EDITEC: Editor Gráfico de Templates de Composição para Facilitar a Autoria de Programas para TV Digital Interativa*

Jean Ribeiro Damasceno[1] Joel A. Ferreira dos Santos[2] Débora C. Muchaluat Saade[2]

Laboratório MídiaCom
[1]Departamento de Engenharia de Telecomunicações
[2]Departamento de Ciência da Computação
Universidade Federal Fluminense
R. Passo da Pátria, 156 - Bloco E - Sala 408 - São Domingos, Niterói, RJ, Brasil
{damascon, joel, debora}@midiacom.uff.br

ABSTRACT
This paper presents EDITEC, a graphical editor for hypermedia composite templates based on the XTemplate 3.0 language. EDITEC was designed for the creation of templates using a user-friendly visual graphical approach. It provides several options for representing iteration structures and a graphical interface for creating basic XPath expressions. The system provides a friendly environment with multiple views, giving the user a complete control of the composite template during the authoring process. Composite templates can be used in NCL programs for embedding spatio-temporal semantics into NCL contexts and making the production of interactive content easier for the Brazilian Digital TV System.

RESUMO
Este artigo apresenta o EDITEC, um editor gráfico de *templates* de composição hipermídia baseado na linguagem XTemplate 3.0. O EDITEC foi projetado para a criação de *templates* usando uma abordagem gráfica visual de fácil uso e aprendizado. Ele fornece diversas opções para representar estruturas de iteração e uma interface gráfica para a criação de expressões XPath básicas. O sistema provê um ambiente amigável com múltiplas visões, dando ao autor um completo controle do *template* de composição durante o processo de autoria. *Templates* de composição podem ser usados em programas NCL para embutir semântica espaço-temporal em contextos NCL e facilitar a produção de conteúdo interativo para o Sistema Brasileiro de TV Digital.

Categories and Subject Descriptors
H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *graphical user interfaces (GUI), interaction styles.*

General Terms
Design, Languages

Keywords
EDITEC, NCL, XTemplate, XPath, templates, editor gráfico, templates de composição.

WebMedia'10, October 5–8, 2010, Belo Horizonte, MG, Brazil.

1. INTRODUÇÃO
No Sistema Brasileiro de TV Digital, o ambiente declarativo Ginga-NCL utiliza a linguagem NCL (*Nested Context Language*) [8] para o desenvolvimento de aplicações interativas. Ginga-NCL também foi adotado como padrão internacional ITU para serviços de IPTV na recomendação H.761 [5]. Com a difusão do uso da linguagem, o número de aplicações NCL tende a crescer, sendo importante o desenvolvimento de ferramentas que auxiliem em seu desenvolvimento. Com o uso de NCL para TV digital, os telespectadores podem ter acesso a uma programação interativa e de maior qualidade. Exemplos de aplicações são programas de governo eletrônico (*t-Government*), de saúde (*t-Health*), além de outros programas não-lineares, onde a continuação pode ser escolhida pelo telespectador, ou mesmo um *quiz* realizado em programas de auditório.

A autoria de aplicações declarativas mais elaboradas para TV digital que possuem muitos nós, a necessidade de se definir um número grande de relacionamentos entre eles, bem como uma especificação de regiões e descritores, acaba introduzindo mais dificuldade em sua autoria, pois os documentos NCL ficam maiores, tornando o surgimento de erros mais frequentes. Neste cenário, o reúso de especificações genéricas de programas e o oferecimento de editores gráficos facilitam bastante a autoria.

A linguagem XTemplate 3.0 [9] apresenta-se como uma forma de se definir tais estruturas genéricas através do conceito de *templates* de composição, permitindo o reúso de especificações estruturais em documentos distintos, mas que têm características estruturais semelhantes. O usuário de um *template* só precisa especificar as mídias que serão usadas em seu documento. Os leiautes espaciais e as relações temporais, bem como as características de interatividade dos documentos, são automaticamente gerados pelo uso do *template*, assim, os usuários não precisam se preocupar com essas definições, tornando a criação de documentos NCL uma tarefa mais fácil.

Um problema que persiste é que a criação de *templates* usando a abordagem textual através da linguagem XTemplate não é uma tarefa tão simples para usuários inexperientes. O usuário precisa

* EDITEC: Composite Template Graphical Editor to Facilitate Interactive Digital TV Program Authoring

dominar bem a sintaxe e semântica da linguagem XTemplate e ter uma certa capacidade de abstração, além da dificuldade no entendimento da estrutura do documento conforme a quantidade de informação cresce. Edição textual permite alta flexibilidade e poder de edição, por outro lado, consome muito tempo. Para certos usuários, o tempo gasto especificando o documento na forma puramente textual torna o processo de autoria trabalhoso.

Assim, a disponibilidade de uma ferramenta gráfica para a autoria de *templates* é desejável. Em um editor gráfico, o uso de ícones para rotular as ações de edição permite que o autor estabeleça uma associação entre as imagens de componentes da interface gráfica com eventos de edição, ao invés de guardar nomes de elementos XML e comandos da linguagem textual. Entretanto, o design de tal ferramenta não é uma tarefa simples. O desenvolvimento de uma ferramenta que seja fácil de usar e ao mesmo tempo seja flexível o bastante para permitir a construção de diversos *templates* é um desafio.

Alguns requisitos são desejáveis em uma ferramenta gráfica de edição de *templates* de composição, tais como: várias visões do documento, definição de componentes genéricos, definição de estruturas de repetição, descrição semântica do *template*, definição de expressões XPath [13], portabilidade em diferentes plataformas e uma interface flexível com capacidade de realizar edições de estruturas mais elaboradas.

Visando a atender a esses requisitos, este trabalho propõe o EDITEC, um editor gráfico de *templates* de composição NCL baseado na linguagem XTemplate 3.0, que permite a criação de *templates* de forma interativa. O objetivo do EDITEC é auxiliar a criação de *templates* principalmente por usuários mais leigos em XTemplate. O usuário precisa de menos esforço na autoria usando o sistema do que teria se os *templates* fossem construídos diretamente de forma textual.

