StoryToCode: Um Modelo Baseado em Componentes para Especificação de Aplicações de TV Digital e Interativa Convergentes

Manoel C. Marques Neto
DMCC - UNIFACS
Rua Ponciano de Oliveira, 126
Salvador - Ba, Brasil CEP 41950-275
manoelnetom@gmail.com

Celso A. S. Santos
DMCC - UFBA
Av. Adhemar de Barros s/n sala 138
Salvador - Ba, Brasil CEP 40.170-110
saibel@ufba.br

ABSTRACT
This paper presents a model, called the StoryToCode, which allows the specification of IDTV programs with focus on the use of software components. First, the StoryToCode allows the transformation of a storyboard in an abstract description of an element set that make up the storyboard. After this, the StoryToCode transform these elements in a specific programing language source code. In StoryToCode a software component is treated as a special element that can be reused in other contexts (web, mobile and etc). The StoryToCode is based on MDA (Model Driven Architecture) and allows design and implement of an application, independent of context, taking into account the particularities of an IDTV program.

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
MDA, Transformações, TVDI, Storyboard

1. INTRODUÇÃO
A disseminação das tecnologias digitais nas diversas áreas do conhecimento levou a modificações na execução da maioria das atividades da sociedade atual, tais como o trabalho, a educação, a saúde e o lazer. O primeiro impulsionador deste fenômeno, a Internet, cresceu a partir da constatação de que não se trata apenas de uma ferramenta para o uso exclusivo da comunidade científica, mas de um veículo que permite, de forma convergente, a execução de diversas atividades cotidianas tais como: ir ao banco, falar ao telefone ou até assistir filmes, animações e etc. Outro veículo que também tem potencial para desempenhar papel importante no uso dessas tecnologias é a de TV Digital. Dentre os principais benefícios proporcionados pela TV Digital estão melhoria na qualidade da imagem e do som e a inclusão de novos, serviços tais como a TV Digital Interativa (TVDI). A TVDI cria a possibilidade de veiculação de softwares interativos e personalizados junto com qualquer outro programa da grade de uma emissora [5].

WebMedia'09, October 5–7, 2009, Fortaleza, CE, Brazil.

Atualmente, conceber uma atração de TV significa de forma genérica a produção e composição dos fluxos elementares de áudio e vídeo do programa. Uma das principais características desse modelo de concepção é a forte centralização, no gerador de conteúdo, das etapas de produção, composição, empacotamento, e distribuição das mídias que fazem parte do programa. O uso desse modelo centralizador no contexto da TVDI dificulta a participação no processo produtivo de "atores" que não estão inseridos usualmente ao universo da TV como engenheiros de software, programadores e etc. Outros problemas do uso desse modelo são a sua baixa flexibilidade e extensibilidade que geram re-trabalho na concepção de programas diferentes para plataformas diferentes.

O problema anterior é semelhante ao enfrentado pelas aplicações Web. Originalmente, os módulos que compunham uma aplicação Web deveriam ser sempre especificados, implementados e executados em conjunto. Em um site de comércio eletrônico, por exemplo, tanto o módulo "carrinho de compras" quanto o de "venda por cartão de crédito" deveriam ser especificados e implementados dentro de um mesmo contexto. A extensão da aplicação para tratar uma interface de utilização diferente (requisito não funcional) como, por exemplo, um celular, significaria refazer boa parte do sistema. Isso, de forma semelhante às aplicações de TVDI, também gerava forte centralização, baixa flexibilidade / extensibilidade e aumento do re-trabalho.

A solução adotada na Web e em outros casos semelhantes foi aplicar técnicas de engenharia de software focadas na solução deste tipo de problema como, por exemplo, o desenvolvimento baseado em componentes (DBC). O DBC se preocupa com a criação de componentes de software que possam ser reutilizados em outras aplicações. O conceito de reutilização está relacionado ao uso de produtos de software, sem acoplamento, a um domínio específico. Para atingir esse objetivo, um requisito fundamental no DBC é a aplicação

de um processo sistematizado de concepção desses componentes. Este processo considera o item reutilização em todas as fases de desenvolvimento do software. Dessa forma, o DBC oferece métodos, técnicas e ferramentas que suportam desde a identificação e especificação dos componentes, referente ao domínio de um problema, até o seu projeto e implementação em uma linguagem executável [1].

A reutilização de softwares originados no ambiente de TV em outros contextos é algo que deve a ser considerado a partir do momento em que uma radiodifusora tem possibilidade de usar canais de comunicação dentro dos seus programas de TV (ex.: Sites Web, Mensagens SMS etc). Por exemplo, em um programa com participação ativa do telespectador, no qual ele deve responder a uma enquete, é muito comum que sejam oferecidas diversas formas de interação como: "Acesse o site wwww.MinhaTV.com.br e responda a enquete", "Mande uma mensagem SMS para o número 8888 e responda a enquete" ou ainda "Aperte o botão azul do controle remoto e responda a enquete". Nesse exemplo a aplicação tem as mesmas funcionalidades nas três situações. A única diferença está no tratamento dado ao requisto não-funcional contexto.

