An XML-based multimedia document processing model for content adaptation

Lionel Villard, Cécile Roisin, Nabil Layaïda

INRIA-Rhône-Alpes
{Lionel.Villard,Cecile.Roisin,Nabil.Layaida}@inrialpes.fr

Abstract. In this paper we present a general framework for document production that covers generic document model needs and adaptation needs. We define a multimedia document model called Madeus that describes multimedia scenarios. We show how this model is used for the generation of adaptable presentations. This presentation process is based on XSLT transformation techniques and constraint technologies for document formatting.

1 Introduction

Designing structured multimedia authoring systems is a great challenge for the industry which has to handle large amount of information. Several years ago, the notion of document classes was introduced for static documents (c.f SGML [6]) in order to enhance document productivity and quality. With the advent of standards like XML [22] and the increasing diversity of media types, there is also a need to have classes for multimedia documents. Typical examples of document classes are touristic guides, slideshow presentations, technical documentation (for installation and maintenance) or courseware. In addition, XML also allows the specification of the structure of document classes independent of their final presentation.

There is another feature that must be addressed when presenting multimedia documents: the adaptation of document content to the current presentation context. This operation must take into account user capabilities, user preferences, physical location, network and system resources. It also increases the complexity of editing these contextually-adaptable documents. Authoring tools must help the author to design documents that can have different renderings.

The aim of this paper is to propose a general framework for document production through the specification of a document presentation model. In our context, a document must not only be considered through its final presentation but also through a richer semantical representation defined by its class and its content. This approach is needed in order to take into account the increasing diversity of visualization and interaction devices : PDA (Personal Device Assistant), cellular phone, workstation, microphone, etc.

This paper is organized as follows: we firstly describe the general framework of our approach for multimedia document processing. We present then the multimedia document model called the *Madeus model* which aims to represent a multimedia

scenario through its different dimensions: logical, temporal, spatial, etc. In the third section we describe the process that enables our system to generate the presentation of any structured document using this document model (see figure 1). The presentation is generated through a set of transformation steps. Finally we show how this model address the contextual adaptability problem and we compare it to related work. In the last section, we suggest some perspectives for addressing the problems of authoring such documents.

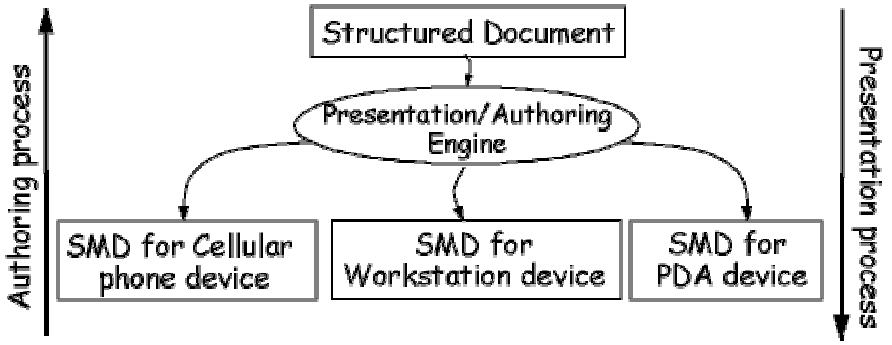

Fig 1. General presentation and authoring architecture of a structured document in the context of a SMD (Structured Multimedia Document)

The ideas presented in this paper are presented through a sample document which is a slideshow (see section 2.2). Its presentation has been obtained using our presentation engine called Kaomi.

1 Document processing model

2.1 Motivation and requirements

In order to provide a general framework for document processing, there is a need for an intermediate document representation between the higher level XML-based source formats and the lower level rendering format (execution-oriented) (see figure 2). The aim of this intermediate level is to capture all the information required for the formatting process. This intermediate representation can be compared to XSL-FO [20] or the Grif-Thot Abstract Image structure [5] [13].

Fig. 2. Different document formats

In our system, we propose a generalization of this approach to time-based multimedia documents. Our goal is not only to handle structure and spatial layout but also synchronization and interactivity. The Madeus processing model is designed in order to render any kind of document classes encoded using a XML DTD [22] or a schema [23]. We try also to cover a wide range of multimedia scenarios using heterogeneous objects such as video, sounds, text, etc.

