3
Templates de Documentos

Este capítulo apresenta algumas abordagens para a autoria de famílias de documentos baseadas: em ADLs, na Seção 3.1; na linguagem *XTemplate*, Seção 3.2; e em *SMIL Timesheets*, Seção 3.3. Esse estudo de trabalhos relacionados visa fornecer novos requisitos, ainda não atendidos por tais abordagens, para uma linguagem voltada à autoria de famílias de documentos, os quais são apresentados na Seção 3.4. A Seção 3.5 fecha o capítulo com um resumo de tais requisitos, incluindo também os já apontados no Capítulo 2.

3.1.
Templates baseados em ADLs

Linguagens de descrição de arquitetura ou ADLs (Clements, 1996) permitem a programadores especificar a arquitetura de sistemas de *softwares* em vez de se preocuparem com detalhes de implementação. Há quatro conceitos principais em ADLs: componentes, conectores, configurações e estilos.

Componentes são unidades de computação. Conectores são unidades de comunicação, as quais associam componentes entre si. Configurações podem conter componentes, conectores e outras configurações recursivamente (definindo a estrutura de uma arquitetura de *software*). Estilos (como são chamados os templates em ADLs) definem um vocabulário de componentes e conectores de tal forma que restrições limitam a forma que tais componentes e conectores podem ser instanciados em uma configuração.

Com base na especificação de estilos de ADLs, os templates de composição para aplicações hipermídia devem permitir especificar três informações principais:

- Vocabulário: definindo tipos de componentes, interfaces e relações;
- Restrições: que especificam regras sobre os tipos definidos no vocabulário;
- Relacionamentos hipermídia entre tipos definidos no vocabulário

Entretanto, em documentos hipermídia o conceito de estilo deve ser estendido, como propõe (Muchaluat-Saade, 2003), acrescentando a definição de recursos e ampliando a definição de relacionamentos:

• Recursos: objetos (instâncias de componentes, interfaces ou relacionamentos e não tipos) não-editáveis;

• Relacionamentos hipermídia: que podem se dar entre tipos, como nos estilos das ADLs, mas também entre recursos ou entre tipos e recursos.

Esse conjunto de conceitos formula a proposta desta tese de *composição hipermídia em aberto*. A analogia de uma composição em aberto com os conceitos de ADLs é a de se comportar como sendo, ao mesmo tempo, um estilo e uma configuração. Composições em aberto definem não apenas um vocabulário de tipos e restrições na forma que tais tipos podem ser instanciados (o que em ADLs é expresso por estilos), mas também permite a definição de *recursos*, que são elementos existentes em todos os documentos baseados naquela composição (o que, por outro lado, são especificados nas configurações de ADLs).

Assim, uma composição hipermídia em aberto estabelece, entre outras coisas, um vocabulário, que permite expressar uma composição com alguns elementos internos propositalmente incompletos. Esse vocabulário é composto por tipos de componentes, tipos de interfaces, tipos de relações e recursos (elementos não-editáveis). A composição em aberto também possui restrições, as quais definem regras sobre como esses elementos em aberto podem ser especificados. Relacionamentos estabelecem sentenças causais entre elementos desse vocabulário, o que corresponde à semântica de apresentação da composição em aberto.

A Figura 12 ilustra um exemplo de composição hipermídia em aberto. Tal composição descreve um *slideshow* de fotos, exemplo clássico de família de documentos. Durante toda a apresentação desses documentos, um áudio de fundo é exibido e uma sequência de fotos vai sendo apresentada. Sempre quando uma foto termina sua apresentação, inicia a apresentação da foto seguinte.

Um *slideshow* de fotos pode ser modelado por uma composição em aberto como a seguir. Um tipo de componente “audio” é definido com a restrição que só pode haver uma única sua instância. Um tipo de componente “photo” é definido, com a restrição que deve haver pelo menos uma instância do tipo, mas não há limite superior para a quantidade dessas instâncias. Um recurso (interface)

estabelece que o ponto de entrada da composição é o “audio”, o que é representado na figura pelo retângulo na parte superior, com uma linha associada à primeira instância do componente “audio”. Três relacionamentos na composição em aberto: o primeiro deles estabelece que a exibição da instância única do componente “audio” faz disparar a primeira instância do componente “photo” (corresponde à seta da esquerda na figura); o segundo relacionamento define que o término do componente “audio” deve terminar todos os componentes “photo”, finalizando a apresentação do documento (corresponde à seta da direita da figura); o terceiro relacionamento define que o término de cada componente “photo” faz exibir o componente seguinte (corresponde à seta de baixo da figura).

