4
TAL: Template Authoring Language

Template Authoring Language (TAL) é uma linguagem modular XML. Um caso de uso simples da linguagem, utilizado como exemplo em todo o capítulo, é descrito pelo template "Botão-Texto-Imagem" na Seção 4.1, o qual é completamente especificado em TAL nas seções seguintes. A Seção 4.2 apresenta os conceitos gerais da linguagem. A Seção 4.3 descreve os elementos e atributos de TAL. Na Seção 4.4 é apresentada a sintaxe para a definição dos seletores da linguagem. Na Seção 4.5, é discutida a sintaxe para descrever restrições. A Seção 4.6 apresenta a especificação proposta para relacionamentos. Na Seção 4.7 são apresentados outros exemplos de templates especificados em TAL.

4.1.
Exemplo de Uso: Template Botão-Texto-Imagem

A Figura 16, a seguir, ilustra dois exemplos de aplicações interativas para TV Digital, oriundas de duas instituições diferentes, que seguem o template proposto de exemplo, chamado "Botão-Texto-Imagem" neste trabalho. Em ambas as aplicações, há um menu de botões. Quando um desses botões é selecionado (ou ganha o foco) usando as teclas direcionais do controle remoto, um texto e uma imagem respectivos são apresentados, desaparecendo com o texto e imagem que estava sendo exibido anteriormente. O primeiro botão faz aparecer um primeiro texto e imagem respectivos. O segundo botão faz aparecer outro texto e imagem respectivos e assim por diante, para todos os botões. Os textos e imagens são exibidos sempre na mesma região da tela. Esse padrão de interação é comum em aplicações interativas para TV digital, conforme pode ser visto em algumas submissões do Clube NCL (Clube NCL, 2010), um repositório público de aplicações para o Middleware Ginga-NCL. As telas das aplicações descritas na Figura 16 foram extraídas desse repositório.

(a) Aplicação sobre Saúde da PRODERJ.

(b) Aplicação de Serviços da CAIXA®.

Figura 16. Exemplos de aplicações que seguem o template Botão-Texto-Imagem: (a) Aplicação sobre saúde da PRODERJ; (b) Aplicação de Serviços da CAIXA®, desenvolvida pela HxD Interactive Television.

É direto, mas muito trabalhoso, modelar essas aplicações como um contexto (composição hipermídia) em NCL. A Figura 17 apresenta a visão estrutural de um desses documentos, focando apenas na parte do documento relacionada ao template "Botão-Texto-Imagem". Conforme foi dito anteriormente, não é possível em NCL, como em nenhuma outra linguagem declarativa hipermídia, descrever um número variável dessas mídias em um contexto. É por essa razão que a Figura 17 exemplifica um documento dessa família que tem três botões (cujos identificadores são "botao1", "botao2" e "botao3"), três textos ("texto1", "texto2" e "texto3") e três imagens ("img1", "img2" e "img3"). Note como é trabalhoso e repetitivo o trabalho, especialmente quando o número de botões cresce.

Na Figura 17, três elos dão a semântica de navegação pelos botões no documento. Todo elo em NCL é uma relação de causa e efeito, de forma que uma ação é realizada quando uma condição é satisfeita. O primeiro elo tem como

condição o “botao1” ser selecionado, e como ação resultante, parar a exibição de todos os textos e imagens (fazendo desaparecer aqueles que estavam sendo exibidos anteriormente) e iniciar a exibição da “img1” e “texto1”. Os outros dois elos são similares, com a diferença que o segundo elo tem como condição a seleção do “botao2” e como ação resultante iniciar a “img2” e o “texto2”. O terceiro elo, por sua vez, tem como condição a seleção do “botao3” e como ação resultante iniciar a “img3” e o “texto3”.

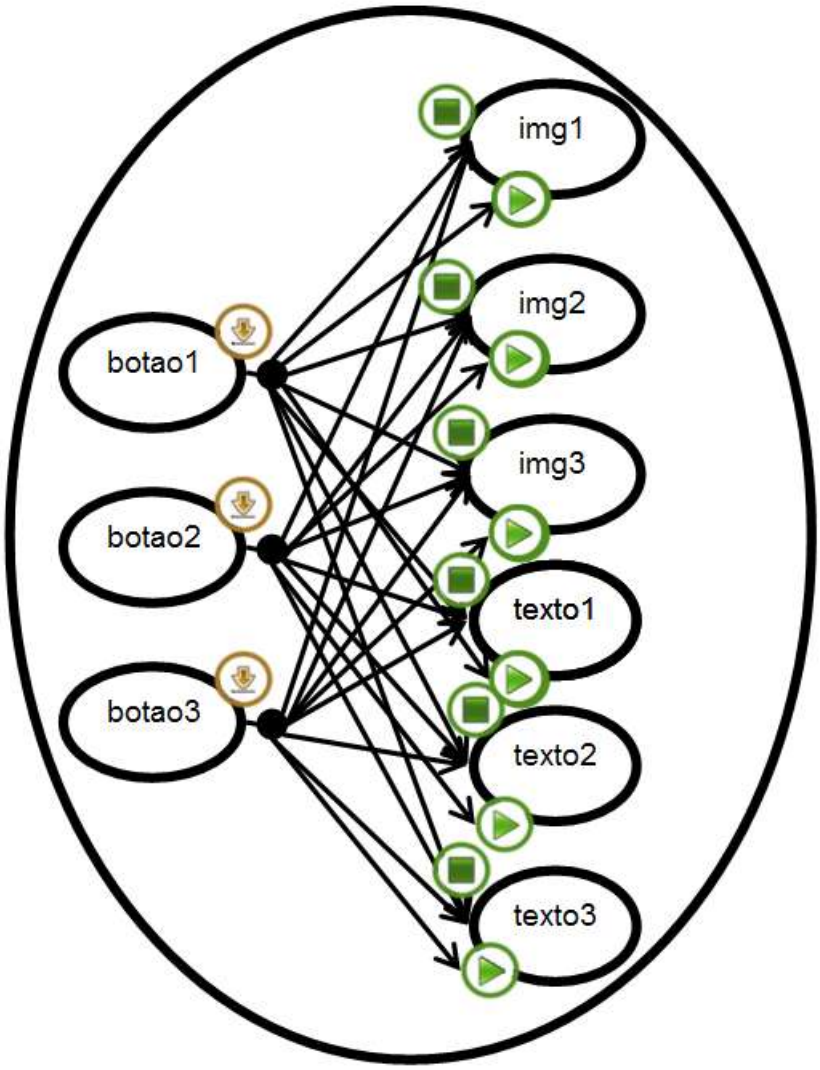

Figura 17. Visão Estrutural de um Documento seguindo o Template “Botão-Texto-Imagem”.