2.2 The sample document

In this paper, the sample document that we will use is presented in figure 3. It has been written according to the DTD developed by Norman Walsh which can be found at [24].

```
<slides>
  <title>XML And Web Applications</title>
  <screen>
    <videodata fileref="http://.../film.mpg"/>
  </screen>
  <foil>
    <title>Introduction</title>
    <para>...</para>
  </foil>
  <foil>
    <title>Multimedia systems</title>
    <para>...</para>
  </foil>
</slides>
```

Fig 3. A XML source format of the slideshow

The *slides* element is the element root for a slideshow document. It contains a title element, a screen element which can be used as a background image or/and a video, and a list of foil elements. The *foil* element defines one slide. This content can be paragraphs, sounds, video, images, etc.

2.3 Architecture of the multimedia presentation system

The presentation process (see figure 4) used in our system takes as input any XML document, and produces a multimedia document. The source document belongs to a document class, which means that it can be validated against the class's DTD. The production of the multimedia document is achieved in two steps. First, the

transformation step allows the creation of multimedia objects from the source document. It allows also us to set the synchronization relationship between these objects, the layout and the hyperlinks (see next section). In the second step, the multimedia document is formatted so that the result can be directly used by the presentation engine.

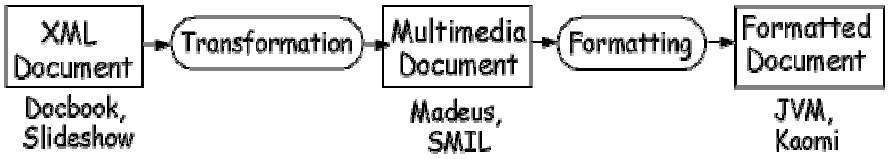

Fig. 4. Presentation process

For instance, one possible result of this entire process is the presentation of the XML source slideshow (see figure 5). On the bottom right of the screen there is the video of the show. In the top right appears the slide currently presented. The table of contents is presented on the left part of the screen. The title of slide currently presented is highlighted. The table of contents can be used to directly access a particular slide by clicking on the corresponding title.

Fig 5. A particular presentation of slideshow documents

3 Madeus document model

Numerous examples of other work exist in the area of modeling multimedia documents. SMIL 2.0[19] is an XML-based synchronization language. The media objects are hierarchically organized through sequential, parallel and exclusive

operators. In addition, the author can explicitly specify the beginning and the end of object with both absolute (date or user event) or relative offsets. This language is too rigid and so is not suitable for document adaptation. The document model used in ISIS [16] is a timed Petri-net TPN* for modeling timed user interaction. It allows flexible time specification thanks to a constraint-based approach. Only the temporal dimension is taken into account. Vazirgiannis's document model [17] is expressed through algebraic and spatio-temporal compositions of events (from user, media, systems, etc.). The resulting specification is compiled into a Java program which renders the multimedia scenario. The Madeus model presented below takes advantage of these different approaches in order to wider cover multimedia scenario needs: the XML-based structuration, constraint-based specification for temporal and spatial properties and external event specification.

In Madeus, the description of a multimedia document is organized around four dimensions : logical, temporal, spatial and hypermedia. In this section, we discuss the model for each of these dimensions and show how to combine them together. The syntax used for the multimedia document model presented here corresponds to the intermediate format introduced earlier. In our system this syntax is formally described as a XML DTD and therefore it takes full advantage of all the XML existing tools. The DTD itself can be found at [18] and in the remaining part of this section we only use fragments of document instances encoded according to this DTD.

In accordance with the idea of separating document information into dimensions, the general structure of each document instance is decomposed in four main parts:

- MediaContent and MediaUse dimensions that describe the logical structure of the document
- A Temporal dimension for synchronization between document parts
- A spatial dimension for layout

Because hypermedia information is closely related to interactivity, it is encoded in either the Temporal or MediaUse parts. A Madeus XML source document look like this:

```
<Madeus>
  <MediaContent> ... </MediaContent>
  <MediaUse> ... </MediaUse>
  <Temporal> ... </Temporal>
  <Spatial> ... </Spatial>
</Madeus>
```

Each part is detailed in following sections.

3.1 Logical model

A multimedia presentation is composed of a set of *media objects*, for instance a picture, a sound, a 3D animation, etc. In order to reuse the same content, its specification is separated from its use context. The *MediaContent* element contains raw media data, for instance, pixels of a picture, characters of a text, etc., and the intrinsic properties of the media, like the duration of a video or its size. The *MediaUse* element indicates a particular use of the content with specific style properties, for instance a line border color, a font size, etc.

