The LimSee3 Multimedia Authoring Model

Romain Deltour
INRIA Rhône-Alpes
655 avenue de l'Europe
38334 Saint Ismier, France
Romain.Deltour@inria.fr

Cécile Roisin
UPMF, INRIA Rhône-Alpes
655 avenue de l'Europe
38334 Saint Ismier, France
Cecile.Roisin@inria.fr

ABSTRACT

For most users, authoring multimedia documents remains a complex task. One solution to deal with this problem is to provide template-based authoring tools but with the drawback of limited functionality. In this paper we propose a document model dedicated to the creation of authoring tools using templates while keeping rich composition capabilities. It is based on a component oriented approach integrating homogeneously logical, time and spatial structures. Templates are defined as constraints on these structures.

Categories and Subject Descriptors

I.7 [**Document and Text Processing**]: Document Preparation—*Hypertext/hypermedia, Multi/mixed media, Standards*

General Terms

Design, Languages

Keywords

multimedia documents, document models, document authoring, template-based editing

1. INTRODUCTION

The LimSee3 project aims at defining a document model dedicated to adaptive and evolutive multimedia authoring tools, for different categories of authors and applications, in order to easily generate documents in standard formats. It is a follow-up to LimSee2 (a SMIL authoring tool) and is tightly related to our involvement in the Urakawa project [4].

Direct manipulation of well-established languages such as SMIL (or XMT) is too complex for most users because it requires a deep understanding of the semantics of the language (e.g. the SMIL timing model). From the user perspective a document is logically structured in several high level objects, but in SMIL these are distributed in several parts of the document (spatial information in the header, time information in the body). Several software tools [1], [2], [5] hide the intrinsic complexity of SMIL behind advanced GUI or by adding dedicated information via namespaced attributes. However, SMIL is too low level for this kind of tools to be completely satisfactory. Hence we are also looking at template-based solutions such as [3] or [6].

In this paper we propose to define a model for these templates. Our approach is to focus on the logical structure of the document while keeping some semantics of proven technologies such as SMIL. This provides better modularity, facilitates the definition of document templates, and improves manipulation and reusability of content.

DocEng'06, October 10–13, 2006, Amsterdam, The Netherlands.

2. APPROACH

Existing structured multimedia models generally put the time structure at the heart of the multimedia document. It was for instance the case of CMIF [9] and Madeus [7], as well as SMIL. However, the time dimension does not always reflect exactly the logical structure in the way it is considered by the author. This latter logical structure is made of at least both time and spatial dimensions, which are specified in SMIL in distinct sections of the document. Our approach defines the logical dimension as the master structure of the document, which is a tree of modular components that can be constrained by a dedicated template mechanism.

A template document is a kind of reusable document skeleton that provides a starting point to create document instances. Domain specific template systems are a user-friendly authoring solution but require hardly extensible dedicated transformation process to output the rendering format. We chose on the contrary to tightly integrate the template syntax in the document: the template is itself a document constrained by schema-like syntax. The continuum between both template and document permits to edit templates generically as any other document and within the same environment. It enables an evolutive authoring of document instances from templates. There is no need to define a dedicated language to adapt to each different use case.

With these objectives, we define a structured authoring language independently of any publication language. Elements of the master structure are components that represent semantically significant objects. Both time and spatial dimensions are integrated inside each component. This permits components to be authored independently, integrated in the document structure, extracted for reusability, constrained by templates or referenced by other components.

While truly modular, this component approach raises the

issues of inter-object relations and extraction of components from their context. The different components of a multimedia document are indeed often tightly related one with another: when they are synchronized, when they are aligned in space, when one contains an interactive link to the other, and so on. Our approach, which is close to the one proposed in [10] is for each component to abstract its dependencies to external components by giving them symbolic names that are used in the timing and layout sections. This abstraction layer facilitates the extraction of a component from its context, thus enhancing modularity.

One might think that this component-based approach provides less reusability of the layout definition as in SMIL, but the layout is still reusable in the scope of the component. Actually, the major drawback is that global views of the document (global timing scenario, global spatial layout) are not directly accessible but need to be computed, which might be costly. This is more an implementation issue though.

Finally, the goal was to rely on proven existing technologies, in both contexts of authoring environments and multimedia representation. The timing and positioning models are wholly taken from SMIL. Using XML provides excellent structuring properties and enables the use of many related technologies. Among them are XPath, used to provide fine-grained access to components, and XSLT, used in templates for structural transformation and content generation.