A possibilidade de introduzir elementos interativos em um programa de TV sob a forma de componentes de softwares deve mudar a forma como esses programas são concebidos. Entretanto, a definição de uma metodologia formal e padronizada que permita modelar e construir um programa de TVDI voltado ao uso de componentes de software ainda é um desafio. Alguns trabalhos tratam a questão da modelagem de aplicações como [4] [14] e apontam que a rota mais curta para resolver esse desafio é usar um modelo para o desenvolvimento, baseado em metodologias tradicionais da engenharia de software. Especificamente, o modelo definido em [14] propõe a integração das atividades inerentes ao processo de produção de TV e software.

Buscando contribuir para a solução do problema de modelagem de aplicações para TVDI este artigo apresenta um modelo denominado StoryToCode. Este modelo permite especificar programas de TVDI voltados à utilização de componentes de software. O StoryToCode 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. No StoryToCode um componente de software (aplicação) é tratado como um elemento especial que pode ser reutilizado em outros contextos (outras plataformas de execução como Web, móvel e etc). O StoryToCode é inspirado no MDA (*Model Driven Architecture*) [2] e permite projetar e implementar uma aplicação, independente do contexto, levando em consideração as particularidades de um programa de TVDI. O objetivo do StoryToCode é 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. Espera-se, como conseqüência da sua utilização, permitir a participação de outros atores no processo produtivo de um programa de TVDI e a diminuição do esforço despendido durante esse processo e ainda, o reaproveitamento dos componentes de software gerados em domínios diferentes da TV.

O artigo está organizado da seguinte forma: A seção 2 trata da produção de softwares para TVDI. A seção 3 apresenta o StoryToCode modelo proposto nesse trabalho. A quarta seção mostra como o StoryToCode pode ser utilizado para a especificação de uma enquete no contexto da TVDI e da Web. A última seção é reservada às conclusões do trabalho.

2. PRODUÇÃO DE SOFTWARES PARA TV DIGITAL INTERATIVA

A TV Digital representa uma maior possibilidade de democratização na geração e distribuição de conteúdo. Isso se reflete no aumento (potencial) do número de canais, na possibilidade de permitir que um programa possa ter seu conteúdo adaptado a um contexto específico (usuário, receptor, região e etc) e de oferecer programas interativos (TVDI) entre outras.

Em termos computacionais, um programa de TVDI é um software, especificamente uma aplicação multimídia, através do qual um telespectador pode interagir via controle remoto [10]. Isso significa que um telespectador pode receber do difusor, além do vídeo/aúdio, softwares que possibilitam que ele interaja com o conteúdo veiculado.

De acordo com [7], os softwares para TVDI podem ser categorizados em:

1. Softwares que não têm relação com a semântica do conteúdo de áudio e vídeo apresentados, por exemplo: e-mail e TV-Banking;

2. Softwares que têm relação com a semântica do conteúdo de áudio e vídeo apresentados, mas sem restrições fortes de sincronização, por exemplo: A cotação das ações da bolsa durante um programa de economia e;

3. Softwares que têm relação com a semântica do conteúdo de áudio e vídeo apresentados exibidos de forma sincronizada, por exemplo: anúncios interativos de produtos exibidos em momentos específicos de uma transmissão. Esta última categoria pode ainda ser subdividida em:

   (a) Softwares com a semântica do conteúdo de áudio e vídeo conhecido a priori;

   (b) Softwares com a semântica do conteúdo de áudio e vídeo gerado ao vivo.

O processo de construção de softwares da primeira categoria não se diferencia dos processos tradicionais. Assim, a única diferença de um software de TVDI para os demais está no tratamento dado aos requisitos não-funcionais como, por exemplo, "plataforma de execução" (TV), "mecanismos de entrada e saída" (controle remoto) e etc.

Já a construção de softwares para TVDI pertencentes às outras categorias é feita respeitando particularidades do ambiente de TV. Essas particularidades influenciam diretamente o processo de produção e por isso, essas duas categorias devem receber dos projetistas um tratamento diferenciado. Entre as principais particularidades podem ser citadas:

1. Os softwares são parte de um programa de TV tradicional e esses, por sua vez, tem formato e contexto próprios.

2. Eles utilizam como única interface o receptor de TV que é de uso tradicionalmente coletivo.

3. Eles exigem infra-estrutura de transmissão e componentes de software/hardware adequados para o seu funcionamento.

4. Eles modificam os programas de TV tradicionais de forma a capacitá-los para lidar com diferentes níveis de interatividade e com uma organização do conteúdo não-linear.

Um processo de produção de um software específico para TVDI deve considerar tanto estas particularidades como também fornecer um suporte diferenciado a cada uma delas. O problema é que este processo não é usual nem para a indústria de TV, que não tem cultura no desenvolvimento de softwares, nem para indústria de software, que não tem cultura de desenvolvimento de conteúdo multimídia para TV.

O trabalho proposto em [14] aborda, especificamente, as etapas conceituais da modelagem de aplicações para TVDI. Ele apresenta um modelo para o desenvolvimento de programas de TVDI, baseado em metodologias ágeis, que integra atividades inerentes ao processo de produção de TV e atividades do desenvolvimento de software. O modelo é pautado em quatro aspectos: 1) Filosofia de trabalho, 2) Processo, 3) Papéis e Responsabilidades e 4) Artefatos. A definição da filosofia de trabalho fundamenta o processo de modelagem e implementação. Este processo deve ter fases e ciclos bem definidos com detalhamento das atividades a realizar. A descrição do perfil dos recursos humanos envolvidos (programador, engenheiro de teste, consultor, roteirista, diretor e etc) bem como a responsabilidade de cada um deles dentro das atividades definidas, evita distorções ao longo da execução do modelo e permite que toda a equipe construa, de forma colaborativa, um programa de TVDI. O final de cada atividade resulta na produção de artefatos (documentos, códigos, mídias e submodelos produzidos durante o processo) dentre os quais pode-se destacar: *storyboards*, *timeline*, fluxo de interatividade e esboço de interface a ser contemplado. O modelo divide o ciclo de vida do desenvolvimento de programas de TVDI em 5 fases curtas, com iterações constantes e com forte integração do usuário (telespectador) à equipe de desenvolvimento. São elas: Concepção, Elaboração, Construção, Prototipação/Teste e Liberação conforme pode ser visto na figura 1.

