5
Módulo XTemplate

Como discutido no Capítulo 2, o conceito de estilos arquiteturais foi trazido para o domínio hipermídia, introduzindo os chamados templates de composição hipermídia (Muchaluat-Saade, 2002b; Muchaluat-Saade, 2002c). Através de um template de composição, estruturas genéricas de documentos hipermídia, contendo nós e elos, podem ser definidas de forma independente do documento final, permitindo a reutilização dessas estruturas por vários documentos distintos.

Um template de composição especifica tipos de componentes, tipos de relações, componentes e relacionamentos que uma composição possui ou pode possuir; sem identificar quem são todos os componentes e relacionamentos propriamente ditos. A identificação é responsabilidade das composições que usam o template.

XTemplate é uma linguagem XML para a definição de templates de composição hipermídia. A definição completa da linguagem XTemplate usando XML Schema (W3C, 2001b) está disponível no Apêndice C.

A definição de um template de composição é similar à definição de em estilo arquitetural em ADLs, adicionando ao vocabulário, de tipos e pontos de interface, e às restrições sobre o vocabulário, a possibilidade de definir instâncias específicas de componentes e conectores, o que é bastante útil em documentos hipermídia. A definição de um template de composição é feita em duas partes, onde somente a primeira é obrigatória:

1. vocabulário, que define tipos de componentes, tipos de conectores e pontos de interface;
2. conjunto de restrições sobre elementos do vocabulário e inclusão de instâncias de componentes e conectores, representando relacionamentos entre componentes.

Através do uso de templates de composição, composições com semânticas diferentes podem ser criadas e os autores de documentos não precisam ficar

limitados a um conjunto pré-definido de composições oferecido por uma linguagem de autoria, como já discutido na Seção 2.5. Conceitualmente, um template de composição pode especificar qualquer tipo de relacionamento entre seus componentes, entretanto, na proposta corrente, será dada ênfase aos relacionamentos espaço-temporais, que podem ser especificados usando a linguagem XConnector, apresentada no capítulo anterior.

Para ilustrar como um template de composição é criado, suponha uma composição representando a sincronização temporal entre um objeto de áudio, representando uma música ou uma palestra, com as legendas correspondentes a cada trecho do áudio (letra da música ou texto da palestra) e ainda o logotipo da empresa que detém os direitos autorais. A Figura 16 ilustra a visão estrutural e a visão temporal desse exemplo. A relação do tipo *L* especifica que "o início de um evento de apresentação causa o início de outro evento de apresentação". A relação *L* é usada para criar os relacionamentos entre o áudio (*audio*) e o logotipo (*logo*) (elo L_1), e também entre cada trecho do áudio ($track_i$) e a legenda ($subtitle_i$) correspondente (elos $L_2 \ldots L_{n+1}$). A relação do tipo *P* especifica que "o fim de um evento de apresentação causa o fim de outro evento de apresentação". A relação *P* é usada para criar os relacionamentos entre o áudio e o logotipo (elo P_1), e também entre cada trecho do áudio e a legenda correspondente (elos $P_2 \ldots P_{n+1}$).

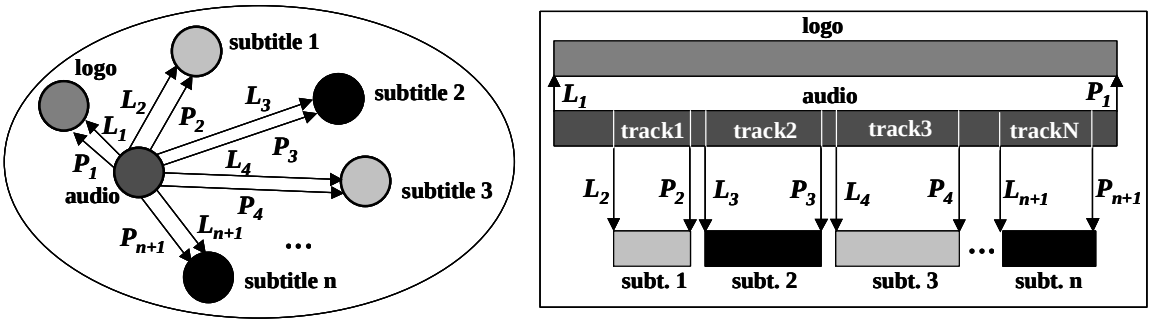

Figura 16. Visões estrutural e temporal de uma mesma composição hipermídia

Se essa composição hipermídia for especificada como um template, a mesma especificação de relacionamentos temporais pode ser reusada por áudios distintos e suas legendas correspondentes, como por exemplo, todas as músicas de uma gravadora, que detém os direitos autorais. As próximas seções apresentam em detalhes como um template de composição é criado, usando o mesmo exemplo como ilustração.

5.1
Especificando o Vocabulário de um Template

Um template de composição pode definir tipos específicos de componentes (*component*), tais como texto, áudio, vídeo, imagem, composição etc. Cada tipo pode declarar um atributo rótulo (*label*) e ainda os números mínimo (*minOccurs*) e máximo (*maxOccurs*) de instâncias para cada um.

