IMPROVING THE EXPRESSIVENESS OF XML-BASED HYPERMEDIA AUTHORING LANGUAGES

M.J. ANTONACCI, D.C. MUCHALUAT-SAADE, R.F. RODRIGUES, L.F.G. SOARES
{mjuliana, debora, rogerio}@telemidia.puc-rio.br; lfgs@inf.puc-rio.br

Laboratório TeleMídia, Depto. de Informática, PUC-Rio
R. Marquês de São Vicente, 225 – 22453-900 – Rio de Janeiro, Brasil

This paper presents some desirable requirements for hypermedia authoring declarative languages, discussing which ones are satisfied by existent languages, such as SMIL, and which facilities are not offered. It also presents how the missing requirements could be offered in an XML-based language, in order to improve its expressiveness, presenting elements and attributes that could be incorporated in its DTD. Among the facilities, we can highlight the possibility for reusing document components and their presentation characteristics, offering flexibility in temporal duration specifications, adapting a document presentation according to user navigation, and specifying n-ary relationships expressing causality or constraint among components.

1 Introduction

The increase of the Web utilization for tasks that it was not originally prepared to support, specially the need to incorporate multimedia synchronized information into documents, has encouraged the raising of hypermedia authoring languages that offer more expressiveness than HTML.

The new proposed languages follow different approaches. Some of them are HTML extensions trying to maintain the HTML simplicity. This is the case of HTML+TIME [1] and the proposal of [2]. Although these proposals support temporal relationship definition, they inherit some of HTML limitations. For example, links are still defined in a content-embedded way, preventing content reuse without link inheritance.

Other proposals, as HyTime [3], are based on the SGML standard [4], whose processing complexity makes it difficult to use. In the last years, the need for a proposal specified in some meta-language simpler to deal with than SGML was recognized. The XML standard [5], a subset of SGML adopted as a recommendation by the W3C, appeared to fulfill the need of a new meta-language. Among XML-based languages for hypermedia document authoring, SMIL [6], also published as a W3C recommendation, is the most widespread. However, SMIL, as the aforementioned HTML extensions, solves some of HTML limitations but still have others, as discussed in [7].

The limited expressiveness of all mentioned languages comes mainly from the limited expressiveness of the models on which they are based. The main goal of this paper is to present how to increase language expressiveness introducing some

concepts found in the hypermedia conceptual model called NCM (*Nested Context Model*) [8, 9].

Through the use of nested compositions that can be specialized into other classes, NCM can support several types of relations not supported in other models. Moreover, new relations can be easily added when needed [8]. Among other facilities, NCM allows logical structuring of documents, it offers the possibility of defining constraint and causal temporal relationships among its components, it allows the specification of temporal and spatial behavior of documents in a flexible way, and it offers support for cooperative work.

The concepts found in the NCM model will be presented through XML elements and attributes that could be incorporated to an existent XML-based language, such as SMIL. On the other hand, they could also be used in the definition of a new declarative language for hypermedia document authoring. Both possibilities have already been implemented. An extension of SMIL, called X-SMIL [10], and a new language, called NCL – Nested Context Language [11] have been developed, validating the concepts that will be presented. Incorporating these concepts into an authoring language allows it to satisfy some desirable requirements, among which we can highlight the facility for reusing hypermedia document components and component presentation characteristics, specifying flexible temporal duration, adapting a document presentation according to user navigation, and specifying n-ary relationships expressing causality or constraint among components.

This paper is organized as follows. Section 2 discusses some desirable characteristics of hypermedia document authoring languages. Section 3 presents and compares related work according to the requirements discussed in Section 2. Still in Section 3, requirements that are not offered by the compared languages are highlighted. Section 4 presents XML elements and attributes that a declarative hypermedia language should have to satisfy the missing requirements. Those modules could be added to existent languages or incorporated into new ones, as also discussed in Section 4. Finally, Section 5 is reserved to conclusions and future work.

2 Desirable Characteristics in Hypermedia Authoring Declarative Languages

There are several important requirements a declarative language should satisfy.

R1: Document logical structuring

An essential requirement is to allow a structured specification of documents, offering a form of grouping components into blocks that can be used to build the specification. Usually, this facility is provided by the definition of compositions that group document components, which can be other compositions, recursively.

Often, authoring is facilitated if we can use compositions that have some embedded presentation semantics, for example, compositions whose components will be presented in parallel or in sequence during document exhibition. However, it is also desirable to offer compositions that do not have associated presentation semantics, but only offer a form of contextually structuring a document, as a book is structured into chapters, which are structured into sections and so on.

R2: Representation of media objects

Another important requirement is associated to media object representation. It is necessary that authoring languages offer support at least to basic media types, such as text, audio, image and video. This differentiation facilitates the definition of specific anchors to each type.

Another desirable characteristic is to allow the definition of a media object to be independent from its media content[1]. This facility permits the same content to be used by different objects, avoiding its replication in the case of its reuse in different documents or in different parts of the same document.