The content part can also be used to refine the media description. For instance, the content of a video can be structured in sequences, scenes, shots, etc. [14]. In our sample document, this will allow synchronization between the table of contents entries and the video of the show.
The logical model allows us to hierarchically organize contents and objects. In the example of figure 6, a group element of type *C-Group* or *U-Group* just plays an aggregate role. Its semantics depend on the document type and not on its presentation. For instance, for the slideshow document, media contents can be gathered by media types and media uses can be gathered by the slide structure of the slideshow. Thanks to a simple inheritance facility that is applied on the logical structure, a group element can define default values for some attributes of its children elements (for instance the color, the character size, etc.).

```
<MediaContent> <!-- Content specification part -->
  <C-Group>
    <C-Group ID="Text" MIMEType="text/plain">
      <DefContent ID="ST1">Introduction</DefContent>
      <DefContent ID="ST2">Multimedia</DefContent>
      <C-Group ID="Video" MIMEType="video/mpg">
        <DefContent ID="Film" src="http://./Film.mpg">
          <Scene StartFrame="0" EndFrame="20">
            <Shot StartFrame="0" EndFrame="5"/>
            <Shot StartFrame="5" EndFrame="20"/>
          </Scene>
        </DefContent></C-Group></C-Group>
</MediaContent>
<MediaUse>  <!-- Objects specification part -->
  <U-Group>
    <DefUse ID="U-Film" Content="Film" BorderWidth="1"
            BorderColor="black"/>
    <U-Group ID="TOC_entries" FontColor="black">
      <DefUse ID="U-title1_toc" Content="SlideTitle1"
              FontSize="12"/>
      <DefUse ID="U-title2_toc" Content="SlideTitle2"
              FontSize="12"/>
    </U-Group>
    <U-Group ID="Slide1" FontColor="blue">
      <DefUse ID="U-title1" Content="SlideTitle1"
              FontSize="32"/>
      <U-Group ID="U-Body1"> ... </U-Group></U-Group>
    <U-Group ID="Slide2"> ... </U-Group>
  </U-Group>
</MediaUse>
```

Fig. 6. The slideshow logical model

3.2 Temporal model

The temporal model allows the organization of media objects over time. It is based on previous work on Madeus where a specific markup (a language) has been specified to

synchronize the presentation [7]. The underlying model of this language is interval-based. This means that each object has a corresponding time interval characterized by a *begin*, a *duration* and an *end* attribute. Each of these attributes has a range of values *[min, pref, max]* (from zero to *indefinite*) instead of a single value as in SMIL [19].
Every MediaUse element is associated a temporal *interval* element that carries all the temporal attributes required for its schedule. In particular the *Duration* attribute can override the intrinsic duration of the media. If this interval duration is larger than the intrinsic one, the additional attribute *Fill* allows the specification of the desired behavior: *Fill="repeat"* (the media is replayed during the interval), *Fill="freeze"* (the last image is displayed during the remaining time) or *Fill="cut"* (the media is withdrawn). The specification can state that some parts of the scenario may become active only when an external interaction is performed (for example when the user fires a hyperlink, see section 3.4 and 3.6). Such links are defined with a classical *HRef* attribute attached to the source interval. Notice that the beginning of interval for such target elements is set to the indefinite value.
For instance, the interval T-slide1_toc associated with the media slide1_toc is specified by:

```
<Interval ID="T-slide1_toc" Object="slide1_toc"
          Duration="min:50s pref:60s max:300s"
          Fill="freeze" HRef="@T-slide1.begin"/>
```

In the Madeus language, the synchronization is specified both by composite nodes and temporal relations. A composite node (*T-Group* element) is used to temporally group interval elements. A *during* constraint is set between each child and its parent. This basic synchronization can be refined with temporal relations between the descendants of a composite node. For instance, specifying that two slides play in sequence can be done by placing the *meets* relation between them.
There are two ways to declare the intervals that are involved in a temporal relation: either by their explicit name (the *ID* attribute) or by an implicit name defined by the relative position of the intervals: *prev*, *all* and *next*. This last facility is needed because in some cases, the interval ID is not known beforehand since new elements are produced during the transformation process (see section 4). In some other situations this relative naming simplifies the description, for example when a temporal relation applies on all the children elements of a group. For instance, the specification of a sequence of slides can be achieved like this :

```
<T-Group>
  <Interval ID="T-slide1" Duration="pref:20s max:25s"/>
  <Interval ID="T-slide2" Duration="pref:20s max:25s"/>
  <T-Relations>
    <T-Relation Name="Meets" Intervals="all"/>
  </T-Relations>
</T-Group>
```

