Uma Abordagem Dirigida por Modelos para Integração de Aplicações Interativas e Serviços Web: Estudo de Caso na Plataforma de TV Digital

Raoni Kulesza,[1,4]
Silvio R. L. Meira[1]
[1]CIN/UFPE
Cidade Universitária – Campus I
Recife - PE- Brasil -
CEP 50740-560
[rk,srlm]@cin.ufpe.br

Thales P. Ferreira,[2]
Álan Lívio[2], Guido L. S. Filho[2]
[2]DI e [4]DCE/UFPB
Cidade Universitária – Campus I
João Pessoa PB , Brasil
CEP 58059-900
[thales,alan,guido]@lavid.ufpb.br

Manoel C. Marques Neto[3]
Celso A. S. Santos[3]
[3]DMCC/UFBA
Av. Adhemar de Barros s 138
Salvador – BA, Brasil
CEP 40170-110
manoelnetom@ifba.edu.br,
saibel@ufba.br

ABSTRACT
This work proposes a model-driven development approach, which integrates interactive multimedia applications, and Web services. It is based on existing modeling language extended, which integrates modeling concepts for interactive applications and ads support for web services. Three Digital TV applications were modeled and developed. As we show, the evaluation of the approach brought benefits not supported by related works, like requirements structuring and reducing amount of work needed to finalize the code generated.

Categories and Subject Descriptors
H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia Architectures Navigation Theory Multimedia Information Systems.

General Terms
Algorithms, Design, Languages.

Keywords
Model-Driven Development, Templates, Digital TV, Broadband TV, Web Services, Web 2.0.

1. INTRODUÇÃO
O processo de convergência digital pode ser encarado sobre dois diferentes pontos de vista. No primeiro, a convergência é vista como um casamento de tecnologias ou indústrias, por exemplo, através da integração das redes de Internet, telefonia móvel e televisão digital. No segundo, a convergência pode ser vista como uma (re)união de diferentes tipos de mídia por meio de uma tecnologia única. Tal cenário tem favorecido o oferecimento de novos serviços, conhecidos como aplicações multimídia interativas [8].

Por outro lado, paralela a evolução das aplicações multimídia interativas, surgem as aplicações Web com "*Mashups*", que agregam conteúdo ou funcionalidades de outras aplicações ou serviços Web. Tal cenário também pode ser observado recentemente no domínio das aplicações multimídia interativas, através das TVs com acesso direto à Internet, denominadas TV Conectadas (ou *Broadband TVs* [1]). Por exemplo, em 2009, a Yahoo! lançou a plataforma *Connected TV*, na qual o acesso Web é simplificado nos receptores de TV Digital, através de aplicações pré-instaladas, que são executadas simultaneamente com os programas de TV, mas sem interferir no conteúdo de vídeo principal apresentado. Exemplos dessas aplicações são a apresentação de vídeos (*YouTube*), acesso a redes sociais (*News Corp's* e *MySpace*), troca de mensagens usando *Twitter*, compartilhamento de fotos usando o *Flickr* e entretenimento através de jogos sociais.

Uma aplicação multimídia interativa é definida como um software que possui interface com o usuário com pelo menos um objeto de mídia (áudio, vídeo, imagens, animações, gráficos 2D ou 3D) e que estão intimamente vinculados à lógica da aplicação. Isso implica que a relação entre os objetos de mídia pode incluir: (i) criação, exclusão, alteração de objetos de mídia a partir da lógica de aplicação em tempo de execução; e (ii) geração de eventos de objetos de mídia propagados para a lógica da aplicação. Adicionalmente, a concepção dessas aplicações também deve envolver um processo de produção de mídias [5].

O desenvolvimento de aplicações multimídia interativas ainda é um desafio. Além da lógica de negócio, aplicações multimídia tipicamente oferecem uma interface gráfica sofisticada e integrada com objetos de mídia (imagens, gráficos 3D, áudio e vídeo). Consequentemente, o processo de desenvolvimento deste tipo de aplicação deve considerar o envolvimento de especialistas das áreas de projeto de software, projeto de interface gráfica e produção de mídias. Atualmente, é possível identificar uma lacuna na definição de processos, métodos e ferramentas que auxiliem o desenvolvimento sistemático e que integrem esses três aspectos das aplicações multimídia interativas [15].

Uma das metodologias que procura diminuir a complexidade da construção de aplicações de multimídia interativas é o Desenvolvimento Dirigido por Modelos (MDD - *Model-Driven Development*) [15]. O MDD permite a transformação de um modelo de concepção de alto nível em uma descrição do conjunto abstrato de elementos que o compõem e posteriormente a transformação desses elementos em código para uma plataforma específica. Os principais benefícios são: (i) inclusão de uma fase

WebMedia'11: Proceedings of the 17th Brazilian Symposium on Multimedia and the Web. Full Papers.
October 3 -6, 2011, Florianópolis, SC, Brazil.
ISSN 2175-9642.
SBC - Brazilian Computer Society

(projeto) intermediária entre a especificação dos requisitos e a implementação; (ii) facilidade de comunicação devido a existência de representações do sistema em vários níveis e (iii) automatização da geração de código através do uso de motores de transformação entre os modelos e ferramentas de edição dos modelos. Entretanto, a utilização dessa abordagem no domínio de aplicações multimídia ainda é bastante limitada [16].

Alguns trabalhos tratam a questão da modelagem de aplicações multimídia interativas e apontam que a rota mais curta para resolver esse desafio é adaptar algumas das abordagens usadas em modelos de processo da engenharia de software às restrições e peculiaridades da multimídia. Este artigo, está especificamente relacionado ao modelo de processo de software denominado *StoryToCode*. Esse modelo, proposto em [13], trata alguns dos problemas da modelagem de aplicações multimídia interativas como, por exemplo, o reuso de componentes, porém, deixa em aberto um problema clássico da engenharia de sistemas: a estruturação de requisitos.

Uma das soluções adotadas para melhor definir e tratar escopo de requisitos de um sistema é o desenvolvimento baseado em arquétipos, que consiste em analisar um conjunto considerável e aleatório de aplicações desenvolvidas por vários autores, para verificar a repetição de estruturas e comportamentos [17].

