Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Ferramenta baseada em templates para automação do processo de criação de aplicações t-commerce

Marina Ivanov Pereira Josué

JUIZ DE FORA
FEVEREIRO, 2014

Ferramenta baseada em templates para automação do processo de criação de aplicações t-commerce

Marina Ivanov Pereira Josué

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Marcelo Ferreira Moreno

JUIZ DE FORA

FEVEREIRO, 2014

Ferramenta baseada em templates para automação do processo de criação de aplicações t-commerce

Marina Ivanov Pereira Josué

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Marcelo Ferreira Moreno
Doutor em Informática

Eduardo Barrére
Doutor em Engenharia de Sistemas e Computação

José Maria Nazar David
Doutor em Engenharia de Sistemas e Computação

JUIZ DE FORA

07 DE FEVEREIRO, 2014

Resumo

Em sistemas de TV Digital modernos, aplicações interativas tomam a forma de documentos hipermídia, especificados em linguagem declarativa, descrevendo os objetos de mídia que as compõem, as relações de sincronização entre eles, incluindo as possibilidades de interação do usuário. No Sistema Brasileiro de TV Digital, as aplicações interativas podem ser desenvolvidas utilizando a linguagem declarativa denominada "Nested Context Language" (NCL). As aplicações podem oferecer diferentes serviços ao telespectador, incluindo o serviço de comércio eletrônico pela TV (t-commerce). A fim de facilitar o desenvolvimento de aplicações, ambientes gráficos de autoria são oferecidos aos profissionais de criação de conteúdo. Estes ambientes permitem que autores sem conhecimento da linguagem de autoria desenvolvam aplicações, e para isso, são utilizados templates que representam sua estrutura e, às vezes, alguma semântica. Neste contexto, o presente trabalho tem como objetivo automatizar o processo de desenvolvimento de aplicações para TV Digital interativa, especificamente do tipo t-commerce. A fim de alcançar este objetivo, foi desenvolvida uma ferramenta que serve de apoio ao processo de criação de aplicações multimídia, e tem como conceito-base o desenvolvimento baseado em templates, motivado por razões como coerência entre as aplicações e reúso. Além disso, a ferramenta permite ao autor visualizar as diferentes fases que compõem a aplicação. A solução proposta gera como resultado um documento NCL que representa a aplicação t-commerce.

Palavras-chave: TV Digital, NCL, Ferramenta de autoria, Template, T-Commerce

Abstract

In modern Digital TV systems, interactive applications take the form of hypermedia documents, specified in declarative language, describing their comprising media objects, the synchronization relationships among them, including the possibilities for user interaction. In the Brazilian Digital TV System, interactive applications can be developed by using the a declarative language called "Nested Context Language"(NCL). The applications can offer different services to the viewer, including electronic TV commerce (t-commerce). In order to facilitate application development, graphical authoring environments are available for content creation professionals. These environments enable authors without knowledge of the authoring language to develop applications. For this sake, templates that represent the application's structure and some of its semantics are used. In this context, this work aims at automating the development process of Digital TV applications, specifically t-commerce applications. In order to achieve this goal, a tool to support the creation process of multimedia applications was developed. The tool's main concept is the template-based development motivated by issues such as coherence between different applications as well as their reuse. The proposed solution generates as a result an NCL document that comprises the t-commerce application.

Keywords: Digital TV, NCL, Authoring Tool, Template, TCommerce.

Agradecimentos

A Deus pela proteção, força e principal responsável por mais essa conquista em minha vida.

Aos meus pais e minha irmã, pelo incentivo, compreensão, amor e apoio na formação do meu caráter. Sem eles nada seria possível.

Ao professor Marcelo Moreno, pela paciência, dedicação, incentivo e conhecimento transmitido que muito me auxiliou durante o desenvolvimento e conclusão desta monografia.

Ao professor Eduardo Barrére pelas sugestões, dicas e apoio que enriqueceram este trabalho.

Aos colegas do Laboratório de Aplicações e Inovação em Computação (LApIC), que fizeram parte da minha trajetória nestes anos.

Aos meus amigos e família pelo companheirismo, palavras de incentivo e por todos os momentos que estiveram ao meu lado.

Aos professores do Departamento de Ciência da Computação, pelos ensinamentos durante esses anos, que contribuíram muito para minha formação.

A todos, muito obrigada!

"O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo. Mesmo não atingindo o alvo, quem busca e vence obstáculos, no mínimo fará coisas admiráveis".

José de Alencar

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

EPG	Electronic Programming Guide
NCL	Nested Context Language
API	Application Programming Interface
IPTV	Internet Protocol Television
TAL	Template Authoring Language
NEXT	NCL Editor supporting XTemplate
RFID	Radio-frequency identification

1 Introdução

A implantação do Sistema Brasileiro de TV Digital (SBTVD) introduz modificações no modo com que o telespectador assiste e interage com sua TV, comparado ao antigo sistema de TV analógica. Este novo sistema proporciona melhoria na qualidade de imagem e som, a multiprogramação, pela qual vários programas com resolução mais baixa podem ser agrupados ocupando o espaço de um único canal de frequência, e a inserção de poder computacional nos dispositivos receptores.

Tais características fazem com que a transição do sistema de TV analógico para o digital possibilite a concessão de uma gama de novos serviços ao usuário, como guias eletrônicos de programação (EPG), comércio eletrônico (t-commerce), acesso a serviços governamentais (t-gov), educação a distância (t-learning), serviços bancários (t-banking), jogos, entre outros.

Os novos serviços oferecidos, são na maioria, aplicações multimídia residentes em um dispositivo receptor, ou fornecidas pelo envio de dados juntamente com o áudio e o vídeo principal de um programa. As aplicações consistem de documentos multimídia que podem conter diferentes tipos de mídias, como texto, imagem, áudio e vídeo, com relacionamentos síncronos no tempo e no espaço.

O desenvolvimento de aplicações para TV Digital pode ser feito utilizando tanto o paradigma de programação declarativo quanto o imperativo. Uma linguagem de programação declarativa apresenta mais alto nível de abstração, de modo que o programador informa apenas as tarefas a serem realizadas, sem precisar se preocupar no modo que o compilador ou interpretador da linguagem irá implementar as tarefas. Já na linguagem de programação imperativa, é preciso que o autor informe cada passo a ser executado, decompondo o resultado desejado em uma implementação algorítmica.

O Sistema Brasileiro de TV Digital Terrestre adotou como padrão o middleware Ginga (NBR 15604, 2008), que é uma camada de software responsável por executar os aplicativos e representa uma interface entre a aplicação e a plataforma de hardware/software do receptor. O middleware Ginga é composto por dois subsistemas, o Ginga-NCL (NBR

15606-2, 2008), declarativo, e o Ginga-J (NBR 15606-4, 2010), imperativo. O Ginga-NCL dá suporte a aplicações desenvolvidas utilizando a linguagem NCL (Nested Context Language) juntamente com a linguagem de script Lua. Utilizando a linguagem NCL, um autor pode descrever o comportamento temporal de uma apresentação multimídia, associar hyperlinks (interação do usuário) com objetos de mídia, definir alternativas para apresentação (adaptação) e descrever o layout da apresentação em múltiplos dispositivos. Já o subsistema Ginga-J é responsável pelo processamento de conteúdo desenvolvido na linguagem Java.

O desenvolvimento de aplicações para TV Digital, seja pelo uso de linguagem declarativa ou não-declarativa, normalmente requer conhecimento de programação, o que restringe o perfil de possíveis autores de aplicações. Com o objetivo de minimizar tais dificuldades relacionadas à criação de aplicativos, são desenvolvidas ferramentas de autoria. Estas ferramentas visam facilitar o trabalho de autoria fornecendo um ambiente gráfico para criação e edição de documentos multimída por autores com pouca ou nenhuma especialização em linguagens de programação para desenvolvimento de aplicações.

As ferramentas de autoria devem considerar os diversos tipos de dados que uma aplicação multimídia pode englobar, como áudio, vídeo, imagem, animações, texto e metadados; e também os possíveis relacionamentos de sincronização entre as mídias, seja no tempo ou no espaço. Além disso, no caso de aplicações para TV Digital, é preciso também levar em conta aspectos como modelo de negócios da emissora, questões de regulamentação, infraestrutura de hardware, software e canal de interatividade, exigidos pela aplicação para que o telespectador possa ter acesso a ela. O canal de interatividade é o meio de comunicação pelo qual cada usuário pode passar a interagir de forma individual com os programas. Este canal pode estar associado ao uso de dois canais, o canal de descida (vai dos produtores de conteúdo para os usuários) e o canal de retorno (vai do usuário para os outros componentes e está relacionado com as redes de telecomunicação). (Soares, 2009)

Durante a concepção de uma ferramenta de autoria deve-se analisar questões como o perfil do usuário, quem ele é, o que quer fazer e como quer fazer. No caso de edição de documentos NCL através de templates, é necessária a participação de dois perfis de autores

no processo de autoria. Autores com experiência em NCL, responsáveis pela autoria de templates, os quais poderão ser utilizados por autores com pouca ou nenhuma experiência na linguagem, de maneira declarativa ou através de editores gráficos que ofereçam o uso de templates.

Algumas ferramentas de autoria, como o NEXT (Silva, 2012) utilizam o conceito de desenvolvimento baseado em templates, visando facilitar ainda mais o processo de criação de aplicativos. De acordo com Silva (2012), um template é um modelo pré-definido de documento sem conteúdo, isto é, possui apenas partes essenciais de um documento, como cabeçalho e definições de como as mídias devem se comportar durante sua execução. E por isso, uma das tarefas do autor, se não a única, será indicar quais mídias serão utilizadas no documento que utiliza o template.

Segundo Neto (2010), o uso de templates representando um grupo de aplicações estruturalmente ou semanticamente similares entre si, oferece vários benefícios. Algumas destas vantagens são:

- Coerência entre aplicações construídas: autores de conteúdo definem e seguem um mesmo estilo de programação;
- Usabilidade: como as aplicações seguem um mesmo padrão de interface recorrente, o desenvolvimento de novas aplicações a partir de um mesmo template já utilizado pelo autor torna-se mais fácil;
- Aceleração da autoria: uma aplicação é desenvolvida mais rapidamente por quem já fez previamente aplicações utilizando o mesmo template.
- Aumento do reúso, direcionando o autor: com o uso de templates, uma aplicação pode ser feita não mais a partir do zero, mas sim, com base em outra aplicação com as mesmas características estruturais e semânticas.

Considerando que nenhuma das ferramentas de autoria atuais permitem a criação de aplicações de forma totalmente automatizada, sem a necessidade de escrita de códigos de programação por parte do autor de aplicações, o objetivo deste trabalho é desenvolver uma ferramenta capaz de dar suporte à geração automática de uma categoria específica

de aplicações para serem executadas na TV. Esta ferramenta visa facilitar o processo de criação de conteúdo para TV Digital interativa, por qualquer classe de usuário, seja ele especialista nas linguagens alvo de desenvolvimento para TV, ou usuários sem conhecimentos de programação. Para atingir tal objetivo, a solução proposta se apoiará no conceito de desenvolvimento baseado em templates, motivado pelos benefícios apresentados acima. Além disso, a ferramenta fornece ao autor a visão de fases, de modo que este tenha uma visão real do que está sendo exibido na aplicação, com pouquíssima abstração de mídias. Estas fases representam pontos da exibição da aplicação em que os objetos de mídia que a compõem se modificam, alterando de posição na tela ou sendo removidos. A visualização das fases da aplicação, possibilita que o autor tenha uma visão real do que está sendo exibido na aplicação, com pouquíssima abstração das mídias.

As aplicações geradas a partir da ferramenta, pertencem à categoria de comércio eletrônico televisivo (t-commerce) e são descritas através de documentos utilizando a linguagem NCL 3.0 e Lua.

Atualmente existem algumas ferramentas para geração de aplicações NCL para o middleware padrão brasileiro. Porém, a utilização destas ferramentas e a geração de conteúdo para TV ocorrem, na maioria dos casos, em etapas disjuntas. A criação de aplicação interativa integrada ao processo de gravação da programação é um cenário ainda não explorado pelos provedores de conteúdo televisivo. Nesse contexto, a justificativa deste projeto consiste na oportunidade de tornar a tarefa de desenvolvimento de conteúdo interativo uma das etapas do processo de concepção da programação para TV.

A monografia está estruturada da seguinte forma. O Capítulo 2 apresenta a fundamentação teórica, a fim de expor os trabalhos relacionados a esta proposta e o estado da arte de ferramentas de autoria de aplicações para TV Digital. Além disso, será apresentado o método de autoria orientada a template e algumas características de aplicações utilizadas no Sistema Brasileiro de TV Digital. O Capítulo 3 define a arquitetura proposta pelo trabalho para o processo de autoria de aplicações para TV Digital utilizando a ferramenta desenvolvida e apresenta um levantamento dos requisitos de aplicações t-commerce. O Capítulo 4 trata dos testes funcionais realizados na ferramenta proposta a fim de avaliá-la quanto o atendimento as requisitos funcionais levantados. No Capítulo 5,

o processo de implementação do projeto é descrito. O Capítulo 6 apresenta as conclusões do trabalho, e contém um breve resumo da solução proposta, as contribuições obtidas e os trabalhos futuros.

2 Fundamentação Teórica

Este capítulo apresenta conceitos e características de aplicações compatíveis com o Sistema Brasileiro de TV Digital, ferramentas de autoria e método de autoria orientada a templates. Esse estudo visa obter embasamento teórico para a seleção de requisitos necessários a um editor que possibilite a geração automática de aplicações para o ambiente de TV Digital.

2.1 Aplicações interativas

As aplicações para TV Digital, em sua maior parte, são compostas por um vídeo principal e um conjunto de tipos de dados associados a ele, como imagem, texto, áudio e até mesmo outro vídeo. Os diferentes tipos de dados que constituem uma aplicação interativa se relacionam no tempo e no espaço, de maneira sincronizada.

O conteúdo de uma aplicação para TV Digital pode ser exibido de modo linear, seguindo uma sequência de exibição pré-definida, ou não-linear, permitindo inclusive a interação por parte do telespectador, que conduz como a aplicação será exibida. Uma aplicação interativa pode ser um programa relacionado à prestação de serviços sociais (*t-Government*), saúde (*t-Health*), jogos interativos, programas com mais de um final possível, onde o telespectador escolhe um final desejado, quiz, enquetes sobre a programação, venda de produtos (*t-commerce*), entre outros.