R3: Presentation characteristic specification separated from a component definition

A very desirable characteristic in a hypermedia authoring language is the specification of component[2] presentation characteristics in a separated object. This allows specifying different presentations for the same component or associating a single set of presentation characteristics to different components.

Another desirable characteristic is to allow a composition presentation not only by presenting the content of its components but also by presenting its structure of components and their relationships. In order to offer that, an authoring language needs to define mechanisms that will be used by an author to specify if a composition presentation refers to the presentation of its constituents or of its structure.

Another interesting requirement is the possibility of defining presentation alternatives for the same component to be chosen during run time. The choice can be done depending on the navigation made to reach that component or depending on quality of service parameters specified for a certain presentation platform.

R4: Reuse

A fundamental requirement, and perhaps one of the most important, is to allow the reuse of components in other documents or in other parts of the same document, besides the reuse of their media content, as previously mentioned,. This facilitates

[1] A media object has other attributes besides its content, such as title, author, description, anchor list, etc.

[2] From now on, a component is any media object or composition contained in a document.

the authoring process, once it avoids information redefinition. However, even more desirable is to reuse components not through copies, but through object references, avoiding that changes in an object specification have to be done more than once, and thus making the maintenance process simpler.

Besides the reuse of components, it is also desirable to allow the reuse of their presentation characteristics, since they are defined separately.

R5: Definition of links in a composition domain

Another interesting facility is to allow links associated to a certain component to be confined to a composition domain. This facility, added to the possibility of reusing components, allows the same component to have different associated links depending on the nesting of compositions where it is contained. For example, in Figure 1, component *A* inside composition *Comp1* has two associated links, while the same component inside composition *Comp2* has just one associated link.

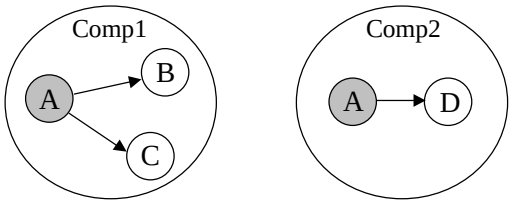

Figure 1 – Example of reuse of component *A* with different associated links

R6: Complex relationships among components

In a hypermedia document, it is not enough to provide support for the specification of traditional hypermedia relations that present a component when an anchor is selected. An important characteristic is the possibility of creating spatio-temporal relationships connecting different parts of a document or different documents. Furthermore, sometimes it is necessary to define relations that involve more than two components and that have causal or constraint semantics.

In causal relationships, some action is executed when a specific condition is satisfied. An example is "if component *A* is being presented (condition) and component *B* has finished its presentation (condition), present component *C* (action)."

In some other cases, relationships represent constraints among components. For example, a constraint that specifies that two components must finish their presentation together. Notice that there is no causality involved. The occurrence of the presentation of one component without the occurrence of the presentation of the other also satisfies the constraint, that only specifies that, if and only if these two components are presented, their end times have to coincide. Another example, now of spatial constraint, would be to define a relationship of spatial alignment between

the exhibition areas of two components, specifying that their windows must be aligned at the top when both are being presented. In this case, if one window is moved, the other must follow it, respecting the constraint relation.

No matter if the relation has a causal or constraint semantics, it can be specified using a composition or a link. An example of composition to represent temporal relationships is the sequential composition, already mentioned, that specifies that all its components must be presented in a given sequence. Relationships specified using compositions make the authoring task easier, since the author can specify just once, what would be alternatively specified by defining several links.

Another desirable characteristic in authoring languages is to allow the definition of links among components contained in different compositions C_1 and C_2. Of course, in order to satisfy *R5*, the link must be contained in a composition that contains C_1 and C_2. This gives more flexibility to change the behavior of internal components inside a composition, without the need to modify the document composition structure itself.

It is also desirable to offer a fine granularity, allowing relationships to anchor on component internal points and not only on their ends. This makes it possible, for example, to begin an object presentation from any point in its content.

R7: Flexibility in temporal specifications

Another desirable requirement is the possibility of specifying the temporal behavior of components in a flexible way. This allows the definition of elastic cost functions [12, 13] that will serve as metrics that a formatter can use to choose the best possible presentation quality. These function domains define temporal intervals that are associated to media object duration, to time segments, etc. The flexibility will allow maintaining document temporal consistency, that is, satisfying all spatial and temporal requirements specified by an author, even if it is not possible to use ideal values (minimal cost values) inside the intervals. The flexibility may also allow adjustments to be made in order to correct the presentation when unexpected events occur, as for example, communication and operating system delays, or user actions.

R8: Adaptation to a presentation platform

Finally, another desirable characteristic is the possibility of adapting a document to the platform where it will be presented. That adaptation can be made during run time, for example, by selecting components according to platform description parameters. These parameters can specify user profile information, as the native language, or even quality of service information, such as the communication system available bandwidth.