Buscando solucionar os problemas relacionados anteriormente, este artigo apresenta uma evolução do modelo *StoryToCode* através de 3 elementos adicionais: (i) classificação de um conjunto de aplicações para especializar os modelos; (ii) utilização de conceitos de autoria orientada a arquétipos para aumentar velocidade, simplicidade e eficiência do desenvolvimento e (iii) emprego de ferramentas de modelagem e geração de código para facilitar a construção das aplicações. O objetivo é melhorar o processo de desenvolvimento através de uma melhor estruturação dos requisitos, que permite: (i) padronizar a estruturação dos requisitos através do uso de arquétipos; (ii) coletar informações de instância importantes para otimizar o processo de geração de código; (iii) diminuir o retrabalho na concepção e codificação final das aplicações interativas e (iv) reduzir a distância entre os profissionais de TV e de software.

Para avaliar a proposta deste artigo, foi realizado um estudo de caso da aplicação da abordagem no domínio de TV Digital Interativa (TVDI). As seguintes etapas foram executadas: (i) definição de uma classificação das aplicações em três categorias (*push, pull, distributed*), incluindo aplicações integradas com serviços Web; (ii) modelagem dos arquétipos a partir dos tipos definidos em (i); (iii) instanciação e geração de código das aplicações para execução em ambiente de produção.

O artigo está organizado da seguinte forma: a próxima seção discorre sobre os trabalhos relacionais. A terceira seção detalha a abordagem proposta. A quarta descreve uma primeira avaliação através de um estudo de caso com três aplicações de TVDI. A quinta e última discute as considerações finais e trabalhos futuros.

2. TRABALHOS RELACIONADOS

Este artigo usou como referência vários trabalhos que representam o estado da arte no desenvolvimento de aplicações multimídia interativas e, particularmente, alguns trabalhos no domínio de TVDI que tratam a autoria de aplicativos. Alguns desses trabalhos são destacados a seguir.

Inspirado no MDD, o *StoryToCode* [13] estrutura os requisitos em um modelo de concepção de alto nível. O processo de transformação desse modelo de alto nível em código, que é o principal objetivo do *StoryToCode*, parte do princípio que este modelo já está pronto. O processo de criação do modelo de alto nível define que: a equipe de engenheiros de software deve traduzir as informações obtidas nos documentos de cenários e *Storyboards* (ambiente de produção de mídia) em um diagrama de elementos (ambiente software). Esta definição, que é tradicionalmente usada para concepção de softwares, não se mostra adequada para a concepção de aplicações multimídia interativas, uma vez que ela não diminui a distância entre profissionais de produção de mídia e software na elaboração dos modelos de alto nível. A principal conseqüência disso é o aumento do retrabalho (pela equipe de software) nos ajustes necessários ao modelo antes de sua aprovação final (pela equipe de produção de conteúdo multimídia).

Outro problema relacionado ao *StoryToCode* está na quantidade de trabalho necessária para finalizar o código gerado a partir do diagrama de elementos criado. Isso ocorre porque, o diagrama de elementos que representa uma abstração dos requisitos da aplicação modelada não contempla as informações de instância. Esse diagrama descreve apenas a estrutura das informações que devem estar presentes em uma aplicação e não dos valores de instância dessa estrutura. Além disso, o *StoryToCode* possui representações apenas de estrutura da aplicação, não tratando outras visões que são essenciais para a elaboração das aplicações multimídia interativas, por exemplo interação e interface gráfica com o usuário. O *StoryToCode* também não modela características de aplicações mais recentes, que utilizam, por exemplo, serviços Web, uma vez que seu modelo tem foco na estrutura da informação e não no tratamento de dependências ou comportamentos externos que possam afetar tal estrutura.

Os trabalhos [15] e [16] têm foco no desenvolvimento de aplicações multimídia interativas por meio da especificação da MML (*Multimedia Modeling Language*). O principal objetivo é permitir que o processo envolva diferentes especialistas pertencentes aos três tipos de projeto existente: projeto de software; interface gráfica com o usuário e produção de mídias. O trabalho aqui proposto utiliza a UML, adotando seus modelos numa abordagem MDD. Além de utilizar a linguagem MML numa metodologia de desenvolvimento, que estrutura melhor os requisitos através do uso de arquétipos e da geração semiautomática de código através de uma ferramenta, também foi possível validar a linguagem em cenários ainda não explorados anteriormente: aplicações de TV Digital integradas com serviços Web.

A Composer [9] é uma ferramenta de autoria hipermídia para a codificação de programas audiovisuais interativos em NCL. Seu principal objetivo é possibilitar que usuários possam produzir código NCL por meio de abstrações visuais. A hipótese é que esta abordagem requer menos conhecimento especializado de NCL. Para que isso seja possível, a filosofia utilizada pelo Composer é a de que o usuário que irá construir o programa poderá fazê-lo através de visões. É importante ressaltar que essas visões não representam os diferentes níveis de abstração encontrados na MDD. Na Composer, cada visão mostra uma parte do programa NCL e, caso uma alteração ocorra em uma das visões, as outras visões são atualizadas automaticamente. Existem quatro visões definidas: Estrutural (exibe, hierarquicamente, os objetos

multimídia que fazem parte de uma apresentação), Temporal (mostra a relação temporal entre os objetos), *Layout* (mostra a relação espacial que existe entre os objetos) e Textual (exibe o código fonte NCL de uma apresentação). O foco da Composer é a geração do código NCL de uma aplicação interativa por "não programadores". Não existe qualquer funcionalidade de apoio ao reaproveitamento dos módulos interativos gerados para outros contextos (Web, móvel e etc.). O uso da ferramenta parte do princípio que todos os requisitos funcionais e não funcionais do programa já estão especificados e não oferece módulos de apoio a esta especificação.

Uma avaliação semelhante à do Composer pode ser feita ao *plug-in* NCL Eclipse [2]. Ele é um editor textual de código livre para a linguagem NCL integrado ao ambiente do Eclipse que também tem o foco em oferecer uma ferramenta de apoio à geração de código fonte seguindo o paradigma WYSIWYG. O *plug-in* também não oferece funcionalidades de apoio ao reaproveitamento de código ou mesmo a especificações de projeto mais abstratas. Dentre as principais funcionalidades implementadas pelo NCL Eclipse estão: coloração de elementos e atributos XML, suporte para que determinados elementos XML sejam escondidos ou exibidos pelo usuário, *wizards* para a criação de documentos NCL simples, autoformatação do código XML, validação do documento NCL, sugestão de código NCL de forma contextual (*autocomplete*), navegação no documento como uma árvore, execução do documento NCL, dentre outras.