4.2.
Conceitos Gerais de TAL

Conforme previamente descrito, um template em TAL pode ser visto como a especificação de uma composição em aberto. Uma composição em aberto é definida por quatro elementos principais:

• Vocabulário: definindo tipos de seus componentes, interfaces e relações;

• Restrições: que especificam regras sobre os tipos definidos no vocabulário;

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

• Relacionamentos hipermídia: entre tipos, ou entre recursos, ou entre tipos e recursos.

Na especificação do vocabulário, componentes são a representação de conjuntos de objetos de mídia ou de outras composições internas aos documentos finais. Interfaces são a representação de conjuntos de âncoras em objetos. Âncoras podem ser de conteúdo, representando parte do conteúdo de um objeto de mídia, ou uma propriedade (atributo) de um objeto de mídia ou de uma composição. Relações são a representação de seus homônimos (habitualmente de causalidade) em documentos finais. Ao definir o vocabulário, também se está definindo uma hierarquia imposta aos componentes e interfaces, de forma que um componente (representando uma composição) pode conter recursivamente outros componentes e outras interfaces. O vocabulário permite que um template seja visto como uma composição com a quantidade de elementos internos em aberto, o que aumenta a expressividade na especificação dessa composição.

Restrições explicitam regras relativas à cardinalidade dos tipos de componentes, interfaces e relações. Restrições podem também definir expressões algébricas correlacionando a cardinalidade de tipos diversos. Restrições são utilizadas para verificar a correção do processo de instanciação de templates. Isso porque erros no preenchimento das lacunas de templates podem tornar as restrições inválidas.

Recursos são a parte imutável do template, separando claramente a parte editável da não-editável. Um autor poderia, por exemplo, especificar uma família de documentos em que um logotipo sempre está presente (como um recurso), sem

que um autor, usando aquele template, precise se preocupar em incluir essa informação ao instanciar documentos.

A especificação de relacionamentos hipermídia dá a uma composição sua semântica. Como a especificação dos elementos internos ao template pode ser deixada em aberto, com o uso de tipos definidos no vocabulário, os relacionamentos podem se dar também entre tipos e não apenas entre âncoras de objetos instanciados.

Como um tipo representa um conjunto de objetos ou âncoras no documento final, relacionamentos para tipos precisam também especificar como iterar sobre os elementos desse conjunto. Como exemplo, pode ser preciso criar um relacionamento hipermídia para cada elemento de um tipo, correlacionando-o com todos os elementos de outro tipo. Outras formas de correlações são possíveis, como a criação de um único relacionamento do primeiro elemento encontrado de um tipo para todos os elementos de outro tipo.

Relacionamentos também podem ser criados entre recursos. Isso permite, por exemplo, definir uma composição incompleta com alguma semântica de apresentação não-editável. Como exemplo, um template poderia não apenas definir um logotipo como um recurso, mas também que sempre que aquela composição for iniciada, ele é exibido até que a composição termine sua exibição.

Voltando ao exemplo do template “Botão-Texto-Imagem” ele poderia ser modelado em TAL de acordo com a Figura 18. Três tipos de componentes são definidos: “button”, “text” e “picture”. As restrições na cardinalidade desses tipos são as seguintes: deve haver pelo menos uma instância de cada um dos três tipos e, ainda, a cardinalidade de “text” e de “picture” devem ser ambas iguais à de “button”.

Os documentos que seguem esse template exemplo devem iniciar apresentando todos os botões e um primeiro texto e imagem correspondendo ao botão que começa com foco. Isso exige que se definam recursos para expressar esse comportamento inicial na composição. Em NCL, esse comportamento inicial pode ser obtido colocando-se uma porta para todos os botões e mais uma porta para o primeiro texto e imagem. Portas em NCL são pontos de interface em contextos que expõem âncoras de elementos internos à composição. Além disso, quando se dá início a um contexto, o resultado é equivalente a dar início à exibição de todos os seus elementos internos apontados por portas. A Figura 18

não exibe esses recursos por razões didáticas, mas, como é visto na seção seguinte, eles podem ser expressos em TAL.

Ainda na Figura 18, há um relacionamento hipermídia principal definido entre os tipos “button”, “text” e “picture”. Esse relacionamento especifica que, para cada instância do tipo “button”, sua seleção deve ter como ação resultante tanto o término da exibição de todas as instâncias dos tipos “text” e “picture” quanto o início da exibição da respectiva instância de “text” e “picture” (relacionada à instância de “button” selecionada).

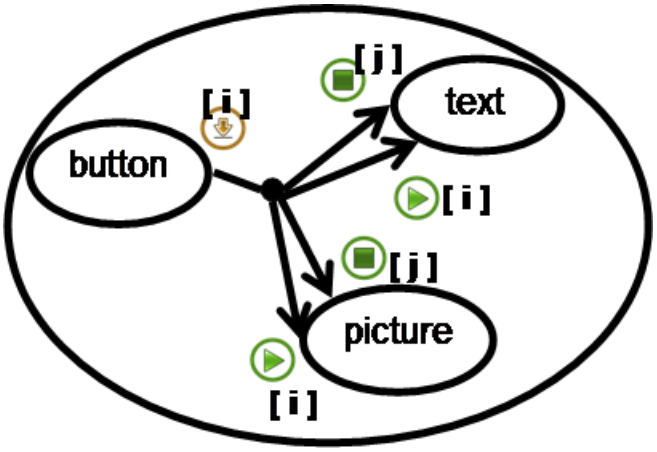

Figura 18. Template “Botão-Texto-Imagem”.