3. THE AUTHORING LANGUAGE

The authoring language is twofold: it consists in a generic document model for the representation of multimedia documents, and it defines a dedicated syntax to represent templates for these documents.

3.1 Document Model

A *document* is no more than a `document` element wrapping the root of the object hierarchy and a `head` element containing metadata. This greatly facilitates the insertion of the content of a document in a tree of objects, or the extraction of a document from a sub-tree of objects.

A *compound object* is a tree structure composed of nested objects. Each compound object is defined by the `object` element with the `type` attribute set to `compound`. It contains a `children` element that lists children objects, a `timing` element that describes its timing scenario and a `layout` element that describes its spatial layout.

The value of the required `localId` attribute uniquely identifies the component in the scope of its parent object, thereby also implicitly defining a global identifier `id` when associated with the `localId` of the ancestors. In Example 1, the first child of object `parent` has the local id `child1` and hence is globally identified as `parent.child1`.

The timing model is taken from SMIL 2.1. The `timing` element defines a SMIL time container (a `par` by default) as specified in the section "Attributes for timing integration" of the timing and synchronization module of SMIL 2.1. The timing scenario of a component is obtained by composition of the timed inclusions defined by the `timeRef` elements, whose `refId` attributes are set to local ids of children.

The positioning model is inspired from the SMIL 2.1 layout modules and relies on a similar inclusion mechanism.

```
<document xmlns="http://wam.inrialpes.fr/limsee3/"
          xmlns:smil="http://w3.org/smil/">
 <head><!-- some metadata --></head>
 <object type="compound" localId="parent">
  <children>
   <object localId="child1">...</object>
   <object localId="child2">...</object>
   <object localId="child3">...</object></children>
  <timing>
   <timeRef refId="child1" begin="0s"/>
   <smil:seq begin="1s">
    <timeRef refId="child2"/>
    <timeRef refId="child3"/></smil:seq></timing>
  <layout height="100" width="100">
   <layoutRef refId="child1" top="0"/>...
  </layout></object></document>
```

Example 1: A Partial LimSee3 Document

A *media object* is actually a simple object that wraps a media asset, i.e. an external resource (such as an image, a video, an audio track, a text...) referenced by its URI. It is defined by the `object` element with the `type` attribute set to either `text`, `image`, `audio`, `video` or `animation` (this list can be extended in the future). The URI of the wrapped media asset is the value of the `src` attribute. Example 2 shows an image media object with local id `linkImage` which wraps the media asset identified by the URI `./medias/image.jpg`.

Area objects inspired from the SMIL `area` element can be associated with media objects. They are used for instance to structure the content of a media object or to add a timed link to a media object. An area is defined as an `object` element with the `type` attribute set to `area`. For instance, in example 2 the image media object `linkImage` has a child area which defines a hyperlink.

Relations of dependency between objects are described independently of their semantics in the document. External dependencies are declared with `ref` elements grouped inside the `related` child element of objects. The value of `refId` of a `ref` element is the id of the related element and the value of `localId` is a symbolic name that is used within the object to refer to the related object. For instance, in Example 2, object `linkImage` describes an image that links to the object `extObj1`, by first declaring the relation in a `ref` element and then using this external object named `target` to set the value of the `href` attribute of the link, using `attribute` and `value-of` elements taken from XSLT.

```
<object type="img" localId="linkImage"
 src="./medias/image.jpg">
 <related>
  <ref localId="target" refId="extObj1"/>
  <ref localId="start" refId="extObj2"/></related>
 <children>
  <object type="area" localId="link"/></children>
 <timing>
  <attribute name="begin">
   <value-of refLocId="start" select="@id"/>.begin
  </attribute>
  <timeRef refId="link">
   <attribute name="href">
    #<value-of refLocId="target" select="@id"/>
   </attribute>
  </timeRef></timing>...</object>
```

Example 2: A LimSee3 Object with External Dependency Relations

3.2 Templates

Template nodes aim at guiding and constraining the edition of the document. In order to have better control and easy GUI set up, the language defines two template nodes: media zone and repeatable structure.

A *media zone* is a template node that defines a reserved place for a media object. It is represented by the `zone` element with a `type` attribute to define what types of media object can be inserted in this zone. Possible values for this attributes are `text`, `img`, `audio`, `video`, `animation`, `any`, or a list of these types. The author can also specify content that will be displayed to invite the user to edit the media zone with the `invite` element (of any media type). For instance Example 3 shows a media zone for an image, with textual invitation. During the authoring process `zone` elements aim at being replaced by media objects inserted by the user.