3. A ABORDAGEM PROPOSTA

Nesta seção é descrita uma abordagem de desenvolvimento dirigido por modelos, que permite a construção de aplicações multimídia interativas. Inspirada no *StoryToCode* [13], o objetivo é aplicar metodologias de desenvolvimento para um domínio específico de 3 características adicionais: (i) classificação das aplicações (escopo do processo); (ii) utilização de conceitos de autoria orientada a arquétipos (velocidade, simplicidade e eficiência do desenvolvimento) e (iii) emprego de ferramentas de modelagem e geração de código (automatiza parcialmente a construção das aplicações).

A dinâmica da abordagem consiste em partir de um *Storyboard*/Cenário e usar conceitos de autoria orientada a arquétipos e modelagem de sistemas para criar um conjunto de elementos que compõem um programa interativo. Esses elementos representam tanto as mídias, quanto os componentes de software. Essa criação é feita de forma a destacar visões sistêmicas (estrutural, cenas, objetos de mídias, interação do usuário, interface com o usuário e etc.) a fim de tornar o desenvolvimento mais rápido, fácil e eficiente, além de permitir o reuso dos artefatos gerados em outros contextos.

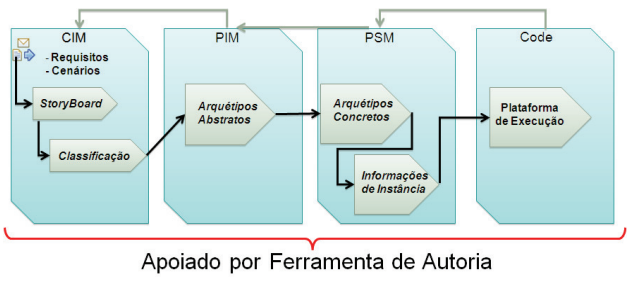

Figura 1 - Visão Geral da Abordagem Proposta.

Assim, como o *StoryToCode*, a arquitetura proposta aqui é inspirada no MDE [14] e está dividida em 4 partes relacionadas entre si, conforme a Figura 1. A dinâmica dessa arquitetura define como deve ser realizada a transformação dos Storyboards/Cenários em um conjunto de elementos abstratos e, posteriormente, a transformação desses elementos em código. É importante observar que as etapas estão em sequência, porém é possível voltar para a etapa anterior em qualquer parte do processo para refinar um modelo definido. A seguir cada etapa da abordagem é descrita.

A primeira etapa tem como objetivo definir os requisitos ou cenários de uso das aplicações, através das descrições de narrativas no formato de um *Storyboard,* que normalmente são definidos pela equipe de produção de mídia (por exemplo, equipe de TV, projetista gráfica de jogos e etc.). A partir daí, é possível definir o escopo através da classificação de um conjunto de elementos de uma aplicação. Os artefatos gerados são conhecidos como modelos CIM (*Computational Independent Model*), uma vez que são independentes de representação computacional.

Na segunda etapa são gerados os *Arquétipos Abstratos*, que representam modelos PIM (*Plataform Independent Model*), ou seja, independentes de plataforma. Tais modelos são elaborados a partir da categorização das aplicações. Dentre as vantagens do uso de arquétipos e a classificação é possível citar: (i) consistência entre as aplicações (*branding*), onde é possível definir e seguir um mesmo padrão para a estrutura, identidade visual e etc; (ii) a definição de um conjunto de aplicações permite facilmente a indexação e agrupamento das aplicações; (iii) como conseqüência de (i) e (ii), existe também a possibilidade de aumentar o reuso; (iv) apesar de demandar um esforço inicial para o estabelecimento de um conjunto comum de aplicações, uma vez realizado este trabalho, o desenvolvimento de aplicações a partir dos arquétipos se torna mais rápido. É importante ressaltar que esse conjunto de elementos não representa apenas um documento para auxílio no entendimento, na manutenção ou evolução do software, usualmente encontrado nos modelos conceituais para especificação de softwares. Esse conjunto é também um artefato que pode ser compilado (transformado) diretamente para modelos PSM (*Plataform Specific Model*). Para que isso seja possível, a construção do conjunto de *Arquétipos Abstratos* deve usar uma notação não ambígua e padronizada a fim de permitir a sua transformação em códigos para plataformas diferentes. Para tanto devem ser utilizada alguma linguagem de modelagem com nível de abstração que seja independente de tecnologia e até mesmo de linguagem de programação. Além disso, devido a necessidade de modelar não apenas o projeto de software das aplicações, é interessante utilizar uma linguagem ou conjunto delas que suportem mais de uma visão sistêmica das aplicações No domínio de aplicações multimídia interativas, sugere-se a adoção da linguagem MML [16], discutida na na seção 3.1..

A terceira etapa consiste em transformar os *Arquétipos Abstratos* em *Arquétipos Concretos*. Estes representam os modelos PSM, que estão associados a algum paradigma e linguagem de programação e são especializações dos arquétipos PIM. Apesar de ser impossível mapear os modelos PIM para todas as tecnologias representadas pelos PSM, a estruturação dos *Arquétipos Abstratos* num conjunto específico e limitado (a classificação) facilita essa transformação. Nesta etapa também são preenchidas as informações de instâncias, uma vez que um mesmo *Arquétipo Concreto* pode gerar inúmeras aplicações diferentes.

A última etapa transforma os *Arquétipos Concretos* com informações de instância para o código-fonte de aplicação e uma determinada plataforma alvo através de uma transformação. Cada instância de aplicação contém um conjunto de regras de transformação, baseadas no conhecimento de origem (*Arquétipos Concretos*), e da estrutura dos elementos de destino (código para uma plataforma específica). Assim, um único conjunto de elementos pode ser transformado em código para plataformas diferentes. Para isso, deve existir um módulo de transformação específica para cada plataforma.

Conforme ilustrado na Figura 1, todas essas etapas devem ser apoiadas pela adoção de uma ferramenta. Essa ferramenta deve oferecer técnicas simples e intuitivas para a construção de uma aplicação de modo a esconder os detalhes de implementação das linguagens de modelagem e programação.