The following specification represents the part of a possible temporal scenario of the slideshow example. When the document starts, the table of contents, the first slide and the first frame of the video are displayed. From line 7 to 13, slides are presented in sequence separated in preference by 20 seconds. In order to show a particular slide,

temporal links are added on each slide title entry in the table of contents (see section 3.4).

```
1.<Temporal>
2.  <T-Group ID="T-Root">
3.    <T-Group ID="T-TOC" Duration="pref:indefinite">
4.      <Interval ID="T-slide1_toc"
                  Object="U-title1_toc" Fill="freeze"
                  HRef="@T-slide1.begin"/>
5.      <Interval ID="T-slide2_toc"
                  Object="U-title2_toc" Fill="freeze"
                  HRef="@T-slide2.begin"/>
6.    </T-Group>
7.    <T-Group ID="T-Slideshow" Speed="pause">
8.      <T-Group ID="T-slide1"
              Duration="pref:20s max:25s">...</T-Group>
9.      <T-Group ID="T-slide2" Duration="pref:20s
                max:25s"> ... </T-Group>
10.     <T-Relations>
11.       <T-Relation Name="Meets" Intervals="all"/>
12.     </T-Relations>
13.   </T-Group>
14.   <Interval ID="T-Film" Object="Film"
               Speed="pause"/>
15.   <T-Relations>
16.     <T-Relation Name="Starts" Interval1="T-Root"
                    Interval2="T-TOC"/>
17.     <T-Relation Name="Starts" Interval1="TOC"
                    Interval2="T-Slideshow"/>
18.     <T-Relation Name="Equals" Interval1="Slideshow"
                    Interval2="T-Film"/>
19.   </T-Relations>
20.  </T-Group>
21.</Temporal>
```

3.3 Spatial model

The spatial model is basically similar to the temporal model. The main differences are the use of a spatial vocabulary (left_align, bottom_spacing, etc.) and the extension to support two dimensions unlike the temporal language which has a single dimension. In addition, a spatial attribute cannot have indefinite value. More precisely, the spatial model organizes the document space as a 2D box hierarchy. A composite node (*S-Group* element) allows the grouping of a set of 2D shapes (*Shape* element) inside 2D boxes as illustrated below.

```
1.<Spatial>
2.  <S-Group ID="SpatialRoot">
3.    <S-Group ID="S-TOC">
4.      <Shape ID="S-slide1_toc" Object="U-title1_toc"
               Left="pref:10px" Top="pref:20px"/>
5.      <Shape ID="S-slide2_toc"
               Object="U-title2_toc"/>
```

```
6.       <S-Relations>
7.         <S-Relation Name="Left_align"
     Shape1="S-slide1_toc" Shape2="S-slide2_toc"/>
8.         <S-Relation Name="bottom_spacing"
     Distance="min:5px max:15px" Shapes="all"/>
9.       </S-Relations>
10.    </S-Group>
11.   ...
12. </S-Group>
13.</Spatial>
```

3.4 Hypermedia model

The goal of this model is to describe links between elements or different parts of elements. The basis of this model is the XLink standard [21] enhanced with temporal and spatial behavior. The supported properties of hypermedia links are the following:

- **Display behavior** When activated, the target element can either be displayed in a new frame, replace the source element or be embedded into it.
- **Target location** This property indicates the target location of the link. Currently, this value is a URI-reference. But in a multimedia context, it can be extended to reference a temporal and a spatial location. Temporal locations can be absolute (a date) or relative to the beginning of a scenario part (a timestamp, a beginning of a T-Group element or an Interval element, etc.). Likewise spatial locations can be absolute or relative to the position of an element.
- **Activation type** This property defines how the link must be activated. It can be specified through the following attributes: *HRefDur* (the period of time when the element can be activated), *HRefNumber* (the number of allowed activations) and *HRefActivation* (the link can be activated automatically, or interactively using differents interactive sources, see section 3.6).

For instance, the previously defined table of contents can be completed by the following temporal links:

```
...
<T-Group ID="T-TOC">
  <Interval ID="T-slide1_toc" HRef="@T-slide1.begin"
            HRefNumber="indefinite"
          HRefActivation="OnPointingCursorActivation"/>
  <Interval ID="T-slide2_toc" HRef="@T-slide2.begin"
            HRefNumber="indefinite"
          HRefActivation="OnPointingCursorActivation"/>
</T-Group>
...
```

3.5 Links between dimensions

In the previous sections, we have described the different dimensions of the multimedia documents without considering interactions between them. If examined

two by two, twelve combinations may exist between these dimensions. Here are some meaningful examples of such combinations :

- **Style over time** This combination is used to specify what is called *style animation*. A style animation is a discrete or continuous modification of a style attribute during a time interval. For example, changing the color from black to white over a two seconds period is specified by:

```
<AnimateMotion ID="ColorAnim" Attribute="TextColor"
               Values="from:black to:white"/>
```

- **Spatio-temporal** This combination is used to specify *spatial animations* like motion, zoom effects, such as proposed in animation section of SVG [4]. In our model, we extend animation specification with the ability to enforce a spatial relation during a given time interval:

```
<S-Relation Name="Left_align" Shapes="all"
            Interval="T-IntroInterval"/>
```

- **Link over time** This combination is used to specify hypermedia properties that change over time, like activation, target location, display behavior, etc
- **Spatial-Link** This combination is used to specify that the target location of a link depends on a given spatial location of an element.