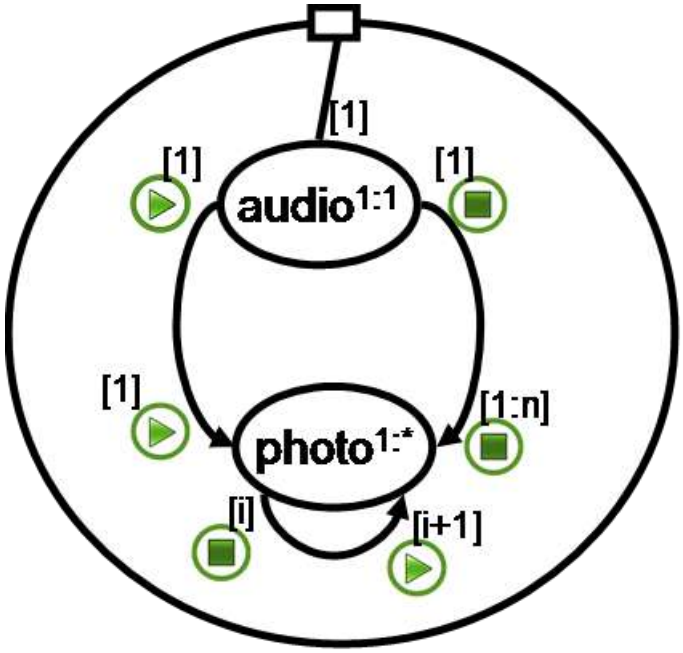

Figura 12. Composição hipermídia em aberto: um *slideshow* de fotos.

O exemplo, apesar de bem simples, ilustra uma limitação importante na expressividade de composições hipermídia nas linguagens declarativas para autoria hipermídia. Uma composição não poderia ser usada para fazer a modelagem de um objeto de mídia com um número variável de instâncias. Isso porque composições não são empregadas para descrever um documento incompleto. Já fazendo uso do conceito de composição hipermídia em aberto, o tipo de componente “photo” é usado para modelar essa necessidade.

3.2. Templates baseados em XTemplate

XTemplate (Muchaluat-Saade, 2003) é uma solução para definir templates de composição para a definição de relacionamentos entre objetos e tipos de objetos especificados em documentos NCL. XTemplate define estruturas genéricas de nós e elos que podem ser reusadas em diferentes cenários de uso de documentos NCL. XTemplate foi o primeiro a propor as extensões mencionadas na seção anterior para os estilos ADLs.

Em (Santos & Muchaluat-Saade, 2009) é descrita a versão mais recente de XTemplate (versão 3.0), uma proposta para a definição de templates para documentos escritos em NCL 3.0. Além de XTemplate 3.0 ser dirigida a uma linguagem hipermídia alvo específica, a linguagem foca bem mais criar facilidades para um autor mais especializado. Um usuário de XTemplate 3.0, mesmo aquele que apenas instancia templates de composição, precisa ter o conhecimento de certos pré-requisitos técnicos, como XPath (W3C, 1999a) e XSLT (W3C, 1999b).

A Figura 13 exemplifica um template de composição que poderia ser especificado em XTemplate. O template ilustra um áudio com vários intervalos de tempo marcados onde são exibidas legendas textuais. Uma imagem de um logotipo é sincronamente apresentada durante a exibição do áudio.