Os rótulos são usados para distinguir os tipos de componentes e especificar restrições e relacionamentos entre eles. Por isso, se for necessário especificar relacionamentos entre âncoras/atributos específicos de um tipo de componente, o template pode ainda determinar rótulos para pontos de interface de um componente, chamados de portas (*port*) (Muchaluat-Saade, 2001c), representando suas âncoras/atributos.

No vocabulário de um template também pode-se definir tipos de conectores (*connector*) que serão usados para criar elos entre os componentes do template. Cada tipo de conector tem um atributo *src* que identifica a URI de um elemento *xconnector* (definido usando a linguagem XConnector, veja o Capítulo 4), determinando também seu rótulo e os números mínimo e máximo de instâncias para cada tipo.

Para o exemplo da Figura 16, a especificação do vocabulário do template é dada na Figura 17. São definidos três tipos de componentes, um tipo áudio, chamado “audio”, que só pode ocorrer uma vez; um tipo texto, chamado “subtitle”, que pode ter várias ocorrências; e um tipo imagem, chamado “logo”, que só tem uma ocorrência. Componentes do tipo áudio podem ter várias portas distintas, chamadas “track”, representando cada trecho do áudio. Além dos componentes, são definidos dois tipos de conectores *L* e *P*, considerando que esses conectores estejam armazenados em uma biblioteca chamada “connector_base.xml”.

```
<xtemplate id=”audio-with-subtitles”>
 <vocabulary>
   <component label="audio" type="audio" maxOccurs="1">
      <port label="track" maxOccurs="unbounded" />
   </component>
   <component label="subtitle" type="text" maxOccurs="unbounded" />
   <component label="logo" type="img" maxOccurs="1" />

   <connector src="../connector_base.xml#L" label="L"
              maxOccurs="unbounded" />
   <connector src="../connector_base.xml#P" label="P"
```

```
              maxOccurs="unbounded" />
 </vocabulary>
...
</xtemplate>
```

Figura 17. Exemplo of vocabulário de um template

5.2
Especificando o Conjunto de Restrições de um Template

A especificação de restrições de um template é feita em três partes distintas:

- restrições sobre componentes, conectores e pontos de interface,
- definição de instâncias de componentes, identificando nós específicos,
- definição de instâncias de conectores, definindo elos específicos entre componentes.

O vocabulário de um template especifica que seus componentes devem satisfazer os requisitos impostos com relação ao tipo e número de elementos. Entretanto, restrições adicionais podem ser necessárias. Por exemplo, um template pode especificar restrições sobre componentes, conectores e pontos de interface, indicando que não pode haver nenhum outro tipo de componente, que não seja os declarados no vocabulário do template, ou que dois tipos de componentes devem ter o mesmo número de instâncias etc. Expressões da linguagem XPath (W3C, 1999b) podem ser usadas com esse propósito, generalizando os tipos de restrições que podem ser especificadas. O uso de XPath também torna possível realizar a validação de restrições facilmente, como será comentado na Seção 6.3.

Voltando ao exemplo da Figura 16, são definidas três restrições, como exibido na Figura 18. Cada restrição define uma expressão XPath, em seu atributo *select*, e pode definir um atributo descrição (*description*) cujo valor pode ser usado como uma mensagem de erro, se a restrição não for satisfeita. Cada restrição deve definir uma expressão XPath que retorne um valor booleano.

```
<xtemplate id=”audio-with-subtitles”>
...
   <constraints>
      <constraint select="count(child::link[@type='L']) =
count(child::link[@type='P'])" description=" The number of links of type
R must be equal to the number of links of type P"/>
      <constraint select="count(//*[@label='track']) =
count(child::*[@label='subtitle'])" description="The number of tracks
must be equal to the number of subtitles"/>
      <constraint select="count(child::*[@label!='audio'] |
child::*[@label!='subtitle'] | child::*[@label!='logotele']) =
(count(child::*)-count(child::linkBase))" description="All components
must be audios, subtitles or logos"/>
```

```
      ...
   </constraints>
</xtemplate>
```

Figura 18. Exemplo de restrições de um template

Diferente de estilos em ADLs, templates de composição hipermídia também podem definir instâncias de componentes e conectores. Assim sendo, um template de composição pode definir nós específicos, como por exemplo, recursos web, como instâncias de componentes e ainda elos específicos, como instâncias de conectores. Essas definições também são feitas dentro do conjunto de restrições de um template de composição.

A definição de instâncias de componentes é feita por elementos *resource*, que especificam a URI do conteúdo; o tipo de componente (*type*), que deve identificar um *label* já definido no vocabulário do template; e outro rótulo (*label*) para essa instância de componente. Como exemplo, considere a definição do logotipo da empresa que detém os direitos autorais do mesmo template que vem sendo utilizado, dada na Figura 19.

```
<xtemplate id=”audio-with-subtitles”>
...
   <constraints>
      ...
      <resource src=”http://www.telemidia.puc-rio.br/img/logo.jpg”
                type=”logo” label="logotele"/>
      ...
   </constraints>
</xtemplate>
```

Figura 19. Exemplo de definição de instâncias de componentes em um template

A definição de instâncias de conectores (elos) dá a semântica de um template de composição, já que os elos fornecem um comportamento de sincronização temporal ou espacial específico entre componentes. Na definição dos elos, os componentes e conectores são identificados pelos rótulos que foram especificados no vocabulário do template (atributo *label* de um tipo de componente ou conector) ou na definição de uma instância específica de componente (atributo *label* de um elemento *resource*).