Segundo Soares (2009), a interatividade em uma aplicação para TV deve ser utilizada de forma moderada, já que diferentemente de uma aplicação voltada para computador essa deve considerar certos fatores, como:

- A TV em geral, é assistida a uma distância razoável da tela (isso pode não ser verdade no caso de dispositivos portáteis);
- Os dispositivos de interação (controle remoto) são ainda pobres em termos de expressividade e usabilidade;

- O vídeo principal é a principal fonte de sincronismo, incluindo a interação;
- Na maioria das vezes, assistir a um programa de TV é uma atividade coletiva, e por isso, o surgimento de informações adicionais pela demanda de um telespectador pode atrapalhar os outros telespectadores ao lado que não estão interessados nela;
- A TV é usada para lazer e o telespectador não quer nada de muito complexo em seu uso.

Estes fatores limitam as aplicações para TV. Por exemplo, já que a TV é assistida a uma certa distância do telespectador, as aplicações não podem exibir textos longos e com letras pequenas. Além disso, uma vez que a TV é um meio de comunicação popular, com público diversificado, de diferentes níveis de idade, social e de conhecimento, um requisito fundamental para uma aplicação voltada para este ambiente é a usabilidade, visto que aplicações complexas poderiam excluir certos usuários.

O middleware do Sistema Brasileiro de TV digital terrestre, o Ginga, apresenta dois ambientes, o Ginga-NCL, declarativo e o Ginga-J, imperativo. O ambiente declarativo é baseado em uma linguagem declarativa, a NCL, que define uma separação bem demarcada entre o conteúdo e a estrutura de uma aplicação, provendo um controle não invasivo da ligação entre o conteúdo, o leiaute e sua apresentação. (Soares, 2009)

Através de elementos da linguagem, NCL dá suporte a requisitos para aplicações de TV, como suporte a sincronismo espaço-temporal, suporte à adaptação de conteúdo e da forma como o conteúdo é exibido, suporte a múltiplos dispositivos de exibição, suporte à edição em tempo de edição e suporte à definição de relacionamentos de sincronismo espacial e temporal separado da definição do conteúdo dos objetos de mídia relacionados.

O Ginga-J é composto por um conjunto de interfaces de programação de aplicações (API - Application Programming Interface) definidas para atender funcionalidades necessárias ao desenvolvimento de aplicativos para TV Digital, desde a manipulação de dados multimídia até protocolos de acesso(Kulesza et al., 2010). Sua especificação é formada por uma adaptação da API de acesso a informação de serviço do padrão japonês (ISDB ARIB B.23)(ARIB, 2004), pela especificação Java DTV (JavaDTV, 2010) que inclui a API JavaTV (JavaTV, 2008) , além de um conjunto de APIs adicionais de extensão

ou inovação. As APIs adicionais contém um conjunto de classes disponíveis responsáveis por fazer a ligação entre as aplicações NCL e Java, funcionalidades adicionais para sintonia de canais, envio de mensagens assíncronas pelo canal de interatividade e integração de dispositivos externos ao middleware, viabilizando a interação simultânea de múltiplos usuários e dispositivos em aplicações de TV Digital.

2.2 Autoria orientada a template

O aumento da largura de banda média da Internet, viabilizando a transmissão de aplicações, e o avanço do sistema de TV Digital e do serviço de TV por IP (IPTV), são alguns fatores que levam a um crescimento considerável do uso e desenvolvimento de aplicações multimídia, que relacionam diferentes tipos de mídias no tempo e no espaço.

Com isso, surgem novas necessidades, como desenvolver ferramentas e metodologias que possibilitem a autoria de aplicações multimídia de acordo com a necessidade dos diferentes autores.

A autoria orientada a templates é uma metodologia utilizada para criação de documentos hipermídia. Esta abordagem restringe o autor a criar um novo documento de acordo com uma classe específica de aplicações de TV Digital, que são chamadas de arquétipos de documentos ou templates. Neto et al (2008) definem o template como um molde de aplicação com características estruturais e semânticas definidas. Além disso, dividem o processo de autoria utilizando esta abordagem em dois níveis de complexidade, o do autor de templates, que precisa conhecer a linguagem-alvo do documento e identificar estruturas de programas que se repetem, e o do autor de documentos, que é restrito a algumas lacunas que necessitam ser preenchidas para instanciar um template, como ilustra a Figura 2.1.

De acordo com Lima et al. (2009) as estruturas recorrentes comuns em aplicações multimídia podem ser de composição, apresentação ou dos relacionamentos entre as mídias que compõem a aplicação.

Uma aplicação multimídia pode servir como arquétipo para autoria de outras aplicações, desde que forneça um mecanismo para anotar quais são as partes editáveis no documento, também chamadas de lacunas. Além disso, é preciso que o autor consumidor

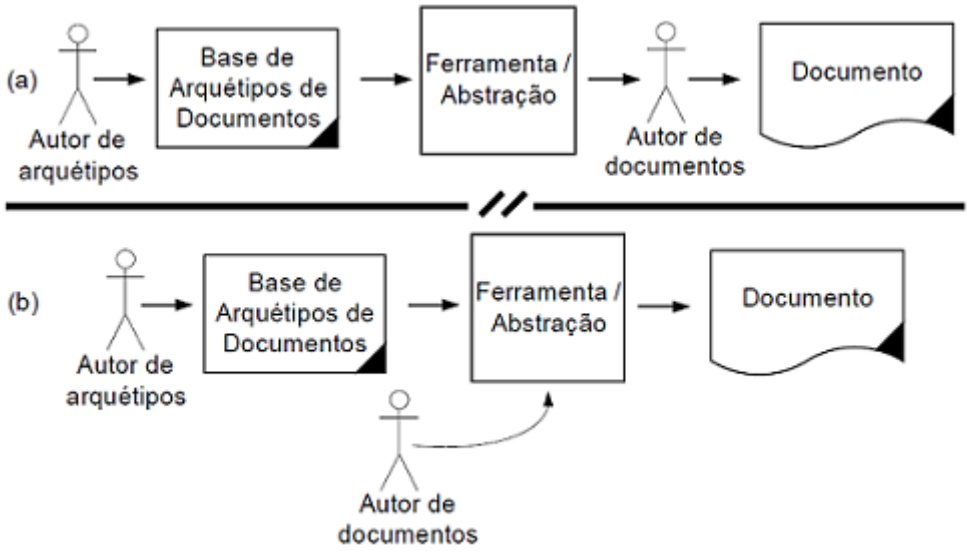

Figura 2.1: Autoria orientada a templates

do template seja capaz de expressar os dados para o preenchimento das lacunas, de modo a torná-lo um documento único.

Existem algumas abordagens para autoria de templates de documentos hipermídia, como as linguagens XTemplate e Template Authoring Language (TAL), descritas nas subseções a seguir.

2.2.1 Linguagem XTemplate

Segundo Santos et al. (2009) um template de composição especifica componentes genéricos e os relacionamentos entre estes, sem identificar necessariamente quem são estes componentes. A composição multimídia que utiliza um template tem, então, a função de identificar tais componentes e relações.

A linguagem XTemplate (Saade et al., 2002; Saade, 2003) em sua primeira versão permitia a definição de componentes presentes na composição ou a serem inseridos pelo template, as relações temporais e espaciais entre eles, e restrições de uso dos mesmos. Estas definições eram divididas em duas partes no template, vocabulário e restrições.

O vocabulário era representado pelo elemento *vocabulary* e era obrigatório na definição do template de composição. Definia os tipos de componentes e seus pontos de interface (elementos *component* e *port*) e os tipos de relacionamentos (elemento *connector*) que uma composição pode possuir, e também os números máximo e mínimo de ocorrências

de cada tipo.

Já as restrições (elemento *constraints*) eram responsáveis por definir restrições adicionais aplicadas aos elementos do vocabulário através de expressões XPath (XPath, 1999), declaração de instâncias de componentes do vocabulário e especificação de relacionamentos entre os componentes usando conectores. A linguagem melhorava o reúso de estruturas nos documentos multimídia, porém tinha como limitação o fato de não permitir a definição de regiões e descritores a serem usados pelos elementos do template e por instanciar apenas nós de mídia, não permitindo a instanciação de elementos compostos.

Na versão seguinte da linguagem, XTemplate 2.1 (Silva, 2005), houve uma considerável redefinição, passando a linguagem a ser dividida em duas partes: o cabeçalho (elemento *head*) e o corpo do template (elemento *body*).

O cabeçalho define o vocabulário, estendendo o elemento da versão anterior permitindo o aninhamento de elementos do tipo *component*, as restrições aplicadas sobre os elementos do vocabulário e as instâncias de elementos, com especificações semelhantes as da versão anterior. Já o corpo do template especificava dois tipos de relacionamentos, possíveis através do uso de transformadas XSL XSL (1999) estendidas a fim de permitir o uso de elementos próprios da linguagem XTemplate (*link*, *bind* e *resource*). A adição de um corpo ao template proporcionou maior expressividade a linguagem e uma maior flexibilidade na criação de relações entre componentes, porém teve como desvantagem o aumento da complexidade da linguagem, devido a necessidade de uso de transformadas XSL.

A versão atual da linguagem, XTemplate 3.0 (Santos et al., 2012), busca manter a simplicidade na especificação e utilização de um template de composição, de modo que a complexidade se encontra no software responsável por realizar o processamento de um template, chamado processador de templates. Além disso, foi introduzida a possibilidade de especificação de descritores e regiões no próprio template. Os descritores são responsáveis por especificar como as mídias serão apresentadas em uma aplicação e as regiões definem o leiaute da aplicação.

A especificação da linguagem XTemplate 3.0 foi feita utilizando uma abordagem modular, com os seguintes módulos:

- Component: módulo que define os elementos “component” e “port”, responsáveis por definir componentes e pontos de interface de componentes de um template e seus respectivos comportamentos.

- Connector: define o elemento “connector”, que identifica um conector para definição de relações entre componentes do vocabulário.

- Constraint: define o elemento “constraint”, usado para descrever restrições sobre elementos pertencentes ao vocabulário, através de expressões XPath (XPath, 1999). Pode conter uma mensagem de descrição de erro caso a restrição não seja satisfeita durante o processamento do template.

- Context: estende a semântica e sintaxe do elemento de mesmo nome na linguagem NCL.

- ImportBase: módulo responsável por indicar a base de importação de documentos com informações de conectores ou descritores, quando desejado.

- Iteration: define os elementos “for-each” e “variable” baseado nos elementos de mesmo nome na linguagem XSLT (XSL, 1999), permitindo a utilização de instruções de repetição para a declaração de instâncias de elementos do vocabulário que ocorrem mais de uma vez. Pode declarar elos, pontos de interface, portas de uma composição e elementos de controle de conteúdo.

- Link: estende a semântica e sintaxe dos elementos “link” e “bind” da linguagem NCL para criação de elos.

- MaxUnion: define o atributo “maxOccurs” usado para definir a quantidade de um determinado elemento permitido em um vocabulário.

- Media: estende a sintaxe e semântica dos elementos “media”, “area” e “property” pertencentes a linguagem NCL, adicionando um atributo de rótulo para cada elemento.

- Port: adiciona atributos ao elemento “port” da linguagem NCL a fim de permitir a definição de rótulos para portas de um contexto e identificação de componentes

através do uso de expressão XPath.

- Structure: define os elementos estruturais de um template: o template em si, seu cabeçalho, vocabulário, corpo e restrições.

- Switch: estende a sintaxe e semântica de elementos de controle conteúdo, através do elemento "switch" da linguagem NCL, adicionando um atributo para definição de um rótulo para o mesmo.

XTemplate 3.0 não pode ser utilizada isoladamente para especificação de um documento multimídia e é dirigida a uma linguagem hipermídia alvo específica focando em criar facilidades para um autor mais especializado. Para que a linguagem de autoria de aplicação utilize um template de composição XTemplate 3.0 é necessário a adição de poucas extensões. Em NCL, por exemplo, estas extensões são fornecidas pelo EXTProfile.

Os templates de composição utilizando a linguagem XTemplate podem ser armazenados em uma base de templates, que pode ser acessada por autores não especializados para a criação de documentos hipermídia. Esta base de templates pode conter tanto templates quanto metadados sobre eles. Usando estes metadados, um template pode ser escolhido automaticamente por uma ferramenta de autoria. A Figura 2.2 ilustra o uso da linguagem XTemplate para a criação de uma aplicação do tipo quiz.

```
<?xml version="1.0" encoding="UTF-8"?>
<xt:xtemplate id="quiz" ...>
<head>
    <connectorBase>
        <importBase alias="connectors" documentURI="connectorBase.ncl"/>
    </connectorBase>
    <descriptorBase>
        <importBase alias="descriptors" documentURI="descriptorBase.ncl"/>
    </descriptorBase>
</head>
<vocabulary>
    <component xlabel="video" descriptor="descriptors#dp_video"/>
    <component xlabel="background" minOccurs="1" maxOccurs="1"
        xtype="image/png" descriptor="descriptors#dp_quiz"/>
    <component xlabel="counter" xtype="application/x-ginga-NCLua"
        minOccurs="1" maxOccurs="1" descriptor="descriptors#dp_lua">
        <port xlabel="answer"/>
        <port xlabel="correct"/>
    </component>
        ...
    <connector src="connectors#onEndTestGTEStopStart"
        xlabel="last_screen_win"/>
    ...
</vocabulary>
<body>
    <port id="port" select="child::*[@xlabel='video']"/>
    <media id="bkg" type="image/png" src="bkg.png" xlabel="background"/>
    <media id="ctrlua" src="main.lua" type="application/x-ginga-NCLua"
        xlabel="counter">
        <property name="answer" xlabel="answer"/>
        <property name="correct" xlabel="correct"/>
    </media>
    ...
    <variable name="i" select="1"/>
    <for-each select="child::*[@xlabel='screen'][position()!=last()]">
        <!-- links for each key -->
        <link xtype="change_screen">
            ...
        </link>
        <variable name="i" select="$i + 1"/>
    </for-each>
    <link xtype="last_screen_win">
        <bind role="onEnd"
            select="child::*[@xlabel='screen'][position()=last()]"/>
        <bind role="test" select="child::media[@xlabel='counter']
            /child::property[@xlabel='correct']">
            <bindParam name="var"
                select="count(child::*[@xlabel='screen'])*0.7"/>
        </bind>
        <bind role="stop" select="child::*[@xlabel='screen']"/>
        <bind role="stop" select="child::media[@xlabel='counter']"/>
        <bind role="start" select="child::*[@xlabel='win']"/>
    </link>
    ...
</body>
</xt:xtemplate>
```

Figura 2.2: Template de composição de uma aplicação de quiz utilizando a linguagem XTemplate

2.2.2 Template Authoring Language

Template Authoring Language (TAL) (Neto, 2010) é uma linguagem declarativa de autoria para templates de documentos hipermídia. A linguagem é uma evolução da linguagem XTemplate (Santos et al., 2009) e permite a especificação de composições hipermídia incompletas independente da linguagem de autoria alvo. Uma composição hipermídia engloba objetos de mídia e outras composições, recursivamente, a semântica de relacionamento entre esses objetos, e é uma das mais importantes abstrações para o autor. De acordo com Neto (2010), uma composição hipermídia incompleta é um documento que contém algumas lacunas a serem preenchidas para tornar esta composição plenamente especificada, e restrições que definem a maneira correta de preenchimento dessa composição.