Usando dos conceitos de XTemplate, no template de composição da Figura 13, há três componentes (“logo”, “subtitle” e “audio”) e uma interface (“track”). O template restringe a “um” o número de instâncias dos componentes “logo” e “audio”, e ainda força a serem iguais os números de instâncias de “subtitle” e “track”. O template também define o relacionamento “L”, que associa o início do “audio” ao “logo” e mais o início de cada “track” a cada respectivo “subtitle”. Há ainda o relacionamento “P”, que sincroniza o término do “audio” ao “logo” e novamente os diversos “track” ao respectivo “subtitle”.

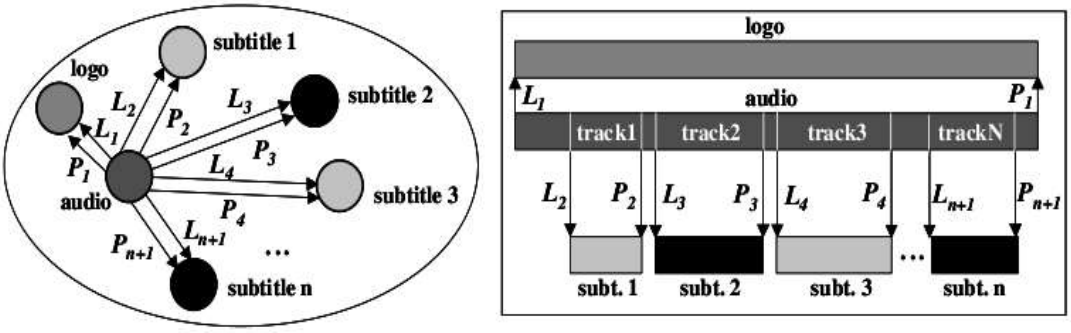

Figura 13. Visões Estrutural e Temporal de um Template de Composição (Santos & Muchaluat-Saade, 2009).

Um autor NCL pode usar a especificação de template de composição representada pela Figura 13 para criar bem mais facilmente documentos compostos por um áudio com legendas. Para isso, ele só precisa definir um documento NCL incompleto, onde especifica o objeto de mídia que representa o logotipo, o objeto de mídia com o áudio e com as âncoras de tempo e ainda um objeto de mídia textual representando uma legenda para cada uma dessas âncoras. Note que os vários relacionamentos de sincronismo expressos na Figura 13 pelas letras L e P são especificados não na composição NCL, mas em XTemplate, o que diminui o trabalho do autor. De posse desse documento incompleto e da especificação em XTemplate, o processador XTemplate gera um documento completo em NCL, pronto para apresentação.

No método proposto por esta tese, conforme apresentado no Capítulo 5, esse documento incompleto em NCL é chamado de *documento de preenchimento*. Um documento de preenchimento é especificado pelo autor de documentos para definir as lacunas as quais tornam um documento único em uma família. No caso de XTemplate, o documento de preenchimento só pode ser especificado em NCL.

Um ponto crítico no emprego de XTemplate para a criação de documentos em NCL é que não há um mecanismo facilitador para o autor de documentos (e não o autor do template) poder identificar a semântica do template e, assim, os objetos e relacionamentos que precisa, ou pode, especificar no documento de preenchimento. O autor NCL precisa entender toda a especificação do template de composição em XTemplate para poder usá-la. Isso implica a necessidade desse autor conhecer XPath e XSLT. XPath é usada para navegar em elementos e atributos em um documento XML. XSLT é empregada para definir transformações de formato em documentos XML. Ambas as tarefas para as quais

XPath e XSLT foram projetadas são incidentalmente relacionadas à autoria XML e tais tecnologias não foram projetadas no nível do domínio de problema em questão. Esse ponto em particular é crítico porque restringe o tipo de usuário da linguagem para um perfil mais especializado.