A *repeatable structure*, represented by the `objList` element, is a template node that defines a homogeneous list of object. Each item of the list matches a model object declared in the `model` child of the list. The cardinality of the list can be specified with the `minOccurs` and `maxOccurs` attributes. Example 3 shows a simple slideshow as an `objList` named `list` and containing image media objects as specified by the model `slide`.

It is possible to lock parts of a document with the `locked` attribute, to prevent the author from editing anything. This permits for instance to guide more strongly inexperienced users by restricting their access to the only parts of the document that make sense to them.

```
<object type="compound" localId="slideshow">
 <children>
  <objList localId="list" maxOccurs="20">
   <model name="slide">
    <zone type="img">
     <invite type="text">Add an image</invite>
    </zone></model>
   <object type="img" ...>...</object>
</objList></children>
 <timing>
  <smil:seq begin="1s">
   <for-each
 select="children/objList[@localId="list"]/object">
    <timeRef>
     <attribute name="refId">
      <value-of select="@localId"/>
     </attribute></timeRef></for-each></smil:seq>
 </timing>...</object>
```

Example 3 : A Simple Slideshow Template

4. THE AUTHORING PROCESS

The tight integration of template nodes in the document model ensures a continuous authoring workflow. As shown in Figure 1, a document is progressively instantiated from a template by providing content to template nodes (for instance, in Example 3 the object list `list` is partially instantiated) ; a template can conversely be authored starting from a document instance. Once fully authored, a document can be exported to any target format, provided the semantics of this latter is included in our document model. This authoring process enables generic or dedicated authoring tools with appropriate user-friendly GUI.

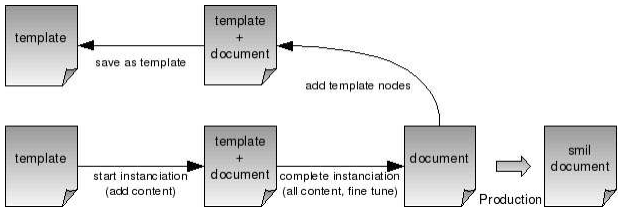

Figure 1: The Authoring Process

5. CONCLUSION

The model presented in this paper improves reusability not only with template definitions but also by the homogeneous structuring of documents. The homogeneous use of components (for instance in XPPath expressions) facilitates the extensibility of the language and the evolution of existing documents. This document model is being implemented as cross-platform java software and, once completed, new authoring tools will be quickly developed thanks to our previous experience with LimSee2. The declarative and modular approach of LimSee3 model paves the way for providing extensions such as authoring adapted documents or defining behavioral reactivity in documents as proposed by [8].

6. REFERENCES

[1] *IBM Resarch: Authoring in XMT*. http://www.research.ibm.com/mpeg4/Projects/AuthoringXMT/.

[2] *LimSee2*. http://wam.inrialpes.fr/software/limsee2/.

[3] *Microsoft Producer for PowerPoint*. http://www.microsoft.com/office/powerpoint/producer/prodinfo/.

[4] *Urakawa Project*. http://www.daisy.org/projects/urakawa/.

[5] D. Bulterman and L. Hardman. Structured multimedia authoring. *ACM TOMCCAP*, 1(1):89–109, 2005.

[6] X. Hua, Z. Wang, and S. Li. Lazycut: Content-aware template based video authoring. In *Proc. ACM Multimedia'05*, pages 792–793. ACM Press, 2005.

[7] M. Jourdan, N. Layaïda, C. Roisin, L. Sabry-Ismaïl, and L. Tardif. Madeus, an authoring environment for interactive multimedia documents. In *Proc. ACM Multimedia'98*, pages 267–272. ACM Press, 1998.

[8] P. King, P. Schmitz, and S. Thompson. Behavioral reactivity and real time programming in xml: functional programming meets smil animation. In *Proc. ACM DocEng'04*, pages 57–66. ACM Press, 2004.

[9] L.Hardman, G. van Rossum, and D. Bulterman. Structured multimedia authoring. In *Proc. ACM Multimedia'93*, pages 283–289. ACM Press, 1993.

[10] H. Silva, R. Rodrigues, L. Soares, and D. M. Saade. Ncl 2.0: integrating new concepts to xml modular languages. In *Proc. ACM DocEng'04*, pages 188–197. ACM Press, 2004.