A linguagem apresenta algumas melhorias em relação à linguagem XTemplate, pois não necessita do 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), diminuindo a necessidade por um autor mais especializado. Adicionalmente, é possível empregar TAL na autoria de outras linguagens declarativas, como SMIL(SMIL, 2008), SVG (SVG, 2000), ou mesmo HTML/ECMAScript.

Neto (2010), define uma composição incompleta utilizando a linguagem TAL por quatro elementos principais, o vocabulário, que descreve tipos de componentes, interfaces e relações; restrições para especificar regras sobre os tipos definidos pelo vocabulário; objetos não editáveis no template, denominados recursos e relacionamentos hipermídia entre tipos, ou entre recursos, ou entre tipos e recursos.

O vocabulário define não só a representação de conjuntos de objetos de mídia, como também impõe uma hierarquia aos componentes e interfaces, de modo que um componente pode conter recursivamente outros componentes e interfaces.

As restrições são utilizadas para verificar se o template foi instanciado corretamente, porque explicitam a cardinalidade dos tipos de componentes, interfaces e relações, e erros no preenchimento das lacunas levam à invalidação das restrições. Os relacionamentos hipermídia de um documento fornecem a sua semântica, e podem se dar entre tipos de elementos definidos no vocabulário, entre âncoras de objetos instanciados ou

entre recursos.

Neto (2010) apresenta os oito módulos que compõem a linguagem. O módulo *Classification* é usado apenas no documento de preenchimento para especificar a extensão ao perfil da linguagem de preenchimento. O módulo *TemplateBase* também é definido apenas para uso pela linguagem de preenchimento e é opcional. Permite a definição da base de templates utilizada pelo documento.

O módulo *Structure* define o elemento raiz da linguagem e corresponde aos templates passíveis de serem referenciados por um documento de preenchimento. O módulo *Importing* permite a importação de templates de outro documento para um documento TAL, de modo que um template pode estender outros templates.

O módulo *Template* define o elemento de mesmo nome e seus atributos, cada template deve ser especificado com um identificador único em um documento TAL. O módulo *Vocabulary* é responsável pela especificação dos tipos de elementos permitidos em um template. O módulo *Constraint* especifica as restrições a serem aplicadas a um template, e especifica testes lógicos que devem ser efetuados para validar uma restrição. Por fim, o módulo *Relationship* permite especificar relacionamentos hipermídia em TAL.

Os relacionamentos em TAL se baseiam nos elos causais utilizados na linguagem NCL (Soares et al., 2005), nos quais uma certa condição deve ser satisfeita para que uma ação resultante seja disparada. A linguagem de relacionamento dá suporte à definição de eventos de apresentação, atribuição e seleção. Os eventos de apresentação estão relacionados com a exibição de conteúdo de uma mídia. Os eventos de atribuição consistem na alteração de uma propriedade da mídia. E os eventos de seleção são aqueles relacionados à interatividade do usuário.

Um template especificado em TAL pode ser instanciado como um documento final através da utilização de um processador TAL, que recebe como entradas a especificação de templates e um documento de preenchimento. É importante destacar que a definição de recursos de um template pode particularizar um especificação TAL para uma linguagem alvo específica, já que o processador TAL simplesmente copia os recursos para o documento final, sem transformá-los. A Figura 2.3 apresenta a especificação completa em TAL de um template “Botão-Texto-Imagem”, contendo um menu de botões que ao serem selecionados

disparam a exibição de um texto e uma imagem respectivos.

Um objetivo alcançado com TAL é a possibilidade de especificar a semântica de uma composição de maneira explícita. Ao definir um template como uma composição hipermídia em aberto, novas composições podem ser criadas através da especialização desta especificação.

```
<?xml version="1.0" encoding="UTF-8"?>
<tal:tal id="templateExample">
<tal:template id="ButtonTextPicture">
 <port id="pButton" component="button[1]"/>
 <port id="pText" component="text[1]"/>
 <port id="pPicture" component="picture[1]"/>
 <tal:component id="button"
                selects="media[class=button]"/>
 <tal:component id="text"
                selects="media[class=text]"/>
 <tal:component id="picture"
                selects="media[class=picture]"/>
 <tal:link id="startAllButtons">
  onBegin button[1] then
  <tal:forEach instance="button" iterator="i">
   start button[i+1],
  </tal:forEach>
  end
 </tal:link>
 <tal:link id="buttonSelection">
  <tal:forEach instance="button" iterator="i">
   onSelection button[i] then
   <tal:forEach instance="text" iterator="j">
    stop text[j], stop picture[j],
   </tal:forEach>
   start text[i], start picture[i] end
  </tal:forEach>
 </tal:link>
 <tal:assert test="#button>0">
  It must be at least one BUTTON element.
 </tal:assert>
 <tal:assert test="#text>0">
  It must be at least one TEXT element.
 </tal:assert>
 <tal:assert test="#picture>0">
  It must be at least one PICTURE element.
 </tal:assert>
 <tal:assert test="#button==#text">
The cardinality of BUTTON and TEXT must be the
same.
 </tal:assert>
 <tal:assert test="#button==#picture">
The cardinality of BUTTON and PICTURE must be
the same.
 </tal:assert>
</tal:template>
</tal:tal>
```

Figura 2.3: Especificação do template “Botão-Texto-Imagem” utilizando a linguagem TAL

2.3 Ferramentas de autoria

A especificação de aplicações para TV digital interativa pode ser realizada através do uso de ferramentas de autoria, a fim de abstrair parte da complexidade envolvida. Deste modo, é importante que uma abstração seja de fácil entendimento, e forneça a maior parte, ou todos os recursos providos pela linguagem de autoria alvo. Abaixo são apresentados alguns trabalhos que buscam facilitar o processo de autoria de aplicações através da proposta de novas ferramentas de autoria.

2.3.1 Composer 3

A ferramenta Composer 3 (Lima et al., 2010) foi desenvolvida com o objetivo de atender os diferentes perfis de usuários potencialmente interessados na criação de aplicações para TV Digital Interativa. Como cada grupo de usuário possui uma expectativa e um ponto de vista distinto em relação ao que se espera de uma ferramenta de autoria de aplicações, o projeto deste tipo de ambiente tem como grande desafio conseguir suprir tais necessidades destes usuários dentro da cadeia de produção e distribuição de conteúdo.

Desse modo, Composer 3 propõe uma base para a construção de um ambiente integrado, capaz de adaptar-se aos diferentes perfis de usuário e oferecer suporte a certos requisitos não-funcionais. Entre os requisitos não-funcionais destacados por (Lima et al., 2010) estão extensibilidade, adaptabilidade, confiabilidade, desempenho, escalabilidade e portabilidade .Sua arquitetura baseia-se em um micro-núcleo, um modelo central e extensões. O micro-núcleo contém as funcionalidades mínimas do sistema e coordena como as extensões devem colaborar. No Composer 3, as extensões das funcionalidade oferecidas pelo micro-núcleo são realizadas através de plug-ins. A Figura 2.4 exibe o funcionamento do micronúcleo, que realiza a comunicação com as outras partes do sistema através da troca de mensagens .

O modelo central representa a aplicação em desenvolvimento pelo autor. Quando este interage com um certo plugin, tal plugin emite um sinal para o núcleo notificando-o sobre as mudanças realizadas. Para que as mudanças sejam efetuadas no modelo central é necessário que o núcleo realize uma validação que não retorne erros. Os plugins só podem realizar operações de consultas ao modelo central, já o micronúcleo é responsável

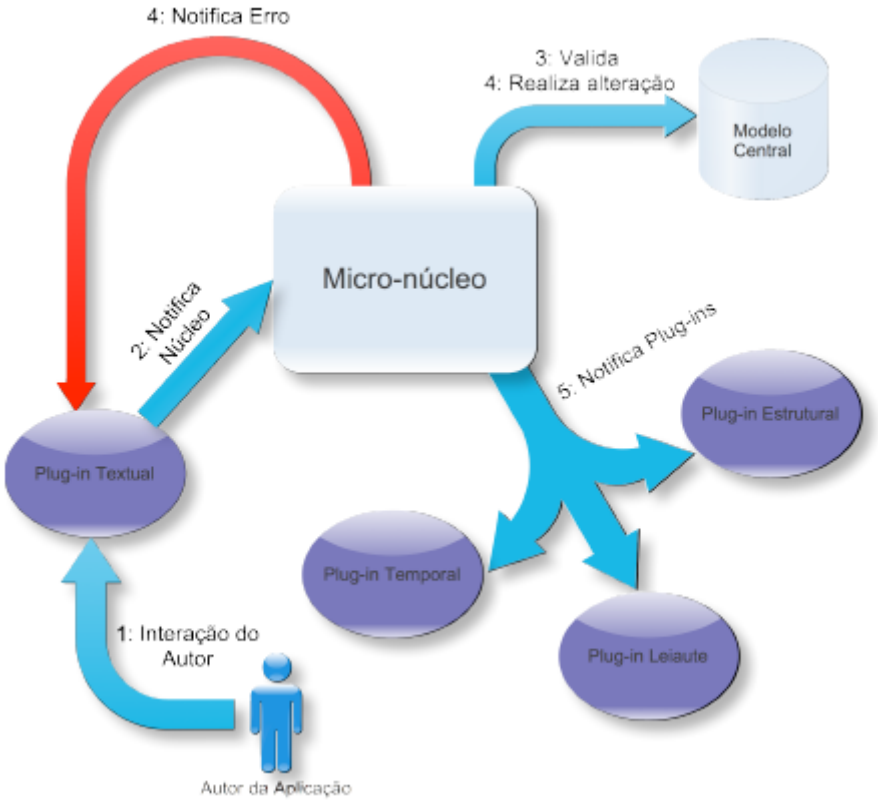

Figura 2.4: Comunicação entre micronúcleo e plugins, retirado de [Lima et al. 2010].

pelas operações de modificações, possibilitando o controle de acesso concorrente ao modelo central.

Para desenvolver um plugin, é necessário implementar duas interfaces: *IPlugin-Factory* e *IPluginMessage*. A primeira define como criar e destruir instâncias do plugin e a segunda trata da comunicação do mesmo com o micronúcleo e da definição do filtro de interesse, que evitam que plugins recebam mensagens relacionadas a elementos da linguagem que não são de seu interesse.

O uso desta arquitetura faz com que o Composer 3 apresente certas vantagens em relação às ferramentas de autoria atuais, como a permissão de sincronismo incremental entre as visões dos diversos plugins, a validação das mensagens trocadas e controle de transação.

Entretanto, o Composer possui como deficiências atuais a falta de suporte ao uso de templates, a impossibilidade de simulação do documento final e a falta suporte à análise do comportamento de execução do mesmo.

2.3.2 EDITEC

O EDITEC (Damasceno et al., 2010) é 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. Ele fornece algumas visões integradas, que possibilitam uma melhor visão global do documento, dando ao autor um amplo controle do template durante sua edição. As visões estrutural, de leiaute e textual são desenvolvidas em módulos distintos a fim de facilitar a manutenção e desenvolvimento do sistema.

Para a criação do template, o editor possui um pacote, que reflete a estrutura da linguagem XTemplate, e realiza o mapeamento de todos os elementos de uma linguagem em classes Java.

O processo de autoria e documentos hipermídia pode levar o autor a ter dificuldades com a quantidade de informação a ser manipulada, e com a localização de cada pedaço da informação. Na maioria dos casos, este problema é causado devido ao elevado número de nós e relacionamentos no documento. Neste contexto, o EDITEC apresenta funcionalidades que facilitam a visão e a busca por determinada informação, como uma visão geral em miniatura da área de trabalho, onde é mostrada a área visível ao usuário e regiões ao seu redor que não estão visíveis.

Na visão estrutural, os elementos gráficos representam as principais entidades definidas na linguagem XTemplate, indicando nós de mídia, elos, conectores, portas e contextos. Já a visão de leiaute apresenta as regiões da tela onde as mídias do template poderão ser exibidas. Nessa visão, o autor pode criar, editar e apagar regiões graficamente. E a visão textual é dividida em duas partes, uma que exibe o código do template e outra, o código da base de descritores, regiões e regras do template.

Através da interface gráfica, não é possível oferecer todas as funcionalidades lógicas permitidas pela linguagem XTemplate 3.0. Desse modo, em alguns casos, é necessário usar a edição textual, o que requer mais conhecimento da linguagem por parte do autor. O editor permite o desenvolvimento de templates, entretanto não oferece a criação de documentos multimídia utilizando templates previamente criados.

2.3.3 Berimbau

Berimbau iTV Author (Berimbau iTV Author, 2011) é uma ferramenta de autoria gráfica para aplicações de TV digital dirigida a profissionais de mídias que não possuem conhecimentos de programação.

A ferramenta oferece um repositório de mídias para uso durante a criação da aplicação e utiliza uma linguagem própria para representação de seu modelo de documento, o qual é utilizado para gerar aplicações finais na linguagem NCL.

O editor não oferece diferentes visões, não consegue contemplar todas as funcionalidades da linguagem NCL, e só permite a edição de documentos criados pela própria ferramenta e somente documentos que contenham mídias de imagens, nenhum outro tipo de mídia é suportado pelo editor. Além disso, não oferece o uso de templates para a criação do documento final.