O sistema oferece um ambiente gráfico amigável com múltiplas visões/janelas que ajudam os autores a criar *templates* eficientemente. A apresentação de um documento em diferentes visões fornece ao usuário uma maior intuição no desenvolvimento de suas aplicações, dando a eles uma visão global prévia do documento, na qual o autor pode ter um retorno visual mais imediato durante a edição.

O restante do artigo está organizado da seguinte forma. A Seção 2 apresenta alguns trabalhos relacionados. A Seção 3 aborda as linguagens NCL e XTemplate 3.0. A Seção 4 apresenta o EDITEC e as visões disponíveis para edição. A Seção 5 apresenta um modelo usado para representar graficamente estruturas com diferentes opções de iteração. A Seção 6 apresenta a interface para declarar expressões XPath básicas. A Seção 7 aborda um exemplo de *template* construído no EDITEC. As considerações finais e trabalhos futuros são apresentados na Seção 8.

2. TRABALHOS RELACIONADOS

O *LimSee2* [2, 6] é uma ferramenta de código aberto para autoria de documentos SMIL [12], oferecendo várias visões integradas (temporal, espacial, textual etc.). Ele providencia duas visões mais aprimoradas (visão de leiaute e visão temporal) e duas visões mais simples (visão XML e visão de código).

Usuários podem ajustar a sincronização de mídias através de movimentação e sincronização de caixas de objetos em uma visão temporal. A visão de leiaute é a mais sofisticada. Ela produz imagens, fotos em movimento, áudio e estilo de texto. Ela também simula a reprodução de uma apresentação, incluindo animações e diversos efeitos, tais como a transparência.

O *LimSee3* [7] é uma ferramenta de autoria multimídia de código aberto baseada em *templates*. Em [1], o modelo *LimSee3* é proposto. Esse modelo usa *templates* para ajudar na autoria de documentos multimídia. Embora os nomes possam sugerir versões diferentes de um mesmo editor, *LimSee3* não é uma continuação direta ou uma nova versão do *LimSee2*. *LimSee2* é um editor SMIL e, como tal, é depende da linguagem base. *LimSee3* objetiva autoria independente de linguagem, baseado em uma estruturação lógica de documentos multimídia. Ele pode produzir documentos baseados no padrão SMIL, mas seu modelo interno não é dependente do SMIL.

No *LimSee3*, um documento é criado a partir da instanciação de um *template*, em um tipo de aplicação guiada. Um *template* pode ser visto, no *LimSee3*, como um "documento com buracos" chamados de zonas e o processo de instanciação consiste no preenchimento dos "buracos" com conteúdo de mídia. Ele não provê uma distinção clara entre o *template* e sua instância, já que o autor deve editar um *template* diretamente para preencher suas lacunas.

LimSee3 organiza mídias em bibliotecas de mídias. Depois de escolhido um *template*, o usuário pode, por exemplo, através de cliques e arrastes do mouse, mover suas mídias específicas das bibliotecas para as zonas definidas no *template*. Ele permite a manipulação, através de múltiplas visões, de todos os elementos definidos no modelo, isto é, documentos, objetos compostos, detalhes de tempo, leiaute e relações.

O *Composer* [4] é uma ferramenta de autoria de código aberto implementada em Java que fornece várias visões integradas para a criação de documentos NCL. Ele permite ao usuário trabalhar com as visões estrutural, temporal, textual e de leiaute. Sempre que é feita alguma alteração em uma visão ela é refletida nas demais visões.

A visão estrutural abstrai as principais entidades definidas em NCL [11]. Nessa visão, o autor pode criar, editar e deletar objetos de mídia que compõem um documento, contextos (conjunto de objetos e seus relacionamentos) e elos (relacionamentos entre objetos). Na visão de leiaute o autor pode criar, editar e remover regiões espaciais associadas com a exibição inicial dos objetos. A visão temporal abstrai os relacionamentos temporais entre objetos de mídia, fornecendo uma visualização gráfica a esses relacionamentos. Essa visão permite a representação e simulação de eventos imprevisíveis, como interações do telespectador e adaptações de conteúdo. O *Composer* não oferece o uso de *templates* para autoria.

O *Lamp* [3] é um ambiente de autoria que contém um editor visual para autoria de apresentações, um simulador de execução para testar o comportamento da apresentação e um *player* para exibir e executar a apresentação completa. Ele permite a definição de *templates* de apresentação que podem ser instanciados com diferentes dados. No modelo usado pelo *Lamp*, são definidas cinco relações de sincronização básicas e mais dois tipos de controle de mídia chamados de marcador de tempo e objeto de interação do usuário.

O principal componente do ambiente de autoria é o editor visual baseado em uma notação gráfica, na qual os nós são objetos de mídia e as arestas são relações de sincronização entre eles. O editor visual também provê facilidades para desenhar o leiaute da

tela. O ambiente de autoria visual é desenvolvido em linguagem Java. O editor visual possui duas visões: uma para definir os canais e o leiaute da apresentação e outra para definir o comportamento temporal. O resultado da apresentação é salvo em um documento XML.

3. TEMPLATES DE COMPOSIÇÃO PARA NCL

3.1 A Linguagem NCL

A Linguagem NCL (*Nested Context Language*) [11] é a linguagem declarativa adotada no Sistema Brasileiro de TV Digital, para o desenvolvimento de aplicações interativas declarativas. Como toda linguagem de autoria hipermídia, NCL foca no sincronismo entre mídias (textos, imagens, vídeos, áudio etc), não se preocupando com a definição do conteúdo das mídias em si. NCL define nós (mídias e nós de contexto) e elos que representam o sincronismo entre esses nós. Um elo NCL utiliza um conector hipermídia, que define uma relação genérica entre nós, sem se preocupar com os nós que efetivamente participarão dessa relação.

Por ser uma linguagem declarativa, a linguagem NCL não tem como foco funções de uma linguagem de programação tradicional, como cálculos matemáticos ou a definição de estruturas de repetição. Quando esse tipo de tarefa é necessário, o autor usando NCL emprega scripts Lua [8], definidos como nós de mídia para esse fim.