De posse dessa especificação de template, um autor NCL pode criar um documento de preenchimento de forma mais simples e rápida. O mesmo exemplo de três botões, três imagens e três textos, que aparece na Figura 17, é representado na Figura 19, mas com o uso do template, na linguagem NCL. Note que não é necessário especificar nem os elos nem as interfaces do contexto, as quais são informação extraída pelo processador de templates e aparece apenas no documento final gerado por ele. Note também que o template a ser usado é referenciado pelo atributo template do elemento <context>; como é esclarecido na Seção 4.3.

```
<context id="meuMenu" template="templates.xml#ButtonTextPicture">
   <media id="botao1"  ... class="button"/>
   <media id="texto1"  ... class="text"/>
   <media id="img1"    ... class="picture"/>
   <media id="botao2"  ... class="button"/>
   <media id="texto2"  ... class="text"/>
   <media id="img2"    ... class="picture"/>
```

```
   <media id="botao3"  ... class="button"/>
   <media id="texto3"  ... class="text"/>
   <media id="img3"    ... class="picture"/>
</context>
```

Figura 19. Documento NCL de preenchimento do template “Botão-Texto-Imagem”.

4.3. Definição de TAL

A linguagem TAL é composta por oito módulos. O módulo *Classification* é aquele usado apenas no documento de preenchimento e corresponde à extensão ao perfil da linguagem de preenchimento dada pelos atributos *class* (conforme discutido na Seção 4.4) e *template* (que referencia o documento TAL associado). O módulo *LanguageExtension* também é definido só para uso pela linguagem de preenchimento e faz referência aos módulos *TemplateBase* e *Template*, definidos adiante nesta seção. O módulo *LanguageExtension* permite a definição da base de templates (por meio do elemento <templateBase>) usada pelo documento de preenchimento. Elementos <importTAL>, filhos de <templateBase> permitem a importação dos templates que poderão ser usados. Nesse módulo, o elemento <template> pode ser incluso como filho do elemento <templateBase>, motivo que diferencia esse módulo do *Structure*, definido como segue.

A Tabela 2 sumariza a estrutura hierárquica dos elementos e atributos dos demais módulos, que são usados pela linguagem TAL no documento de definição de templates. Os nomes de atributos que aparecem sublinhados na tabela são obrigatórios. A notação para os filhos de elementos é a seguinte: o símbolo | denota uma lista de opções; o símbolo + denota a exigência de pelo menos um elemento filho; e o símbolo * denota a exigência por zero ou mais elementos filhos. Em alguns elementos, seu conteúdo textual (*Cdata*) é uma mensagem de erro ou ainda especificações de relacionamentos ou relações, conforme será explicado no que se segue.

Tabela 2. Elementos e atributos da linguagem TAL.

Elemento	**Atributos**	**Filhos**
tal	id	templateBase (template)+
templateBase		(importTal)+
importTAL	documentURI, alias	
template	id, extends	(component | interface | relation | assert | report | warning | link)*
component	id, selects, template	(component | interface)*
interface	id, selects	
relation	id, selects	Cdata=relation specification
assert	test	Cdata=message
report	test	Cdata=message
warning	test	Cdata=message
link	id	Cdata=link specification (forEach)*
forEach	instance, iterator, step	Cdata=link specification (forEach)*

O módulo *Structure* define o elemento <tal> raiz da linguagem, que tem um identificador obrigatório (atributo *id*), e corresponde aos templates passíveis de serem referenciados por um documento de preenchimento.

O módulo *TemplateBase* define o elemento <templateBase>, seu elemento filho <importTAL> e atributos. O elemento <importTAL> permite importar templates de outros documentos para um novo documento TAL. Nesse caso, a URI do documento TAL importado é definida pelo atributo *documentURI* com um nome interno usado no documento importador definido pelo atributo *alias*. Templates importados para a base de templates de um documento podem também ser usados para a extensão de novos templates.

O módulo *Template* define o elemento homônimo e seus atributos. Um template é especificado pelo elemento <template>, novamente com um identificador único (*id*) em um documento TAL. O atributo *extends* pode ser

inserido em um elemento <template> para que ele se comporte como uma extensão de um template já existente. Nesse caso, o template herda todo o vocabulário, restrições, recursos e relacionamentos do template base.

O módulo *Vocabulary* especifica o vocabulário de tipos de um template, por meio dos elementos <component>, <interface> e <relation>, e seus atributos. Tipos de componentes são expressos em TAL pelo elemento <component>, que também possui o atributo *id*. O atributo *selects* de <component> indica o tipo a ser aplicado nos elementos instâncias daquele componente. Tais elementos instâncias devem ser descendentes do elemento onde o <template> é aplicado. A sintaxe para valores possíveis do atributo *selects* é detalhada na Seção 4.4. O atributo *template* do elemento <component> permite que uma composição presente em um template possa ser definida em aberto, como seguindo outro template, o que é exemplificado na Seção 4.7.2.

Tipos de interface são definidos em TAL pelo elemento <interface>, novamente com os atributos *id* e *selects*.

Um tipo de relação é especificado pelo elemento <relation>, que também possui os atributos *id* e *selects*. O conteúdo do elemento informa uma sentença que descreve uma relação hipermídia baseada no paradigma de causalidade ou restrição. A definição de tipos de relações em um template permite, entre outras coisas, que restrições sejam impostas na cardinalidade dessas relações no documento final. A sintaxe para relacionamentos é descrita em maiores detalhes na Seção 4.6.

