MANOEL CARVALHO MARQUES NETO

CONTRIBUIÇÕES PARA A MODELAGEM DE APLICAÇÕES MULTIMÍDIA EM TV DIGITAL INTERATIVA

Tese apresentada ao Programa Multiinstitucional de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia, Universidade Estadual de Feira de Santana e Universidade Salvador, como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Orientador: Dr. Celso Alberto Saibel Santos

Salvador
2011

Ficha catalográfica elaborada pela Biblioteca XXXXXX,
Instituto de Matemática

Neto, Manoel Carvalho Marques
S111d Contribuições para a Modelagem de Aplicações Multimídia para TV Digital Interativa / Manoel Carvalho Marques Neto. – Salvador, 2011.

148p. : il.

Orientador: Prof. Dr. Celso Alberto Saibel Santos.
Tese (doutorado) – Universidade Federal da Bahia, Instituto de Matemática, Programa Multiinstitucional de Pós-Graduação em Ciência da Computação, 2011.

1. Multimídia. 2. TV Digital. 3. Estruturação Conteúdo. I. Santos, Celso Alberto Saibel. II. Universidade Federal da Bahia. Instituto de Matemática. III Título.

CDD 20.ed. 003.83

MANOEL CARVALHO MARQUES NETO

CONTRIBUIÇÕES PARA A MODELAGEM DE APLICAÇÕES MULTIMÍDIA EM TV DIGITAL INTERATIVA

Esta tese foi julgada adequada à obtenção do título de Doutor em Ciência da Computação e aprovada em sua forma final pelo Programa Multiinstitucional de Pós-Graduação em Ciência da Computação da UFBA-UEFS-UNIFACS.

Salvador, 15 de dezembro de 2011

Professor e orientador Celso Alberto Saibel Santos, Doutor
Universidade Federal da Bahia

Professora Maria da Graça Campos Pimentel, Doutor
Universidade de São Paulo

Professor Guido Lemos de Souza Filho, Doutor
Universidade Federal da Paraíba

Professor Manoel Gomes de Mendonça Neto, Doutor
Universidade Federal da Bahia

Professora Christina von Flach Garcia, Doutor
Universidade Federal da Bahia

Dedico este trabalho aos meus pais e a minha querida "Vozinha Dona Cotinha".

AGRADECIMENTOS

A lista de agradecimentos é meio extensa por isso vou tentar resumir. Vou começar pela família. Agradeço ao meu querido pai GESSÉ e a minha mãe ROSA pelo esforço e dedicação em todos esses anos de vida....É uma pena Pai que voçê não esteja aqui para presenciar esse momento....Sei que ficaria muito feliz....

Agradeço, imensamente, aos meus irmãos (LARA, GESSÉ, CRISTIANE, ROSINHA, MARLENE e DAI) pelo apoio depois que nosso pai partiu. Compartilho com todos vocês os méritos alcançados

Agradecimentos especiais ao meu orientador pela compreensão, paciência, persistência e amizade, aos colegas de turma, e do laboratório (não vou arriscar esquecer de alguém!!!!) especialmente ao meu amigo ROMILDO, que sempre dividiu comigo todas as alegrias e tristezas desses últimos anos. Um agradecimento especial a todos os meus alunos e ex-alunos.

When you think you know something, you have to look otherwise. Even though it seems silly or wrong, you should try.

—PETER WEIR (1989)

RESUMO

Os programas para TV Digital Interativa (TVDI) são aplicações multimídia cujo desenvolvimento envolve um conjunto de recursos, tecnologias e pessoas diferente daquele encontrado em um ambiente de TV convencional. A definição de processos e métodos voltados modelagem de programas para TVDI ainda pode ser considerado um problema em aberto. A maior parte dos trabalhos relacionados a TVDI não apresenta soluções para questões como: reuso, estruturação de requisitos etc. Esta tese apresenta dois modelos, chamados de IDTVS e StoryToCode, que permitem a especificação de programas para TVDI a partir da estruturação do conteúdo dessas aplicações e a partir de eventos.

Palavras-chave: Multimídia, TV Digital, Storyborads, TVDI, Conteúdo Extra.

ABSTRACT

Interactive Digital TV programs (IDTV) are multimedia applications whose development involves a set of resources, technologies and different people from that found in an environment of conventional TV. The definition of processes and methods aimed to modeling IDTV programs can still be considered an open problem. The majority of studies related to IDTV offers no solutions to questions such as: reuse, requirements structuring, and so on. This thesis presents tow models named IDTVS and StoryToCode, which allows the specification of IDTV programs based on structuring the content of applications and events.

Keywords: Multimedia, Digital TV, Storyborads, IDTV, Extra Content.

SUMÁRIO

Apêndices

LISTA DE FIGURAS

LISTA DE TABELAS

Capítulo 1

Este capítulo aborda a motivação principal para esta Tese, bem como sua justificativa e seus objetivos.

INTRODUÇÃO

A TV, assim como outros veículos de comunicação, está passando por um processo de transformação importante. Um dos principais motivos para isso é o processo global de digitalização, que impõe a substituição de plataformas analógicas por plataformas digitais interoperáveis. Os impactos da digitalização no ambiente de TV não estão concentrados apenas nas tecnologias envolvidas na codificação, transmissão e captura do sinal, mas também na estruturação de uma nova cadeia produtiva de conteúdos. Esta cadeia envolve a produção tanto dos conteúdos tradicionais de TV (vídeo e áudio) quanto a de conteúdos específicos para TV Digital como, por exemplo, os componentes interativos (softwares).

Os processos produtivos de programas que antes eram específicos do ambiente de TV, a partir do processo de digitalização devem começar a agregar uma variedade de recursos oriundos de outras áreas como computação, design etc. Esse é o caso dos programas para TV Digital Interativa (TVDI). A definição de processos e métodos voltados à modelagem de programas para TVDI ainda pode ser considerado um problema em aberto. Quando comparado a outras áreas, o desenvolvimento de programas para TVDI ainda está longe do ideal.

Ao longo dos anos, muitos trabalhos tiveram foco na criação/evolução de tecnologias voltadas para o universo de TV Digital. Dentre eles podem-se destacar: (i) os trabalhos relacionados com a produção de conteúdo declarativo para TVDI, como em [9], [10], [11] e (ii), naqueles associados com as ferramentas de apoio a elaboração de aplicações, como em [12], [13], [14], [15], [16] e [17]. Outros trabalhos esboçam o uso de arquiteturas que permitem a integração sistêmica de outras plataformas com a TV Digital, como em [18], [19] e [20].

Por outro lado, poucos trabalhos trataram o processo de criação desse tipo específico de aplicação multimídia. Dentre esses trabalhos destacam-se [3], [21] e [22]. Um dos problemas com o desenvolvimento de programas para TVDI, e de outras aplicações multimídia, não é apenas a ausência de modelos de processos de desenvolvimento sofisticados, mas também a falta de notações para permitir uma especificação integrada de um sistema em diferentes níveis de abstração e a partir de diferentes perspectivas [23].

O desafio para o desenvolvimento de programas para TVDI é aliar ao processo de modelagem de elementos interativos para o ambiente de TV, as técnicas tradicionalmente usadas na especificação de sistemas para solução de questões como: reuso, estruturação de requisitos, comunicação em equipes multidisciplinares, definição de ferramentas etc.

Para esta tese, a concepção de elementos para aplicações interativas na TV Digital deve responder a três questões importantes:

i) Como podem ser estruturados os conteúdos nessas aplicações?

ii) Como pode ser definida uma dinâmica de comunicação (transmissão, recepção, armazenamento temporário etc.) para os programas de TVDI e os possíveis geradores de conteúdo?

iii) Como podem ser modelados e construídos os programas voltados ao uso de componentes de software que permitam o seu reaproveitamento em contextos diferentes da TV?

1.1 OBJETIVO

Buscando responder essas questões, o objetivo dessa tese é apresentar um novo processo de concepção de componentes interativos para aplicações multimídia com foco em TVDI.

Os objetivos específicos da tese são:

i) Permitir estruturar conteúdo multimídia em aplicações de TVDI;

ii) Permitir o projeto e implementação de uma aplicação multimídia, para os contextos da TV digital e Web;

iii) Diminuir a responsabilidade do difusor de conteúdos através da descentralização das etapas de produção que estão fora do seu universo de trabalho original: a especificação e implementação de um componente interativo (artefato de software);

iv) Permitir a participação de outros atores (analistas de sistemas, programadores, designers etc.) no processo produtivo de um programa de TVDI;

1.2 CONTRIBUIÇÕES

1.2.1 Interactive Digital TV Show- IDTVS

Uma das motivações dessa tese foi mostrar a necessidade e provar a viabilidade da estruturação de dados extras, aliada à definição de classes de eventos que caracterizem a dinâmica do programa de TV apresentado e que podem ser utilizados para a construção da aplicações executadas no receptor. A chegada de dados no receptor que caracterizem a ocorrência destes eventos possibilita que uma aplicação faça tratamento de eventos, por exemplo, de acordo

com informações de perfil do usuário. Os modelos que permitem estruturar dados extras são as principais contribuições dessa tese. Eles estimulam o desenvolvimento de aplicações interativas para programas de TV, contribuído para disseminação da tecnologia e proporcionando tratamento personalizado a um conteúdo tipicamente voltado ao consumo de massa.

Durante as discussões iniciais que definiram os rumos dessa tese, algumas questões, levantadas durante a fase de fundamentação teórica, foram preponderantes para o desenvolvimento deste trabalho. Entre essas questões, pode-se destacar:

a) Existem mecanismos nativos, que possibilitem o desenvolvimento de aplicações imperativas que tenham relação semântica com programa de TV apresentado, de forma que seja possível alterar o comportamento da aplicação em função da ocorrência de determinados eventos ocorridos no programa de TV? Ou seja, é possível implementar aplicações interativas (e imperativas) com comportamento dinâmico?

Embora a parte declarativa do Ginga preveja e suporte, a edição de programas feitas em NCL [9], através de comandos de edição, não foi identificado nenhum mecanismo semelhante com relação ao sub-sistema imperativo. Esta constatação impeliu a busca por soluções voltadas ao sub-sistema imperativo.

b) Como é possível oferecer suporte ao desenvolvimento de aplicações interativas dinâmicas?

O primeiro modelo concebido nessa tese, denominado *Interactive Digital TV Show* - IDTVS, propõe o tratamento dessa questão por meio de uma abordagem orientada a objetos, que consiste na estruturação dos dados extras que compõem a aplicação, acompanhados da definição de classes de eventos que representam a dinâmica do programa de TV.

Para validar a proposta, foi identificada a necessidade de implementar um ambiente de suporte a aplicações procedurais que representasse o ambiente de geração, distribuição e execução de aplicações interativas de TV. A necessidade de implementar tal ambiente de simulação é justificada pela ausência, até a data em questão, de uma implementação confiável do sub-sistema imperativo do Ginga.

A escolha natural, por questões de compatibilidade com a proposta de implementação do sub-sistema imperativo, foi utilizar a linguagem Java no desenvolvimento do ambiente e das aplicações interativas. Para isso utilizou-se o Protocolo em Tempo Real (RTP) [24] e o Java Media Framework (JMF) [25], para a transmissão e apresentação do fluxo de vídeo. O fato do JMF ser pouco flexível com relação a diversidade de formatos de arquivos suportados limita a capacidade do ambiente em reproduzir alguns formatos de arquivos. A implementação, além de suportar a reprodução, em tempo real, de um fluxo de vídeo capturado de um dispositivo de captação de imagens (*webcam*) é possível reproduzir arquivos MOV e JPEG.

Em sua versão final, a implementação do carrossel de dados responsável pela entrega de dados extras aos receptores foi realizado com entrega confiável de dados (TCP) em unicast. Anteriormente, na tentativa de reproduzir, com fidelidade, o ambiente de entrega de dados, foi utilizada comunicação em multicast com transporte não confiável (UDP). Porém, questões

envolvendo o sincronismo entre o envio e a recepção de dados, aliados a ocorrência de entrega de arquivos corrompidos adicionou ao ambiente a necessidade de tratar uma série de problemas não triviais que fugiam do foco do trabalho.

Para representar a ocorrência de eventos durante a exibição do fluxo principal de vídeo, optou-se pelo envio de arquivos XML que descrevem os eventos. A chegada destes arquivos no receptor caracteriza a ocorrência de um evento relacionado ao programa de TV. Na implementação realizada, estes arquivos são gerados no transmissor e adicionados a uma estrutura de diretório pré estabelecida, que permite ao receptor reconhecer, através da organização da estrutura, a finalidade dos arquivos que a compõem.

Uma vez implementados os mecanismo de entrega e apresentação do fluxo principal de vídeo e da entrega de dados extras deu-se início a implementação da aplicação interativa. Esta nova etapa apresentou mais uma questão endereçada ao trabalho:

c) Como modelar uma aplicação interativa transmitida ao vivo?

Como exemplo de uso para demonstrar como tratar essa questão escolheu-se uma transmissão ao vivo de um jogo de futebol. A implementação de aplicações interativas dinâmicas, que tenha relação semântica com fluxo principal de áudio e vídeo é completamente dependente da natureza do vídeo apresentado. Um dos problemas dessa dependência é a sintonização tardia. Esse problema ocorre se um o usuário sintoniza o canal muito depois do início da transmissão e quando eventos importantes já ocorreram. Neste caso, os objetos decorrentes desses eventos seriam transmitidos depois da sintonização e não estariam disponíveis no receptor. O mecanismo de atualização definido no modelo IDTVS apresenta uma solução para esse problema.

Para modelar esse tipo de aplicação foi necessário mapear informações e eventos, assim como seus relacionamentos, relevantes para um jogo de futebol. Para isso, foi necessário definir um modelo que definisse a dinâmica não apenas da estruturação do conteúdo, mas também da sua distribuição. Por meio deste modelo e do conhecimento a respeito da organização dada aos arquivos que compunham a aplicação, foi possível implementar mecanismos capazes de criar menus e itens de menus, relacionando-os a eventos e arquivos de mídia extra vinculados ao programa apresentado. Este conhecimento ainda permitiu que a aplicação em execução no receptor, pudesse dispor de meios que identificassem informações referentes ao perfil do usuário. De posse destas informações, foi então possível realizar o tratamento automático, de determinados eventos, em função do perfil definido.

A implementação desta solução envolveu o desenvolvimento de uma série de funcionalidades concorrentes de finalidade específica. Essas funcionalidades são as responsáveis pelo tratamento dos eventos esperados no receptor. Entre tais funcionalidades cabe citar a atualização automática de menus e itens de menus, a exibição, em paralelo a apresentação do fluxo de vídeo principal, de vídeos e imagens extras, e a apresentação automática de mídias em função da ocorrência de eventos de acordo com o perfil do usuário.

Para identificar qual, dentre as funcionalidade descritas, deve ser invocada em função da ocorrência de um evento, assim como que atributos do evento são necessários para gerar a

instância que representa sua ocorrência no receptor, foi implementado um parser XML, responsável por extrair dos arquivos que descrevem a ocorrência de eventos e estas informações, para depois passá-las aos objetos apropriados.

O IDTVS é claramente focado nas dimensões de arquitetura e visão da especificação de sistemas [23]. Dessa forma, apesar da dinâmica do modelo permitir tratar alguns problemas do ambiente de TV ele é limitado na dimensão de processo. O IDTVS não define como devem ser executadas as atividades específicas de um processo de concepção, quais seriam os atores participantes deste processo.

Buscando também tratar a dimenção de processo, O IDTVS evoluiu para um modelo voltado à estruturação visual dos elementos de software que podem fazer parte de um programa para TVDI e que especifica os participantes e as atividades envolvidas na sua elaboração. Esta evolução do modelo foi denominada de StoryToCode.

1.2.2 StoryToCode

Um aspecto relevante no universo de TVDI é forma como esses novos programas devem ser produzidos. Na TV convencional pode-se perceber que 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. Para [26], especialmente no Brasil, a produção de conteúdo televisivo revela uma grade de programação maciçamente horizontal. Essa grade é composta quase na sua totalidade por produções próprias e, em menor escala, por outras produções, com presença mínima de peças audiovisuais independentes e de caráter regional. Ao contrário do que ocorre no Brasil, em alguns países da Europa e nos EUA as emissoras são obrigadas a ter em sua programação produções independentes, impedindo que apenas a visão da emissora seja veiculada diariamente. Essa obrigatoriedade também incentiva o mercado audiovisual, essencial para o desenvolvimento econômico do país. 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 etc.

Outros problemas do uso do modelo convencional são a sua baixa flexibilidade e extensibilidade que gerariam, no contexto da TVDI, “retrabalho” na concepção de softwares interativos diferentes para plataformas diferentes. Essa situação é semelhante à enfrentada 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 um celular, significaria refazer boa parte do sistema. Isso, de forma semelhante aos programas de TVDI, também gerava forte centralização, baixa flexibilidade / extensibilidade e aumento do "retrabalho".

A solução adotada na Web e em outros casos semelhantes foi aplicar técnicas de especificação de sistemas focadas na solução deste tipo de problema, como o desenvolvimento baseado em componentes (DBC) [27] [28]. O DBC se preocupa com a criação de componentes que possam ser reutilizados em outras aplicações. O conceito de reutilização está relacionado ao uso de produtos, 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. 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 [27].

A reutilização de softwares originados no ambiente de TV em outros contextos é algo que deve ser considerado, a partir do momento em que uma teledifusora tem possibilidade de usar outros canais de comunicação dentro dos seus programas de TV (ex.: Sites Web, Mensagens SMS etc.). 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 www.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 o software tem as mesmas funcionalidades nas três situações. A única diferença está no tratamento dado ao requisito não-funcional contexto.

O modelo StoryToCode foi concebido como uma evolução do modelo IDTVS. Ele está concentrado nas atividades que surgem depois das fases de concepção e elaboração de um programa de TVDI. Essas atividades podem ser agrupadas em duas fases: atividades de produção/geração de mídias (imagens, vídeos, textos, roteiros, cenários, storyboard etc.) e atividades de produção de software (definição de classes, codificação etc.). O StoryToCode trata especificamente as atividades da segunda fase e a responsabilidade da sua execução é da equipe de projeto de software (engenheiros, analistas, programadores, administradores de banco de dados e de redes entre outros) em colaboração com toda a equipe de TV (Diretores, produtores, cenógrafos etc).

O objetivo do StoryToCode é permitir a especificação e construção e reaproveitamento 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 etc.). O StoryToCode parte de um storyboard/cenário 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 etc.), a fim de poder reusar os artefatos gerados em outros contextos.

O StoryToCode está dividido em três etapas relacionadas entre si, *storyboard/cenário* de

onde os profissionais de software devem obter os requisitos de uma aplicação interativa, arquitetura de elementos que serve como modelo base de estrtuturação dos requisitos em classes a partir de ferramentas CASE (nesta tese usou-se o EMF [29]) e geração de código feita de forma semi-automática a partir do uso de motores de trasnformação inspirados na MDA [30]. A fim de permitir o intercâmbio entre definições mais abstratas (arquitetura e design) até, possivelmente, o código o StoryToCode baseia-se no conceito de RTE (*RoundTrip Engineering*) [30]. O modelo define como deve ser realizada a transformação de um *storyboard/cenário* em um conjunto de elementos abstratos e, posteriormente, a transformação desses elementos em código.

A arquitetura de elementos do StoryToCode representa uma evolução do modelo *Nested Context Presentation Model* (NCPM) [6] que é o modelo no qual está baseada da linguagem NCL [9]. O modelo NCPM não suporta, explicitamente, a extensão de estruturas de dados que abstraiam elementos de negócio ou mesmo elementos visuais. Uma hipótese é que isso poderia ser feito através do elemento `Script`, definido na hierarquia do modelo NCPM. Um `Script` é um elemento cuja função é armazenar o código de um programa (escrito em uma linguagem de programação) e cujas instruções (operações), quando executadas, geram mensagens que invocam o acionamento de métodos válidos de objetos de apresentação. Assim, enquanto o `Script` NCPM é um elemento não extensível com foco nos aspectos visuais de objetos de apresentação, um elemento `Application` do StoryToCode pode ser estendido para abstrair de forma não ambígua tanto as interfaces visuais quanto a lógica de negócio de uma aplicação. Este diferencial do StoryToCode com relação ao NCPM é outra das contribuições dessa tese.

Uma das limitações do StoryToCode está associado a extração dos requisitos que são necessários para criação do conjunto de elementos que é feita de forma não automatizada. Outra limitação é a quantidade de retrabalho necessária para completar o código gerado depois das transformações realizadas.

Para tratar essas limitações o StoryToCode passou a permitir o desenvolvimento baseado em arquétipos, que consiste em analisar um conjunto considerável e aleatório de aplicaçẽs desenvolvidas por vários autores, para verificar a repetição de estruturas e comportamentos [31].

Dois exemplos de uso foram realizados para demonstrar a viabilidade de uso do modelo. O primeiro deles consistiu em executar a primeira versão do StoryToCode na construção de um jogo interativo para os contextos de TV, Web e dispositivos móveis. O segundo exemplo de uso permitiu demonstrar a viabilidade de uso do desenvolvimento baseado em arquétipos para minimizar as limitações iniciais do StoryToCode.

1.3 ORGANIZAÇÃO DA TESE

A estrutura deste documento foi definida com o objetivo de fornecer ao leitor subsídios para o fácil entendimento do trabalho realizado. Após a introdução são apresentados três capítulos contendo a fundamentação teórica e revisão da literatura relacionada ao tema. A descrição do trabalho desenvolvido nesta tese encontra-se nos próximos dois capítulos. Após as con-

siderações gerais, são apresentadas as referências bibliográficas utilizadas neste trabalho e os apêndices.

Sendo assim, esta tese está dividida nos seguintes capítulos:

i) Capítulo 1 - descreve a motivação, os objetivos, as contribuições e o escopo da pesquisa, como também a organização da tese.

ii) Capítulo 2 - apresenta o desenvolvimento de aplicações multimídia.

iii) Capítulo 3 - apresenta uma descrição da evolução do processo para produção de conteúdos para programas televisivos e os principais trabalhos relacionados.

iv) Capítulo 4 - apresenta os modelo IDTVS e StoryToCode para estruturação de componentes interativos de TV.

v) Capítulo 5 - apresenta os exemplos de uso realizados para verificar a viabilidade da construção de aplicações multimídia através do IDTVS e StoryToCode.

vi) Capítulo 6 - Este capítulo apresenta as considerações gerais sobre esta tese, apontando as principais contribuições, limitações e as possíveis perspectivas.

Capítulo 2

Este Capítulo apresenta conceitos básicos relacionados ao desenvolvimento de aplicações Multimídia.

DESENVOLVIMENTO DE APLICAÇÕES MULTIMÍDIA

O termo multimídia surgiu a partir da digitalização de conteúdos e da evolução dos computadores pessoais que permitiam o uso de mídias mais sofisticadas como vídeo e áudio. A ideia de multimídia elevou as expectativas para a criação de interfaces de usuário mais inteligentes que poderiam usar comandos acionados por voz e por gestos ou ainda usar gráficos complexos para exibir informações de forma mais intuitiva. Por consequência, existiu também um aumento da expectativa relacionada aos conteúdos e aos impactos do uso de aplicações multimídia. Como qualquer palavra da moda, o termo multimídia foi mal utilizado durante muito tempo. Por exemplo, computadores pessoais foram chamados de sistemas multimídia apenas por conter uma unidade de CD-ROM. A falta de uma definição precisa para multimídia levou alguns autores a afirmarem, no final da década de 90, que esse termo nunca existiu [32] [33] [34].

O fato é que, logo após a fase inicial de euforia com a multimídia, boa parte das aplicações criadas não atendiam às expectativas levantadas. Ao longo dos anos, embora a implementação dessas aplicações tenha se tornado mais fácil, principalmente por conta do apoio de ferramentas e linguagens de programação, o seu uso foi limitado devido à sua complexidade [35] [36].

Este capítulo apresenta definições importantes usadas nesta tese, relativas ao desenvolvimento de aplicações multimídia e, além disto, pontua quais são os principais desafios deste tipo de desenvolvimento e analisa o espectro de possíveis soluções.

As próximas seções abordam o estado da arte da literatura existente sobre os conceitos básicos, as tecnologias, os arcabouços e as APIs relacionados ao desenvolvimento de aplicações multimídia, além de discutir características da produção de conteúdos para estas aplicações.

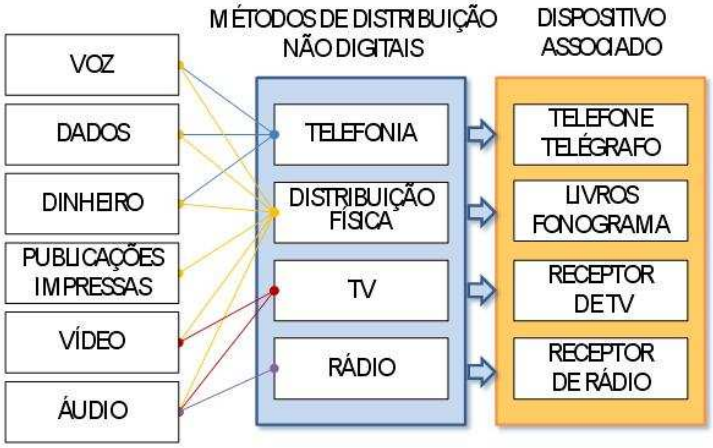

Figura 2.1. Cadeia Produtiva antes da era da digitalização de conteúdos

2.1 PRODUÇÃO DE CONTEÚDO DIGITAL

Antes da era da digitalização de conteúdos, os diferentes serviços de comunicação formavam uma cadeia discreta de componentes que restringiam tipos distintos de conteúdo para redes e terminais específicos. Na Figura 2.1 pode-se observar que, em muitos casos, a empresa fornecedora do serviço estava verticalmente integrada ao longo de toda a cadeia produtiva e impedia que informações fossem facilmente transferidas de um serviço para outro. Especialmente na telefonia, a rede que permitia a transmissão da voz era exclusiva assim como os respectivos terminais (aparelhos telefônicos) também só eram usados para este fim.

A digitalização de conteúdos possibilitou a sua modularização, armazenamento e distribuição por diferentes meios de telecomunicações. Esse fato, aliado à inovação tecnológica das redes de telecomunicações, fez crescer um fenômeno conhecido como convergência. O termo convergência pode ser definido como a união das telecomunicações e conteúdos digitais que produz um novo tipo de interconexão e de interoperabilidade entre diferentes tipos de mídia [1].

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 que podem se tornar: (1) competitivas ou (2) complementares (ou, melhor ainda, dependentes). No segundo, a convergência pode ser vista como uma (re)união de diferentes tipos de mídia por meio de uma tecnologia única. Um exemplo de competitividade criada pela convergência é a do acesso a conteúdos de jornais e outras mídias de comunicação pela Web, que resultou em profundas mudanças nos jornais e agências de notícias, na indústria fonográfica entre outras.

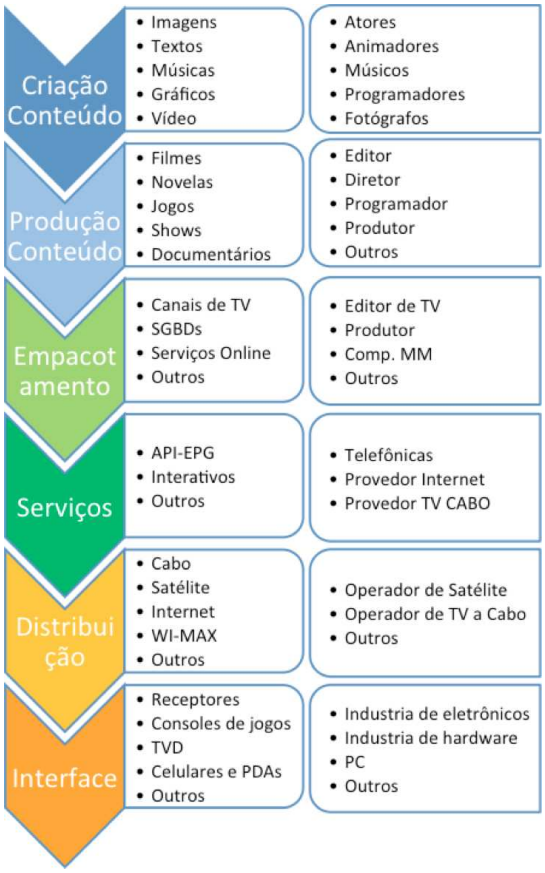

Figura 2.2. Nova Cadeia Produtiva [1]

Recentemente, em julho de 2010, o Jornal do Brasil, que já foi um dos mais importantes jornais da imprensa brasileira, além de ser o primeiro jornal do País a ter versão Web, anunciou o fim da sua versão impressa. Os canais de TV na Web, as transmissões IPTV e os sites provedores de vídeo na internet são os exemplos mais recentes de formas de acesso a conteúdo audiovisual que competem com o modelo tradicional de comunicação de massa da TV broadcast. O processo de convergência, além dos ganhos tecnológicos, possui também uma forte vertente socioeconômica que implica na adoção e uso coordenado de protocolos e padrões comuns pelas partes que se integram. Em outras palavras, produtores e consumidores devem adotar plataformas compatíveis, além de utilizarem padrões específicos (para codificação, transmissão e armazenagem) para permitir o trabalho em conjunto no mundo convergente. Outra consequência da digitalização de conteúdos foi o surgimento do conceito de multimídia. Este conceito, que será detalhado na próxima seção, está normalmente associado a uma apresentação controlada por um sistema computacional, que pode envolver objetos de mídia interativos (como: texto, fotografia, gráfico, vídeo, áudio, animação etc.).

Especificar uma mesma apresentação multimídia para contextos diferentes (como: TV e

Web) envolve o uso de competências multidisciplinares que normalmente não são encontradas nestes contextos. Este fato provocou a substituição da antiga segmentação vertical da cadeia produtiva de conteúdos (Figura 2.1) por um novo modelo apresentado na Figura 2.2 [1] onde a primeira coluna exibe as seis etapas distintas da nova cadeia, a segunda coluna o que é produzido ou executado em cada etapa e a terceira coluna quem são os participantes de cada etapa:

Criação e Produção: refere-se à criação e produção de conteúdo digital codificado em um determinado formato (como filmes, programas de televisão, jornais, artigos, livros, gravações de áudio, sites etc.). Na primeira etapa, os agentes responsáveis pela criação de conteúdo são atores, músicos, animadores, esportistas, engenheiros de softwares entre outros. A etapa de produção é feita por editores de TV, diretores, designers, programadores, produtores etc.

Empacotamento: refere-se aos processos intermediários em que diferentes tipos de conteúdo e / ou software são montados em um único produto ou serviço. Atuam nesse segmento operadores e editores de TV, produtores, empresas de marketing multimídia etc.

Serviços: a camada de serviços de software é essencial para o modelo convergente. De forma geral, o software participa de toda a cadeia produtiva, mas ainda assim pode ser encarado como uma camada individual. São exemplos do segmento de serviços de software: aplicações desktop, protocolos de roteamento e gerenciamento de redes, protocolos de armazenamento e recuperação de informação, protocolos de codificação (ex.: compressão e descompressão) de sinais, motores de indexação etc.

Distribuição: reúne as atividades de transporte da informação através de redes de comunicação (redes de telefonia, TV, sem fio etc.) gerenciadas pelos operadores de cada uma delas.

Interface: reúne os dispositivos de hardware que permitem aos usuários transmitir, receber e exibir sinais, como telefones celulares, televisores, aparelhos de fax, computadores, pagers, switches, roteadores, modems etc.

O desenvolvimento de aplicações multimídia passa pela compreenção de como essas aplicações são construídas na nova cadeia produtiva de conteúdos digitais. Para isso, as próximas seções apresentam definições básicas relacionadas à multimídia e ao desenvolvimento de aplicações.

2.2 MULTIMÍDIA

A palavra multimídia se refere à apresentação ou recuperação de informações que se faz, com o auxílio do computador, de maneira multissensorial, integrada, intuitiva e interativa [37].

No contexto da ciência da computação, o padrão MHEG [38] estabelece que o termo multimídia é classificado na categoria de mídia percepção. Esta categoria está relacionada à forma como um ser humano pode perceber uma porção de informação usando sua visão e/ou audição. São exemplos de mídias voltadas para comunicação visual: texto, imagens estáticas, animações e filmes. Já os exemplos de mídias que usam o sentido da audição são: música, som, e fala (voz) [39]. Mais recentemente, alguns trabalhos passaram também a usar o tato para possibilitar a percepção de informações digitais, como em [40] e [41]. Os outros sentidos humanos (olfato e paladar) são raramente utilizados para percepção de informações digitais nos dias de hoje.

Cada tipo de mídia está associado com até três dimensões espaciais (ex: som(1), imagens(2) e hologramas(3)) e com uma dimensão temporal opcional (ex: vídeo). Dessa forma, uma classificação de mídia importante para o contexto desta tese separa os tipos de mídias em discretas ou contínuas [42]. As mídias contínuas têm uma dimensão temporal e mudam sua representação dinamicamente ao longo do tempo (ex: áudio, vídeo ou animação). Já as mídias discretas são independentes do tempo (ex: texto, imagem e gráficos).

Outra classificação possível é separar os tipos de mídias em capturadas ou sintetizadas [42]. As mídias capturadas são obtidas através de dispositivos que captam e digitalizam informações do mundo real, como por exemplo um vídeo feito a partir de uma câmera filmadora. As mídias sintetizadas são criadas a partir de um computador, como textos, gráficos e animações. Uma característica importante das mídias sintetizadas é que, muitas vezes, elas podem possuir algum tipo de estruturação interna explícita. Por exemplo, uma imagem vetorial consiste em um estrutura de dados organizada, enquanto para as imagens capturadas este tipo estrutura não está disponível.

O entendimento claro do conceito de mídia é importante para a definição de multimídia. Para esta tese, três definições devem ser consideradas:

i) O termo multimídia refere-se a um documento que contém um conjunto de informações codificadas em pelo menos um tipo de mídia contínua e um tipo de mídia discreta [43].

ii) Um documento multimídia é um conjunto de componentes discretos de dados que estão unidos no tempo e espaço e que são apresentados para um usuário (ou leitor) como um todo coordenado [44].

iii) Um documento multimídia é um elemento de mídia, que forma uma composição de elementos contínuos e discretos de mídia em uma unidade multimídia lógica coerente [45].

