# 4
# X-SMIL

Visando aumentar o reuso e a expressividade da linguagem SMIL (W3C, 2001b), descrita no Capítulo 2, este capítulo apresenta a linguagem X-SMIL. X-SMIL é a combinação de duas extensões à SMIL (conforme citado no Capítulo 1): XT-SMIL, que introduz o conceito de *templates* de composição; e XC-SMIL, que permite a definição de elos por meio do reuso de conectores hipermídia.

Este capítulo descreve, inicialmente, a extensão XT-SMIL. Em seguida, é definida a linguagem XC-SMIL e, como conseqüência, apresenta-se X-SMIL.

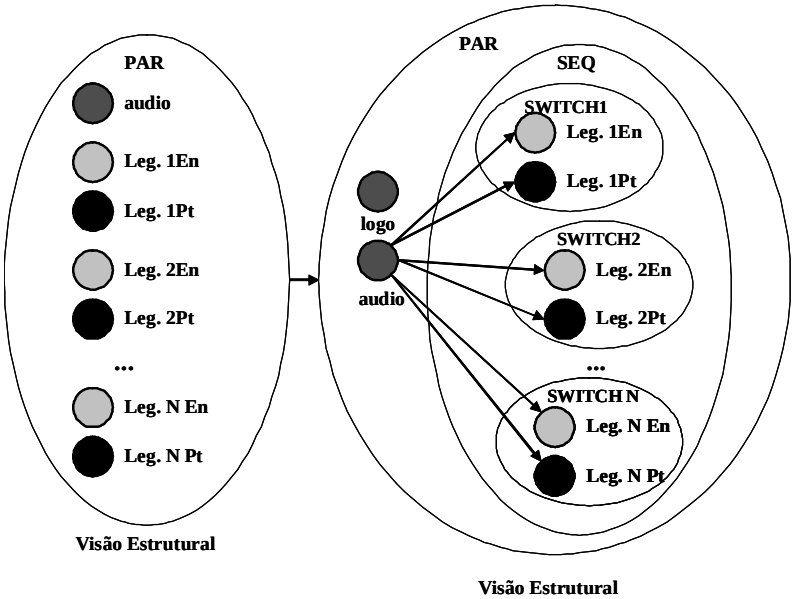


## 4.1.
## XT-SMIL: SMIL + XTemplate

O capítulo anterior definiu a linguagem XTemplate 2.1 para especificação de *templates* de composição. XTemplate é estruturada em módulos, podendo, dessa forma, definir perfis de linguagem. Para incorporar o conceito de *templates* em SMIL, gerando a linguagem XT-SMIL (Silva et al., 2004b), é utilizado um perfil de XTemplate 2.1 que reúne todos os módulos dessa linguagem, exceto *ConnectorVocabulary* e *BasicLinking*. Esse perfil XTemplate, independente de conectores, permite, entre outras funcionalidades, a definição de componentes compostos e de relações de inclusão. A fim de permitir a utilização de *templates* em documentos XT-SMIL, adotou-se uma abordagem similar ao uso do módulo *XTemplateUse* em NCL, sendo adicionado o atributo *xtemplate* às composições *seq*, *par* e *excl*, e o atributo *type* aos elementos que podem estar contidos nessas composições. O uso desses atributos é ilustrado no final desta seção.

As principais funcionalidades adicionadas por XT-SMIL serão analisadas a partir de um exemplo de *template*: *audioComLegendasEnPt*. Esse *template* define

uma nova semântica[31] para composições XT-SMIL *par* que contenham um áudio e pares de legendas (do tipo texto) para cada trecho do áudio, sendo uma legenda em português e outra em inglês. Quando uma composição XT-SMIL paralela, como a exemplificada à esquerda da Figura 4:1, referencia esse *template*, ela ganha outra semântica, herdando todas as definições da configuração do *template*. O *template* define que seus pares de legendas serão agrupados em elementos do tipo *switch*, que serão sincronizados com as faixas do áudio. Esse sincronismo será obtido incluindo os *switches* em um container *seq*, e relacionando o término da apresentação de cada *switch* (ou seja, de suas legendas internas) com o término do trecho de áudio correspondente. Uma imagem (*logo*) será também adicionada à composição e exibida em paralelo com o áudio, devido à semântica tradicional da composição *par*. A visão estrutural da nova composição XT-SMIL gerada após o processamento do *template*[32] *audioComLegendasEnPt* é ilustrada na parte direita da Figura 4:1, onde é possível observar que o *template* definiu novas relações de inclusão. A visão temporal da mesma composição, após seu processamento, é apresentada na Figura 4:2.