O principal objetivo da etapa de concepção é a compreensão e o entendimento da oportunidade de criação de um programa para TVDI. Nesta etapa define-se o que o programa para TVDI vai apresentar ao usuário/telespectador. As atividades mais relevantes da etapa são: (i) busca de oportunidades; (ii) desenvolvimento do projeto inicial; (iii) teste de concepção. Entende-se como (i) o uso de métodos (reuniões, brainstorming e etc) para incentivar a busca pela oportunidade de um novo negócio ou para a inovação de um programa já existente. Na atividade (ii) elabora-se um *storyboard* que representa o projeto inicial com a idéia central

Figura 1: Fases do Processo de Desenvolvimento de Programas de TVDI [14]

do programa, as intenções de serviços e as funcionalidades a serem oferecidas ao usuário. A atividade (iii) é responsável por validar o projeto inicial. Participam desta etapa o produtor, o gerente de criação, o roteirista, o gerente de projeto e o usuário/telespectador.

A fase de elaboração tem como principal meta tentar prever os diversos fatores que podem afetar o desenvolvimento do programa para TVDI. Entre as principais atividades dessa fase pode-se destacar: (i) Planejamento; (ii) Roteirização; (iii) Projeto arquitetural; (iv) Programação visual. A atividade (i) consiste em estimar o custo do programa e o cronograma, criar um plano de liberação das iterações do programa, definir os papéis de cada equipe de trabalho assim como recrutar os profissionais envolvidos. Entende-se como (ii) a elaboração do roteiro com uma descrição textual do programa e indicações que mostrem as necessidades especiais da aplicação tais como: utilização do canal de retorno e banco de dados, utilização de serviços externos. Este artefato pode ser comparado com o documento de requisitos e como tal pode sofrer diversas revisões até a liberação do programa para TVDI. A atividade (iii) consiste em criar um projeto arquitetural para descrever exatamente o que deve ser construído para a aplicação como um todo (software, plano de interatividade e elementos de mídia). O projeto arquitetural completo deve conter informações sobre o sistema do *middleware* que será utilizado, o nível de interatividade do programa para TVDI, as funcionalidades requeridas e a infra-estrutura existente e o projeto de segurança. Por fim, a atividade (iv) contempla a criação dos elementos gráficos e visuais do programa. Todos os recursos humanos participam dessa fase.

A fase de construção do programa contempla a implementação dos componentes (software e mídia) definidos na fase anterior. Quatro atividades são destaque nesta fase: (i) Codificação de mídias, (ii) Codificação do programa de software, (iii) Produção do programa de TV, (iv) Integração do programa de software ao programa de TV. Entende-se como (i) a execução do projeto gráfico da aplicação. Esse projeto se concentra em questões como: ajuste das imagens com as diversas definições possíveis, utilização da área útil de uma tela de televisão, ajuste da qualidade do som e etc. A atividade (ii) se concentra nos métodos para codificação

dos módulos que compõem o software. Algumas práticas, herdadas das metodologias ágeis, são adotadas aqui para evitar que falhas na execução do software atrapalhem o programa de TV. O ciclo de vida do processo de produção de programa para TV convencional é representado pela atividade (iii). Por fim, a atividade (iv) representa a integração do software com as mídias codificadas. A equipe de projeto gráfico e de software bem como os editores e produtores são os principais participantes dessa fase.

Algumas observações devem ser ressaltadas sobre a fase de construção, especificamente sobre a atividade de codificação dos softwares que podem fazer parte de um programa de TVDI. A codificação da parte do programa de TVDI referente ao software pode ser estruturada através de composições sincronizadas de nós que representam as mídias codificadas (vídeo, áudio, texto, imagens, dados, componentes e serviços de software). Apesar disso a concepção "ideal" desse programa de TVDI deve ser diferente da forma como programas hipermídia / multimídia tradicionais são concebidos. O projeto de um programa para TVDI deve levar em consideração requisitos que são específicos do ambiente de TV como citado anteriormente.

A implementação do programa de TVDI pode ser feita através de dois paradigmas: declarativo e procedural. Na programação procedural, o programador codifica cada passo a ser executado pela máquina de execução. Essa codificação é feita através da decomposição algorítmica de um problema. A vantagem desse paradigma é o maior controle do código exercido pelo programador que permite estabelecer todo o fluxo de controle e execução de seu programa. Entre as linguagens procedurais para a implementação de um programa de TVDI mais comuns pode-se citar Lua [8], Java [3], ECMAScript [6]. Na programação declarativa, o programador fornece o conjunto das tarefas a serem realizadas usando um nível de abstração mais alto que não leva em consideração detalhes de como realmente essas tarefas serão executadas. Em outras palavras, a linguagem enfatiza a declaração descritiva de um problema ao invés de sua decomposição em implementações algorítmicas [12]. São exemplos de linguagens declarativas para TVDI o NCL [11] e o XHTML [9].