2.3.4 ISB Designer

O ISB Designer (Araujo, 2012) é uma ferramenta que tem como público-alvo profissionais de cinema e televisão interessados na criação de conteúdo para TV digital interativa. Ela oferece um storyboard interativo capaz de considerar eventos de adaptação, interação e distribuição.

O processo de autoria de aplicações é realizado a partir de estruturas lineares, denominadas sequências, que não suportam eventos de adaptação, interação e distribuição. Estes eventos ocorrem através de elos que podem relacionar várias sequências lineares definidas. As sequências são compostas de painéis, que possuem os objetos que serão utilizados pela aplicação.

A ferramenta permite três tipos de visões, rascunho, autoria e narrativa. Na visão de rascunho é possível a criação de painéis e a especificação da alteração do leiaute da aplicação ao longo do tempo. A visão de autoria refere-se a detalhes de especificação do documento final, como a duração de cada painel e a especificação das condições dos elos, por exemplo. E a visão narrativa exibe a estrutura da aplicação, que permite que o autor navegue mais facilmente pelas sequências e tenha uma visualização geral da organização da aplicação.

2.3.5 NEXT

A proposta do editor NEXT (NCL Editor supporting XTemplate) (Silva, 2012) é facilitar o desenvolvimento de aplicações para TV digital utilizando a linguagem NCL 3.0, através do uso de templates de composição.

A arquitetura da ferramenta baseia-se em um núcleo e diversos plugins, que podem estender o conjunto de funcionalidades do NEXT. Até o momento, existem três plugins que requerem conhecimento da linguagem NCL e um que não requer conhecimento da linguagem, que permite a criação de uma aplicação com a utilização de templates. São oferecidas ao usuário uma visão estrutural, de leiaute e visão textual não editável.

O núcleo da ferramenta é responsável pela interface gráfica do editor e também trata da consistência do documento através da API aNaa - API for NCL Authoring and Analysis (Santos, 2012). O uso da API permite a criação de novos documentos NCL e a edição de qualquer documento NCL, independente de onde este tenha sido criado. Além disso, ela cria objetos Java para representar os elementos de um documento NCL. Com os objetos criados pela API, é gerada uma representação do documento NCL em forma de árvore, a fim de facilitar a indexação dos elementos no documento e a visualização do documento no editor.

O editor não permite a simulação da aplicação em desenvolvimento, e não é capaz de exibir o conteúdo de um documento referenciado pelo documento principal.

3 Solução Proposta

Este capítulo apresenta a ferramenta TVComm Author, proposta para dar suporte ao processo de autoria de aplicações de comércio eletrônico para o ambiente de TV digital interativa, as chamadas aplicações t-commerce. A ferramenta tem como base o conceito de desenvolvimento orientado a templates, apresentado na Seção 2.2.

Sua arquitetura é baseada em módulos funcionais possibilitando sua extensibilidade. Cada módulo é responsável por uma etapa do processo de autoria, como a inserção de elementos de mídia a serem utilizados e o processamento do template.

Como a utilização da ferramenta não exige conhecimento específico em uma linguagem de programação para o desenvolvimento de aplicativos, seu uso é permitido a qualquer perfil de autor. Porém, a elaboração de templates a serem processados pela ferramenta requer conhecimento da linguagem NCL e da extensão acrescida pela solução proposta, apresentada na Seção 3.3.

A ferramenta fornece ao usuário uma visualização de templates de aplicação interativa disponíveis na base de dados de templates e visão da base de dados de produtos.

Além disso, é permitido ao autor visualizar as diferentes fases que compõem a aplicação. Estas fases representam estágios da aplicação, constituídas de elementos de mídias e suas relações de sincronismo temporal e espacial. As fases podem ser editadas pelo usuário através da exclusão e inclusão de objetos de mídia. A ferramenta permite ainda, a criação de uma pasta contendo a aplicação desenvolvida e as mídias que a compõem. Com estes arquivos é possível a execução da aplicação em um exibidor NCL.

As seções a seguir apresentam um levantamento dos requisitos de aplicações de comércio eletrônico para TV Digital, a arquitetura geral da ferramenta, o processo de autoria de aplicações, além da definição da extensão da linguagem NCL desenvolvida e o processador de templates.

3.1 Levantamento de Requisitos para Aplicações de T-Commerce

Aplicações t-commerce podem ser apresentadas de diversas maneiras, como em um canal específico de vendas (loja tradicional de comércio eletrônico, porém pela TV), ou relacionada ao conteúdo de uma programação geral, ou em uma propaganda interativa, entre outros. Podem, também, apresentar diferentes modos de transações de compras, incluindo compra pela TV, via telefone ou pela Web.

Com o objetivo de analisar as diferentes aplicações t-commerce que podem ser desenvolvidas e definidas através de templates, foram enumerados alguns requisitos funcionais, para dois modos de apresentação de uma aplicação tcommerce. Estes requisitos foram obtidos a partir do estudo de aplicações de comércio eletrônico para TV existentes atualmente, e do modelo de negócios de sites de venda disponíveis na Web. Para o levantamento de requisitos foram identificados dois modos de apresentação de aplicação, o t-commerce como canal de vendas e t-commerce relacionado ao conteúdo, que são apresentados nas seções a seguir.

Outra característica a ser considerada por aplicações t-comerce é o meio de transação de compra utilizado por ela, independente de ser um canal de vendas ou aplicação relacionada ao conteúdo. Uma transação de compra em uma aplicação t-commerce pode ser realizada através do próprio aparelho de televisão, por telefone, pela Internet ou por QR code, por exemplo, que é um código de barras bidimensional que pode ser facilmente escaneado usando a maioria dos telefones celulares equipados com câmera fotográfica.

Uma transação de compra realizada através do aparelho de televisão pode utilizar diferentes modelos de pagamento, como (i) pagamento por cartão de crédito, (ii) débito em conta cadastrada ou (iii) pagamento através de contas integradoras, como o PayPal. Caso a aplicação utilize o modelo de pagamento (i), ela deve fornecer ao usuário uma tela para a inserção dos dados do cartão de crédito e suas informações pessoais. No pagamento (ii), a aplicação deve fornecer ao usuário a opção de informar os dados de login e senha cadastrados para o serviço. Para o modelo de pagamento (iii) a aplicação deve oferecer ao

usuário a opção de inserir os dados do cartão da conta integradora e seus dados pessoais, como informação de endereço de entrega.

Uma aplicação que utilize o modo de transação através do telefone, deve oferecer ao usuário um número de telefone pelo qual a compra possa ser realizada, e um código identificador do produto que está sendo vendido, para que o comprador possa informá-lo durante a compra.

Para utilizar o mecanismo de QR code para transação, a aplicação deve fornecer ao usuário a imagem referente ao código de barras relacionado ao produto a ser vendido, posicionado ao lado do mesmo.

Outra opção de transação que pode ser utilizada por aplicações t-commerce é o fornecimento de um cupom de desconto de produto durante a exibição do mesmo na programação do canal. Este cupom pode ser utilizado no site de vendas da empresa anunciante do produto e fornecer desconto na compra do mesmo. Utilizando este modelo, a compra não necessita ser realizada em tempo real de visualização do programa, visto que o usuário pode escolher o momento em que deseja utilizar o cupom disponibilizado.

As aplicações t-commerce podem também utilizar o mecanismo de segunda tela para efetuação de pagamento de compras. Este mecanismo possibilita que a aplicação envie conteúdo para dispositivos móveis conectados à mesma rede que a TV, para que estes interajam com a aplicação.

Existem outros requisitos importantes de aplicações t-commerce, porém a solução proposta concentra foco nos requisitos acima para o desenvolvimento de templates que representam categorias de aplicações t-commerce.

3.2 T-commerce como um canal de vendas

Um canal de vendas consiste em uma loja tradicional de comércio eletrônico, porém via TV, que oferece um catálogo de produtos e serviços ao telespectador. Alguns requisitos funcionais considerados importantes para este modelo de apresentação são:

- RF1: A aplicação t-commerce deve oferecer ao telespectador a opção de visualização de mais detalhes de um produto de sua escolha.

- RF2: A aplicação deve permitir que o telespectador visualize produtos pertencentes à mesma categoria do produto que está em destaque.

- RF3: A aplicação deve oferecer ao telespectador a opção de navegação por categorias de produtos.

- RF4: A aplicação deve possibilitar que o telespectador realize uma busca por um certo produto.

- RF5: A aplicação deve oferecer ao telespectador uma tela para adicionar os produtos escolhidos por ele. Estes produtos são adicionados em uma espécie de carrinho de compras, que armazena os produtos escolhidos pelo telespectador.

- RF6: A aplicação deve permitir que o telespectador remova um item de seu carrinho de compras.

- RF7: A aplicação deve permitir que o telespectador efetue um pedido. Deve ser fornecido ao usuário uma tela para as informações de conclusão de pedido, onde as informações da compra devem ser exibidas.

- RF8: A aplicação deve permitir que o telespectador escolha a forma de pagamento. Quando uma aplicação permitir mais de uma forma de pagamento de compra, estas deverão ser disponibilizadas para escolha.

- RF9: A aplicação deve oferecer a opção de cancelamento de pedido, caso o usuário desista de sua compra.

- RF10: A aplicação deve informar ao telespectador a situação da compra após seu término, informando se a compra foi confirmada ou se houve falha.

3.3 T-commerce relacionado ao conteúdo

As aplicações de comércio eletrônico relacionadas ao conteúdo aparecem em programas regulares de televisão e podem ser disparadas em certos pontos da programação. Neste tipo de aplicação, o produto a ser oferecido pode estar restrito a um tempo de exibição e

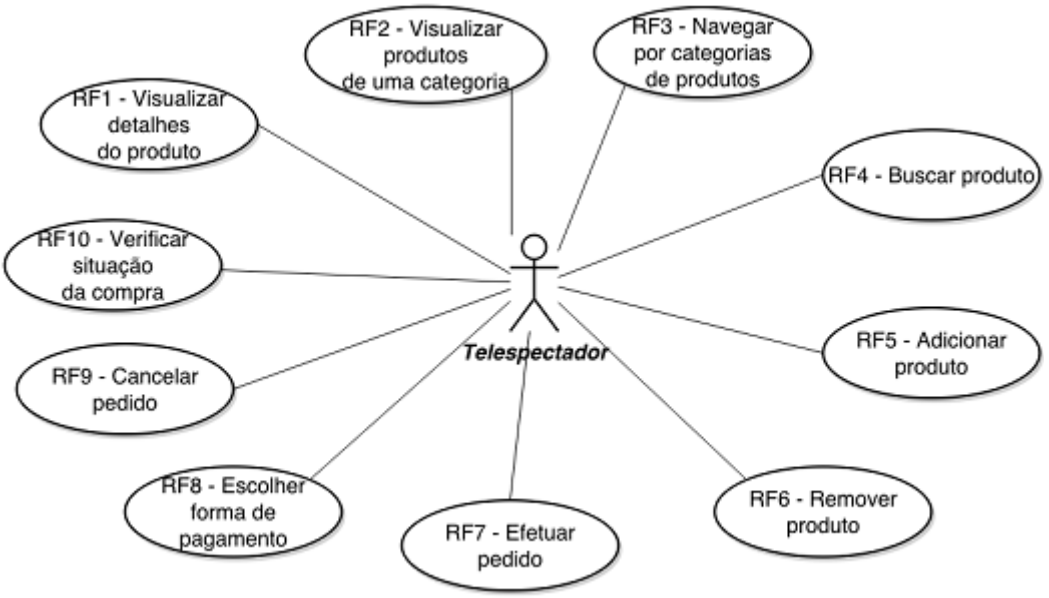

Figura 3.1: Diagrama de Casos de Uso de aplicação como canal de vendas

uma condição de início dentro da aplicação, ou mesmo aparecer por tempo indeterminado na programação. Além disso, a aplicação pode conter um ícone demarcando a opção de interação, de modo que o telespectador tenha a opção de escolha em interagir ou não com o conteúdo que está sendo oferecido. Os requisitos RF1, RF7, RF8, RF9 e RF10 de aplicações do tipo canal de vendas se aplicam também às aplicações relacionadas ao conteúdo. Outros requisitos destacados deste modelo de aplicação são listados a seguir:

- RF11: A aplicação deve permitir que o telespectador selecione um produto durante sua exibição em meio à programação.
- RF12: A aplicação deve oferecer ao telespectador a opção de comprar um produto durante o tempo em que está especificado como disponível.

3.4 Arquitetura Geral da Ferramenta TVComm Author

Lima et al. (2010) levantam alguns requisitos não-funcionais desejáveis em ferramentas de autoria para aplicações hipermídia, que serviram como base para a elaboração da arquitetura da solução proposta. Esta arquitetura é composta de módulos responsáveis pelas principais funcionalidades desempenhadas durante o processo de autoria de aplicações, como apresentado na Figura 3.3.

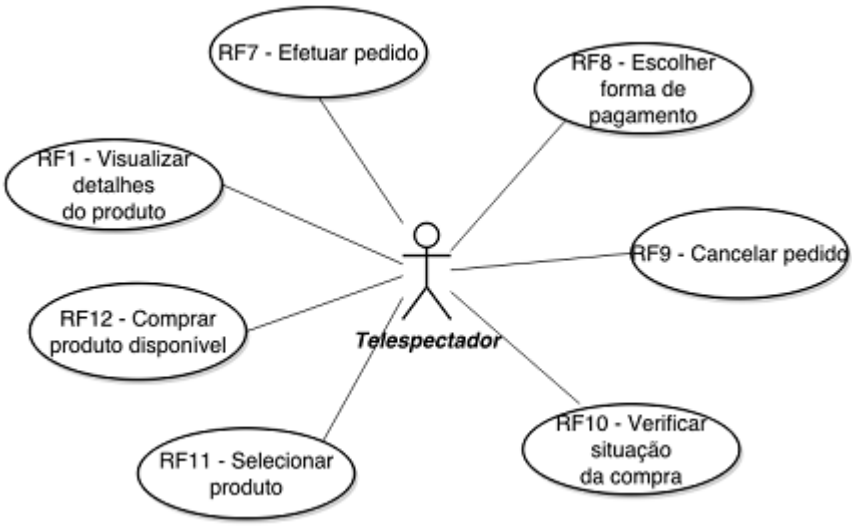

Figura 3.2: Diagrama de Casos de Uso de aplicação relacionada ao conteúdo

O módulo de Mapeamento é responsável por receber uma lista de códigos identificadores de produtos e obter informações sobre eles a partir da base de dados da empresa de comércio eletrônico. Os dados do produto, necessários para a geração da aplicação t-commerce, são capturados e armazenados em um arquivo de descrição de produtos.