3.6 Abstract devices for interactivity

Interval-based models are well adapted for storytelling documents in which no interactivity is needed. We propose to extend this model with interactivity while keeping schedule characteristics for predictive parts of the scenario.
In the section 3.2, we defined a temporal model in which the beginning of an interval is indefinite. Activating such an interval can only be done interactively. The consequence of runtime activation of an interval is the dynamic calculation of the beginning value of target intervals. Therefore, the scenario must be formatted dynamically since not all the values are known at the beginning of the presentation (c.f. section 5).
In order to render the document correctly to the user, the devices used as the source of interactivity must be specified. Given the diversity of input devices and our desire to cover different types of devices, an abstraction of these input devices is required. Each system has a device pointer, such as a mouse device for workstation, a pen device for PDA, a tactile screen for interactive kiosks. In our model, we define several abstract devices such as the *PointerCursor* that allows the specification of that the user can cross *over* an element, can *activate* it, etc.
For instance, animating the color of a table of contents entry when the user crosses over the entry can be done as follows:

```
<MediaUse>
  <DefUse ID="slide1_toc" TextColor="Black">
    <AnimateMotion ID="ColorAnim" Attribute="TextColor"
                   Values="from:black to:white"/>
  </DefUse>
</MediaUse>
```

```
<Temporal>
  <Interval Object="slide1_toc">
    <Interval HRefNumber="indefinite"
              HRefActivation="OnPointerCursorOver"
              Object="slide1_toc.ColorAnim"/>
  </Interval>
  ...
</Temporal>
```

4 Transformation

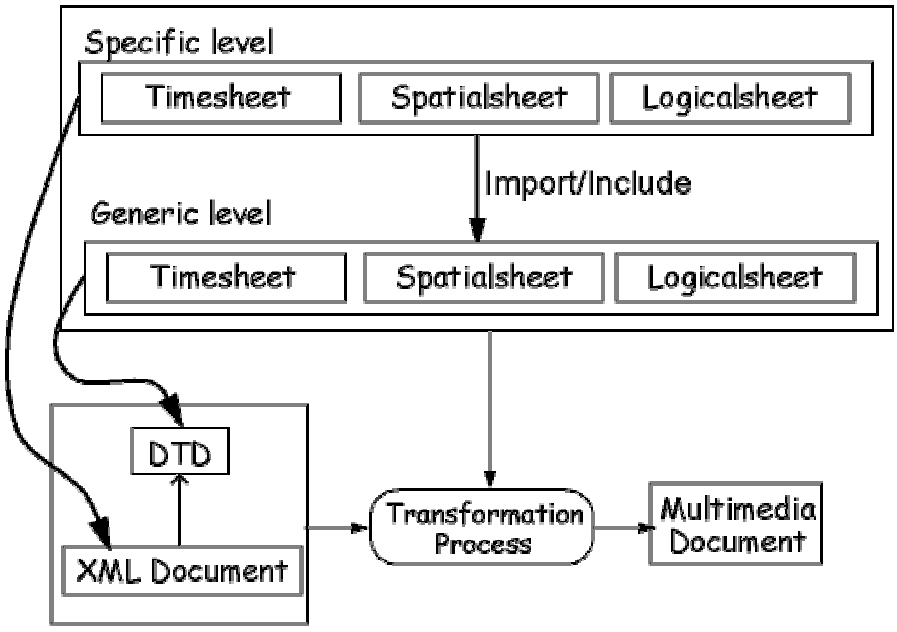

Fig. 7. Multimedia presentation with transformation sheets

The transformation process enables the creation of a multimedia presentation from both the source document and one or many presentation sheets that specify their temporal, spatial and navigation behavior (see the bottom part of figure 7). Moreover, this process can be used in order to generate table of contents, index, numbering, etc.
Several transformation languages exist. Balise [1] is a script language in which some functions of tree manipulation (creation and copy) are provided. Omnimark [10] is a streaming programming language. It consists of rules that define data events such as general markup events produced when parsing a XML document. XSLT [3] [9] is a semi-declarative language designed especially for XML document transformation. It consists of transformation rules (templates) associated with patterns. When a rule