NCL é uma linguagem baseada no padrão XML, cujo elemento raiz, que representa um documento, tem como filhos o cabeçalho e o corpo.

No cabeçalho, são definidas as bases de regiões, de descritores e de conectores, entre outras. A base de regiões é responsável por definir onde as mídias serão apresentadas na tela, como por exemplo, a posição na tela em que um determinado vídeo ou imagem será apresentado. A base de descritores define como as mídias serão apresentadas, como por exemplo, o volume de uma mídia de áudio, qual a região utilizada, uma borda na região quando ela está com o foco etc. A base de conectores define os conectores que serão usados nos elos posteriormente definidos no corpo do documento. Por ser uma definição genérica, um mesmo conector pode ser reusado em diferentes elos.

No corpo do documento NCL, são definidos os nós de mídia e os elos que definem os relacionamentos entre os nós. Além disso, o corpo pode ainda definir composições (nós de contexto), que são utilizadas para estruturar logicamente o documento, elementos de controle de conteúdo (*switches*), que são utilizados para a apresentação alternativa de conteúdo e pontos de interfaces dos nós de contextos e dos nós de mídia, dentre outros.

Um elo é formado por um conector, que define sua semântica espaço-temporal, e por um conjunto de *binds*, que relacionam cada papel do conector a um nó. Por exemplo, se fizéssemos uma comparação de um elo NCL com a seguinte frase "O professor ensina ao aluno", 'professor' e 'aluno' seriam os nós e 'ensina ao' seria o conector. De acordo com esse conector alguém tem o papel de ensinar e outro alguém tem o papel de aprender. A idéia do conector é definir a semântica do relacionamento, mas sem dizer quem serão os participantes deste relacionamento, permitindo assim a reutilização do conector em outros relacionamentos (elos) definindo apenas novos nós. Outro exemplo seria "O pai ensina ao filho". Neste outro elo, o conector é o mesmo da frase anterior 'ensina ao', mas os participantes são outros. Se em um documento NCL, tivéssemos uma mídia vídeo e uma mídia áudio e um conector causal hipermídia do tipo "onBeginStart", poderíamos criar um elo NCL do tipo "Quando o vídeo começar, inicia o áudio".

3.2 A Linguagem XTemplate 3.0

A linguagem XTemplate 3.0 [9] provê suporte para definição de *templates* de composição hipermídia. *Templates* de composição descrevem de uma forma genérica todo o conteúdo de uma composição, um contexto em NCL, definindo componentes genéricos e os relacionamentos entre eles. O documento cuja composição utiliza um *template* fica responsável apenas por indicar os nós que serão utilizados, simplificando a autoria de documentos hipermídia.

Para que um documento hipermídia usando *templates* possa ser executado em uma implementação padrão do formatador da linguagem alvo (por exemplo, NCL), é necessário o processamento deste documento transformando-o em um documento completo (sem *templates*). A Figura 1 ilustra essa ideia.

Com o uso de *templates* de composição, torna-se possível o reúso de especificações estruturais e espaço-temporais em diferentes contextos NCL. Além disso, facilita-se o trabalho do autor de aplicações hipermídia, uma vez que não há a necessidade de se definir toda a sincronização entre os elementos de um documento hipermídia, somente os componentes que serão utilizados.

A linguagem XTemplate, assim como NCL, também é baseada em XML. A definição de um *template* de composição é feita em quatros partes principais: cabeçalho, vocabulário, corpo e restrições. No vocabulário, são definidos os componentes genéricos do *template*, onde cada um deles é identificado por um atributo *xlabel*. Este atributo deve ser usado pelos nós do documento NCL que usa o *template*. Um nó, componente de um contexto NCL que utiliza um *template*, deve definir um atributo *xlabel* com mesmo valor do *xlabel* do elemento genérico correspondente definido no *template*. No corpo do *template*, são definidos os relacionamentos genéricos (elos) entre os componentes (nós) do *template*. Finalmente, a parte de restrições permite a definição de restrições adicionais sobre os componentes (nós e elos) utilizando a linguagem XPath, que são usadas para garantir a consistência do documento final gerado, evitando erros no programa criado.

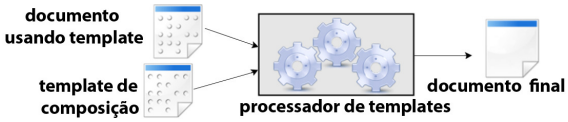

Figura 1. Utilização de templates na autoria de documentos hipermídia.

4. EDITEC: EDITOR GRÁFICO DE TEMPLATES DE COMPOSIÇÃO

O EDITEC, apresentado na Figura 2, é uma ferramenta desenvolvida em linguagem Java. A escolha de Java oferece portabilidade, permitindo o uso do editor em diferentes plataformas.

EDITEC fornece diversas visões integradas, que facilitam uma melhor visão global do documento, dando ao autor um completo controle do documento durante a edição. As visões estrutural, de

leiaute e textual são desenvolvidas em módulos separados para facilitar a manutenção e desenvolvimento do sistema. O editor possui um pacote, que é usado pelas diversas visões, que reflete a estrutura da linguagem XTemplate, similar a proposta apresentada em [10], que propõe um mapeamento de todos os elementos de uma linguagem em classes Java.

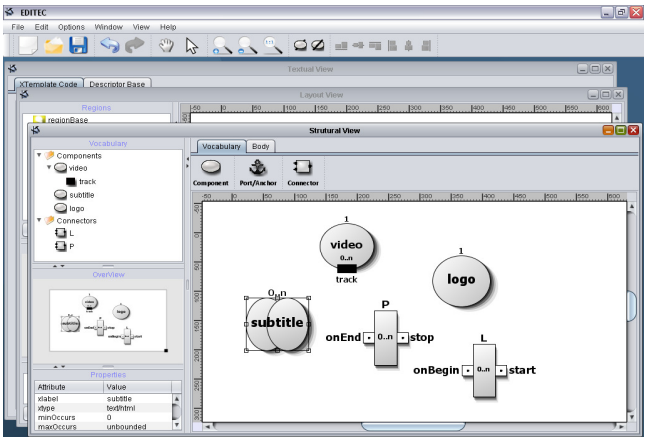

Figura 2. Ambiente de autoria EDITEC.