A fase de Prototipação/Testes é uma das mais importantes etapas do processo. Ela é executada em paralelo com as etapas anteriores e os principais participantes dessa etapa são o engenheiro de software e o produtor. Nessa fase são realizados: prototipações, testes unitários, testes de integração, testes de desempenho, testes de usabilidade, revisão tipográfica, e testes de infra-estrutura.

A liberação ocorre após o término de todos os testes, indicando que o programa está concluído. Quando o programa de TVDI está pronto para ser veiculado cabe ao produtor promover comercialmente o programa junto aos usuários do programa.

A abordagem descrita anteriormente [14] não é voltada à estruturação de componentes de softwares no ambiente de TV. Ela concentra-se em definir quais são as atividades gerais do processo de produção e não como essas atividades devem ser executadas. O modelo apresentado a seguir, StoryToCode, permite especificar programas de TVDI voltados à utilização de componentes de software. Ele se concentra nas fases de elaboração e construção do programa e tem como objetivo principal o reuso desses componentes em outros contextos.

O StoryToCode está concentrado nas atividades que surgem depois das fases de concepção e elaboração de um programa de TVDI. Essas atividades podem ser agrupadas em dois módulos: atividades de produção/geração de mídias (imagens, vídeos, textos e etc) e atividades de produção de software. O StoryToCode trata especificamente as atividades do segundo módulo e a responsabilidade da sua execução é da equipe de projeto de software em colaboração com toda a equipe de TV.

3. O MODELO STORYTOCODE

O objetivo do StoryToCode, modelo proposto nesse trabalho, é permitir a especificação e construção de componentes de software para uso em programas de TVDI e em outros contextos. No âmbito do modelo, entende-se como contexto a plataforma de execução do software (TV, Mobile, Web e etc). O StoryToCode parte de um *storyboard* e usa conceitos de modelagem de sistemas para criar um conjunto de elementos que representem tanto as mídias quanto os componentes de software que compõem um programa interativo e ainda destacar algumas visões sistêmicas (estrutura, eventos e etc) a fim de poder reusar os artefatos gerados em outros contextos.

Um dos diferenciais do StoryToCode está na possibilidade de estruturar um software a partir de informações extraídas de um *storyboard*. A escolha do *storyboard* como ponto de partida se deve ao fato desse artefato ser um dos mais utilizados no ambiente multidisciplinar de um programa de TV. A hipótese é que isso facilitará o entendimento e a execução de todo o processo de desenvolvimento de uma aplicação pela equipe multidisciplinar. Outro diferencial (herdado da MDA) é a capacidade de gerar visões diferentes de um mesmo software facilitando o seu reaproveitamento em outros contextos.

A arquitetura do StoryToCode está dividida em três partes relacionadas entre si, ilustradas na figura 2: *storyboard*, arquitetura de elementos e geração de código. A arquitetura do StoryToCode baseia-se no conceito de RTE (*RoundTrip Engineering*) [2] a fim de permitir o intercambio bidirecional entre definições mais abstratas (arquitetura e design) até, possivelmente, o código. Ela define como deve ser realizada a transformação de um *storyboard* em um conjunto de elementos abstratos e, posteriormente, a transformação desses elementos em código ou ainda, o caminho inverso, isto é a partir do código gerar o modelo de elementos e depois disso o *storyboard*.

Neste trabalho, considera-se uma transformação como a geração (automática ou manual) de um modelo destino a partir de um modelo origem.

3.1 Storyboard

Em um ambiente de geração de conteúdo para TVDI, um *storyboard* pode ser definido como uma técnica usada na descrição da seqüência fundamental de cenas que melhor representam um programa. Essa técnica fornece uma descrição de alto nível dos elementos que irão compor cada uma

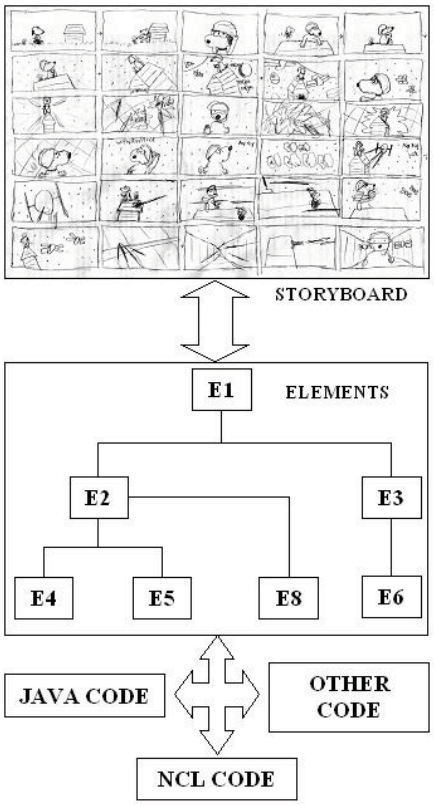

Figura 2: Arquitetura do StoryToCode

das cenas e dos fluxos de interatividade oriundos do uso de softwares. Apesar de ser uma técnica considerada informal e "fracamente" estruturada, é muito usada para modelar o fluxo de apresentação das cenas (transição), os efeitos interativos e temporais, narrativas (dublagem, narrações e etc), layout das cenas entre outros. As informações contidas em um *storyboard* vão desde a lista das cenas que compõem o programa, passando pela visualização do seu esboço até a descrição dos seus conteúdos (imagens, vídeos, textos, gráficos, animações, elos [$de \rightarrow para$]). A figura 3 exemplifica a estrutura de um possível *storyboard* e o utiliza como exemplo de uso do modelo.