O módulo *Constraint* especifica as restrições de um template e inclui os elementos <assert>, <report> e <warning>, e seus atributos. Restrições são descritas em TAL baseando-se, parcialmente, na forma como regras são descritas em Schematron (ISO, 2006). Nos três elementos, o atributo *test* especifica o teste lógico que deve ser efetuado na restrição. A sintaxe para esses testes lógicos é definida em maiores detalhes na Seção 4.5. Ainda nos três elementos, a mensagem de erro ou aviso é extraída do conteúdo dos elementos. O elemento <assert> exige que seu teste lógico seja avaliado como verdadeiro, caso contrário o processador exibe a mensagem daquele elemento. O elemento <report> é similar, com a diferença que o teste lógico deve ser avaliado como falso, caso contrário a mensagem respectiva é indicada. Enfim, o elemento <warning> informa a mensagem de aviso quando o valor do teste lógico for avaliado como verdadeiro.

A ocorrência de uma mensagem de erro (<report> ou <assert>) para a geração do documento final pelo processador de templates. Por outro lado, a existência de mensagens de aviso (<warning>) não impede a geração do documento final.

O módulo *Relationship* permite especificar relacionamentos hipermídia em TAL e é composto pelos elementos <link> e <forEach>, e seus atributos. Relacionamentos hipermídia são definidos no template por meio do elemento <link>, que possui um atributo opcional *id*. Como um relacionamento pode se dar entre tipos, elementos filhos <forEach> podem ser usados para iterar sobre as instâncias desses tipos. Maiores detalhes sobre a especificação de relacionamentos são dados na Seção 4.6.

Recursos podem ser escritos diretamente no template, bastando para isso não utilizar o *namespace* padrão da linguagem (TAL) naquele elemento. O processador de templates TAL interpreta cada elemento fora de seu *namespace* como um que deve ser incluído no documento final gerado. O exemplo a seguir ilustra o conceito. É importante salientar que a definição de recursos pode particularizar uma especificação TAL para uma linguagem alvo específica, já que o processador TAL apenas repete os recursos no documento final, sem nenhuma transformação, o que é exemplificado como se segue.

Retomando o exemplo do template "Botão-Texto-Imagem", a Figura 20 apresenta sua especificação completa em TAL. Por questões didáticas, a explicação sobre alguns valores de atributos é feita apenas nas subseções seguintes. Nesta seção é explicada apenas a estrutura geral do template.


```
1:  <?xml version="1.0" encoding="UTF-8"?>
2:  <tal:tal id="templateExample">
3:  <tal:template id="ButtonTextPicture">
4:    <port id="pButton" component="button[1]"/>
5:    <port id="pText" component="text[1]"/>
6:    <port id="pPicture" component="picture[1]"/>
7:    <tal:component id="button" selects="media[class=button]"/>
8:    <tal:component id="text" selects="media[class=text]"/>
9:    <tal:component id="picture" selects="media[class=picture]"/>
10:   <tal:link id="startAllButtons">
11:      onBegin button[1] then
12:      <tal:forEach instance="button" iterator="i">
13:         start button[i+1],
14:      </tal:forEach>
15:      end
16:   </tal:link>
17:   <tal:link id="buttonSelection">
18:      <tal:forEach instance="button" iterator="i">
19:         onSelection button[i] then
```

```
20:         <tal:forEach instance="text" iterator="j">
21:            stop text[j], stop picture[j],
22:         </tal:forEach>
23:         start text[i], start picture[i] end
24:      </tal:forEach>
25:   </tal:link>
26:   <tal:assert test="#button>0">
27:      There must be at least one BUTTON element.
28:   </tal:assert>
29:   <tal:assert test="#text>0">
30:      There must be at least one TEXT element.
31:   </tal:assert>
32:   <tal:assert test="#picture>0">
33:      There must be at least one PICTURE element.
34:   </tal:assert>
35:   <tal:assert test="#button==#text">
36:      The cardinality of BUTTON and TEXT must be the same.
37:   </tal:assert>
38:   <tal:assert test="#button==#picture">
39:      The cardinality of BUTTON and PICTURE must be the same.
40:   </tal:assert>
41: </tal:template>
42: </tal:tal>
```

Figura 20. Especificação do template “Botão-Texto-Imagem”.

A linha 1 da Figura 20 é o cabeçalho padrão XML do arquivo. Nas linhas 2 a 42 é definida a base de templates daquele documento, que possui apenas o template “ButtonTextPicture” (linhas 3 a 41). Esse template se aplica apenas aos elementos no documento de preenchimento cujo valor do atributo *template* é igual ao seu id “ButtonTextPictue”.

As linhas 4, 5 e 6 apresentam recursos que devem ser inseridos no documento final gerado depois do processamento do template. Em um template concebido para o uso pela linguagem NCL, as três portas definem os pontos de entrada da composição em NCL onde o template será aplicado. Elas estão relacionadas à primeira instância do componente “button” (atributo *component* igual a “button[1]”), primeira instância do componente “text” (atributo *component* igual a “text[1]”) e ainda a primeira instância do componente “picture” (atributo *component* igual a “picture[1]”). Note que o emprego de recursos pode particularizar o template para uso por uma certa linguagem alvo.

As linhas 7, 8 e 9 definem os tipos de componentes do template. O tipo de componente “button” seleciona todos os nós de mídia cujo valor de atributo *class* seja igual a “button” e representa os botões que são exibidos no template. De forma similar são definidos os tipos de componente “text” (para os respectivos textos) e “picture” (para as imagens ilustrativas).

Para efeito de indexação, o primeiro elemento encontrado no documento de preenchimento que atender ao seletor “media[class=button]” do componente “button” é identificado como button[1]. O segundo elemento encontrado é identificado como button[2] e assim por diante. Tomando como exemplo o documento de preenchimento da Figura 19, button[1] é o elemento de id “botao1” (primeiro que atende ao seletor), button[2] é o elemento de id “botao2” e button[3] é aquele de id “botao3”. O mesmo raciocínio se aplica aos demais componentes. Esse entendimento é particularmente útil para a compreensão da especificação dos relacionamentos.

