TAL - Linguagem para Autoria de Templates de Documentos Hipermídia[1]

Carlos de Salles Soares Neto[1,2]
Luiz Fernando Gomes Soares[1]
Clarisse Sieckenius de Souza[1]

[1] Departamento de Informática – PUC-Rio
Rua Marquês de São Vicente, 225
Rio de Janeiro/RJ – 22453-900 – Brasil

[2] Departamento de Informática – UFMA
Av. dos Portugueses, Campus do Bacanga
São Luís/MA – 65080-040 – Brasil

csalles@deinf.ufma.br, lfgs@inf.puc-rio.br, clarisse@inf.puc-rio.br

ABSTRACT

This paper presents TAL (Template Authoring Language), an authoring language for hypermedia document templates. Templates describe document families that are structurally or semantically similar among them. TAL enables the description of a template independently of the target hypermedia authoring language. The paper also presents a TAL processor that generates complete hypermedia documents taking as input the template specification in TAL and a data file with the information that makes that document unique in its family.

RESUMO

Este artigo apresenta TAL (Template Authoring Language), uma linguagem para autoria de templates de documentos hipermídia. Templates descrevem famílias de documentos estruturalmente ou semanticamente similares entre si. TAL permite que se descreva um template de forma independente da linguagem hipermídia alvo. O artigo também apresenta um processador TAL que é capaz de gerar documentos hipermídia completos tendo como entrada a especificação do template em TAL e um arquivo de preenchimento com aquilo que torna o documento único em sua família.

Categories and Subject Descriptors

I.7.2 [**Document and Text Processing**]: Document Preparation – *hypertext/hypermedia, languages and systems, markup languages.*

General Terms

Design, Standardization, Languages, Verification.

Keywords

TAL, Nested Context Language, Archetype Oriented Programming.

1. INTRODUÇÃO

Este artigo apresenta TAL (Template Authoring Language), uma linguagem de autoria para templates de documentos hipermídia. TAL é uma linguagem declarativa modular que permite a especificação de *composições hipermídia incompletas* (conforme definido a seguir) de forma independente da linguagem de autoria alvo. Um processador TAL também foi desenvolvido de forma a permitir a instanciação de famílias de documentos como documentos hipermídia completos.

TAL é uma evolução da linguagem XTemplate [8] e tem como objetivo principal permitir que autores mais especializados especifiquem templates de documentos que possam ser usados por outros autores para criarem novos documentos de forma mais simples, rápida e livre de erros. Há várias razões que motivam o desenvolvimento baseado em template. Em primeiro lugar, templates promovem a coerência de aplicações (*branding*), onde produtoras podem definir e seguir um mesmo estilo de programação. Em segundo lugar, pelo fato das apresentações seguirem um mesmo padrão de interface recorrente, baseado no mesmo template, tais aplicações são mais facilmente usáveis. Famílias naturalmente permitem a indexação e agrupamento de documentos. Há ainda o já mencionado aumento do reúso, direcionando o autor no preenchimento apenas das lacunas que tornam um documento único em sua família. Templates também criam conceitos de domínio em aplicações similares, definindo um vocabulário próprio e um conjunto específico de restrições na forma que se compõem documentos de uma família.

Documentos hipermídia são usados para especificar apresentações de conteúdo oriundo de diversos tipos (vídeo, áudio, texto, imagem etc) e relações de sincronismo espaço-temporal entre tais mídias. A autoria desses documentos é comumente feita por meio de linguagens declarativas, entre elas as linguagens baseadas no tempo, como NCL[1] e SMIL[12]. NCL, assim como diversas dessas linguagens, faz uso do conceito de composição hipermídia como uma das mais importantes abstrações para o autor. Composições hipermídia englobam objetos de mídias e outras composições, recursivamente, e ainda a semântica de relacionamento entre esses objetos. Essa descrição da semântica de relacionamento pode ser implícita, como nos contêineres temporais "par" e "seq" em SMIL, ou explícita, como nos elos em NCL. Pode-se dizer que uma composição hipermídia encapsula um relacionamento semântico entre objetos. Note que mesmo em linguagens que não oferecem a abstração de composição, o corpo

[1] TAL: Template Authoring Language for Hypermedia Documents

de um documento forma uma composição. Em outras palavras, o conceito de composição sempre está presente, mesmo que em apenas um nível de aninhamento. Suporte à autoria de composições é, assim, uma necessidade requerida por todas essas linguagens.