Como conectores devem ter sido especificados através da linguagem XConnector (veja o Capítulo 4), cada elo define um conjunto de *binds* relacionando rótulos de componentes a papéis do conector. Ao especificar um *bind*, o papel de um conector é identificado pelo seu *id* e o(s) componente(s) do template é (são) identificado(s) através de uma expressão XPath que o(s) seleciona, fazendo uso de seu(s) rótulo(s). Para especificar os relacionamentos do

template, pode-se utilizar instruções para declaração de variáveis (*variable*), instruções para realizar repetição (*for-each*) e ainda funções XPath adicionadas pela linguagem XSLT (W3C, 1999c), como por exemplo *current()*.

A Figura 20 apresenta a definição dos elos do mesmo exemplo de template ilustrado na Figura 16, onde cada elo identifica o rótulo de um conector declarado no vocabulário do template (exibido na Figura 17).

```
<xtemplate id=”audio-with-subtitles”>
<constraints>
...
   <link type="L">
      <bind role="source" select="child::*[@label='audio']"/>
      <bind role="target" select="child::*[@label='logotele']"/>
   </link>
   <link type="P">
      <bind role="source" select="child::*[@label='audio']"/>
      <bind role="target" select="child::*[@label='logotele']"/>
   </link>

   <variable name="i" select="1"/>

   <for-each select="child::*[@label='audio']/child::*[@label='track']" >
      <link type="L">
         <bind role="source" select="current()" />
         <bind role="target" select="child::*[@label='subtitle'][$i]"/>
      </link>
      <link type="P">
         <bind role="source" select="current()" />
         <bind role="target" select="child::*[@label='subtitle'][$i]"/>
      </link>
      <variable name="i" select="$i + 1"/>
   </for-each>

</constraints>
</xtemplate>
```

Figura 20. Exemplo de definição de elos em um template

5.3
Outras Considerações sobre Templates de Composição

De forma similar a estilos arquiteturais, templates podem ser definidos como especialização de outros templates, herdando todas as definições anteriores e ainda introduzindo novas. Para ilustrar essa facilidade, um outro exemplo de template será considerado.

Como discutido em (Rodrigues, 2002a), composições paralelas e seqüenciais podem ser especificadas por elos que determinam os relacionamentos de sincronização entre seus componentes. Uma composição paralela com a semântica idêntica à do elemento *par* do SMIL, representada por elos, é mostrada na Figura 21. O elo L_{in} causa o início da apresentação dos componentes da

composição, disparado pelo início da apresentação da própria composição. O elo P_{out} causa o fim da apresentação dos componentes da composição, disparado pelo fim da apresentação da composição. A Figura 22 ilustra a definição do template de composição *par*. Note que qualquer tipo de componente pode ser incluído em uma composição que use esse template, pois o template não determina nenhum tipo específico de componente.

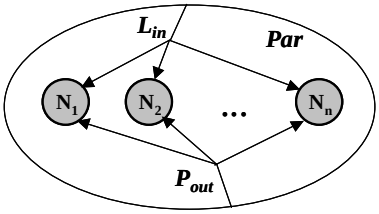

Figura 21. Composição paralela representada por elos

```
<xtemplate id="par">
<vocabulary>
      <connector src="../connector_base.xml#L" label="L" />
      <connector src="../connector_base.xml#P" label="P" />
   </vocabulary>
   <constraints>
      <link type="L">
         <bind role="source" select="self::composite" />
         <bind role="target" select="*" />
      </link>
      <link type="P">
         <bind role="source" select="self::composite" />
         <bind role="target" select="*" />
      </link>
   </constraints>
</xtemplate>
```

Figura 22. Template de composição *par*

Agora, o template de composição *par* pode ser estendido para adicionar a semântica de terminar a composição pelo fim de algum componente específico. Nesse caso, um elo que causa o término da apresentação da composição disparado pelo término da apresentação do componente específico é adicionado, como ilustrado pelo elo P_{sync} na Figura 23(a). No caso da composição terminar pelo primeiro ou último componente, os elos P_{first} e P_{last}, representados respectivamente na Figura 23(b) e na Figura 23(c), precisam ser adicionados. A Figura 24 ilustra os templates de composição *par-sync*, *par-first* e *par-last* representando os três tipos de composição mencionados. Note que cada um deles estende o template *par*. O template de composição *par-sync* define um componente especial com rótulo "*endsync*" que representa o componente específico que determina o término da apresentação da composição.