As principais vantagens que podem ser obtidas a partir do emprego desta abordagem são:

- Diminuir a responsabilidade do gerador de conteúdo através da descentralização das etapas de produção que estão fora do seu universo de trabalho original: a especificação e implementação de um artefato de software.
- Permitir a participação de outros atores (analistas de sistemas, programadores, designers e etc.) no processo produtivo de uma aplicação multimídia interativa.
- Diminuir do esforço despendido durante o processo de produção dos componentes interativos.
- Permitir o reaproveitamento dos componentes de software, criados para um domínio para outro domínio.
- Reduzir a quantidade de código que os programadores precisam produzir quando na criação da aplicação.
- Automatizar (parcialmente) a geração de código de aplicações a partir de um conjunto consistente de arquétipos.

3.1 MML: Multimedia Modeling Language

As linguagens de modelagem existentes não são suficientes para o desenvolvimento de aplicações multimídia, pois elas não cobrem nem a integração das mídias nem os aspectos gerais de interface com o usuário. Por outro lado, apesar das abordagens tradicionais do domínio multimídia fornecerem amplo suporte para criação de mídia, elas negligenciam a lógica do aplicativo e os princípios da Engenharia de Software [16].

O desenvolvimento de aplicações multimídia envolve 3 diferentes tipos de projeto: (i) projeto de software, que consiste no desenvolvimento da lógica do aplicativo; (ii) projeto da interface gráfica e (iii) projeto de mídias, pois a criação de objetos de mídia requer normalmente o conhecimento de ferramentas (estudo da cor, forma e etc.) [15].

A linguagem MML procura endereçar a natureza interdisciplinar existente no desenvolvimento das aplicações multimídia. Através de uma linguagem baseada na especificação UML 2.0, a MML procura integrar as diferentes equipes e os artefatos produzidos que permeiam os três diferentes tipos de projeto anteriormente descritos. Para tanto, a MML define cinco tipos de modelos: (i) modelo estrutural, que representa entidades da aplicação (modelo de domínio e objetos de mídia); (ii) modelo de cenas, que auxilia o projeto da interface com o usuário, descrevendo como os objetos de mídia podem influenciar a escolha de diferentes cenas (telas da aplicação); (iii) modelo de interface abstrata, que define para cada cena, os elementos abstratos associados à interface do usuário; (iv) modelo de interface de mídia, derivado de modelo de interface abstrata e objetos de mídia do modelo estrutural , que é responsável pela integração dos componentes de mídia com a interface com o usuário abstrata e (v) modelo de interação, que especifica como os eventos de interface do usuário e os eventos de mídia podem disparar chamadas de operação nas entidades do domínio.

Na abordagem proposta, recomenda-se utilizar MML para gerar "Arquétipos Abstratos", que descrevem conceitos independentes de plataforma e podem ser definidos a partir das informações advindas dos modelos CIM, como o *Storyboard*. Mais detalhes do emprego dessa linguagem na abordagem são descritos a seguir.

4. ESTUDO DE CASO: APLICAÇÕES PARA TV DIGITAL INTERATIVA

Além da melhoria significa da qualidade de som e imagem, um sistema de TV digital interativo permite também a evolução na forma das pessoas acessarem o conteúdo através da possibilidade de integrar softwares e dados aos fluxos audiovisuais transmitidos pelas emissoras. Esta inovação cria oportunidades para a produção e desenvolvimento de aplicações multimídia interativas dos mais diversos tipos, como por exemplo: guias eletrônicos de programas, aplicações de entretenimento (jogos) e educacionais, vídeo sob demanda (VoD), aplicações de comércio eletrônico e governamentais, entre outros. Tais serviços impõem os mesmos desafios de desenvolvimento citados anteriormente, pois demandam a definição de novos processos produtivos que agreguem as características de produção de TV (objetos de mídia) e de software [8].

Esta seção busca avaliar o *StoryToService* no contexto de desenvolvimento de aplicações para TV Digital Interativa, mais especificamente, tratando um caso específico de aplicações: aquelas enviadas aos receptores por uma emissora de TV juntamente com seus programas, utilizando o canal que está sob seu controle e numa plataforma com ou sem canal de comunicação com a emissora ou a Web.

4.1 ARQUÉTIPOS E CLASSIFICAÇÃO

Um fator importante para melhorar o processo de produção das aplicações para TVDI é a redução do escopo da diversidade de possibilidades existentes, até pouco tempo desconhecidas, já que surgem novos formatos e modos de interação a cada dia. Uma vez que é inviável gerar todos os modelos de arquétipos, a abordagem proposta visa organizar as características das aplicações de acordo com uma estrutura que contemple os principais cenários de uso dos serviços de TVDI. Baseado em [3], as aplicações podem ser categorizadas em 3 categorias: *pull*, *push* e *distributed*.

Modo Push: agrupa os programas pertinentes à perspectiva tradicional da TV analógica. Apesar de acessar uma aplicação, o usuário é quase passivo diante do conteúdo produzido e transmitido pela emissora. São consideradas apenas as narrativas lineares e aquelas com possibilidade limitadas de opções. A modificação do desenvolvimento do fluxo da narração (conhecida como ramificação) é obtida com a adição de estórias paralelas dentro de um mesmo vídeo ou utilizando mais de um fluxo na transmissão. No primeiro, a mudança da narração é realizada pela emissora e, no segundo, o telespectador toma a decisão de mudança ao mudar de fluxo para acompanhar outro ponto de vista da estória atual. Nesses cenários, é importante observar que o usuário tem influência limitada para alterar o desenvolvimento da estória assim que ele recebe. Tal modelo permite a navegação e

interatividade local através de dispositivos de interação como controle remoto, teclado ou *joystick*. A resposta da interação com a emissora é sempre indireta, utilizando telefone, SMS, fax, e-mail, entre outros. O conteúdo e os dados da aplicação se limitam ao que foi predefinido pela TV e que foi transmitido através de um canal unidirecional (difusão) controlado pela emissora (daí o nome *push*). Os exemplos desses tipos de aplicações que foram definidos para o estudo de caso são: (1) "*Cul-del-sac*": aplicações com narrativas lineares sem relação temporal com o conteúdo principal[1]; (2) "*TV Voting":* aplicações com interatividade através de canal indireto com a emissora (telefone, SMS, email, site *Web* e etc.); (3) "*Add-On*": aplicações que complementam e são sincronizadas (dependência temporal) e enviadas paralelamente com o conteúdo principal, também conhecidas como "*Enhanced TV"*[4]; (4) "*Y*"": aplicações que alteram a sequência ou o fluxo de exibição do conteúdo principal.