Por encapsularem uma semântica de relacionamento, o suporte à autoria de composições hipermídia permite que se criem documentos com alto grau de reúso [11]. As linguagens baseadas no tempo que dão suporte a essa abstração, no entanto, não permitem a criação de composições com algum conteúdo interno (ou relacionamento entre conteúdos) em aberto, não-especificado. Como exemplo simples, nas linguagens citadas, uma composição pode ser usada facilmente para representar a exibição sequencial (*slideshow*) de um número fixo de fotos específicas com uma música de fundo determinada. Por outro lado, não é possível usar uma composição para descrever apenas essa semântica de apresentação (de forma a poder ser reusada), sem que o autor tenha que se comprometer em especificar quais são as fotos a serem exibidas, ou mesmo a música. Isso ocorre em tais linguagens porque nelas o objetivo é especificar um documento único, e não uma família de documentos.

Uma família de documentos é definida por um conjunto de documentos em que se percebe a recorrência de uma mesma estrutura de composição, a qual chamamos de *template*, ou semântica de relacionamento entre eles [1]. Neste trabalho, um template é descrito formalmente por meio de composições hipermídia com sua especificação feita de forma incompleta, através da definição de tipos. Dito de outra forma, uma composição hipermídia incompleta possui lacunas que devem ser preenchidas para que aquela composição seja plenamente especificada, além de regras que restringem a forma como essa composição em aberto pode ser preenchida.

As próximas seções do artigo estão estruturadas como se segue. A Seção 2 apresenta os trabalhos relacionados. Na Seção 3 é apresentada a linguagem TAL. A Seção 4 apresenta as conclusões e considerações finais.

2. TRABALHOS RELACIONADOS

Uma das principais inspirações para o desenvolvimento de TAL são os templates de composição propostos em [6][7], desenvolvidos na linguagem XTemplate. Por sua vez, a inspiração para a criação de XTemplate é oriunda da separação entre os conceitos de estilo e de configuração, feita em Linguagens de Descrição de Arquitetura (ADLs) [2]. Um estilo descreve a arquitetura conceitual de um sistema, enquanto uma configuração é a instância de um estilo. Há uma clara similaridade entre os conceitos de template de composição e de estilo arquitetural e ainda entre o conceito de composição hipermídia e o de configuração arquitetural. O trabalho aqui proposto, no entanto, é pautado no conceito de composição hipermídia *em aberto*, cuja analogia mais próxima das 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).

Em [8] 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. Diferente da presente proposta, XTemplate 3.0 é dirigida a uma linguagem hipermídia alvo específica e foca bem mais em criar facilidades para um autor mais especializado. Um usuário de XTemplate 3.0, mesmo aquele que apenas instancia templates de composição, precisa ter certos pré-requisitos técnicos, como XPath e XSLT [8]. Por sua vez, TAL tem como um de seus objetivos diminuir a necessidade por um autor mais especializado, o que é obtido ao evitar o uso de notações externas ao modelo conceitual e fora do nível de abstração da linguagem (como as instruções de processamento XSLT em XTemplate 3.0). Adicionalmente, é possível empregar TAL na autoria de outras linguagens declarativas, como SMIL, SVG[14], ou mesmo HTML/ECMAScript.

Em um trabalho anterior, estabelecemos características e métodos que possam vir a reger o processo de autoria orientado a templates [10]. Naquele trabalho dois principais atores envolvidos no processo são descritos: i) o autor de templates, que é mais especializado e responsável pela identificação e concepção dos templates; e ii) o autor de documentos, que tem uma base de conhecimento menos especializada e precisa apenas entender o superficialmente o template e como preencher suas lacunas. Um desafio apontado naquele trabalho é realizar a comunicação suave entre o significado de um template, algo tipicamente relacionado ao nível de autoria do primeiro ator, de forma a permitir que o segundo ator realize sua tarefa principal de instanciação do template projetado. No trabalho atual, os templates de TAL são concebidos de forma a permitir o entendimento também por parte do autor de documentos, sem que ele precise de maiores pré-requisitos de programação. Idealmente, busca-se diminuir a distância cognitiva entre os dois atores, tornando o autor de documentos um possível autor de templates em outro cenário de uso.

SMIL Timesheets[15] é uma contraparte temporal ao CSS [13], ambos do W3C, que tem por objetivo permitir a uma linguagem XML qualquer incorporar os elementos e atributos do controle temporal de SMIL. Timesheets especificam que elementos são ativos em um dado momento e qual seu escopo temporal dentro de um documento. Enquanto os Timesheets permitem a incorporação de aspectos temporais em documentos escritos em linguagens atemporais, a linguagem TAL permite especificar em separado a semântica temporal de composições que depois possa ser incorporada na especificação de documentos em linguagens baseadas no tempo.