O módulo de Edição de Produto permite ao usuário remover produtos da lista de produtos selecionados e alterar as tags de marcação do produto. Estas tags são rótulos do produto que o categorizam e também servem para marcar produtos em oferta ou promoção. As alterações na lista de produtos modificam o arquivo XML de informações de produtos.

O módulo de Processamento modifica um template, acrescentando as informações dos produtos fornecidos, gerando um documento NCL que especifica a aplicação t-commerce.

Por fim, o módulo de Edição de Aplicação é responsável por permitir alterações no documento NCL por parte do usuário. Este módulo oferece navegação pelas fases da aplicação, além de uma interface gráfica para modificações no documento de especificação da aplicação.

3.5 Processo de autoria

O desenvolvimento de uma aplicação t-commerce utilizando a ferramenta proposta segue um processo de autoria composto pelos seguintes passos: seleção de template, inserção

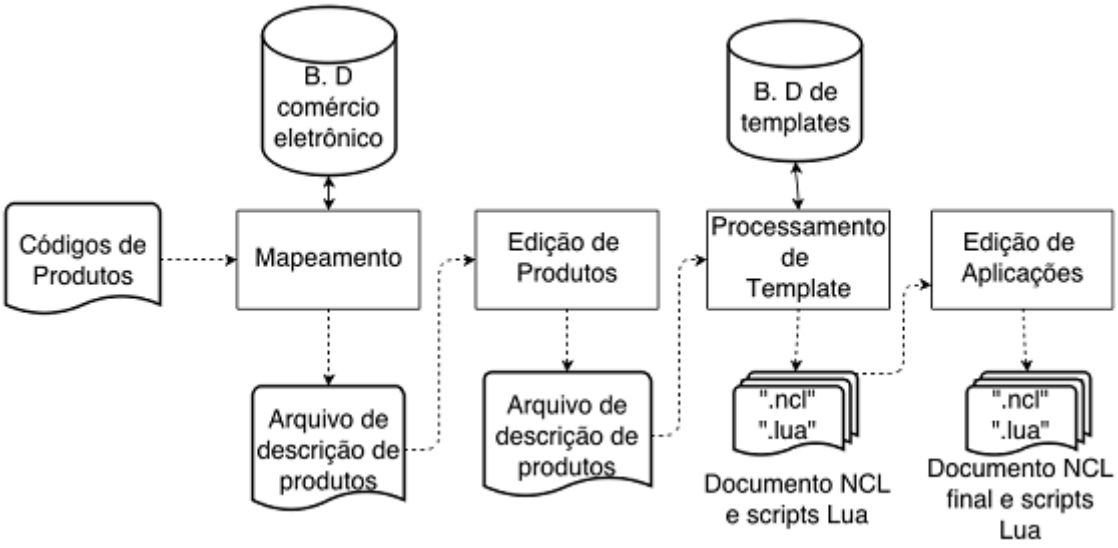

Figura 3.3: Arquitetura geral da ferramenta TVComm Author

de código de produtos, conexão com o servidor da base de dados de comércio eletrônico, edição de produtos selecionados, processamento de template e edição da aplicação.

A ferramenta oferece ao usuário uma lista de templates disponíveis para criação de aplicações t-commerce. Como definido na Seção 2.2, o template pode agrupar classes de aplicações com características estruturais e semânticas definidas. Além disso, o template define os modos de interação permitidos pela aplicação que o utilizará e o relacionamento entre as mídias que o compõem. A especificação do template em uma aplicação t-commerce final é feita através do preenchimento de lacunas. No caso da ferramenta proposta, estas lacunas são preenchidas com informações de produtos fornecidos pelo autor da aplicação.

Para dar início ao desenvolvimento de uma aplicação, é preciso que o autor selecione um template na lista de templates disponibilizados na base. Nesta etapa, o autor pode verificar os diferentes templates contidos na base de templates e informações de que tipo de aplicação ele permite criar. Estas informações estão na descrição do template e podem informar se ele permite a disponibilização da aplicação para múltiplos dispositivos, qual o modo de compra que ele utiliza, etc.

A etapa de inserção de códigos de produtos permite que o autor escolha quais produtos deseja oferecer em sua aplicação. Os códigos de produtos podem ser fornecidos através de um arquivo de texto, ou preenchidos em um formulário pelo autor da aplicação.

Esta opção viabiliza a possibilidade de integrar o processo de concepção de conteúdo da programação para televisão com a tarefa de desenvolvimento de conteúdo interativo, para o caso de aplicações relacionadas ao conteúdo.

Após inserir os códigos de produtos, o usuário necessita se conectar com o servidor da sua base de dados de comércio eletrônico, a fim de obter informações detalhadas de tais produtos como nome, preço, descrição, imagens, entre outros. Desse modo, toda a infra-estrutura de dados de produtos utilizados pelo comércio eletrônico poderá ser reutilizada no comércio através da televisão.

Os códigos de produtos são mapeados para a base de dados e listados para o usuário. A partir desta lista, a etapa de edição de produtos permite que o autor da aplicação selecione quais produtos deseja manter para serem vendidos e quais deseja excluir.

As informações inseridas nas etapas anteriores são utilizadas para a etapa de processamento do template, em que o usuário não realiza nenhuma ação. Nesta fase, o template e os dados de produtos fornecidos são processados a fim de gerar uma nova aplicação. Ao fim da fase de processamento do template, são gerados arquivos na linguagem NCL e mídias necessárias para a reprodução da aplicação em um exibidor da linguagem.

3.6 Extensão ExTN

O desenvolvimento de uma aplicação t-commerce utilizando a ferramenta proposta, ocorre através do uso de templates. Estes templates definem um grupo de aplicações com sintaxe e semântica similares, que se diferenciam entre si, por alguns componentes instanciáveis pelo usuário.

A fim de permitir a criação e o processamento de templates, foi desenvolvida uma extensão à linguagem NCL, denominada ExTN 1.0 (Extension for Template NCL).

As linguagens atuais para criação de templates utilizam o 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. Para utilizar a abstração de fases utili-

zando essas linguagens, uma solução possível é representam cada fase como um contexto implementando um template diferente, e adicionar elos entre estes contextos. Porém, quando mídias se repetem entre fases, estas devem ser repetidas em cada composição, causando uma grande redundância. Desse modo, optou-se por desenvolver uma nova extensão que possibilita o reúso das mídias em diferentes fases da aplicação sem necessidade de repetição, através da definição de elos que disparam a mudança de fases em uma aplicação. Além disso, como a ferramenta TVComm Author é direcionada para o desenvolvimento de aplicações t-commerce, a extensão permite a especificação de elementos que definem produtos e suas características de maneira direta.

Considerando que a extensão desenvolvida é totalmente voltada para o comércio eletrônico, a definição dos elementos que compõem a especificação da linguagem permitem que produtos e suas características expressas de modo direto. Isto é possível atravé do elemento *Products*, que representa uma lista de produtos a serem comercializados via TV. Este elemento possui o atributo *maxOccurs*, que indica a quantidade máxima de produtos permitidos naquele template específico. Durante a exposição de um produto para venda, o autor da aplicação t-commerce pode desejar exibir diferentes características e informações do produto, como sua descrição, diferentes imagens que o representam, ou até mesmo um vídeo explicativo sobre o mesmo.

Desse modo, ExTN define o elemento *Info*, filho do elemento *Products* que identifica o tipo de informação do produto que será utilizada para a construção da aplicação. Este elemento contém os atributos *type* (obrigatório) e *descriptor* (opcional). Os tipos de informação de produto permitidos para o atributo type são: imagem do produto, nome, descrição simples, descrição completa, preço e vídeo representados pelos valores *"image"*, *"name"*, *"shortDescription"*, *"fullDescription"*, *"price"* e e *"movie"* respectivamente. O atributo *descriptor* define o elemento de mesmo nome da linguagem NCL, e identifica o descritor que controla a apresentação da mídia.

A extensão define, também, um atributo para a seleção de um componente de interface (atributo *select*), adicionado ao elemento *Port* pertencente a linguagem NCL, que é um ponto de interface do contexto do documento e é utilizado para indicar qual mídia irá inicializar a aplicação. Este mesmo atributo é adicionado ao elemento *Bind*, para

especificar um componente de mídia, que faz parte de um relacionamento de sincronismo entre mídias (o link). A adição deste atributo aos elementos da linguagem NCL permite a construção automática de elementos na linguagem, que são definidos de acordo com os produtos inseridos pelo autor de aplicação.

A marcação de fases em um template utilizando ExTN 1.0 é feita através da adição do atributo *stage* no elemento *Link* definido na linguagem NCL. Este atributo determina que o elemento *Link* que o contém é responsável por disparar uma nova fase da aplicação, e tem como valor o número da fase que é disparada. O elemento *Link* define relacionamentos entre componentes da aplicação. Estes relacionamentos são do tipo condição-ação, de modo que quando uma condição é satisfeita sobre um componente da aplicação, uma ação é disparada em um ou mais componentes. Portanto, quando o disparo de um elo resulta na modificação das mídias que compõem a aplicação, levará, consequentemente a alteração da fase da aplicação. O que torna a abstração de fases utilizando a extensão, algo simplificado e de fácil entendimento para quem já conhece a linguagem NCL.

Durante o processamento, alguns elementos podem ser criados a partir da iteração sobre os itens da lista de produtos, pelo uso dos elementos *for-each* e *variable*. Estes elementos são baseados nos elementos de mesmo nome na linguagem XSLT (XSL, 1999), que permitem a utilização de instruções de repetição. EM ExTN 1.0, o elemento *for-each* é definido de forma a poder declarar elos (elemento *link*), portas de uma composição (elemento *port*), elementos de controle de conteúdo (elemento *switch*), contextos (elemento *context*), papéis de conectores (elemento *bind*) e regras (elemento *binRule*).

O Apêndice A apresenta um exemplo de uso da extensão desenvolvida para a criação de um template que define uma aplicação com três fases de exibição.

3.7 Processador de Templates

Um documento multimídia que utiliza templates necessita ser processado para que possa ser apresentado em um exibidor da linguagem de apresentação hospedeira. A linguagem hospedeira é aquela que descreve a aplicação, neste trabalho será utilizada a linguagem NCL.

A ferramenta TVComm Author possui um processador de templates integrado, que recebe como entrada um template e arquivos que descrevem as mídias a serem inseridas nas lacunas definidas pelo template, e produz como saída um documento do tipo NCL.

O template, utilizado como entrada para o processador, é fornecido pela base de templates. Esta base é um diretório que contém um arquivo do tipo XML que descreve informações sobre os templates disponíveis, e os documentos que os representam. Dentre as especificações do template, contidas no arquivo XML, estão um identificador único do template, seu nome, descrição, suas mídias constantes (imagens, vídeos, texto, arquivos *".lua"*, arquivos *".ncl"*, etc), entre outras.

Ao receber um template da base, o processador inicia a transformação do mesmo, gerando um documento NCL final que representa a aplicação interativa. Os elementos pertencentes ao cabeçalho (elemento *head*) são copiados, sem alteração, pois definem características constantes do template tais como: a base de regiões, que especifica áreas espaciais onde um nó de mídia pode ser exibido; a base de descritores, que especifica como um nó de mídia será exibido; e a base de conectores, que definem ações que devem ser executadas sobre eventos, quando determinada condição for satisfeita.

Após o processamento do cabeçalho, o corpo do template (elemento *body*) é analisado e transformado. Os elementos *Media* pertencentes ao corpo do template, são adicionados ao documento NCL final.

Durante o processamento, um elemento *Products* se transforma em um ou mais elementos de mídia NCL. O processador analisa o elemento *Products*, verifica a quantidade de produtos máxima que podem ser inseridos (atributo *maxOccurs*), e os elementos *Info* filhos do nó. Cada elemento *Info* pertencente ao elemento *Products* será convertido em um elemento do tipo ¡media¿ no documento NCL final.

Os elementos *For-each* e *Variable* são processados a fim de criar elementos que se repetem de acordo com a ocorrência de outros. Por exemplo, o elemento for-each pode ser utilizado para criar um relacionamento de sincronismo (elemento *link*) para cada produto da lista de produtos.

Em seguida, os relacionamentos (elemento *link*), contextos (elemento *context*),

nós de decisão (elemento *switch*) e as portas de interface são processadas, gerando elementos de mesmo nome, no documento NCL final.

Ao final do processamento do template, é criado um diretório contendo as mídias constantes e instanciadas pelo autor, que compõem a aplicação, um documento NCL principal que contém a especificação da aplicação interativa, e os documentos NCL e scripts Lua utilizados por ela, como mostrado na Figura 3.4.

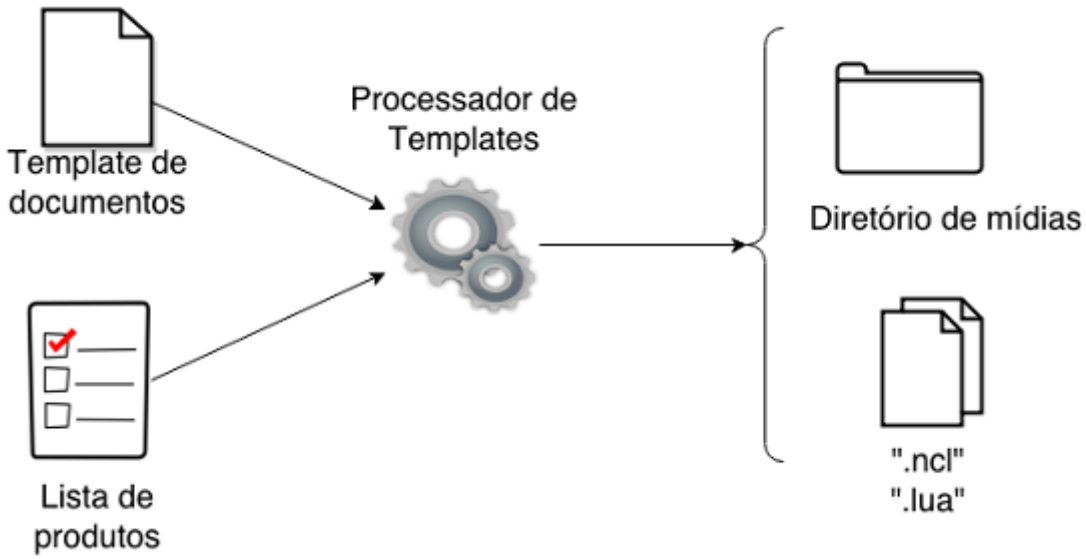

Figura 3.4: Processamento de template de documento

4 Implementação

Este capítulo descreve a implementação dos pacotes desenvolvidos para atender aos requisitos considerados importantes para uma ferramenta de autoria que utiliza o desenvolvimento orientado a templates.