O usuário é livre para organizar o espaço de trabalho de acordo com a sua conveniência. Em particular, os elementos visuais podem ser movidos, minimizados, maximizados e desacoplados.

4.1 Facilidades Apresentadas pelo Editor

Na autoria de documentos hipermídia, o autor pode ter dificuldades com a quantidade de informação para manipular e com a localização de cada pedaço da informação. Em documentos hipermídia, essa desorientação é causada principalmente pelo número elevado de nós e elos (relacionamentos).

O EDITEC apresenta diversas funcionalidades que facilitam a visão e a busca por determinada informação. Ele apresenta em miniatura uma visão geral (*Overview*) da área de trabalho, tanto na visão estrutural quanto na visão de leiaute, onde é mostrada a área visível para o usuário e regiões em volta dessa área que não estão visíveis. Assim, elementos que não estão na região visível da área de trabalho continuam visíveis na visão geral.

Além da visão geral em miniatura da área de trabalho, o editor apresenta recursos de *zoom, scroll* e filtragem de elementos. Por exemplo, na visão estrutural, um duplo clique no conector, exibe ou oculta os nomes dos papéis, e um duplo clique em contextos os deixam colapsados. Na visão de leiaute podemos, por exemplo, fazer filtragem de regiões.

4.2 Visão Estrutural

A visão estrutural é a principal visão do EDITEC. Ela foi desenvolvida visando a facilitar o entendimento da linguagem XTemplate 3.0. Os elementos gráficos representam as principais entidades definidas na linguagem, indicando os nós de mídia, elos, conectores, portas e contextos.

A visão estrutural é dividida em duas subvisões (duas abas na interface), uma que apresenta os elementos do vocabulário chamada de *Vocabulary View* e outra que apresenta o corpo do *template* chamada de *Body View*. É na visão do corpo, através da criação de elos, que os relacionamentos espaço-temporais entre componentes do *template* são definidos.

Para criar um novo elemento (nó componente, porta, conector, elo), o usuário seleciona o ícone que representa o elemento desejado e o arrasta para a área de trabalho na posição desejada. Quando o botão do mouse é solto, é aberta uma janela referente ao objeto que o usuário criou, onde o usuário insere informações complementares. As opções disponíveis na janela dependerão do tipo de objeto criado. Depois de solto na posição desejada, o objeto pode ser movido e redimensionado, de forma a obter uma disposição espacial que favoreça a construção do *template*. Os elementos são criados de acordo com a linguagem XTemplate. Componentes, portas e conectores são criados na visão de vocabulário, já os elos e portas do *template* são criados na visão de corpo. O usuário pode tanto criar, como editar e apagar graficamente portas, componentes e elos. As Figuras 2 e 17 apresentam a visão estrutural.

Quando um elemento é selecionado, as suas propriedades são apresentadas na parte da visão estrutural referente a propriedades (canto inferior esquerdo da visão estrutural).

4.3 Visão de Leiaute

A visão de leiaute apresenta as regiões da tela onde as mídias do documento poderão ser exibidas, como ilustrado na Figura 3. Nessa visão, o autor pode criar, editar e apagar regiões graficamente. Toda vez que uma nova região é criada, é inserido um nó na árvore de regiões correspondente a nova região (apresentada no canto superior esquerdo da tela). As regiões podem ser filtradas clicando nos nós correspondentes a elas na árvore de regiões, permitindo uma melhor visualização da área de trabalho. A Figura 3 apresenta um exemplo da visão de leiaute.

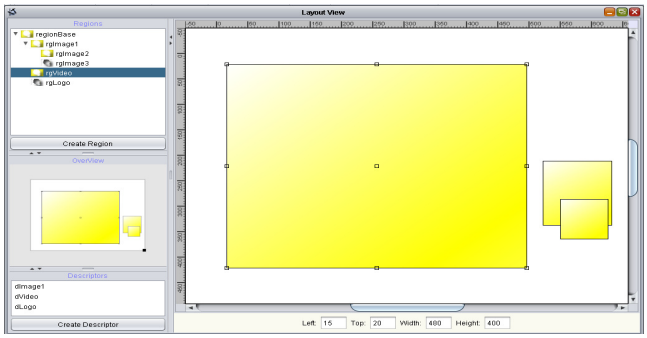

Figura 3. Visão de leiaute do EDITEC.

O leiaute espacial das regiões pode ser definido interativamente, através de redimensionamento e movimentação. Uma região filha tem sua posição definida de acordo com a posição de sua região pai, quando uma região pai é movimentada, todas as suas subregiões filhas também o são, mantendo as posições relativas. Para esta visão, também existem, na barra de ferramenta do EDITEC, várias opções de alinhamento de grupo de regiões.

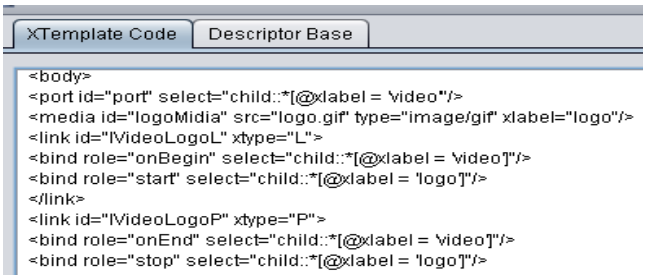

Figura 4. Parte da visão textual do EDITEC.

Na visão de leiaute também é possível a criação de descritores, com a opção de referenciar uma região criada. Os descritores criados são listados no canto inferior esquerdo da visão de leiaute.