As linhas 10 a 16 mostram o relacionamento hipermídia definido entre os botões, o que expressa a sentença causal: quando for exibido o primeiro botão, os demais botões devem ser exibidos também.

As linhas 17 a 25 mostram o relacionamento hipermídia principal desse template. Ele se dá quando um dos botões é selecionado. Nesse caso, o respectivo texto e imagem são exibidos, conforme previamente descrito. A Seção 4.6 descreve em maiores detalhes a definição de relacionamentos.

As restrições de cardinalidade do template são descritas nas linhas 26 a 40. A primeira restrição (linhas 26 a 28) exige que seja instanciado pelo menos um componente “button”. A segunda restrição (linhas 29 a 31) exige a instanciação de pelo menos um componente “text”. A terceira restrição (linhas 32 a 34) exige a instanciação de pelo menos um componente “picture”. A quarta restrição (linhas 35 a 37) exige que a quantidade de componentes “button” seja igual a de componentes “text”. A quinta e última restrição (linhas 38 a 40) exige que a quantidade de componentes “button” seja igual a de componentes “picture”.

4.4. Seletores da Linguagem

Os templates e tipos definidos em um documento TAL fazem uso de seletores similares aos de CSS (Lie & Bos, 1997). A função do seletor é indicar quais elementos do documento de preenchimento devem ser considerados daquele tipo pelo processador de templates. A

Tabela 3 sumariza as diferentes formas de fazer essa seleção.

Tabela 3. Seletores da Linguagem TAL.

Padrão	**Significado**
*	Seleciona qualquer elemento
E	Seleciona qualquer elemento de nome E
EF	Seleciona qualquer elemento F descendente de um elemento E
E > F	Seleciona qualquer elemento F que é filho de um elemento E
E: i-child	Seleciona o elemento E quando ele é o i-ésimo filho de seu pai
E + F	Seleciona qualquer elemento F imediatamente precedido por um elemento E
E[foo]	Seleciona qualquer elemento E que tenha o atributo "foo"
E[foo=val]	Seleciona qualquer elemento E cujo valor do atributo "foo" seja igual a "val"
E[foo~=val]	Seleciona qualquer elemento E cujo valor do atributo "foo" seja uma lista de valores separados por espaço, um dos quais é exatamente igual a "val"
E.val	Equivalente a E[class~=val]
E#myId	Seleciona qualquer elemento E com identificador igual a "myId"

No exemplo "Botão-Texto-Imagem" (código na Figura 20), a composição em que deve ser aplicado aquele template (contexto de identificador "meuMenu") referencia ao template (vide Figura 19) pelo seu atributo *template* igual a "templates.xml#ButtonTextPicture". Ainda na Figura 19, os elementos <media> de id "botao1", "botao2" e "botao3", todos com atributo *class* igual a "button", são identificados como sendo do tipo de componente "button" por meio de seu seletor "media[class=Button]". O mesmo se aplica aos outros dois tipos de componentes do template ("text" e "picture").

4.5. Linguagem de Restrições

Autores em TAL podem especificar restrições na cardinalidade dos tipos definidos em seu vocabulário. Isso permite definir regras sobre a forma como os tipos podem ser instanciados por documentos que usam o template. A

especificação dessas restrições é feita com base no paradigma de programação por restrição (Wallace, 1996). Na programação por restrições, o autor especifica as propriedades de uma solução a ser encontrada (as restrições) em vez de uma sequência passo-a-passo (algorítmica) para se obter essa solução. A escolha do paradigma de restrições foi feita por permitir uma especificação mais natural e mais específica para os objetivos de TAL. Comparativamente, XTemplate exige o emprego de XPath e XSLT, que requerem conhecimento especializado e tornam seu uso bem mais difícil.

A Figura 21 descreve a sintaxe em EBNF[9] para a linguagem de restrições embutida em TAL. Os termos entre aspas e negrito são os terminais da gramática. As chaves ({}) denotam um trecho opcional. O símbolo | denota uma lista de opções para uma regra gramatical. A mesma notação é usada na seção seguinte, para definir a linguagem de relacionamentos.

Note que um seletor TAL poderia ser usado em uma restrição, o que dá mais versatilidade à linguagem. Na prática, restrições são descritas como sentenças lógicas entre duas expressões opcionalmente aninhadas com parênteses. Essas expressões podem ser formadas por constantes inteiras ou ainda conter a cardinalidade de tipos ou do resultado da aplicação de seletores.

```
constraint = exp cmp exp ;
exp        = term { operator exp };
term       = "(" exp ")" |
             integer |
             "#" id |
             "#" "{" selector "}";
cmp        = "=="|"~="|"<" | ">" | "<=" | ">=";
integer    = [0-9]+;
id         = "*"|[a-zA-z_][a-zA-Z0-9_]*;
selector   = TAL selector  ;
operator   = "+" |"-" | "*" | "/" | "^";
```

Figura 21. EBNF da linguagem de restrições embutida em TAL.

As restrições contidas no template de exemplo “Botão-Texto-Imagem” da Figura 20 são de fácil entendimento. As três primeiras restrições exigem a cardinalidade mínima 1 para o tipo “button” (#button>0), para o tipo “text” (#text>0) e para o tipo “picture” (#picture>0) (linhas 26, 29 e 32). A quarta

restrição (#button==#text) exige que a cardinalidade do tipo “button” seja igual à do tipo “text” (linha 35). Enfim, a quinta e última restrição (#button==#picture), exige que a cardinalidade dos tipos “button” e “picture” sejam iguais (linha 38).