As três definições citadas tratam multimídia como um tipo de documento e consideram que a interatividade com usuário se resume a: (i) nevegação sobre a apresentação, (ii) adaptação da apresentação (ex.: alterar a resolução de um vídeo) e (iii) controles básicos disponíveis no *palyer* (ex.: inciar, pausar, adiantar etc.). No entanto, este é um entendimento muito restrito de interatividade. A navegação não pode nem mesmo ser chamada de interatividade, pois não permite controlar um objeto de mídia, representado na interface com usuário [46].

A compreensão de multimídia apenas como um tipo de documento, apesar de coerente, pode ser considerada desatualizada nos dias de hoje. Atualmente, criar sistemas multimídia altamente interativos, inteligentes e fáceis de usar, exige, muitas vezes, uma estreita integração com a lógica complexa de um aplicativo de software. Esta nova visão levou ao surgimento do termo aplicação multimídia.

2.3 APLICAÇÕES MULTIMÍDIA

Uma aplicação multimídia é definida como um tipo de software que permite ao usuário interagir com dados multimídia [43]. Para isso, a ideia é combinar multimídia com qualquer tipo de lógica de aplicação. Quando isso ocorre, percebe-se que os aplicativos oferecem uma interface multimídia geralmente mais sofisticada que interfaces de usuário padrão. Três fatores influenciam o uso de uma aplicação multimídia [47] [48]:

i) Aumento da eficiência e produtividade da interface de usuário;

ii) Transferência do conhecimento e de informações de forma mais eficaz, e;

iii) Melhoria do grau de entretenimento de um aplicativo de software.

Outra definição de aplicação multimídia importante pode ser encontrada em [23] onde estas aplicações podem ser entendidas como sistemas de software interativos que combinam e apre-

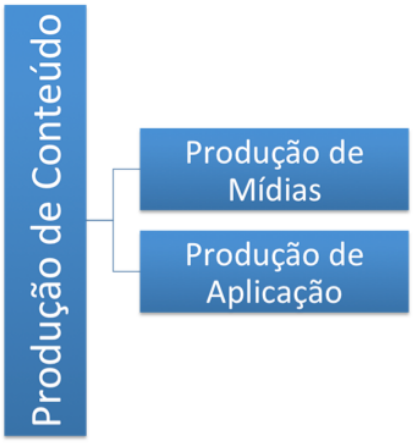

Figura 2.3. Sub-fases da etapa Produção: Produção de Mídias e Produção de Aplicação

sentam um conjunto de objetos de mídia independentes, de diversos tipos, que têm relações espaço-temporais e dependências de sincronização, que podem ser organizados em hierarquias de composições estruturadas, e podem ser acoplados com um modelo de ação baseado em eventos para possibilitar a interação e navegação.

Também é importante posicionar as aplicações multimídia na nova cadeia produtiva de conteúdos digitais (ver seção 2.1). Nesta nova cadeia, as atividades gerais das etapas Criação e Produção referem-se à criação e produção apenas de conteúdo digital codificado em um determinado tipo de mídia. Por exemplo, para uma aplicação de e-learning, o conteúdo de uma aula pode ser codificado em um vídeo.

A Figura 2.3 apresenta duas sub-fases da etapa de produção do conteúdo digital: **Produção de Mídias** e **Produção de Aplicação**. A primeira delas é pré-requisito para as atividades de produção de uma aplicação multimídia que ocorre na sub-fase Produção de Aplicação.

As atividades da sub-fase Produção de Aplicação incluem a seleção e integração de objetos de mídia previamente codificados (para a criação da interface do usuário e dos mecanismos de interação) bem como a criação da lógica da aplicação e sua integração com os objetos de mídia selecionados. Conforme dito anteriormente, apesar do software participar de toda a cadeia produtiva, como por exemplo na etapa de Serviços (ex.: protocolos de roteamento e gerenciamento de redes, padrões de codificação etc.), esta tese considera como uma aplicação multimídia apenas o resultado (produto) da sub-fase Produção de Aplicação.

Vale observar que as tarefas de desenvolvimento na sub-fase Produção de Aplicação devem combinar pelo menos dois aspectos [35]: design criativo e desenvolvimento de softwares.

O design criativo é necessário pois, os desenvolvedores têm que criar a interface do usuário, integrar o conteúdo multimídia a esta interface e ainda, em alguns casos, é necessário pós-editar o conteúdo existente ou criar conteúdo adicional. Por outro lado, de forma semelhante ao desenvolvimento de software convencional, os desenvolvedores têm que criar a lógica da aplicação e de interatividade e integrá-las com a interface do usuário. Neste contexto, o uso de ferramentas de apoio para autoria multimídia e para programação facilita a execução das tarefas.

2.4 DISCUSSÃO: DESAFIOS NO DESENVOLVIMENTO DE APLICAÇÕES MULTIMÍDIA

Ao longo dos anos, muitos trabalhos foram focados no desenvolvimento de serviços e tecnologias de sistemas multimídia como bancos de dados e redes, mas poucos trabalhos trataram o desenvolvimento de aplicações multimídia [23]. Uma das razões para isso é o alto custo/esforço para a criação de interfaces multimídia de usuário sofisticadas e que estejam totalmente integradas com a lógica de uma aplicação [36].

2.4.1 Falta de Métodos Específicos

Depois da fase de crescimento do uso de aplicações multimídia observado durante a década de 90, muitos trabalhos encontrados na literatura já advertiam sobre a falta de uma abordagem sistemática no desenvolvimento de aplicações multimídia como, por exemplo, [49], [50], [51], [33], [52], [34], [53] e [23]. No trabalho [50], os autores sinalizavam para a situação do desenvolvimento de aplicações multimídia, na qual a total ausência de métodos e processos sistemáticos levava a projetos excessivamente complexos com aumento do tempo de trabalho necessário para sua conclusão.

De forma geral, esses autores afirmam que, naquele período, um dos principais problemas para o desenvolvimento de aplicações multimídia era a falta de apoio nas pré-fases de implementação que levava a um processo de desenvolvimento ad-hoc, com resultados não-

estruturados, difíceis de se compreender, manter e estender. Não existiam processos que ajudassem a construir uma ponte entre a estruturação de requisitos e a implementação. Além disso, os documentos resultantes do que se poderia chamar de processo de estruturação de requisitos eram bastante informais como, por exemplo, storyboards [54]. Como consequência, eles poderiam facilmente tornar-se ambíguos e inconsistentes em diferentes níveis de abstração.

Alguns autores tratam o processo de produção e apresentação coordenada de mídias como [55] e [56]. Em [57] os autores propõem um modelo configurável de produção multimídia composto por nove processos canônicos (premeditar, criar, anotar, empacotar, consultar, construir mensagens, organizar, publicar e distribuir). Em [58] os autores apresentam um processo voltado ao usuário final que permite enriquecer uma apresentação multimídia. Porém, nenhum desses trabalhos define fases específicas para o tratamento da lógica contida nos módulos de software de uma apresentação multimídia.

O problema com o desenvolvimento de aplicações multimídia não é apenas a ausência de modelos de processos de desenvolvimento sofisticados, mas também a falta de notações visuais utilizáveis para permitir uma especificação integrada de um sistema em diferentes níveis de abstração e a partir de diferentes perspectivas [23]. O paradigma “implementar e testar”, utilizado nos dias de hoje para a autoria de aplicações multimídia, assemelha-se ao estado de desenvolvimento de aplicações antes da crise do software da década de 80 [59]. O caminho para sair deste estado passa pela evolução de técnicas, ferramentas e modelos que possam auxiliar o processo de desenvolvimento. Nesse contexto, o trabalho publicado por [60] ressalta que novas abordagens para o desenvolvimento de aplicações multimídia devem ser práticas e facilmente utilizáveis. Para isso, é necessária boa documentação e uma boa aceitação da indústria.

2.4.2 Caráter Multidisciplinar

Outro problema associado ao desenvolvimento de aplicações multimídia interativas é a necessidade de integrar o uso de ferramentas, profissionais e conhecimentos de áreas muitas vezes completamente diferentes. Podem estar associadas à produção de aplicações multimídia: a criação de animações e de gráficos; a gravação/edição de vídeos e de áudios; habilidades de atuação, de direção, de redação e de produção artistica; conhecimentos de legislação sobre direitos autorais; habilidades de programação de computadores e de design de interface de usuário; conhecimentos sobre marketing e habilidades comerciais, entre outros aspectos [61]. Coordenar

equipes diferentes no desenvolvimento de aplicações multimídia é uma atividade importante e difícil.

Alguns dos aspectos citados também são comuns ao processo tradicional de desenvolvimento de sistemas, mas alguns são específicos de aplicações multimídia. Em [35] [62] os autores separam esses aspectos em duas categorias: (i) design de software e (ii) design de mídias. Outros autores consideram que, como o design da interface de usuário não faz parte obrigatoriamente das atividades relacionadas ao software e nem daquelas associadas à produção de mídias, ele deve ser considerado também como uma categoria à parte das demais [34]. Assim, observam-se três categorias para as tarefas de desenvolvimento de aplicações multimídia:

i) **Design de software**: agrupa as tarefas de desenvolvimento necessárias para conceber e produzir softwares convencionais que fazem parte de uma aplicação multimídia como, por exemplo, a lógica de uma aplicação [63];

ii) **Design da interface de usuário**: agrupa as tarefas de desenvolvimento necessárias para conceber e produzir a interface de usuário, de acordo com os princípios de interação humano-computador (IHC), como em [64] e [65];

iii) **Design e integração de mídias**: agrupa as tarefas de desenvolvimento específicas de multimídia necessárias para conceber, produzir e compor objetos mídia. Além disso, contempla a integração (ex: sincronização espacial e/ou temporal) desses objetos com uma aplicação.

Outras categorias também poderiam ser consideradas, como o gerenciamento de projetos e o design de componentes de hardware específicos para aplicações multimídia [34]. Porém, esta tese trata apenas dos aspectos de software relacionados ao desenvolvimento de aplicações. Assim, os métodos para o desenvolvimento de aplicações multimídia deve considerar o design de software, o design da interface de usuário, o design de mídias e sua integração com o software.

2.4.3 Módulos de Apoio para Ferramentas de Autoria

Uma das dificuldades para o desenvolvimento de aplicações multimídia diz respeito às ferramentas de autoria existentes. Normalmente, as aplicações multimídia são desenvolvidas utili-

zando ferramentas de autoria [23] [36], que são necessárias para as tarefas de design de mídias e de interface de usuário.

Um problema bem conhecido de ferramentas de autoria é o dilema de decidir se o foco da ferramenta deve ser o suporte à programação ou à edição de documentos. Por um lado, uma ferramenta com a estrutura de suporte para programação embutida pode facilitar a manutenção de software, mas isso implica no aumento da curva de aprendizado para os usuários finais (ex.: designers), que provavelmente serão os principais usuários das ferramentas. Por outro lado, uma ferramenta de autoria sem este suporte pode ser mais intuitiva e fácil de usar, mas também suscetível a problemas de manutenção de software. O uso de padrões e modelos de software pode ajudar a reduzir este problema [66].

Outro aspecto negativo também associado ao uso de ferramentas de autoria é a falta de módulos voltados ao controle de versão e configuração. Mesmo em ferramentas profissionais como Flash [67] o controle de versão permite apenas o bloqueio de arquivos mas não oferece funcionalidades básicas como comparação e recuperação de arquivos. O uso de ferramentas de controle de versão externas pode ajudar significativamente mas não resolve este problema, porque as ferramentas de autoria geralmente usam um formato de arquivo binário proprietário que dificulta operações de comparação e recuperação.

2.5 DESENVOLVIMENTO DE APLICAÇÕES MULTIMÍDIA

Como discutido na seção anterior, existe a necessidade de melhor integrar os princípios de desenvolvimento de softwares com os de desenvolvimento de aplicações multimídia. Possíveis soluções devem se ajustar às características específicas e aos desafios do desenvolvimento multimídia como, por exemplo, o seu caráter altamente interdisciplinar.

Ao longo dos anos, alguns trabalhos foram propostos para integrar o desenvolvimento multimídia com o de software. Por exemplo, o uso de métodos formais baseados em lógica matemática já foi proposto para o desenvolvimento multimídia (ex: [68], [69] e [70]). Eles permitem um processo de desenvolvimento muito bem estruturado, com um alto grau de automação e de validação de um sistema desenvolvido. A desvantagem dos métodos formais é o elevado esforço necessário para sua aprendizagem, bem como para a sua aplicação, no ambiente multidisciplinar das aplicações multimídia. Além disso, o foco de uma aplicação multimídia, muitas vezes

está associado a requisitos não-funcionais que são difíceis de medir e formalizar como: efeitos visuais, transferência de informação eficiente, usabilidade etc.

Os métodos ágeis [71] por sua abordagem menos formal e sua estreita relação os usuários, eles são, obviamente, candidatos ao uso no desenvolvimento de aplicações multimídia. O uso de metodologias ágeis torna o problema da falta de modelos e métodos específicos para multimídia obsoleto, pois não exige qualquer tipo de artefato de alto nível de design. Além disso, abordagens ágeis se encaixam perfeitamente com a elevada ocorrência de mudanças de requisitos no desenvolvimento de aplicações multimídia. No entanto, existem também algumas sérias dificuldades para aplicar tais abordagens neste contexto.

Algumas abordagens ágeis, como a Extreme Programming (XP) [72], listam vinte práticas que devem ser seguidas para compensar o processo menos formal e garantir a qualidade dos produtos gerados. O problema é que algumas dessas práticas são difíceis de aplicar em projetos multimídia. Por exemplo, os projetos feitos a partir da XP devem ter todos os testes (de unidade e funcionais) escritos antes de implementar as suas respectivas funcionalidades. Em projetos de aplicações multimídia isso só é possível de forma limitada uma vez que os requisitos são pouco estruturados e as ferramentas de autoria nem sempre contemplam o suporte a módulos de automação de testes. Certamente, as abordagens ágeis de desenvolvimento podem ser úteis para o desenvolvimento de aplicações multimídia desde que adaptações para este contexto sejam realizadas.

Os processos iterativos representam outra abordagem convencional da engenharia de software que pode ser considerada o meio termo entre os métodos formais e métodos ágeis. Estes processos incluem uma fase de projeto de software onde um sistema é especificado de maneira semi-formal, como, por exemplo, usando linguagens de modelagem visual. Um caso frequentemente encontrado na literatura é o processo unificado (RUP) [73] que usa a linguagem de modelagem visual UML (Unified Modeling Language) [74]. O uso de linguagens de modelagem ganhou popularidade na última década e é algumas vezes citado como padrão de fato para o desenvolvimento de software orientado a objetos.

Uma utilização de RUP específica para aplicações multimídia, denominada de processo SMART, foi apresentada por [75]. No entanto, o principal problema dessa abordagem reside na falta de conceitos de modelagem visual específicos para aplicações multimídia. O processo SMART usa a UML, que não é suficiente para modelagem multimídia, pois nem suporta os

conceitos concretos para a modelagem da interface de usuário, nem para a modelagem de mídias.

Visando resolver esta limitação, uma extensão (profile) da UML, específica para modelagem multimídia chamada de MML (Multimedia Modeling Language), foi apresentada por [76]. A MML visa apoiar o processo de design no desenvolvimento de aplicações multimídia e faz parte de uma abordagem de desenvolvimento orientado por modelos. Ela integra os resultados de duas linhas de pesquisa diferentes: aplicação de modelagem orientada a multimídia e modelo de desenvolvimento baseado em interface de usuário.

O uso de modelos visuais oferece (pelo menos parcialmente) as mesmas vantagens dos métodos formais, como em especificações não ambíguas e também a possibilidade de processamento automático para geração e validação de código fonte. Para esta tese, a capacidade de gerar automaticamente o código ou esqueletos de código a partir de modelos é um conceito importante. Outra vantagem é representação visual baseada em diagramas que é mais familiar tanto para desenvolvedores quanto para o restante da equipe multidisciplinar de um projeto multimídia.

A geração automática de código a partir de modelos não só pode melhorar significativamente a eficiência no desenvolvimento, mas também pode aumentar a qualidade de código gerado que é mais consistente com as especificações de projeto. Além disso, um modelo que permite gerar código para uma plataforma específica pode ser automaticamente aproveitado para projetos futuros, inclusive em plataformas diferentes. Isto é particularmente útil para projetos de multimídia, cujas plataformas de implementação exigem conhecimento avançado sobre a estruturação do código.

2.6 RESUMO

Este capítulo apresentou uma revisão da literatura sobre aplicações multimídia interativas. As seções anteriores elaboraram inicialmente duas definições importantes: (i) uma aplicação multimídia é qualquer aplicação com uma interface (de usuário) multimídia que (ii) tem como foco principal melhorar a interação com os seus usuários.

Do ponto de vista do desenvolvimento de softwares, uma aplicação multimídia interativa geralmente envolve o design de mídias, o design da interface de usuário e integração desses

elementos com a lógica de aplicação. Ela é implementada com o apoio específico para implementação multimídia. No entanto, quando se olha para aplicações multimídia avançadas, objetos de mídia nem sempre são concebidos como parte de um processo de design de mídia, mas também podem ser gerados em tempo de execução.

Para esta tese, uma aplicação multimídia é qualquer aplicação que: (i) tem uma interface de usuário não padronizada pois usa uma quantidade relevante de objetos de mídia como: áudio, vídeo, imagem, gráficos 2D e 3D, e animações; (ii) esteja fortemente acoplada com a lógica da aplicação e (iii) normalmente é desenvolvida em uma sub-fase da etapa de produção na nova cadeia de conteúdos digitais.

Conforme apresentado neste capítulo, diversas tecnologias, disciplinas e pessoas estão envolvidas no processo de desenvolvimento de aplicações multimídia. Apesar disso, metodologias de desenvolvimento específicas para multimídia que contemple este ambiente multidisciplinar ainda representam um desafio em aberto.

Esta tese trata da produção de uma categoria especial de aplicações multimídia: programas interativos para TV Digital. O desenvolvimento desses aplicativos apresentam problemas semelhantes ao desenvolvimento de aplicações listados neste capítulo, além de outros que são específicos do ambiente de TV.

O próximo capítulo apresenta uma revisão detalhada dos principais tópicos relacionados ao desenvolvimento de programas de TV convencionais e discute a evolução para TV digital interativa.

Capítulo 3

Este capítulo apresenta uma descrição da evolução do processo para produção de programas televisivos descrevendo as atividades, os profissionais envolvidos (atores), com suas respectivas responsabilidades (papéis), e artefatos envolvidos na geração de conteúdo multimídia para TV.

GERAÇÃO DE CONTEÚDO PARA TV NA ERA DA TV DIGITAL

Criada em 18 de setembro de 1950, a primeira emissora de televisão no Brasil foi trazida por Assis Chateaubriand. Desde então, a televisão cresceu no país e hoje representa um fator importante na cultura popular moderna da sociedade brasileira. Durante esse período é possível observar a sua evolução a partir de 4 grandes mudanças: i) a primeira delas foi a passagem da TV ao vivo para a TV gravada em meio magnético; ii) depois disso aconteceu a passagem da TV preto-e-branco para a TV em cores; iii) em seguida houve a integração com circuito mundial de transmissão via satélite e iv) mais recentemente, a migração da TV analógica para a TV digital.

Apesar de toda evolução observada no universo de TV desde sua criação como, por exemplo, a incorporação de novos equipamentos e de novos profissionais, o formato dos programas de TV bem como seu processo de desenvolvimento se manteve estável. Na concepção tradicional, desenvolver um programa de TV pode ser visto como a união de esforços de equipes que trabalham de forma sincronizada. A produção de um programa de TV inclui: (i) a seleção dos recursos humanos envolvidos; (ii) a definição do fluxo de trabalho e a geração dos artefatos auxiliares; (iii) a produção e a gravação do programa de TV e (iv) a finalização [77].

Diferente da TV convencional, em um programa TV digital existe possibilidade de integrar softwares e dados aos fluxos de áudio e vídeo transmitidos pelas emissoras. Essa é uma das maiores inovações trazidas pela TVD e isso cria oportunidades para que novos modelos de

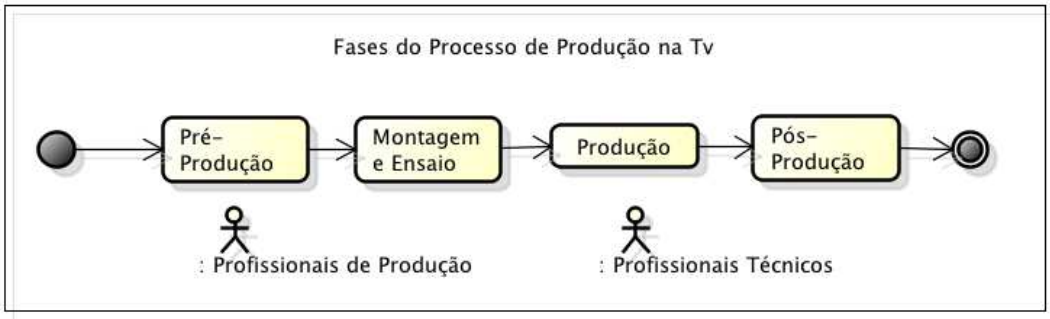

Figura 3.1. Fases da produção de programas para TV convencional

produção, novos serviços e aplicações dos mais diversos tipos sejam desenvolvidos para esta nova mídia interativa.

Este capítulo apresenta uma descrição da evolução do processo para produção de programas televisivos descrevendo as atividades, os profissionais envolvidos (atores), com suas respectivas responsabilidades (papéis), e artefatos envolvidos na sua produção. A descrição contempla a produção de TV convencional e TV Digital, além de fazer uso dos diagramas UML para facilitar o mapeamento do modelo de processo no universo de TV com a realidade de software. Por fim, o capítulo apresenta os principais trabalhos relacionados com a tese.

3.1 TV CONVENCIONAL

A Figura 3.1 apresenta o diagrama de atividades para a produção de programas de TV convencional. No diagrama cada atividade corresponde a uma das quatro fases do processo de produção: (i) pré-produção, (ii) montagem e ensaio, (iii) produção e (iv) pós-produção. Cada uma delas envolve a participação de profissionais com responsabilidades (papéis) diferentes e envolve a produção de vários artefatos. Ainda de acordo com o diagrama, o conjunto de profissionais envolvidos na construção de um programa de TV convencional pode ser dividido em dois grupos: (i) produção e (ii) técnico. Esta seção detalha os papéis, profissionais, artefatos e fases envolvidas no processo de produção de um programa de TV convencional.

3.1.1 Profissionais e Papéis

O grupo de produção assume a responsabilidade das funções administrativas como: conhecer a capacidade e limitação da tecnologia/equipamento a ser aplicado na produção do pro-

grama, saber trabalhar em equipe de forma colaborativa e descentralizada, saber coordenar pessoas e atividades, conhecer e saber gerenciar orçamento e custo, além de cumprir cronogramas. A figura 3.2 exibe os profissionais que fazem parte do grupo de produção:

Produtor: É papel do produtor desenvolver o conceito e o orçamento do programa. É dever desse profissional: garantir que a produção do programa cumpra o cronograma e o orçamento; coordenar a equipe de produção; aprovar a versão final do programa; coordenar a divulgação e publicidade do programa; verificar se o objetivo do programa foi atingido; testar a aceitabilidade do programa por meio de análise de audiência.

Diretor: É responsabilidade do diretor: apoiar o roteirista na elaboração do roteiro; gerenciar a fase de montagem e ensaio; executar a fase de produção; supervisionar o processo de edição.

Gerente de palco: O papel do gerente de palco é garantir o sucesso de todas as atividades no palco/estúdio e apoiar o diretor nas atividades relacionadas a esse ambiente.

Roteirista: O roteirista tem a responsabilidade de, junto com o produtor e o diretor, desenvolver o roteiro do programa e acompanhar as fases de montagem, ensaio e produção a fim de ajustar o roteiro, no que for preciso.

Assistente de direção: O assistente de direção tem a responsabilidade de apoiar o diretor na execução de suas tarefas e garantir o cumprimento do tempo estabelecido para o programa.

Assistente de produção: Os assistentes de produção servem como equipe de apoio ou grupo produção na execução de atividades associadas a todo o processo.

O grupo técnico tem como principais responsabilidades o domínio sobre os equipamentos e a tecnologia empregada, forte compromisso com prazo e espírito de colaboração. A figura 3.3 exibe os profissionais que fazem parte do grupo de técnicos:

Diretor técnico: É responsabilidade do diretor técnico: garantir a harmonia do grupo técnico com o de produção; coordenar o grupo técnico; garantir a qualidade técnica do programa; apoiar o grupo de produção na criação dos requisitos do programa; garantir a disponibilidade de todos os equipamentos necessários para a produção do programa.

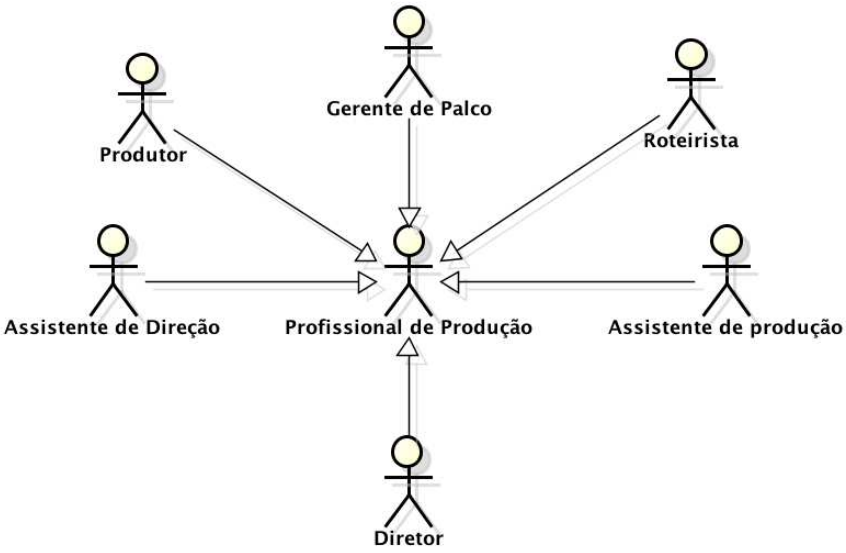

Figura 3.2. Profissionais do grupo de produção

Diretor de fotografia/Iluminador: O diretor de fotografia/iluminador deve apoiar o grupo de produção na definição do conceito e do projeto do programa, além de preparar projeto de iluminação e operar painel de que controla a iluminação no ambiente de gravação.

Diretor de imagens: É papel do diretor de imagens garantir a qualidade técnica do programa e apoiar, como consultor, o grupo de produção no que diz respeito aos aspectos técnicos associados as imagens da produção de TV.

Operador de áudio: É responsabilidade do operador de áudio apoiar o grupo de produção na gravação (captura) de áudio do programa e coordenar a equipe de áudio. Além disso, esse profissional é responsável por mixar todo o áudio do programa e apoiar a sua edição na fase de pós-produção.

Operador de câmera: O operador de câmera é responsável por ensaiar os movimentos e enquadramentos e operar a câmera durante a fase de produção.

Operador de vídeo: É papel do operador de vídeo montar e alinhar as câmeras além de ajudar o diretor a atingir os efeitos visuais especiais.

Cenógrafo: O cenógrafo deve apoiar o grupo de produção na concepção e desenho do cenário. Esse profissional deve prototificar os cenários, apoiar as mudanças necessárias a esse ambiente durante a fase de produção, coordenar a sua construção e desmontagem.

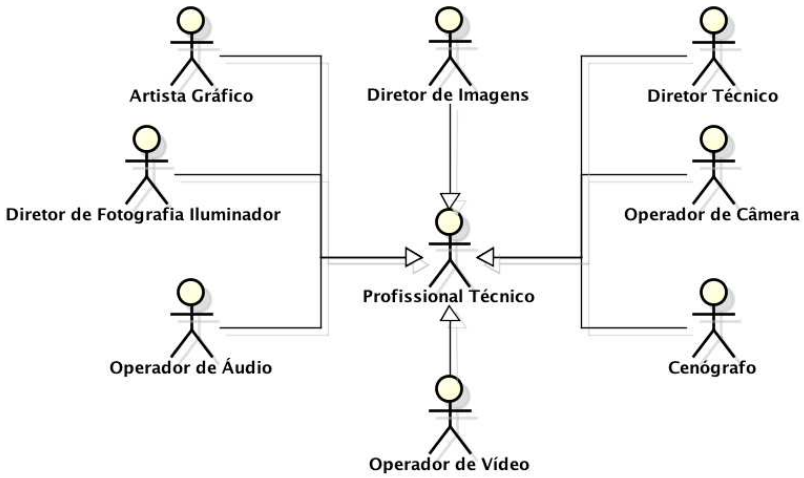

Figura 3.3. Profissionais do grupo técnico

Artista gráfico: É responsabilidade do artista gráfico apoiar o grupo de produção na construção do projeto gráfico do programa. Além disso, é dever desse profissional operar geradores de gráficos que podem ser adicionados no momento da edição.

3.1.2 Fases e Artefatos

A primeira fase para construção de um programa de TV convencional é a **pré-produção**. É nesta fase que ocorrem a criação e organização do conceito do programa e a definição da metodologia de trabalho e da estratégia do programa. Os papéis envolvidos nesta fase envolvem profissionais tanto o grupo de produção quanto o técnico, mas dois deles merecem destaque: o produtor e o roteirista.

Na fase de pré-produção é responsabilidade do produtor desenvolver o conceito, a montagem e o monitoramento do orçamento, a escolha do diretor, entre outras atividades. Um dos artefatos importantes gerados nesta fase é o **orçamento**, que contém as informações financeiras relacionadas com os custos da produção. Neste artefato são discriminadas as estimativas e os custos reais com viagens e deslocamentos para gravações feitas fora do estúdio, a folha de pagamento dos grupos (produção e técnico), a montagem do cenário e figurinos e os custos com a gravação. Uma boa definição do orçamento depende da integração do produtor com os membros-chave do grupo produção (diretor, iluminador, cenógrafo, diretor técnico, técnico de áudio), pois eles são responsáveis pelo levantamento das informações que fazem parte do estudo de viabilidade do programa.

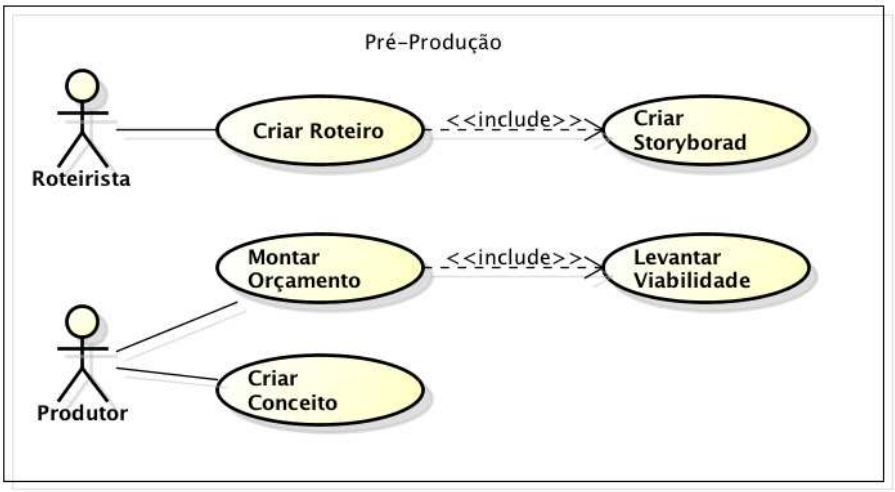

Figura 3.4. Fase de Pré-Produção

Outro profissional de destaque na fase de pré-produção é o roteirista, cujo papel é criar do **roteiro**. Esse artefato, o mais importante na elaboração de um programa de TV convencional, permite o mapeamento da ideia do programa em palavras, sons e imagens, além de conter um conjunto explícito de metas e instruções para os grupos de produção e técnico. A elaboração de um roteiro: (i) deve usar uma linguagem acessível ao público alvo do programa, (ii) deve ser concisa, (iii) deve conter termos que toda a equipe entenda e (iv) deve fazer uso da técnica de **storyboard**.

Um storyboard é uma técnica usada na descrição da sequência fundamental de cenas que melhor representam um programa [54] de TV. O storyboard, feito pelo roteirista com o auxílio do artista gráfico, visa auxiliar a compreensão do roteiro e a sua elaboração. O estilo de roteiro varia de acordo com o tipo de programa, mas todos eles devem, no mínimo, conter o diálogo falado, as informações de som e música, os elementos visuais que acompanham o áudio e detalhes importantes de produção como o tempo de duração do segmento e as fontes de áudio e vídeo. A qualidade do roteiro define a eficiência da fase de pré-produção, pois acredita-se que um roteiro bem planejado normalmente implica em diminuição de custo do programa nas fases posteriores. A figura 3.4 exibe o diagrama de caso de uso simplificado da fase de Pré-Produção, pois além do roteiro e do orçamento, são gerados na fase de pré-produção:

Agenda de produção: Esta agenda contém todas as atividades da produção (o que deve ser feito, quem faz o que, o prazo de cumprimento da atividade e como uma atividade é integrada à outra) e a sequência adequada para sua execução. Ela ajuda ao produtor a

manter o controle da execução das diversas atividades a realizar.

Lista de elenco/equipe: Este artefato contempla a listagem completa dos nomes, números de telefone e endereços de todos os membros do grupo de produção e técnico, assim como do elenco. Além disso, caso a produção seja feita fora do estúdio, este artefato deve conter os endereços de todos locais de hospedagem que abrigarão os profissionais.

Agenda dos ensaios: A agenda dos ensaios contém o detalhamento das atividades que serão realizadas pelos profissionais envolvidos no programa. Esta agenda deve apresentar uma identificação do programa e o cronograma (data, horário e local) de todos os ensaios.

Ficha de requisição operacional: Este artefato representa uma descrição detalhada de toda a estrutura de produção necessária para o programa. Esta ficha apoia o produtor e o diretor na checagem dos equipamentos e pessoal técnico necessário para a contratação. Deve conter a descrição do programa, a data de requerimento do equipamento, tempo de utilização, descritivo técnico dos equipamentos assim como a quantidade de cada um.