3. A Linguagem TAL

TAL é uma linguagem modular XML. Um caso de uso bem simples da linguagem, utilizado como exemplo em todo o texto, é descrito pelo template "Botão-Texto-Imagem" na Seção 3.1, o qual é completamente especificado em TAL nas subseções seguintes. A Seção 3.2 apresenta os conceitos gerais da linguagem. A Seção 3.3 descreve os elementos e atributos de TAL. Na Seção 3.4 é apresentada a sintaxe para a definição dos seletores da linguagem. Na Seção 3.5, é discutida a sintaxe para descrever restrições. A Seção 3.6 apresenta a especificação proposta para relacionamentos.

3.1 Exemplo de Uso: Template Botão-Texto-Imagem

A Figura 1, a seguir, ilustra dois exemplos de aplicações interativas para TV Digital, oriundas de duas instituições diferentes, que seguem o template proposto de exemplo, o qual chamaremos "Botão-Texto-Imagem". 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 [3], um repositório público de aplicações para o Middleware Ginga-NCL, padrão do Sistema Brasileiro de TV Digital. As telas das aplicações descritas na Figura 1 foram extraídas desse repositório.

(a) Aplicação sobre Saúde da PRODERJ.

(b) Aplicação de Serviços da CAIXA®.

Figura 1 – 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 2 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 descrever um número variável dessas mídias em um contexto. É por essa razão que a Figura 2 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.

Ainda na Figura 2, 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".

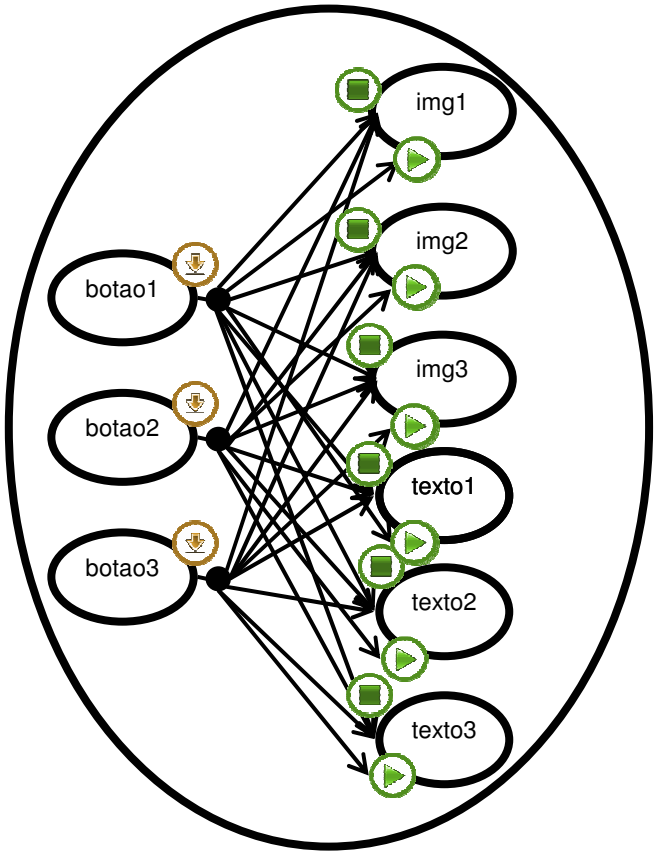

Figura 2 – Visão Estrutural de um Documento seguindo o Template "Botão-Texto-Imagem".

3.2 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 e interfaces;
- Restrições: que especificam regras sobre os tipos definidos no vocabulário;
- Recursos: objetos não-editáveis da composição;
- 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. Ao definir o vocabulário, também está-se 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 e interfaces. 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 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.

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 3. 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 3 não exibe esses recursos por razões didáticas, mas, como será visto na seção seguinte, eles podem ser expressos em TAL.