4.6. Linguagem de Relacionamentos

Uma linguagem para a definição de relacionamentos hipermídia também foi embutida em TAL. A Figura 22 apresenta a sintaxe dessa linguagem em EBNF. Relacionamentos em TAL são inspirados nos elos causais de NCL, onde uma determinada condição deve ser satisfeita para se executar uma ação resultante. Na prática, a linguagem de relacionamentos proposta expressa a mesma semântica dos elos NCL, com suporte à definição de eventos de apresentação, de atribuição e de seleção. Conforme já mencionado, eventos de apresentação são aqueles relacionados à exibição de um conteúdo de mídia (como o início de sua apresentação, sua pausa ou término). Eventos de atribuição envolvem a modificação de uma propriedade de uma mídia (volume de som, largura, ou mesmo uma variável definida pelo usuário). Eventos de seleção são aqueles relacionados à interatividade do usuário (como apertar um botão colorido ou selecionar uma mídia em foco).

```
link     = condlist "then" actlist "end";
condlist = "(" condlist ")" withparams
            {lop condlist}?
           | condition {lop condlist}?;
condition = condname perspective withparams
            | assessment withparams;
assessmt = assessexpr rop assesexpr;
assesxpr = perspective {"+" string}?
           | string {"+" perspective}?;
condname = "onAbort" | "onBegin" | "onEnd"
          |"onBeginAttribution"|"onEndAttribution"
          |"onPause" | "onResume" | "onSelection";
actlist  = "(" actlist ")" withparams
           {aop actlist}?
          | action {aop actlist}?;
action   = actnoset perspective withparams
         | "set" perspective "=" string withparams
         | "set" perspective "=" perspective withparams;
actnoset = "abort" | "pause" | "resume"
           | "start" | "stop";
perspective = idref {"." idref}?;
withparams ={"with"{parameter,}*parameter{","}?}?;
parameter = idref "=" string;
string    = ""character sequence""
```

```
          | "'"character sequence"'";
aop   = "||" | {";"}?;
lop   = "and" | "or";
rop   = "<" | ">" | "<=" | ">=" | "==" | "~=";
idref = XML IDRef
```

Figura 22. Linguagem para relacionamentos hipermídia.

As condições de um relacionamento em TAL podem ser simples ou compostas. Condições simples estão associadas às transições de estado de evento "onAbort", "onBegin", "onBeginAttribution", "onEndAttribution", "onEnd", "onPause", "onResume" ou "onSelection". Uma explicação mais detalhada sobre essas palavras reservadas pode ser encontrada em (ABNT, 2007). Condições compostas são expressões lógicas (operadores "and" ou "or") entre condições simples ou compostas.

As ações de um relacionamento em TAL também podem ser simples ou compostas. As ações simples podem ser "abort", "pause", "resume", "start", "stop" ou "set". Uma explicação mais detalhada sobre essas palavras reservadas pode ser encontrada em (ABNT, 2007). As ações compostas são formadas pela associação de ações simples ou compostas através dos operadores paralelo ("||") ou sequencial (";").

Alguns exemplos ajudam no entendimento da linguagem de relacionamentos. Caso o autor queira definir um relacionamento entre duas mídias "A" e "B", onde o término de "A" deve fazer ser exibida "B", então teríamos "onEnd A then start B end". Em outro exemplo, e supondo que o autor queira tornar mudo o volume de som de um vídeo com identificador "V" sempre que ele for redimensionado (o que ocorre quando se altera a propriedade "bounds" do vídeo), isso pode ser escrito assim: "onEndAttribution V.bounds then set V.soundLevel = '0' end".

Retomando o exemplo de template "Botão-Texto-Imagem", há dois relacionamentos definidos na Figura 20. O primeiro, identificado por "startAllButtons" e apresentado nas linhas 10 a 16, tem como condição o inicio da apresentação do primeiro componente do tipo "button", (linha 11: "onBegin button[1] then"). A ação resultante dessa condição está escrita nas linhas 12 a 14 (<tal:forEach instance="button" iterator="i"> start button[i+1]; </tal:forEach>), comandando que todos os demais componentes do tipo "button" devem ser exibidos.

O segundo relacionamento do template “Botão-Texto-Imagem” é apresentado nas linhas 17 a 25 da Figura 20, com identificador “buttonSelection”. Esse é um relacionamento que envolve os três tipos de componente do template. Como a definição desse relacionamento inicia pelo elemento <forEach>, ele é transformado em vários elos quando o template é usado para gerar um documento NCL final. A condição desse relacionamento é especificada nas linhas 18 e 19, onde para cada componente do tipo “button”, sua seleção dispara o respectivo elo. A ação resultante desse elo é parar a exibição de todos os textos e imagens (linhas 20 a 22) e fazer ser exibido o texto e imagem relacionados ao botão selecionado (linha 23).

O Apêndice A apresenta a especificação completa de TAL em XML Schema (W3C, 2001). O Capítulo 6 revisita os requisitos propostos para TAL e discute de que forma a linguagem atende a tais requisitos.

4.7. Exemplos de Templates

Esta seção descreve outros exemplos do emprego de TAL para a especificação de templates. Na Seção 4.7.1 é ilustrado o mecanismo de extensão de templates da linguagem por meio da modelagem de *slideshows* de fotos, conforme descritos na Seção 3.1. Na Seção 4.7.2 é estendido o exemplo de template “Botão-Texto-Imagem” para permitir que um slideshow de imagens seja exibido no lugar de uma imagem estática, o que ilustra o aninhamento de templates da linguagem. Por fim, a Seção 4.7.3 descreve um template simples cujo objetivo é apenas estabelecer restrições na construção de uma família de documentos, as quais ajudam a ilustrar o emprego de TAL, por exemplo, para apoiar a definição de um padrão de interface para aplicações DTV.

4.7.1. Slideshow de fotos

Conforme descrito na Seção 3.1, um slideshow de fotos define a apresentação de uma lista ordenada de fotos, onde cada foto é exibida por determinado tempo para depois dar lugar à exibição da próxima foto. A Figura 23, a seguir, descreve a especificação básica de um slideshow de fotos em TAL, cujo

identificador é “temporalSlideshow” (linha 3). A apresentação inicia pelo primeiro componente “photo” encontrado, o que é particularizado para NCL na especificação de uma porta para essa instância de componente como um recurso (linha 4).