[31] As composições XT-SMIL *par*, *seq* e *excl* possuem a mesma semântica temporal que as composições homônimas em SMIL. Porém, quando referenciam *templates* XT-SMIL, essas composições podem adquirir novas semânticas que estendam suas semânticas originais.

[32] O mecanismo de processamentos de *templates* e a geração das composições processadas serão discutidos no próximo capítulo.

Figura 4:1 - Visão estrutural de uma composição XT-SMIL *par* antes e após o processamento de *template*.

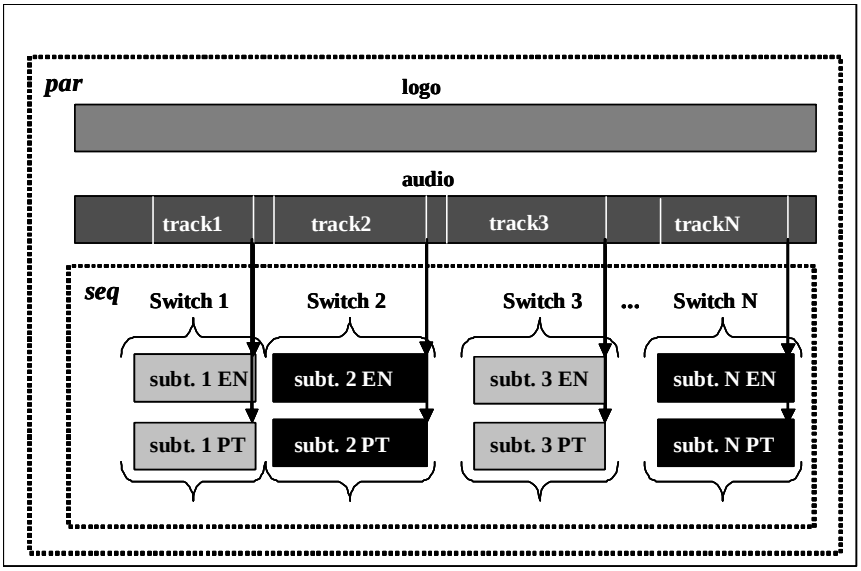



Figura 4:2 - Visão temporal de uma composição *par* em SMIL.