Agenda de elenco: Este artefato contempla a agenda individual de cada membro do elenco. O objetivo dessa agenda é disciplinar o elenco e a equipe de apoio tanto na fase de ensaio quanto na de produção. As informações contidas nessa agenda são: a identificação do programa, a descrição das cenas e o respectivo horário de apresentação do elenco.

Agenda das gravações: Esta agenda guarda a ordem e a duração das gravações das cenas, a identificação do programa, a data e horário da gravação, o local de gravação, os indicativos da cena (ex.: se é uma cena noturna ou se é uma cena gravada no interior de um estúdio ou se é uma cena externa), a sinalização das locações e do elenco necessários, assim como o cenário e os equipamentos que serão utilizados.

Autorização de direitos de som e imagem: Este artefato registra a autorização de direito de imagem para todas as pessoas que forem aparecer nas cenas. Além disso, ele pode registrar a autorização de uso para todas as músicas utilizadas no programa. O documento deverá conter um texto básico que concede o direito de som e imagem ao programa e deverá ser assinado pelo participante, pelo autor (no caso de músicas, textos literários etc.), pelo produtor e por testemunhas.

Depois da fase de pré-produção, chega-se à fase da **montagem e ensaio**. Nesta fase ocorrem a montagem do estúdio e testes (ensaios) tanto da estrutura física (iluminação, cenário etc.)

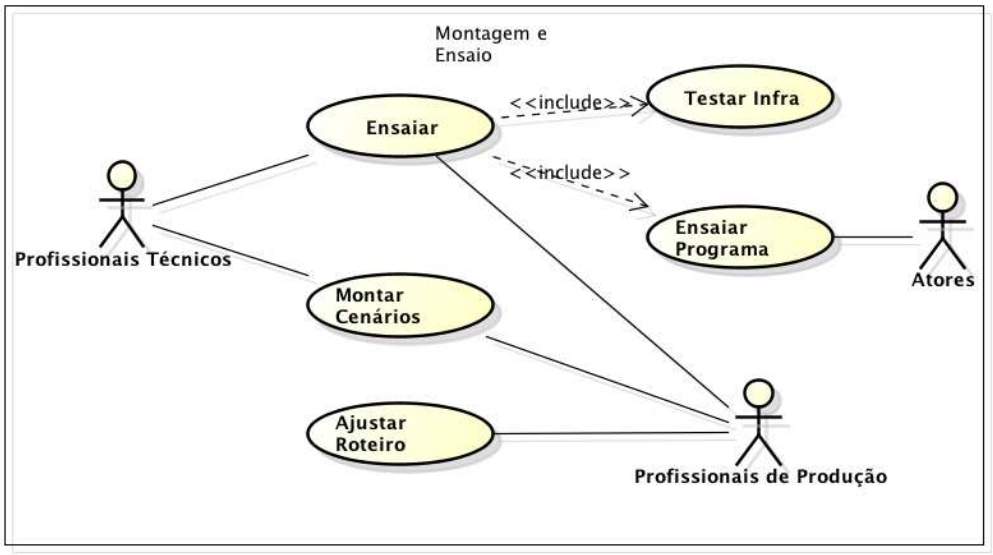

Figura 3.5. Fase de Montagem e Ensaio

quanto da dinâmica do programa antes da sua gravação ou exibição (no caso uma locação ao vivo). O papel do produtor nesta fase é liderar os grupos de trabalho (iluminador, cenógrafo, diretor técnico, técnico de áudio, atores e demais profissionais). O papel do roteirista é ficar atento a possíveis alterações para ajustar o roteiro. Como produto final, tem-se o alinhamento dos grupos de trabalho envolvidos no programa assim como o ambiente físico necessário estruturado. A figura 3.5 exibe o diagrama de casos de uso dessa fase.

Uma vez concluída a montagem e ensaio, chega-se à fase de **produção** do programa de TV convencional. Essa fase ocorre durante a gravação de um programa ou durante a sua exibição (para programas ao vivo). Nesta fase, com exceção do roteirista, todos os profissionais são envolvidos, com destaque especial para o produtor e o diretor. É papel atribuído a esses dois profissionais a tomada de decisões estratégicas durante o processo de gravação ou exibição. Por exemplo, cabe aos diretores e produtores decidir se o programa deve ser gravado sem paradas/edições ou gravado em segmentos (com uma ou múltiplas câmeras). Como produto final, tem-se a **fita de vídeo gravada com programa** que ainda pode ser editado antes de ser exibido ou o **programa ao vivo no ar**.

A última fase da construção de um programa de TV convencional é a **pós-produção**. Nesta fase ocorre a desmontagem de cenários, a desmobilização de equipamentos, a edição e a sonorização do programa (quando gravado) para uma fita máster e, também, a avaliação dos resultados obtidos. Na pós-produção é responsabilidade do produtor e do diretor aprovar a versão final do programa editado, coordenar a divulgação e a publicidade do programa além de avaliar o al-

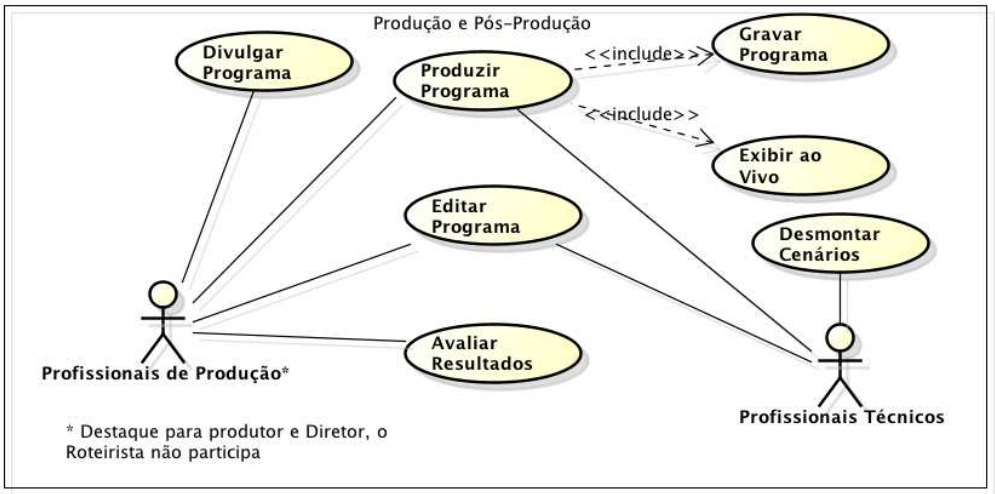

Figura 3.6. Fases de Produção e Pós-Produção

cance dos objetivos previamente concebidos na fase de pré-produção. No final desta fase o programa está pronto para a reprodução. A figura 3.6 exibe o diagrma de casos de uso das fases de produção e pós-produção.

3.2 TV DIGITAL

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. Além da Internet, um dos impulsionadores mais recentes deste fenômeno é a TV Digital, que aparece como um veículo com potencial para desempenhar papel importante no uso dessas tecnologias.

Dentre os principais benefícios proporcionados pela TV Digital estão a melhoria na qualidade da imagem e do som e a 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 etc.) e de oferecer programas interativos. Estes programas, que também são chamados de “programas de TV Digital Interativos” (TVDI), criam a possibilidade de veiculação de softwares interativos e personalizados junto com qualquer outro programa da grade de uma emissora [78].

Uma definição precisa do termo TVDI ainda é uma questão que está longe de ser resolvida. A TV Interativa é algo que permite que o telespectador ou telespectadores e as pessoas

que fazem um canal de televisão, programa ou serviço se engajem em um diálogo [22]. Mais especificamente, pode ser definida como um diálogo que leva os telespectadores a além da experiência passiva de assistir e os permita fazer escolhas e tomar ações. A TVDI é um termo genérico utilizado para todos os sistemas de televisão que oferecem ao consumidor interatividade além da troca de canais e do teletexto [21]. A TVDI é a coleção de serviços que suporta escolhas e ações iniciadas pelo telespectador e que são relacionadas a um ou mais canais de programação de vídeo [79]. Além disso, termos como Web TV, TV na Internet, TV conectada e outros são muitas vezes utilizados como sinônimos de TVDI.

A inclusão da interatividade nos programas de TV deve se tornar um grande atrativo para as indústrias de radiodifusão e para os geradores de conteúdo que estão sempre em busca de novas maneiras de fidelizar seus espectadores. Alguns exemplos recentes de programas líderes de audiência utilizam a colaboração (interação) do telespectador para: (i) por meio de mecanismos de preferência popular, definir os rumos do programa (Big Brother Brasil); (ii) produzir e enviar conteúdos a serem integrados aos programas (Quadro Bola Cheia Bola Murcha do Fantástico); ou ainda (iii) interagir com participantes de um programa (mensagens com questões para os comentaristas durante uma partida de futebol).

Este novo desenho de conteúdo televisivo tem impactos diretos na forma pela qual se assiste a esse conteúdo (recepção), como também na forma com a qual este conteúdo é feito (produção). O processo de produção de conteúdo para TV passa a englobar o desenvolvimento de software (interatividade) e, portanto, passa a ser caracterizado como um novo processo, não linear, iterativo, ágil, multidisciplinar e convergente.

3.2.1 Geração de Conteúdo para TV Digital

Além a melhoria da qualidade do conteúdo visual por meio da digitalização do sistema de TV, novos problemas devem ser enfrentadas pelas emissoras na geração de conteúdo interativo para TV digital. A Figura 3.7 exibe a arquitetura básica do sistema de TV Digital dividida em 6 partes. No modelo de *broadcast* tradicional, toda a produção do conteúdo fica sob responsabilidade da emissora (parte 1: Ilha de Edição e Aplicativos), a qual integra uma série de elementos (ícones, vinhetas, animações) sobre um conteúdo único audiovisual (parte 2: Multiplexador), que é transmitido sobre o canal controlado pela emissora (parte 3: Modulador). Nesse modelo, o receptor de TV sintonizado no canal da emissora decodifica o sinal e apresenta o conteúdo

Figura 3.7. Arquitetura Básica de TV Digital [2]

integrado produzido a uma certa taxa de quadros e de forma sincronizada com o áudio correspondente (parte 4: Set-Top-Box e DTV). Quando conteúdos interativos são enviados pelas emissoras, porções (no tempo e no espaço) do conteúdo integrado podem gerar algum tipo de processamento (surgimento de informações extras na tela, disparo de um elo, mudança de ângulo da câmera) ao serem acessadas pelo usuário-telespectador através do seu controle remoto ou outro dispositivo conectado ao receptor como, por exemplo, Palms, Celulares etc (parte 5: Dispositivos). As aplicações interativas enviadas pelas emissoras podem ainda fazer uso dos canais de interatividade disponíveis (parte 6: Canal de Retorno e Internet) para acessar conteúdos oriundos de outros geradores (diferentes da emissoras) como portais de notícias, servidores de vídeo (ex: youtube), redes sociais entre outros. Para que esse processamento seja possível, uma série de requisitos deve ser satisfeitos, tanto da parte de quem gera, de quem transmite, quanto de quem recebe e processa essas aplicações.

Para que todas as partes da cadeia de produção, distribuição e consumo de conteúdos audiovisuais interativos para TV digital se encaixem perfeitamente, é necessário que essas partes utilizem padrões comuns. Nesse sentido, o Sistema Brasileiro de TV Digital (SBTVD ou ISDB-Tb) especifica vários padrões de referência, dentre os quais se destacam o H.264 [80] para a codificação de vídeo, o MPEG-4 [81] para o áudio, além do MPEG-2 System [82] para o

transporte (multiplexação) dos fluxos de áudio, vídeo e dados. Dentre os padrões estabelecidos, a camada middleware Ginga [83] é, sem dúvida, a mais importante diferença do SBTVD com relação aos outros padrões internacionais.

3.2.2 Programas de TVDI

O uso do termo "programa de TVDI" pode ter significados diferentes, pois a palavra "programa", que normalmente é usada como sinônimo de software ou aplicação no contexto da computação, pode também significar "atração" no contexto da TV. Neste trabalho um "programa de TVDI" é definido como uma aplicação multimídia através da qual um telespectador pode interagir via controle remoto [9] com o programa de TV. Isso significa que o uso desse termo envolve tanto a difusão do conteúdo de áudio/vídeo quanto dos componentes de softwares que possibilitam a interação do telespectador. O termo "aplicação de TVDI" quando usado neste trabalho se refere apenas aos componentes de software que podem fazer parte de um "programa de TVDI".

3.2.2.1 Processamento de Dados no Receptor de TV

Uma das funções do middleware Ginga é dar suporte ao desenvolvimento de aplicações para a plataforma de TV digital associada ao SBTVD. De forma semelhante à abordagem adotada por outros sistemas de TV digital, o Ginga tem uma estrutura que dá suporte a execução de aplicações declarativas escritas em *Nested Context Language* (NCL) [83] e outra que dá suporte a aplicações imperativas Java [84]. O Ginga integrou as duas soluções, chamadas de Ginga-J e Ginga-NCL, tomando por base as recomendações internacionais da ITU [85]. Desta forma, o Ginga é subdividido em dois subsistemas interligados, também chamados de Máquina de Execução (Ginga-J) e Máquina de Apresentação (Ginga-NCL) conforme ilustrado na Figura 3.8. O conteúdo imperativo é executado por uma Máquina Virtual Java (JVM).

Outro aspecto importante é que os dois subsistemas do Ginga não são necessariamente independentes, uma vez que a recomendação do ITU inclui uma ponte, que deve disponibilizar mecanismos para intercomunicação entre os mesmos, de um modo que as aplicações imperativas utilizem serviços disponíveis nas aplicações declarativas, e vice-versa. Portanto, é possível a execução de aplicações híbridas em um nível acima da camada dos ambientes de execução e

Figura 3.8. Visão geral do middleware Ginga

apresentação, permitindo agregar as facilidades de apresentação e sincronização de elementos multimídias da linguagem NCL com o poder da linguagem orientada a objetos Java. O tipo de funcionalidade a ser adicionada a um programa de TVDI influencia diretamente o tipo de linguagem (paradigma) a ser utilizado na implementação. Assim como em outros domínios, (como na Web) as aplicações para TV digital são, normalmente, construídas usando abordagens declarativas, imperativas (algumas vezes também chamadas de procedurais) ou ainda, com o uso de uma abordagem híbrida, integrando os dois tipos de linguagens.

O Núcleo Comum Ginga (*Ginga Common Core*) é o subsistema do Ginga responsável por oferecer funcionalidades específicas de TV Digital comuns para os ambientes imperativo e declarativo, abstraindo as características específicas de plataforma e hardware para as outras camadas acima. Como suas principais funções, podemos citar: a exibição e controle de mídias; o controle de recursos do sistema (o canal de retorno, dispositivos de armazenamento); acesso a informações de serviço; sintonização de canais, entre outros.

A partir do momento em que os receptores de TV foram dotados de poder de processamento e que dados e objetos puderam ser transmitidos junto ao conteúdo audiovisual dos programas, criou-se um novo mercado para os desenvolvedores de software. Esses softwares permitem agregar uma série de funcionalidades e serviços à cadeia de produção de conteúdo para a plataforma de TV digital. Os programas de TVDI representam uma categoria de aplicações multimídia e também estão inseridos na nova cadeia de produção de conteúdo digital (ver seção 2.1).

A oferta de serviços e funcionalidades em programas de TVDI é dependente da infraestrutura disponível ao telespectador. Esta estrutura é formada pelo decodificador, pela geradora de conteúdo e pelo provedor de serviços (no caso de dependência de um meio de comunicação, ou canal de interatividade, para que os telespectadores acessem os serviços interativos). Os novos serviços incluem os tradicionais de previsão do tempo ou cotação de ações da bolsa, além de acesso a redes sociais, navegação Web, entre outros.

3.2.2.2 Transmissão de Dados para o Receptor de TV

A transmissão de dados e objetos gerados pelas emissoras e processados nos receptores de TV é feita de acordo com um mecanismo denominado carrossel de dados e/ou de objetos. Este carrossel é o mecanismo responsável por enviar de forma cíclica programas e dados multiplexados ao conteúdo audiovisual dos programas no fluxo de transporte MPEG-2. O carrossel funciona como um disco virtual que armazena dados e aplicativos que podem ser disponibilizados aos telespectadores.

Todos os sistemas de TV digital terrestre utilizam carrossel definido a partir do padrão DSM-CC (*Digital Storage Media - Command and Control*) [86]. Através do carrossel de objetos torna-se possível remontar no receptor a estrutura de diretórios da aplicação e seus dados da maneira em que se encontravam no gerador de conteúdo. Com o envio cíclico de conteúdo por meio do carrossel é possível que programas e dados sejam transmitidos corretamente para o receptor mesmo se o canal for sintonizado após o início da transmissão do programa interativo.

Para programas que utilizam dados e aplicações provenientes do canal de interatividade (por exemplo, acessar uma aplicação de correio eletrônico pela TV) ou dados e aplicações locais (por exemplo, a apresentação de fotos armazenadas em um pendrive conectado à porta USB do receptor), não é necessário o acesso à estrutura do carrossel. Assim, a depender da sua natureza, os dados e objetos podem ser disponibilizados de 3 formas para um receptor: (i) enviados pela própria emissora de TV, em *broadcast*, junto ao conteúdo audiovisual; (ii) acessados por meio do canal de interatividade ou (iii) acessados, de forma alternativa, a partir de um repositório.

3.2.2.3 Classificação de Programas de TVDI

De acordo com [5] e [87], os tipos de programas para TVDI podem ser categorizados em:

i) Programas que usam componentes de software cujos dados, consumidos ou produzidos (pelos componentes), não têm relação com a semântica do conteúdo de áudio e vídeo apresentados. Por exemplo, ler e-mails (TV-mail) durante a exibição de um filme ou acessar a conta bancária (TV-banking) durante a exibição de um jogo de futebol;

ii) Programas que usam componentes de software cujos dados, consumidos ou produzidos, têm relação com a semântica do conteúdo de áudio e vídeo apresentados, mas sem restrições fortes de sincronização. Por exemplo, visualizar cotação das ações da bolsa durante um programa de economia e

iii) Programas que usam componentes de software cujos dados, consumidos ou produzidos, têm relação com a semântica do conteúdo de áudio e vídeo apresentados e, são exibidos de forma sincronizada com este conteúdo. Por exemplo, a exibição de anúncios interativos (propaganda interativa) pode ser feita durante a transmissão de um filme no exato instante em que a camêra enquadra o produto anunciado (ponto de sincronização temporal). Esta última categoria pode ainda ser subdividida em:

   a. Programas cujo conteúdo de áudio e vídeo é conhecido a priori (programas gravados como filmes, novelas etc.);

   b. Programas cujo conteúdo de áudio e vídeo é gerado ao vivo (jogos de futebol, corridas de automóveis etc.).

O processo de construção de aplicações de TVDI para a primeira categoria de programas não se diferencia dos processos tradicionais. Assim, a única diferença de uma aplicação de TVDI para a mesma aplicação no contexto web está no tratamento dado aos requisitos não-funcionais, tais como: “plataforma de execução” (TV), “mecanismos de entrada e saída” (controle remoto) etc.

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

i) As aplicações são, apenas, uma parte de um programa de TV e esses, por sua vez, têm formato e contexto próprios.

ii) As aplicações utilizam como interface preferencial o receptor de TV que é de uso tradicionalmente coletivo.

iii) As aplicações exigem infraestrutura de transmissão e componentes de software/hardware adequados para o seu funcionamento.

iv) As aplicações podem modificar 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.

Assim, o processo de concepção destas apliacações deve considerar as categorias de programas existentes e respeitar as suas respectivas particularidades. Para esta tese, isso significa executar um conjunto de ações com o objetivo de responder a três questões importantes:

i) Como podem ser estruturados os conteúdos nessas aplicações?

ii) Como pode ser definida uma dinâmica de comunicação (transmissão, recepção, armazenamento temporário etc.) para os programas de TVDI e os possíveis geradores de conteúdo?

iii) Como podem ser modelados e construídos os programas voltados ao uso de componentes de software que permitam o seu reaproveitamento em contextos diferentes da TV?

Devem fazer parte de qualquer processo de desenvolvimento de aplicações multimídia: Processo, Arquitetura e Visão [23].

A dimensão de processo distingue diferentes fases de desenvolvimento (ou atividades), tais como especificação de requisitos, análise, projeto, implementação e testes. As fases de modelagem e implementação geralmente se correlacionam com diferentes níveis de abstração. A dimensão de arquitetura inclui os diferentes componentes de um sistema como, por exemplo, interfaces de usuário, sistemas externos, controle de processos etc. A dimensão de visão pode ser distinguida, por exemplo, em estrutura de dados, funções, comportamento dinâmico, entre outras onde cada visão capta aspectos específicos de um sistema.

Ao longo dos anos, muitos trabalhos foram focados em tecnologias voltadas para o desenvolvimento de programas para TVDI. Por outro lado poucos trabalhos trataram o processo de

desenvolvimento desse tipo específico de aplicação multimídia e não usam as 3 dimensões citadas. As próximas seções apresentam uma revisão dos principais trabalhos relacionados ao desenvolvimento de programas para TVDI.

3.3 TRABALHOS RELACIONADOS

O objetivo desta seção é apresentar os principais trabalhos relacionados ao tema desta tese. Alguns desses trabalhos têm foco na construção das aplicações a partir da sua estruturação visual (comportamento no tempo e no espaço). Isso pode ser observado, por exemplo, em dois casos: (i) nos trabalhos relacionados com a produção de conteúdo declarativo para TVDI e (ii), naqueles associados com as as ferramentas de apoio a desenvolvimento de aplicações.

Outros trabalhos, esboçam o uso de arquiteturas que permitem a integração sistêmica de plataformas. Porém, é importante ressaltar que essa integração não está ligada, necessariamente, ao reuso de componentes de software (em contextos diferentes) para concepção das aplicações. Isso pode ser observado (i) no suporte ao uso de linguagens imperativas e declarativas para concepção de aplicações, (ii) no suporte a múltiplos dispositivos de exibição e (iii) na arquitetura de criação de EPGs para múltiplos contextos.

3.3.1 Ferramentas e Tecnologias para Aplicações Multimídia

O universo de ferramentas e tecnologias, que podem apoiar o desenvolvimento de aplicações multimídia, envolve três categorias: (i) arcabouços e APIs, (ii) linguagens e (iii) ferramentas de autoria [23].

3.3.1.1 Arcabouços e APIs

A primeira categoria agrupa os arcabouços e APIs usados no desenvolvimento de aplicações multimídia. Alguns exemplos de arcabouços e APIs frequentemente encontrados na literatura [88] incluem a Java Media API [25], o Piccolo [89], o Windows Media Technology [90], o DirectX [91], a Digital Media Library [92] e, ainda, o IBM Lotus Sametime [93] [94].

A Java Media API (JMA) é um conjunto de soluções unificadas (APIs e arcabouços), não

proprietárias e de plataforma neutra para adicionar recursos multimídia em aplicações feitas na linguagem Java. Com a JMA é possível construir e incluir imagens 2D/3D em uma aplicação (Java 2D e Java 3D APIs), capturar som e voz a partir de dispositivos de I/O (Java Media Framework - JMF), apoiar a integração (sinconização) de áudio e vídeo (Java Advanced Imaging API - JAI), dentre outras possibilidades.

Em termos de programação, as possibilidades de uso da JMA são bastante heterogêneas e dependem do tipo de mídia a ser usada. O JMF, por exemplo, interpreta os objetos de mídia como fontes abstratas de dados através do uso de padrões de projeto de software. A estrutura da JMA e os exemplos disponíveis visam principalmente o desenvolvimento de aplicações para criar e editar objetos de mídia (primeira etapa na nova cadeia produtiva de conteúdo digital - ver seção 2.1) em vez da criação de aplicações com interfaces multimídia. Esse foco na criação de objetos de mídia é evidenciado em uma limitação importante para essa API: a falta de suporte para animação 2D. Esse tipo de animação é importante para interfaces multimídia de usuário e é um elemento central em ferramentas de criação profissionais como Flash [67].

Além da JMA, existem outros arcabouços Java, que se centram principalmente em um tipo específico de mídia. No contexto de animações 2D existe o arcabouço Piccolo, desenvolvido pela *Human-Computer Interaction Lab* da Universidade de Maryland [89]. O Piccolo é o sucessor do arcabouço Jazz, fornecido pelo mesmo grupo, e centra-se (em particular) sobre interfaces gráficas de usuário aproximáveis (*zoomable*). Além da versão Java, existe também outra versão para a linguagem C# disponível.

O Piccolo estrutura os elementos da interface de usuário como uma hierarquia de nós de mídia (grafo de cena). Os tipos de nós usados no Piccolo são: gráfico (classe PPath), imagem (classe PImage), texto (classe PText), câmera (classe PCamera) e contêiner (classe PCanvas). Um nó do tipo PCamera representa uma janela de exibição para os outros nós e pode aplicar transformações sobre eles. Um nó do tipo PCanvas representa uma tela, onde Widgets convencionais da API Swing Java podem ser colocados. O Piccolo também permite criar tipos de nós personalizados através da herança de classes. Logo, o arcabouço suporta gráficos e animações para implementação de interfaces de usuário em um alto nível de abstração.

De forma semelhante à JMA, o Windows Media Technology é um conjunto de componentes que podem ser usados para construir aplicações multimídia no Windows. São alguns de seus recursos: (i) o componente Player, que fornece uma interface de programação para a renderiza-

ção de alguns formatos multimídia; (ii) o componente Format (formatador), que pode ser usado para ler, escrever e editar arquivos no padrão do Windows (Windows Media Format), tanto para reprodução quanto para criação de conteúdos multimídia, (iii) o componente Encoder, que suporta entrega de dados via rede (network streaming), juntamente com a automação do processo de codificação local e remoto e (iv) o Component Services, que é usado para configurar, gerenciar e administrar servidores Windows Media.

Outro exemplo encontrado na literatura é o DirectX, que é uma coleção de APIs para construção de aplicações multimídia para o Windows e para o console de jogos XBox. As APIs do DirectX fornecem suporte para todos os tipos de mídia, incluindo a criação e renderização de gráficos 2D e 3D e animação (DirectDraw, Direct3D, DirectAnimation), reprodução e gravação de áudio (DirectSound, DirectMusic), reprodução e processamento de vídeo e outras mídias (DirectShow).

O DirectX pode ser classificado como um software do sistema, pois as suas funções básicas são de nível bastante baixo. Através do DirectX o programador pode abstrair e acessar as funcionalidade dos dispositivos de hardware existentes através de interfaces padrões. Por outro lado, ele também fornece um conjunto de funcionalidades multimídia de alto nível e é muitas vezes usado diretamente pelos desenvolvedores.

As APIs do DirectX fornecem objetos e interfaces no estilo do Microsoft Component Object Model (COM) [95], que podem ser usadas com diferentes linguagens de programação como C++, Visual Basic ou C#. Isso permite que programadores multimídia desenvolvam seus próprios componentes COM e os reutilizem em qualquer aplicação desenvolvida para a plataforma Windows. Dentre os exemplos mais comuns de uso do DirectX estão o desenvolvimento de jogos [96] [97] e outras aplicações sensíveis (dependentes) ao desempenho. O Microsoft NetMeeting, por exemplo, usa o DirectX e suporta conferências de áudio e vídeo, compartilhamento de aplicações Windows e quadros compartilhados.

A principal desvantagem do DirectX (e também do Windows Media Technology) é que os componentes resultantes dessas APIs podem ser executados apenas no Windows, o que reduz drasticamente o seu (re)uso em outras plataformas.

A Digital Media Library (DML) da Silicon Graphics engloba um conjunto de APIs que permitem uma descrição do formato de dados em aplicações multimídia, I/O de áudio e vídeo ao vivo e algumas facilidades para trabalhar com arquivos multimídia. A Silicon Graphics tam-

bém fornece ferramentas desktop de mídia, para o usuário final, que permitem capturar, editar, gravar, reproduzir, comprimir e converter áudio e imagens. Usar a DML implica, obrigatoriamente, em usar programação estruturada a partir da linguagem C. Isso, do ponto de vista da reutilização de código e modularização, pode ser visto como uma desvantagem.

Por fim, o IBM Lotus Sametime engloba uma família de APIs usadas para desenvolver aplicações multimídia colaborativas personalizadas, baseadas na web, que fornecem serviços em tempo real como: reunião com elementos de percepção, compartilhamento de telas e videoconferência. Além disso, o Sametime inclui também ferramentas de desenvolvimento (toolkits) para escrever e incluir aplicações Java no servidor Sametime. Embora esses toolkits cubram outras plataformas de desenvolvimento de software como COM e C++, as APIs do Sametime não estão preparadas para construção de componentes autônomos reutilizáveis.

3.3.1.2 Linguagens

A segunda categoria de tecnologias relacionadas ao desenvolvimento de aplicações multimídia, envolve as linguagens que visam apoiar especificação dessas aplicações. Essas linguagens são aderentes a definição de que as aplicações multimídia podem ser vistas como tipos de documentos (ver seção 2.2). As linguagens declarativas permitem a estruturação (espaço-temporal) das mídias e, algumas vezes, apoiam a criação dos mecanismos de interatividade e da lógica de aplicação. Porém, muitas vezes tanto a lógica da aplicação quanto os mecanismos de interatividade são adicionados a partir de um código de programação externo escrito em uma linguagem de script.

Alguns exemplos de linguagens declarativas voltadas ao desenvolvimento de aplicações multimídia interativas serão vistos a seguir.

A Synchronized Multimedia Integration Language (SMIL) é uma linguagem declarativa importante para o desenvolvimento de aplicações multimídia. A SMIL é um padrão baseado em XML, definido pelo World Wide Web Consortium (W3C) [98], que suporta a integração e a sincronização de diferentes objetos multimídia em uma aplicação, de forma declarativa.

O conceito básico de integração dos objetos de mídia reúne aspectos espaço-temporais. Para o aspecto espacial, cada documento SMIL especifica várias regiões na tela de apresentação. Assim, os diferentes objetos de mídia podem ser atribuídos a essas regiões. Para o aspecto

temporal, é possível definir, dentre outros, a apresentação sequencial ou paralela de diferentes objetos de mídia. As apresentações feitas em SMIL podem ser executadas em navegadores com plugins de mídia como o RealPlayer ou Quicktime Player.

Em SMIL é possível definir o tratamento de eventos do usuário, como cliques do mouse para iniciar e parar uma apresentação. Para o tratamento de eventos mais complexos, os documentos feitos em SMIL podem ser manipulados por scripts escritos em linguagens de programação, como em Javascript. No entanto, esse tratamento não é o objetivo da linguagem, pois não existe apoio para a integração da linguagem de script em apresentações SMIL.

Assim, a linguagem suporta o desenvolvimento de apresentações multimídia, mas dificilmente suportaria as aplicações multimídia interativas como aquelas consideradas nesta tese.

MHEG-5 [38] é nome dado ao padrão ISO/IEC 13522. Ele especifica uma linguagem usada para o desenvolvimento de aplicações interativas para TV Digital (que serão detalhadas na seção 3.2.1) em países como Austrália, África do Sul, Inglaterra, entre outros. O padrão suporta a integração de áudio, vídeo, gráficos, bitmaps e texto aos programas de TV digital, além de oferecer suporte aos objetos interativos para a interface de usuário, como botões, sliders, caixas de texto, entre outros. O MHEG-5 suporta diferentes tipos de eventos apesar de não definir uma especificação detalhada de comportamento para os eventos suportados. Assim, a depender do formatador que executa aplicações MHEG-5, o comportamento desses eventos pode variar [99].

A Nested Context Language (NCL) é uma linguagem baseada em XML também voltada ao desenvolvimento de aplicações interativas de TV Digital [9]. Baseada no Nested Context Model (NCM) [100] a linguagem NCL visa não apenas o suporte declarativo à interação do usuário, mas ao sincronismo espacial e temporal, em sua forma mais geral, tratando a interação do usuário como um caso particular. A NCL oferece também o suporte declarativo a adaptações de conteúdo e de formas de apresentação de conteúdo, o suporte declarativo a múltiplos dispositivos de exibição [19] interligados através de redes residenciais ou mesmo em área mais abrangente e a edição/produção de aplicações ao vivo [101].

Para o tratamento de casos mais complexos como, por exemplo, quando a geração dinâmica de conteúdo é necessária, NCL provê o suporte da linguagem de script Lua [102] [103]. Lua é uma linguagem de programação poderosa, rápida e leve, projetada para estender aplicacões em NCL. Ela combina sintaxe simples para programação procedural com construções para descri-

ção de dados baseadas em tabelas associativas e semântica extensível. Lua é tipada dinamicamente, é interpretada a partir de bytecodes para uma máquina virtual baseada em registradores e tem gerenciamento automático de memória com coleta de lixo incremental.

3.3.1.3 Ferramentas de Autoria

A terceira categoria de tecnologias que podem apoiar o desenvolvimento de aplicações multimídia reúne as ferramentas de autoria multimídia. Elas são importantes na prática, uma vez que o desenvolvimento de aplicações multimídia envolve também tarefas criativas e visuais. Além disso, desenvolvedores como designers multimídia e designers de interface de usuário muitas vezes não estão familiarizados com linguagens declarativas ou com linguagens de programação. Dessa forma, o uso das ferramentas de autoria deve dar suporte tanto ao processo de criação quanto as tarefas de implementação importantes na prática do desenvolvimento de aplicações multimídia [104] [105] [61] [23] [36] [106] [107]. Porém, a maioria das ferramentas encontradas na literatura se concentram apenas na codificação das aplicações e não suportam a dimensão do processo de criação.

As ferramentas de autoria podem ser inicialmente dividas em dois grupos: (i) aquelas que permitem a inclusão de código de programação na aplicação (usado para definir a lógica da aplicação) ou (ii) aquelas que apenas encapsulam uma linguagem declarativa, ou linguagem de programação, como em [108] e [109]. Nesse caso, o mesmo resultado pode ser obtido tanto editando uma linguagem diretamente quanto usando a respectiva ferramenta de autoria.