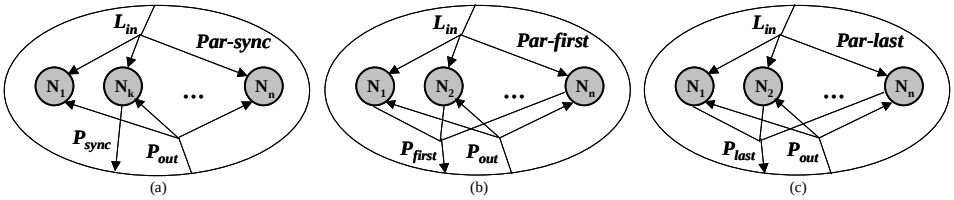

Figura 23. Composição paralela (a) terminada por um componente específico; (b) terminada pelo primeiro; (c) terminada pelo último

```
<xtemplate id="par-sync">
   <extends xtemplate="par" />
   <vocabulary>
      <component label="endsync" maxOccurs="1" />
      <connector src="../connector_base.xml#P " label="P-sync"
                 maxOccurs="1"/>
   </vocabulary>
   <constraints>
      <link type="P-sync">
         <bind role="source" select="child::*[@label='endsync']" />
         <bind role="target" select="self::composite" />
      </link>
   </constraints>
</xtemplate>

<xtemplate id="par-first">
   <extends xtemplate="par" />
   <vocabulary>
      <connector src="../connector_base.xml#P-first" label="P-first"
                 maxOccurs="1" />
   </vocabulary>
   <constraints>
      <link type=”P-first">
         <bind role=“source” select="*" />
         <bind role=“target” select="self::composite" />
      </link>
   </constraints>
</xtemplate>

<xtemplate id="par-last">
   <extends xtemplate="par" />
   <vocabulary>
      <connector src="../connector_base.xml#P-last" label="P-last"
                 maxOccurs="1" />
   </vocabulary>
   <constraints>
      <link type="P-last">
         <bind role=“source_condition1” select="*" />
         <bind role=“source_condition2” select="*" />
         <bind role=“target” select="self::composite" />
      </link>
   </constraints>
</xtemplate>
```

Figura 24. Templates de composição *par-sync*, *par-first* e *par-last*

Com o propósito de tornar a linguagem XTemplate mais genérica e independente da linguagem de autoria hipermídia principal, em um template de composição, não é possível especificar características de apresentação de componentes ou elos definidos pelo template. Essa limitação é necessária, pois normalmente a especificação de características de exibição depende da linguagem

em questão. Por exemplo, em NCL, isso é feito através de elementos descritores (*descriptor*), já em SMIL, são usados atributos definidos em cada componente do documento. Dessa forma, quando uma composição utiliza um template, deve-se definir tais características de apresentação após o processamento do template, ou seja, diretamente no documento final. Na próxima seção, que apresenta um exemplo de uso de template, esse caso também é exemplificado.

Finalmente, vale a pena mencionar que um dos objetivos de XTemplate é prover composicionalidade, permitindo a definição de pontos de interface e do aninhamento de templates. Na proposta corrente, entretanto, a composicionalidade foi limitada a um único ponto de interface, que representa a composição inteira. Como trabalho futuro, pretende-se refinar a linguagem XTemplate para permitir a especificação de um template de composição que possa utilizar outros templates aninhados como seus componentes.

5.4
Exemplo de Uso de Templates de Composição

Para ilustrar o uso de templates de composição, considere um documento NCL com uma composição identificada por “samba-document” contendo um nó de áudio com o samba “Coisa de Pele”, interpretado por Beth Carvalho, que define uma série de âncoras, uma para cada trecho cantado do áudio. O mesmo documento contém ainda uma série de nós com texto HTML, cada um deles representando partes da letra do samba, correspondentes a cada trecho cantado. Para sincronizar a apresentação da letra do samba com cada trecho correspondente do áudio, pode-se utilizar o template “audio-with-subtitles.xml”, que propositalmente foi o exemplo de template de composição usado nas seções anteriores, para ilustrar a definição de templates. A Figura 25 mostra a definição completa desse template.

```
<?xml version="1.0" ?>
<xtemplate name="XTemplate Example" id="audio-with-subtitles"
 xmlns="http://www.telemidia.puc-rio.br/specs/xml/XTemplate"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.telemidia.puc-rio.br/specs/xml/XTemplate
http://www.telemidia.puc-rio.br/specs/xml/XTemplate.xsd">

<vocabulary>
   <component label="audio" type="audio" maxOccurs="1">
      <port label="track" maxOccurs="unbounded" />
   </component>
```

```
   <component label="subtitle" type="text" maxOccurs="unbounded" />
   <component label="logo" type="img" maxOccurs="1" />

   <connector src="../connector_base.xml#L" label="L"
              maxOccurs="unbounded" />
   <connector src="../connector_base.xml#P" label="P"
              maxOccurs="unbounded" />
 </vocabulary>

<constraints>

   <constraint select="count(child::link[@type='L']) =
count(child::link[@type='P'])" description=" The number of links of type
R must be equal to the number of links of type P"/>
   <constraint select="count(//*[@label='track']) =
count(child::*[@label='subtitle'])" description="The number of tracks
must be equal to the number of subtitles"/>
   <constraint select="count(child::*[@label!='audio'] |
child::*[@label!='subtitle'] | child::*[@label!='logotele']) =
(count(child::*)-count(child::linkBase))" description="All components
must be audios, subtitles or logos"/>

   <resource src="http://www.telemidia.puc-rio.br/img/logo.jpg"
             type="logo" label="logotele"/>

   <link type="L">
      <bind role="source" select="child::*[@label='audio']"/>
      <bind role="target" select="child::*[@label='logotele']"/>
   </link>
   <link type="P">
      <bind role="source" select="child::*[@label='audio']"/>
      <bind role="target" select="child::*[@label='logotele']"/>
   </link>
   <variable name="i" select="1"/>
   <for-each select="child::*[@label='audio']/child::*[@label='track']" >
      <link type="L">
         <bind role="source" select="current()" />
         <bind role="target" select="child::*[@label='subtitle'][$i]"/>
      </link>
      <link type="P">
         <bind role="source" select="current()" />
         <bind role="target" select="child::*[@label='subtitle'][$i]"/>
      </link>
      <variable name="i" select="$i + 1"/>
   </for-each>
</constraints>
</xtemplate>
```