A especificação do *template audioComLegendasEnPt* está ilustrada na Figura 4:3. Em primeiro lugar, como pode ser verificado nas linhas 8 a 13 da Figura 4:3, existe a declaração de um componente composto. Esse componente é do tipo *seq*, que pode possuir uma quantidade ilimitada de componentes do tipo *switch*, sendo que cada *switch* possui, exatamente, dois componentes: um do tipo *subtitleEn* e outro do tipo *subtitlePt*. Esse aninhamento de componentes, como visto na Seção 3.5, é uma nova facilidade oferecida por XTemplate 2.1. A definição de componentes compostos no vocabulário do *template* permite que, além de restrições sobre cardinalidade e tipo, sejam especificadas restrições sobre o aninhamento de componentes. Na Figura 4:3, por exemplo, foram especificados os tipos dos componentes que podem estar contidos nos componentes de tipo *seq* e *switch*.

```
01  <xtemplate>
02  <head>
03   <vocabulary>
04    <component type="song" ctype="audio" maxOccurs="1" minOccurs="1">
05     <port type="track" maxOccurs="unbounded" />
06    </component>
07    <component type="logo" ctype="img" maxOccurs="1" />
08    <component type="seq" ctype="seq">
09     <component type="switch" ctype="switch" maxOccurs="unbounded">
10      <component type="subtitleEn" ctype="text" maxOccurs="1" minOccurs="1" />
11      <component type="subtitlePt" ctype="text" maxOccurs="1" minOccurs="1" />
12     </component>
```

```
13    </component>
14   </vocabulary>
15   <constraints>
16    <constraint select="count(//*[@type='track']) =
count(//*[@type='subtitleEn'])" description="The number of tracks must be equal to
the number of subtitleEn"/>
17    <constraint select="count(//*[@type='track']) =
count(//*[@type='subtitlePt'])" description="The number of tracks must be equal to
the number of subtitlePt"/>
18   </constraints>
19   <resources>
20    <resource src="logo.jpg" type="logo" label="logoJPG" />
21   </resources>
22  </head>
23  <body>
24   <xsl:stylesheet>
25    <xsl:template match="/*/*/*">
26     <xsl:if test="not(./@type='subtitleEn') and not(./@type='subtitlePt')">
27      <xsl:copy-of select="." />
28     </xsl:if>
29    </xsl:template>
30    <xsl:resource type="seq" >
31     <xsl:for-each select="child::*[@ type ='subtitleEn']" >
32      <xsl:variable name="i" select="position()"/>
33      <xsl:resource type="switch">
34      <xsl:attribute name="id">switch<xsl:value-of select="$i"/></xsl:attribute>
35      <xsl:copy>
36       <xsl:for-each select="text()|@*">
37         <xsl:copy/>
38       </xsl:for-each>
39       <xsl:attribute name = "systemLanguage">en-us</xsl:attribute>
40       <xsl:attribute name="end">
41        <xsl:value-of
select="//*/child::*[@type='song']/child::*[@type='track'][$i]/@id"/>.end
42       </xsl:attribute>
43       <xsl:apply-templates/>
44      </xsl:copy>
45      <xsl:call-template name="ptSwitchElement">
46        <xsl:with-param name="i" select="$i" />
47      </xsl:call-template>
48      </xsl:resource>
49     </xsl:for-each>
50    </xsl:resource>
51    <xsl:template name="ptSwitchElement">
52     <xsl:param name="i"></xsl:param>
53     <xsl:for-each select="/*/*/child::*[@ type ='subtitlePt'][$i]" >
54      <xsl:copy>
55       <xsl:for-each select="text()|@*">
56        <xsl:copy/>
57       </xsl:for-each>
58       <xsl:attribute name = "systemLanguage">pt-br</xsl:attribute>
```

```
59       <xsl:attribute name="end">
60        <xsl:value-of
select="//*/child::*[@type='song']/child::*[@type='track'][$i]/@id"/>.end
61       </xsl:attribute>
62       <xsl:apply-templates/>
63      </xsl:copy>
64     </xsl:for-each>
65    </xsl:template>
66   </xsl:stylesheet>
67  </body>
68  </xtemplate>
```

Figura 4:3 - *Template audioComLegendasEnPt* em XT-SMIL.

As linhas 16 e 17 definem duas restrições adicionais às restrições definidas no vocabulário: o número de componentes do tipo *subtitleEn* e do tipo *subtitlePt* devem ser iguais ao número de pontos de interface do tipo *track*. A linha 20 define uma instância do componente *logo*, sendo seu identificador (*label*) *logoJPG*.

As linhas 24-66 demonstram uma outra facilidade de XTemplate 2.1: a possibilidade de aplicar uma transformada XSLT diretamente aos elementos de uma composição, pelo uso do elemento *xsl:stylesheet* como filho direto do elemento *body* (o que possibilita, por exemplo, a definição de relações de inclusão). A explicação dessa transformada será realizada por partes, iniciando pelas linhas 25-29. Essas linhas especificam que os elementos do tipo *subtitleEn* e *subtitlePt* deixam de ser filhos diretos da composição. Como será visto a seguir, elementos desses tipos passam a ser contidos *recursivamente* pela mesma (ou seja, eles são definidos como filhos de outros componentes da composição). Esse trecho do *template* declara, portanto, que todos os componentes de uma composição, à exceção dos de tipo *subtitleEn* e *subtitlePt*, devem ser mantidos como seus filhos.

As linhas 30-50 definem a criação de um elemento *seq*. O elemento *seq* é declarado como um recurso de *TemplateXSLT* (elemento *xsl:resource* referenciando o tipo de componente *seq* do vocabulário) e tem como conteúdo elementos do tipo *switch*. Mais especificamente, as linhas 31-49 definem uma iteração sobre os componentes do tipo *subtitleEn* para definição dos *switches* que compõem o elemento *seq*. Os *switches* são definidos nas linhas 33-48, referenciando o tipo de componente *switch* do vocabulário.

