summary and Conclusion

We have discussed four encompassing characteristics of multimedia documents, namely time, space, text-flow and composition structures. We discriminate three cases of integrating two or more of these within a single document model, namely integrating two similar structures, integrating structures where one can be flattened without losing essential information, and then the case of full integration of two orthogonal structures. On the Web, the resulting document model needs to be encoded in an XML document markup language. XML basically provides (embedded) hierarchical markup plus optional inline text content but is, with respect to the four multimedia characteristics, free of document

semantics. In order to be useful in a multimedia environment, some means of defining document semantics is needed. On the Web, three methods are currently in use:

- semantics are defined in the specification of a particular document language, such as HTML 4.0 and SMIL 1.0,
- semantics are defined in the specification of a language module, such as XLink and SVG, which can be used in other languages,
- semantics are defined in the specification of an external language, such as CSS, which can be used to superimpose semantics on any XML document language.

Recent developments on the Web indicate a move away from the first method to the more flexible second and third methods. For the second method, we need some means of integrating the different modules that are to be combined together in a single XML document language. The main document hierarchy of an XML document language is traditionally based on what is considered to be the most important document characteristic in the application domain. We described three methods for encoding other document characteristics within a single language, namely adding attributes and elements within the main document hierarchy, incorporating the extra information in a secondary hierarchy, for instance in the document head, or specifying the extra information externally to the document, for example in a style sheet.

Finally, it is not sufficient to integrate document characteristics on the document model and language levels, but the rest of the document processing environment also needs to be considered. For example, in whatever way time is integrated into a document model or document language, the full browser environment, including the browser's rendering model and style sheet model, needs to support scheduling and synchronization.

Acknowledgments

The authors would like to thank members of the W3C SYMM working group for the constructive discussions throughout the development of SMIL.

References

1. Philipp Ackermann. Direct Manipulation of Temporal Structures in a Multimedia Application Framework. In *Proceedings of the second ACM international conference on Multimedia '94*, pages 15–58, San Francisco, October 15 - 20, 1994.
2. Adobe Systems Incorporated. Portable Document Format (PDF). See http://www.adobe.com/products/acrobat/adobepdf.html.
3. Adobe Systems Incorporated. PostScript. See http://www.adobe.com/products/postscript/main.html.
4. James F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–844, November 1983.
5. Jeff Ayars, Dick Bulterman, Aaron Cohen, Ken Day, Erik Hodge, Philipp Hoschka, Eric Hyche, Muriel Jourdan, Kenichi Kubota, Rob Lanphier, Nabil Layaïda, Philippe Le Hégaret, Thierry Michel, Debbie Newman, Jacco van Ossenbruggen, Lloyd Rutledge, Bridie Saccocio, Patrick Schmitz, and Warner ten Kate. Synchronized Multimedia Integration Language (SMIL) 2.0 Specification. Work in progress. W3C Working Drafts are available at http://www.w3.org/TR, 21 September 2000.
6. Bert Bos, Håkon Wium Lie, Chris Lilley, and Ian Jacobs. Cascading Style Sheets, level 2 CSS2 Specification. W3C Recommendations are available at http://www.w3.org/TR, May 12, 1998.
7. Tim Bray, Dave Hollander, and Andrew Layman (Editors). Namespaces in XML. W3C Recommendations are available at http://www.w3.org/TR, Januari, 14, 1999.
8. Tim Bray, Jean 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.
9. M. Cecelia Buchanan and Polle T. Zellweger. Automatically Generating Consistent Schedules for Multimedia Documents. *Multimedia Systems*, 1(2):55–67, 1993.
10. Stephen Deach. Extensible Stylesheet Language (XSL) Specification, April, 21, 1999. W3C Working Draft. Work in progress. Available at http://www.w3.org/TR/WD-xsl/.
11. Steve DeRose, David Orchard, and Ben Trafford. XML Linking Language (XLink). Work in progress. W3C Candidate Recommendations are available at http://www.w3.org/TR, July, 3, 2000.

12. Steve DeRose and Jr. Ron Daniel. XML Pointer Language (XPointer). Work in progress. W3C Candidate Recommendations are available at http://www.w3.org/TR, June, 7, 2000.
13. Jon Ferraiolo. Scalable Vector Graphics (SVG) 1.0 Specification. Work in progress. W3C Candidate Recommendations are available at http://www.w3.org/TR, August 2, 2000.
14. L. Hardman and D. C. A. Bulterman. Document Model Issues for Hypermedia. In William I. Grosky, Ramesh Jain, and Rajiv Mehrotra, editors, *Handbook of Multimedia Information Management*, pages 39 – 68. Prentice Hall, 1997.
15. 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.
16. Lynda Hardman. *Modelling and Authoring Hypermedia Documents*. PhD thesis, University of Amsterdam, 1998. ISBN: 90-74795-93-5, also available at http://www.cwi.nl/~lynda/thesis/.
17. International Organization for Standardization. Information Processing — Text and Office Information Systems — Standard Generalized Markup Language (SGML), 1986. International Standard ISO 8879:1986.
18. 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.
19. Jon Ferraiolo et al. Scalable Vector Graphics (SVG) 1.0 Specification. Work in progress. W3C Working Drafts are available at http://www.w3.org/TR, June 25, 1999.
20. M. Jourdan, N. Layaida, C. Roisin, and L. Sabry-Ismailand L. Tardif. Madeus, an Authoring Environment for Interactive Multimedia Documents. In *Proceedings of ACM Multimedia '98*, Bristol UK, 1998.
21. M.Y. Kim and J. Song. Multimedia Documents with Elastic Time. In *Proceedings of ACM Multimedia '95*, pages 143–154, San Francisco CA, 1995.
22. Donald E. Knuth. *TeX: The Program*, volume B of *Computers and Typesetting*. Addison-Wesley Publishing Company, 1986.
23. Mark A. Linton, John M. Vlissides, and Paul R. Calder. Composing User Interfaces with InterViews. *IEEE Computer*, 22(2):8–22, 1989.
24. Macromedia. Authorware. http://www.macromedia.com/software/authorware/.
25. J.K. Ousterhout. An X11 Toolkit Based on the Tcl Language. In *USENIX*, 1991.
26. Hamakawa R. and Rekimoto J. Object composition and playback models for handling multimedia data. *Multimedia Systems*, 2:26–35, 1994.
27. Dave Raggett, Arnaud Le Hors, and Ian Jacobs. HTML 4.0 Specification. W3C Recommendations are available at http://www.w3.org/TR, April, 24, 1998.
28. Lloyd Rutledge, Jacco van Ossenbruggen, Lynda Hardman, and Dick C.A. Bulterman. Anticipating SMIL 2.0: The Developing Cooperative Infrastructure for Multimedia on the Web. In *Proceedings of The Eighth International World Wide Web Conference (WWW8)*, May 1999.
29. Patrick Schmitz, Jin Yu, and Peter 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.
30. G. van Rossum, J. Jansen, K. S. Mullender, and D.C.A. Bulterman. CMIFed: A Presentation Environment for Portable Hypermedia Documents. In *The First ACM International Conference on Multimedia*, pages 183–188, August 1993.
31. W3C. Synchronized Multimedia Integration Language (SMIL) 1.0 Specification, June 15, 1998.

Edited by Philipp Hoschka. W3C Recommendations are available at http://www.w3.org/TR.

32. Ted Wugofski. Considering Broadcast HTML. See http://www.tvbroadcast.com/archive/98.10.23.7.htm.