No aspecto relativo à transformação do *storyboard* em elementos de um software, a primeira parte do StoryToCode define que esse artefato deve ser usado como repositório dos

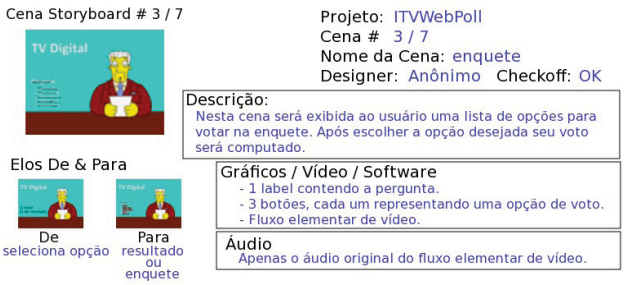

Figura 3: Exemplo de quadro do *storyboard*

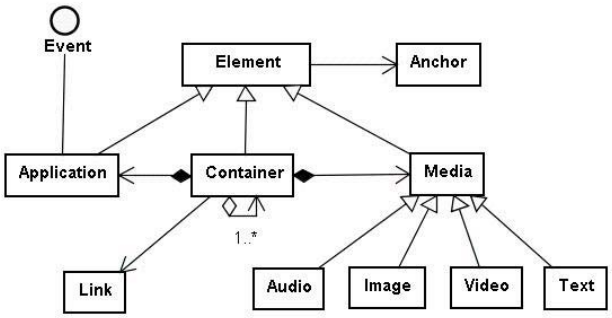

Figura 4: Hierarquia de elementos do StoryToCode

requisitos que deverão ser atendidos na construção dos elementos. Assim, a equipe de projeto de software deve usar o *storyboard* (construído na fase de concepção) para extrair os requisitos e gerar uma representação abstrata do conjunto de elementos contidos nas cenas de um programa com as suas relações e os seus eventos de interatividade. Esse conjunto de elementos está representado pela segunda parte do StoryToCode.

3.2 Conjunto de elementos

É importante ressaltar que o 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 diretamente em outros modelos ou em códigos de linguagens de programação. Para que isso seja possível, a construção do conjunto de elementos deve usar uma notação não ambígua e padronizada a fim de permitir a sua transformação em códigos para plataformas diferentes. O StoryToCode define uma hierarquia de elementos abstratos representados através da notação UML que podem ser estendidos para se adequar aos requisitos específicos de um *storyboard*. Essa hierarquia de elementos é semelhante a hierarquia de classes do *Nested Context Presentation Model* (NCPM) [13] e pode ser observada na figura 4.

Na raiz dessa hierarquia está o elemento *Element* cuja principal função é servir como modelo base para criação de outros elementos agrupando suas características e comportamentos comuns. Um *Element* é uma entidade que possui como atributos um identificador único, um descritor, que contempla as informações para determinar como a entidade deve ser apresentada, e uma lista de âncoras. Uma âncora é definida como uma região marcada (no tempo ou no espaço) do conteúdo de um elemento, usada para "amarrar" os elos (*Link*) e é representada no modelo pelo elemento *Anchor*. Um *Element* pode ainda ser especializado em três entidades: *Media*, *Application* e *Container*. Elementos do tipo *Media* são aqueles cuja principal responsabilidade é abstrair uma mídia presente em uma apresentação. Um elemento *Media* tem os atributos "conteúdo" e "lista de âncoras" (herdado de *Element*) cuja definição dos valores dependem da dinâmica de um programa específico a ser apresentado. O StoryToCode permite que o elemento *Media* seja especializado em outros elementos a fim de abstrair as diversas mídias contempladas em um *storyboard* como vídeo, imagem, texto e áudio. O ele-