4.4 Visão Textual

A visão textual, apresentada parcialmente na Figura 4, é dividida em duas partes, uma que mostra o código do *template* e outra, relacionada à visão de leiaute, que mostra o código da base de descritores, regiões e regras do *template*.

5. ESTRUTURAS DE ITERAÇÃO

Esta seção apresenta um modelo, composto de diversas opções, para representar graficamente estruturas de iteração usadas na criação de *templates*. Ele é usado pelo EDITEC para criar, de forma gráfica, diversos tipos de relacionamentos (elos) entre componentes de um *template*. O EDITEC analisa a estrutura gráfica do *template* com as opções escolhidas, processa-a e gera o código XTemplate correspondente. As opções de iteração são especificadas em *binds* e *port mappings,* quando esses se ligam a componentes que representam um conjunto de mídias. A Tabela 1 mostra as diversas opções de uso de estruturas de iteração e as próximas subseções as apresentam mais detalhadamente.

Tabela 1. Opções de uso de estruturas de iteração

Opções	Definição
i	Associa, a cada iteração, um papel de conector ao elemento da posição i de um conjunto de elementos. Onde i, é incrementado a cada iteração até referenciar todas as posições possíveis do conjunto.
All	Associa um papel de conector a cada elemento de um conjunto de elementos ou cria uma nova porta para cada elemento de um conjunto de elementos.
All-i	Associa um papel de conector a cada elemento de um conjunto de elementos, mas exclui o nó gerador do evento de sofrer a ação imposta ao conjunto, caso esse nó faça parte do conjunto.
Prev/Next	Associa, a cada iteração, um papel de conector ao elemento anterior/seguinte ao da posição atual.

5.1 Opção "i"

A Figura 5 apresenta um exemplo de elo XTemplate criado na visão estrutural do EDITEC usado para ilustrar o uso da opção "i" na criação de um relacionamento espaço-temporal em um *template*. O elo XTemplate possui dois componentes que representam dois conjuntos distintos de mídias, a opção "i" é usada nos *binds* associados aos componentes "x" e "y". Esse elo XTemplate depois de processado gerará um conjunto de elos no documento NCL final.

Figura 5. Exemplo "i para i".

Na Figura 6, é apresentado o trecho de código do *template* relacionado a esse elo. O código XML é gerado automaticamente pelo EDITEC e exibido na visão textual. Pode-se notar que é mais fácil e intuitiva para o usuário a criação do elo através da abordagem visual do que através do paradigma textual, principalmente quando o documento apresenta muitos elos e mídias.

```
<variable name = "i" select = "1"/>
<for-each select = "child::*[@xlabel = 'x']">
      <link id = "link" xtype = "onBeginStart">
          <bind role = "onBegin" select = "current()"/>
          <bind role = "start" select = "child::*[@xlabel = 'y']
                                          [position() = $i]"/>
      </link>
      <variable name = "i" select = "$i+1"/>
</for-each>
```

Figura 6. Trecho de código XTemplate, exemplo "i para i".

Um contexto NCL que referencia a um *template* que contenha esse elo, ao ser processado, se transforma em um contexto com um conjunto de elos. O processador seleciona do documento que referencia o *template* todas as mídias que têm *xlabel* igual a "x", essas mídias receberão a mesma lógica definida para o componente genérico "x" do *template*. Depois de selecionadas, é definida uma estrutura de repetição que varia desde a primeira mídia até a última criando um elo a cada iteração. O primeiro elo criado tem como origem o primeiro nó de mídia com *xlabel* igual a "x" do documento, pois um *bind* associa esse nó ao papel de condição *onBegin*. O primeiro elo também tem como destino o primeiro nó de mídia com *xlabel* igual a "y", pois um outro *bind* é usado para associar esse nó de destino ao papel de ação *start*. O segundo elo criado cria um relacionamento entre a segunda mídia com *xlabel* igual a "x" e a segunda mídia com *xlabel* igual a "y", de forma similar ao feito no primeiro elo. Os demais elos seguem essa mesma lógica. A Figura 7 apresenta graficamente como seriam os elos resultantes no documento NCL final depois de processado.

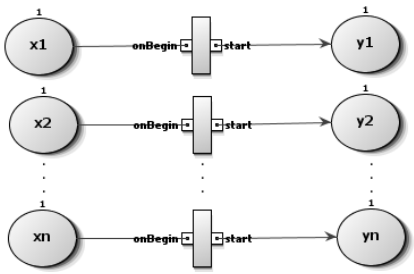

Figura 7. Representação gráfica em um documento NCL final, exemplo de opção "i para i".

Em algumas situações, restrições contendo expressões XPath são inseridas automaticamente pelo EDITEC ao código do *template* para facilitar o seu uso. Restrições mais elaboradas devem ser editadas diretamente no documento XTemplate gerado. Está em estudo uma interface para a construção de restrições pelo usuário de forma interativa que poderá ser inserida no EDITEC futuramente. No exemplo mostrado anteriormente o editor insere automaticamente uma restrição, pois para que o documento possa ser processado corretamente o número de mídias/âncoras com *xlabel* igual a "x" deve ser igual ao número de mídias com *xlabel* igual a "y". A criação de restrições automáticas seria para este caso semelhante à exibida na Figura 8.

```
<constraints>
  <constraint select = "count(child::*[@xlabel = 'x']) =
    count(child::*[@xlabel = 'y'])" description = "The number
    of x's must be equal to the number of y's."/>
</constraints>
```

Figura 8. Restrição inserida automaticamente.

5.2 Opção “All”

Para ilustrar o uso da opção “All”, suponha um elo XTemplate com dois conjuntos distintos de mídias, um na origem e outro no destino, onde usamos a opção “i” no *bind* referente ao componente que representa o conjunto de mídias de origem e a opção “All” no *bind* referente ao componente que representa o conjunto de mídias de destino. A Figura 9 apresenta um elo criado na visão estrutural do EDITEC que ilustra esse exemplo.

Figura 9. Exemplo de opções “i” e “All”.