Ainda na Figura 23, o principal elemento do vocabulário desse template é o componente “photo”, que seleciona todos os elementos <media> cujo valor do atributo *class* é igual a “photo” (linha 5). Apenas um relacionamento é definido para esse template, cujo identificador é “changingPhotos” e define que o término da exibição de cada instância do componente “photo” faz ser exibido o próximo componente de forma circular (linhas 7 a 11). Há também uma restrição no template estabelecendo que o número mínimo de instâncias do componente “photo” deve ser 1 (linhas 12 a 14).

```
1:  <?xml version="1.0" encoding="UTF-8"?>
2:  <tal:tal id="templateDocument1">
3:     <tal:template id="temporalSlideshow">
4:        <port id="pPhoto1" component="photo[1]"/>
5:        <tal:component id="photo"
                         selects="media[class=photo]"/>
6:        <!-- circular slideshow -->
7:        <tal:link id="changingPhotos">
8:           <tal:forEach instance="photo" iterator="i">
9:              onEnd photo[i] then start photo[i%#photo+1] end
10:          </tal:forEach>
11:       </tal:link>
12:       <tal:assert test="#photo>0">
13:          There must be at least one PHOTO component.
14:       </tal:assert>
15:    </tal:template>
16: </tal:tal>
```

Figura 23. Documento temporalSlideshow.xml: exibição básica de fotos.

O template “temporalSlideshow” da Figura 23 é estendido pelo template “audioSlideshow” da Figura 24, a seguir. Esse novo documento representa fielmente a estrutura descrita pela Figura 12, onde um áudio de fundo toca durante a apresentação das fotos e seu término faz terminar também toda a apresentação. O documento “temporalSlideshow.xml” é importado para a base de templates desse novo documento com nome “doc” (linha 4). Dessa forma, o template “audioSlideshow” é definido como uma extensão, por meio do emprego do atributo *extends* (linha 6). Isso significa que o novo template contém todo

vocabulário, relacionamentos e restrições do template “temporalSlideshow” e ainda poder incluir nova informação especializada.

Ainda com relação ao template “audioSlideshow” definido na Figura 24, ele possui um recurso que adiciona a primeira instância do componente “audio” como iniciada no começo da apresentação desse template (linha 7). O componente “audio” é definido selecionando os elementos <media> com atributo *class* igual a “audio” (linha 8). O relacionamento “stopPresentation” (linhas 9 a 15) define que o término da exibição da primeira instância do componente “audio” deve terminar a exibição de todas as instâncias do componente “photo”. Há, também, a definição de uma restrição que exige apenas uma instância do componente “audio” (linhas 16 a 18).

```
1:  <?xml version="1.0" encoding="UTF-8"?>
2:  <tal:tal id="templateDocument2">
3:     <tal:templateBase>
4:        <tal:importBase documentURI="temporalSlideshow.xml"
                          alias="doc"/>
5:     </tal:templateBase>
6:     <tal:template id="audioSlideshow"
                     extends="doc#temporalSlideshow">
7:        <port id="pAudio" component="audio[1]"/>
8:        <tal:component id="audio" selects="media[class=audio]"/>
9:        <tal:link id="stopPresentation">
10:          onEnd audio[1] then
11:             <tal:forEach instance="photo" iterator="i">
12:                stop photo[i],
13:             </tal:forEach>
14:          end
15:       </tal:link>
16:       <tal:assert test="#audio==1">
17:        There must be at least and at most one AUDIO component.
18:       </tal:assert>
19:    </tal:template>
20: </tal:templateBase
```

Figura 24. Documento audioSlideshow.xml: slideshow de fotos com música de fundo.

Novamente com o propósito de ilustrar o recurso de herança ou extensão de templates em TAL, a Figura 25 apresenta outro documento, o qual define o template “navigationSlideshow”. Documentos que seguem esse template permitem, em adição ao comportamento padrão descrito no template-base “temporalSlideshow”, que as fotos sejam navegadas para frente ou para trás através das teclas de navegação “RIGHT” e “LEFT”.

Ainda com relação à Figura 25, o documento que contém o template “temporalSlideshow” é importado com nome “doc” (linhas 3 a 6). O novo

template “navigationSlideshow” é definido como extensão desse template importado (linha 7). Três novos relacionamentos são definidos: o primeiro estabelece a navegação para a próxima instância do componente “photo” por meio da tecla “RIGHT” (linhas 9 a 14); e os dois últimos especificam a navegação para a instância anterior do componente “photo” por meio da tecla “LEFT” (linhas 16 a 25). A navegação para a instância anterior do componente “photo” é descrita por dois relacionamentos para melhorar a clareza do código e pelo fato dessa navegação ser circular (a foto anterior à primeira para efeito de navegação é a última).

```
1:  <?xml version="1.0" encoding="UTF-8"?>
2:  <tal:tal id="templateDocument3">
3:     <tal:templateBase>
4:        <tal:importTAL documentURI="temporalSlideshow.xml"
5:                       alias="doc"/>
6:     </tal:templateBase>
7:     <tal:template id="navigationSlideshow"
                     extends="doc#temporalSlideshow">
8:        <!-- RIGHT button -->
9:        <tal:link id="nextPhoto">
10:          <tal:forEach instance="photo" iterator="i">
11:             onSelection photo[i] with {key="RIGHT"} then
12:             abort photo[i], start photo[i%#photo+1] end
13:          </tal:forEach>
14:       </tal:link>
15:       <!-- LEFT button -->
16:       <tal:link id="previousPhoto">
17:          <tal:forEach instance="photo" iterator="i">
18:             onSelection photo[i] with {key="LEFT"} then
19:             abort photo[i], start photo[i-1] end
20:          </tal:forEach>
21:       </tal:link>
22:       <tal:link id="circularPreviousPhoto">
23:          onSelection photo[1] with {key="LEFT"} then
24:          abort photo[1], start photo[#photo] end
25:       </tal:link>
26:    </tal:template>
27: </tal:tal>
```

Figura 25. Documento navigationSlideshow.xml: inclunido a navegação por teclas em um slideshow de fotos.