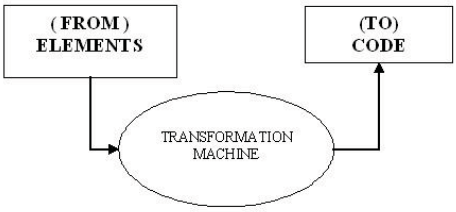

Figura 5: Definição de Regras de Transformação

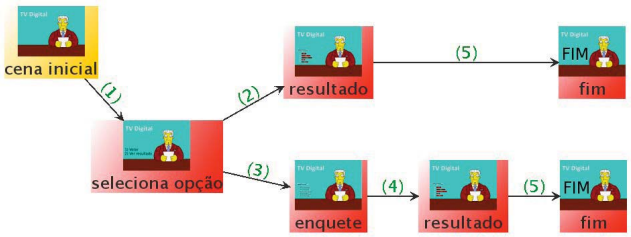

Figura 6: Fluxo de cenas do ITVWebPoll

mento *Application* presente na hierarquia tem como função definir as estruturas de dados de um componente de software cujas operações (instruções) ao serem executadas, representam a responsabilidade de tratar os eventos de interatividade especificados em um *storyboard*. Para isso, esse elemento precisa ser especializado e cada elemento concreto criado a partir de um *Application* deve cumprir uma interface do tipo *Event*. Essa interface também deve ser especializada para definir quais são e como devem ser tratados os eventos de interatividade. Assim esses componentes de software podem abstrair elementos de um *storyboard* e as ações decorrentes do uso de cada um deles como, por exemplo, selecionar botões, exibir barras de progresso, exibir alertas e etc. Por fim, existe ainda o *Container*, um elemento que representa uma composição entre os elementos do tipo *Media* e *Application* cuja multiplicidade é $1 \rightarrow N$ (um para muitos). Ele pode ser usado para agrupar elementos de um *storyboard* que vão desde menus, botões, imagens até mesmo as cenas de todo o programa. Para isso, um *Container* tem como atributos uma lista de elos entre elementos (*Link*) e uma outra lista de elementos *Container*. Cada *Link* define uma ligação entre um elemento de origem e de destino.

O StoryToCode define que cada um dos elementos do modelo deve ser especializado para conter sua própria lista de características a fim de abstrair corretamente o *storyboard*. Essa capacidade de extensão do modelo permite adequá-lo a qualquer especificação tanto no que diz respeito a estruturação das mídias quanto dos componentes de software de um *storyboard*. Essa possibilidade de descrever cada elemento que compõe um *storyboard* com riqueza de detalhes é que permite o uso das transformações para alcançar um nível mais baixo de abstração que no modelo significa obter o código de uma aplicação para um contexto específico.

3.3 Código

Uma vez que o conjunto de elementos esteja definido, chega-se a terceira etapa do StoryToCode: geração de código. O objetivo dessa etapa é gerar o código de uma aplicação para um contexto específico a partir do conjunto de elementos genéricos gerado na etapa anterior. Para isso, o StoryToCode define um componente especial chamado de *transformation machine*. Ele recebe como entrada o conjunto de elementos e gera como saída o código, conforme figura 5.

Para que seja possível transformar conjuntos de elementos independentes de contexto em seus respectivos códigos na plataforma de destino, é preciso definir as regras de transformação da *transformation machine*. Essas regras são baseadas no conhecimento das estruturas dos elementos de origem (conjunto de elementos) e destino (código para uma plataforma específica). A *transformation machine* permite adicionar as regras (e outras informações importantes) para possibilitar o mapeamento de um elemento do modelo de origem no código correspondente a esse elemento na linguagem de programação específica de destino. Assim a transformação de um mesmo conjunto de elementos pode ser feita para plataformas diferentes, desde que o *transformation machine* específico para cada uma dessas plataformas esteja disponível. O exemplo de uso abordado na próxima seção mostra um resumo das regras de transformação que foram implementadas em duas instâncias desse componente, uma no contexto da TV (JavaTV) e outra na Web (HTML).

4. EXEMPLO DE USO

Para mostrar a viabilidade da aplicação do StoryToCode, criou-se um programa de TV (um telejornal fictício) que deve conter uma enquete para permitir a um telespectador votar em diferentes opções utilizando o controle remoto ou ainda votar pela Web. Essa enquete é um componente de software, denominado ITVWebPoll (*Interactive TV and Web Poll*), que pode ser utilizado em diferentes situações e nos contextos de TV (JavaTV ou NCL) e Web (HTML).

O *storyboard* diz que neste programa o usuário tem a opção de participar de uma enquete interativa, podendo votar ou não na mesma. Na cena "seleciona opção", ao pressionar o botão verde o usuário deixa de votar e é levado ao resultado. Ao pressionar o botão amarelo, no entanto, ele é levado a uma outra cena ("enquete"), possibilitando votar através de um menu. Após votar ele é direcionado à tela de exibição do resultado. Por fim, o usuário deve pressionar o botão vermelho para terminar a aplicação. A versão simplificada desse *storyboard* da cena da "enquete" pode ser vista na figura 3. O fluxo de cenas do programa pode ser visto na figura 6. Nesse fluxo, (1) indica o início automático, (2) que o usuário pressionou o botão "Ver Resultado", (3) que o usuário pressionou o botão "Votar na Enquete", (4) que o usuário selecionou uma opção da enquete, (5) que o usuário pressionou o botão "Fechar".

A partir do *storyboard* foram obtidos os seguintes requisitos funcionais: (i) Permitir aos telespectadores escolher as opções que lhes convêm, sem nenhum tipo de restrição ou limitação (todas as alternativas devem ser tratadas da mesma maneira); (ii) Confirmar se a opção desejada é a mesma que foi selecionada; (iii) Executar e computar uma opção escolhida; (iv) Esclarecer ao telespectador que o seu voto foi computado; (v) Permitir acompanhar o resultado;