3 Related Work

This section evaluates some hypermedia authoring declarative languages concerning the requirements presented in Section 2. The languages were chosen because they represent significant examples that have extended HTML, such as HTML+TIME [1], or because they are examples of languages specified in XML, following a hypermedia conceptual model of their own, as the declarative language of the Madeus hypermedia system [14], the SMIL language [6] and its extension SMIL Boston[3] [15].

R1: Document logical structuring

In HTML+TIME and SMIL, *seq* and *par* compositions can be defined, meaning that all components are going to be presented in series or in parallel, respectively. In SMIL Boston, *excl* compositions can also be defined, meaning that only one child component may be presented at a time. However, none of the three languages offers compositions without associated presentation semantics.

Madeus[4] offers the concept of composite objects that can group basic and composite objects, allowing a hierarchical organization of the document. However, the specification of temporal and spatial synchronization relationships can only be made among components directly contained in the same composite object. This obliges the author to make the document structure according to its presentation specification.

R2: Representation of media objects

SMIL, HTML+TIME and Madeus offer support for the definition of objects of basic media types and also others that represent more specific types of media.In all these languages, the content of a media object is separated from the hypermedia document itself, and it is specified through a URI.

R3: Presentation characteristic specification separated from a component definition

HTML+TIME inherits from HTML its way of specifying spatial presentation characteristics of a component. These characteristics can be specified both next to the component definition, when it is inserted in an HTML document , or separately using an independent style sheet [16].

SMIL has its own language for defining the spatial layout of elements or it can use another, such as CSS. It is possible to specify presentation areas where

[3] As SMIL Boston is a SMIL extension, SMIL characteristics are also valid for SMIL Boston, but the opposite is not true. Thus, when this section refers to SMIL, it means both.

[4] In this paper, Madeus refers to the declarative language of the hypermedia authoring and formatting system called Madeus [14].

document visual objects will be presented. It is also possible to specify a top-level presentation window, which must be unique in SMIL, but not in SMIL Boston.

Although HTML+TIME and SMIL allow the definition of the spatial presentation characteristics of a component in a separated object, the temporal presentation characteristics, such as the beginning, duration and end of presentation, are embedded in the object definition. In Madeus, all spatial and temporal presentation characteristics of a component are always embedded in its definition.

R4: Reuse

Allowing the definition of a media object to be separated from its content facilitates content reuse in the same or different documents. This is the basic reusing form provided by SMIL, HTML+TIME and Madeus.

SMIL and HTML+TIME also allow the reuse of spatial presentation through region definitions and style sheets, respectively. However, document component specifications are reused only through copies, what makes maintenance difficult.

Madeus provides, besides content reuse, the facility of structure reuse. Composite objects can be reused in another document or in another part of the same document. However, [14] does not make clear if the reuse is made by copies of the composite object definition or simply by references to a previous definition.

R5: Definition of links in a composition domain

In HTML+TIME and SMIL, temporal relationships are specified using compositions and the only type of link that is offered is the traditional hypermedia link, which is not defined in a composition domain.

In SMIL Boston, temporal relationships can also be defined by uni-directional single-headed links that can be traversed automatically, but they are not defined in a composition domain either.

In Madeus, temporal and spatial constraints between components are defined in a composition domain, offering the possibility of reusing the same component with different associated links. However, traditional hypermedia links are not confined to a composition domain and can be defined between components contained anywhere.

R6: Complex relationships among components

Temporal relationship specifications are very similar in HTML+TIME and SMIL, since HTML+TIME was proposed based on the temporal concepts of SMIL. SMIL temporal specifications are based on parallel and sequential compositions or else on traditional hypermedia links, in the case of user interaction. Links always associate one source anchor to one target anchor and are triggered by user selection. The source and target anchors can represent an element as a whole or a spatial or temporal region of a media object. In SMIL Boston, links can also be traversed automatically.

Madeus does not offer compositions defining specific temporal presentation semantics. It allows specifying temporal and spatial behavior of components through the definition of constraints. Authoring is very simple because the language offers a set of high level operators representing several types of relations, including constraint and causal relations. However, in spite of its simplicity, not all types of desirable relations can be specified. For example, it is not possible to specify a causal relation that contains more than one condition based on different types of events (such as the presentation of a component and the selection of an anchor). Besides that, relationships in Madeus can only be defined between objects directly contained in the same composition, which restricts the way documents are structured.

All the four languages allow the definition of relative temporal relationships between components, that is, the temporal positioning of a component in relation to another one and not only in an absolute timeline. In Madeus, it is also allowed to define relative spatial relationships, which facilitates authoring even more.

R7: Flexibility in temporal specifications

SMIL and HTML+TIME do not offer the possibility of specifying flexibility for component duration. Madeus allows a flexible media object duration specification defining an interval, through its minimum and maximum values, for the object presentation. However, there is no concept of cost associated to the presentation quality to guide the choice of a value inside this interval.