TVComm Author foi desenvolvida utilizando a linguagem Java e API JavaFX para geração das interfaces gráficas de usuário. E para a etapa de implementação foi utilizado o ambiente de desenvolvimento integrado, NetBeans 7.3.1.

O pacote Mapping é responsável por mapear os códigos de produtos inseridos pelo usuário para a base de dados de comércio eletrônico e capturar as informações sobre produtos necessárias ao desenvolvimento de uma aplicação t-commerce. Para isso, ele faz a conexão com a base de dados do cliente e gera um arquivo XML com os dados de produtos, que apresenta a estrutura ilustrada na Figura 4.1.

```
<products>
  <product id="8" name="Adobe Photoshop Elements 7">
    <price>75.0</price>
    <weight>2.0</weight>
    <length>2.0</length>
    <width>2.0</width>
    <height>3.0</height>
    <shortDescription>Easily find and view all your photos</shortDescription>
    <tag id="6" name="computer"/>
    <tag id="8" name="awesome"/>
    <img id="28" src="media/picture28.png"/>
  </product>
</products>
```

Figura 4.1: Documento XML com informações sobre os produtos inseridos pelo usuário

Após a manipulação dos dados, a classe ProcessTemplate do pacote Processor é chamada para realizar o processamento do template. A classe ProcessTemplate analisa o documento ExTN que define o template de aplicação em busca de lacunas a serem preenchidas para especificação da aplicação. Essas lacunas são representadas por elementos Products e pelo atributo “select” nos elementos Bind e BindRule, que estendem elementos de mesmo nome da linguagem NCL, e no elemento For-each (XSL, 1999). O pacote Processor é composto por 3 classes, sendo ProcessTemplate responsável por chamar as

classes ProcessHead, que processa o cabeçalho do documento ExTN e ProcessBody, que processa os elementos pertencentes ao corpo do documento. Cada elemento da linguagem é mapeado em um classe de mesmo nome, que são definidas no pacote NCL. Além disso, a classe gera o arquivo NCL resultante do processamento do template.

O pacote View é responsável pela criação da interface gráfica da ferramenta. Sua classe principal, MainView, inicia a ferramenta de autoria exibindo a janela principal da ferramenta e uma caixa de diálogo em que o usuário deve escolher se deseja criar um novo projeto a partir de um template ou abrir um projeto já existente (Figura 4.2). Caso o usuário selecione a opção de criar um novo projeto, a classe ConfigNewProject é chamada para exibir uma caixa de diálogo com um formulário a ser preenchido pelo usuário com o nome do projeto que será criado e a localização do projeto.

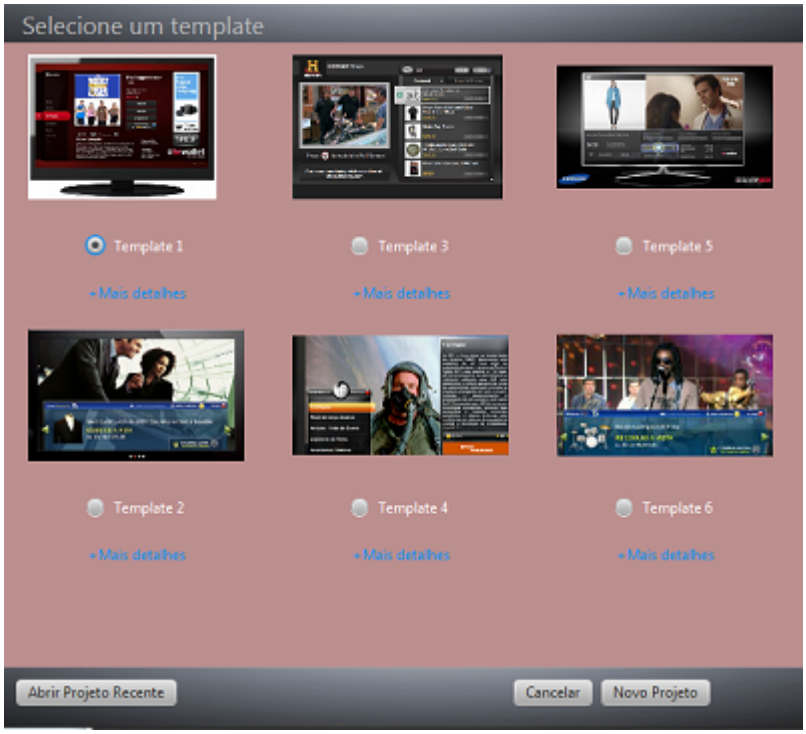

Figura 4.2: Tela de seleção de template para criação de um novo projeto

As classes InsertListProductsView, ConnectBDView, EditProductsView e EditApplicationView pertencem ao pacote View e são responsáveis pela criação das telas de interface gráfica que representam as etapas de criação de uma aplicação.

A classe InsertListProductsView oferece uma interface que permite ao usuário inserir um arquivo com códigos de produtos ou adicionar códigos um por um, na lista de

produtos que serão oferecidos para venda na aplicação, como mostra a Figura 4.3.

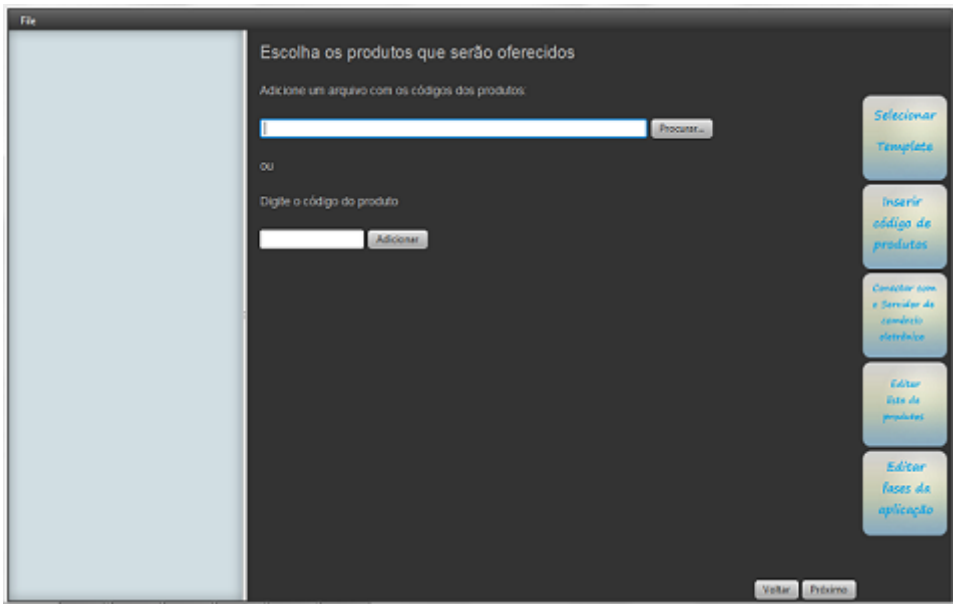

Figura 4.3: Tela de inserção de produtos que serão oferecidos na aplicação

A classe ConnectBDView é responsável por oferecer ao usuário a interface para a conexão com a base de dados do cliente, de acordo com a Figura 4.4.

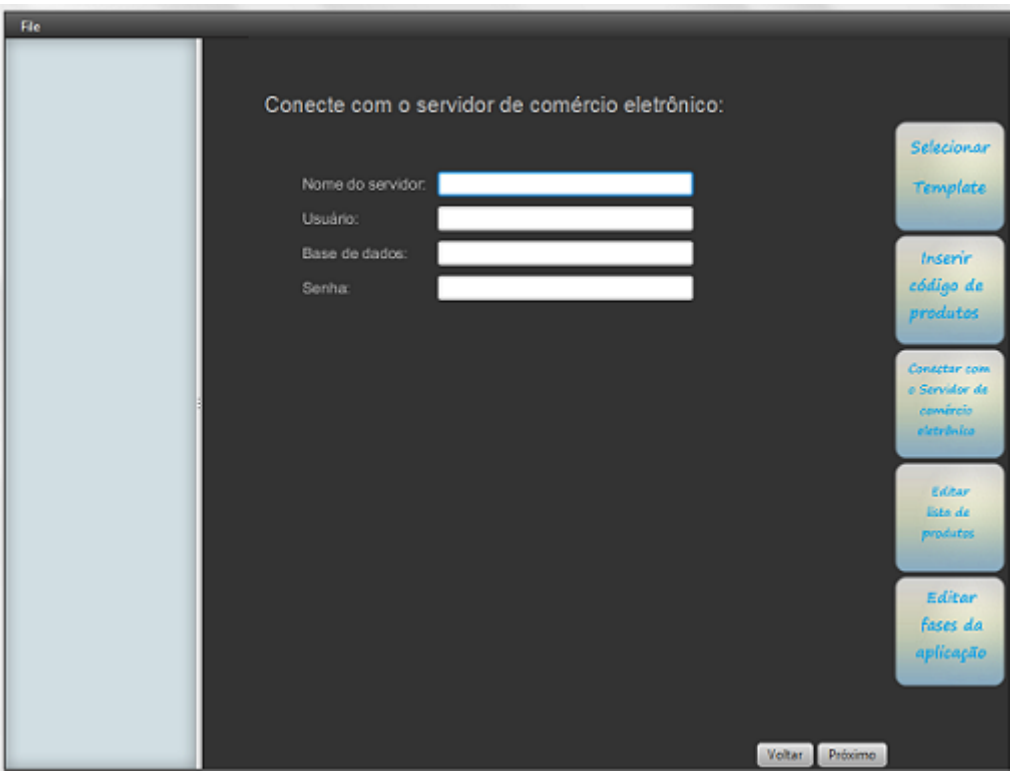

Figura 4.4: Tela de conexão com o servidor de comércio eletrônico

A classe EditProductsView fornece uma interface para permitir ao usuário a edição dos produtos que serão oferecidos na aplicação. Caso o autor deseje eliminar algum produto, este deve ser desmarcado na lista de produtos, como mostra a Figura 4.5.

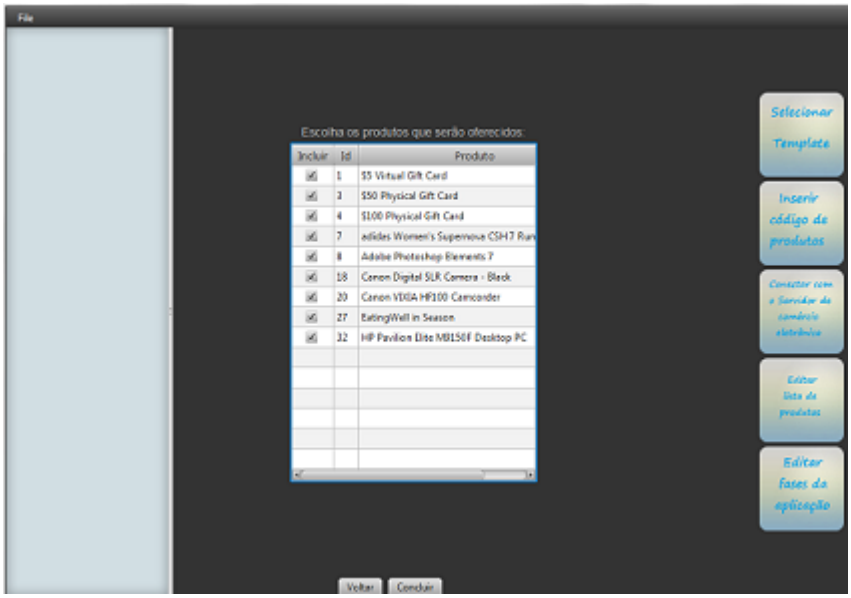

Figura 4.5: Tela de edição de produtos

Por fim, a classe EditApplicationView permite que o usuário refine as fases que compõem uma aplicação (Figura 4.6), excluindo e incluindo certas mídias. O refinamento da aplicação pelo usuário permite que este adicione características próprias que identificam sua marca, como o logotipo da empresa.