A partir do *storyboard* e da lista de requisitos funcionais

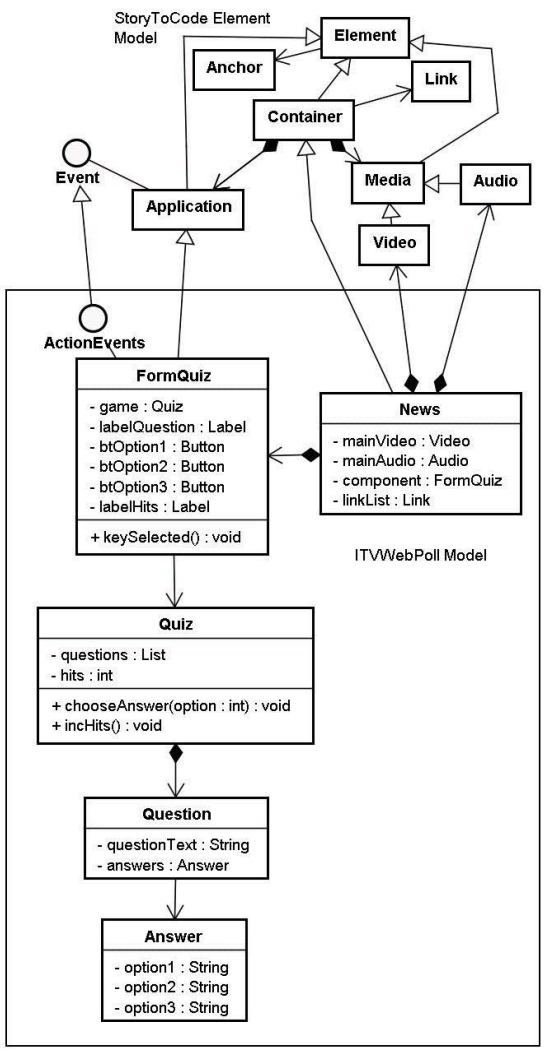

Figura 7: Conjunto de elementos

passou-se a segunda parte do StoryToCode com a criação do conjunto de elementos para o ITVWebPoll. O conjunto de elementos é formado por: *News, FormQuiz, Quiz, Question* e *Answer* além da interface *ActionEvents* conforme pode ser visto na figura 7.

Os elementos *Quiz, Question* e *Answer* representam abstrações da lógica de negócio de um quiz. Dessa forma, um elemento *Question* é uma entidade que tem como principais atributos o texto relativo a uma pergunta e uma lista de elementos *Answer*. Cada elemento dessa lista *Answer* contempla três atributos: dois deles são opções de respostas falsas e a outra uma opção verdadeira. O elemento *Quiz* é uma composição da lista de perguntas (elementos *Question*) e suas respectivas respostas (elementos *Answer*), cuja principal responsabilidade é controlar o número de acertos (atributo *hits* de *Question*) conseguidos por um usuário. O elemento *FormQuiz* é uma especialização de um *Application* que abstrai a interface gráfica usada no quiz. Essa interface inclui dois rótulos (para exibir o texto da pergunta e o número de acertos) e três botões (para exibir as opções de

Elemento	*Atributo*	*HTML*	*JavaTV*
FormQuiz	Button	INPUT	HTextButton
FormQuiz	Label	LABEL	HText
News	FormQuiz	DIV	HContainer
News	Video	EMBED	HVideoComponent

Figura 8: Do Modelo para HTML e JavaTV

respostas). A principal responsabilidade desse elemento é tratar o evento de seleção de uma das opções definido na interface *ActionEvents* através da operação *keySelected*. Nessa operação, cada vez que um dos botões de opção (verdadeira) é selecionado, um acerto é computado e exibido. Por fim, o conjunto de elementos ainda contém o elemento *News*. Esse elemento é um *Container* que representa uma composição de elementos *Audio* e *Video*, de um *FormQuiz* e da lista de elos (*Link*) necessários às sincronizações entre os elementos do programa.

Uma vez que o conjunto de elementos foi definido, passou-se a terceira fase do StoryToCode. Para que fosse possível gerar o código do ITVWebPoll tanto no ambiente de TVDI quanto no Web foram implementados duas instâncias do *transformation machine*: (i) Conjunto de elementos $\rightarrow JavaTV$ e (ii) Conjunto de elementos $\rightarrow HTML$. Essas instâncias recebem como entrada um arquivo no formato XMI referente ao modelo de elementos do ITVWebPoll. Esses componentes implementados recebem como entrada arquivos texto no formato XMI. Dessa forma, usou-se uma ferramenta CASE para converter o arquivo que continha o conjunto de elementos escrito em UML para o formato XMI. A partir dos elementos de entrada (arquivo XMI), os componentes *transformation machine* aplicaram as regras para o mapeamento dos elementos e seus atributos no modelo de origem no código respectivo da linguagem de destino. A figura 8 ilustra alguns exemplos de mapeamentos implementados nessas regras.

A geração de código para o ITVWebPoll não significou o término do trabalho de codificação. O código gerado nesta etapa do StorytoCode representou uma parte do código final do ITVWebPoll. Assim, depois que esse código foi gerado, a equipe de programação ainda precisou completá-lo para finalizar o trabalho de implementação. As duas transformações previstas foram executadas. A figura 9 mostra de forma resumida o resultado da transformação de um arquivo XMI que representa o elemento *News* em seu respectivo código na linguagem de destino JavaTV.