R8: Adaptation to a presentation platform

In SMIL and HTML+TIME, alternative compositions (*switch*) can be defined in a document. In that kind of composition, only one constituent will be selected during document presentation according to platform characteristics. Madeus does not offer such facility.

3.1 What is still missing?

Some of the desirable characteristics mentioned in Section 2 are not offered by SMIL, HTML+TIME and Madeus. They are summarized in what follows.

R1: Document logical structuring – none of the languages offers a composite object without any associated presentation semantics.

R3: Presentation characteristic specification separated from a component definition – none of the languages allows the temporal behavior specification of an object to be defined separately from the object specification. Furthermore, none of them allows an object to be associated to different presentation specifications (spatial and temporal) depending on the navigation done to reach the object, or even depending on where the object is included in the structure of compositions. This means that if an object is included in more than one composition, each one of its

occurrence cannot be associated to a different presentation specification. Moreover, allowing an object presentation specification to be chosen according to QoS parameters defined on a presentation platform is also missing. And at last, allowing the internal structure of a composition to be presented in addition to the presentation of its component contents is neither offered by any of the languages.

R4: Reuse – a clear description of how to reuse object specifications by referencing where they were previously defined needs to be offered.

R5: Definition of links in a composition domain – none of the languages allows the definition of links among objects recursively contained in a composition. Thus, none of them permits an author, when including a composition inside another composition, to define links touching components included in the internal composition. This facility would allow reusing a composition structure and adding a different behavior to its constituents. For example, suppose a composition C_1 representing a book that is given to an expert reader, and suppose a composition C_2 representing the same book given to a beginner. Book C_2 can contain the same book C_1 and some more references (links among C_1's components) that would help a beginner to understand its content.

R6: Complex relationships among components – none of the languages offer the possibility of defining constraint or causal relationships among more than two objects, for example, a link with more than two end points. Furthermore, none of them permits the definition of a link that causes an attribute of an object to be changed or even a link that is triggered by changing an attribute of an object.

R7: Flexibility in temporal specifications – none of the languages permits the specification of metrics that a formatter could use to choose the best temporal adjustment in case the ideal temporal behavior cannot be accomplished.

4 How to provide what is missing?

This section presents XML elements and attributes that a declarative hypermedia language should have to satisfy the missing requirements. These modules could be added to existent languages or incorporated into new ones. An extension of SMIL, called X-SMIL, was implemented incorporating the elements into the SMIL DTD, and a new language, called NCL – Nested Context Language, was created, as will be briefly discussed in this section and in Section 5.

Usually, a hypermedia document specified in a declarative language is represented by a root element composed by two other elements, called *head* and *body*, as illustrated in Figure 2. When the presentation specification is separated from the document content, the head should contain information related to the presentation of the components, and the body should contain the definition of the components and their relations.

```
<root-element>
  <head>
     <!--   ...spatial layout and
               temporal behavior definition -->
  </head>
  <body>
     <!--   ...media object, composition and
               relationship specification -->
  </body>
</root-element>
```

Figure 2 – Hypermedia document structure

The presentation specification should define component spatial layouts and also temporal behaviors.

The spatial layout, represented by the *layout* element in Figure 3, defines windows and areas where components will be presented. Different from other previously mentioned languages, a specific window, represented by a *root-layout* element, can be associated to each one of the presentation output devices. It is also possible to define regions inside the windows, represented by *region* elements, that specify a component presentation area. These regions should be defined apart from the window, allowing the association of the same region to different windows.

Each region has an identification (*id*) and a *title*. Besides that, it should specify attributes such as *width*, *height*, *left* and *top*, which define the width and height of the area and the (*x*,*y*) coordinate of the left upper corner of where it will be placed inside its associated window, identified by its *root-layout* attribute. The attribute values may be absolute, defined in number of pixels, or relative to the window size.

```
<root-element>
  <head>
    <layout>
      <root-layout id="monitor" width="640" height="400" device="monitor"/>
      <root-layout id="tv" width="860" height="600" device="tv"/>
      <region id="north" left="0" top="0" width="100%" height="20%" root-layout="monitor"/>
      <region id="east" left="20%" top="20%" width="80%" height="60%"/>
      <region id="west" left="0" top="20%" width="20%" height="60%"/>
    </layout>
    <descriptors-set>
      <descriptor id="d1" region="west" root-layout="monitor"/>
      <descriptor id="d2" region="north"/>
      <descriptor id="d3" region="east" root-layout="monitor"/>
      <descriptor id="d4" repetitions="5" dur="9 10 11 3 3" root-layout="monitor"/>
      <descriptor id="d5" repetitions="10" dur="10" root-layout="tv"/>
      <descriptor id="d6" region="north" root-layout="tv"/>
      <descriptor id="d7" left="0" top="20%" width="100%" height="80%" root-layout="tv"/>
    </descriptors-set>
  </head>
  <body>
    <!-- ... -->
  </body>
</root-element>
```

Figure 3 – Example of a presentation specification definition

Figure 3 illustrates the definition of two *root-layout* elements. The first one defines a presentation window with width=640 and height=400. The *device* attribute

associates the window to an output device that should be recognized by the document formatter, which is a computer monitor, in this example. The second *root-layout* element defines a window associated to another device, identified by "tv". In the figure, the "north" region defines an area, placed at the (0,0) coordinate, with width=100% and height=20% relative to the presentation window, which corresponds to the one defined by root-layout "monitor."