Figura 25. Template de composição “audio-with-subtitles”

Para que um nó de composição use um template, ele deve fazer referência a URI do template e especificar rótulos em seus nós componentes que casam com a definição de rótulos feita no vocabulário do template. Assim sendo, o documento NCL que descreve a composição com o áudio do samba e seus trechos é ilustrado na Figura 26. Note que os atributos necessários para uso do template “audio-with-subtitles” estão realçados. Para simplificar o exemplo, apenas os três primeiros trechos da letra da música são utilizados.

```
<?xml version="1.0" ?>
<ncl id="ncl_example"
   xmlns="http://www.telemidia.puc-rio.br/specs/xml/NCL"
```

```
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.telemidia.puc-rio.br/specs/xml/NCL
http://www.telemidia.puc-rio.br/specs/xml/NCL.xsd">
<head>
  <layout>
    <topLayout id="window1" title="Music Lyrics" width="600"
               height="200">
       <region id="textRegion1" width="65%" height="100%"/>
       <region id="imageRegion1" width="35%" height="100%" left="65%"/>
    </topLayout>
    <topLayout id="audioWindow" title="Audio Control" top="200"
               width="600" height="50">
       <region id="audioRegion1" width="100%" height="100%"/>
    </topLayout>
  </layout>
  <descriptorBase>
    <descriptor id="audio_d1" region="audioRegion1" enableTimeBar="on" />
    <descriptor id="text_d1" region="textRegion1" />
    <descriptor id="img_d1" region="imageRegion1" />
  </descriptorBase>
</head>
<body>
  <composite id="samba-document" xtemplate="http://engenho.telemidia.puc-
rio.br/users/debora/audio-with-subtitles.xml">
   <audio id="samba" label="audio" src="http://www.telemidia.puc-
rio.br/specs/xml/examples/samba/samba.wav" descriptor="audio_d1">
      <area id="part1" begin="8.4s"  end ="18s" label="track"/>
      <area id="part2" begin="18.5s" end ="28s" label="track"/>
      <area id="part3" begin="29s"   end ="39s" label="track"/>
   </audio>
   <text id="lyrics-part1" label="subtitle"
         src="http://www.telemidia.puc-
rio.br/specs/xml/examples/samba/lyrics01.htm" descriptor="text_d1"/>
   <text id="lyrics-part2" label="subtitle"
         src="http://www.telemidia.puc-
rio.br/specs/xml/examples/samba/lyrics02.htm" descriptor="text_d1"/>
   <text id="lyrics-part3" label="subtitle"
         src="http://www.telemidia.puc-
rio.br/specs/xml/examples/samba/lyrics03.htm" descriptor="text_d1"/>
  </composite>
</body>
</ncl>
```

Figura 26. Documento NCL usando o template de composição “audio-with-subtitles”