***Modo Pull*:** a diferença deste modo em relação ao anterior está na presença de uma comunicação direta e bidirecional com a emissora. Tal conexão, também conhecida com canal de retorno, pode ser realizada através de várias tecnologias: linha discada, ASDL, Wi-Fi, Wi-Max, 3G e etc. Como conseqüência, existe a possibilidade de entrega de conteúdo individual, diferente do modo *push*, onde o conteúdo interativo é coletivo e todos os usuários recebem as mesmas possibilidades de interação. Tal aspecto tem fundamental importância nas possibilidades de construção das narrativas das aplicações, pois agora é possível criar ramificações bem mais elaboradas. Por exemplo, é possível incluir na execução do programa interativo um desafio, condição de acesso ou obstáculo que deve(m) ser resolvido(s) por ou um ou vários usuários antes que o fluxo previsto na narrativa continue. Outra alternativa é a capacidade de guiar o usuário para caminhos específicos da narrativa limitando o número de opções a seguir de acordo com o resultado da interação com o usuário. Nesse contexto, três opções são possíveis: (i) "labirinto" com caminhos obrigatórios; (ii) "labirinto" com gargalos e (iii) "labirinto" dinâmico. Apesar de permitir formas mais elaboradas para interação com o conteúdo e as narrativas, este conjunto de aplicações reutiliza diversas estruturas existentes nos exemplos da categoria *push*. Porém, todos estes novos caminhos podem ser enviados individualmente pela emissora através do canal de retorno (por exemplo, novas fases de um jogo), economizando recursos do receptor e não limitando a quantidade de opções permitidas. Assim, os tipos de aplicações da categoria *pull* são: (1) "*Cul-del-sac ReturnChannel";* (2) *TV Voting ReturnChannel; (3) Add-On ReturnChannel e; (4) Y App ReturnChannel.*

***Modo Distributed*:** nesta categoria se encaixam aplicações mais recentes que exploram duas peculiaridades encontradas também nas aplicações Web 2.0: (i) localização e processamento distribuído pela Internet ou em outros dispositivos dos elementos que podem compor a aplicação interativa (por exemplo, um serviço Web) (ii) colaboração dos usuários na construção da narrativa. O objetivo da primeira característica é desenvolver aplicações que se beneficiam dos efeitos da rede, para se tornarem tanto melhores, quanto mais populares. Nesse caso, o benefício mais importante é agregar conteúdo, funcionalidades ou plataformas completas já disponíveis na rede. Um exemplo deste tipo de aplicação é a integração de uma rede social com a transmissão de algum evento ao vivo para compartilhar suas opiniões e sentimentos com os amigos enquanto assistem a TV [4]. Para o caso de conteúdo produzido com a participação do consumidor, a colaboração pode ser explícita ou implícita. A primeira lida com sistemas que reagem a ações do consumidor através da manipulação do conteúdo transmitido ou criando elementos adicionais para o canal a partir dessas ações. A segunda, por sua vez, lida com sistemas que observam o consumidor e realiza ações em *background*. Um exemplo do primeiro caso é um sistema que auxilia uma emissora na escolha de um conjunto de saídas a partir de centenas de fluxos ao vivo, cobrindo um evento em tempo real onde o usuário pode atribuir notas ao conteúdo enviado e, dependendo da avaliação, a emissora pode modificar o conteúdo transmitido [11]. Já para o segundo caso, em [10] é descrito o *Emotional TV*, que busca a interação do usuário com a interpretação de emoções do telespectador através de uma bola sensitiva como dispositivo de entrada. Segundo [18], esta categoria de aplicação é caracterizada como aplicações de narrativas evolucionárias, uma vez que o estado final da execução deste tipo de aplicação é não determinístico, já que em nenhum momento do tempo é possível determiná-lo. Neste trabalho, para tornar viável a modelagem deste tipo de aplicação no estudo foram tratadas apenas as aplicações de TVDI que contém um serviço Web integrado ao seu conteúdo e que são originadas a partir das aplicações *pull*. Dessa forma, nesta categoria as aplicações suportadas são as mesmas do que *pull*, mas que podem ter os seguintes módulos: (1) *WS TVConsumer*: consome informações de um serviço Web; (2)*WS TV Producer*: produz informações para um serviço Web e (3) *WS TVProsumer*: produz e consome informações para um serviço Web.

4.2 MODELAGEM DOS ARQUÉTIPOS

O uso de arquétipos torna-se ainda mais importante no desenvolvimento de aplicações para TVDI, visto que o conteúdo não é mais necessariamente linear. Por meio da interatividade, podem existir inúmeras formas de assistir a um mesmo conteúdo. Em outras palavras, o programa não é determinado por uma linha do tempo única, mas por um conjunto principal de eventos e por outros secundários que dependem da interação do usuário. Esta característica demanda um processo de produção que dever ser rápido, eficiente e simples [17].

A velocidade está associada à demanda para atender a característica competitiva e dinâmica da construção de programas interativos. A eficiência está relacionada à necessidade de implementar novas aplicações de forma correta. Já a simplicidade se torna importante para inserir e facilitar a comunicação dos vários perfis profissionais existentes na equipe multidisciplinar envolvida na produção da aplicação. Tais características podem ser atacadas com o uso de arquétipos pré-definidos de aplicações.

Para demonstrar as vantagens da linguagem MML dentro da abordagem proposta, são utilizados exemplos de modelagem dos arquétipos das seguintes aplicações: *TV Voting*, *TV Voting RetunChannel* e *TV Voting WSProsumer* (seção 4.1). Para simplificar, abaixo são descritos os dois últimos casos, já que o primeiro caso é um modelo simplificado do segundo. Por conta da limitação do espaço ilustramos apenas o uso de alguns diagramas.

De acordo o MML, a primeira modelagem a ser realizada é a estrutural, onde se evidenciam os elementos do domínio e as mídias relacionadas. Na Figura 2 é exibido o arquétipo de aplicações *TVVotingRC* que contém uma pergunta (*question*) e vários opções de respostas (*Altenative*) e um resultado (*Result*) sobrepondo o vídeo principal (*PrincipalContent*). O modelo

[1]Normalmente representado pelo áudio e vídeo principal do programação ao vivo ou gravado da emissora.

estrutural permite associar os elementos de mídia (elementos *Video, Imag e Text)* aos elementos do domínio. Outro aspecto importante é a representação do canal de retorno (*ReturnChannel)* permitindo configurar a tecnologia de acesso (*type*) e associá-lo a aplicação de forma modular. Tal elemento não existe no arquétipo *TV Voting*.