The temporal behavior of components should be specified in an element apart from the component itself. This element is called a *descriptor*. Each component can be associated to one or more descriptors, allowing the specification of different presentation behaviors for the same component, satisfying requirement R3.

Among descriptor attributes, the following should be highlighted:

- *dur, begin, end* – instead of specifying fixed values for those attributes, the language should offer flexibility for their specification in order to satisfy requirement R7. They should be specified by cost functions relative to the duration, begin and end of the component presentation. For example, assume that a video has a duration of 20s, and a tolerance of more or less 10%, meaning that it could be accelerated or delayed 2s without hindering its comprehension. Assume also that the presentation quality loss was proportional to the deviation from its ideal duration, so that one could define a linear cost function for specifying this video duration. This function could determine minimum (18s), ideal (20s) and maximum (22s) values for its presentation, and two coefficients (c_1 and c_2), respectively indicating the proportion of loosing quality (increasing the cost) if the video is shrunk or stretched. Therefore, the cost of presenting the video with the ideal duration would be 0, with the minimum duration would be $2c_1$ and with the maximum duration would be $2c_2$.
- *repetitions* – specifies the number of times a component presentation should be repeated.
- *root-layout* – references a *root-layout* element already defined in the *layout* element.
- *region* – specifies a component presentation area or references a *region* element already defined in the *layout* element.

All descriptors should be defined in an element, which is called *descriptors-set*. Returning to Figure 3, there are seven descriptors defined. They will be associated to document components during document presentation. Some of them ("d1", "d3" and "d6") define, using their *root-layout* and *region* attributes, a window and an area for presenting the component inside this window. These attribute values correspond to the ids of the associated *root-layout* and *region* elements. Another descriptor ("d2") only defines an area, since it has already specified the window to which the area will be associated. Descriptors "d4" and "d5" refer to a presentation window (*root-layout*), defining the number of times a component should be repeated and also specifying a cost function relative to the component duration (*dur*). For descriptor "d4", a linear cost function was specified with minimum (9s), ideal (10s) and maximum (11s) values, and coefficients that indicate the cost

increase when the duration moves away from the ideal value. For descriptor “d5”, a simple cost function was specified with an ideal value (10s). Descriptor “d7” also refers to a presentation window, but it specifies an area for presenting the component inside this window, defining *left*, *top*, *width* and *height* attributes. Attributes defined in a *descriptor* element must overwrite the homonym attributes defined in a *region* element, as for example the *root-layout* attribute defined in descriptor “d6”, illustrated in Figure 3.

Document components can be simple (media objects) or composite (compositions). Media objects usually have an *src* attribute which specifies the URI of its content file. Figure 4 illustrates three text elements, one image and one audio. Each element specifies an identification attribute (*id*), that it will be defined later, a *uid* attribute, that specifies the component URI, an *src* attribute, that defines the content URI and a *descriptor-list* attribute. The last one contains a list of descriptor ids. These descriptors should have been previously defined in the document header and can be associated to the component in order to specify its presentation characteristics. The list allows the same component to be presented in different ways according to quality of service parameters. It is up to the formatter to choose the descriptor, based on platform specification. In Figure 4, the audio element can be associated to descriptors “d4” or “d5”, which were defined in Figure 3.

```
<root-element>
  <head>
     <!-- ... -->
  </head>
  <body>
     <!-- ... -->
      <text id=“monitor_title”
       src=“./telemidia:introduction.mn_title.txt” descriptor-list=“d2”/>
      <text-ref id=“tv_title” uid=“./puc/logo#title”/>
      <text id=“introduction_menu”
       src=“./telemidia:introduction.menu.txt” descriptor-list=“d1”>
        <text-anchor id=“projects_menu” text=“projects” position=“10”/>
        <text-anchor id=“team_menu” text=“team” position=“22”/>
      </text>
      <img id=“introduction_image”
       src=“./telemidia:introduction.image.gif” descriptor-list=“d3”>
        <img-anchor id=“projects_img” left=“0” top=“10” width=“30” height=“25”/>
        <img-anchor id=“team_img” left=“0” top=“35” width=“30” height=“25”/>
      </img>
      <audio id=“introduction_audio”
       src=“./telemidia:introduction.audio.aiff” descriptor-list=“d4 d5”/>
     <!-- ... -->
  </body>
</root-element>
```

Figure 4 – Example of the definition of specific media objects and anchors

It should be possible to define specific temporal and spatial anchors for each type of media object. A text-anchor determines a character sequence (*text* attribute) that is found in the position specified by a number of characters starting from the text beginning (*position* attribute). For example, in Figure 4, text “introduction_menu” contains two anchors. An audio anchor can be defined by two audio samples (beginning and end) specified in an audio file. In an image

component, a spatial anchor, represented by *img-anchor*, can be defined by a rectangular area, specified by its left upper corner, its height and its width. In Figure 4, image "introduction_image" contains two anchors. Anchors will be used in the specification of relationships among components. For continuous media, anchors may also be specified by temporal positions relative to the start of the media component.