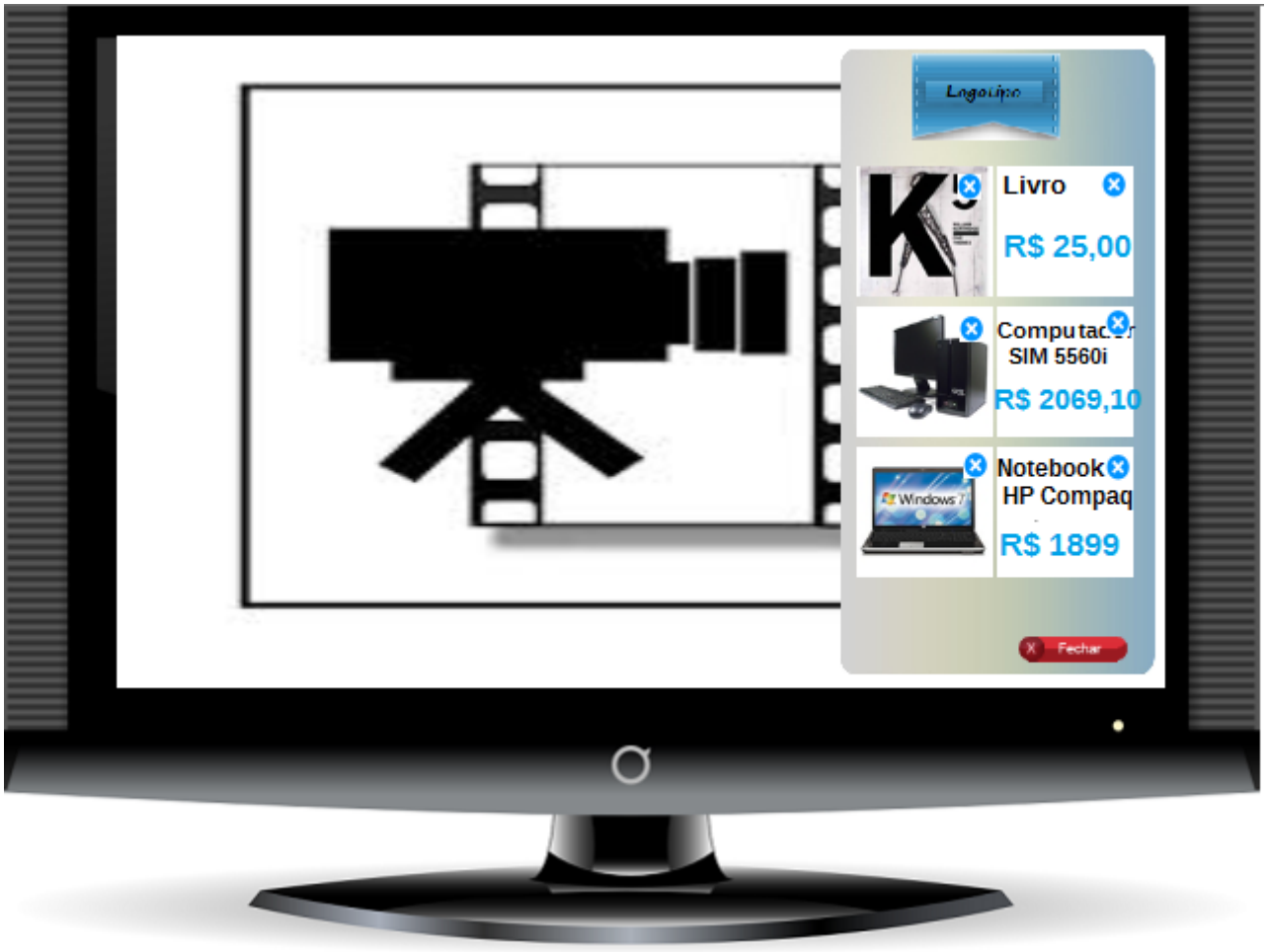

Figura 4.6: Tela de edição de aplicação

4.1 Testes

Com o objetivo de avaliar o atendimento a alguns requisitos funcionais necessários a uma ferramenta de autoria de aplicações t-commerce que utiliza templates, foram executados alguns testes funcionais. Estes testes possuem condições de entrada, resultados esperados após a execução do teste e resultados obtidos, como ilustra a Figura 4.1.

4.2 Avaliação da solução proposta

O conjunto de casos de testes realizados obtiveram resultados que foram ao encontro dos resultados esperados, dando evidências de que a solução proposta atendeu aos requisitos funcionais levantados para as aplicações geradas por este tipo de ferramenta.

Caso de Teste	Resultado Esperado	Resultado Obtido	Requisito Funcional Relacionado
Selecionar um template de aplicação. (Figura 4.2)	Botão radiobutton do template marcado como verdadeiro	Conforme o esperado	RF1: A ferramenta deve permitir que o usuário selecione um template a partir de uma lista de templates disponíveis
Visualizar mais detalhes de um template	Exibição de uma nova tela com as imagens que representam o template, seu nome e sua descrição	Conforme o esperado	RF2: A ferramenta deve permitir que o usuário visualize mais detalhes de um determinado template, a fim de conhecer suas características.
Inserir arquivo correto com códigos de produtos	Arquivo de códigos de produto é validado pela ferramenta (Figura 4.3)	Conforme o esperado	RF3: A ferramenta deve permittir que o usuário informe um arquivo de texto com códigos de produtos que devem ser oferecidos pela aplicação.
Inserir arquivo de códigos de produtos inválido	Exibição de uma caixa de diálogo com a mensagem de erro: "O arquivo especificado não existe"	Conforme o esperado	RF3
Conectar com o servidor de comércio eletrônico	Exibição da tela de edição de produtos	Conforme o esperado	RF4: A ferramenta deve permitir que o usuário se conecte à base de dados do servidor de comércio eletrônico
Conectar com o servidor de comércio eletrônico com usuário inválido	Exibição de uma caixa de diálogo com a mensagem de erro: "Falha de logon do usuário!"	Conforme o esperado	RF4
Conectar a um servidor inválido	Exibição de uma caixa de diálogo com a mensagem de erro: "Falha de logon do usuário!"	Conforme o esperado	RF4
Conectar com o servidor de comércio eletrônico com senha inválido	Exibição de uma caixa de diálogo com a mensagem de erro: "Falha de logon do usuário!"	Conforme o esperado	RF4
Conectar com o servidor de comércio eletrônico com base de dados inválida	Exibição de uma caixa de diálogo com a mensagem de erro: "Não foi possível conectar a base de dados 'X' !"	Conforme o esperado	RF4
Excluir um produto da lista ao desmarcar o checkbox do produto	Exclusão do produto da lista de produtos oferecidos	Conforme o esperado	RF5: A ferramenta deve permitir que o usuário exclua um produto da lista de produtos a serem oferecidos
Excluir mídia de uma fase da aplicação	A mídia é removida do documento NCL, e deixa de ser exibida na tela da fase	Conforme o esperado	RF6: A ferramenta deve permitir que o usuário edite uma fase, excluindo mídias que a compõem

Figura 4.7: Quadro de Casos de Teste da ferramenta

Através do uso da ferramenta o autor de aplicações t-commerce pode gerar uma aplicação de modo simplificado. Durante o processo de autoria utilizando a ferramenta, o usuário só interage nas etapas de seleção de template, adição do arquivo com códigos de produtos, conexão com a base de dados, edição da lista de produtos e edição das fases da aplicação. A etapa de processamento do template selecionado é realizada de forma totalmente automatizada sem a intervenção do usuário. Desse modo, os conhecimentos necessários ao usuário que utilize a ferramenta são básicos, sem exigência de conhecimento na linguagem de programação de especificação de aplicações t-commerce.

A solução proposta apresenta como limitação o fato de não permitir a simulação da aplicação em desenvolvimento. Tal funcionalidade é deixada como um trabalho futuro.

Uma característica importante da ferramenta é possibilitar que esta seja associada ao processo de geração de conteúdo para televisão, por utilizar como entrada um arquivo com códigos de produtos que podem ser obtidos através de métodos de marcação de objetos em vídeo. Estes códigos de produtos são identificadores dos mesmos na base de dados de comércio eletrônico, e podem ser obtidos através de etiquetas de identificador de rádio frequência (RFID) (RFID, 2005), por exemplo, durante a captura do vídeo.

Apesar de algumas limitações apresentadas, a ferramenta possibilita a criação de documentos NCL completos que definem aplicações t-commerce e de acordo com a norma da linguagem.

5 Conclusão

O cenário de desenvolvimento de aplicações para TV Digital necessita de métodos eficientes e fáceis para sua execução, de modo a permitir que autores de diferentes níveis de conhecimento possam participar desta etapa. Com a finalidade de facilitar o processo de autoria de aplicações deste ramo, ferramentas de autoria são desenvolvidas, e oferecem um ambiente gráfico para a criação e edição de aplicações.

Aplicações de comércio eletrônico pelo ambiente da televisão (t-commerce), permitem que telespectadores interajam com a programação e comprem produtos durante sua exibição em uma emissora, de forma prática, através do controle remoto. A aplicação pode ser relacionada ao conteúdo que está sendo transmitido, ou não possuir relação nenhuma.

Desta maneira, o presente trabalho propôs uma ferramenta de autoria para aplicações t-commerce utilizando as linguagens NCL e Lua, baseada no conceito de desenvolvimento orientado a templates. A ferramenta proposta permite que um autor desenvolva uma aplicação a partir de um template que define a estrutura genérica da aplicação, e pode gerar diferentes documentos de acordo com a instanciação de seus elementos em aberto. Além da proposta da ferramenta, o trabalho desenvolveu uma extensão à linguagem NCL, a ExTN, que permite a criação de templates para aplicações t-commerce, com marcação de fases de exibição.

Pode-se destacar como principal contribuição deste trabalho, uma ferramenta que possibilita a automação do processo de desenvolvimento de aplicação em conjunto com a captura de dados durante a gravação de conteúdo. Isto ocorre devido a possibilidade de inserção de códigos de produtos tanto por meio de arquivos, quanto através da adição de produtos individualmente. Além disso, a ferramenta permite a visualização das diferentes fases de exibição de uma aplicação, o que não é visto em nenhum outro editor de aplicações compatíveis com o Sistema de TV Digital Brasileiro.

Outra característica diferencial da ferramenta proposta é o módulo de geração de QR code que permite criar códigos de barra bidimensional para produtos, quando

a aplicação optar por utilizar este modelo de transação de compras. Neste modelo, o telespectador escanea a imagem do código de barras através do seu dispositivo móvel e é redirecionado para o site de vendas que oferece os produtos.

A avaliação da solução proposta foi realizada através de testes funcionais, a fim de detectar problemas em potencial durante o processo de autoria de uma aplicação. Além disso, foram executados testes nas aplicações geradas pela ferramenta, de modo a avaliar o processo de autoria.

Como trabalhos futuros relacionados à geração automática de aplicações para TV digital, pode-se citar (i) extensão da ferramenta para geração de diferentes tipos de de aplicações, não se limitando apenas ao comércio eletrônico, (ii) adição de um plugin que permita uma visão temporal das mídias dentro de uma fase da aplicação, (iii) adição de um plugin para desenvolvimento de templates e (iv) simulação da execução da aplicação NCL.

Como a ferramenta permite a inserção de novos templates para definição de aplicações t-commerce, uma questão que deve ser tratada é o gerenciamento destes templates pela ferramenta, para facilitar a visualização dos mesmos pelo autor, de acordo com as características de aplicação que ele busca. Portanto, outro trabalho futuro que pode ser citado é o desenvolvimento de uma linha de produtos de template, que permite ao autor visualizar as características e variabilidades dos templates oferecidos pela ferramenta.

A fim de validar a usabilidade da ferramenta torna-se necessária a execução de uma prova de conceito, incluindo uma avaliação dos requisitos não-funcionais.

Referências Bibliográficas

of Radio Industries, A.; (ARIB), B. B23 – application execution engine platform for digital broadcasting. **ARIB Standard B23**, p. 1–3, 2004.

Araújo, E. C. **Projetando Aplicações para TVDI atraves de Storyboards Interativos**. 2012. 60p. Tese de Doutorado - Dissertaç ao de Mestrado. PUC-Rio, Rio de Janeiro.

BatuqueTV. **Berimbau itv author**. http://www.batuque.tv/, 2011. acesso em 20-dez-2013.

Damasceno, J. R.; dos Santos, J. A. F. ; Saade, D. C. M. Editec: Editor gráfico de templates de composição para facilitar a autoria de programas para tv digital interativa. **XVI Simpósio Brasileiro de Sistemas Multimídia e Web-WebMedia2010. Belo Horizonte, Brasil**, p. 8, 2010.

JavaTV, A. Java tv specification 1.1-jsr 92. **Sun Microsystems. Disponível em URL: http://jcp. org/en/jsr/detail**, 2008.

Microsystems, S. **Java dtv api 1.3 specification, sun microsystems (2009)**, 2010.

Kuleszao, A.; others. Ginga-j: Implementação de referência do ambiente imperativo do middleware ginga. **WebMedia 2010**, p. 35–42, 2010.

Lima, B. S.; Neto, C. d. S. S. ; Azevedo, R. G. d. A. **Multimedia document authoring based on identification and filling of recurrent structures**. In: Proceedings of the XV Brazilian Symposium on Multimedia and the Web, p. 33. ACM, 2009.

Lima, B. S.; Azevedo, R. G. d. A.; Moreno, M. F. ; Soares, L. F. G. **Composer 3: Ambiente de autoria extensível, adaptável e multiplataforma**. In: XVI Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia 2010)-Workshop de TV Digital Interativa, p. 6, 2010.

ABNT. **NBR 15604: Televisão digital terrestre - Receptores**. Associação Brasileira de Normas Técnicas, Rio de Janeiro, abril 2008.

ABNT. **NBR 15606-2: Televisão digital terrestre - Codificação de dados e especificações de transmissão para radiodifusão digital**. Associação Brasileira de Normas Técnicas, Rio de Janeiro, abril 2008.

ABNT. **NBR 15606-2: Televisão digital terrestre - Codificação de dados e especificações de transmissão para radiodifusão digital**. Associação Brasileira de Normas Técnicas, Rio de Janeiro, abril 2010.

de Salles Soares Neto, C.; Soares, L. F. G. **Autoria orientada a arquetipos para tv digital: uma abordagem restritiva e direcionada**. In: XXXIV Conferencia Latino Americana de Informatica., p. 10, 2008.

Soares Neto, C. **Autoria de Documentos Hipermídia Orientada a Templates**. 2010. 143p. Tese de Doutorado - Doutorado em Informática - Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio).

Neto, C. d. S. S.; Soares, L. F. G. ; de Souza, C. S. Tal-linguagem para autoria de templates de documentos hipermidia. **WebMedia-Simpósio Brasileiro de Sistemas Multimídia e Web (Belo Horizonte, MG 2010)**, p. 8, 2010.

Shepard, S. **RFID: radio frequency identification**. McGraw-Hill New York, 2005.

Bulterman, D. e. a. Synchronized multimedia integration language (smil 3.0). w3c recommendation. **World Wide Web Consortium. Available at http://www. w3. org/TR/SMIL2**, 2005.

Ferraiolo, J.; Jun, F. ; Jackson, D. **Scalable vector graphics (SVG) 1.0 specification**. iuniverse, 2000.

Muchaluat-Saade, D.; Soares, L. F. G. Xconnector e xtemplate: Estendendo xlink para aumentar expressividade e reuso. **VIII Simpósio Brasileiro de Sistemas Multimdia e Hipermídia-SBMídia 2002, Fortaleza, Ceará**, 2002.

Saade, D. C. M. **Relações em Linguagens de Autoria hipermídia: aumentando reuso e expressividade**. 2003. Tese de Doutorado - Doutorado em Informática – Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Rio de Janeiro, Brasil.

dos Santos, J. A. F.; Saade, D. C. M. Linguagem xtemplate 3.0: Facilitando a autoria de programas ncl para tv digital interativa. **XV Simpósio Brasileiro de Sistemas Multimídia e Web (Fortaleza, CE 2009)**, p. 8, 2009.

dos Santos, J. A. F.; Muchaluat-Saade, D. C. Xtemplate 3.0: spatio-temporal semantics and structure reuse for hypermedia compositions. **Multimedia Tools and Applications**, v.61, n.3, p. 645–673, 2012.

dos Santos, J. A. F. **Multimedia and hypermedia document validation and verification using a model driven approach**. 2012. 164p. Tese de Doutorado - DissertaÃ§Ã£o de Mestrado. Universidade Federal Fluminense, Niterói.