O valor do atributo *id* de cada *switch* é definido na linha 34. As linhas 35-47 contêm a definição do conteúdo de cada elemento *switch*, feita através da iteração

sobre os elementos do tipo texto definidos na composição. Primeiramente, define-se o elemento *subtitleEn*, referente ao passo *i* da iteração, como conteúdo do *switch i* (linhas 35-44). Em seguida, o mesmo ocorre para o elemento *subtitlePt*, através de uma chamada (linhas 45-47) ao *xsl:template* (W3C, 1999d) *templatePtSwitchElement*, definido nas linhas 51-65.

As linhas 39 e 58 definem, respectivamente, o atributo *systemLanguage* para os elementos de tipo *subtitleEn* e *subtitlePt* como sendo *en-us* (inglês americano) e *pt-br* (português brasileiro); ou seja, a escolha entre as alternativas para os componentes (textos) do *switch* deve ser baseada no idioma do contexto de exibição. As linhas 40-42 e 59-61 especificam a sincronização das legendas com as faixas de áudio. Para isso, é incluído um atributo *end* nos elementos de texto representando legendas. Esse atributo determina que cada legenda termine juntamente com a i-ésima faixa de áudio. O início da legenda seguinte é obtido pela semântica do elemento *seq*, que contém todas as legendas. Note que a inclusão de atributos em elementos, como os atributos *systemLanguage* e *end*, é, também, uma nova facilidade introduzida por XTemplate (por possibilitar que folhas de estilo sejam aplicadas diretamente aos nós de uma composição).

A Figura 4:4 ilustra uma composição XT-SMIL *par*, que referencia o *template audioComLegendasEnPt* da Figura 3:2 através do atributo *xtemplate*. Os tipos de seus elementos filho são declarados através do atributo *type*. Quando um documento contendo a composição da Figura 4:4 tem seus *templates* processados (o processamento de *templates* é apresentado em detalhes no próximo capítulo), o resultado é um documento contendo a composição XT-SMIL *par* da Figura 4:5. Note que uma composição XT-SMIL, quando tem seu *template* processado, torna-se uma composição válida, também, em SMIL.

```
01  <par id="coisaPele" xtemplate="audioComLegendasEnPt.xml">
02   <audio type="song" region="r0" id="samba" src="coisadepele.wav">
03    <area id="part1" type ="track" begin="8.4s" end="18s"/>
04    <area id="part2" type ="track" begin="18.5s" end="28s"/>
05    <area id="part3" type ="track" begin="29s" end="39s"/>
06   </audio>
07   <text type ="subtitleEn" region="r1" id="lyrics1a" src="versos01en.html"/>
08   <text type ="subtitlePt" region="r1" id="lyrics1b" src="versos01pt.html"/>
09   <text type ="subtitleEn" region="r1" id="lyrics2a" src="versos02en.html"/>
10   <text type ="subtitlePt" region="r1" id="lyrics2b" src="versos02pt.html"/>
11   <text type ="subtitleEn" region="r1" id="lyrics3a" src="versos03en.html"/>
```

```
12   <text type ="subtitlePt" region="r1" id="lyrics3b" src="versos03pt.html"/>
13  </par>
```

Figura 4:4 - Composição XT-SMIL *par* utilizando um *template*.

```
01  <par id="coisaPele">
02   <img id="logoResource0" src="logoTele.jpg"/>
03   <audio region="r0" id="samba" src="coisadepele.wav">
04    <area begin="8.4s" end="18s" id="part1"/>
05    <area begin="18.5s" end="28s" id="part2"/>
06    <area begin="29s" end="39s" id="part3"/>
07   </audio>
08   <seq>
09    <switch id="switch1">
10     <text   region="r1"   end="part1.end"   id="lyrics1a"   src="versos01en.html"
systemLanguage="en-us"/>
11     <text   region="r1"   end="part1.end"   id="lyrics1b"   src="versos01pt.html"
systemLanguage="pt-br"/>
12    </switch>
13    <switch id="switch2">
14     <text   region="r1"   end="part2.end"   id="lyrics2a"   src="versos02en.html"
systemLanguage="en-us"/>
15     <text   region="r1"   end="part2.end"   id="lyrics2b"   src="versos02pt.html"
systemLanguage="pt-br"/>
16    </switch>
17    <switch id="switch3">
18     <text   region="r1"   end="part3.end"   id="lyrics3a"   src="versos03en.html"
systemLanguage="en-us"/>
19     <text   region="r1"   end="part3.end"   id="lyrics3b"   src="versos03pt.html"
systemLanguage="pt-br"/>
20    </switch>
21   </seq>
22  </par>
```