A composition can contain media objects and other compositions[5], as well as relationships among them. In order to satisfy requirement R1, besides *par* and *seq* elements, a structuring composition should be offered, and it is represented by a *context* element. As mentioned, structuring compositions allow grouping document components without any associated presentation semantics.

Figure 5 illustrates a *context* identified by "telemidia" that contains two parallel compositions and another structuring composition. The parallel composition "team" contains a nested sequential composition.

```
<root-element>
  <head>
    <!-- ... -->
  </head>
  <body>
    <context id="telemidia">
      <par id="introduction">
        <!-- ... -->
      </par>
      <context id="projects">
        <text id="projects_description"
         src="./telemidia:projects.description.txt" descriptor-list="d1 d3">
        <audio id="projects_audio"
         src="./telemidia:projects.audio.aiff" descriptor-list="d4 d5"/>
      </context>
      <par id="team" descriptor-list="d3">
        <text id="team_description"
         src="./telemidia:team.description.txt" descriptor-list="d3">
        <img id="team_image"
         src="./telemidia:team.image.gif" descriptor-list=d1">
        <seq id="team_seq" >
          <video id="team_video1"
           src="./telemidia:team.video1.mov" descriptor-list="d7">
          <video id="team_video2"
           src="./telemidia:team.video2.mov" descriptor-list="d7">
        </seq>
      </par>
      <!-- ... -->
    </context>
  </body>
</root-element>
```

Figure 5 – Example of the definition of *context* compositions

It should be possible to reuse, in the same document or in a different one, any element representing a media object or a composition. In order to allow this and satisfy requirement R4, components (media objects and compositions) must specify a required attribute:

[5] The only constraint is that it cannot be nested in itself.

- *id* – useful for identifying a component occurrence inside a document. Its value must be unique for components explicitly defined in the document.

When a component is being reused, its tag should have its original name followed by "*-ref*". For example, if a *text* component is reused, *text-ref* should be used, and similarly if a *seq* component is reused, *seq-ref* should be used. Reused components have two required attributes:

- *id* – as defined before.
- *uid* – specifies the URI that identifies the component. If the component is an entire document, its value should be the document's URI. If the component is defined inside another document, its value should be that document's URI followed by the component's *id* attribute inside that document. If the component is defined inside the same document, its value should be the *id* attribute declared in the definition of the component.

For example, suppose document Doc_1 defines a *text* element T_1 with *id*="text1". If T_1 is reused inside Doc_1, the *text-ref* element should have another *id*, say "text2", and *uid*="text1". If T_1 is reused in a different document Doc_2, the *text-ref* element should have a unique *id* inside Doc_2, say "my_text", and *uid*=URI(Doc_1)#id(T_1), where URI(Doc_1) is the URI of document Doc_1 and *id*(T_1) is the *id* of T_1 inside Doc_1, which is "text1".

A reused component may define additional anchors, such as a *text-anchor* if it is a *text-ref* or an *img-achor* if it is an *img-ref*.

Hypermedia links should allow representing complex relationships among two or more document components, according to requirement R6. In order to provide it, a link is composed by a source end point set, a target end point set and a meeting point. Source and target end points define events in components that constitute link sources and targets. Each end point should specify the *id*, *node-list*, *anchor*, *event*, *attr-name* and *descriptor-list* attributes. The *id* attribute has the usual meaning. The *node-list* attribute specifies a list of values of nested component's *id* that should be followed to find the component that acts as link source or target (the last *id* in the *node-list*, called endpoint component). The *anchor* attribute specifies the anchor *id* contained in the endpoint component. The *event* attribute defines an event associated to the anchor. Its possible values are *presentation*, *selection* or *attribution*. As in NCM [9], an event can be the presentation or the selection of a certain anchor or even the attribution of a value to a component's attribute. If it is an attribution event, *attr-name* should identify the name of the corresponding attribute. The *descriptor-list* attribute has the same meaning of the descriptor-list attribute of a component. In the case of a target end point, it optionally specifies a descriptor list from which one descriptor is chosen to be associated to the endpoint component, when someone navigates through the link. Therefore, this specification optionally bypasses the descriptor-list specification of a component.

In order to satisfy requirement R5, a link should be confined to a composition domain, thus it must be defined inside a composite component and its endpoint components must be recursively contained in this composition.