Um importante desafio na concepção de uma ferramenta de autoria visual é encontrar uma representação adequada para o comportamento temporal da aplicação. As ferramentas de autoria existentes são frequentemente classificadas de acordo com o paradigma de autoria (ou metáfora autoria) usado para resolver este desafio. Uma classificação utilizada por [104], [36] e [23] faz uma distinção entre ferramentas de autoria baseadas em quadros (frame-based), ferramentas de autoria baseadas em grafos (graph-based) e ferramentas de autoria baseadas em linhas de tempo (timeline-based). As próximas seções apresentam estas três categorias.

3.3.1.4 Ferramentas de Autoria Baseadas em Quadros

As ferramentas de autoria baseadas em quadros representam uma aplicação multimídia a partir dos diferentes quadros, telas ou cenas que contêm o conteúdo da interface do usuário. Quando um aplicativo é executado, as diferentes telas são, por padrão, apresentadas em ordem sequencial. Normalmente, é possível controlar a ordem das telas através do uso de scripts ou de links. Todas as classes de ferramentas de autoria são, no mínimo, baseada em quadros. A diferença é que as outras classes usam, além disto, uma metáfora visual adicional (ex: fluxograma ou linha do tempo) para especificar a ordem e a duração de um quadro.

Nesse tipo de sistema, o conteúdo de cada cena é geralmente editado visualmente, arrastando e soltando elementos de interface de usuário (fornecidos pela própria ferramenta). As propriedades de cada um desses elementos são exibidas e podem ser editadas em uma janela de propriedades. Além disso, é possível adicionar código de script para os elementos da interface (do usuário), para tratar (manipular) dados de entrada (do usuário).

Um exemplo clássico (que se enquadra nesta classe de ferramentas) é a Apple HyperCard [110]. Uma aplicação feita a partir da HyperCard é representada como uma pilha de cartões onde cada um deles corresponde a um tela exibida ao usuário. A ferramenta suporta elementos de interface de usuário como imagens, botões e caixas de texto. Os botões e caixas de texto podem ser manipulados com scripts escritos na linguagem orientada a objetos HyperTalk [111]. Esses scripts podem tratar eventos (associados a interface de usuário), como clicar em um botão ou abrir um novo cartão da pilha, ou funções associadas com outros elementos. O código de script pode controlar a navegação da aplicação multimídia, bem como ler e escrever as propriedades dos elementos de interface de usuário.

3.3.1.5 Ferramentas de Autoria Baseadas em Grafos

As ferramentas de autoria baseadas em grafo fornecem as mesmas funcionalidades básicas daquelas baseadas em quadros, além de permitir a representação visual do comportamento temporal de uma aplicação multimídia em termos de um grafo. O exemplo mais popula para essa classe de ferramentas é a Authorware [112] da Adobe, que utiliza fluxogramas [105] [107]. A maioria dos autores chama essa classe, mais especificamente, de ferramentas baseadas em fluxogramas. Para o suporte da lógica mais complexa de uma aplicação multimídia interativa, a

Authorware inclui uma linguagem de script, a Authorware Scripting Language (AWS) [104], e também suporta Javascript.

Outros exemplos de ferramentas baseadas em grafos são a Firefly [113] e a Eventor [114]. A Firefly permite estabelecer uma estruturação de documentos multimídia baseada em relações temporais, que podem ser especificadas pelo desenvolvedor como grafos dirigidos. A Eventor combina autoria baseada em quadros e linhas de tempo.

3.3.1.6 Ferramentas de Autoria Baseadas em Linhas de Tempo

As ferramentas de autoria baseadas em linhas de tempo representam visualmente a dimensão temporal de uma aplicação multimídia através da metáfora de linha do tempo. Normalmente, essa metáfora consiste em várias faixas, sendo que cada uma delas é associada a algum conteúdo da aplicação, como um objeto de mídia. A linha do tempo permite visualizar os períodos em que uma determinada faixa torna-se ativa e, por exemplo, um objeto de mídia é exibido ou reproduzido.

Atualmente, muitas das ferramentas de autoria profissionais mais populares são baseadas em linha do tempo. Enquadram-se nessa classe a Director [115] e a Flash [67], ambas produzidas pela Adobe [107].

A Director usa janelas que representam metáforas visuais da área de produção de um vídeo (filme, seriados, novelas etc.). Essas janelas exibem elementos arbitrários de mídia, chamados de membros do elenco (Cast Members), que podem ser utilizados em uma aplicação multimídia. Cada instância de um membro do elenco é chamada de Sprite e são estruturados na interface de usuário de uma aplicação a partir de uma área denominada de Stage. O controle temporal de uma aplicação é feito por uma linha do tempo que na Director recebe o nome de Score. A Score é dividida em trilhas, denominadas de canais (Channels), que definem quando e por quanto tempo cada Sprite vai aparecer, e em quadros (Frames) que representam um ponto no tempo. Assim, executar uma aplicação na Director significa exibir uma sequência de quadros organizados em uma determinada ordem.

Na Director, a lógica mais complexa de uma aplicação multimídia interativa é implementada a partir de código script escrito na linguagem Lingo [115], que é uma linguagem de programação baseada em objetos e pode ser usada para controlar a aplicação, o seu conteúdo e suas

propriedades, bem como para adicionar interação. Além disso, os scripts podem ser adicionados apenas como manipuladores de eventos. As aplicações feitas na Director são compiladas para o formato Shockwave Flash [67] que é interpretado por um player disponível como plugin para maioria dos navegadores web.

A ferramenta de autoria Flash fornece conceitos muito semelhantes à Director. Ao longo dos anos, a ferramenta tornou-se uma plataforma muito popular, tanto que o termo "Flash" é muitas vezes utilizado para designar a tecnologia como um todo ou o formato dos aplicativos resultantes. Flash é o nome da ferramenta de autoria que utiliza um formato de arquivo proprietário com a extensão de arquivo FLA. Para a execução dos arquivos FLA é preciso convertê-los em arquivos Shockwave Flash.

Em comparação com a Director, Flash fornece um melhor suporte para a criação de gráficos vetoriais 2D e animações. Neste sentido um conceito importante é representação de linhas de tempo com conteúdos próprios denominada de MovieClip. Os quadros internos de um MovieClip podem conter qualquer tipo de conteúdo (gráficos, outras animações, áudio, vídeo etc.) tal como linha de tempo principal da aplicação. É possível também estruturar hierarquicamente múltiplos MovieClips em qualquer profundidade. Uma vez que um MovieClip foi criado pelo desenvolvedor, ele pode ser instanciado diversas vezes e assim é possível criar animações com qualquer grau de complexidade.

O controle de toda aplicação e do seu conteúdo pode ser feito em Flash a partir de scripts feitos na linguagem ActionScript. A primeira versão da ActionScript [116] foi baseada em objetos semelhantes escritos na linguagem de script ECMAScript [117]. As versões seguintes permitiram a inclusão de mecanismos da orientação a objetos como, por exemplo, definir arquivos de classe (similar aos arquivos de classe em Java) e associá-los a um MovieClip. Isso fez com que cada instância de MovieClip estivesse associada a um objeto ActionScript correspondente. Assim, é possível desenvolver um aplicativo Flash completo somente por programação ActionScript e sem uso das características visuais da ferramenta autoria.

3.3.1.7 Autoria Baseada na Interação do Usuário Final

Além da autoria feita por especialistas (a partir do uso de ferramentas), existe também outra categoria de aplicações multimídia cujos autores são os próprios usuários. Os avanços nas

tecnologias de comunicação e nos dispositivos de captura multimídia combinados com os incentivos para distribuição de conteúdo em redes sociais, têm habilitado os usuários finais a tornarem-se parte ativa da cadeia de produção multimídia. Porém, tanto as tecnologias e ferramentas citadas quanto os processos de autoria multimídia são voltados para uso exclusivo de profissionais. Buscando suprir esta lacuna alguns trabalhos cujo foco é usar a interação do usuário como base da atividade de autoria multimídia têm sido propostos. Entre eles podem-se destacar [118], [119], [120] e [121].

O paradigma de autoria *Watch-and-Comment* (WaC) apresentado em [118] permite que um usuário possa comentar trecho de um vídeo enquanto o assiste, adicionando anotações em texto, marcas de tinta, comentários em áudio, seleção de trechos, marcas de referência a momentos etc. As informações fornecidas podem ser usadas para a geração de vídeos interativos automaticamente (o vídeo original anotado com os comentários capturados).

O trabalho apresentado por [119] especializa o paradigma WaC na discriminação coletiva de momentos e segmentos de mídias contínuas para uso em diversas aplicações. O termo "discriminação", no contexto do trabalho, significa "reconhecimento da diferença entre uma coisa e outra". O objetivo desse trabalho foi prover mecanismos para permitir a edição de documentos multimídia por múltiplos usuários, através de dispositivos presentes em redes locais, de forma transparente, utilizando o protocolo de descoberta automática de serviços *Universal Plug and Play* (UPnP). Para isso, o trabalho definiu um conjunto de operadores para permitir a discriminação e mostrou como esses operadores podem ser utilizados por aplicações de processamento de anotações geradas coletivamente e em instâncias distintas de mídia.

Uma possível aplicação das discriminações realizadas pelos usuários é a seleção de trechos [119]. Em uma partida de futebol, por exemplo, cada telespectador poderia realizar marcações nos lances que mais gostou, usando dispositivos próprios (smarthphones, controles de consoles de jogos etc.), e ao final da partida seria gerado de forma colaborativa, um documento interativo multimídia (NCL, SMIL, ou outra linguagem) reunindo todas as seleções, mostrando os melhores lances da partida de acordo com as opiniões coletadas. No contexto de discriminação de segmentos de mídia, cada dispositivo é responsável pela transferência ao receptor digital da marcação realizada pelo usuário, tal como um algarismo numérico acionado por um telespectador no controle da TV. Com isso, pode-se registrar o momento em que foi realizada a marcação, quem a realizou e que discriminação foi feita.

O trabalho [120] apresenta um uso particular do paradigma WaC, e explora a possibilidade de usar um controle remoto ou ou outro dispositivo móvel do próprio usuário para interagir com o vídeo. A interação é feita por um sistema que permite aos usuários personalizar seus vídeos através do uso de mídia *stickers* (“Adesivos” de mídia criados previamente). Mídia *stickers* são porções de conteúdo que podem ser anexados a vídeos (atrelados ou não ao tempo). Eles podem ter uma duração intrínseca (baseada em intervalo), como animações e hinos ou ter duração de tempo intrínseco igual a zero (discretos), como gráficos e fotos.

No trabalho apresentado por [121] os autores defendem a necessidade de um processo de produção multimídia que contemple a geração de conteúdo pela participação do usuário final. Para os autores um processo como este deve atender a alguns requisitos básicos: i) assistir os usuários finais na tarefa de enriquecer o conteúdo gerado por profissionais; ii) reconhecer os usuários finais como elementos ativos em todas as etapas dos fluxos de trabalho de produção totalmente orientado para o conteúdo gerado por eles; e iii) desenvolver de mecanismos para, de forma fácil, capturar e transferir metadados ao longo do processos de produção orientada ao usuário final.

3.3.2 Produção de Conteúdo Declarativo para TV Digital

O trabalho proposto em [9] discute a produção de programas não-lineares, para o sistema de TV digital brasileiro, com forte foco na linguagem NCL. O trabalho apresenta o paradigma declarativo como base para explicar a estruturação de programas feitos em NCL e apresenta uma metodologia para o desenvolvimento de um programa não-linear usando NCL. Alguns tipos de aplicações encontradas no ambiente de TV digital interativa são apresentados e os conceitos básicos sobre linguagens declarativas com o foco em sincronismo de mídias também são introduzidos. Todos os conceitos e funcionalidades da NCL são discutidos com riqueza de detalhes no trabalho. Porém, é importante ressaltar que a metodologia apresentada pelos autores não tem o mesmo significado de um processo formal de desenvolvimento de software. Na verdade, o que os autores chamam de metodologia é a apresentação de um exemplo de uso através de um programa interativo cujos módulos de software foram implementados em NCL. O trabalho não define fases, artefatos gerados e outras características comumente encontradas em um processo formal de especificação de sistemas. Essa limitação também é encontrada nos trabalhos citados a seguir.

O trabalho apresentado por [10] propõe novas funcionalidades para a criação de programas MPEG-4 [81] utilizando a linguagem NCL. Os autores destacam que a linguagem permite definir relações entre os objetos de mídia dentro de uma cena MPEG-4 e objetos de mídia externos, incluindo outros objetos em cenas MPEG-4. O trabalho apresenta também uma abordagem que permite definir relações com diversas semânticas diferentes e não os definidas pela norma do SBTVD. Essas novas funcionalidades introduzidas são suportadas através da integração do formatador NCL com players MPEG-4. O artigo também tira proveito conceito de template de nó de composição da NCL, para adicionar novas funcionalidades de criação ao especificar documentos MPEG-4 usando a linguagem XMT-O [122].

O trabalho de [11] apresenta um estudo empírico sobre a NCL cujo objetivo foi a obtenção de indicadores de usabilidade da linguagem na geração de conteúdo para TVDI. O estudo foi feito através de dados coletados em formulários preenchidos por 220 alunos de várias classes em ações de treinamento realizadas em alguns estados do Brasil. Em uma análise posterior, feita por uma combinação de métodos qualitativos e quantitativos, foram encontrados alguns aspectos onde o perfil NCL pode e deve ser melhorado para facilitar as atividades de criadores de conteúdo para TVDI. Esses criadores de conteúdo têm um perfil heterogêneo e não estão necessariamente familiarizados com programação de computadores. O trabalho indica a carência de métodos e processos na elaboração dos módulos interativos de um programa de TV.

Os artigos [15] e [16] têm o objetivo de apresentar algumas experiências desenvolvidas através do uso do framework AppTV na produção de conteúdo declarativo para TV Digital Interativa. Esse framework foi concebido para auxiliar a construção de módulos de software para essa nova plataforma de execução. Sua principal função é abstrair e encapsular parte dos conceitos da TV Digital que o desenvolvedor necessitaria assimilar para produzir um software considerando as particularidades do ambiente de TV. Além disso, possibilita que editores de TV criem os módulos interativos de seus programas através da edição de documentos XML, abstraindo a linguagem de programação usada. O framework usa uma linguagem declarativa, com um nível de abstração mais alto, com tags intuitivas e de simples manipulação.

3.3.3 Relacionando linguagens declarativas e imperativas

O trabalho apresentado em [18] enfoca o apoio fornecido pela linguagem NCL para relacionar os objetos com teor de código imperativo e objetos hipermídia declarativos (objetos com

conteúdo de código declarativo especificando documentos hipermídia). O trabalho destaca a NCL como linguagem declarativa do Sistema Brasileiro de TV Digital (SBTVD) e também como uma das recomendações ITU-T para serviços de IPTV.

Os autores destacam como principal contribuição deste trabalho a integraçao permitida pela NCL dos paradigmas de linguagem imperativo e declarativo. Essa integração ocorre de forma pouco invasiva, com a manutenção de um limite claro entre objetos incorporados independente do seu conteúdo, codificação e definição. Este modelo de comportamento evita os efeitos colaterais que ocorrem quando existe a necessidade de inegração entre plataformas. O suporte de uma arquitetura de software a múltiplos paradigmas é um requisito importante para o reuso de componentes de software em contextos diferentes. Este trabalho confirma que esse requisito encontra suporte tecnológico no ambiente de TVDI e isso abre espaço para que um modelo convergente de produção de aplicações possa ser concebido e utilizado.

3.3.4 Ginga-NCL: Suporte a Múltiplos Dispositivos

O trabalho apresentado em [19] descreve a arquitetura de suporte a múltiplos dispositivos para programas de TVDI. Esta arquitetura permite tirar proveito da existência de múltiplos dispositivos de exibição para aumentar o número de programas de TV interativos exibidos. A partir dessa arquitetura é possível criar novos programas em que telespectadores podem agir cooperativamente, além de poderem ter uma navegação individualizada sem incomodar seus vizinhos em um mesmo ambiente. Esta navegação individualizada cria a possibilidade de, pela primeira vez, alterar a forma coletiva de consumo tradicional dos programas de TV.

O trabalho ratifica o interesse da comunidade científica pela integração de sistemas no contexto da TV Digital. Ele descreve o suporte dado pela linguagem NCL ao desenvolvimento de aplicações para múltiplos dispositivos seguindo um modelo hierárquico suportado pelo ambiente de apresentação Ginga-NCL. Isso é realizado através de um conjunto de componentes voltados à orquestração dos dispositivos envolvidos em uma apresentação. Os autores apresentam o suporte oferecido pela implementação de referência do middleware Ginga-NCL, destacando sua arquitetura e ilustrando, com exemplos, as possibilidades abertas pelas aplicações multi-dispositivos. Apesar disso, essa arquitetura não é algo inovador no contexto da integração de sistemas. O suporte oferecido pelo middleware Ginga-NCL é semelhante ao suporte oferecido pela arquitetura de serviços Web. Assim, como o Ginga possui outros módulos que permitem

a camunicação com esse ambiente, a arquitetura de serviços Web também poderia ser utilizada como arquitetura de suporte a múltiplos dispositivos.

3.3.5 EPG para múltiplos contextos

O trabalho apresentado em [20] visa fornecer arquitetura para criar um *Electronic Program Guide* (EPG) e aplicações que funcionem como serviços sobre um EPG adaptáveis em tempo real. A arquitetura apresentada pode ser aplicada para gerar o código fonte em linguagens diferentes. No trabalho isso é feito através da separação entre o estilo, a estrutura e os aspectos de navegação do EPG. O trabalho apresenta um estudo de caso que segue a arquitetura proposta e serve como prova de conceito e que foi implementado usando o ambiente declarativo do Ginga. Neste trabalho, é importante ressaltar que, apesar da arquitetura proposta não usar uma metodologia formal para projeto e implementação de softwares, o suporte à geração de código fonte para linguagens diferentes é um aspecto do trabalho que demostra a necessidade do desenvolvimento de alternativas ao reuso de aplicações em contextos diferentes.

3.3.6 Ferramentas de Autoria para Aplicações de TVDI

Um grupo especial de ferramentas de autoria, importantes para esta tese, dão suporte aos desenvolvimento de aplicações multimídia interativas para TV digital. Entre as diversas iniciativas, pode-se destacar as ferramentas de autoria Composer [12] e NCLEclipse [13].

Composer [12] é uma ferramenta de autoria hipermídia usada na codifcação de programas audiovisuais interativos em NCL. Seu principal objetivo é possibilitar que usuários possam produzir código NCL através de abstrações visuais, o que requer menos conhecimento especializado de NCL. Para que isso seja possível, a filosofia utilizada pela Composer é a de que o usuário que irá construir o programa poderá fazê-lo através de visões que incluem as metáforas visuais de quadros, grafos e linhas de tempo além do acesso direto ao código NCL.

Na ferramenta, cada visão mostra uma parte do programa NCL e, caso uma alteração ocorra em uma das visões, as outras são atualizadas automaticamente. Existem quatro visões definidas: Estrutural (exibe os objetos de mídia 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 de TVDI. Não existe qualquer funcionalidade de apoio ao reaproveitamento dos módulos interativos gerados para outros programas ou ainda outros contextos (Web, móvel etc.). O uso da ferramenta parte do princípio que todos os requisitos funcionais e não funcionais da aplicação já estão especificados e não oferece módulos de apoio a esta especificação.

Uma descrição semelhante a Composer pode ser feita para o plugin NCLEclipse [13]. 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 a geração de código fonte na linha WYSIWYG (*What You See is What You Get*). O plugin também não oferece funcionalidades de apoio ao reaproveitamento de código ou mesmo a especificações de projeto mais abstratas. Entre as principais funcionalidades implementadas pelo NCL Eclipse estão: coloração de elementos e atributos XML, exibir/ocultar elementos XML (permitindo que determinados elementos sejam escondidos ou exibidos pelo desenvolvedor), *wizards* para a criação de documentos NCL simples, auto-formatação 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, entre outras.

Outro plug-in acoplável ao ambiente do Eclipse, denominado de SAGA, é apresentado por [14]. O SAGA é baseado em componentes gráficos - que podem ser dispostos espacialmente no ambiente de desenvolvimento visual do IDE - criados a partir de bibliotecas específicas para TVDI. O plugin compõe um ambiente de desenvolvimento e simulação para a linguagem Java e tem o propósito de auxiliar a codificação dos módulos interativos dos programas de TV Digital.

As ferramentas JAME Author [123] e iTV Suite Author (antiga *Cardinal Studio*) [124] são baseadas na linguagem Java. A JAME Autor tem a intenção de tornar fácil as tarefas de autoria e usa para isso uma abstração, composta por páginas, similar às páginas Web. Cada página é composta de objetos de mídia e é especificada usando uma linguagem baseada em XML chamada JAME PDL (*JAME Page Description Language*). A iTV Suite Author é uma ferramenta de criação que permite estruturar as aplicações em “atos” para desenvolvimento de conteúdo para TV Digital no middleware MHP. O paradigma e as funcionalidades da ferramenta são muito semelhantes a JAME Author. No entanto, ao contrário da JAME Author, a iTV Suite Author fornece suporte para a definição de relações espaciais e temporais entre os objetos de

mídia que compõem um ato, embora essa tarefa exija um esforço de codificação.

O GriNS [125] e o LimSee2 [126] são sistemas de autoria e apresentação para documentos SMIL. Assim como o Composer, elas também são voltadas para autores com pouco conhecimento da linguagem SMIL e também utilizam a abordagem de visões, embora de uma forma um pouco diferente da abordagem do Composer. O GRiNS possui basicamente quatro visões: Visão de Estrutura Lógica, Visão de Linha de Tempo Virtual, Visão de Playout e Visão Textual. A Textual View do GriNS é um simples editor de texto que aplica coloração sintática apenas aos elementos <par> e <seq> da SMIL, nenhuma sugestão de código contextual é implementada para facilitar a autoria textual. O foco da LimSee2 é a sincronização espaço-temporal que é editada através das visões espacial e temporal. A interatividade é obtida através da edição direta do código fonte. Ela oferece ainda componentes auxiliares para edição de atributos e visualização da estrutura do documento.

3.3.7 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 de especificação de sistemas [76] [127].

O desenvolvimento de aplicações multimídia envolve 3 diferentes tipos de projeto: (i) projeto de software, desenvolvimento da lógica do aplicativo; (ii) projeto da interface gráfica com o usuário e (iii) projeto de mídias, pois a criação de objetos de mídia requer normalmente o conhecimento e ferramentas e conhecimentos (estudo da cor, forma etc.) [76].

A linguagem MML [76] procurar endereçar 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 3 diferentes tipos de projeto anteriormente descritos. Para tanto, MML define 5 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, responsável pela integração dos componentes de mídia com a interface com o usuário abstrata e derivado de modelo de interface abstrata e objetos de mídia do modelo estrutural 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.

3.3.8 Produção de Programas de TVDI

Um processo de produção de uma aplicação específico para TVDI deve considerar tanto as particularidades do ambiente de TV (ver seção 3.2.2.3) 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. Esta seção apresenta os principais trabalhos voltados ao processo de produção de uma aplicação para TVDI.

O foco do trabalho apresentado por [21] é o projeto de interface de usuário e a avaliação de entretenimento mediado por computador via TV em um ambiente doméstico. Ele aborda três questões principais: i) modelo conceitual, ii) os princípios de interface de usuário e iii) a avaliação de usabilidade. O autor traz o conceito de Virtual Channel (canal virtual) que pode ser definido como uma extensão de um canal tradicional, com base em padrões espaciais e de personalização temporal.

O trabalho foi dividido em duas fases. A primeira teve como objetivo identificar os elementos comuns em projetos de interface de usuário em TVDI e a segunda etapa utilizou os elementos identificados na fase anterior para construir e avaliar uma aplicação para TVDI.

O modelo conceitual de [21] utilizou uma biblioteca de programação concebida e desenvolvida de acordo com o canal virtual. Esta biblioteca utiliza uma linguagem que os profissionais de comunicação estão mais habituados. Uma constatação importante é que o trabalho não faz a utilização de roteiros para a especificação de requisitos do software. Ao invés disso, faz uso de protótipos baseado nas informações dos *storyboards/cenários* para construção da interface de usuário.

Os princípios de interface de usuário são bastante genéricos ou direcionados para uma deter-

minada funcionalidade, e isto é muito útil quando se trata de um projeto de interface para TVDI. Todo o processo foi demostrado através da construção de um programa interativo musical.

Outro importante trabalho voltado ao processo de produção de aplicações para TVDI foi apresentado por [22]. O autor apresenta um modelo de processo para o desenvolvimento de programas interativos dividido em quatro fases: desenvolvimento, especificação, produção e teste, entrega e operação.

Na fase de desenvolvimento faz-se: i) um estudo de viabilidade para verificar se a ideia do programa apresenta algum interesse no mercado; ii) desenvolvimento da ideia, onde se procura melhorar a ideia inicial; iii) realiza-se um teste de conceito, respondendo perguntas como, “As pessoas vão utilizar?”, “É necessário?”, “É tecnicamente possível?”.

A fase de especificação é responsável por definir o foco do programa, especificar as ferramentas de gerenciamento, identificação dos stakeholders, especificação dos requisitos, plano de projeto, planejamento dos custos e contratação da equipe. A especificação dos requisitos é feita através de uma lista de requisitos que contém o que todos os envolvidos acham que a aplicação deve realizar. Esta lista é quebrada em diversas seções como: requisitos comerciais, requisitos técnicos, requisitos de usabilidade, entre outros.

Os papéis envolvidos no projeto são a equipe de produção, equipe de design, equipe técnica, equipe de conteúdo e operações, equipe de marketing e comercial. A primeira é responsável pela definição do programa, construção do *storyboard/cenário*, design do protótipo e entrega da aplicação final. A equipe de design é responsável pela parte gráfica do programa. A equipe técnica cuida da especificação técnicas e define qual tecnologia deve ser utilizada na construção da aplicação. A equipe de conteúdo e operações executa a aplicação após a entrega. Por fim a equipe de marketing e comercial promove a aplicação para os telespectadores.

Na fase de produção e teste ocorre a construção do design gráfico, utilizando os *storyboards/cenários*, e da arquitetura, a implementação da aplicação interativa, o controle de mudanças e de prazos e os testes. A última etapa é a de entrega e operação onde o programa interativo é entregue e lançado. Ainda nesta fase pode ocorrer algum tipo de evolução no programa.

O trabalhos propostos em [3], [4] e [5] abordam, especificamente, as etapas conceituais da modelagem de aplicações para TVDI. Eles apresentam modelos para o desenvolvimento de programas de TVDI, baseado em metodologias ágeis, que integram atividades inerentes ao processo de produção de TV e atividades do desenvolvimento de software. Esses modelos são

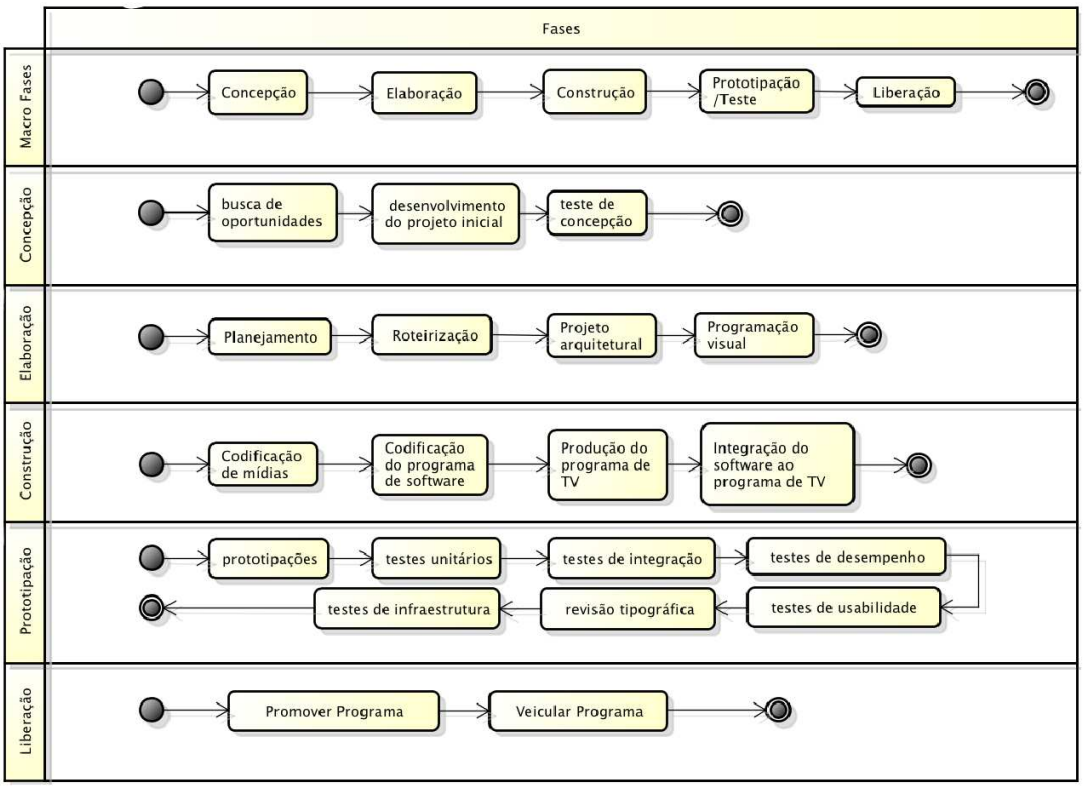

Figura 3.9. Fases do Processo de Desenvolvimento de Programas de TVDI [3], [4] e [5]

pautados em quatro aspectos: 1) Filosofia de trabalho, 2) Processo, 3) Papéis / 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 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) entre 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 3.9.

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 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/cenário* que representa o projeto inicial com a ideia central 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 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 [102], Java [128] e ECMAScript [117]. 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 [83]. São exemplos de linguagens declarativas para TVDI o NCL [129] e o XHTML [130].

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 infraestrutura.

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.

As abordagens descritas não são voltadas à estruturação de aplicações no ambiente de TV. Elas concentram-se em definir quais são as atividades gerais do processo de produção e não como essas atividades devem ser executadas. Os modelos apresentados no próximo capítulo permitem estruturar os conteúdos de um programa de TVDI e especificar suas aplicações. Para isso, devem ser utilizados nas fases de elaboração e construção do programa e permitem o reuso das aplicações especificadas em outros contextos diferentes da TV.

3.4 RESUMO

Este capítulo apresentou uma revisão detalhada dos principais tópicos relacionados ao desenvolvimento de programas de TV convencionais e discutiu a evolução para TV digital interativa. A elaboração de um programa de TV é um conjunto de tarefas que envolve pessoas com competências diversas e o processo que permite organizar a execução dessas tarefas é algo bastante complexo. Do ponto de vista tecnológico, esse conjunto de tarefas se resume a produzir e sincronizar os fluxos de áudio e vídeo de um programa de TV. Apesar de toda a evolução vista no ambiente de TV, o modelo usado para produção desses programas mudou muito pouco.

Conforme visto na seção 3.2, a mudança da cadeia produtiva de conteúdos para serviços de comunicação, provocada pela digitalização, abriu a possibilidade de introduzir elementos interativos em um programa de TV, sob a forma de componentes de softwares, e deve mudar a maneira como esses programas são concebidos. A concepção de elementos de software que podem fazer parte de um programa de TV digital e ainda reaproveitá-los em outros contextos é foco desta tese.

O próximo capítulo, apresenta dois modelos voltados a especificação de aplicações para TVDI denominados IDTVS e StoryToCode.

Capítulo 4

Este capítulo apresenta os modelo denominados IDTVS e StoryToCode que permitem especificar os componentes interativos de um programa de TV digital. Esses modelos foram inspirados em metodologias tradicionais de desenvolvimento de softwares.

APLICAÇÕES INTERATIVAS DE TV DIGITAL

Com o surgimento da TV Digital, os processos produtivos de programas, que antes eram específicos do ambiente de TV, devem agregar uma variedade de recursos oriundos de outros contextos (computação, design etc.). Esse é exatamente o caso dos programas para TV Digital Interativa (TVDI). A introdução de elementos interativos (softwares) nesses programas passa a envolver tanto profissionais de produção de TV (diretores, produtores, editores etc.), quanto aqueles ligados à produção de software (programadores, analistas etc.). Além disso, migrar do modelo de concepção de um programa de TV convencional, que é pré-determinado e sequencial, para um modelo não linear e interativo não é uma atividade trivial. Assim como em outros exemplos de desenvolvimento de aplicações multimídia, o desafio nesse caso é aliar ao processo de modelagem de elementos interativos para o ambiente de TV as técnicas tradicionalmente usadas na especificação de softwares para solução de questões como: reuso, estruturação de requisitos, entre outras.

Nenhum dos trabalhos relacionados usa a estruturação do conteúdo interativo de programas de TVDI sob a forma de elementos (objetos) como um mecanismo que permita o reuso (destes elementos) em contextos diferentes de TVDI como Web e Mobile. Mesmo abordagens focadas nos processos de produção de programas de TV digital como [3], [4] e [5] não são voltadas à estruturação de elementos interativos no ambiente de TV. Elas concentram-se em definir apenas quais são as atividades gerais do processo de produção e não como essas atividades devem ser executadas. Buscando preencher as essas lacunas, esta tese apresenta como a sua principal

contribuição dois modelos que permitem a especificação de programas para TVDI a partir da estruturação do conteúdo. Este capítulo apresenta os dois modelo denominados IDTVS (*Interactive Digital TV Show*) e StoryToCode.

4.1 IDTVS - INTERACTIVE DIGITAL TV SHOW

O primeiro modelo resultante dos estudos ligados a esta tese permite estruturar elementos interativos de uma aplicação de TVDI. O modelo, denominado IDTVS, tem foco na estruturação dos dados extra de um programa de TV através de classes e objetos, tanto no lado do difusor, quanto no lado do receptor de TV. A ideia básica é associar todos os conteúdos relevantes ao programa de TV transmitido (ex.: momentos importantes em um jogo de futebol; ultrapassagens, pilotos e equipes numa corrida de F1 etc.) com objetos que representam os dados extras a serem enviados durante a transmissão. Além de abstrair o conteúdo a ser transmitido, cada objeto está associado a uma marca temporal que significa o instante de ocorrência do evento que ele representa. Isso pode permitir tanto o tratamento de eventos (ex.: highlights) relacionados à transmissão do programa, quanto o processamento de todas as outras informações relevantes para o programa, mas que não são dependentes de sua dinâmica (conforme seção 3.2.2.3).