Figura 4:5 - Resultado do processamento de *template* em uma composição XT-SMIL *par*.

## 4.2. SMIL + XConnector (XC-SMIL) e X-SMIL

SMIL 2.0 (W3C, 2001b) somente permite a especificação de elos ponto-a-ponto, que podem ser disparados por eventos temporais predefinidos pela linguagem (ver Capítulo 2). Ao permitir o uso de eventos na especificação temporal de um documento, SMIL 2.0 oferece, em relação à sua versão anterior (W3C, 1998b), uma maior flexibilidade para a autoria dos documentos. Entretanto, quando comparada à NCL, as possibilidades de especificação de elos em SMIL são limitadas. NCL oferece elos multiponto, representando relações com semântica causal ou de restrição, de acordo com o conector usado pelo elo.

Além disso, uma mesma relação causal, por exemplo, pode relacionar eventos de diversos tipos, além dos tradicionais eventos de seleção e apresentação, contemplados por SMIL (ver Capítulo 3).

O autor de documentos SMIL deve mesclar o uso das composições temporais (*par*, *seq* e *excl*) com o uso de elos para especificar relacionamentos que envolvem a ocorrência de vários tipos de evento. Em NCL, uma relação, por mais complexa que seja, é representada por um único conector[33] (Muchaluat-Saade, 2003); e elos NCL, simples ou complexos, são sempre especificados da mesma forma: referenciando conectores.

Para aumentar a expressividade e o reuso da linguagem SMIL, é proposta a extensão XC-SMIL, que introduz o conceito de conectores hipermídia àquela linguagem. A linguagem XC-SMIL é formada pela adição do módulo *Linking* de NCL à linguagem SMIL. Assim, essa extensão de SMIL adiciona o elemento *linkBase* aos elementos *body*, *par*, *seq* e *excl*, permitindo a definição de bases de elos. Elementos *linkBase*, assim como em NCL, possuem elementos filhos do tipo *link*, para definição de elos referenciando conectores. Cada *link* possui um conjunto de elementos *bind*, que relaciona papéis do conector a componentes do documento XC-SMIL (pelos atributos *role* e *component*).

A Figura 4:6 ilustra uma composição *par* em SMIL, com um elo relacionando a imagem *img1* com o áudio *audio1*. O atributo *end* declarado pela imagem determina que seu término deve coincidir com o valor desse atributo, ou seja, deve ser junto com o término da apresentação de *audio1* (evento de término de apresentação - "*end*" - do componente *audio1*: "*audio1.end*")

```
01. <par>
02.   <video id="video1" />
03.   <audio id="audio1" />
04.   <img id="img1" end="audio1.end" />
05. </par>
```

Figura 4:6 - Exemplo de elo em uma composição SMIL.

A Figura 4:7 ilustra a definição do elo "*link1*" em XC-SMIL, que referencia o conector "*finishes.xml*". Esse elo é semanticamente igual ao elo entre *audio1* e *img1* da Figura 4:6. Elos multiponto, em XC-SMIL, são especificados de forma semelhante ao "*link1*", como apresentados na Figura 4:8. Nessa figura, o elo

[33] Obviamente, algumas relações podem necessitar de mais de um conector para serem especificadas em XConnector, como relações que envolvem tanto causalidade quanto restrição.