Figure 6 illustrates the definition of a link (*link* element) named “link1”, defined in composition “telemidia”. The link contains two source end points (*source-ep*), one corresponding to the selection of anchor “projects_menu” of component “introduction_menu” and other corresponding to the selection of anchor “projects_img” of component “introduction_image”. This link also contains two target end points (*target-ep*) corresponding to the presentation of the whole content of the components “projects_description” and “projects_audio.”

```
<root-element>
  <head>
     <!-- ... -->
  </head>
  <body>
    <context id="telemidia">
      <!-- ... -->
      <link id="link1">
        <source-ep id="sep_projects_menu" node-list="introduction introduction_menu"
         anchor="projects_menu" event="selection" descriptor-list="d1"/>
        <source-ep id="sep_projects_img" node-list="introduction introduction_image"
         anchor="projects_img" event="selection" descriptor-list="d3"/>
        <target-ep id="tep_projects_description" node-list="projects projects_description"
         event="presentation" descriptor-list="d3"/>
        <target-ep id="tep_projects_audio" node-list="projects projects_audio"
          event="presentation" descriptor-list="d5"/>
        <meeting-point>
          <condition>
            <compound-condition>
              <simple-condition id="sc_projects_menu" previous-SEP="sep_projects_menu"/>
              <simple-condition id="sc_projects_img" previous-SEP="sep_projects_img"/>
              <expression-condition exp="sc_projects_menu OR sc_projects_img"/>
            </compound-condition>
          </condition>
          <action>
            <compound-action>
              <simple-action id="sa_projects_description" TEP="tep_projects_description"
                action-name="start"/>
              <simple-action id="sa_projects_audio" TEP="tep_projects_audio"
                action-name="start"/>
              <expression-action exp="PAR(sa_projects_description, sa_projects_audio)"/>
            </compound-action>
          </action>
        </meeting-point>
      </link>
     <!-- ... -->
    </context>
  </body>
</root-element>
```

Figure 6 – Example of the definition of an n-ary link

A link meeting point (*meeting-point* element) defines a set of conditions (*condition*) and a set of actions (*action*). Conditions are related to source end points and actions are operations that should be executed over target end points. That definition of meeting point represents a causal relationship among the end points of the link. In case links represent a constraint relationship, only conditions must be defined.

The *condition* element represents conditions that evaluate boolean values. They can be simple (*simple-condition*) or compound (*compound-condition*). A compound condition is constituted by one or more simple conditions and a logical expression

relating them (*expression_condition*), through operators AND, OR and NOT. A compound condition is satisfied if its expression is true.

Returning to Figure 6, the *expression-condition* defined in "link1" uses the OR operator, and it will be true if one of the two selection events specified by the link source end points happens.

The *action* element represents actions that should be executed in the target end points of a causal link. They can be simple (*simple-action*) or compound (*compound-action*).

Simple actions can be applied to end points that define presentation events (*prepare*, *start*, *stop*, *pause*, *resume*, *abort*) or to end points that define attribution events (*relative-assign*, *absolute-assign*, *enable*, *disable*, *activate*). Those values are defined through the *action-name* attribute of the *simple-action* element. The semantics of each one of these values can be found in [9].

Every compound action (*compound-action*) is constituted by one or more simple actions and one expression relating these actions (*expression_action*) through PAR and SEQ operators. These operators specify the execution order of each simple action, indicating if they will start in parallel or respecting some predefined order.

Before finishing this section, it is worth to mention that it has presented only some of the main elements that should be offered by a declarative hypermedia language in order to provide the missing requirements according to Section 3.1. The aforementioned elements can be added to an existent language, extending it, in order to increase its expressiveness. Notice that when doing so, probably, there will be some redundant elements in the extended language. This happens because original elements offered by the source language have to be kept in order to maintain compatibility. An extension of the SMIL language, called X-SMIL, was created by adding these elements to the SMIL 1.0 DTD. Details of the X-SMIL specification can be found in [10].

Another possibility is to create a new declarative language that satisfies all requirements presented in Section 2. This language will certainly offer the elements previously presented and will also have some elements similar to the ones offered by existent languages. However, redundancies will not be needed in this new language. They can only happen in order to make the authoring process easier. A new declarative language called NCL – Nested Context Language – was also created using all the facilities presented in this paper. More details about the NCL specification can be found in [11].

5 Conclusions and Future Work

This work presented some ideas to improve the expressiveness of hypermedia declarative languages in order to satisfy some requirements identified as important for the specification of hypermedia documents.

Among the provided facilities, we can highlight:

- document structuring through the use of compositions with or without associated presentation semantics;
- separation of the spatial and temporal behavior specification from the definition of document components;
- reuse of components and presentation characteristics;
- flexibility in the temporal specification and possibility of adapting a document presentation according to a presentation platform and depending on user navigation;
- possibility of specifying n-ary relationships expressing causality or constraint among components.