Na Figura 10, é apresentado o trecho de código do *template* relacionado a esse elo. O código XML é gerado automaticamente pelo EDITEC.

```
<variable name  =  "i" select = "1"/>
<for-each select = "child::*[@xlabel = 'x']">
      <link id = "link" xtype = "onBeginStartN">
          <bind role = "onBegin" select = "current()"/>
          <bind role = "start" select = "child::*[@xlabel = 'y']"/>
      </link>
      <variable name = "i" select = "$i+1"/>
</for-each>
```

Figura 10. Trecho de código XTemplate, exemplo “i” e “All”.

Um contexto NCL que referencia um *template* que contenha esse elo, ao ser processado, se transforma em um contexto com um conjunto de elos. O processador seleciona do contexto NCL todas as mídias que têm *xlabel* igual a “x”. Depois é definida uma estrutura de repetição que varia desde a primeira mídia até a última criando um elo a cada iteração. O primeiro elo criado contém como origem o primeiro nó de mídia com *xlabel* igual a “x”, pois um *bind* associa essa primeira mídia ao papel *onBegin*, e outros *binds* associam o papel *start* a todas as mídias que tenham *xlabel* igual a “y”. O segundo elo criado usa um *bind* para associar a segunda mídia com *xlabel* igual a “x” ao papel *onBegin* e outros *binds* para associarem todas as mídias com *xlabel* igual a “y” ao papel *start*, de forma similar ao feito no primeiro elo. Os demais elos seguem essa mesma lógica. A Figura 11 apresenta graficamente como seriam os elos resultantes no documento NCL final depois de processado.

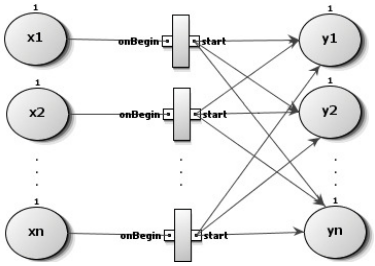

Figura 11. Representação gráfica em um documento NCL final, exemplo de opções “i” e “All”.

Neste exemplo da Figura 9, poderíamos ter colocado a opção “All” no *bind* relacionado ao papel *onBegin*, assim seria criado apenas um elo com vários *binds* ligados ao papel *onBegin*, um para cada mídia com *xlabel* igual a “x” e vários *binds* ligados ao papel *start*, um para cada mídia com *xlabel* igual a “y”.

A opção “All” também é usada na criação de portas. Neste caso é criada uma nova porta para cada mídia que tenha *xlabel* igual ao do componente alvo de *port mapping*.

A opção “All-i” é uma especificidade da primitiva “All”. Por exemplo, podemos usá-la em uma aplicação em que N botões (ou imagens) aparecem na tela e que, depois do pressionamento de um deles, todos os demais param a sua exibição, exceto o pressionado.

5.3 Opção “Next”

Em um relacionamento com a opção “Next”, exemplificado na Figura 12, temos um conjunto de elos gerado a partir de um elo XTemplate que tem um mesmo tipo de componente como nó de origem e de destino. Esse tipo de componente representa um conjunto de mídias, sendo que cada uma delas, exceto a primeira e a última, em um determinado elo, receberá o papel de ação e no seguinte, o papel de condição. A primeira receberá somente o papel de condição e a última somente o papel de ação.

Na Figura 13, temos o trecho de código do *template* relacionado a esse elo. O código XML é gerado automaticamente pelo EDITEC.

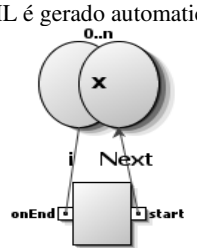

Figura 12. Exemplo de opção “Next”.

```
<variable name = "i" select = "1"/>
<for-each select = "child::*[@xlabel = 'x'][position !=  last()]">
      <link id = "link" xtype = "onEndStart">
            <bind role = "onEnd" select = "current()"/>
            <bind role = "start" select = "child::*[@xlabel = 'x']
                                          [position() = $i+1]"/>
      </link>
      <variable name = "i" select = "$i+1"/>
</for-each>
```

Figura 13. Trecho de código XTemplate, exemplo “Next”.

Um contexto NCL que referencia um *template* que contenha esse elo, ao ser processado, se transforma em um contexto com um conjunto de elos, onde cada um possui dois *binds*. Como pode ser visto na Figura 14, o primeiro elo criado contém como origem o primeiro nó de mídia do contexto NCL com *xlabel* igual a “x”. Um *bind* associa essa primeira mídia ao papel *onEnd* e o outro *bind* associa a segunda mídia com *xlabel* igual a “x” ao papel *start*. O segundo elo criado usa um *bind* para associar a segunda mídia com *xlabel* igual a “x” ao papel *onEnd* e o outro *bind* associa a terceira mídia com *xlabel* igual a “x” ao papel *start*, de forma similar ao feito no primeiro elo. Os demais elos seguem essa mesma lógica, sendo que a última mídia só sofre a ação, não sendo usada no papel de condição em nenhum elo.

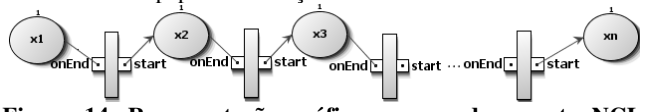

Figura 14. Representação gráfica em um documento NCL final, exemplo “Next”.

A opção “Prev” (*previous)* é semelhante à opção “Next”, só que ocorre uma inversão entre os papéis de condição e ação.

As opções “All-i”, “Prev” e “Next” são normalmente usadas quando um tipo de componente, que representa um conjunto de mídias, é a origem e o destino do elo. Em casos em que os componentes de origem e destino são diferentes, essas opções também podem ser usadas. Entretanto, as opções só têm sentidos em casos específicos, por isso, é importante ter muita atenção na hora de definir os tipos de opções.

No processamento das opções gráficas de interação, o EDITEC leva em conta os casos especiais, onde é possível ter, por exemplo, em um único elo, as opções “All” e “Next” em diferentes *binds*.