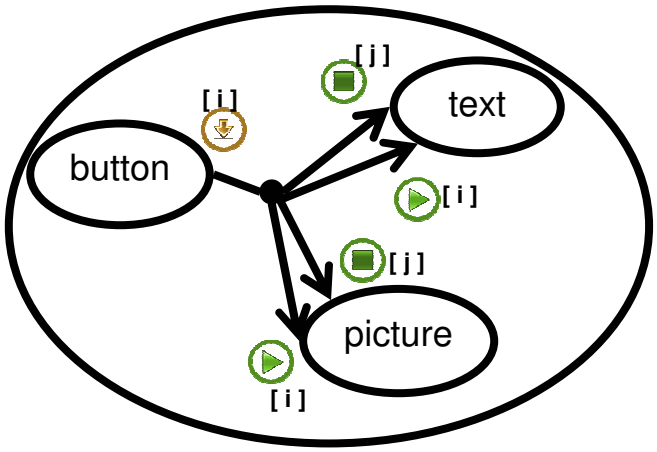

Figura 3 – Template "Botão-Texto-Imagem".

Ainda na Figura 3, 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).

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 2, é representado na Figura 4 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 3.3.

```
<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 4 – Documento NCL usando "Botão-Texto-Imagem".

3.3 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 3.4) e *template* (que referencia o documento TAL associado). O módulo *TemplateBase* também é definido só para uso pela linguagem de preenchimento, mas é opcional. Ele permite a definição da base de templates (por meio do elemento <templateBase>) usada pelo documento de preenchimento. Elementos <importTal>, filhos de <templateBase>, também definido pelo módulo, permitem a importação dos templates que poderão ser usados.

A Tabela 1 sumariza a estrutura hierárquica dos elementos e atributos dos demais módulos, que são usados pela linguagem TAL para 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 1. Elementos e atributos da linguagem TAL.

Elemento	Atributos	Filhos
tal	id	(template | importBase)+
importBase	documentURI, alias	
template	id, extends	(component | interface | relation | assert | report | warning | link)*
component	id, selects	(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 *Importing* permite importar templates de outros documentos para um novo documento TAL, usando o elemento <importBase>. 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 3.4.

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 3.6.

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[5]. 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 3.5. 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 falso. 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 3.6.

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.

Retomando o exemplo do template "Botão-Texto-Imagem", a Figura 5 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:   It must be at least one BUTTON element.
28:  </tal:assert>
29:  <tal:assert test="#text>0">
30:   It must be at least one TEXT element.
31:  </tal:assert>
32:  <tal:assert test="#picture>0">
33:   It 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 5 – Especificação do template "Botão-Texto-Imagem".

A linha 1 da Figura 5 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 4, 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 3.6 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".

3.4 Seletores da Linguagem

Os templates e tipos definidos em um documento TAL fazem uso de seletores similares aos de CSS [13]. A função do seletor é indicar quais elementos do *documento de preenchimento* devem ser considerados daquele tipo pelo processador de templates. A Tabela 2 sumariza as diferentes formas de fazer essa seleção.

Tabela 2. 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 5), a composição em que deve ser aplicado aquele template (contexto de identificador "meuMenu") referencia ao template (vide Figura 4) pelo seu atributo *template* igual a

"templates.xml#ButtonTextPicture". Ainda na Figura 4, 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").

3.5 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 [16]. Na programação por restrições, o autor especifica as propriedades de uma solução a ser encontrada (as restrições) ao invés 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 mais difícil.

A Figura 6 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 { operand exp };
term       = "(" exp ")" |
             integer |
             "#" id |
             "#" "{" selector "}";
cmp        = "=="|"~="|"<" | ">" | "<=" | ">=";
integer    = [0-9]+;
id         = "*"|[a-zA-z_][a-zA-Z0-9_]*;
selector   = TAL selector ;
operand    = "+" |"-" | "*" | "/" | "^";
```

Figura 6. EBNF da linguagem de restrições embutida em TAL.

As restrições contidas no template de exemplo "Botão-Texto-Imagem" da Figura 5 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).

3.6 Linguagem de Relacionamentos

Uma linguagem para a definição de relacionamentos hipermídia também foi embutida em TAL. A Figura 7 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. 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 7. 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". 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". 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 5. 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 5, 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).

4. CONCLUSÕES

Um template especificado em TAL é passível de ser instanciado como um documento final, o que é obtido como saída do *processador TAL*. Para isso, o processador TAL recebe como entrada a especificação de templates e ainda um *documento de preenchimento*. Essa abordagem permite que autores mais especializados criem novos templates usando TAL (conforme exemplo da Figura 5) e ainda que outros autores menos especializados utilizem essa especificação para criarem novos documentos hipermídia, apenas definindo novos documentos de preenchimento (conforme Figura 4 exemplifica em NCL), reusando toda a estrutura semântica do template. Esse paradigma de desenvolvimento proposto por TAL parece atender ao ambiente de desenvolvimento de aplicações interativas para TV digital, onde é comum a existência de programas que repetem um mesmo formato de aplicação.