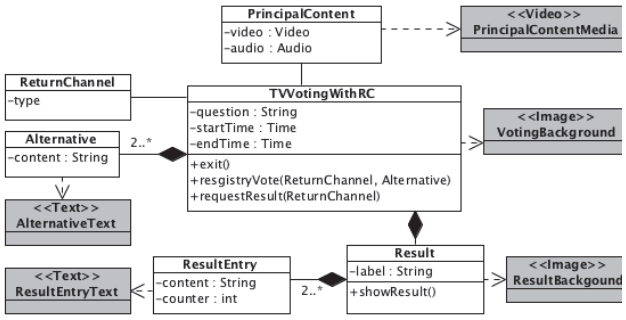

Figura 2 - *TV Voting RetunChannel* - Modelo estrutural.

Para representar a integração dos elementos de mídia e a interação do usuário é utilizado o modelo de interface de mídia (Figura 33), onde uma entidade que deve expressar a confirmação do registro do voto (*TVVotingWithRC.registryVote*) deve ser exibido após um evento de tecla (*KeyPressed*) e escolha de alternativa(s) (TVVotingWithRCAlternatives).

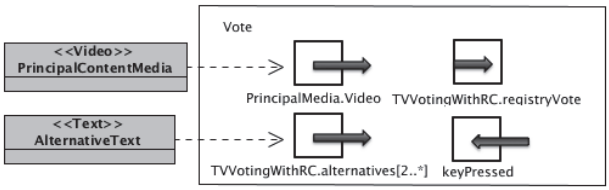

Figura 3 - TVVotingWithRC – Modelo de Interface de Mídia.

O arquétipo das aplicações do tipo *TVVoting Prosumer* consiste numa executar uma aplicação que envia e recebe informações de uma entidades externas representadas por um serviço Web. Um exemplo é um cliente de rede social que permite postar mensagens e ao mesmo tempo recebe atualizações sobre algum assunto de interesse (por exemplo, resultados de jogos ou condição de tempo).

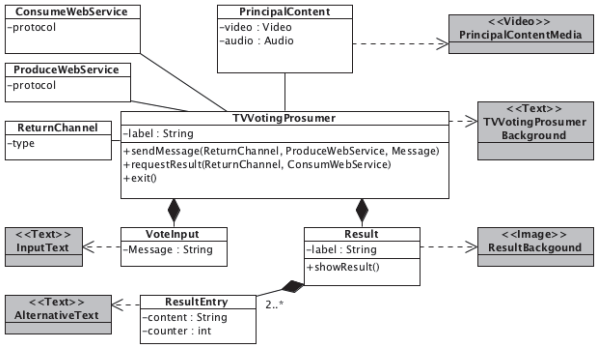

Figura 4 - *TVVoting Prosumer* - Modelo estrutural.

Na Figura 44, além de elementos para entrada da votação (*VoteInput*) e para resultado da votação(*Result*), são evidenciados os elementos relacionados ao acesso ao serviços Web: *ProduceWebsService* para produção de informações e *ConsumeWebService* para consumo. A abstração e modularização desses elementos visam facilitar a customização dos mesmos numa aplicação de TVDI, além de permitir a predefinição de um conjunto de serviços prontos (exemplo, *ProduceWebService* para envio de mensagens para o Twitter, *ConsumeWebService* de recepção de mensagens do *News Corp´s*, entre vários outros).

A Figura 55 evidencia o relacionamento dos elementos de interface gráfica com o fluxo de execução da cena *Vote* (este e outros elementos são representados por um diagrama individual de cenas do MML, não exibido aqui para simplificação). Dois relacionamentos devem ser ressaltados: (i) os elementos de interface gráfica com o usuário (*keyPressed* e *TextInputArea*), que se relacionam com a operação de *inputKeyPressed* respectivamente pare receber e exibir o texto de votação e (ii) a necessidade de relacionamento da operação *sendMessage* com um elemento de interface gráfica (*TVVoting Prosumer*) para confirmação do envio. O *Result* encapsula a imagem (ver *ResultBackGround* da Figura 44) que exibe a resposta do resultado para o usuário.

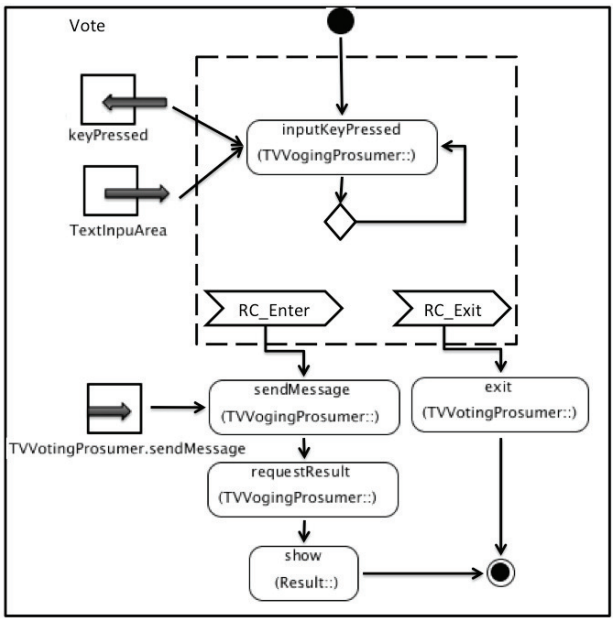

Figura 5 - *TVVoting Prosumer* - Modelo de Interação.

4.3 FERRAMENTA DE EDIÇÃO DE ARQUÉTIPOS

A ferramenta utilizada na abordagem se baseia num trabalho proposto em [6], cujo objetivo era modelar os arquétipos da aplicação com base na técnica de prototipação visual da interface gráfica. Esta técnica tem como principais vantagens a aproximação do sistema com as necessidades dos usuários, a diminuição dos equívocos entre os usuários e desenvolvedores e ainda, a redução no esforço de desenvolvimento [7].