4.1.1 O Modelo Formal IDTVS

Formalmente um programa interativo de TV digital pode ser definido por:

- IDTVS={MC, BEC, REC}, onde:
  - MC representa o conteúdo principal (*Main Content*) de um programa interativo;
  - BEC representa o conteúdo extra (síncrono e assíncrono) no lado do difusor (*Broadcast Extra Content*) e
  - REC representa o conteúdo extra (síncrono e assíncrono) no lado dos receptores de TV (*Receiver Extra Content*).

O conteúdo principal é definido como:

- MC={V,A}, onde:

– V=$\{v_1, v_2, v_3, ...\}$: é o conjunto de *streams* de vídeo ;

– A=$\{a_1, a_2, a_3, ...\}$: é o conjunto de *streams* de áudio.

Para facilitar o entendimento do BEC e do REC, algumas definições são necessárias:

- Seja C_i um conjunto de tipos de objetos IDTVS (classes). Um objeto IDTVS é um tipo estruturado que pode abstrair tanto o conteúdo principal, quanto o conteúdo extra.
- Seja O um objeto IDTVS (uma instância de um tipo IDTVS pertencente ao conjunto $\in C_i$).
- Seja App uma interface de serviços para aplicações interativas. A aplicação que implementa essa interface é um software residente que habilita o usuário a acessar diretamente os dados extras.

Assim, no difusor de conteúdos, um BEC_t é uma instância de BEC no instante t. Ela pode ser definida formalmente pelo par $BEC_t = (C_i, L_0)$, onde:

L_0: representa uma lista de objetos O cujo tipo $\in C_i$.

Nos receptores de TV, um REC_t é uma instância de REC no instante t. Ela pode ser definida formalmente pela tripla $REC_t = (C_i, L_A, L_0)$, onde:

L_A : representa uma lista de objetos que cumprem a interface App.

4.1.2 O Modelo Conceitual: Abordagem Orientada a Objetos

Conforme visto na seção 2.5, o uso de modelos visuais facilita o processo de desenvolvimento de uma aplicação multimídia. Por isso, é importante utilizar uma linguagem de modelagem visual para formalizar o modelo IDTVS, como um conjunto de dados estruturados que abstrai os conteúdos principal e extra de um programa de TVDI. Como o modelo representa a estrutura do conteúdo extra, a abordagem escolhida foi a “orientada a objetos” e utilizou-se o diagrama de classes da UML para representar o visualmente modelo, conforme ilustrado na Figura 4.1.

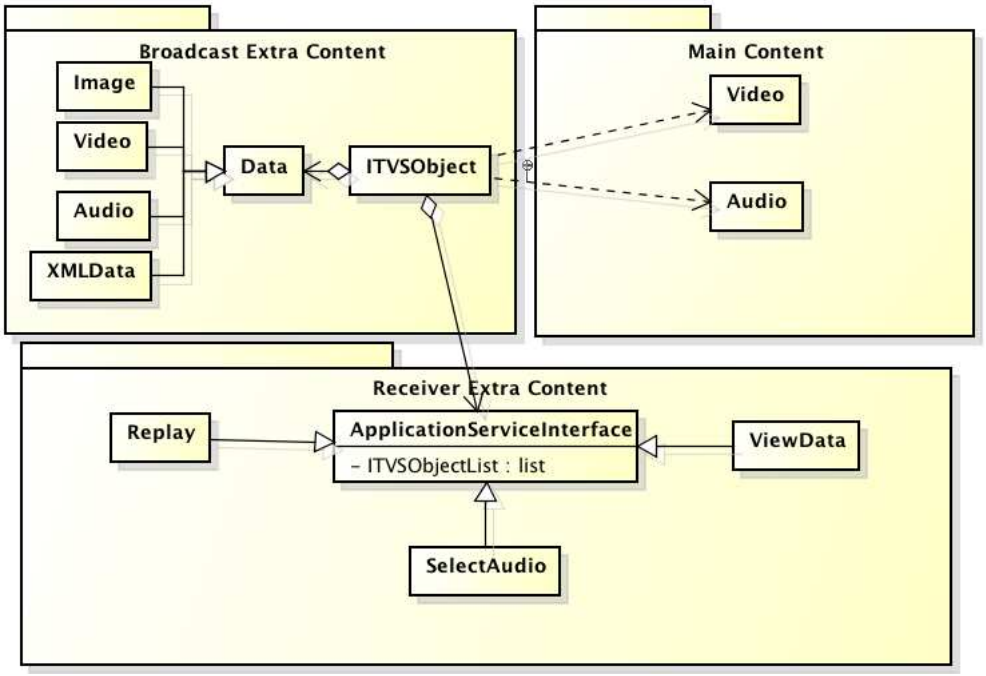

Figura 4.1. Uma representação orientada a objeto para o modelo IDTVS.

Na Figura 4.1, a representação do modelo é dividida em 3 pacotes de classes, denominados *Main Content* (MC), *Broadcast Extra Content* e *Receiver Extra Content* (BEC e REC). O MC contém as classes `Video` e `Audio` que representam uma abstração típica de um conteúdo de um programa de TV (um conjunto de *streams* de vídeo e áudio). O BEC inclui a estrutura de classes dos dados extras do lado difusor (broadcast). O REC contém as classes que representam uma aplicação no receptor (*receiver*) e que permitem processar o conteúdo extra transmitido pelo difusor.

No pacote BEC, a classe `Data` representa uma generalização de conteúdos extra no difusor. Ela pode ser especializada para representar qualquer tipo de conteúdo extra, tais como imagens, vídeo e áudio extras (diferentes do conteúdo principal), dados estruturados em XML e outros. Essa classe tem a mesma função da classe `URI` do modelo IDTVS descrito na seção 4.1.1. A classe mais importante desse pacote é a `ITVSObject`. Esta classe representa uma agregação de objetos `Data` que pode ser especializada para permitir estruturar atributos específicos de uma aplicação.

A relação de dependência entre a classe `ITVSObject` e as classes do pacote Main Content (`Video` e `Audio`) denotam que os objetos `ITVSObject` podem ser dependentes da dinâmica do conteúdo. É exatamente o que ocorre em algumas aplicações. Por exemplo, no contexto da transmissão de uma corrida de Formula 1 (F1), um objeto `PitStop` pode ser um `ITVSObject`

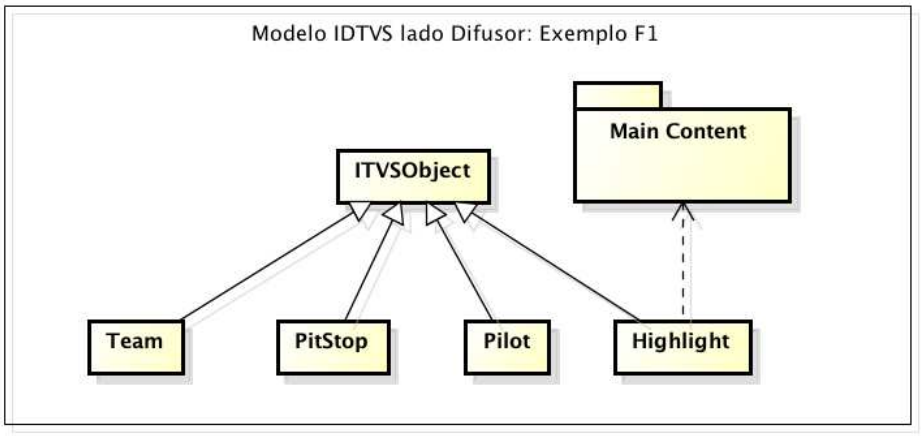

Figura 4.2. Instância do modelo IDTVS para um programa interativo de F1.

que agrega dois objetos `Data`, um vídeo com o replay do pitstop e um XML cujo conteúdo representa a contagem do tempo desse mesmo pitstop, conforme Figura 4.2. Estes dois objetos `Data` são fortemente dependentes da dinâmica do conteúdo. Porém, em outras situações, a relação de dependência pode não ser tão forte. No mesmo contexto da F1, um objeto `Team` poderia ser um `ITVSObject` que agregaria objetos `Data` com informações em XML sobre cada equipe do campeonato (ex: país, escudo, estatísticas etc.). Neste caso, o objeto `Data` é independente da dinâmica do conteúdo, já que as informações sobre a equipe já existiam antes do início da transmissão da corrida.

No pacote REC, a classe mais importante é a `ApplicationServiceInterface` que encapsula uma lista de `ITVSObjects`. Esta classe representa uma interface genérica de aplicação dos serviços disponíveis para processar instâncias de `ITVSObject` enviadas pelo difusor. Ela pode ser especializada para dar suporte aos requisitos de serviço das aplicações, tais como serviço de replay, serviço de seleção de câmera, serviço de seleção de idiomas (legendas) etc.

4.1.3 A Dinâmica do Modelo

Uma das limitações iniciais do IDTVS foi a indefinição da dinâmica de criação, processamento e transmissão do conteúdo extra. No modelo, a dinâmica de criação de objetos é também dependente da dinâmica do conteúdo. No contexto da F1 citado anteriormente, fica claro quando o difusor irá criar o objeto `PitStop`. A cada vez que um pit stop ocorrer, e somente quando ele ocorrer, o objeto `PitStop` será criado, armazenado em uma lista de objetos `PitStop` (que cabem em uma lista de `ITVSObjects`) e, finalmente, transmitido. Entretanto, a dinâmica

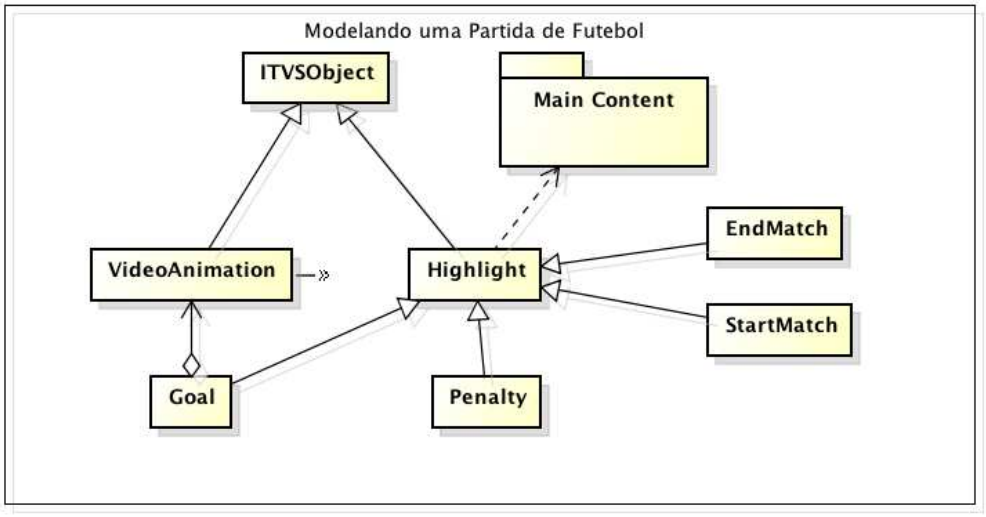

Figura 4.3. Diagrama de Classes dos pacotes MC e BEC para um cenário de aplicação de uma partida de futebol.

de criação do objeto não é a mesma quando o difusor cria os objetos `Team` associados às equipes. Estes últimos devem ser criados, armazenados e transmitidos antes do início da transmissão da corrida. A dinâmica de transmissão pode ser implementada sobre o carrossel de dados [86] ou sobre qualquer outro canal de interatividade disponível. Estas dinâmicas são consideradas uma evolução do IDTVS serão apresentadas a seguir a partir de um cenário envolvendo uma partida de futebol interativa.

4.1.3.1 Modelando uma Partida de Futebol Interativa

Considerando um cenário envolvendo um jogo de futebol com funcionalidades interativas que permitem que o usuário reveja os gols e os pênaltis acontecidos durante a transmissão exibido no diagrama de classes da Figura 4.3. Neste caso, o MC é composto pelas classes `Video` e `Audio` da partida. Os tipos de `ITVSObject` considerados nesta transmissão são os replays dos highlights “gols” e “pênaltis”. Mais ainda, neste cenário, o início e o fim do jogo são considerados como extensões de highligths.

No lado difusor, a classe `Highlight` é uma abstração dos eventos de uma partida de futebol que estende a classe `ITVSObject` e tem forte dependência da dinâmica do conteúdo. Essa classe representa uma agregação de `Video` e `Audio`. A classe `Highlight` é descrita por seu identificador, seu tempo de ocorrência (timestamp) e suas restrições espaço-temporais de apresentação (ex.: descritas em NCL). A classe `Goal` é uma extensão de `Highlight` descrita pelos atributos

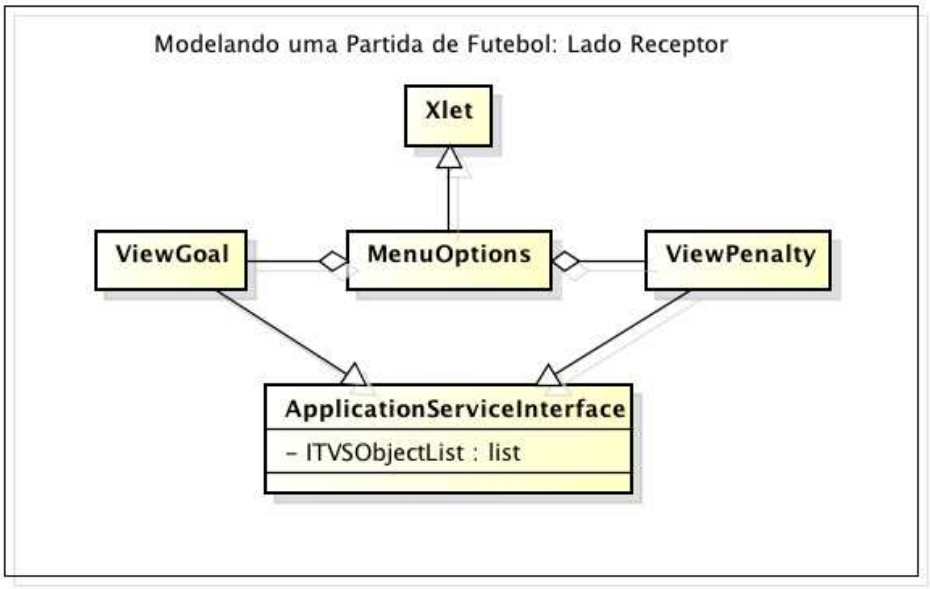

Figura 4.4. Diagrama de classes do pacote REC para um cenário de aplicação de uma partida de futebol.

team, player e animation. A classe `Penalty` é também uma extensão de `Highlight` descrita pelos atributos `for` e `against`. A classe `VideoAnimation` é uma extensão de `ITVSObject` e representa uma abstração da animação do mascote do time que deve ser exibida quando um o gol acontecer. Esta classe é associada com a classe `Goal`. As classes `Goal`, `Penalty` e `VideoAnimation` são extensões de `ITVSObject` representando agregações de Video e Audio.

No lado receptor, o exemplo de aplicação IDTVS pode ser basicamente um menu que permite ao usuário rever os highlights da partida. Este menu pode ser implementado, por exemplo, como uma aplicação do tipo Java TV `Xlet` [128]. A Figura 4.4 ilustra uma visão da aplicação da partida de futebol e o diagrama de classes do pacote REC.

No pacote REC, as classes `ViewPenalty` e `ViewGoal` são interfaces de serviço de aplicações, ambas estendidas da classe `ApplicationServiceInterface` que tem a responsabilidade do processamento das suas respectivas instâncias de objetos `ITVSObject`. Nos dois casos, todos os objetos do tipo `Penalty` e `Goal` armazenados na lista de `ITVSObject` são recuperados e seus atributos de conteúdo são estruturados em um menu na tela da TV através do objeto `MenuOptions`. Esta classe é uma extensão de uma aplicação `Xlet` (Java TV) que permite ao usuário acessar os highlights associados aos gols e pênaltis da partida. A responsabilidade das classes `ViewGoal` e `ViewPenalty` é atualizar suas listas assim que um novo objeto `Goal` ou `Penalty` chegar durante a transmissão da partida. O objeto `MenuOption` usa as informações dos atributos de `Goal` e `Penalty` (ex: time, jogador, vídeo, animação etc.) para permitir que

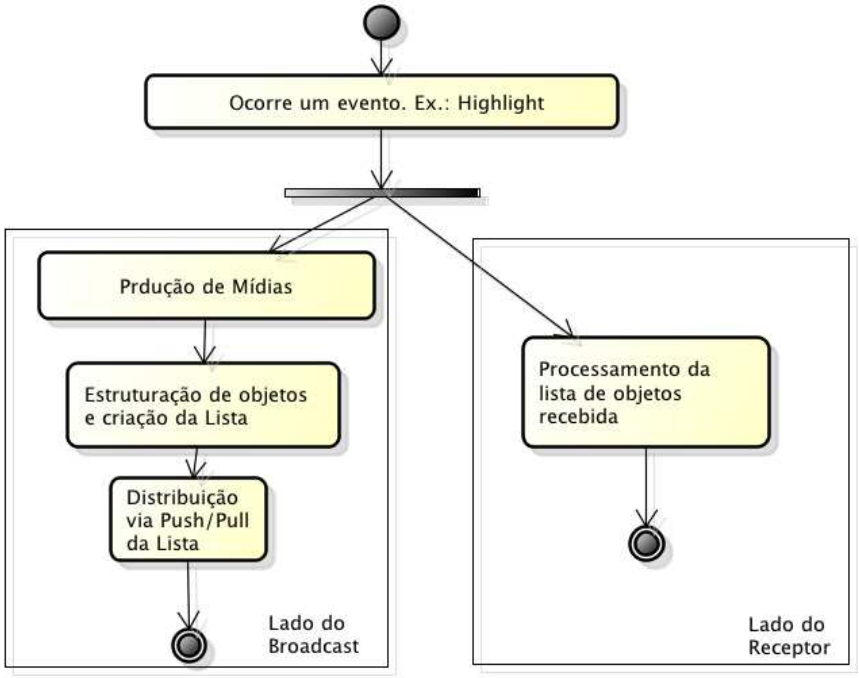

Figura 4.5. Produção e Tratamento de Conteúdo Extra

os replays dos highlights sejam apresentados. A cada seleção (do telespectador) de um item do `MenuOption`, um replay (objeto `Goal` ou `Penalty` que encapsulam um vídeo e demais atributos) será apresentado de forma sincronizada ao conteúdo principal.

4.1.3.2 Fases da Dinâmica

A dinâmica do modelo IDTVS consiste em definir como estruturar e processar o conteúdo extra para programas de TV interativos. No modelo isso deve ser feito em 4 fases relacionadas entre si conforme o diagrama de atividades da Figura 4.5: produção de mídias, estruturação de objetos, distribuição e processamento. A necessidade de transmitir conteúdo extra ou a ocorrência de um evento importante (um highlight) dispara a execução de ações diferentes no difusor e nos receptores de TV. No IDTVS, as três primeiras fases (atividades de produção de mídias, estruturação de objetos e distribuição) são exclusivas do difusor e apenas a última fase (atividade de processamento da lista de objetos) é realizada nos receptores.

Na primeira fase, o IDTVS define que todas as mídias que são independentes da dinâmica do conteúdo devem ser produzidas antes do início da transmissão do programa. Para as mídias dependentes da dinâmica, o IDTVS define que elas devem ser produzidas a partir da ocorrência de eventos (highlights) associados ao conteúdo representado pelas mídias. Por exemplo, no

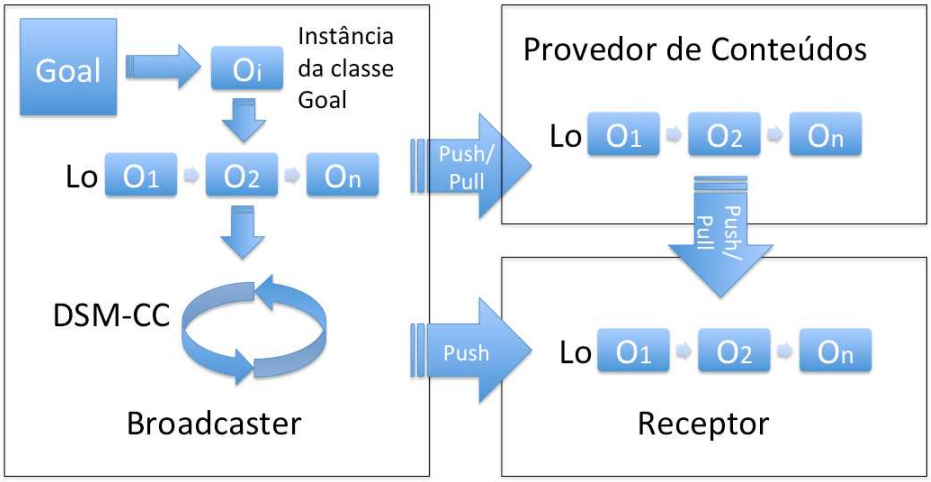

Figura 4.6. Distribuição da Lista de Objetos

cenário descrito na seção 4.1.3.1 a cada ocorrência de um gol a equipe de edição (no difusor) seleciona os quadros do vídeo da partida (main content) que contém esse highlight e produz um novo vídeo (replay). Durante este processo de produção é possível modificar as propriedades do replay, reduzir sua resolução, adicionar outras trilhas de áudio etc.

Na segunda fase, a abordagem permite mapear todo o conteúdo extra criado em um conjunto de dados estruturados (orientado a objetos). Assim, ainda no cenário descrito na seção 4.1.3.1, cada gol pode ser representado, por exemplo, como uma instância da classe `Goal` com os atributos: tipo de mídia, id, timestamp, uri, áudio entre outros. A classe `Goal` é um tipo de objeto IDTVS que pertence ao conjunto C_i e cada instância desta classe é armazenada na lista L_0 de objetos IDTVS.

Na terceira fase, é realizada a distribuição da lista L_0 de objetos IDTVS. Na Figura 4.6 pode-se perceber que esta distribuição pode ser feita no modelo de comunicação pull ou push / pull. No modelo pull, a lista L_0 é transmitida via carrossel de dados, com o conteúdo produzido na primeira fase. Neste caso, o carrossel deve ser atualizado com os objetos e as suas respectivas mídias a cada ocorrência de um evento. No modelo push / pull, as mídias e a lista de objetos são transmitidas (via push ou push / pull) para um provedor de conteúdos (ex: provedor de conteúdos Web) que pode ser acessado pelos receptores de TV via canal de interatividade. É importante ressaltar que o provedor de conteúdos também pode ter sua própria lista de objetos. Dessa forma, ela pode conter objetos interativos complementares, além daqueles produzidos no broadcaster.

A quarta fase é responsável por definir como é realizado o processamento de objetos que

foram gerados e transmitidos (ou acessados) nas fases anteriores. Este processo é feito por um aplicativo residente que implementa a interface de serviços $\in L_A$ e é formalmente é definido por [131]:

On *Event* **If** *Condition(Event)* **Do**(*Action*)

Assim, quando um evento ocorre, **se** uma condição é alcançada, **então** uma ação é realizada. Para isso, o aplicativo residente recebe entradas (inputs na forma de eventos ou composições de eventos) de entidades externas (broadcaster ou outro provedor de conteúdos), usa uma função de avaliação para a informação carregada nas entradas (*Condition(Event)*) e reage a esta função resultando em ações que podem:

i) Modificar / atualizar as estruturas de conteúdos armazenados e

ii) Disparar a execução de funções internas definidas no domínio da aplicação.

A dinâmica definida neste modelo funciona da mesma forma que um protocolo para aplicativos de rede. Ela permite separar da aplicação a responsabilidade de implementar a ocorrência de eventos e o mecanismo de notificação desse eventos. A separação de responsabilidades é um requisito importante para a construção de uma aplicação interativa. Este requisito permite a construção de aplicativos de uma forma menos dependente da transmissão de conteúdos extra. Além disso, há também a possibilidade de que as aplicações construídas por atores diferentes possam tratar eventos a partir do mesmo programa.

4.1.4 Contribuições e Limitações do IDTVS

A possibilidade de distribuir a lista de objetos através dos dois modelos de comunicação representa uma contribuição importante do modelo IDTVS: o tratamento do problema de sintonização tardia (*Late Tuning*) [132].

A sintonização tardia é o problema que ocorre se um o usuário sintoniza o canal muito depois do início da transmissão e quando eventos importantes já ocorreram. Neste caso, os objetos decorrentes desses eventos seriam transmitidos depois da sintonização e não estariam disponíveis no receptor. O mecanismo de atualização definido no modelo IDTVS evita este

problema para o modelo de comunicação push. No caso do modelo push / pull, a sintonização tardia não se torna um problema, pois as mídias e a lista de objetos estão sempre disponíveis para download.

Outra abordagem que pode evitar a sintonização tardia pode ser encontrada em [101]. Este trabalho apresenta um mecanismo que permite ao broadcaster editar, ao vivo (durante a apresentação de um programa interativo), aplicações feitas em NCL. Porém, para que isso seja possível é necessário interromper (pausar) a aplicação a cada edição. A vantagem da solução proposta no modelo IDTVS é poder poder atualizar o conteúdo de uma aplicação sem necessariamente pausar a sua execução. Além disso, diferente de [101], o IDTVS suporta conteúdo proveniente de outras fontes (terceiros).

Outra contribuição importante do IDTVS foi a possibilidade de tratar eventos complexos (ex: associados a dinâmica do conteúdo) e não apenas aqueles associados as mídias apresentadas (ex: instantes de inicio e fim de um vídeo) ou a interatividade do usuário, como nas aplicações feitas em NCL. Em NCL isso é feito a partir dos conectores que são utilizados para especificar relações de sincronização espaço-temporais, tratando relações de referência (de interação com usuário) como um caso particular de relações de sincronização temporal. Neste caso, os eventos tratados estão relacionados às mídias (onBegin, onEnd, onResume etc.) ou à interação do usuário (`onKeySelectionAbort`, `onKeySelectionStart` etc.). O diferencial do IDTVS é que a função de avaliação (*Condition(Event)*) pode reagir a um conjunto de relações entre eventos que vão além daqueles suportados em NCL.

O IDTVS é claramente focado nas dimensões de arquitetura e visão da engenharia de software. Dessa forma, apesar da dinâmica do modelo permitir tratar alguns problemas do ambiente de TV, como por a sintonização tardia, o IDTVS é limitado na dimensão de processo. Os motivos para essa limitação são: a indefinição de como devem ser executadas as atividades específicas de um processo de concepção, quais seriam os atores participantes deste processo e falta de suporte ao processo de reaproveitamento das aplicações especificadas.

Buscando resolver as limitações do modelo IDTVS, uma nova evolução foi proposta. Esta nova evolução resultou em um modelo voltado à especificação de programas para TVDI e que utilizou a experiência dos trabalhos encontrados na literatura que tratam a questão do processo de desenvolvimento de programas interativos para TV. As próximas seções apresentam o novo modelo.

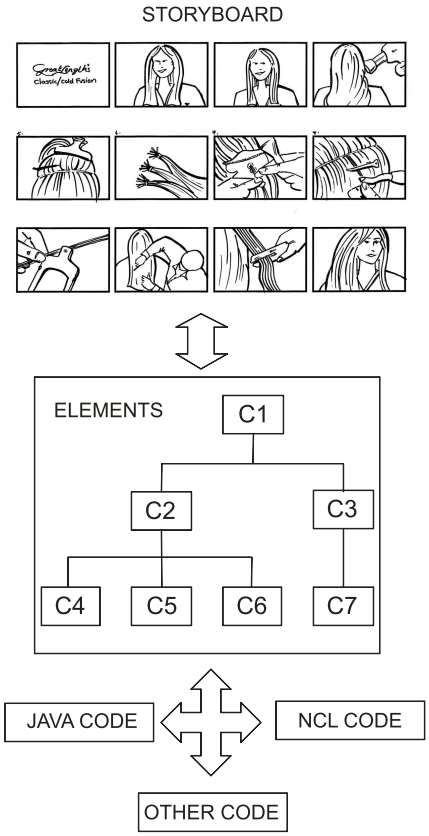

Figura 4.7. Etapas do StoryToCode

4.2 O MODELO STORYTOCODE

O modelo StoryToCode foi concebido como uma evolução do modelo IDTVS apresentado na seção 4.1. Ele está concentrado nas atividades que surgem depois das fases de concepção e elaboração de um programa de TVDI. Essas atividades podem ser agrupadas em duas fases: atividades de produção/geração de mídias (imagens, vídeos, textos, roteiros, cenários, storyboard etc.) e atividades de produção de software (definição de classes, codificação etc.). O StoryToCode trata especificamente as atividades da segunda fase e a responsabilidade da sua execução é da equipe de projeto de software (engenheiros, analistas, programadores, administradores de banco de dados e de redes entre outros) em colaboração com toda a equipe de TV (Diretores, produtores, cenógrafos etc).

O objetivo do StoryToCode é permitir a especificação e construção e reaproveitamento 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 etc.). O StoryToCode parte de um *storyboard/cenário* e usa conceitos de modelagem de sistemas para criar um conjunto de elementos que representem tanto as mídias quanto os com-

ponentes de software, que compõem um programa interativo e ainda destacar algumas visões sistêmicas (estrutura, eventos etc.), a fim de poder reusar os artefatos gerados em outros contextos.

O StoryToCode, assim como o IDTVS, abre a possibilidade de estruturar os módulos interativos de um programa interativo com o diferencial de partir de informações extraídas de um *storyboard/cenário*. A escolha do *storyboard/cenário* como ponto de partida se deve ao fato desse artefato ser um dos comumente utilizado 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.

O StoryToCode está dividido em três etapas relacionadas entre si, ilustradas na Figura 4.7: *storyboard/cenário* (topo da Figura), arquitetura de elementos (meio da Figura) e geração de código (base da Figura). A fim de permitir o intercâmbio entre definições mais abstratas (arquitetura e design) até, possivelmente, o código o StoryToCode baseia-se no conceito de RTE (*RoundTrip Engineering*) [30]. O modelo define como deve ser realizada a transformação de um *storyboard/cenário* em um conjunto de elementos abstratos e, posteriormente, a transformação desses elementos em código.

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

4.2.1 Storyboards / Cenários

Em um ambiente de geração de conteúdo para TVDI, um *storyboard/cenário* pode ser definido como uma técnica usada na descrição da seqüência fundamental de cenas que melhor representam um programa [54]. Para programas interativos de TV estes artefatos registram informações que podem ser úteis para o enriquecimento de um conteúdo interativo. O *storyboard/cenário*, por exemplo, que guia a produção também deve auxiliar a estruturação do programa interativo, fornecendo informações que auxiliem a segmentação do conteúdo. Além disso, informações que detalham uma cena podem ser oferecidas adicionalmente para os telespectadores de um programa interativo. Outro aspecto importante é que, no momento em

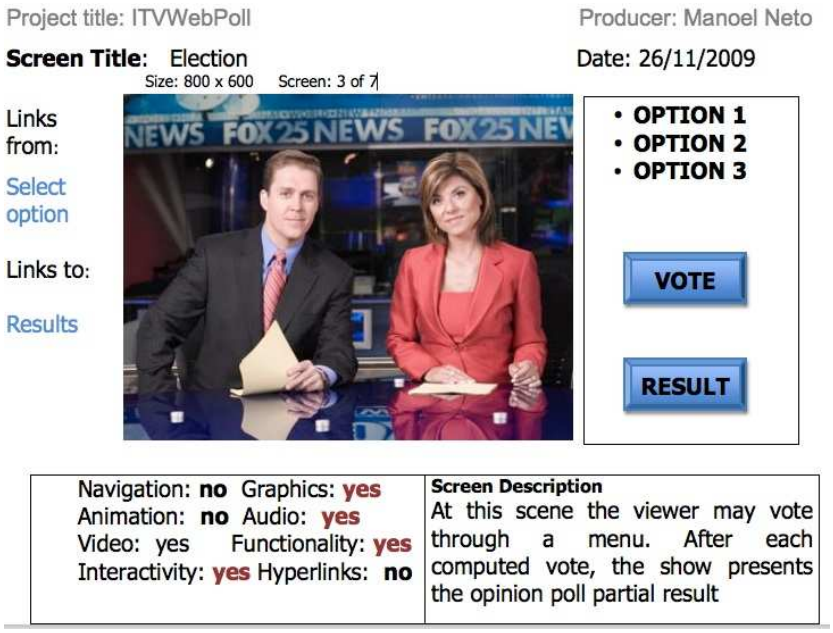

Figura 4.8. Exemplo de quadro do *storyboard/cenário*

que se cria um storyboard/cenário, também se deve criar as possibilidades de inserção para os mecanismos de interatividade (aplicativos). Por exemplo, se um aplicativo que disponibiliza informações sobre a tabela de um campeonato de futebol deve fazer parte do programa, então o storyboard/cenário deve conter um ponteiro para esse aplicativo.

Um *storyboard/cenário* fornece uma descrição de alto nível dos elementos que irão compor cada uma das cenas e dos fluxos de interatividade oriundos do uso de softwares. Apesar de ser uma técnica considerada informal e fracamente estruturada, ela pode ser usada para modelar o fluxo de apresentação das cenas (transição), os efeitos interativos e temporais, narrativas (dublagem, narrações etc.), layout das cenas entre outros. As informações contidas em um *storyboard/cenário* devem conter 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, animaçẽs, elos [$de \rightarrow para$]) [77] [3]. A Figura 4.8 exemplifica a estrutura de um possível *storyboard/cenário* e o utiliza como exemplo de uso do modelo.