Silva, H. **X-SMIL: Aumentando Reuso e Expressividade em Linguagens de Autoria HipermÃdia**. 2005. Tese de Doutorado - Master thesis, Computer Science Department, PUC-Rio, Rio de Janeiro, Brazil.

da Silva, J. V. **Next-editor gráfico para programas ncl com suporte a templates**. 2012. 135p. Dissertação de Mestrado - Mestrado em Redes e Sistemas Distribuídos e Paralelo - Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio).

Soares, L. F. G.; Rodrigues, R. F. Nested context model 3.0: Part 5-ncl. **Relatório Técnico de Pesquisa da série de Monografias do Departamento de Informática da PUC-Rio**, 2005.

Soares, L. F. G. **Programando em NCL 3.0: desenvolvimento de aplicações para middleware Ginga**. 2. ed., Elsevier, 2009, 590p.

Consortium, W. W. W.; others. **Xml path language (xpath), version 1.0. w3c recommendation**, 1999.

Transformation, X. **Version 1.0, w3c recommendation**, 1999.

A APÊNDICE - Template Tcommerce com 3 produtos oferecidos

A fim de exemplificar modelos de templates que podem ser desenvolvidos e processados pela ferramenta, foi criado o template abaixo, que possibilita ao autor exibir 3 produtos para venda, além de mostrar mais detalhes de cada produto.

O template utilizado é composto por três documentos especificados em NCL. O *main.ncl* é o documento principal que comanda a apresentação da aplicação. O arquivo *connectorBase.ncl* contém os conectores responsáveis por definir os relacionamentos utilizados pelo documento principal. E o arquivo *descriptor-regionBase.ncl* contém a descrição das regiões espaciais da tela que as midias da aplicação podem ocupar e os descritores que definem como as mídias serão executadas.

```
1
2  <?xml version="1.0" encoding="ISO-8859-1"?>
3  <ncl id="main" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile"
4                 xmlns:xsl="http://www.w3.org/1999/XSL/Transforms"
5                 xmlns:nmt="http://lapic.ufjf.br/NMT1.0">
6  <head>
7       <ruleBase>
8            <rule id="prodRule1" var="service.currentFocus"
                comparator="eq" value="1"/>
9            <rule id="prodRule2" var="service.currentFocus"
                comparator="eq" value="2"/>
10           <rule id="prodRule3" var="service.currentFocus"
                comparator="eq" value="3"/>
11      </ruleBase>
12
13      <descriptorBase>
14           <importBase documentURI="descriptor-regionBase.ncl"
                alias="desc"/>
15      </descriptorBase>
16
17      <connectorBase>
18           <importBase documentURI="ConnectorBase.ncl" alias="conn
                "/>
19      </connectorBase>
20 </head>
21 <body>
22      <port id="pInicio" component="video"/>
```

```
23
24      <media id="video" descriptor="desc#dVideo" src="media/video.
           mp4"/>
25
26      <media id="iconeVenda" src="media/icone_comprar.png"
           descriptor="desc#dIconeVenda"/>
27
28      <link xconnector="conn#onBeginStartNDelay" stage="2">
29           <linkParam name="delay" value="4s"/>
30           <bind role="onBegin" component="video"/>
31           <bind role="start" component="iconeVenda"/>
32      </link>
33
34      <link xconnector="conn#onKeySelectionStopNStartN" stage="3">
35           <bind role="onSelection" component="iconeVenda">
36                <bindParam name="keyCode" value="1"/>
37           </bind>
38           <bind role="stop" component="iconeVenda"/>
39           <bind role="start" component="contextoVitrine"/>
40      </link>
41
42      <context id="contextoVitrine">
43           <port id="pContV" component="fundoVenda" />
44
45           <media id="fundoVenda" src="media/fundo_venda.png"
                descriptor="desc#dFundoVenda"/>
46           <media id="logo" src="media/logo.png" descriptor="desc#
                dLogo"/>
47           <media id="fecharVitrine" src="media/fechar.png"
                descriptor="desc#dFechar"/>
48           <media id="fecharDetalhes" src="media/fechar.png"
                descriptor="desc#dFechar"/>
49           <media id="fundoDesc" src="media/fundo_desc.png"
                descriptor="desc#dFundoDesc"/>
50           <media id="comprar" src="media/comprar.jpg" descriptor=
                "desc#dComprar"/>
51
52           <nmt:products nmt:maxOccurs="3" >
53                <nmt:info nmt:type="image" descriptor="desc#
                     dProduto"/>
54                <nmt:info nmt:type="name" descriptor="desc#
                     dNomeProduto"/>
55           </nmt:products>
56
57           <link xconnector="conn#onBeginStartN">
58                <bind role="onBegin" component="fundoVenda"/>
59                <bind role="start" component="logo"/>
60                <xsl:variable name="i" select="1"/>
61                <xsl:for-each select="child::body/child::context/
                     child::media[@element='image']">
62                  <bind role="start" select="child::body/
                       child::context/child::media[@element='image
                       '][position()=$i]"/>
```

```
63                  <bind role="start" select="child::body/
                       child::context/child::media[@element='name'][
                       position()=$i]"/>
64                  <variable name="i" select="$i+1"/>
65               </xsl:for-each>
66               <bind role="start" component="fecharVitrine"/>
67          </link>
68
69          <link xconnector="conn#orOnSelectionStopNStartN" stage=
               "4">
70               <xsl:variable name="i" select="1"/>
71               <xsl:for-each select="child::body/child::context/
                    child::media[@element='image']">
72                    <bind role="onSelection" select="child::body/
                         child::context/child::media[@element='image
                         '][    position()=$i]"/>
73                    <xsl:variable name="i" select="$i+1"/>
74               </xsl:for-each>
75               <bind role="stop" component="contextVitrine"/>
76               <bind role="start" component="switchDetalhes"/>
77               <bind role="start" component="comprar"/>
78               <bind role="start" component="fundoDesc"/>
79               <bind role="start" component="fecharDetalhes"/>
80          </link>
81
82          <switch id="switchDetalhes">
83               <bindRule rule="prod1Rule" select="child::body/
                    child::context/child::switch/child::context[
                    position()=1]" />
84               <bindRule rule="prod2Rule" select="child::body/
                    child::context/child::switch/child::context[
                    position()=2]" />
85               <bindRule rule="prod3Rule" select="child::body/
                    child::context/child::switch/child::context[
                    position()=3]" />
86
87          <xsl:variable name="var" select="1"/>
88          <xsl:for-each select="child::body/child::context/
               child::media[@element='image']">
89               <context id="contextDesc">
90                    <nmt:products nmt:maxOccurs="1" >
91                    <nmt:info nmt:type="shortDescription"
                         descriptor="desc#dDescricao"/>
92                    <nmt:info nmt:type="image" descriptor="desc#
                         dProdutoDesc"/>
93                    </nmt:products>
94
95                    <link xconnector="conn#onBeginStart">
96                         <xsl:variable name="j" select="1"/>
97                         <xsl:for-each select="../child::media[
                              @element='shortDescription']">
98                              <bind role="onBegin" select="../
                                   child::media[@element='
```

```
                                    shortDescription'][position()=$j]
                                    "/>
                                <bind role="start" select="../
                                    child::media[@element='image'][
                                    position()=$j]"/>
                            <xsl:variable name="j" select="$j+1"/>
                            </xsl:for-each>
                        </link>
                    </context>
                <xsl:variable name="var" select="$var+1"/>
            </xsl:for-each>
            </switch>

                <link xconnector="conn#onKeySelectionStopN">
                    <bind role="onSelection" component="fecharDetalhes
                        ">
                    <bindParam name="keyCode" value="f"/>
                    </bind>
                    <bind role="stop" component="switchDetalhes"/>
                    <bind role="stop" component="fundoDesc"/>
                    <bind role="stop" component="comprar"/>
                    <bind role="stop" component="fecharDetalhes"/>
                </link>
            </context>

            <media id="fecha" refer="fecharVitrine" instance="instSame"/
                >

            <link xconnector="conn#onKeySelectionStop">
                <bind role="onSelection" component="fecha">
                    <bindParam name="keyCode" value="f"/>
                </bind>
                <bind role="stop" component="contextoVitrine"/>
            </link>
</body>
</ncl>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Generated by NCL Eclipse -->
<ncl id="decriptor-regionBase" xmlns="http://www.ncl.org.br/NCL3
    .0/EDTVProfile">
    <head>
        <regionBase>
            <region id="rgTv">
                <region id="rgVideo" width="100\%" height="
                    100\%" />
                <region id="rgIconeVenda" width="6\%" height=
                    "6\%" left="92\%" top="3\%" zIndex="1"/>
                <region id="rgFundoVenda" top="2\%" height="
                    96\%" left="69\%" width="30\%" zIndex="1" /
                    >
                <region id="rgFundoDesc" top="64.5\%" height=
```

```
                    "32.5\%" width="99\%" left="0.5\%" zIndex="
                    1"/>
11                  <region id="rgLogo" top="7\%" height="4\%"
                    width="15\%" left="76.5\%" zIndex="2"/>
12                  <region id="rgProduto1" top="20\%" height="
                    20\%" left="70.5\%" width="13\%" zIndex="2"
                    />
13                  <region id="rgProduto2" top="41.5\%" height="
                    20\%" left="70.5\%" width="13\%" zIndex="2"
                    />
14                  <region id="rgProduto3" top="63\%" height="
                    20\%" left="70.5\%" width="13\%" zIndex="2"
                    />
15                  <region id="rgNomeProduto1" top="20\%" height
                    ="20\%" left="84\%" width="13\%" zIndex="2"
                    />
16                  <region id="rgNomeProduto2" top="41.5\%"
                    height="20\%" left="84\%" width="13\%"
                    zIndex="2"/>
17                  <region id="rgNomeProduto3" top="63\%" height
                    ="20\%" left="84\%" width="13\%" zIndex="2"
                    />
18                  <region id="rgFechar" top="92\%" height="4\%"
                     width="10.5\%" left="86\%" zIndex="2"/>
19                  <region id="rgComprar" top="92\%" height="4\%
                    " width="10.5\%" left="74.5\%" zIndex="2" /
                    >
20                  <region id="rgDescricao" top="65\%" height="
                    32\%" width="67\%" left="30\%" zIndex="2"/>
21                  <region id="rgProdDesc" top="67\%" height="
                    26\%" width="26\%" left="3\%" zIndex="2"/>
22              </region>
23          </regionBase>
24          <descriptorBase>
25              <descriptor id="dVideo" region="rgVideo"/>
26              <descriptor id="dIconeVenda" region="rgIconeVenda"
                    />
27              <descriptor id="dFundoVenda" region="rgFundoVenda"
                    >
28                  <descriptorParam name="transparency" value="
                    20\%"/>
29              </descriptor>
30              <descriptor id="dLogo" region="rgLogo"/>
31              <descriptor id="dProduto1" region="rgProduto1"
                    focusIndex="1" moveUp="3" moveDown="2"
32                  focusBorderColor="gray" />
33              <descriptor id="dProduto2" region="rgProduto2"
                    focusIndex="2" moveUp="1" moveDown="3"
34                  focusBorderColor="gray"/>
35              <descriptor id="dProduto3" region="rgProduto3"
                    focusIndex="3" moveUp="2" moveDown="1"
36                  focusBorderColor="gray"/>
37              <descriptor id="dNomeProduto1" region="
```

```
                    rgNomeProduto1"/>
38                <descriptor id="dNomeProduto2" region="
                    rgNomeProduto2"/>
39                <descriptor id="dNomeProduto3" region="
                    rgNomeProduto3"/>
40                <descriptor id="dFechar" region="rgFechar"/>
41                <descriptor id="dComprar" region="rgComprar"/>
42                <descriptor id="dDescricao" region="rgDescricao"/>
43                <descriptor id="dProdutoDesc" region="rgProdDesc"/
                    >
44                <descriptor id="dFundoDesc" region="rgFundoDesc">
45                     <descriptorParam name="transparency" value="
                         20\%"/>
46                </descriptor>
47           </descriptorBase>
48      </head>
49 </ncl>
```

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <ncl id="connectorBase" xmlns="http://www.ncl.org.br/NCL3.0/
     CausalConnectorProfile" xmlns:xsi="http://www.w3.org/2001/
     XMLSchema-instance" xsi:schemaLocation="http://www.ncl.org.br/
     NCL3.0/CausalConnectorProfile http://www.ncl.org.br/NCL3.0/
     profiles/NCL30CausalConnector.xsd">
3       <head>
4            <connectorBase>
5                 <causalConnector id="onBeginStartNDelay">
6                      <connectorParam name="delay"/>
7                      <simpleCondition role="onBegin"/>
8                      <simpleAction role="start" delay="$delay" max
                         ="unbounded" qualifier="par"/>
9                 </causalConnector>
10                <causalConnector id="onKeySelectionStopNStartN">
11                     <connectorParam name="keyCode"/>
12                     <simpleCondition role="onSelection" key="$
                         keyCode"/>
13                     <compoundAction operator="seq">
14                          <simpleAction role="stop" max="unbounded
                              " qualifier="par"/>
15                          <simpleAction role="start" max="
                              unbounded" qualifier="par"/>
16                     </compoundAction>
17                </causalConnector>
18                <causalConnector id="onBeginStartN">
19                     <simpleCondition role="onBegin"/>
20                     <simpleAction role="start" max="unbounded"
                         qualifier="par"/>
21                </causalConnector>
22                <causalConnector id="orOnSelectionStopNStartN">
23                     <simpleCondition role="onSelection" max="
                         unbounded" qualifier="or"/>
24                     <compoundAction operator="seq">
```

```
25                          <simpleAction role="stop" max="unbounded
                               "/>
26                          <simpleAction role="start" max="
                               unbounded" qualifier="seq"/>
27                     </compoundAction>
28                </causalConnector>
29                <causalConnector id="onKeySelectionStopN">
30                     <connectorParam name="keyCode"/>
31                     <simpleCondition role="onSelection" key="$
                          keyCode"/>
32                     <simpleAction role="stop" max="unbounded"
                          qualifier="par"/>
33                </causalConnector>
34           </connectorBase>
35      </head>
36 </ncl>
```