Os arquétipos não contemplam as informações referentes à interface com o usuário (informações de instância). Por exemplo, um diagrama (conjunto de elementos) que represente um arquétipo para uma Enquete descreve a estrutura de elementos como Pergunta e Resposta. Essa descrição pode informar que um elemento Pergunta contém um atributo, que representa o texto de uma pergunta e outro atributo que representa uma lista de

elementos Resposta. Porém, não existem, neste diagrama, informações que definam quais são as perguntas e respostas (instâncias) da Enquete e nem como elas estão formatadas (fonte do texto, posição na tela, cor e etc.). No *StoryToCode* essas informações são informadas diretamente no código, depois da transformação. Nesta proposta essas informações são inseridas nos *Arquétipos Concretos*, facilitando uma avaliação antes da geração do código pela equipe de produção de mídias e interface com o usuário.

Uma vez que o conjunto de elementos foi criado e validado, pode-se continuar com o processo realizando transformação desse conjunto de modelos (*Arquétipos Concretos* com informações de estância) em código para uma plataforma específica. De forma resumida, cada modelo gráfico da ferramenta possui uma representação XMI/XML que é transformada em código para uma determinada linguagem de programação, técnica já utilizada em vários outros trabalhos [13][19]. Neste estudo de caso foram geradas aplicações para o Ginga (Java ou NCL).

A ferramenta combina a visualização do código do conjunto de elementos abstratos e da interface (dos componentes visuais) sob cinco visões distintas e representam a prototipação visual descrita na Figura 66: (i) editor gráfico; (ii) elementos da UI; (iii) barra de propriedades, (iv) informações e (v) editor de código.

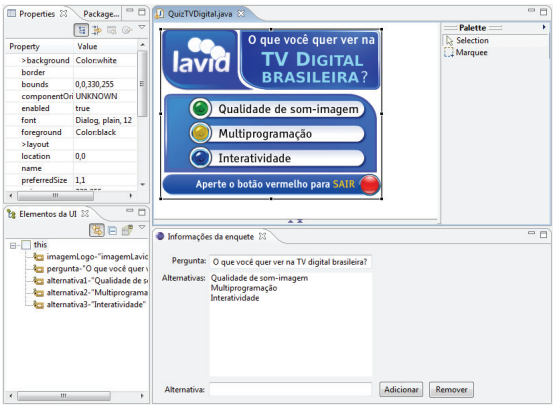

Figura 6 – Ferramenta para edição visual dos arquétipos.

O editor gráfico é a área visual WYSIWYG. A visão de elementos da UI é a árvore hierárquica de componentes visuais e tem como objetivo de auxiliar o processo edição do arquétipo. A visão barra de propriedades exibe as dado do componente selecionado, permitindo editá-las. A visão informações personaliza os dados da aplicação definidos nos arquétipos. E por fim, o editor de código permite exibir, escrever e editar arquivos na linguagem escolhida para a aplicação.

A partir dessas visões foram definidos os principais requisitos funcionais da ferramenta. O desenvolvedor deve poder usar como apoio a barra de propriedades e a *Elementos da Interface gráfica* para alterar os atributos dos componentes da interface. Além disso, a ferramenta deve oferecer a possibilidade de alterar qualquer uma das propriedades dos componentes visuais através do editor de código. Por exemplo, é possível configurar os valores dos atributos dos componentes visuais (ex.: A dimensão de um botão com largura w=30px e altura h=20px, a cor de fundo da cena com valor azul e etc.). Além destes valores visuais, também é possível personalizar os dados específicos do arquétipo. Sendo assim, o usuário informa as alternativas e a pergunta da enquete. Esses valores caracterizam um conjunto de informações de instância, que não são contempladas pelo *StoryToCode*. Com essas informações adicionadas ao conjunto de elementos é possível fazer com que gere o código de um componente interativo especifico para a plataforma de destino escolhida. Esse código é mais completo e gera menor quantidade de (re)trabalho.

A validação da modelagem dos "*Artefatos Concretos*" e geração de código da ferramenta foi realizada através do *deployment* de três aplicações Ginga num ambiente composto por um servidor *Headend* Mux ISDB-Tb da empresa Linear (responsável por enviar o conteúdo audiovisual, metadados e aplicação) e um aparelho de TV modelo 42LH45ED da empresa LG (responsável pela recepção do áudio e vídeo principal e capaz de executar aplicações Ginga-NCL e Ginga-J). Tal cenário proporciona uma condição praticamente idêntica a de um ambiente de produção real.

A primeira aplicação corresponde a uma Enquete *push* sem canal de comunicação direto (as respostas são enviadas por telefone o resultado é enviado pelo canal de difusão para todos com o resultado final da votação) e que representa uma instanciação do arquétipo *TV Voting* (Figura 77).

Figura 7 – Arquétipo de *TV Voting* instanciado.

A segunda aplicação corresponde a uma Enquete *push* e arquétipo *TV Voting ReturnChannel* (Figura 88), onde o canal de comunicação é direto (a resposta de cada usuário é enviada diretamente para emissora por uma conexão à Internet e o resultado parcial é enviado individualmente para cada receptor).

Figura 8 – Arquétipo de *TV Voting* RC instanciado.

A última aplicação permite uma competição de palpites para jogos de futebol (TV Bolão), representando o modo *distributed* e o arquétipo TV *Voting WSProsumer* (Figura 99). No caso, a aplicação contém um serviço Web que envia o palpite para o Twitter e recebe o ranking atual através de outro serviço Web conectado a um servidor de aplicação do TV Bolão.

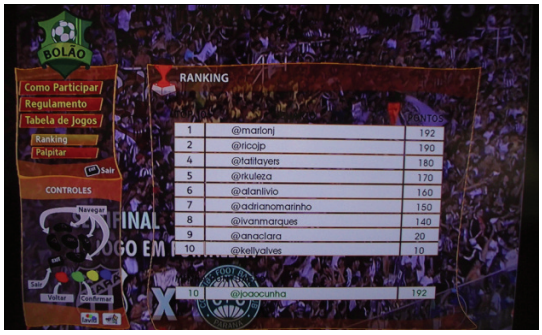

Figura 9 – Arquétipo de *TV Voting Prosumer instaciado*

5. CONSIDERAÇÕES FINAIS

Este trabalho propõe uma evolução do *StoryToCode* para apoiar a construção de aplicações para TVDI a partir de uma classificação dos principais cenários de uso, definição de estruturas pré-concebidas (arquétipos) e uma ferramenta que emprega técnicas de prototipação visual e geração semi-automática de códigos.