No aspecto relativo à transformação do *storyboard/cenário* em elementos de um software, a primeira parte do StoryToCode define que esse artefato deve ser usado como repositório dos requisitos que deverão ser atendidos na construção dos elementos. No modelo, essa transformação é feita de forma não automatizada (Figura 4.9). Assim, a equipe de projeto de software deve usar o *storyboard/cenário* (construído na fase de concepção) para extrair os requisitos e gerar uma representação abstrata do conjunto de elementos contidos nas cenas de um programa

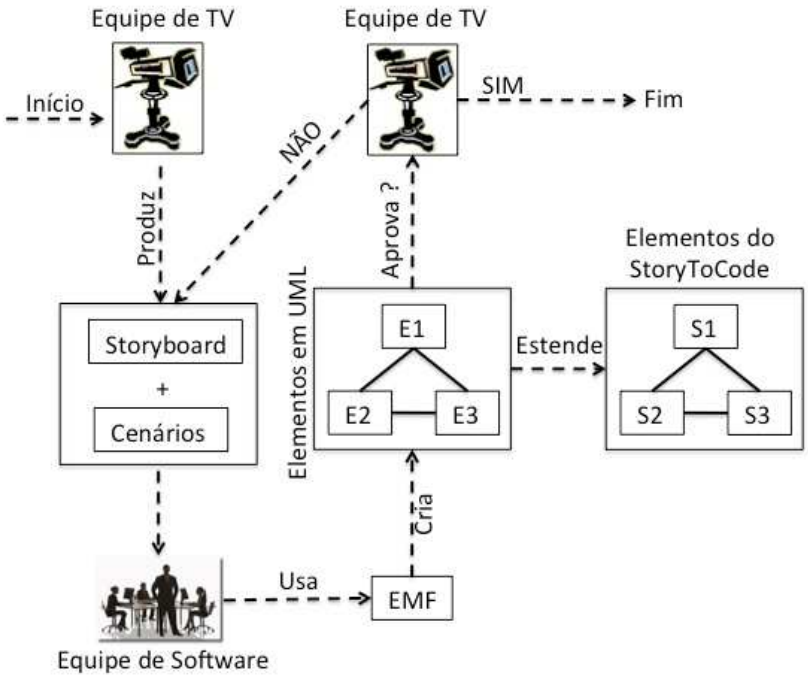

Figura 4.9. Partindo de um *storyboard/cenário* para criar um conjunto de Elementos

com as suas relações e os seus eventos de interatividade. Esse conjunto de elementos está representado pela segunda parte do StoryToCode e é denominado StoryToCodeElements.

4.2.2 Conjunto de Elementos

É importante ressaltar que o StoryToCodeElements não representa apenas um documento para auxílio no entendimento, na manutenção ou evolução de uma aplicação, usualmente encontrado nos modelos conceituais para especificação de softwares. Esse conjunto é também um artefato que pode ser transformado 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, denominada StoryToCodeElements, representados através da notação *Unified Modeling Language* - UML (visual) e XMI (código fonte da notação UML) que podem ser estendidos para se adequar aos requisitos específicos de um *storyboard/cenário*. Essa hierarquia de elementos representa uma evolução do modelo IDTVS (ver seção 4.1.1), foi e pode ser observada na Figura

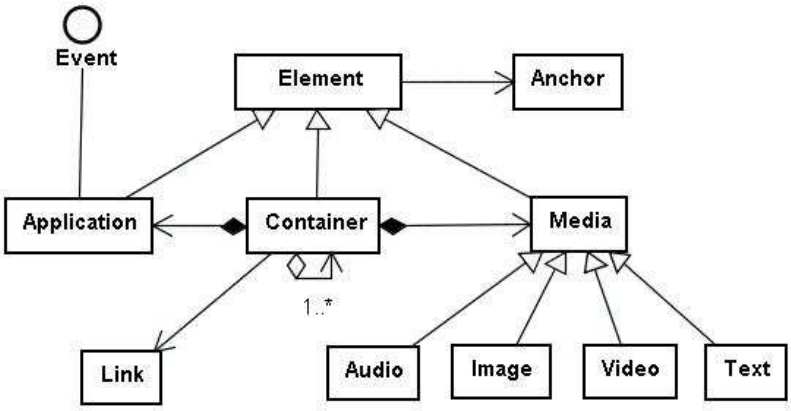

Figura 4.10. StoryToCodeElements: Hierarquia de elementos do StoryToCode

4.10. Ela foi inspirado na hierarquia de classes do *Nested Context Presentation Model* (NCPM) [6] (Figura 4.11) e permite especificação e extensão de elementos nos formatos UML e também XMI. O StoryToCode estabelece que a equipe de software deve usar um framework denominado *Eclipse Modeling Framework* (EMF) [29] (ver Figura 4.9) como ferrameta de auxílio no processo de especificação (não automática) desses elementos.

4.2.2.1 Hierarquia de Elementos

Na raiz da hierarquia do StoryToCodeElements 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 classe que representa uma abstração de um elemento interativo. Essa classe possui como atributos um identificador único, um descritor, que contempla as informações para determinar como um elemento interativo deve ser apresentado, 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 (estruturado) 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* contém 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

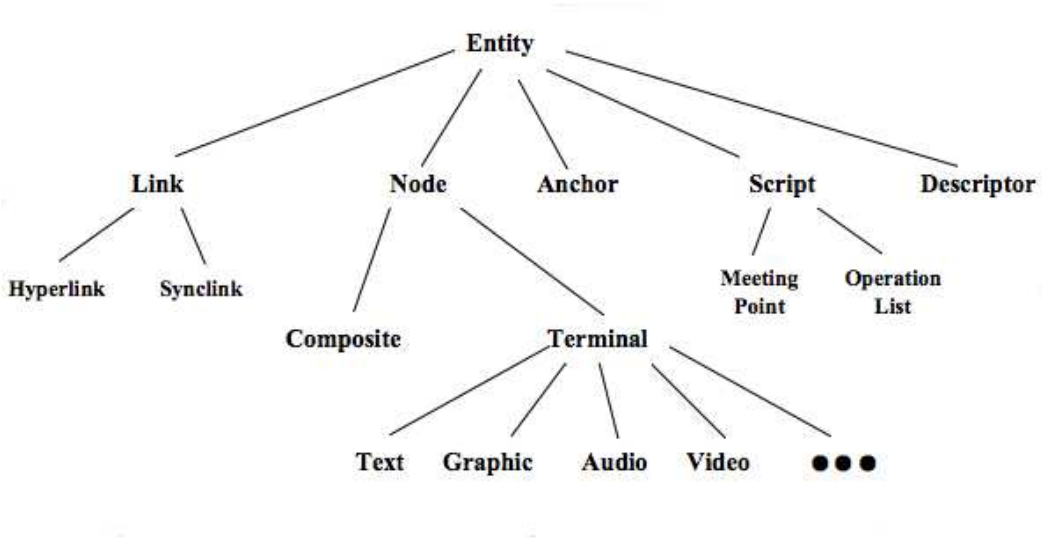

Figura 4.11. Hierarquia de Classes NCPM [6]

storyboard/cenário como vídeo, imagem, texto e áudio.

O elemento *Application* presente na hierarquia tem como função definir as estruturas de dados de um elemento de software cujas operações (instruções) ao serem executadas, representam a responsabilidade de tratar os eventos de interatividade especificados em um *storyboard/cenário*. 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 elementos de software podem abstrair elementos de um *storyboard/cenário* e as ações decorrentes do uso de cada um deles, como selecionar botões, exibir barras de progresso, exibir alertas 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 deve ser usado para agrupar elementos de um *storyboard/cenário* 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.

A abordagem usada no modelo StoryToCode para tratar estuturas de dados representa uma diferença importante entre o NCPM e o StoryToCode. O modelo NCPM [6] não suporta, explicitamente, a extensão de estruturas de dados que abstraiam elementos de negócio ou mesmo elementos visuais. Uma hipótese é que isso poderia ser feito através do elemento *Script*, de-

finido na hierarquia do modelo NCPM. Um *Script* é um elemento cuja função é armazenar o código de um programa (escrito em uma lingagem de programação) e cujas instruções (operações), quando executadas, geram mensagens que invocam o acionamento de métodos válidos de objetos de apresentação. Assim, enquanto o *Script* NCPM (Figura 4.11) é um elemento não extensível com foco nos aspectos espaciais/temporais (visuais) de objetos de apresentação, um elemento *Application* do StoryToCode pode ser estendido para abstrair de forma não ambígua tanto as interfaces visuais quanto a lógica de negócio de uma aplicação.

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/cenário*. Essa capacidade de extensão do modelo permite adequá-lo a outras especificações, tanto no que diz respeito a estruturação das mídias quanto dos componentes de software de um *storyboard/-cenário*. Essa possibilidade de descrever cada elemento que compõe um *storyboard/cenário* 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.

4.2.3 Código

Uma vez que o conjunto de elementos esteja definido pela equipe de software e aprovado pela equipe de TV, 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*, que recebe como entrada o conjunto de elementos e gera como saída o código. A Figura 4.12 exibe a dinâmica de funcionamento do modelo StoryToCode.

Para usar um conjunto de elementos como dados de entrada para um *transformation machine* é essencial convertê-los do formato UML (visual) para o formato XMI (textual). Para isso, o StoryToCode define que a equipe de software deve usar o framework EMF. Esse framework contém um parser XML que permite gerar o código fonte XMI a partir de especificações visuais feitas em UML. De fato, o formato XMI foi escolhido porque muitas ferramentas CASE existentes permitem a criação de diagramas UML e, ainda, permitem exportá-los para XMI. Dessa forma, o StoryToCode torna-se flexível, uma vez que permite ao projetista decidir

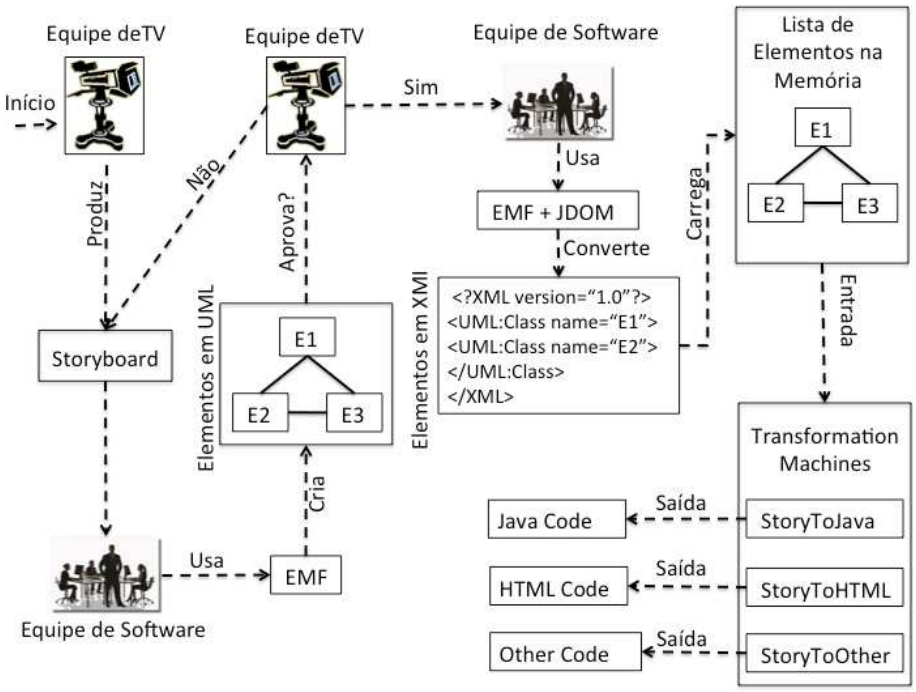

Figura 4.12. Dinâmica completa do processo de geração de código no StoryToCode

que ferramenta CASE deve ser utilizada para criar um conjunto de elementos.

Depois de convertido para o formato XMI, o conjunto de elementos deve ser carregado como entrada para um *transformation machine*. Para isso, o StoryToCode usa a biblioteca JDOM [133] [134]. A JDOM é uma API Java de código fonte aberto, que é baseada nos padrões SAX e DOM, usada para manipular documentos XML. Assim, a JDOM permite carregar em uma árvore DOM (na memória) o conjunto de elementos XMI e essa árvore é usada como entrada para um *transformation machine*. Depois de receber o arquivo XMI e de carregá-lo na memória, a equipe de software escolhe uma das instâncias disponíveis de *transformation machine* para gerar código fonte em uma plataforma específica. Cada instância contem um conjunto de regras de transformação, baseadas no conhecimento de origem (conjunto de elementos), e da estrutura dos elementos de destino (código para uma plataforma específica). Um *transformation machine* permite adicionar regras (e outras informações importantes) para permitir mapear um elemento no seu código fonte correspondente em uma linguagem de programação. Assim, uma transformação de um único conjunto de elementos pode ser realizada para plataformas diferentes, uma vez que a instância do *transformation machine*, específica para cada uma das plataformas, esteja disponível.

A Figura 4.13 ilustra a arquitetura de um *transformation machine*. Cada instância de *trans-*

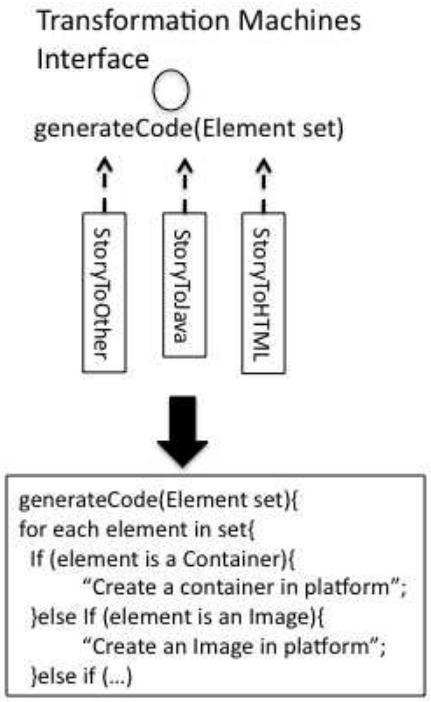

Figura 4.13. Arquitetura dos *transformation machines*

formation machine (ex.: StoryToHTML, StoryToJava) deve implementar a interface *TransformationMachine*, responsável por definir o método "*generateCode*", que recebe como parâmetro o conjunto de elementos carregados do XMI. Nesse método, o componente trabalha como um *parser*, cujo principal objetivo é decidir como lidar com cada elemento do conjunto. Por isso, esses elementos são classificados e tratados como *tokens* em um processo de *parsing*. O tratamento de cada *token* encontrado consiste em invocar um método que realize o mapeamento de um elemento de origem em seu correspondente na plataforma de destino. Por exemplo, se o *token* é um elemento *Container*, o método "*generateContainer*" para uma plataforma específica deve ser executado; se o *token* é um elemento *Image* o método "*createImage*" para uma plataforma específica deve ser executado e assim sucessivamente.

A Figura 4.14 ilustra um exemplo de processo de *parsing*, onde o método *generateCode* é usado para gerar uma tag image do HTML. Assume-se que E1 faz parte do conjunto de elementos e que é modelado como uma extensão do elemento *Image* do StoryToCode com os atributos: *content* (nome do aquivo que contém a imagem) e *hint* (mensagem que deve ser exibida quando a imagem é selecionada). Depois de converter o elemento E1 para XMI e carregá-lo na memória, o *generateCode* faz a chamada ao método *createImage* passado E1 como parâmetro. Este método contém as regras que permitem mapear E1 (*Image*) em uma tag

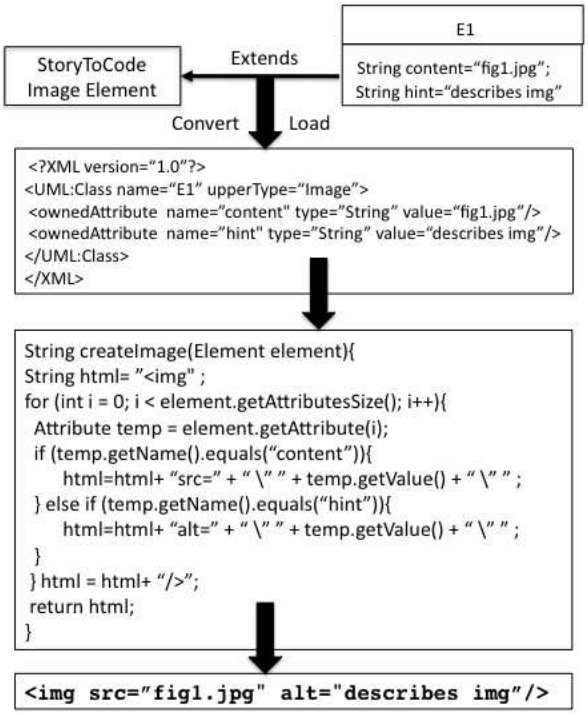

Figura 4.14. Conversão de um Elemento Image para uma tag image em HTML

image (img) da HTML . Para isso, o método usa uma estrutura de repetição para percorrer todos os atributos de E1. Cada atributo é então mapeado para o atributo correspondente em HTML:

```
1 content = "fig1.jpg" é mapeado para src = "fig1.jpg"
2 hint = "describes img" é mapeado para alt = "describes img"
```

4.2.4 Modelando uma Aplicação de TVDI

Para mostrar a aplicabilidade do StoryToCode, criou-se um programa de TV (um telejornal fictício) que deve conter uma enquete interativa 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/cenário* 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.

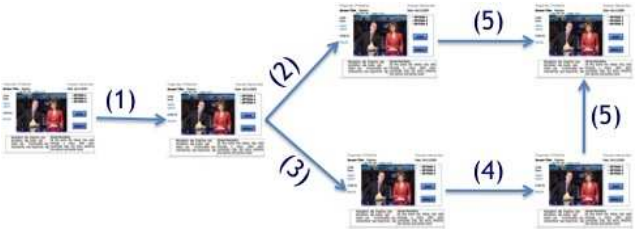

Figura 4.15. Fluxo de cenas do ITVWebPoll

Após votar ele é direcionado à tela de exibição do resultado. Por fim, o usuário deve pressionar o botão vermelho para terminar a aplicação. Se o botão vermelho não for pressionado depois de alguns segundos a aplicação termina. A versão do *storyboard* da cena da "enquete" pode ser vista na Figura 4.8. O fluxo de cenas do programa pode ser visto na Figura 4.15. 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/cenário* 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 Figura 4.16 exibe um diagrama completo do conjunto de classes para o ITVWebPoll. A partir do *storyboard* e da lista de requisitos funcionais passou-se à segunda parte do StoryToCode com a criação do conjunto de elementos que é formado por: *News, FormQuiz, Quiz, Question* e *Answer* além da interface *ActionEvents*. 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

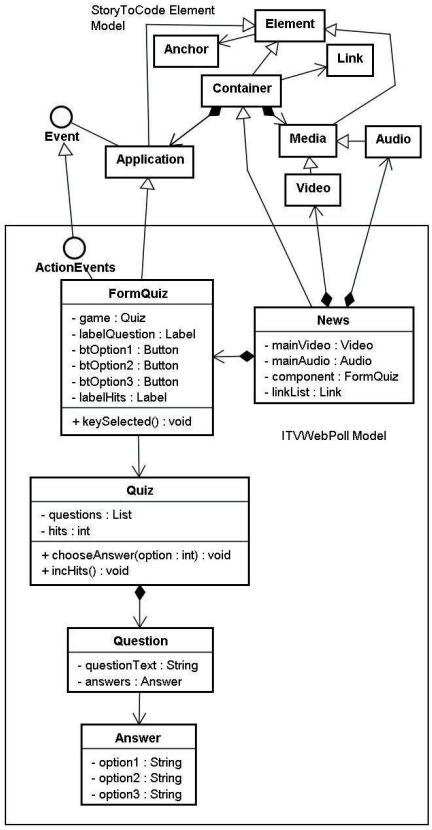

Figura 4.16. Conjunto de elementos

de acertos) e três botões (para exibir as opções de 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 à 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

ELEMENT	ATTRIBUTE	HTML	JAVATV
FormQuiz	Button	Input	HTextButon
FormQuiz	Label	Label	HText
News	FormQuiz	Div	HContainer
News	Video	Embed	HVideoComponent

Figura 4.17. Do Modelo para HTML e do Modelo para JavaTV

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 4.17 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 4.18 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.

4.3 CONTRIBUIÇÕES E LIMITAÇÕES DO STORYTOCODE

O StoryToCode 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.

O StoryToCode 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 consequência disso é o aumento do retrabalho (pela equipe de software) nos ajustes necessários ao modelo antes de sua

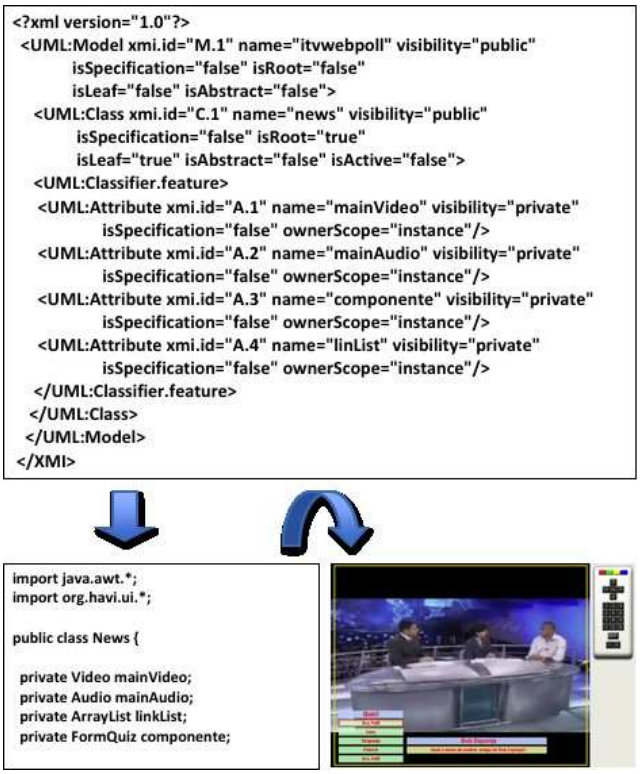

Figura 4.18. Geração de código

aprovação final (pela equipe multimídia).

Outro problema relacionado ao StoryToCode reside 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.

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 [31].

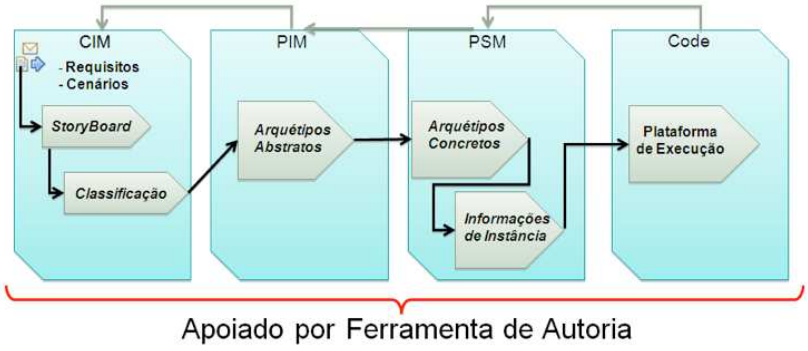

Figura 4.19. Visão Geral da Abordagem Proposta [7]

Buscando solucionar os problemas relacionados anteriormente, a próxima seção 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.

4.4 AUTORIA ORIENTADA A ARQUÉTIPOS

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 etc.) a fim de tornar o desenvolvimento mais rápido, fácil e eficiente, além de permitir o reuso dos artefatos gerados em outros contextos.

Assim, como o StoryToCode, arquitetura proposta aqui é inspirada no MDE [30] e está dividida em 4 partes relacionadas entre si, conforme a Figura 4.19. 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 impor-

tante 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 será descrita.

A primeira etapa tem com 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 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 Mode*l), 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 da classificação é possível citar: (i) consistência entre as aplicações (*branding*), onde é possível definir e seguir um mesmo padrão para a estrutura, identidade visual etc; (ii) a definição de um conjunto de aplicações que permite facilmente a indexação e agrupamento das aplicações; (iii) como consequência 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 deve 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 sugerimos a adoção da linguagem MML [76], tratada na seção 3.3.7.

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 instância, 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 módulo de transformação específico para cada plataforma.

Como ilustrado na Figura 4.19, 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 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.

- Redução da quantidade de código que os programadores precisam produzir quando na criação da aplicação.

- Semi-automatizar a geração de código de aplicações a partir de um conjunto consistente de arquétipos

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 no capítulo de exemplos de uso.

Capítulo 5

Este capítulo apresenta três exemplos de uso realizados para verificar a viabilidade da construção de aplicações multimídia através do IDTVS e do StoryToCode, principais contribuições dessa tese.

EXEMPLOS DE USO

Como prova de conceito para verificar se o StoryToCode e o IDTVS são viáveis para a estruturação de aplicações multimídia em múltiplos contextos essa tese adotou a estratégia de realizar exemplos de uso. Essa decisão levou em consideração a ideia de introduzir o StoryToCode em um ambiente real de produção de aplicações multimídia, que já utilizasse um processo de desenvolvimento de software tradicional validado pela indústria, pois seria questionável aplicar um modelo que está sendo avaliado em um ambiente sem a cultura de uso de modelos de processo. Não foi possível realizar um exemplo de uso em um ambiente real de TV devido a falta dessa infraestrutura local.

Os exemplos de uso realizados visaram identificar e solucionar possíveis problemas durante os experimentos, para que os modelos propostos se tornassem mais maduros e seus resultados não fossem oriundos de variações humanas ou erros experimentais. Para o IDTVS foi construído um simulador que permitisse enviar e exibir um jogo de futebol interativo construído através desse modelo. Para o StoryToCode o primeiro exemplo de uso proposto consistiu na execução das etapas (fases) definidas no modelo para a construção de um jogo interativo nos contextos da Web, TV Digital e dispositivos móveis. A construção do jogo para cada um dos contextos foi feita de modo a reaproveitar elementos interativos especificados. O segundo exemplo de uso envolvendo o StoryToCode para aplicação da abordagem baseada em Arquétipos. As seguintes etapas foram executadas: (i) definição de uma classificação das aplicações em três categorias (*push, pull, hybrid*), 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.

As seções a seguir apresentam os três exemplos de uso e apresentam também os pontos fortes e fracos identificados, bem como as possibilidades de trabalhos futuros para o IDTVS e StoryToCode.

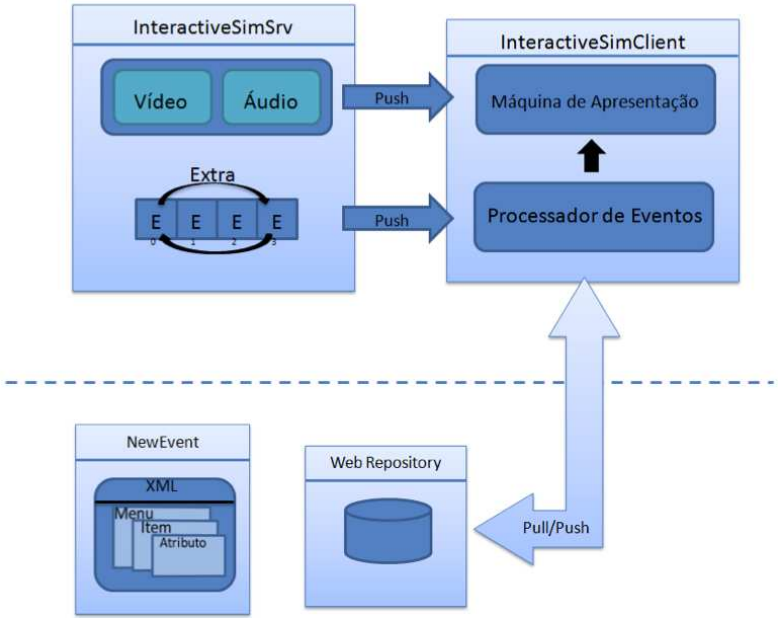

Figura 5.1. Estrutura do Ambiente de Simulação [8]

5.1 EXEMPLO DE USO: IDTVS

Para validar o modelo de aplicação interativa com estruturação de dados extras e classificação de eventos, foi implementado um simulador, denominado de InteractiveSim, para representar o envio de dados (áudio e vídeo principais, e extras), recepção, tratamento e apresentação de programas de TV interativos. Este simulador foi a principal contribuição do trabalho de mestrado [8] que teve entre seus objetivos testar o modelo IDTVS.

5.1.1 Visão da estrutura do Ambiente de Simulação

A Figura 5.1 apresenta o ambiente de simulação desenvolvido (InteractiveSim), que foi montado dentro dos moldes de um ambiente de TV interativa. Neste ambiente, além da emissora de TV (representada pelo InteractiveSimSrv), responsável pelo envio do fluxo principal de áudio e vídeo e pelos dados de programas interativos, encontra-se o receptor (representado pelo InteractiveSimClient), responsável pela decodificação do fluxo principal, interpretação e execução de programas interativos.

Outros elementos, que compõem o ambiente de simulação, representado na Figura 5.1, são: (i) uma fonte externa de dados de programas interativos disponível na Web e (ii) uma aplicação responsável por definir o tipo de aplicação e a estrutura de dados que será utilizada no programa interativo do exemplo de uso.

O primeiro elemento (InteractiveSimSrv) permite o armazenamento de mídias e dados que irão compor o programa interativo, recuperados por mecanismos do tipo *pull*. Este elemento também é capaz de gerar eventos de edição na apresentação do programa, desde que o mesmo esteja previsto na estruturação dos dados da aplicação e seja suportado pela interface de serviços

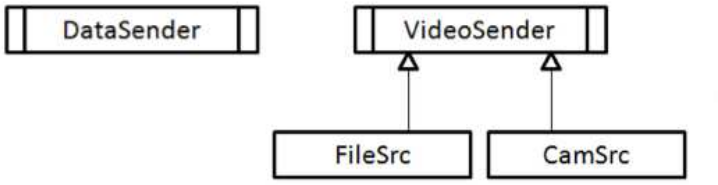

Figura 5.2. Diagrama de classes do InteractiveSimSrv [8]

de aplicação no receptor. Os eventos gerados a partir desta fonte externa são transmitidos ao receptor por mecanismos do tipo *push*.

O segundo elemento (InteractiveSimClient) funciona como um assistente para criação de modelos de dados. Ele é responsável por definir a estrutura de dados que será utilizada no programa interativo e o tipo de serviço de aplicação responsável pelo processamento desses dados. O serviço de aplicação e a estrutura de dados definida pode ser transmitida para o receptor pelo InteractiveSimSrv, via carrossel de dados, ou levada até o receptor por uma mídia de armazenamento externo (ex., um pendrive). A estrutura de dados criada pode definir a fonte das mídias a serem apresentadas no cliente. Estas fontes podem encontrar-se tanto no armazenamento local, recebidas via carrossel de dados, ou em localidades remotas, disponíveis na Web.

Os requisitos funcionais definidos para o ambiente foram: (i) suportar alterações dinâmicas na estrutura de apresentação, possibilitando a edição em tempo de execução da aplicação interativa; (ii) permitir que agentes externos à cadeia regular de produção de conteúdo de TV possam implementar e fornecer aplicações capazes de interagir com programas ao vivo; (iii) permitir que aplicações interativas recuperem mídias em repositórios externos (Web) e (iv) manter a coerência entre a estrutura lógica e a estrutura de apresentação em situações de sintonização tardia.

5.1.1.1 Arquitetura do Ambiente

O InteractiveSimSrv, representado no diagrama de classes da Figura 5.2, é o elemento responsável pela transmissão do fluxo principal de áudio e vídeo, e pelo envio cíclico de dados que compõe o conteúdo extra do programa. A classe ativa `VideoSender` abstrai dados e funcionalidades responsáveis por gerar o fluxo principal do programa interativo e difundi-lo via RTP (Real-time Transport Protocol) [24]. A classe `VideoSender` é especializada pela classe `FileSrc`, caso o fluxo principal de vídeo transmitido seja oriundo de um arquivo. Caso, o fluxo principal de vídeo seja gerado por um dispositivo de captura (*webcam*), a especialização `CamSrc` é a responsável pela implementação da solução.

O envio cíclico de dados (carrossel de dados) é realizado via unicast, utilizando o protocolo de transporte TCP. Esta funcionalidade é abstraída pela classe ativa `DataSender`. Os dados extras, armazenados em um diretório pré-estabelecido, são difundidos de maneira cíclica pela instância da classe. Durante o processo de transmissão, dados de controle, permitem que a es-

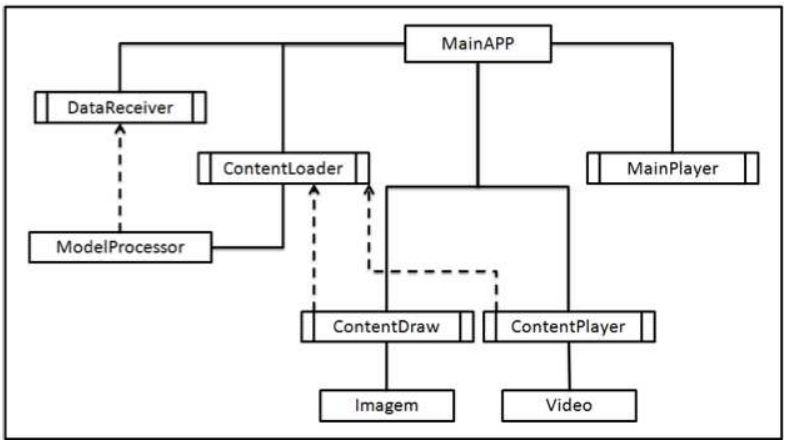

Figura 5.3. Diagrama de classes do InteractiveSimClient [8]

trutura de diretórios adotada na origem seja mantida no destino. A classe `DataSender` realiza o envio de objetos representando eventos ocorridos no programa por meio da adição e atualização dos arquivos que compõem os dados de conteúdo extra.

O InteractiveSimCliente (Figura 5.3) faz a decodificação e apresentação do fluxo principal de áudio e vídeo, executa aplicações e apresenta mídias. A classe `MainApp` define o layout da aplicação e está associada a quatro classes ativas. A classe ativa `MainPlayer` recebe o fluxo principal de vídeo enviado pelo transmissor e o reproduz. A classe `DataReceiver` armazena os dados extras enviados pelo transmissor mantendo a estrutura de diretórios utilizada na origem. A partir do primeiro ciclo completo de transmissão do carrossel de dados, ela verifica se os dados extras enviados pelo transmissor são mais recentes que as cópias locais. Caso haja arquivos novos ou mais recentes, o conteúdo transmitido no novo ciclo do carrossel sobrescreve os dados armazenados anteriormente.

Uma vez que os dados extras encontram-se armazenados localmente, uma instância da classe `ModelProcessor` identifica e processa os objetos recebidos. Este processamento permite a manipulação do layout da aplicação. A adição de novos itens a interface da aplicação e a disponibilização de novos recursos interativos, fica a cargo da classe ativa `ContentLoader`. As classes `ContentDraw` e `ContentPlayer` abstraem funcionalidades associadas a exibição de vídeo e imagens extras disponíveis para a aplicação.