O mecanismo de herança ou extensão permite que um novo template especialize um template existente, o que promove maior reúso durante a autoria. Como exemplo, o template “navigationSlideshow” poderia ter ser definido como extensão ao template “audioSlideshow”, o que incorporaria o suporte à navegação por teclas no slideshow com áudio de fundo e o reusaria por completo.

4.7.2. Template Botão-Texto-Slideshow

O template "Botão-Texto-Slideshow", apresentado nessa seção, é parecido com o template "Botão-Texto-Imagem" descrito na Seção 4.1. A diferença é que esse novo template exibe, em adição ao texto descritivo, um slideshow de fotos em vez de uma imagem estática quando se navega pelos botões. Para expressar essa necessidade, TAL permite que o autor defina um template contendo uma composição (um tipo de componente) que internamente é definido de acordo com esse template referenciado.

A Figura 26 apresenta o código do template, cujo identificador é "ButtonTextSlideshow". Para evitar repetição, o foco dessa seção é explicar apenas o que esse template tem de diferente com relação ao "ButtonTextImage", descrito na Seção 4.1. O template "temporalSlideshow" é importado para ser reusado (linhas 4 a 6). No lugar do componente "picture" do exemplo anterior, é definido o componente "slideshow" (linha 13). Esse componente deve representar um template interno ao template em que ele está inserido, o que é expresso pelo seu atributo *template*, que faz referência ao template "temporalSlideshow", apresentado na Seção 4.7.1. O restante da especificação continua similar, com os relacionamentos e restrições que antes se referiam ao componente "picture" agora ligados ao componente "slideshow". Esse recurso de aninhamento de templates aumenta a expressividade de TAL e facilita a modelagem de vários cenários de uso.


```
1: <?xml version="1.0" encoding="ISO-8859-1"?>
2: <tal:tal id="butttontextslideshowTemplateBase">
3:    <!-- importing slideshow template base -->
4:    <tal:templateBase>
5:       <tal:importTAL documentURI="temporalSlideshow.xml"
                        alias="base"/>
6:    </tal:templateBase>
7:    <tal:template id="ButtonTextSlideshow">
8:       <port id="pButton"    component="button[1]"/>
9:       <port id="pText"      component="text[1]"/>
10:      <port id="pSlideshow" component="slideshow[1]"/>
11:      <tal:component id="button"
                        selects="media[class=button]"/>
12:      <tal:component id="text"
                        selects="media[class=text]"/>
13:      <tal:component id="slideshow"
                        selects="context[class=slideshow]"
                        template="base#temporalSlideShow"/>
14:      <tal:link id="startAllButtons">
```

```
15:          onBegin button[1] then
16:          <tal:forEach instance="button" iterator="i">
17:             start button[i+1],
18:          </tal:forEach>
19:          end
20:       </tal:link>
21:       <tal:link id="buttonSelection">
22:          <tal:forEach instance="button" iterator="i">
23:             onSelection button[i] then
24:             <tal:forEach instance="text" iterator="j">
25:                stop text[j], abort slideshow[j],
26:             </tal:forEach>
27:             start text[i], start slideshow[i] end
28:       </tal:forEach>
29:       </tal:link>
30:          <tal:assert test="#button>0">
31:             There must be at least one BUTTON component.
32:          </tal:assert>
33:          <tal:assert test="#text>0">
34:             There must be at least one TEXT component.
35:          </tal:assert>
36:          <tal:assert test="#slideshow>0">
37:             There must be at least one SLIDESHOW component.
38:          </tal:assert>
39:          <tal:assert test="#button==#text">
40:             The cardinality of BUTTON and TEXT components
                must be the same.
41:          </tal:assert>
42:          <tal:assert test="#button==#slideshow">
43:             The cardinality of BUTTON and SLIDESHOW
                components must be the same.
44:          </tal:assert>
45:       </tal:template>
46: </tal:tal>
```

Figura 26. Documento buttontextslideshow.xml: template “Botão-Texto-Slideshow”.

4.7.3. Template Padrão de Interface

Outro recurso oferecido pela linguagem TAL é a possibilidade de usá-la com o foco para a especificação de restrições, sem muito impor a estrutura hierárquica de seu vocabulário. Isso pode ser empregado, por exemplo, para exigir um padrão recorrente de interface de interação.

A Figura 27 apresenta o template “smallMenus”, bem simples, que contém o tipo de componente “anyElement” (linha 4), o tipo de relação “selectionLinks” (linhas 5 a 7) e uma restrição de aviso (linhas 8 a 10). O componente “anyElement” seleciona qualquer elemento. A relação “selectionLinks” seleciona todos os elementos <link> com atributo *class* igual a “selection” (particularizando o emprego desse template a NCL). O conteúdo dessa relação é uma especificação de relacionamento (linha 6), a qual pode ser usada para seleção em detrimento ao

atributo *selects* em certos casos de uso (como quando se está fazendo a seleção em um documento NCL). A restrição única do template exige que não haja mais de quatro elos de seleção em um menu, evitando que possam ser definidos menus com muitas opções para o usuário.

```
1: <?xml version="1.0" encoding="ISO-8859-1"?>
2: <tal:tal id="interfaceBase">
3:    <tal:template id="smallMenus">
4:       <tal:component id="anyElement" selects="*"/>
5:       <tal:relation id="selectionLinks"
                       selects="link[class=selection]">
6:          onSelection anyElement then start anyElement end
7:       </tal:relation>
8:       <tal:warning test="#selectionLinks<=4">
9:          Do not use more than 4 selection links in a menu.
10:      </tal:warning>
11:    </tal:template>
12: </tal:tal>
```

Figura 27. Template que restringe menus a quatro eventos de seleção.