Depois do processamento do documento usando o template, o documento NCL final gerado contém os elos e nós definidos pelo template, além dos já definidos pelo documento NCL original. O documento final é exibido na Figura 27. Note que os elementos e atributos gerados pelo processamento do template estão realçados. Lembre-se que o template, além de gerar os elos sincronizando o áudio com os nós texto, define também um componente específico que é um nó imagem com o logotipo do TeleMídia.

```
<?xml version="1.0" encoding="UTF-8"?>
<ncl id="ncl_example-processed"
     xmlns="http://www.telemidia.puc-rio.br/specs/xml/NCL"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.telemidia.puc-rio.br/specs/xml/NCL
http://www.telemidia.puc-rio.br/specs/xml/NCL.xsd">

<head>
```

```
  <layout>
    <topLayout height="200" id="window1" title="Music Lyrics"
               width="600">
      <region height="100%" id="textRegion1" width="65%"/>
      <region height="100%" id="imageRegion1" left="65%" width="35%"/>
    </topLayout>
    <topLayout height="50" id="audioWindow" title="Audio Control"
               top="200" width="600">
      <region height="100%" id="audioRegion1" width="100%"/>
    </topLayout>
  </layout>

  <descriptorBase>
    <descriptor enableTimeBar="on" id="audio_d1" region="audioRegion1"/>
    <descriptor id="text_d1" region="textRegion1"/>
    <descriptor id="img_d1" region="imageRegion1"/>
  </descriptorBase>
</head>

<body>
  <composite id="samba-document">
    <audio descriptor="audio_d1" id="samba"
           src="http://www.telemidia.puc-
rio.br/specs/xml/examples/samba/samba.wav">
      <area begin="8.4s" end="18s" id="part1"/>
      <area begin="18.5s" end="28s" id="part2"/>
      <area begin="29s" end="39s" id="part3"/>
    </audio>
    <text descriptor="text_d1" id="lyrics-part1"
          src="http://www.telemidia.puc-
rio.br/specs/xml/examples/samba/lyrics01.htm"/>
    <text descriptor="text_d1" id="lyrics-part2"
          src="http://www.telemidia.puc-
rio.br/specs/xml/examples/samba/lyrics02.htm"/>
    <text descriptor="text_d1" id="lyrics-part3"
          src="http://www.telemidia.puc-
rio.br/specs/xml/examples/samba/lyrics03.htm"/>

    <img id="logotele1" src="http://www.telemidia.puc-
rio.br/img/logo.jpg"/>

    <linkBase>
      <link id="link1" xconnector="../connector_base.xml#L">
        <bind component="samba" role="source"/>
        <bind component="logotele1" role="target"/>
      </link>
      <link id="link2" xconnector="../connector_base.xml#P">
        <bind component="samba" role="source"/>
        <bind component="logotele1" role="target"/>
      </link>
      <link id="link3" xconnector="../connector_base.xml#L">
        <bind component="samba" port="part1" role="source"/>
        <bind component="lyrics-part1" role="target"/>
      </link>
      <link id="link4" xconnector="../connector_base.xml#P">
        <bind component="samba" port="part1" role="source"/>
        <bind component="lyrics-part1" role="target"/>
      </link>
      <link id="link5" xconnector="../connector_base.xml#L">
        <bind component="samba" port="part2" role="source"/>
        <bind component="lyrics-part2" role="target"/>
      </link>
      <link id="link6" xconnector="../connector_base.xml#P">
        <bind component="samba" port="part2" role="source"/>
        <bind component="lyrics-part2" role="target"/>
      </link>
```

```
      <link id="link7" xconnector="../connector_base.xml#L">
        <bind component="samba" port="part3" role="source"/>
        <bind component="lyrics-part3" role="target"/>
      </link>
      <link id="link8" xconnector="../connector_base.xml#P">
        <bind component="samba" port="part3" role="source"/>
        <bind component="lyrics-part3" role="target"/>
      </link>
    </linkBase>
  </composite>
</body>
</ncl>
```

Figura 27. Documento NCL final gerado após o processamento do template “audio-with-subtitles”

Depois de gerar o documento NCL final, o parser NCL do sistema HyperProp foi utilizado para importar o documento para o sistema e apresentá-lo com o uso do formatador. A Figura 28 ilustra a visão estrutural do documento importado, usando o browser gráfico do HyperProp. O número (2) próximo de cada seta significa que cada uma delas representa dois elos entre os nós interligados. Note a semelhança da visão exibida na Figura 28 com a visão estrutural da Figura 16.

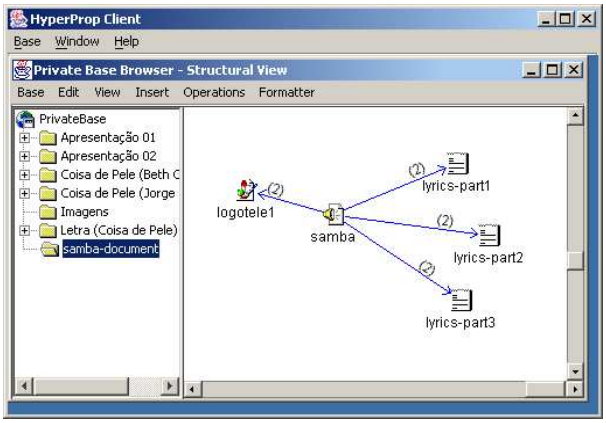

Figura 28. Visão estrutural do documento final no sistema HyperProp

Ao executar o documento final “samba-document” usando o formatador HyperProp, cada trecho do áudio é sincronizado com a parte da letra correspondente, o que foi especificado pelo template utilizado no documento original. A Figura 29, a Figura 30 e a Figura 31 exibem o documento apresentado ao usuário final, mostrando as três primeiras partes da letra do samba sincronizadas com o áudio (note o progresso na barra de rolagem de tempo que controla a exibição do áudio).