A Figura 14, com código extraído de (Santos & Muchaluat-Saade, 2009), ilustra como são especificados os relacionamentos "L" e "P", presentes no template de composição já descrito. O elemento <for-each> itera, conforme especificado por seu atributo *select*, sobre os elementos <area> do documento cujo valor do atributo *xlabel* seja igual a "track" e que sejam filhos de um elemento <media> com atributo *xlabel* igual a "audio". É importante salientar que essa iteração não é definida sobre os tipos do vocabulário, mas sobre uma nova expressão XPath, que pode não ter relação alguma com o vocabulário de tipos previamente definido. O primeiro elemento <link> da Figura 14 especifica o relacionamento "L", em que, para cada um desses i-ésimos elementos <area>, é encontrado no documento o i-ésimo elemento <media> com atributo *xlabel* igual a "subtitle" e criado um elo entre ambos, associando o início da interface "track" ao início do componente "subtitle". O relacionamento "P" é parecido, com a diferença de expressar o término de cada "subtitle" com base no término de cada "track", e corresponde ao segundo elemento <link> da Figura 14.

```
<for-each select="child::media[@xlabel='audio']/
                  child::area[@xlabel='track']">

  <link xtype="L">
    <bind role="onBegin" select="current()"/>
    <bind role="start"
     select="child::media[@xlabel='subtitle'][position() = $i]"/>
  </link>

  <link xtype="P">
    <bind role="onEnd" select="current()"/>
    <bind role="stop"
     select="child::media[@xlabel='subtitle'][position() = $i]"/>
  </link>

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

</for-each>
```

Figura 14. Relacionamentos em XTemplate.

A grosso modo, pode se dizer que XTemplate não representa apenas a especificação de um template propriamente dito, mas de um gerador de documentos NCL que, conjuntamente, comporão uma família.

3.3. Templates em SMIL Timesheets

SMIL Timesheets (W3C, 2008c) (ou Timesheets de agora em diante) é uma contraparte temporal ao CSS (Lie & Bos, 1997), ambos do W3C, que tem por objetivo permitir a uma linguagem XML qualquer incorporar os elementos e atributos do controle temporal de SMIL (W3C, 2008). Timesheets especificam que elementos são ativos em um dado momento e qual seu escopo temporal dentro de um documento. O principal objetivo de Timesheets é permitir a incorporação de aspectos temporais em documentos escritos em linguagens atemporais. É importante salientar que esse objetivo muito se assemelha à necessidade de especificar em separado a semântica temporal de composições para que depois ela possa ser incorporada na especificação de documentos em linguagens baseadas no tempo.

Tomando como caso de uso, um documento HTML (W3C, 2002) poderia ser enriquecido com a especificação de sincronismo em Timesheets. Isso daria ao documento HTML não apenas a capacidade já existente de formatação hipertexto, mas também o sincronismo espaço-temporal.

Há quatro conceitos básicos por trás de Timesheets: a temporização e sincronismo; o mecanismo de seleção; o modelo de eventos; e a funcionalidade de indexação. A temporização e sincronismo é herdada da especificação homônima de SMIL. Os elementos da linguagem hospedeira podem estar ativos ou inativos, de acordo com a semântica SMIL dos contêineres temporais <seq>, <par> e <excl>. Um elemento não referenciado por um código Timesheets estará sempre ativo por omissão. Em simplificação, os atributos *begin*, *dur* e *end* são os três principais utilizados para essa temporização e sincronismo. O atributo *begin* define o instante de tempo em que aquele elemento deve ser exibido na linha temporal do elemento pai. O atributo *dur* define a duração daquele elemento. O atributo *end* define o instante de término para o elemento na linha de tempo do pai. Há, obviamente, outras formas de expressar o sincronismo e que são propositalmente omitidas por simplificação.

O segundo conceito básico de Timesheets é o mecanismo de seleção. Uma especificação Timesheets faz referência a elementos na linguagem hospedeira por

meio dos seletores CSS (Lie & Bos, 1997). Os elementos podem ser selecionados pelo tipo do elemento, valor de um atributo, pela existência de um determinado atributo, entre outros exemplos. A mesma abordagem é adotada na proposta de linguagem desta tese e maiores detalhes podem ser vistos na Seção 4.4.

O terceiro conceito básico é o modelo de eventos. Eventos são divididos em dois grupos: internos e do usuário. Os internos são aqueles disparados de dentro dos elementos na especificação Timesheets e que podem ser usados para construir relacionamentos entre diferentes partes da linha de tempo. Os eventos de usuário são disparados pela ação dos usuários. Na prática, os atributos *begin, end* e *dur* podem conter eventos DOM (W3C, 2000), os quais, por exemplo, podem estar relacionados ao início da exibição de um elemento ou à interação por parte do usuário.