Following the W3C's current proposals, these facilities were implemented using XML elements and attributes that could be incorporated into existent languages and also included in the specification of a new hypermedia declarative language. Both possibilities have already been implemented. An extension of SMIL, called X-SMIL, was implemented incorporating the elements into the SMIL DTD, and a new language, called NCL – Nested Context Language, was created.

The NCL language has been used not only for declarative authoring, but also as an interchange format forhypermedia documents in the HyperProp system. HyperProp is a hypermedia system based on the NCM model and currently implemented in Java. In order to permit exchanging NCM objects using NCL, some converters were implemented. They are capable of transforming an NCL document in a Java representation of NCM objects, and vice-versa. The NCL→HyperProp converter allows using the HyperProp formatter [8] for presenting NCL documents and the HyperProp→NCL converter provides the use of HyperProp graphical authoring tools [8] as an alternative for authoring NCL documents.

An important characteristic for the success of a language is its capability to offer expressiveness maintaining simplicity. Although, whenever possible, simplicity has been a goal in the definition of NCL, the main focus of that work was to reach a maximum in flexibility and expressiveness. In this sense, a future work is to identify points where redundant elements could be added to the language in order to facilitate authoring. A point of possible improvement is the inclusion of high level mechanisms for specifying relationships among components, such as the temporal constraints offered by Madeus. Another important facility, also present in Madeus, and not present in NCL yet, is the possibility of defining spatial constraints among components.

Finally, another work in progress is the modularization of the NCL DTD and the improvement in its extensibility allowing that, for example, new types of media objects, new ways of expressing relationships and defining cost functions be incorporated to the language without the need to change the existent document specifications.

References

1. *"Timed Interactive Multimedia Extensions for HTML (HTML+TIME)". W3C Note, September 1998. Available in http://www.w3.org/TR/NOTE-HTMLplusTIME*
2. *Rousseau, F.; Duda. A. "Synchronized Multimedia for the WWW". Proceedings of the 7th. International World-Wide Web Conference, Brisbane, April 1998.*
3. *"Hypermedia/Time-Based Structuring Language (HyTime)". ISO/IEC 10744, August 1997. http://www.ornl.gov/sgml/wg8/document/n1920/html/n1920.html*
4. *"Standard Generalized Markup Language (SGML)". ISO 8879; October 1986.*
5. *"Extensible Markup Language (XML) 1.0". W3C Recommendation, February 1998. Available in http://www.w3.org/TR/REC-xml*
6. *"Synchronized Multimedia Integration Language (SMIL) 1.0 Specification". W3C Recommendation, June 1998. Avail. in http://www.w3.org/TR/REC-smil.*
7. *Rodrigues, L.M.; Rodrigues, R.F.; Muchaluat-Saade, D.C.; Soares, L.F.G. "Improving SMIL Documents with NCM Facilities", Proceedings of the Multimedia Modeling Conference'99, Canada, October 1999.*
8. *Soares, L.F.G.; Rodrigues, R.; Muchaluat-Saade, D.C. "Modeling, Authoring and Formatting Hypermedia Documents in the HyperProp System". ACM Multimedia Systems Journal, vol. 8; No. 2. March 2000, pp. 118-134.*
9. *Soares, L.F.G. "Nested Context Model Version 2.3". Technical Report of the TeleMídia Laboratory, Departamento de Informática da PUC-Rio, Rio de Janeiro, 2000.*
10. *Antonacci, M.J.; Soares, L.F.G. "X-SMIL Specification" (in Portuguese). Technical Report of the TeleMídia Laboratory, Departamento de Informática da PUC-Rio, Rio de Janeiro, 2000.*
11. *Antonacci, M.J.; Soares, L.F.G. "NCL Specification" (in Portuguese). Technical Report of the TeleMídia Laboratory, Departamento de Informática da PUC-Rio, Rio de Janeiro, 2000.*
12. *Kim, M.; Song, J. "Multimedia Documents with Elastic Time", Proceedings of ACM Multimedia'95, San Francisco, USA, November 1995.*
13. *Bachelet, B., Mahey, P., Rodrigues, R., Soares, L.F.G. "Elastic Time Computation for Hypermedia Documents". Brazilian Symposium on Multimedia and Hypermedia Systems, Natal, Brazil, June 2000.*
14. *Jourdan, M.; Layaïda, N.; Roisin, C.; Sabry-Ismail, L.; Tardif, L. "Madeus, an Authoring Environment for Interactive Multimedia Documents", Proceedings of the ACM Multimedia Conference 98, pp. 267-272, England, September 1998.*
15. *"Synchronized Multimedia Integration Language (SMIL) Boston Specification" — W3C Working Draft. February 2000. Available in http://www.w3.org/TR/2000/WD-smil-boston-20000225.*
16. *"Cascading Style Sheets, level 2 (CSS2) Specification". W3C Recommendation, May 1998. Available in http://www.w3.org/TR/REC-CSS2*