5. CONCLUSÕES

A principal contribuição deste trabalho compreende a integração das diferentes habilidades e competências existentes no ambiente de TV no âmbito da construção de programas interativos. O trabalho exibe um modelo de construção desses programas baseado em componentes, de maneira a permitir o reuso desses componentes em outros contextos. O reaproveitamento dos componentes é feito a partir do uso de tranformações de modelos mais abstratos do programa em modelos concretos. A transformação de modelos é uma contribuição importante nessa área, visto que possibilita o reuso de informações entre plataformas diferentes, proporcionando maior rapidez e qualidade no desenvolvimento de

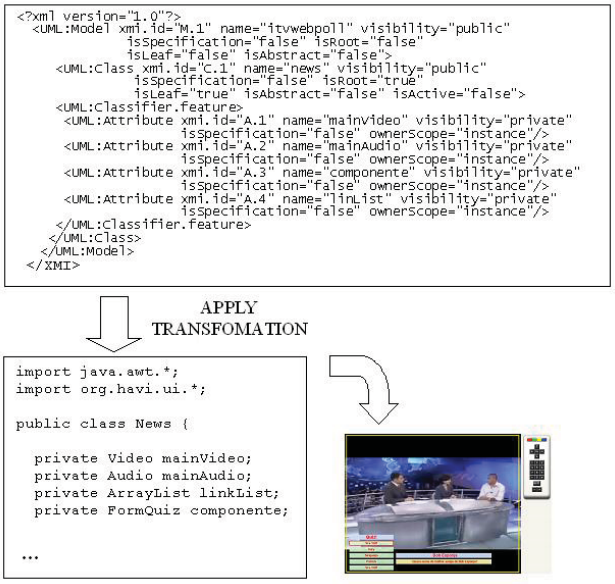

Figura 9: Geração de código

software além de não exigir dos participantes do processo produtivo oriundos do universo da TV um conhecimento específico na área de modelagem de sistemas.

Os resultados da implementação do exemplo de uso estão direcionando os próximos passos do trabalho. Dentre esses passos está a criação de mecanismos para melhorar o desempenho da terceira etapa do modelo. Esta etapa não executa a geração completa do código e ainda não permite que o caminho inverso seja percorrido, ou seja, a partir de um código chegar a uma definição de mais alto nível. Outros passos seriam o desenvolvimento de um transformador genérico o suficiente para permitir que a equipe de desenvolvimento possa fazer alterações nas regras de mapeamento através do uso de um editor gráfico orientado pela sintaxe e semântica e a criação de uma ferramenta que permita construir um *storyboard* de forma a extrair automaticamante o modelo abstrato de elementos.

6. AGRADECIMENTOS

Este trabalho contou apoio da CAPES, através de um bolsa de doutorado do 1o autor e da FAPESB, através de recursos do Projeto "Implantação de uma Infra-Estrutura para Desenvolvimento e Testes de programas para TVDI" (PES0001 / 2007). Os autores agradecem ainda a participação do aluno José Ricardo F. Júnior.

7. REFERÊNCIAS

[1] A. W. Brown. *Large-Scale, Component Based Development*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.

[2] D. Frankel. *Model Driven Architecture: Applying MDA to Enterprise Computing*. Wiley Press, 2003.

[3] J. Jones. Dvb-mhp/java tv^{TM}data transport mechanisms. In *CRPIT '02: Proceedings of the Fortieth International Conference on Tools Pacific*, pages 115–121, Darlinghurst, Australia, Australia, 2002. Australian Computer Society, Inc.

[4] C.-Y. Jung, J.-S. Kim, C.-S. Yoo, and Y.-S. Kim. Model of generating smil document using temporal scripts of animation component. In *Computational Science and Its Applications - ICCSA 2006*, pages 990–1000, Berlin, Heidelberg, 2006. Springer.

[5] L. E. C. Leite, G. L. de Souza Filho, S. R. de Lemos Meira, P. C. T. de Arújo, J. F. de A. Lima, and S. M. Filho. A component model proposal for embedded systems and its use to add reconfiguration capabilities to the flextv middleware. In *WebMedia '06*, pages 203–212, New York, NY, USA, 2006. ACM.

[6] C. Lopez-Nataren and E. Viso-Gurovich. An ecmascript compiler for the .net framework. In *ENC '05: Proceedings of the Sixth Mexican International Conference on Computer Science*, pages 235–239, Washington, DC, USA, 2005. IEEE Computer Society.

[7] M. C. Marques Neto and C. A. Santos. An event-based model for interactive live tv shows. In *MM '08: Proceeding of the 16th ACM international conference on Multimedia*, pages 845–848, New York, NY, USA, 2008. ACM.

[8] F. Mascarenhas and R. Ierusalimschy. Efficient compilation of lua for the clr. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 217–221, New York, NY, USA, 2008. ACM.

[9] S. Pemberton. Xhtml 1.0: The extensible hypertext markup language (second edition). World Wide Web Consortium, Recommendation REC-xhtml1-20020801, August 2002.

[10] R. F. Rodrigues and L. F. Soares. Produção de conteúdo declarativo para tv digital. In *SemiSH*, 2006.

[11] H. V. O. Silva, R. F. Rodrigues, L. F. G. Soares, and D. C. Muchaluat Saade. Ncl 2.0: integrating new concepts to xml modular languages. In *DocEng '04*, pages 188–197, New York, NY, USA, 2004. ACM.

[12] L. F. G. Soares, R. F. Rodrigues, and M. F. Moreno. Ginga-ncl: the declarative environment of the brazilian digital tv system. In *Journal of the Brazilian Computer Society, v. 12*, pages 37–46. SBC, 2007.

[13] L. F. G. Soares, R. F. Rodrigues, and D. C. M. Saade. Modeling, authoring and formatting hypermedia documents in the hyperprop system. *Multimedia Syst.*, 8(2):118–134, 2000.

[14] E. G. Veiga and T. A. Tavares. Um modelo de processo para o desenvolvimento de programas para tv digital e interativa. In *WebMedia '06: Workshop de Teses e Dissertações*, pages 53–57, New York, NY, USA, 2006. ACM.