No intuito de facilitar o processo de modelagem estrutural dos dados que compõem a aplicação interativa, foi desenvolvida uma ferramenta, denominada *NewEvent*. Esta ferramenta é responsável por gerar a estrutura dos dados que irão compor a aplicação de acordo com o modelo IDTVS. A tarefa deste componente é definir a estrutura de dados que será utilizada no programa interativo e gerar arquivos que representam a ocorrência de eventos. Consequentemente, também define características da aplicação. A estrutura de dados definida pode ser transmitida para o cliente pelo InteractiveSimSrv ou transportada em uma mídia externa (pendrive). O *NewEvent* pode ainda definir, a partir dos dados estruturados, a fonte (origem) das mídias extras a serem apresentadas no cliente. Estas fontes podem encontrar-se tanto no arma-

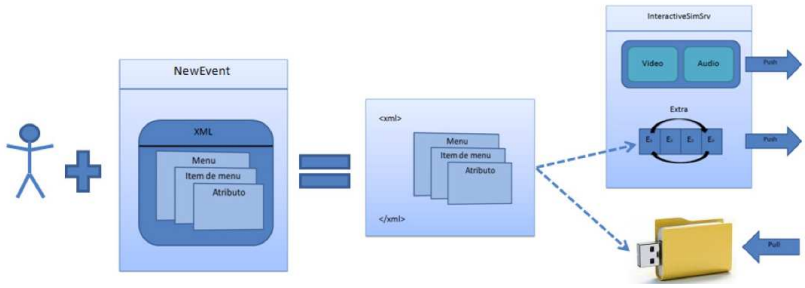

Figura 5.4. Dinâmica de Funcionamento: Estruturação e Distribuição [8]

zenamento local, recebidas via carrossel de dados, ou em localidades remotas, disponíveis na Web.

O *WebRepository*, como o próprio nome já diz, é um repositório de dados disponível na Web que armazena dados extras gerados no decorrer do programa interativo e também pode servir de fonte geradora de eventos de aplicação para o InteractiveSimCliente.

5.1.1.2 Dinâmica de Funcionamento

Dentre as etapas executadas no ambiente de simulação para a apresentação de programas interativos ao vivo, três fases distintas podem ser identificadas na simulação de ambiente real de produção de conteúdo interativo ao vivo, para TV.

Durante a primeira fase (Figura 5.4), denominada estruturação, é realizada a modelagem formal dos dados que compõem a estrutura do conteúdo de mídia. Nesta etapa, o processo de autoria pode ser facilitado com o uso de uma ferramenta, que a partir de templates pré-estabelecidas, gera o modelo de estrutura de dados da aplicação interativa (ex.:*NewEvent*).

Na segunda fase (Figura 5.4), denominada distribuição, o documento XML, que define o modelo, é adicionado ao carrossel de dados do responsável pela difusão do fluxo principal de áudio e vídeo, e dados extras (ex.: a emissora de TV). Esse documento acompanhará os dados extras de aplicação. Outra opção para distribuição do modelo de estruturação é seu armazenamento em uma mídia alternativa, como pendrive, ou mesmo um servidor Web. Nesta última alternativa de distribuição, uma aplicação em execução no receptor faria a recuperação do documento XML, via mecanismos do tipo *pull*, e sua posterior interpretação.

A seguir ocorre a fase de operação (Figura 5.5), onde eventos e dados extras são gerados e adicionados ao carrossel de dados para serem enviados ao receptor. Estas mídias, embora não descritas no documento inicial, que define a estrutura de apresentação da aplicação, são organizadas de acordo com o modelo definido na fase de estruturação. Portanto, embora não se tenha conhecimento prévio a respeito das mídias e relacionamentos que virão a compor a aplicação, a existência de um modelo de estruturação de dados permite que a aplicação identifique e defina em tempo de execução seus relacionamentos e restrições de acordo com tipo de dado recebido. Durante esta fase há também a possibilidade de que eventos e dados extras sejam informados ao receptor por meio de uma fonte externa disponível na Web. Para isso basta que a aplicação

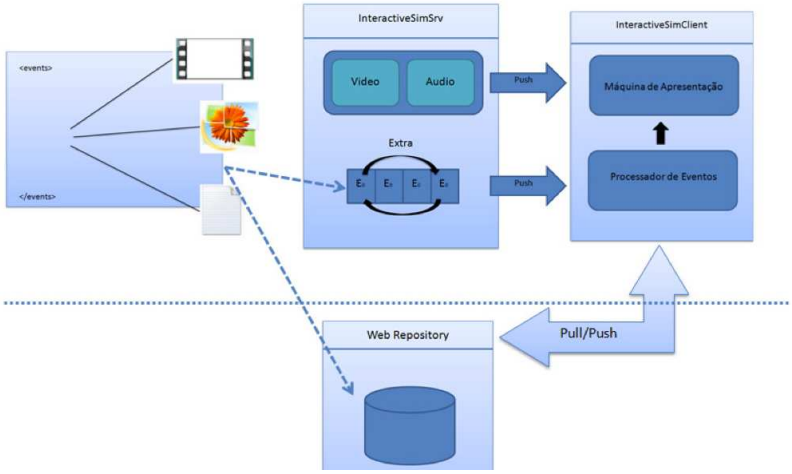

Figura 5.5. Dinâmica de Funcionamento: Operação [8]

em execução no receptor estabeleça uma conexão com a fonte externa.

5.1.2 Modelando uma partida de Futebol interativa

A transmissão de um jogo de futebol ao vivo, permite ilustrar o uso dos recursos oriundos da estruturação de dados e da geração de eventos, pelo transmissor. Sem dependência ao que venha a ocorrer durante a partida, podem ser enviados pelo transmissor dados contendo: a escalação dos times, o esquema tático adotado, posição atual no campeonato, etc. A aplicação interativa, por sua vez, sabendo da existência de tais dados pode tratá-los de acordo com a sua implementação. Estes dados, disponíveis desde o inicio da aplicação, são denominados estáticos. Durante o decorrer da partida, dados e eventos podem ser gerados em função da dinâmica do jogo. Gols, faltas, impedimentos, lances polêmicos, substituições, etc., são exemplos de eventos previstos, porém sem tempo de ocorrência (ou mesmo ocorrência) bem definida durante a partida. A ocorrência de um destes eventos, pode então gerar uma ação que represente a ocorrência do evento. Este evento pode então ser comunicado a aplicação e associado com as mídias e dados extras para processamento (ou exibição) no receptor. Os dados gerados em função da dinâmica da partida são então denominados, dinâmicos.

A classificação em dados estáticos e dinâmicos, entre os dois tipos de dados extras disponíveis para a aplicação, em execução no receptor, assim como sua organização em uma estrutura de diretórios de formato fixo, facilita o acesso aos dados, por parte da aplicação, devido ao seu conhecimento prévio a respeito da localização dos mesmos.

Na aplicação desenvolvida, a estrutura de diretórios utilizada é definida pelo transmissor. Este mesmo modelo pode ser empregado em um ambiente real, onde o radio difusor se encarregaria em definir a estrutura organizacional dos dados extras por ele enviados, aos interessados em desenvolver aplicações interativas para programas de TV.

A transmissão, e posterior acesso, aos dados estáticos e dinâmicos, é realizada por meio de sua organização na estrutura de diretórios. Essa estrutura é transmitida via carrossel de dados ao

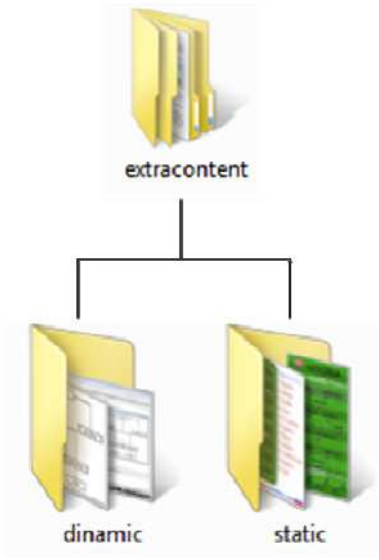

Figura 5.6. Estrutura de Diretórios no Receptor [8]

receptor e recuperada pela aplicação. A estrutura de dados dinâmicos e estáticos é representada na Figura 5.6.

A representação da ocorrência de eventos, acontecidos durante o decorrer da partida é realizada por meio da criação de arquivos XML. Estes arquivos descrevem o evento e o associam a mídias extras (arquivos de vídeo, imagens, áudio, animações, etc.). Os arquivos XML são adicionados ao carrossel de dados, na raiz da estrutura de diretórios criada, e enviados ao receptor. A chegada de arquivos XML, na raiz do diretório de destino, dispara o processamento do arquivo e o tratamento das funcionalidades a ele associadas.

5.1.2.1 Geração, Transmissão e Tratamento de Eventos

Na aplicação desenvolvida, a criação dos arquivos XML, que representam a ocorrência de eventos, é realizada pela ferramenta *NewEvent*. A ocorrência do evento, no receptor, é determinada pela chegada do arquivo XML visto a seguir. A ferramenta *NewEvent*, ao gerar o XML realiza a cópia do arquivo para o diretório raiz de origem dos dados transportados pelo carrossel. No próximo ciclo de transporte do carrossel o arquivo é enviado aos receptores.

```
1 <?xml vesion=1.0 encoding=ISO8859-1>
2    <GOL>
3       <ID>1</ID>
4       <AUTOR>Junior</AUTOR>
5        <TIME>Vitória<TIME>
6        <VIDEO> tex.mov</VIDEO>
7    <GOL>
```

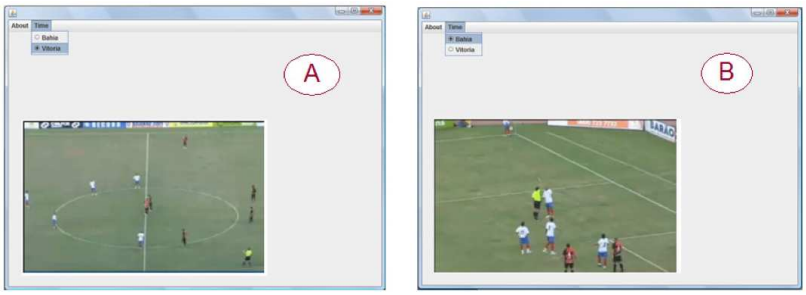

Figura 5.7. Seleção, no receptor, de informação de acordo com o perfil do usuário [8]

Figura 5.8. Comportamento gerado de acordo com o perfil do usuário [8]

No receptor, o processamento do arquivo XML, ou seja, o comportamento adotado pela aplicação, a partir da sua chegada, é dependente da implementação, o que confere ao sistema a capacidade de tratar um conteúdo genérico de forma personalizada, conforme representado na Figura 5.7.

A implementação da aplicação, no receptor, permite que o usuário selecione um time de sua preferência, dentre os times envolvidos na partida em questão. A montagem do menu que permite esta seleção pode ser realizada de acordo com arquivos de dados estáticos contendo informações sobre os times envolvidos na disputa. Uma vez selecionado o time de preferência do usuário (Figura 5.7), a aplicação, ao receber informações referentes a eventos, ocorridos durante o andamento da partida, pode apresentar um comportamento de acordo com o perfil do usuário (Figura 5.8).

No exemplo ilustrado, dois usuários (usuário A e usuário B), selecionam respectivamente Vitória e Bahia como time de sua preferência. A aplicação em execução no receptor do usuário A, ao receber a notificação do evento de GOL do time Vitória, dispara automaticamente, a apresentação de uma mídia definida de acordo com o seu perfil (Figura 5.8 - A). No receptor do usuário B, o mesmo ocorre, porém devido à sua informação de perfil, outra mídia é apresentada, após a notificação do evento de GOL (Figura 5.8 - B).

Além da apresentação automática de mídias em função da ocorrência de um determinado evento, conforme ilustrado anteriormente, a implementação cria itens de menu, associados ao evento ocorrido, que permitem a apresentação de mídias dinâmicas, identificadas no arquivo XML que representa o evento. O arquivo XML que representa a ocorrência de um evento do tipo GOL, associa o evento a uma mídia do tipo VIDEO e identifica o arquivo de vídeo pela tag,

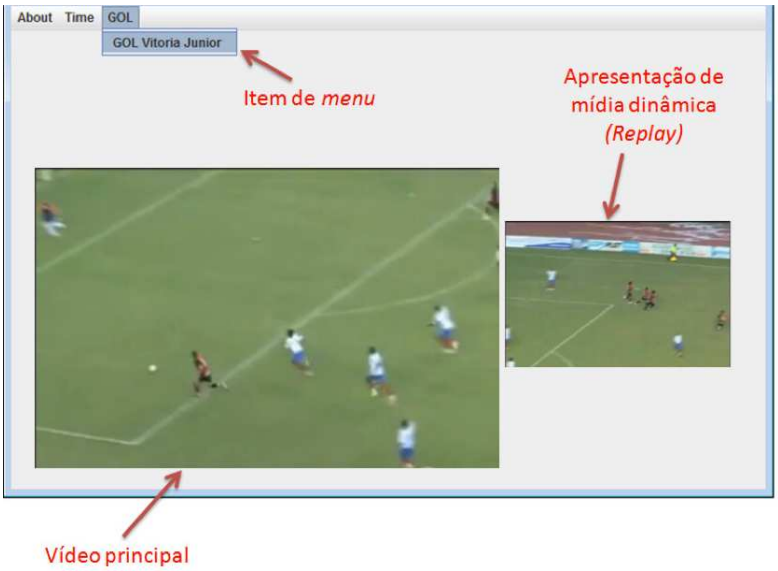

Figura 5.9. Apresentação de mídias dinâmicas (extras) de acordo com solicitação do usuário [8]

<VIDEO>test.mov</VIDEO>. A seleção do item de menu que identifica a ocorrência do gol, dispara a apresentação do vídeo, contendo o recorte do gol, como pode ser observado na Figura 5.9.

A implementação realizada contempla a ocorrência de duas classes de eventos, gerados dinamicamente pelo difusor, a classe GOL e a classe FALTA. Porém, durante uma partida de futebol, outras classes de eventos podem ser esperadas, como IMPEDIMENTO, recebimento de CARTÃO, etc. Como demonstrado, o conhecimento prévio sobre a possibilidade de ocorrência de tais eventos, acompanhados da modelagem estrutural dos elementos que o compõem, permite seu tratamento de acordo com a implementação da aplicação em execução no receptor.

5.1.2.2 Considerações

O ambiente implementado, embora seja capaz de simular o funcionamento de uma estação de TV digital, no que concerne a difusão de um fluxo de dados contendo vídeo e áudio (conteúdo principal) e dados (conteúdo extra), utiliza para a entrega dos dados extras contidos no carrossel redes de computadores como meio de transmissão de dados e o protocolo de transporte confiável (TCP) como padrão. Estas características, adicionadas ao poder computacional das máquinas utilizadas durante os testes, conferem ao ambiente de simulação um comportamento distinto ao encontrado no ambiente real de transmissão e execução de programas de TV interativos.

Em uma transmissão de TV via radiodifusão, dados enviados pelo carrossel de dados não contam com um mecanismo de verificação e correção de erros, implementados pela camada de transporte do TCP. Ao receber um arquivo de dados, o receptor apenas confere a integridade dos mesmos, através do mecanismo de verificação de redundância cíclica (CRC) e, caso identifique a corrupção do arquivo, o receptor aguarda pelo reenvio do arquivo no próximo ciclo do carrossel, disponibilizando o inicio da aplicação apenas quando todos os dados que a compõem

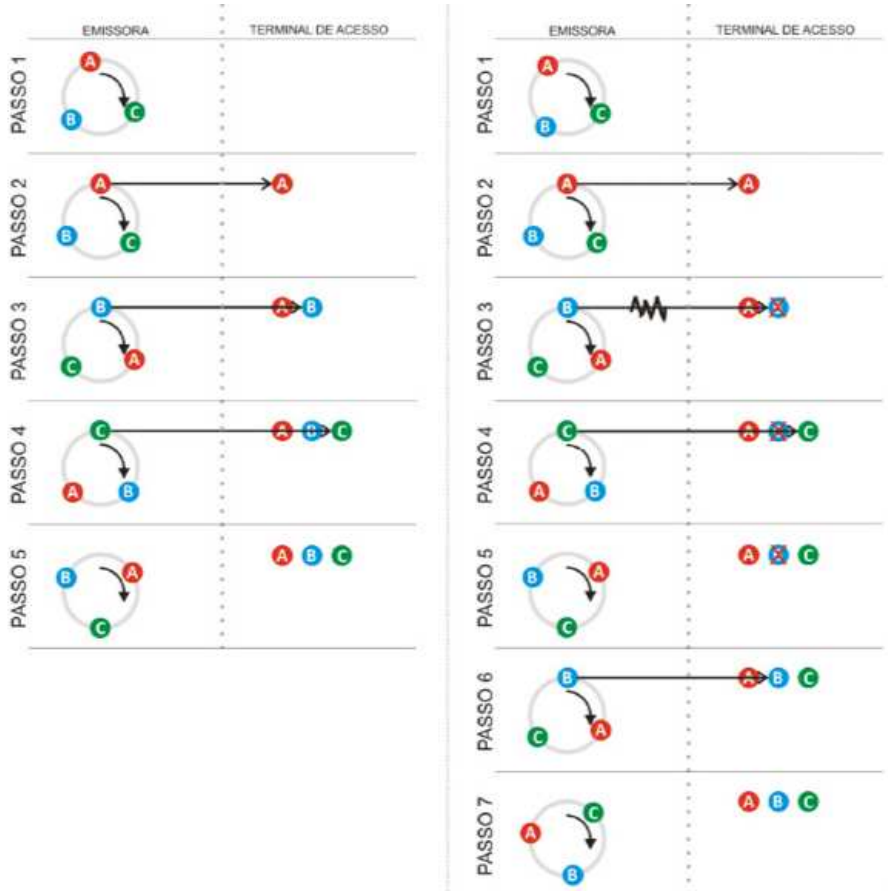

Figura 5.10. Envio de dados pelo carrossel e tratamento de arquivo corrompido [8]

estiverem disponíveis no receptor, conforme ilustrado na Figura 5.10.

Esta característica pode fazer com que dados dinâmicos gerados no decorrer da apresentação demorem mais do que o esperado para que estejam disponíveis no receptor. Este fato pode levar a ocorrência de situações indesejadas, como a apresentação automática de uma animação comemorando um gol, tempos depois da ocorrência do mesmo. Outra consideração se dá ao fato da aplicação só se tornar disponível para execução, quando todos os dados que a compõem estiverem carregados na memória do receptor. Isto exige que a cada atualização do carrossel de dados a aplicação seja reiniciada no receptor, o que também pode levar a situações indesejadas, como a interrupção na apresentação de uma mídia extra, solicitada via interatividade do usuário.

Em função da maior largura de banda disponível para o envio de dados extras em uma rede de computadores, outra questão deve ser observada. Para minimizar o impacto desta característica, no que diz respeito à velocidade com que a inclusão de novos dados ao carrossel seria transmitida aos receptores, foi adicionado um atraso estimado de maneira empírica a cada novo início de um ciclo de transmissão do carrossel de dados.

No problema da sintonização tardia de um aparelho receptor em um programa interativo de TV, após o início do envio de dados extras, foi possível viabilizar o acessoda aplicação às mídias inseridas ao carrossel de dados em momentos anteriores à sintonização. No entanto, a aplicação em execuç ao no receptor, ao receber dados caracterizando a ocorrência de eventos passados,

como gols e faltas, pode apresentar comportamento dinâmico em função dos eventos recebidos fora de seu prazo de validade, como disparar uma animação para um gol que ocorreu há vários minutos atrás. Uma possível solução para este tipo de ocorrência pode ser apresentada na forma de uma restrição temporal que indica a validade daquele evento. Caso o evento recebido não esteja dentro de seu prazo de validade, cabe à aplicação tratá-lo de maneira apropriada, como por exemplo, descartando seu conteúdo.

Para concluir, resta abordar a questão da disponibilidade de maior poder computacional no ambiente simulado. Embora o receptor possa realizar a decodificação e apresentação do fluxo principal de áudio e vídeo de maneira otimizada, em função da adoção de mecanismos de hardware (*firmwares*) com esta finalidade específica, a capacidade de execução de aplicações interativas e a manipulação de dados extras que as compõem é inferior quando comparada com um PC. De acordo com o Forum do SBTVD estarão disponíveis duas versões distintas do *middleware*. A primeira, portada para um *hardware* simplificado e sem a disponibilidade de executar aplicações interativas escritas em Java. A outra, portada em dispositivos de maior capacidade, com suporte a aplicações interativas escritas em Java. Ainda assim, em dispositivos com maior poder de processamento, a capacidade de sua memória principal pode impedir, em alguns aparelhos, a execução de algumas aplicações interativas.

5.2 EXEMPLO DE USO: PROJETO ÁRBARO

O público alvo para o StoryTocode são equipes multidisciplinares com interesse em desenvolver aplicações multimídia interativas para plataformas como TV Digital, Web, dispositivos móveis etc. Equipes multidisciplinares podem ser encontradas em emissoras de TV, indústria de cinema, de jogos, dentre outras. Conforme dito anteriormente, diante da inexistência de emissoras locais de TV com infra necessária para produzir e veicular programas interativos, optou-se para este estudo de caso por uma empresa de desenvolvimento de jogos interativos.

A Gamenyx, ambiente escolhido para o estudo de caso, é uma empresa brasileira de desenvolvimento de jogos eletrônicos interativos, com sede em Salvador-Ba, fundada em abril de 2009. Os jogos produzidos pela empresa são aplicativos eletrônicos que utilizam plataforma Web. Além disso, a empresa também produz jogos para dispositivos móveis e para a plataforma de TV digital brasileira.

A empresa disponibilizou o projeto Árbaro para possibilitar a avaliação do modelo StoryToCode. Este projeto teve como objetivo desenvolver um jogo interativo (para Web, SmartPhones e TV Digital) que trabalhasse noções de ecologia e meio-ambiente, para conscientizar e gerar conhecimentos e aprendizados no âmbito da educação ambiental, focando na área de reflorestamento e plantio de árvores. O público alvo do jogo - os seus usuários - são crianças na faixa etária de 7 a 14 anos. Além destes, estão envolvidos (de forma indireta) os pais e educadores das crianças, bem como os patrocinadores do projeto, com o interesse de fomentar e acompanhar o aprendizado.

Tabela 5.1. Lista dos membros da equipe com seus respectivos papéis e responsabilidades

Equipe do Projeto Árbaro
Game Designer: Criação da artistica do jogo (Fases, personagens etc.)
Consultora: Apoio para as questões de meio ambiente
Ilustrador/Animador: Criação das animações, e das ilustrações do jogo
Ilustrador/ Web Designer: Criação das ilustrações do jogo e Design Web
Projetista: Definição dos requisitos e do conjunto de componentes interativos
Programador: Codificação da aplicação
Gerente do Projeto: Controle diversas tarefas do projeto

5.2.1 Elaboração

Para avaliar o StoryToCode na implementação do projeto Árbaro, foi necessário, inicialmente, definir a equipe. A definição levou em consideração as competências necessárias para construção do jogo, tais como: criação artística, design gráfico, projeto e programação de software e conhecimento específicos de biologia (crescimento de plantas, ecologia etc.). A tabela 5.1 apresenta cada um dos membros dessa equipe e seus respectivos papéis e responsabilidades.

O passo seguinte foi apresentar para equipe tanto a ideia inicial do projeto Árbaro quanto o modelo StoryToCode. Assim, antes de submeter à equipe a utilização do modelo optou-se por ministrar um curso de curta duração (3 horas) dividido em duas etapas, realizadas na sede da Gamenyx. A primeira delas apresentou o modelo StoryToCode através de aulas expositivas e práticas. A segunda etapa apresentou a visão geral do projeto. No projeto, a aplicação simula o plantio e desenvolvimento de uma árvore para reflorestamento, iniciando com a atividade de escolha da semente. Este processo será desenvolvido e apresentado ao usuário em etapas:

i) Plantio da Semente

ii) Crescimento de Flores e Frutos

iii) Árvore Adulta

O usuário terá que completar uma etapa antes de passar à próxima. Cada semente plantada é considerada uma planta ativa do usuário na aplicação. Durante o crescimento das plantas ativas, até que cheguem à idade adulta, o usuário será responsável pelos cuidados necessários à sua saúde. Ao completar o desenvolvimento e atingir a terceira etapa, a planta deixa de estar ativa, ou seja, não necessitará mais de cuidados diários, pois será transferida ao bosque de reflorestamento. O usuário poderá rever todas as plantas desenvolvidas por ele e que não estejam mais ativas ao visitar o bosque de reflorestamento.

Depois de definir e treinar a equipe no domínio do projeto e do StoryToCode passou-se para definição das hipóteses. O estudo de caso buscou verificar as seguintes hipóteses sobre o uso do StoryToCode:

i) A estruturação de conteúdo proporcionada pelo StoryToCode facilitou o desenvolvimento do Jogo (Árbaro) ?

ii) Ele facilitou o processo de comunicação entre os participantes com competências diferentes?

iii) O número de linhas de código geradas representou uma diminuição importante de forma a refletir na diminuição da quantidade de trabalho realizado?

iv) Os componentes de software interativos criados para o contexto da TV foram reaproveitados em outros contextos?

v) Os componentes de software interativos criados para outros contextos poderiam ser reaproveitados no contexto da TV ?

5.2.2 Execução

A fase de execução foi dividida em 3 etapas vistas a seguir.

5.2.2.1 Criação

Na primeira etapa da fase de execução a equipe concentrou os trabalhos na criação artistica do jogo. Os responsáveis por essa etapa foram o game designer, os ilustradores e a consultora de meio ambiente. Levando em consideração a ideia inicial apresentada, esses profissionais produziram os personagens do jogo (ilustrações) conforme pode ser visto na figura 5.11. Os personagens foram inspirados nos elementos básicos presentes na natureza: fogo, terra, água e ar.

Depois disso, os mesmos profissionais produziram o storyboard com a descrição dos principais cenários, conforme, por exemplo, Figura 5.12. Através do storyboard e cenários foram especificados: o conceito visual, cortes, tempo, ações, transições etc. Todo o processo de criação foi iniciado com a apresentação dos artefatos feita pelo gerente do projeto ao cliente. Nesta reunião, o cliente solicitou que os personagens produzidos fossem mais “infantis”. Dessa forma, novas ilustrações de personagens foram produzidas e posteriormente aprovadas pelo cliente. O Apêndice **A** apresenta os storyboards e cenários produzidos nesta fase.

5.2.2.2 Projeto

Depois da etapa de criação, a equipe concentrou os trabalhos na elaboração do projeto de software (componentes interativos) para o jogo. Os responsáveis por essa etapa foram o gerente

Figura 5.11. Personagens do Árbaro: Fogo, Água, Terra e Ar

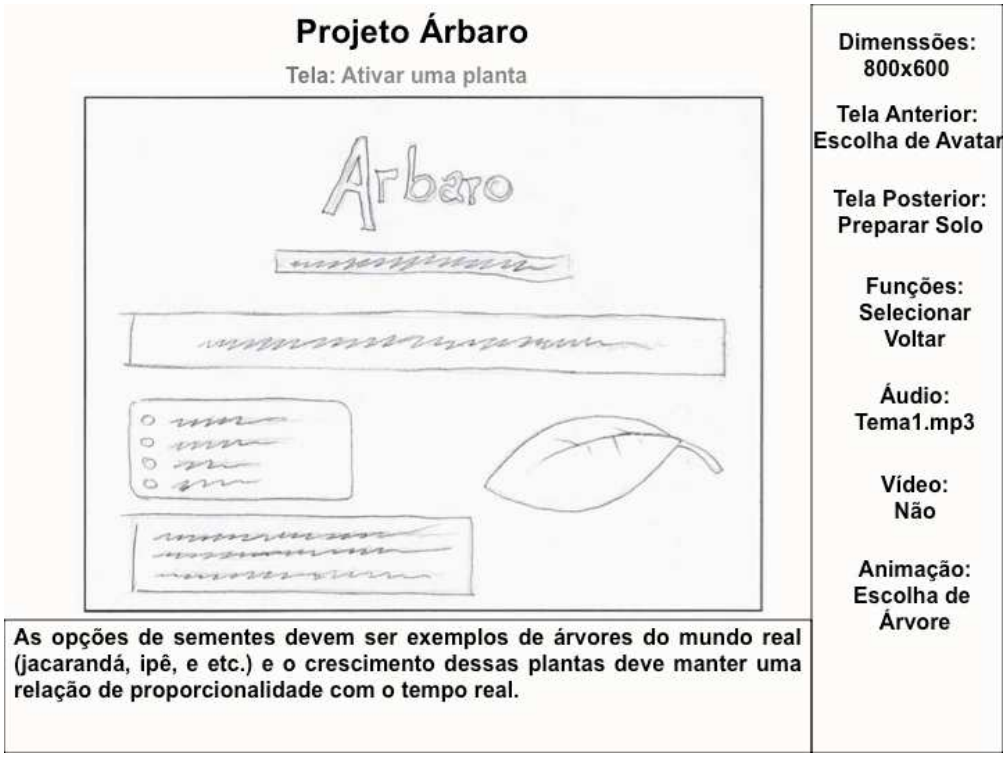

Figura 5.12. Exemplo de Storyboard + Cenário do projeto Árbaro

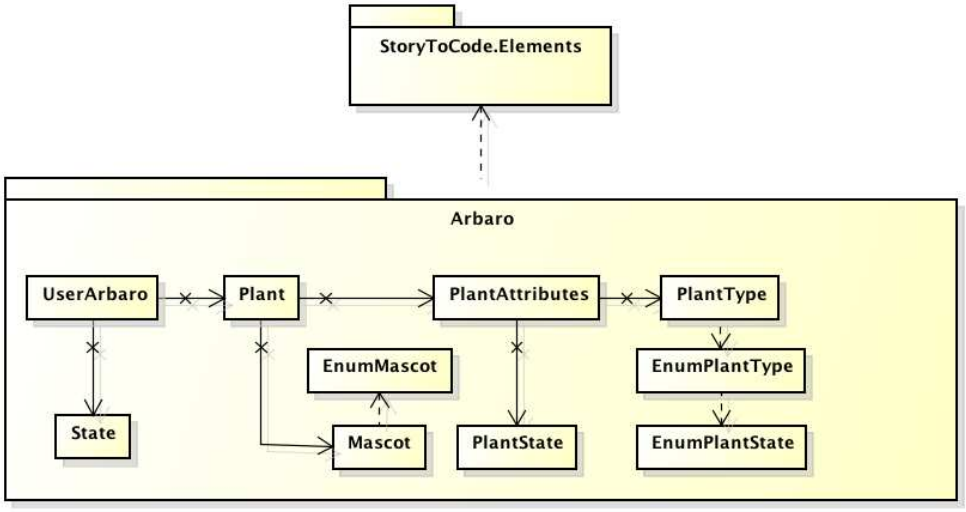

Figura 5.13. Diagrama simplificado do conjunto de componentes do projeto Árbaro

de projeto, o analista de sistemas e os programadores.

A partir dos storyboards/cenários produzidos na etapa anterior, esses profissionais produziram a lista de requisitos funcionais e não funcionais do jogo. Uma vez elaborada, a lista foi apresentada ao cliente e, depois de alguns ajustes, foi aprovada. A lista completa de requisitos funcionais e não funcionais pode ser vista no Apêndice **B**.

O próximo passo, ainda na etapa de projeto, foi definir o conjunto de componentes interativos de software. A responsabilidade desta definição foi do mesmo subgrupo de membros da equipe que trabalharam na lista de requisitos. Os componentes definidos usaram como base o conjunto de elementos abstratos do StoryToCode e são abstrações, tanto de elementos visuais do jogo quanto daqueles que encapsulam a lógica de negócio necessária. A figura 5.13 mostra um diagrama simplificado do conjunto de componentes.

Um dos problemas enfrentados durante o processo de modelagem dos componentes interativos foi definir como modelar o fluxo de interatividade e informações do jogo. O modelo StoryToCode não prevê a geração de artefatos que modelem esse fluxo e, diante disso, a equipe, inspirada em [135], recorreu aos diagramas de atividade da UML, conforme pode ser visto na figura 5.14.

5.2.2.3 Desenvolvimento

A etapa de desenvolvimento reuniu as atividades de geração automática de código e codificação complementar dos componentes modelados e foi a mais longa do estudo de caso (3 meses). A responsabilidade dessas atividades foi do analista e dos programadores.

O primeiro passo dessa etapa consistiu em escolher o contexto Web como alvo inicial da geração de código. Essa abordagem foi adotada de modo a maximizar o reuso de componentes interativos gerados para o contexto móvel, uma vez que ambos usam a linguagem Java. Dessa forma, a geração de código para TV Digital foi a última atividade dessa etapa. É importante

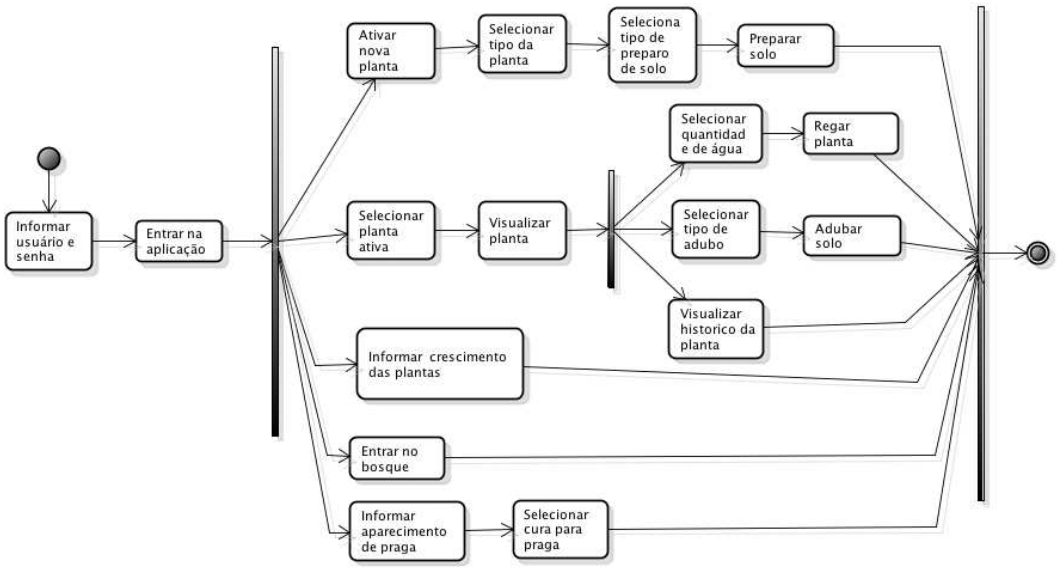

Figura 5.14. Diagrama de atividades do projeto Árbaro

ressaltar que a geração de código não foi feita em paralelo à atividade de codificação complementar. Isso significa que o código para um contexto foi gerado, e só depois de completado procedeu-se a geração para o contexto seguinte.