pattern matches in the source, the corresponding rule is instantiated to create the result tree.
As the transformation power of these three languages are quite similar, we have chosen to develop our transformation system with XSLT. This allows us to take advantage of the ongoing work on this standard. Moreover, XSLT allows us to structure and combine transformation rules as a set of modules.
As we want to benefit from generic specification (as provided by source XML DTD), we have identified two levels of transformation (see figure 7): the generic level and the specific level. The former allows the transformation of any valid XML document that conforms to the DTD (or schema) for which the transformation has been specified. The latter allows the specification of transformation behaviors only for a particular document (or instance). Moreover, it's possible to define a transformation for each dimension of a multimedia document.
In our slideshow example, we have defined several presentation sheets in order to generate the presentation illustrated in figure 5. We give below some excerpts of these sheets :

```
<!-- Root sheet -->
<xsl:stylesheet>
  <xsl:include href="genlogicalsheet.xsl"/>
  <xsl:include href="genspatialsheet.xsl"/>
  <xsl:include href="gentemporalsheet.xsl"/>
  <xsl:include href="speclogicalsheet.xsl"/>
  <xsl:include href="specspatialsheet.xsl"/>
  <xsl:include href="spectemporalsheet.xsl"/>
</xsl:stylesheet>

<!-- Fragment of logical generic sheet -->
<xsl:template match="foil/title"
              mode="content.title.toc">
<!-- {position()} is used in order to have
     unique identifier -->
  <madeus:DefContent ID="SlideTitle{position()}">
    <!-- Gets the title content -->
    <xsl:value-of select="."/>
  </madeus:DefContent>
</xsl:template>
<xsl:template match="foil/title" mode="use.title.toc">
  <madeus:DefUse ID="U-title{position()}_toc"
      Content="SlideTitle{position()}" FontSize="12"/>
</xsl:template>

<!-- Fragment of temporal generic sheet -->
<xsl:template match="slides" mode="temporal">
  <madeus:Temporal>
    <madeus:T-Group ID="T-Root">
      <T-Group ID="T_TOC" Duration="pref:indefinite">
        <xsl:apply-templates select="slide"
                     mode="temporal.title.toc"/>
      </T-Group>
      ...
    </madeus:T-Group>
```

```
  </madeus:Temporal>
</xsl:template>
<xsl:template match="foil/title"
              mode="temporal.title.toc">
  <madeus:Interval ID="T-slide{position()}_toc"
                   Object="U-title{position()}_toc"
                   HRef="@T-slide{position}.begin"/>
</xsl:template>
```

Notice that this process allows the definition of several presentation for the same source document. For instance, with another set of presentation sheets, the slideshow document can be presented just as a sequence of slides, without the table of contents, neither the video. It can also be presented as a table of thumbnails.

5 Formatting

The result of the transformation process is a high level specification of a multimedia scenario. In order to execute it, the formatting process calculates attribute values that will be directly used by the execution engine. The underlying techniques used for formatting heavily depends on the target application. We can identify four levels of such applications (in increasing complexity order) :

- Final multimedia presentation without timing constraint (such as an interactive kiosk).
- Final multimedia presentation with timing constraints (web access). The rendering must be able to adapt to variation delays of media access.
- Adaptable rendering (see section 6).
- Presentation view in an authoring context.

We have developed our formatting process using our Kaomi presentation engine [7]. It takes as input a Madeus specification and either produces a direct rendering (using JMF library for playing dynamic media) or a portable format such as SMIL (with a potential loss of information).

In order to provide high level adaptability features with reliable formatting results, our formatting system relies extensively on constraint technology. Moreover as the Madeus model aims at maintaining relative specifications until the formatting step (through relations), it is possible to compute presentation parameters (scheduling, placement) at the earliest possible moment.

In all cases, the formatting process relies on the following steps:

- **Unit resolution** The Madeus model allows the specification of attribute values using many units. These ones are converted into canonical units (pixels, milliseconds, etc.).
- **Consistency checking** As the Madeus model relies on constraint specification, the system is able to verify that every multimedia document produced by the transformation process is consistent (i.e there exists at least one possible execution). This is done through algorithms on the resulting constraint network managed by Kaomi [8].

- **Search for a solution** Basically this is the step that produces a specific presentation solution among those resulting from the Madeus specification. The presentation values are calculated using constraint technology [8]. The search of a solution can be oriented by context parameters (see next section).