Figura 29. Primeira tela da apresentação do documento “samba-document”

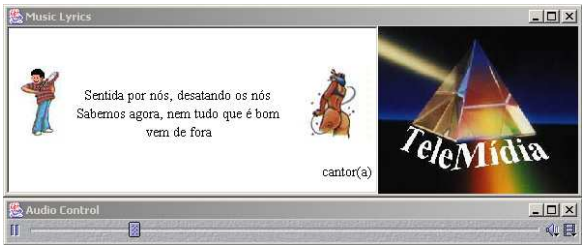

Figura 30. Segunda tela da apresentação do documento “samba-document”

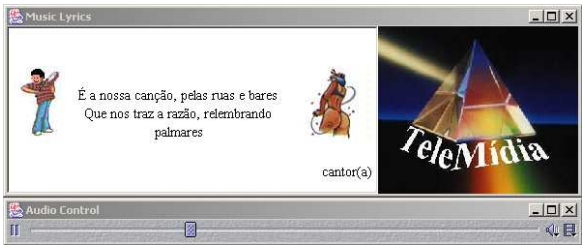

Figura 31. Terceira tela da apresentação do documento “samba-document”

Como discutido detalhadamente nesta tese, as características de apresentação de cada nó em um documento NCL são definidas através de um elemento descritor, que deve ser associado ao nó. Note que o documento NCL original contendo a composição “samba-document” só podia associar descritores aos nós que já pertenciam à composição, ou seja, ao no áudio e aos textos HTML, apesar de ter definido um descritor chamado “img_d1”, não utilizado até então. As características de apresentação de nós e elos gerados automaticamente por um template não são definidas de forma automática. Esse fato aconteceu com o nó imagem contendo o logotipo do TeleMídia. Por esse motivo, torna-se necessário editar o documento NCL final gerado após o processamento do template, ou realizar a edição gráfica disponível no HyperProp, após o documento ser importado para o sistema. No exemplo mostrado, o descritor “img_d1” foi associado ao nó imagem “logotele1” diretamente no sistema HyperProp. Por essa razão, o nó imagem foi mostrado em outra região na mesma janela da letra da música, como especificado pelo conjunto de descritores no documento original.

5.5
Extensões ao padrão XLink

Além das extensões propostas na Seção 4.1.1 ao padrão XLink, permitindo o uso de conectores hipermídia para determinar a semântica de regras de navegação em um elo estendido XLink, novas extensões podem ser propostas ao padrão. Essa nova proposta tem como objetivo melhorar o reuso de especificações XLink, através do uso de alguns conceitos de template de composição.

Já que um elo estendido pode definir várias regras de navegação entre um conjunto de participantes, cada elo estendido XLink, na verdade, representa um conjunto de relacionamentos (elos hipermídia) entre os participantes, ao invés de um relacionamento único. Com isso, um elo estendido pode ser considerado uma composição hipermídia contendo componentes (os participantes) e um conjunto de elos entre eles (as regras de navegação). A idéia central dessa nova proposta de extensão é criar um template para definir um conjunto de regras de navegação XLink, independente de quais recursos web são participantes do elo estendido. Fazendo referência a um mesmo template, vários elos estendidos distintos, cada um deles podendo definir conjuntos distintos de participantes, podem reusar o mesmo conjunto de regras de navegação. Isso aumenta ainda mais o reuso em XLink.

Para fornecer a definição de templates, um novo tipo XLink é proposto, chamado *template*. Um elemento do tipo *template* é bastante similar a um elemento tradicional do tipo *extended*, e deve ainda declarar um atributo *id* que permita que um elo estendido faça uma referência a ele. Elementos do tipo *template* devem declarar elementos-filho do tipo *arc*, especificando um conjunto de regras de navegação. Cada elemento do tipo *arc* deve seguir as extensões propostas na Seção 4.1.1 em sua definição, tendo um atributo *xconnector* que identifica a URI de um conector hipermídia e declarando elementos-filho do tipo *bind*, que relacionam rótulos a papéis. A Figura 32 ilustra um exemplo de definição de template XLink representando os relacionamentos entre um curso, o professor e seus alunos, chamado “courseload-template”. A definição do conector hipermídia “xlink-replace-onRequest”, usado nas regras de navegação da figura, foi exibida na Figura 14 (veja Seção 4.1.1).

```
<xtemplate xlink:type=”template” xtemplate:id=”courseload-template”>
```

```
   <go xlink:type=″arc″ xconnector:xconnector=″xlink-replace-onRequest″>
      <bind xlink:type=″bind″ xconnector:label=″student″
xconnector:role=″from″/>
      <bind xlink:type=″bind″ xconnector:label=″professor″
            xconnector:role=″from″/>
      <bind xlink:type=″bind″ xconnector:label=″course″
xconnector:role=″to″/>
   </go>
   <go xlink:type=″arc″ xconnector:xconnector=″xlink-replace-onRequest″>
      <bind xlink:type=″bind″ xconnector:label=″course″
xconnector:role=″from″/>
      <bind xlink:type=″bind″ xconnector:label=″professor″
xconnector:role=″to″/>
   </go>
<xtemplate/>
```