6. INTERFACE GRÁFICA PARA EXPRESSÕES XPATH BÁSICAS

Uma das principais propostas no desenvolvimento de uma ferramenta de edição gráfica de *templates* de composição foi a de proporcionar a construção de uma grande variedade de *templates* sem que o autor precise saber ou definir expressões XPath de forma textual. Para isso, foi desenvolvida uma interface amigável para auxiliar o usuário, substituindo a necessidade de definição textual de expressões XPath básicas por construções interativas e intuitivas da lógica das expressões XPath através de interface gráfica. Apesar dessa facilidade, a interface também fornece a opção de definir expressões XPath de forma textual.

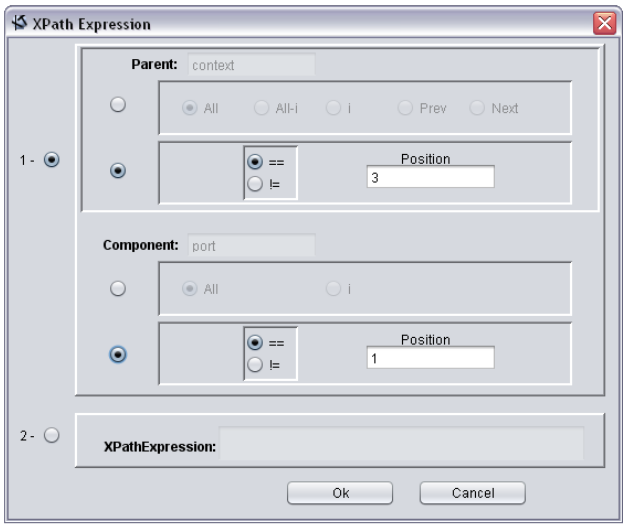

Figura 15. Exemplo de definição de expressão XPath básica.

A Figura 15 apresenta um exemplo de interface gráfica do EDITEC, com diversas opções que podem ser usadas para substituir a necessidade de definição textual de expressões XPath. A interface dá certa liberdade ao usuário para a construção de diversos tipos de expressões XPath básicas de forma interativa, como pode ser visto na opção 1 da Figura 15 e exemplificado mais a frente nesta seção.

A janela para a definição de expressões XPath é aberta quando se clica duas vezes em cima de um *bind* ou em um *port mapping* na visão estrutural do EDITEC. A janela está dividida em duas partes. Na primeira parte existem duas opções, onde o usuário pode escolher entre definir um tipo de generalização, onde é definida uma estrutura de repetição que serve para criar, a cada iteração, dependendo do caso, elos ou portas, ou definir de forma interativa expressões XPath básicas, como por exemplo: selecionar apenas o primeiro elemento (position()=1) ou selecionar o penúltimo elemento (last()-1) com determinado *xlabel*.

Os campos “Component” e “Parent” são preenchidos automaticamente. O campo “Component” é preenchido com o nome do componente alvo do *bind* (ou de um *port mapping*) responsável pela abertura da janela e o campo “Parent” é preenchido com o nome do pai desse componente. Quando o componente alvo do *bind* não é um elemento do tipo *port*, a parte referente a pai não é exibida, e na parte referente a componente são exibidas não só as opções “All” e “i”, mas todas as opções de generalização. Quando a janela é aberta devido a dois cliques em um *port mapping*, das opções de generalização, é exibida apenas a opção “All”.

A segunda parte da janela pode ser usada para a definição textual de expressões XPath, caso o usuário deseje. Esta facilidade deve ser usada com muito cuidado para não criar inconsistências na definição do *template*, pois o usuário tem liberdade para montar a expressão com a lógica desejada.

Caso uma expressão XPath seja definida de forma interativa ou textual, ou seja, sem o uso de operadores de generalização, o *bind* (ou *port mapping*) na visão estrutural recebe a indicação “XE”, mostrando que existe uma expressão XPath específica relacionada ao componente alvo do *bind* (ou *port mapping*).

A Figura 15 mostra um exemplo de construção de uma expressão XPath de forma interativa que seleciona o primeiro filho com o atributo *xlabel* igual a "port" do terceiro elemento com atributo *xlabel* igual a "context". Na Figura 16, temos um exemplo de trecho de código XML de um elo que usa a lógica apresentada na Figura 15, em seu segundo *bind*.

```
<link id= "link" xtype ="onBeginStart">
  <bind role = "onBegin" select = "child::*[@xlabel = 'video']"/>
  <bind role = "start" select = "child::*[@xlabel = 'context']
    [position() = 3] / child::*[@xlabel = 'port'][position() = 1]"/>
</link>
```

Figura 16. Trecho de código XTemplate, exemplo de expressão XPath construída através de interface gráfica.

Quando o objeto alvo do *bind* ou *port mapping* é um componente que representa apenas uma mídia, a janela XPath não é aberta. Já, no caso de um *bind* que associa um componente a um papel de condição, a janela XPath só habilita, das opções de generalização da parte 1, as opções “All” e “i”.

7. EXEMPLO DE USO DO EDITEC

A Figura 17 ilustra um exemplo de *template* de composição definido com EDITEC. Neste exemplo, o *template* de composição especifica a sincronização entre um objeto de vídeo, uma imagem (*logo*) e diversos objetos de texto (*subtitle*). Esse *template* pode ser usado em um documento onde é necessário sincronizar trechos de um vídeo (*tracks*) com as legendas correspondentes. Assim, todo o conjunto de relacionamentos de sincronização é especificado no *template* e o documento que o utiliza só precisa definir o nó de vídeo específico, seus trechos e os nós de texto com as legendas, facilitando bastante a autoria de documentos.