6 Contextual adaptability

Contextual adaptability is the capability for a document processing system to take into account the following parameters:

- **User profile** such as his language, his skill level, his physical deficiencies, his physical location, etc.
- **Hardware profile** such as screen size, CPU type, speakers presence, modem speed, etc.

With the emergence of new supports and devices, this function becomes more and more important, and is currently investigating by several research teams [2] [15].

Adaptation parameters can act in different places: inside a transformation sheet, in separate transformation sheets or as an input of the formatting process. A given parameter can be used at different levels of the process. For instance, a spatial-style sheet can define spatial properties for devices with similar screen sizes, while little differences in screen size can be supported by the formatting step thanks to the constraint approach. Indeed in the first case, the structure of the resulting document may be very different and so the transformation process is required. In the second case, the spatial layout can be adapted thanks to the ability to specify attributes by ranges of values. For instance, the specification of the spacing between text entries in the table of contents can be an interval from 5 to 15 pixels.

In addition to these two adaptation modes, we have identified another one which can be considered as an intermediate adaptation step (see arrow number 2 of figure 8). this adaptation step aims at providing transformations that are independent from document classes. For instance, to take into account the blindness of the user, we need a rule which transforms all text into speech sounds. Notice that this rule can be applied to the multimedia document instead of to the source one without changing the multimedia structure. More generally, this step, called decoration, can be used to add or remove style or media attributes, for instance to adapt colors to colour-blind persons. The advantage of this step relies on its simplicity compare to transformation sheets. However, it requires both flexible specification in the document model and a constraint-based formatting process to allow the final adaptation during the scheduling: for instance the media decoration of a document from text to sound is possible only if the temporal structure of document is specified through relations and range of durations.

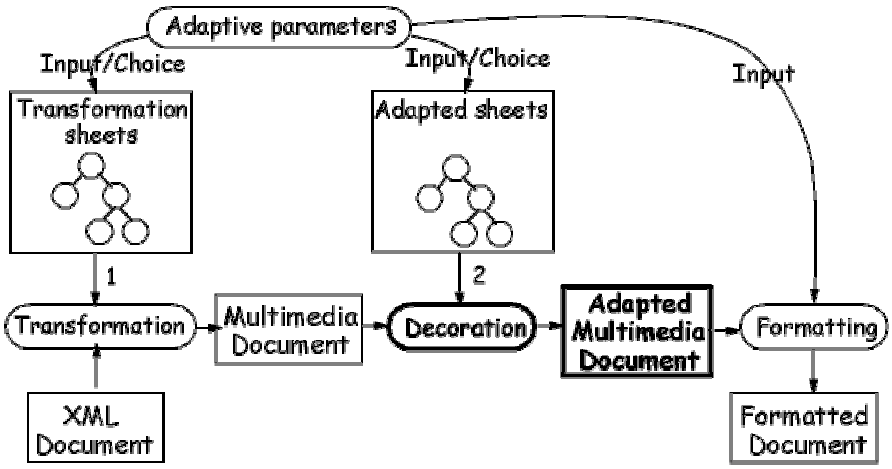

Fig 8. Adaptive Presentation process

To enable the system to choose among several transformation and adaptation sheets, the sheets must be labeled with meta-data about that record the adaptive parameters. The transformation and decoration processes use these parameters to compare them to the effective context parameters and then produce the final adapted multimedia document. This is the subject of a W3C note called CC/PP [12]. This note propose a method for using RDF, the Resource Description Format of the W3C [11], to create a framework for describing user preferences and device capabilities.

7 Perspectives and conclusion

This paper has proposed a general process for presenting generic and adaptable multimedia documents. The system suggests to split the presentation generation in three layer. The first layer us allows to encode the content independently from its presentation using an XML DTD. A first transformation step allows us to obtain a new representation which reflects all the dimensions needed for presentation. This step allows us to adapt the content to the presentation device and to user preferences. A second simple transformation (a decoration) is then applied in order to adapt the media to the end user. Finally, the formatting process produces a representation playable by the presentation engine. The application has been implemented in a Java prototype built on top of the Kaomi multimedia toolkit. It can take as input any XML document thanks to Xalan, a Java implementation of a XSLT processor. This prototype has been experimented for the slideshow document class and allows the production of slideshow presentations on a workstation screen and a PDA.
We are currently investigating new adaptation techniques. In particular, we are looking for a stronger integration with a content negotiation layer. We are also designing a network adaptation layer which allows us to manage the quality of service

and the resource allocation between the different parallel streams. We think our architecture can be a good framework for such extensions.
Another area of investigation is related to authoring such documents. Indeed both genericity and adaptation requirements increase the complexity of editing. We can identify two levels of difficulties:

- **interface level** perception of the temporal dimension, multiple level of specification (from source document, presentation sheets and adaptation sheets) and multiple target presentations.
- **internal data level** complexity of the data management in a tree transformation process.