Figura 32. Exemplo de elemento XLink do novo tipo *template*

Para usar um template XLink, elementos do tipo *extended* (elos estendidos) precisam de um novo atributo, chamado *xtemplate*, que deve identificar a URI de um elemento do tipo *template*. Seguindo as regras já existentes no padrão XLink, cada participante de um elo estendido deve definir um rótulo (*label*), que será usado na definição de regras de navegação. Assim sendo, os participantes que definirem rótulos usados no template, referenciado pelo elo estendido, herdam as regras de navegação definidas no template. Um elo estendido XLink pode continuar definindo outras regras de navegação independentes das que já foram definidas pelo template. A Figura 33 mostra dois elos estendidos XLink usando as novas extensões propostas. Esses dois elos representam dois cursos distintos, “cs101” e “cs105”, que fazem referência ao template “courseload-template” e declara os rótulos “student”, “course” e “professor” para seus participantes, reusando o conjunto de regras de navegação já definidas pelo template.

```
<courseload xlink:type=″extended″
 xtemplate:xtemplate=”../template_base.xml#courseload-template”>
   <person xlink:type=″locator″ xlink:href="students/patjones62.xml"
           xlink:label="student"/>
   <person xlink:type=″locator″ xlink:href="students/peterkorb60.xml"
           xlink:label="student"/>
   <person xlink:type=″locator″ xlink:href="professors/jaysmith7.xml"
           xlink:label="professor"/>
   <course xlink:type=″locator″ xlink:href="courses/cs101.xml"
           xlink:label="course"/>
</courseload>

<courseload xlink:type=″extended″
 xtemplate:xtemplate=”../template_base.xml#courseload-template”>
   <person xlink:type=″locator″ xlink:href="students/kentulm55.xml"
           xlink:label="student"/>
   <person xlink:type=″locator″ xlink:href="students/carolben50.xml"
           xlink:label="student"/>
   <person xlink:type=″locator″ xlink:href="professors/samueljosh5.xml"
           xlink:label="professor"/>
   <course xlink:type=″locator″ xlink:href="courses/cs105.xml"
           xlink:label="course"/>
</courseload>
```

Figura 33. Exemplos de dois elos estendidos reusando o mesmo template “courseload-template”

Note que essa nova proposta de extensão ao padrão XLink só usou um subconjunto das facilidades providas por XTemplate. O objetivo inicial da proposta de extensão foi permitir que um conjunto de regras de navegação XLink pudesse ser reusado por vários elos estendidos com o mesmo comportamento de navegação entre seus participantes, aumentando ainda mais o reuso em definições XLink. Agora, torna-se possível a definição de bibliotecas de templates XLink, armazenando conjuntos de regras de navegação independentes dos recursos web participantes de cada elo estendido.

Em resumo, além das vantagens salientadas na Seção 4.1.1, decorrentes da proposta de extensão do padrão XLink com facilidades da linguagem XConnector, pode-se destacar vantagens adicionais de estender XLink com algumas facilidades de templates de composição:

- facilidade de uso – aplicando a mesma idéia de bases de elos e bases de conectores, bases de templates podem ser criadas e reusadas para definir elos estendidos XLink;
- facilidade de manutenção dos elos – quando um template é modificado, todos os elos estendidos que fazem referência a ele são automaticamente modificados. Apesar de poder ser colocada como uma facilidade, essa característica também pode trazer dificuldades, pois quando um template é modificado, pode ser que os elos estendidos que o utilizam devam ser modificados também, para refletir as mudanças em sua definição. Caso contrário, o documento pode ficar inconsistente.

5.6 Extensões em Outras Linguagens

Assim como XTemplate foi usado como módulo da linguagem NCL 2.0, e como parte das propostas de extensões ao padrão XLink para incorporar facilidades de templates de composição, pode-se considerar a possibilidade de usar XTemplate, propondo extensões em outras linguagens.

Comparando a metodologia proposta por templates de composição com a modularização da versão 2.0 da linguagem SMIL, um perfil chamado

SMIL+XTemplate, poderia ser criado. Esse perfil permitiria a criação de outros tipos de contêineres temporais, além dos famosos *par*, *seq* and *excl*, que poderiam ser usados para a autoria de documentos SMIL. Para dar suporte à definição de templates, os módulos *BasicComposite*, XConnector e XTemplate deveriam ser adicionados ao perfil. O módulo *BasicComposite* é necessário para introduzir o elemento *composite*, que pode adquirir a semântica definida por um template qualquer. Além da inclusão desses três módulos, dois novos módulos teriam que ser criados. O primeiro módulo novo deveria permitir a definição de elos SMIL através do uso de conectores, similar ao que foi feito no módulo *Linking* de NCL 2.0. O outro módulo novo forneceria o uso de templates de composição. Esse último módulo adicionaria o atributo *xtemplate* a elementos *composite* e o atributo rótulo (*xtemplate:label*) a objetos de mídia, âncoras e contêineres temporais tradicionais SMIL, com objetivos similares aos do módulo *XTemplateUse* de NCL 2.0. Formatadores SMIL teriam que ser adaptados para tratar especificações de sincronização feitas através de XConnector e XTemplate.