"*link2*" determina que o término do áudio deve ocasionar o término da exibição tanto do vídeo quanto da imagem. Utilizando a mesma estrutura, mas redefinindo o conector sendo usado (assim como os papéis de cada componente), elos representando qualquer relacionamento entre o áudio, a imagem e o vídeo podem ser definidos em XC-SMIL. Isso também é *representado* na Figura 4:8 pelo elo fictício "*link3*" (referenciando o conector "*xxx.xml*") e pela associação dos papéis *a*, *b* e *c* do conector " *xxx.xml* " com os componentes da composição XC-SMIL.

```
01. <par>
02.   <video id="video1" />
03.   <audio id="audio1" />
04.   <img id="img1 />
05.   <linkBase>
06.     <link id="link1" xconnector="finishes.xml">
07.       <bind component="audio1" role="on_x_presentation_end"/>
08.       <bind component="img1" role="stop_y"/>
09.     </link>
10.   </linkBase>
11. </par>
```

Figura 4:7 - Exemplo de elo em uma composição XC-SMIL.

```
01. <par>
02.   <video id="video1" />
03.   <audio id="audio1" />
04.   <img id="img1 />
05.   <linkBase>
06.     <link id="link2" xconnector="finishes.xml">
07.       <bind component="audio1" role="on_x_presentation_end"/>
08.       <bind component="img1" role="stop_y"/>
09.       <bind component="video1" role="stop_y"/>
10.     </link>
11.     <link id="link3" xconnector="xxx.xml">
12.       <bind component="audio1" role="a"/>
13.       <bind component="img1" role="b"/>
14.       <bind component="video1" role="c"/>
15.     </link>
16.   </linkBase>
17. </par>
```

Figura 4:8 - Exemplo de elos multiponto em uma composição XC-SMIL.

Combinando-se o perfil XT-SMIL com XC-SMIL, obtém-se a linguagem X-SMIL. Nessa linguagem, são possíveis a definição de elos pelo reuso de conectores e o uso de *templates* de composição para especificar a semântica de uma composição. Ao permitir o uso de conectores, X-SMIL utiliza o perfil completo de XTemplate. Assim, as semânticas temporais obtidas pelo uso das composições *par*, *seq* e *excl* em SMIL podem, em X-SMIL, ser obtidas por

*templates* de composição que usam conectores para especificar essas semânticas. Por esse motivo, em X-SMIL, é desencorajado o uso das composições *par*, *seq* e *excl*, sendo sugerido o uso do elemento *composite* (adicionado à X-SMIL utilizando o módulo *BasicComposite* de NCL), que deve ser usado no lugar dessas composições. Para obter a semântica temporal oferecida anteriormente pelas composições SMIL, deve-se utilizar os *templates* pré-definidos por X-SMIL: *templatePar*, *templateSeq*, *templateExcl*[34]. Um exemplo de uma composição X-SMIL, similar à composição da Figura 4:7, é ilustrado na Figura 4:9.

Em X-SMIL, assim com em NCL 2.1, pode-se definir *templates* de composição utilizando todas as facilidades de XTemplate 2.1. Ou seja, é possível definir tanto relações de inclusão (como exemplificado na seção anterior) quanto relações por meio de conectores (como exemplificado no Capítulo 3). Compiladores para documentos nessas linguagens, assim como o processador de *templates* de XTemplate 2.1, serão descritos no próximo capítulo.

```
01. <composition xtemplate="templatePar" >
02.   <video id="video1" />
03.   <audio id="audio1" />
04.   <img id="img1 />
05.   <linkBase>
06.     <link id="link2" xconnector="finishes.xml">
07.       <bind component="audio1" role="on_x_presentation_end"/>
08.       <bind component="img1" role="stop_y"/>
09.     </link>
10.   </linkBase>
11. </composition>
```

Figura 4:9 - Exemplo de uma composição X-SMIL.

[34] Para refletir o uso do atributo *endsync* (W3C, 2001b), definido em SMIL para composições *par* e *excl*, também foram especificados os *templates templateParFirst*, *templateParLast*, *templateParAll*, *templateExclFirst*, *templateExclLast* e *templateExclAll*, que correspondem, respectivamente, aos valores *first*, *last* e *all* que esse atributo pode possuir. Assim, pela semântica definida por SMIL, os *templates templatePar* e *templateExcl* equivalem, respectivamente, aos t*emplates templateParLast* e *templateExclLast*.