O quarto conceito básico é a funcionalidade de indexação. Quando múltiplos elementos do documento hospedeiro forem selecionados por um seletor CSS, o resultado é uma lista ordenada de elementos, os quais são dispostos na mesma ordem em que aparecem esses elementos no documento hospedeiro. A indexação se faz necessária para identificar, nessa lista ordenada, qual elemento se quer referir.

A Figura 15 apresenta um exemplo de uso de SMIL Timesheets para a elaboração de uma exibição sequenciada de elementos (*slideshow*), o qual é extraído de (W3C, 2008c). No exemplo, a linguagem hospedeira é HTML. O elemento <smil:timesheet> contém a especificação temporal que se quer aplicar. Tal especificação é formada apenas por um contêiner temporal <seq>, que estabelece a exibição em sequência de seus três elementos filhos <smil:item>, todos com duração de 5 segundos (atributo *dur*). O atributo *select* de cada um dos três elementos <smil:item> identifica quais elementos da linguagem hospedeira são selecionados. No exemplo, são selecionados respectivamente aqueles identificados pelos valores “Slide1”, “Slide2” e “Slide3”, que são os elementos do documento (cujo conteúdo é omitido na figura sem prejuízo de entendimento). O resultado visual e temporal é que, do início da exibição do documento até o instante 5 segundos, é exibido o elemento com identificador “Slide1”. Dos 5 segundos aos 10 segundos, é exibido o “Slide2” e dos 10 segundos aos 15 segundos é exibido o “Slide3”.

```
<?xml version="1.0"?>
<html xmlns="http://www.w3.org/1999/xhtml"
         xmlns:smil="http://www.w3.org/ns/SMIL30">

  <head>
    <title>Timesheet Example</title>
    <smil:timesheet>
      <smil:seq>
        <smil:item select="#Slide1" dur="5s" />
        <smil:item select="#Slide2" dur="5s" />
        <smil:item select="#Slide3" dur="5s" />
      </siml:seq>
    </smil:timesheet>
  </head>

  <body>
    <div xml:id ="Slide1" class="Slide">
      (...)
    </div>
    <div xml:id ="Slide2" class="Slide">
      (...)
    </div>
    <div xml:id ="Slide3" class="Slide">
      (...)
    </div>
  </body>

</html>
```

Figura 15. Slide-show de fotos em SMIL Timesheets (W3C, 2008c).

Uma limitação de Timesheets é explicitada pelo exemplo de exibição sequenciada (*slideshow*) da Figura 15. O exemplo ilustra uma apresentação com três elementos . Nota-se que não é possível deixar em aberto o número de itens que compõem o container temporal <seq>. A especificação *Timesheets* não poderia ser usada para especificar uma quantidade indefinida de elementos , mesmo com a indexação, e cada elemento selecionado precisa ser univocamente identificado.

A ausência de mecanismos para representar o equivalente às restrições presentes em ADLs é outro aspecto que limita a expressividade em *Timesheets*. Mesmo em *Timesheets* tornando explícita a semântica espacial e temporal de documentos, essas limitações de expressividade (como a ausência de tipos e falta de suporte à definição de restrições), parecem indicar que *Timesheets* é voltada para arquétipos de documentos e é pobre para caracterizar famílias de documentos. Conforme previamente definido, arquétipos apenas definem moldes de onde parte a definição de documentos, porém, como esses documentos podem ser alterados sem restrições, eles não caracterizam uma família de documentos.

3.4. Requisitos para uma Linguagem de Especificação de Famílias de Documentos Vindos de outras Linguagens

O estudo de linguagens voltadas direta ou indiretamente para a representação de famílias de documentos indica a inexistência de uma linguagem que atenda às necessidades de especificação apontadas por esta tese. O estudo motiva o levantamento de alguns requisitos para a especificação de uma nova linguagem para esse fim.