We promote the idea of providing several edition modes inside an integrated tool. Kaomi is also an editing toolkit which proposes high level editing functions for multimedia documents through multiple views. We are on the process to extend this toolkit in order to take into account the different levels of edition: source document, presentation sheets and adaptation sheets.

References

1. Balise, Balise 4, http://www.us.balise.com/products/balise/index.htm , 2000.
2. Susanne Boll, Wolfgang Klas, and Jochen Wanden, A Cross-Media Adaptation Strategy for Multimedia Presentation, Proceedings of the ACM Multimedia'99, October 30 - November 5, 1999, Orlando, Florida, USA, 1999.
3. James Clark, XSL Transformation, http://www.w3.org/TR/xslt , 1999.
4. Jon Ferraiolo, Scalable Vector Graphics (SVG) 1.0 Specification, http://www.w3.org/TR/SVG, 2000.
5. R. Furuta, V. Quint, and J. André, Interactively Editing Structured Documents, Electronic Publishing -- Origination, Dissemination and Design, vol. 1, num. 1, pp. 19-44, April 1988.
6. International Standard ISO 8879, Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML), International Standard Organization, 1996.
7. Muriel Jourdan, Cécile Roisin, and Laurent Tardif, A Scalable Toolkit for Designing Multimedia Authoring Environments, Special number, 'Multimedia Authoring and Presentation: Strategies, Tools, and Experiences' of Multimedia Tools and Applications Journal, Kluwer Academic Publishers, 1999.
8. Muriel Jourdan, Cécile Roisin, and Laurent Tardif, Constraints Techniques for Authoring Multimedia Documents, Constraints Journal, Kluwer Academic Publishers, to appear.
9. Michael Kay, XSLT Programmer's Reference, Wrox Press, 2000.
10. Omnimark, Guide to OmniMark 5, http://www.omnimark.com/develop/om5/doc/ , 2000.
11. Ora Lassila and Ralph R. Swick, Resource Description Framework (RDF) Model and Syntax Specification, http://www.w3.org/TR/REC-rdf-syntax/, 1999.
12. Franklin Reynolds, Johan Hjelm, and Spencer Dawkins, Composite Capability/Preference Profiles (CC/PP): A user side framework for content negotiation, http://www.w3.org/TR/NOTE-CCPP/, 1999.
13. Cécile Roisin and Irène Vatton, Merging Logical and Physical Structures in Documents, Electronic Publishing -- Origination, Dissemination and Design, special issue Proceedings of the Fifth International Conference on Electronic Publishing, Document Manipulation and Typography, EP94, vol. 6, num. 4 , pp. 327-337, April 1994.

14. Cécile Roisin, Tien Tran_Thuong, and Lionel Villard, Integration of structured video in a multimedia authoring system, Proc. of the Eurographics Multimedia'99 Workshop, Springer Computer Science, pp.133-142, septembre 1999.
15. Lloyd Rutledge, Lynda Hardman, Jacco van Ossenbruggen, and Dick C.A. Bulterman, Mix'n'Match : Exchangeable Modules of Hypermedia Style, Proceedings of the ACM Hypertext '99, 1999.
16. Junehwa Song, G. Ramalingam, and Raymond E. Miller, Modeling Timed User interactions in Multimedia Documents, IEEE International Conference on Multimedia Computing and Systems, Hiroshima, pp.407-416, June 1996.
17. Michalis Vazirgiannis, Interactive Multimedia Documents, Lecture Notes in Computer Science n°1564, Springer, 1999.
18. Lionel Villard, Madeus model DTD, http://www.inrialpes.fr/opera/madeusmodel.dtd, 2000.
19. W3C, SMIL Boston Specification, http://www.w3.org/TR/smil-boston, 2000.
20. W3C, Extensible Stylesheet Language (XSL), http://www.w3.org/TR/xsl, 2000.
21. W3C, XML Link Language (XLink), http://www.w3.org/TR/xlink, 2000.
22. W3C, Extensible Markup Language (XML) 1.0, http://www.w3.org/TR/REC-xml, 2000.
23. W3C, XML Schema Part 0 : Primer, http://www.w3.org/TR/xmlschema-0, 2000.
24. Norman Walsh, Slides doctype, http://nwalsh.com/slides/index.html, 2000.