Um objetivo colateral alcançado com TAL é permitir especificar a semântica de uma composição de forma explícita. Ao definir um template como uma composição hipermídia *em aberto*, novas composições podem ser criadas especializando essa especificação. Visto dessa forma, um template pode ser entendido como a descrição de uma semântica de relacionamentos de uma composição, a qual pode ser aplicada livremente para se criar novas composições.

O processador TAL para NCL está disponível como código livre em uma implementação Lua. Como trabalho futuro, ainda visando a linguagem NCL, planeja-se acoplar o uso de templates a ferramentas de autoria para linguagens baseadas no tempo, como o Composer [4]. Uma nova visão pode ser acoplada a essa ferramenta para atender às necessidades de um autor criando novos templates. Como vantagem, essa visão já estaria integrada ao ambiente de autoria de documentos hipermídia e assim seria ainda mais simples o trabalho do autor na criação de documentos de preenchimento.

Outro trabalho futuro envolve a realização de testes com usuários objetivando oferecer mais evidências que tornem clara a melhor usabilidade de TAL quando comparada a outras linguagens para especificação de templates. Na ocasião, é interessante observar empiricamente o poder expressivo de TAL na concepção de templates e, em especial, sua escalabilidade.

5. REFERÊNCIAS

[1] ABNT NBR 15606-2:2007. Televisão digital terrestre - Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 2: Ginga-NCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações. Setembro 2007.

[2] Clements, P. C. A Survey of Architecture Description Languages. In: *International Workshop on Software Specifications & Design. Proceedings of the 8th International Workshop on Software Specification and Design (1996)*. ISBN:0-8186-7361-3.

[3] Clube NCL. http://www.clube.ncl.org.br. Acessado em Abril de 2010.

[4] Guimarães, R. L.; Costa, R. M. R.; Soares, L. F. G. Composer: Ambiente de Autoria de Aplicações Declarativas para TV Digital Interativa. In: XII Simpósio Brasileiro de Sistemas Multimídia e Web – WebMedia 2007, 2007.

[5] ISO/IEC Standard. ISO/IEC 19757-3:2006 Information technology -- Document Schema Definition Language (DSDL) -- Part 3: Rule-based validation – Schematron.

[6] Muchaluat-Saade, D. C. *Relações em Linguagens de Autoria Hipermídia: Aumentando Reuso e Expressividade*. Tese de doutorado, Departamento de Informática, PUC-Rio, Rio de Janeiro, Brasil, Março 2003.

[7] Muchaluat-Saade, D. C.; Soares, L. F. G. XConnector e XTemplate: Estendendo XLink para Aumentar Expressividade e Reuso. In: *VIII Simpósio Brasileiro de Sistemas Multimídia e Hipermídia-SBMedia2002*, 2002.

[8] Santos, J. A. F., Muchaluat-Saade, D. C. Linguagem XTemplate 3.0: Facilitando a Autoria de Programas NCL para TV Digital Interativa. In: *XV Simpósio Brasileiro de Sistemas Multimídia e Web – WebMedia 2009*, Fortaleza, Brasil, Outubro de 2009.

[9] Scowen, R. S. Extended BNF — A generic base standard. In: *Software Engineering Standards Symposium*, 1993.

[10] Soares Neto, C. S., Soares, L. F. G. 2008. Autoria Orientada a Arquétipos. In: *XXXIV Conferencia Latinoamericana de Informática - CLEI 2008*, Santa Fe, Argentina, 2008.

[11] Soares Neto, C. S., Soares, L. F. G. Reúso e Importação em Nested Context Language. In: *XV Simpósio Brasileiro de Sistemas Multimídia e Web – WebMedia 2009*. Fortaleza, Brasil, Outubro de 2009.

[12] W3C. 2008. Synchronized Multimedia Integration Language (SMIL 3.0) W3C Recommendation. Disponível em: http://www.w3.org/TR/2008/REC-SMIL3-200812

[13] W3C. 2009. Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification. Disponível em: http://www.w3.org/TR/CSS2/.

[14] W3C. 2009. Scalabe Vector Graphics. W3C Recommendation. Disponível em: http://www.w3.org/TR/SVG11/.

[15] W3C. SMIL Timesheets 1.0. W3C Working Draft. Disponível em: http://www.w3.org/TR/timesheets/.

[16] Wallace, M. Practical Applications of Constraint Programming. In: *Constraints*, Vol. 1, pp:139-168, 1996.