Um primeiro requisito é a necessidade de haver um modelo conceitual para a especificação de famílias de documentos. Essa necessidade fica clara na análise dos trabalhos relacionados. ADLs fornecem essa base conceitual para a definição de composições hipermídia em aberto. XTemplate se apropria desses conceitos oriundos de ADLs, os estendendo, na definição de seus conceitos próprios, como os templates de composição. *Timesheets* faz uso da temporização e sincronismo SMIL, baseada nos contêineres temporais. Um modelo conceitual para famílias de documentos precisa dar suporte à especificação de composições com partes propositalmente deixadas em aberto e partes pré-definidas (ou não-editáveis). Essas partes em aberto permitem que autores possam instanciar composições preenchendo apenas suas lacunas (no chamado documento de preenchimento) e, assim, tornando seu documento único, obedecendo a restrições específicas na forma de completá-las. As partes pré-definidas forçam uma estrutura básica e seriam úteis para isoladamente definir arquétipos de documento caso não houvesse os demais conceitos. *Timesheets* e XTemplate atendem tal requisito.

Outro requisito para uma linguagem de especificação de famílias de documentos é que ela não exija conhecimento externo a esse domínio em que reside seu modelo conceitual. Não é bem-vindo que seu usuário precise conhecer uma notação completamente alheia à tarefa central que é fazer a modelagem da família de documentos. Com o propósito de servir de ilustração, esse requisito não é atendido por XTemplate, uma vez que há a necessidade do autor saber sobre transformações XSLT e navegação por meio de XPath para usar templates de composição. A justificativa para esse requisito é tornar mais simples a tarefa de autores interessados apenas em instanciar documentos, sem nunca desejarem especificar famílias. *Timesheets* atende a esse requisito.

Em havendo um requisito objetivando simplificar a tarefa de autores interessados apenas em instanciar famílias de documentos, é conveniente complementá-lo com o requisito de não oferecer diminuição em termos de expressividade para usuários mais experientes. Esse requisito, de uma forma geral, objetiva garantir que a simplificação da autoria de famílias de documentos não implique perda de sua aplicabilidade. ADLs preenchem esse requisito.

O modelo conceitual deve permitir a definição de restrições e também a possibilidade de seus tipos endereçarem um número não limitado de elementos. XTemplate preenche esse requisito.

A observação da simplicidade de especificações *Timesheets* na tarefa de preparar arquétipos (ou moldes) de documentos motiva outro requisito, que é a necessidade de especificações de famílias simples não herdarem inconvenientes destinados a especificações complexas. Em outros termos, é necessário que famílias de documentos simples também sejam fáceis e diretas de especificar. *Timesheet* preenche esse requisito.

3.5. Resumo dos requisitos de uma linguagem para autoria de famílias de documentos

Em resumo, os requisitos apontados para uma linguagem voltada à autoria de famílias de documentos, baseados nas considerações levantadas neste capítulo e no Capítulo 2 são:

- precisa ter méritos de usabilidade (ser fácil de usar);
- seu emprego deve ser mais indicado que o uso direto de linguagens para autoria de documentos hipermídia sempre que uma estrutura de repetição for identificada (uma família de documentos é caracterizada);
- autores que criam famílias de documentos podem ser mais especializados, mas autores que apenas usam famílias para conceberem documentos devem precisar conhecer menos requisitos técnicos;
- o reúso precisa ser promovido naturalmente, no nível de domínio da linguagem (não deve ser mais difícil criar famílias apenas porque são mais facilmente reusáveis);
- o conceito de composição hipermídia deve ser revisitado para atender às necessidades de famílias de documentos;

- o reúso precisa ser promovido, não forçado;
- o projeto da linguagem precisa ser pensado e refletir o foco no reúso;
- precisa ser baseada em um modelo conceitual que permita especificar formalmente famílias de documentos;
- a especificação de famílias não pode requisitar pré-requisitos técnicos externos ao domínio de seu modelo conceitual como forma de tornar possível autores menos especializados apenas instanciarem famílias de documentos;
- a expressividade do autor de famílias de documentos não pode ser comprometida em virtude de simplificar o processo de autoria por parte de autores menos especializados;
- famílias de documentos simples também devem ser facilmente especificadas. A modelagem não pode embutir dificuldades para quando poucos recursos forem necessários.