Esse *template* é criado no EDITEC da seguinte forma: na *Vocabulary View*, mostrada na Figura 2 da Seção 4, são definidos os componentes e conectores que poderão participar do *template*, sendo definidos o número mínimo e o máximo de ocorrências de cada um. A indicação de mínimo e máximo é mostrada em cima do símbolo do componente, por exemplo, em vídeo e logo o mínimo e o máximo são um e em *track* e *subtile* o mínimo é zero e o máximo é ilimitado. Quando o máximo é maior que um, o componente é representado por dois círculos sobrepostos (como o componente *subtitle* no exemplo). Na *Body View*, exibida na Figura 17, são definidos os elos responsáveis pelos relacionamentos entre os nós, as portas que especificam os elementos que serão iniciados quando a composição for inicializada e nós de mídia inseridos diretamente pelo *template*. Quando o *template* define nós específicos, não é necessário que o usuário os especifique no documento que usa o *template*. Os nós de mídia específicos são representados por duas circunferências concêntricas (como o nó *logo* no exemplo). A criação de todos os elementos é feita de forma interativa bastando apenas pressionar o mouse no ícone correspondente ao elemento desejado e o arrastar para a posição desejada da área de trabalho.

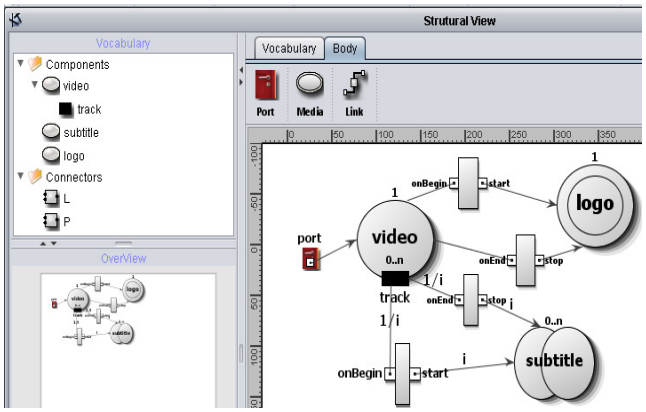

Figura 17. Visão estrutural de um template de composição.

Na construção de um elo (*link*), um dos conectores especificados no vocabulário é escolhido. A associação entre os papéis do conector escolhido e os nós componentes é feita através de *binds*. Para isso basta clicar no papel desejado e soltar o mouse em cima do componente alvo, criando assim um *bind* que associa o papel do conector ao componente desejado. O mapeamento entre portas e componentes é feito de forma similar à criação de *binds*.

Nos *binds* dos elos que relacionam os trechos do vídeo (*track*) e as legendas (*subtitle*), é usada a interface XPath com a opção "i". Parte do código XML desse exemplo de *template* é exibido na Figura 4.

8. CONCLUSÕES

Este artigo apresentou o EDITEC, um editor gráfico de *templates* de composição baseado na linguagem XTemplate 3.0, e o seu modelo para representar estruturas de iteração que permite, de forma fácil e interativa, a criação de diversos *templates* com as mais variadas semânticas espaço-temporais. Ele está inserido em um conjunto de ferramentas que têm como objetivo facilitar a criação de documentos NCL.

O EDITEC foi desenvolvido visando a atender aos diversos requisitos desejados para um editor de *templates* de composição para a linguagem NCL. O editor está em fase de aprimoramento. À medida que ele for sendo usado, novas facilidades poderão ser identificadas e acrescentadas ao editor, até mesmo, novas opções de generalização para representar estruturas de iteração.

Podem existir situações em que a ferramenta de edição gráfica não seja capaz de fornecer todas as funcionalidades para que a lógica desejada pelo autor seja implementada, sendo necessário, nessas situações, o emprego da edição textual.

Como trabalho futuro, pretende-se implementar a visão temporal do editor. Além disso, está em estudo o desenvolvimento de uma interface para a criação de restrições de forma interativa. A criação de restrições exige a construção de expressões XPath mais elaboradas. Outro trabalho futuro é o desenvolvimento de um editor gráfico de documentos NCL que usem *templates*.

9. REFERÊNCIAS

[1] Deltour, R. and Roisin, C. 2006. The Limsee3 Multimedia Authoring Model. In Proceedings of the 2006 ACM Symposium on Document Engineering, pages 173–175.

[2] Deltour, R., Layaida, N. and Weck, D. 2005. LimSee2: A Cross-Platform SMIL Authoring Tool. The European Research Consortium for Informatics and Mathematics – ERCIM News. n. 62, July 2005.

[3] Gaggi, O. and Celentano, A. 2006. A Laboratory for Prototyping and Testing Multimedia Presentations. International Journal of Software Engineering and Knowledge Engineering, vol. 16, No 4. August 2006.

[4] Guimarães, R. L., Costa, R. M. R. and Soares, L. F. G. 2007. Composer: Ambiente de Autoria de Aplicações Declarativas para TV Digital. XIII Simpósio Brasileiro de Sistemas Multimídia e Web - WebMedia2007.

[5] ITU International Telecommunication Union 2009. Nested Context Language (NCL) and Ginga-NCL for IPTV services. ITU-T Recommendation H.761. 2009.

[6] LimSee2, http://limsee2.gforge.inria.fr/

[7] LimSee3, http://limsee3.gforge.inria.fr/public-site/

[8] Norma ABNT 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, 2007.

[9] Santos, J. A. F. and Muchaluat-Saade, D. C. 2009. Linguagem XTemplate 3.0: Facilitando a Autoria de Programas NCL para TV Digital Interativa. XV Simpósio Brasileiro de Sistemas Multimídia e Web - WebMedia2009.

[10] Silva, H. V. O. 2005. X-SMIL: Aumentando Reuso e Expressividade em Linguagens de Autoria Hipermídia. Dissertação de mestrado, Departamento de Ciência da Computação, PUCRio, Rio de Janeiro, Brasil, Abril 2005.

[11] Soares, L. F. G. and Barbosa, S. D. J. 2009. Programando em NCL 3.0: desenvolvimento de aplicações para Middleware Ginga, TV digital e Web. Ed. Elsevier. Rio de Janeiro, 2009.

[12] W3C World-Wide Web Consortium. Synchronized Multimedia Integration Language - SMIL 2.1 Specification. Recommendation 13 December 2005.

[13] W3C World-Wide Web Consortium. XML Path Language - XPath 1.0. W3C Recommendation 16 November 1999.