A proposta segue uma abordagem MDD para sistematizar a transformação de requisitos de *Storyboard* (CIM) em *Arquétipos Abstratos* (PIM) descritos em MML e, através de uma ferramenta, personalizar os *Arquétipos Concretos* (PSM) para a instanciação das aplicações numa plataforma de execução específica. A adoção da linguagem MML, uma linguagem específica de domínio que tem como principal característica a definição de modelos para múltiplas visões de um software, permite atacar a natureza multidisciplinar das aplicações multimídia interativas.

Como primeira avaliação da abordagem foram desenvolvidos exemplos de uso de três tipos de arquétipos (*TV Voting*, *TV Voting with ReturnChannel e TV Voting with WSProsumer*) na construção de três aplicações para TVDI. Um dos principais benefícios do uso da abordagem foi a melhor estruturação dos requisitos. Por exemplo, a inclusão e configuração de serviços Web integrado com outros elementos da aplicação (lógica de aplicação e objetos de mídia). A principal dificuldade encontrada foi o esforço realizado para a modelagem dos arquétipos. Porém, uma vez finalizada esta etapa, foi possível herdar todos os benefícios da autoria orientada a arquétipos.

No prosseguimento desse trabalho, os seguintes pontos deverão ser atacados: (i) realização de outros experimentos para avaliar quantitativamente (por exemplo: velocidade, eficiência e simplicidade) a utilização da abordagem; (ii) extensão da linguagem de modelagem para tratar a semântica de máquinas sociais [12] e; (iii) emprego da metodologia em outros domínios de aplicações multimídia como forma de avaliar a elaboração de arquétipos e transformação de código em outras plataformas de execução e tecnologias (aplicações móveis e Web).

6. REFERÊNCIAS

[1] Alfonsi, B. I Want My IPTV: Internet Protocol Television Predicted a Winner. *IEEE Distributed Systems Online*, 2005.

[2] Azevedo R.G.A.; Teixeira, M. M.; Soares Neto, S. C. Ambiente Integrado para o Desenvolvimento de Aplicações para TV Digital Interativa em NCL. *In*: SBRC 2009. p53–57. 2009

[3] Bachmayer S.; Lugmayr A.; Kotsis, G. Convergence of Collaborative Web Approaches And Interactive TV Program Formats. *Int. Journal of Web Information Systems*, 2010.

[4] Coppens, T.; Handekyn, K.; And Vanparijs, F. AmigoTV: A Social TV Experience Through Triple-Play Convergence. *Alcatel Technology white paper*. 2005. 4p.

[5] Engels G.; Sauer, S. Object-oriented Modeling of Multimedia Applications, *Handbook of Software Engineering and Knowledge Engineering*, S. K. Chang, Ed. Singapore: World Scientific, v. 2, p. 21–53, 2002.

[6] Ferreira, T. P.; Kulesza, R.; Souza Filho, G. L. iTV Visual Design: Uma Ferramenta para Desenvolvimento Visual de Aplicacões Java para o Padrão Brasileiro de TV Digital. *In*: Workshop de Desenvolvimento Rápido de Aplicações, Curitiba, Jun. 2011.

[7] Kaskalis, T. H.; Tzidamis, T. D.; Margaratis, K. Multimedia Authoring Tools: The Quest for an Educational Package. *Educational Technology & Society*, p.135-162, 2007.

[8] Kulesza, R., Marques Neto, M., Tavares, T. Santos, C. A. S., Souza Filho, G. L. Desenvolvimento de Aplicações Imperativas para TV Digital no Middleware Ginga com Java. *Mini-Cursos do Webmedia 2010*. 1 ed. SBC, v. 1, p. 30-71. 2010.

[9] Guimarães, L. R., Costa, R. M. R.;Soares, L. F. G. Composer: Authoring Tool for iTV Programs. *In* Euro iTV 2008. Lecture Notes In Computer Science, vol. 5066. Springer-Verlag, 61-71, 2008.

[10] Lee, C.J.; Chang, C.; Chung, H.; Dickie, C.; Selker, T. Emotionally reactive television. *In* Proceedings of the Twelfth International Conference on Intelligent User Interfaces, pp. 329-32, 2007.

[11] Mac Williams, C. e Wages, R. *Video conducting the Olympic Games 2008: The iTV field trial of the EU-IST project live. In:* Proc. 3th Int. Conf. on Digital Interactive Media in Entertainment and Arts, pp. 436-40, 2008.

[12] Meira, S. R. L. et al. The Emerging Web of Social Machines: Towards the Design of Sociable Applications. *In* 35th COMPSAC, Jul.2011.

[13] Neto, M. C. M. e Santos, C. A. S. StoryToCode: A new model for specification of convergent interactive digital TV applications. *Journal of the Brazilian Computer Society*. p3-21, Oct.2010.

[14] Perovich, D., Bastarrica, M. C., and Rojas, C. *Model-Driven approach to Software Architecture design*. *In*: ICSE Workshop on Sharing and Reusing Architectural Knowledge. p.1-8. 2009.

[15] Pleuss, A. e Hussmann, H. Model-Driven Development of Interactive Multimedia Applications with MML. *Studies in Computation Intelligence*. Springer: v. 340, 199-218, 2011.

[16] Pleuss A.; Vitzthum A.; Hussman H. Integrating Heterogeneous Tools into Model-Centric Development of Interactive Applications. *In:* MoDELS, p.241-255, 2007.

[17] Soares Neto, C. S.; Soares, L.F.G.; de Souza, C.S. TAL – Linguagem para Autoria de Arquétipos de Documentos Hipermídia. *In*: Webmedia 2010. Belo Horizonte, SBC. v.1. Pp.147-154, 2010.

[18] Ursu, M.F. et al. Interactive TV narratives: opportunities, progress, and challenges. *In:* ACM Trans. on Multimedia Computing, Communications, and Applications (TOMCCAP), v. 4, n.4, 2008.

[19] Yang, X., Gu, P., and Dai, H. Mapping Approach for Model Transformation of MDA Based on XMI/XML Platform. *In*: 1st Int. Workshop on Education Technology and Computer Science. p.1016-1019. 2009.