A dinâmica deste processo consistiu em submeter os elementos modelados em UML, na etapa de projeto, ao motor de transformação para código Java. Esse motor usa chamadas nativas do próprio ambiente usado na modelagem (o EMF) para realizar a geração de código. Uma vez que o código foi gerado, coube aos programadores a tarefa de complementá-lo. Depois de pronto, boa parte do código foi reaproveitado para o ambiente móvel.

Por fim, os elementos foram convertido em XMI e submetidos ao motor de transformação para código NCL. É importante ressaltar que esse motor não gera código Lua e por isso o trabalho de complementação, neste caso, foi bem maior quando comparado aos outros dois contextos.

Em todos os contextos foram necessários ajustes para permitir o uso das mídias produzidas pelos membros da equipe responsáveis pela criação artística. Esses profissionais realizaram, em paralelo à etapa de desenvolvimento, todo o trabalho de produção das mídia utilizadas. O conjunto dos principais *screenshots* do projeto Árbaro pode encontrado no Apêndice C.

5.2.3 Considerações

A equipe envolvida no projeto sugeriu que o uso do modelo obteve desempenho satisfatório. Porém, algumas dificuldades no entendimento dos requisitos para construção dos componentes interativos influenciaram diretamente a performance do uso do StoryToCode. Esse foi um problema constantemente apontado pelos participantes no intervalo das fases do estudo de caso.

A geração automática ajudou a diminuir a quantidade de trabalho na codificação do jogo,

mas a falta de infortmações de instância na aumentou o trabalho de recodificação.

Uma das maiores dificuldades encontradas pelos membros da equipe foi conseguir estruturar o conjunto de requisitos funcionais do jogo interativo. As informações contidas nos *storyboards/cenários* ajudaram nessa estruturação, mas seria pouco provável obter a lista de requisitos apenas com as demandas vindas do cliente e se o game designer e o consultor não fizessem parte da equipe. Apesar dessa dificuldade, todos os participantes foram unânimes em confirmar a hipótese que o modelo facilitou a comunicação entre os membros da equipe.

Um dos pontos fracos apontados pelos participantes foi que o processo de modelagem dos componentes interativos não prevê a estruturação dos fluxos de interatividade e de informações. No estudo de caso isso foi modelado a partir do diagrama de atividades da UML. Outra questão relacionada ao componentes interativos é a dificuldade que os membros sem experiência em engenharia de software tiveram para a compreensão desses componentes escritos em UML. Isso gerou atrasos na validação desses componentes e também algum "retrabalho" pela falta de consenso entre aquilo que o analista e o consultor entendiam como um componente interativo.

O código gerado para versão Web e para TV Digital necessitou de bastante "retrabalho" de complementação. O principal motivo percebido pelos participantes foi a ausência de informações de instância nos componentes modelados. Isso confirma que a hipótese "O número de linhas de código geradas representou uma diminuição importante de forma a refletir na diminuição da quantidade de trabalho realizado?" é falsa. Por outro lado, o reaproveitamento dos componentes codificados para a contexto móvel foi bastante significativo, uma vez que este contexto usa a mesma linguagem que o contexto Web e isso pode confirmar esta hipótese.

O objetivo do exemplo de uso foi avaliar o modelo proposto na prática e validar as hipóteses levantadas. Porém, é importante ressaltar que a avaliação foi feita a partir das impressões de uma equipe multidisciplinar pequena que atua em um dos possíveis campos da multimídia. O próximo exemplo de uso foi feito com base na evolução do StoryToCode (utilizando Arquétipos) e ajudou a minimizar algunsdos problemas encontrados no projeto Árbaro.

5.3 EXEMPLO DE USO: UTILIZANDO ARQUÉTIPOS

Para validar a abordagem de uso do StoryToCode baseada em arquétipos foi realizado um exemplo de uso no qual uma aplicação no domínio de TV Digital Interativa (TVDI) foi construída no LAVID - Laboratório de Aplicações de Vídeo Digital da Universidade Federal da Paraíba. As seguintes etapas foram executadas: (i) definição de uma classificação das aplicações em três categorias (*push, pull, hybrid*), 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.

Um fator importante para melhorar o processo de produção das aplicações para TVDI é diminuir o 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 propõe organizar as características das aplicações de acordo com uma estrutura que contemple os principais cenários de uso dos serviços de TVDI. Baseado em [136], as aplicações podem ser categorizadas em 3 categorias: *pull*, *push* e *hybrid*.

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 interaç?o 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: (i) “Cul-del-sac”: aplicações com narrativas lineares sem relação temporal com o conteúdo principal ; (ii) “TV Voting”: aplicações com interatividade através de canal indireto com a emissora (telefone, SMS, email, site Web etc.); (ii) “Add-On”: aplicações que complementam e são sincronizadas (dependência temporal) e enviadas paralelamente com o conteúdo principal, também conhecidas como “Enhanced TV” [137]; (iv) “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 consequência, 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 interaç?o 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: (i) "Cul-del-sac ReturnChannel"; (ii) TV Voting ReturnChannel; (iii) Add-On ReturnChannel e; (iv) Y App ReturnChannel.

Modo Hybrid: nesta categoria se encaixam aplicações mais recentes que exploram 2 (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 [137]. 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 [138]. Já para o segundo caso, em [139] é descrito o Emotional TV, que busca a interaç?o do usuário com a interpretação de emoções do telespectador através de uma bola sensitiva como dispositivo de entrada. Segundo [140], 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: (i) WS TVConsumer: consome informações de um serviço Web; (ii) WS TV Producer: produz informações para um serviço Web e (iii) WS TVProsumer: produz e consome informações para um serviço Web.

5.3.1 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, po-

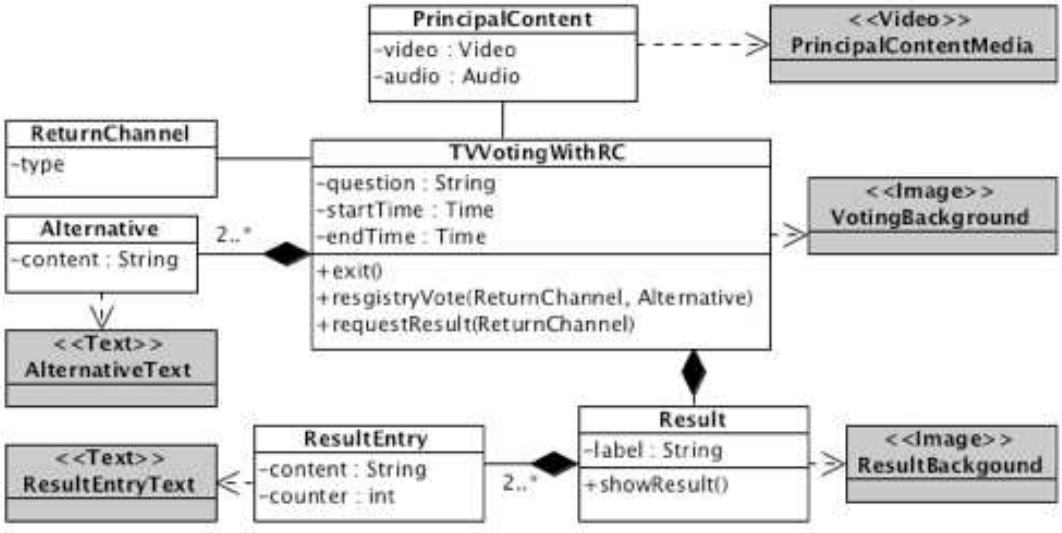

Figura 5.15. TV Voting RetunChannel - Modelo estrutural [7]

dem 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 [31].

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. Para simplificar, abaixo são descritos os dois últimos casos, já que o primeiro caso é um modelo simplificado do segundo.

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 5.15 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 estrutural permite associar os elementos de mídia (elementos *Video*, *Image* 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.

Para representar a integração dos elementos de mídia e a interação do usuário é utilizado o modelo de interface de mídia (Figura 5.16), onde uma entidade que deve expressar a confirmação do registro do voto (*TVVotingWithRC.registryVote*) deve ser exibido após um evento de

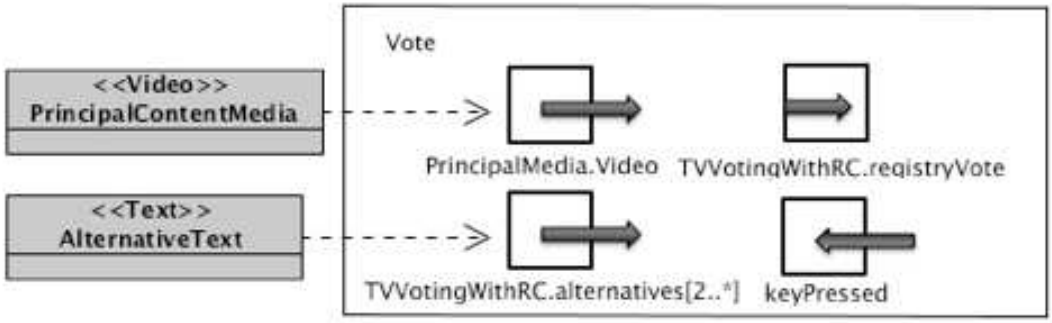

Figura 5.16. TVVotingWithRC - Modelo de Interface de Mídia [7]

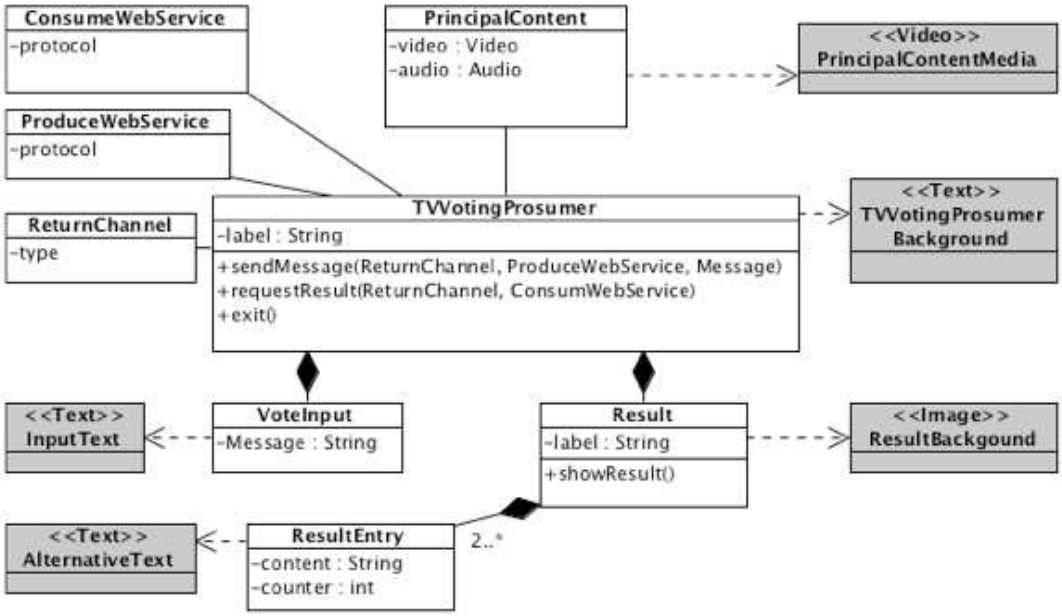

Figura 5.17. TVVoting Prosumer - Modelo estrutural [7]

tecla (*KeyPressed*) e escolha de alternativa(s) (*TVVotingWithRCAlternatives*).

O arquétipo das aplicações do tipo *TVVoting Prosumer* consiste em executar uma aplicação que envia e recebe informações de 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).

Na Figura 5.17, 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 5.18 evidencia o relacionamento dos elementos de interface gráfica com o fluxo

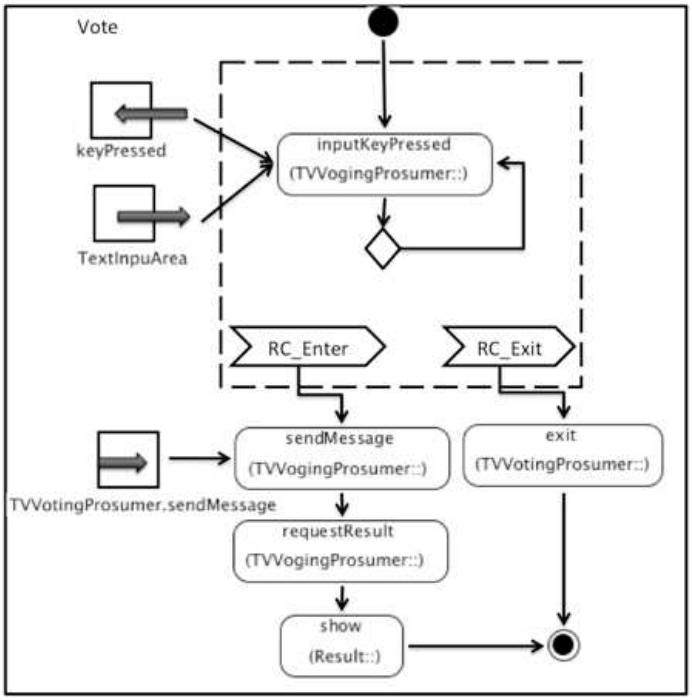

Figura 5.18. TVVoting Prosumer - Modelo de interação [7]

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 5.17) que exibe a resposta do resultado para o usuário.

5.3.2 Ferramenta e a Geração

A ferramenta utilizada na abordagem se baseia num trabalho proposto em [141], 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 [142].

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

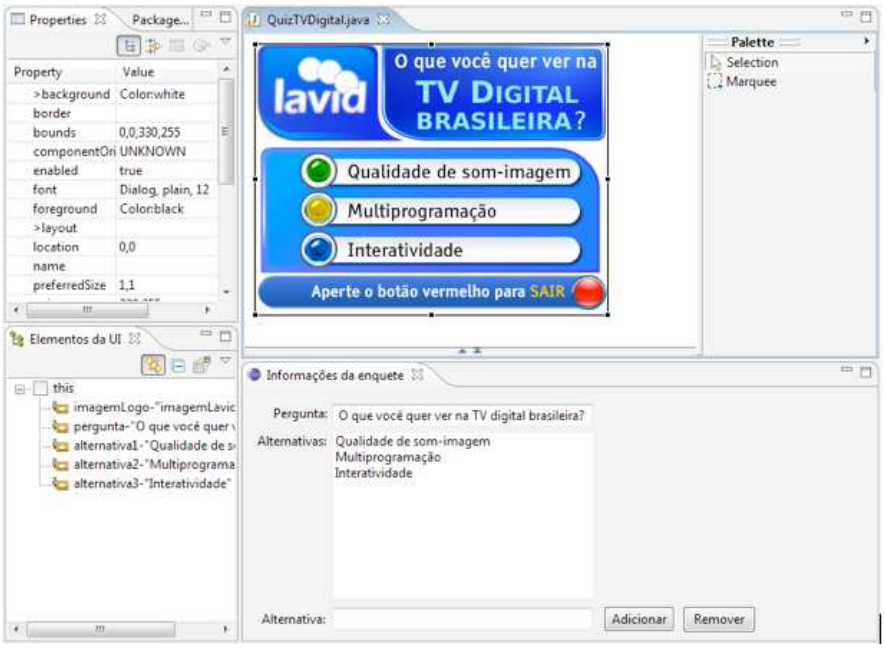

Figura 5.19. Ferramenta para edição visual dos arquétipos [7]

(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 de evolução do StoryToCode 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 outros trabalhos [127] [143]. Neste exemplo de uso 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 5.19: (i) editor gráfico; (ii) elementos da UI; (iii) barra de propriedades, (iv) informações e (v) editor de código.

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 os dados 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 de Elementos da Inter-

Figura 5.20. Arquétipo de TV Voting instanciado [7]

face 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, 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 na sua versão anterior. 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 3 (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 quase a mesma condição de um ambiente de produção.

A primeira aplicação definida foi uma Enquete em modo *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 5.20).

A segunda exemplifica uma aplicação de Enquete *push* e arquétipo TV Voting ReturnChannel (Figura 5.21), onde o canal de comunicação é direto (a resposta de cada usuário é enviada através de uma conexão à Internet direta para emissora e o resultado parcial é enviado individualmente para cada receptor).

A última aplicação permite uma competição de palpites para jogos de futebol (TV Bolão),

Figura 5.21. Arquétipo de TV Voting RC instanciado [7]

Figura 5.22. Arquétipo de TV Voting Prosumer instaciado [7]

representando o modo *hybrid* e o arquétipo TV Voting WSProsumer (Figura 5.22). 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.

5.3.3 Considerações

Este exemplo de uso avalia a 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 numa abordagem MDD para sistematizar a transformação de requisitos de Storyboard (CIM) em Arquétipos Abstratos (PIM) descritos em MML e, através de uma fer-

ramenta, 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.

Foram desenvolvidos exemplos de uso de três tipos de arquétipos (TV Voting, TV Voting with ReturnChannel e TV Voting with WSProsumer) para três aplicações diferentes de TVDI. O emprego da abordagem trouxe como principal benefício uma 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 finalizados a geração das aplicações herda todos os benefícios da autoria orientada a arquétipos.

Como trabalhos futuros, os seguintes pontos deverão ser tratados: (i) realização de outros experimentos para validar quantitativamente (por exemplo: velocidade, eficiência e simplicidade) a utilização da abordagem; (ii) extensão da linguagem de modelagem para tratar a semântica de máquinas sociais 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.

Capítulo 6

Este capítulo apresenta as considerações gerais sobre esta tese, apontando as principais contribuições, limitações e as possíveis perspectivas.

CONCLUSÕES

Depois da apresentação da motivação e dos objetivos feita no capítulo inicial, o segundo capítulo desta tese apresentou a revisão da literatura sobre aplicações multimídia interativas e apontou alguns dos principais desafios do seu desenvolvimento. Estes desafios constituem um conjunto de tarefas complexas que exigem conhecimentos altamente especializados em áreas como processamento gráfico, animação, som etc. Um desenvolvedor multimídia "completo" deve ser capaz de usar abordagens diferentes para lidar com as complexidades adicionais que estão presentes nessas aplicações e que não existem na programação de softwares tradicionais. Encontrar um profissional com esse nível de capacidade é muito difícil ou muito caro.

A solução adotada, na maioria dos casos, é montar uma equipe multidisciplinar e resolver de forma coordenada as diversas tarefas envolvidas no desenvolvimento de aplicações multimídia. Equipes multidisciplinares para desenvolvimento de aplicações multimídia podem ser encontradas em diversos ambientes, como: indústria de cinema, de jogos, emissoras de TV Digital, dentre outras. Apesar disso, as metodologias de desenvolvimento específicas para multimiìdia que contemplem este ambiente multidisciplinar ainda representam um desafio em aberto.

O terceiro capítulo apresentou as etapas de elaboração de um programa de TV convencional e discutiu a evolução necessária à metodologia tradicional para permitir a construção de programas para TV Digital. Especialmente para esse universo, o uso de uma metodologia específica é crucial para permitir migrar do modelo de concepção de um programa de TV convencional (que é sempre linear) para outro de TV Digital Interativo.

Com o surgimento da TV Digital, os processos produtivos de programas, que antes eram específicos do ambiente de TV, devem começar a agregar uma variedade de recursos oriundos de outros contextos e passam a exigir outras competências da equipe de produção. Esta tese buscou contribuir para este ambiente, através da criação do modelos que visam facilitar o desenvolvimento de aplicações multimídia interativas para TV Digital e outros contextos, que foram apresentados no quarto capítulo.

Ao longo dos anos, muitos trabalhos foram focados em tecnologias voltadas para o desenvolvimento de programas para TVDI. Por outro lado poucos trabalhos trataram o processo de desenvolvimento desse tipo específico de aplicação multimídia. Buscando preencher essas lacunas, o quarto capítulo apresentou as principais contribuições da tese: os modelo denominado IDTVS e sua evolução que recebeu o nome de StoryToCode.

Por fim, a quinto capítulo apresentou um estudo de caso que permitiu comprovar a viabilidade de uso do modelo StoryToCode em um universo multidisciplinar de produção de aplicações multimídia. O estudo de caso consistiu em criar um jogo digital, com foco em educação ambiental, que pode ser jogado em três plataformas: TV, Web e celulares.

6.1 CONTRIBUIÇÕES E TRABALHOS FUTUROS

A primeira contribuição desta tese foi um modelo (IDTVS) capaz de estruturar conteúdo multimídia em aplicações de TVDI, de forma a representar este conteúdo (vídeo e áudio, objetos do mundo real, que aparecem no vídeo e no áudio) e, ainda, suportar a representação do conteúdo em termos de interações entre os objetos e em termos de eventos. Ao longo do trabalho, este modelo evoluiu e passou também a definir a dinâmica de trasmissão, recepção, armazenamento dos dados e mídias trocadas entre as aplicações e os possíveis geradores de conteúdo.

Entre os contribuições específicas da tese associadas ao IDTVS podem-se destacar: (i) a definição formal do problema de sintonização tardia (Late Tuning) [132], (ii) a criação da dinâmica de transmissão de conteúdos que permite, entre outras, tratar o problema da sintonização tardia e (iii) a possibilidade de tratar eventos complexos (ex: associados a dinâmica do conteúdo) e não apenas aqueles associados às mídias apresentadas.

A evolução do IDTVS levou a outra contribuição: o modelo StoryToCode. Este modelo contribui para os desafios no desenvolvimento de aplicações multimídia, pois permite a integração das diferentes habilidades e competências existentes em equipes multidisciplinares, no âmbito da construção de aplicações interativas. O trabalho apresenta uma metodologia de construção dessas aplicações baseada em componentes, de maneira a permitir o reuso desses componentes em múltiplos contextos.

A estruturação de conteúdos proposta no StoryToCode foi inspirada na hierarquia de classes do *Nested Context Presentation Model* (NCPM) [6] e representa outra contribuição desta tese. O elemento *Application* do StoryToCode que pode ser estendido de forma a abstrair melhor tanto as interfaces visuais quanto a lógica de negócio de uma aplicação torna este modelo mais flexível que o NCPM.

O reaproveitamento dos componentes é feito a partir do uso de tranformações de modelos mais abstratos de uma aplicação 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 software,

além de não exigir dos participantes do processo produtivo oriundos de universos (ex: TV, Design de mídias etc.), um conhecimento específico na área de modelagem de sistemas.

Apesar de se mostrar adequado para a especificação e reuso de componentes interativos, o StoryToCode deixava dois problemas em aberto. O primeiro deles (**P1**) está associado a extração dos requisitos que são necessários para criação do conjunto de elementos. O segundo problema (**P2**) é a quantidade de retrabalho necessária para completar o código gerado automaticamente.

Conforme mencionado, no StoryToCode a obtenção dos requisitos a partir de Storyboards/-Cenários é uma atividade não automatizada e feita exclusivamente pela equipe de engenheiros de software. Automatizar este processo é ainda um desafio de custo computacional muito alto, pois significa implementar aspectos relativos a cognição humana no reconhecimento de informações. A abordagem adotada no StoryToCode é tradicionalmente usada para concepção de sistemas e define que a participação de profissionais ligados a outros ambientes, como o de TV, deve ocorrer apenas na validação (aprovação) dos requisitos levantados.

Essa situação caracteriza o problema **P1**: A apresentação dos requisitos sob a forma de um conjunto de elementos especificados em UML dificulta o entendimento dos profissionais que não possuem conhecimento de modelagem de sistemas no processo de validação. Isso provoca equívocos entre aquilo que os engenheiros de software e os outros profissionais entendem como um requisito para um componente interativo. Essa situação é semelhante aos encontrados na construção de interfaces gráficas para usuários de sistemas.

O problema **P2** também está associado ao uso de um conjunto de elementos para estruturar os requisitos de um programa interativo. No StoryToCode é impossível inserir informações de instância ao conjunto de elementos abstratos gerados na segunda etapa. A ausência dessas informações tem como consequência **P2**: aumento do (re)trabalho para completar o código gerado.

Por exemplo, um diagrama (conjunto de elementos) que represente um Quiz interativo descreve a estrutura de elementos como Pergunta e Resposta. Essa descrição pode informar que um elemento Pergunta contem um atributo que representa o texto de uma pergunta e outro atributo que representa uma lista de elementos Resposta. Porém, não existe, nesse diagrama, informações que definam quais são as perguntas e respostas (instâncias) do Quiz e nem como elas estão formatadas (fonte do texto, posição na tela, cor etc.). No StoryToCode essas informações, que são essenciais na geração do código, são introduzidas pela equipe de programação, depois da geração parcial do código, para posterior aprovação da equipe multidisciplinar.

A evolução do StoryToCode visou a solução dos problemas **P1** e **P2** através da inclusão de elementos como:

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;

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 software e aqueles de outras áreas.

Como trabalhos futuros, pode-se prever a criação de uma biblioteca de arquétipos para o desenvolvimento de aplicações interativas para programas de TV e outros contextos. Nesta biblioteca de aplicações prevê-se a inclusão de elementos comuns a uma série de aplicações voltadas para diferentes programas de TV, como também de elementos específicos encontrados em outros contextos. O uso dessa biblioteca, além de facilitar o processo de desenvolvimento de programas, permite abstrair e encapsular detalhes sobre os dados extras utilizados na aplicações ocultando a sua complexidade.

6.1.1 Publicações

Os resultados desta tese foram publicados em cinco eventos, conforme listados abaixo:

- An event-based model for interactive live TV shows, Proceeding of the 16th ACM international conference on Multimedia, October 26-31, 2008, Vancouver, British Columbia, Canada (Short paper);
- An approach based on events for treating the late tuning problem in interactive live TV shows, Proceedings of the 1st ACM international workshop on Events in multimedia, October 23-23, 2009, Beijing, China;
- StoryToCode: a model based on components for specifying interactive digital TV convergent applications, Proceedings of the XV Brazilian Symposium on Multimedia and the Web, p.1-8, October 05-07, 2009, Fortaleza, Ceará, Brazil (**Best Paper**)
- StoryToCode: a new model for specification of convergent interactive digital TV applications. Journal of the Brazilian Computer Society (Impresso), v. 16, p. 215-227, 2010.
- Uma abordagem dirigida por modelos para integração de aplicações interativas e serviços web: Estudo de caso na plataforma de tv digital. In: Sbc (Ed.). Proceedings: Webmedia 2011, Florianópolis, SC, Brasil. Porto Alegre, RS, Brasil: SBC, 2011. p. 1 a 8.

Além dessas publicações, o StoryToCode foi um dos tópicos de dois mini-cursos relacionados ao desenvolvimento de aplicações para TVDI: Desenvolvimento de Aplicações Imperativas para TV Digital no Middleware Ginga com Java (WebMedia 2010) e Software Development for the Brazilian Digital TV Platform (CBSoft 2010).

REFERÊNCIAS BIBLIOGRÁFICAS

[1] PAGANI, M. *Multimedia and Interactive Digital TV: Managing the Opportunities Created by Digital Convergence*. Hershey, PA, USA: IGI Publishing, 2003. ISBN 1931777381.

[2] RIOS, J. M. M.; PATACA, D. M.; MARQUES, M. C. *Panorama Mundial de Modelos de Exploração e Implantação - Projeto Sistema Brasileiro de Televisão Digital, CPQD*. 2004.

[3] LULA, M. M. M.; GUIMARãES, A. P.; FILHO, G. L. de S.; TAVARES, T. A. Boas práticas para o desenvolvimento de programas interativos para tv digital. In: *ASSE 2011: 12th Argentine Symposium on Software Engineering*. Cordoba, Argentina: Procedings, 2011.

[4] LULA, M. M. M.; GUIMARãES, A. P.; TAVARES, T. A. Um processo Ágil para especificação de requisitos em programas interativos com foco em roteiros de tv. In: *WDRA 2011: 5th Workshop de Desenvolvimento Rápido de Aplicações*. Porto Alegre, Brasil: SBC, 2011.

[5] VEIGA, E. G.; TAVARES, T. A. Um modelo de processo para o desenvolvimento de programas para tv digital e interativa. In: *WebMedia '2006: Workshop de Teses e Dissertações*. New York, NY, USA: ACM, 2006. p. 53–57. ISBN 85-7669-100-0.

[6] SOARES, L. F. G.; RODRIGUES, R. F.; SAADE, D. C. M. Modeling, authoring and formatting hypermedia documents in the hyperprop system. *Multimedia Syst.*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 8, n. 2, p. 118–134, 2000. ISSN 0942-4962.

[7] KULESZA, R.; MEIRA, S. R. L.; FERREIRA, T. P.; LíVIO, Á.; FILHO, G. L. S. Uma abordagem dirigida por modelos para integraçao de aplicaçoes interativas e serviços web: Estudo de caso na plataforma de tv digital. In: SBC (Ed.). *Proceedings: Webmedia 2011, Florianópoles, SC, Brasil*. Porto Alegre, RS, Brasil: SBC, 2011. p. 1–8.

[8] VILLAR, A. C. *Um Ambiente de Simulação para Aplicações Dinâmicas de TV Digital com Transmissão ao Vivo*. Dissertação de Mestrado, Universidade Salvador, 2010.

[9] RODRIGUES, R. F.; SOARES, L. F. Produção de conteúdo declarativo para tv digital. In: *SemiSH*. [S.l.: s.n.], 2006.

[10] COSTA, R. M. de R.; RODRIGUES, R. F.; SOARES, L. F. G. New facilities for presenting and authoring mpeg-4 documents. In: *MMM*. [S.l.: s.n.], 2006.

[11] NETO, C. de S. S.; SOUZA, C. S. de; SOARES, L. F. G. Linguagens computacionais como interfaces: um estudo com nested context language. In: *IHC '08: Proceedings of the VIII Brazilian Symposium on Human Factors in Computing Systems*. Porto Alegre, Brazil, Brazil: Sociedade Brasileira de Computaç ao, 2008. p. 166–175. ISBN 978-85-7669-203-4.

[12] GUIMARãES, R. L.; COSTA, R. M. D. R.; SOARES, L. F. G. Composer: Authoring tool for itv programs. In: *Proceedings of the 6th European conference on Changing Television Environments*. Berlin, Heidelberg: Springer-Verlag, 2008. (EUROITV '08), p. 61–71. ISBN 978-3-540-69477-9.

[13] AZEVEDO, R. G. A.; NETO, C. d. S. S.; TEIXEIRA, M. M.; SANTOS, R. C. M.; GOMES, T. A. Textual authoring of interactive digital tv applications. In: *Proceddings of the 9th international interactive conference on Interactive television*. New York, NY, USA: ACM, 2011. (EuroITV '11), p. 235–244. ISBN 978-1-4503-0602-7. Disponível em: http://doi.acm.org/10.1145/2000119.2000169.

[14] SANTOS, A. L. d. S.; REIS, E. M. dos; VIRGENS, L. S. das; GOMES, E. A.; NETO, M. C. M. Plug-in saga - editor visual de aplicações interativas para tv digital baseado no middleware ginga. In: *IX Escola Regional de Computação Bahia - Alagoas-Sergipe*. [S.l.]: SBC, 2009. p. 1–6.

[15] HATTORI, L. P.; SILVA, S. S.; TAVARES, T. A.; NETO, M. C. M.; SAIBEL, C. Utilizando o framework apptv no desenvolvimento de aplicações para tv digital interativa. In: *Workshop de TV Digital e Interativa. Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)*. [S.l.: s.n.], 2005. p. 1–6.

[16] HATTORI, L. P.; SILVA, S. S.; TAVARES, T. A.; NETO, M. C. M. Apptv: Um framework de auxílio ao desenvolvimento de aplicações para tv digital. In: *WebMedia/LA-Web 2005*. [S.l.: s.n.], 2005. p. 287–293.

[17] GUIMARãES, R. L.; NETO, C. de S. S.; SOARES, L. F. G. A visual approach for modeling spatiotemporal relations. In: *DocEng '08: Proceeding of the eighth ACM symposium on Document engineering*. New York, NY, USA: ACM, 2008. p. 285–288. ISBN 978-1-60558-081-4.

[18] SOARES, L. F. G.; MORENO, M. F.; SANT'ANNA, F. Relating declarative hypermedia objects and imperative objects through the ncl glue language. In: *DocEng '09: Proceedings of the 9th ACM symposium on Document engineering*. New York, NY, USA: ACM, 2009. p. 222–230. ISBN 978-1-60558-575-8.

[19] SOARES, L. F. G.; COSTA, R. M. R.; MORENO, M. F.; MORENO, M. F. Multiple exhibition devices in dtv systems. In: *ACM Multimedia*. [S.l.: s.n.], 2009. p. 281–290.

[20] MORENO, M. F.; NETO, C. de S. S.; SOARES, L. F. G. Adaptable software components in an electronic program/service guide application architecture for context aware guide presentation. *IJAMC*, v. 3, n. 4, p. 351–364, 2009.

[21] CHORIANOPOULOS, K. *Virtual television channels: Conceptual model, user interface design and affective usability evaluation*. 198 p. Tese (PhD) — Athens University Economics and Business (AUEB), 2004. Disponível em: http://phdtheses.ekt.gr/eadd/handle/10442/17648.

[22] GAWLINSKI, M. *Interactive television production*. Focal, 2003. ISBN 9780240516790. Disponível em: http://books.google.com.br/books?id=TwKfJj9U3V8C.

[23] ENGELS, G.; SAUER, S. *Object-oriented Modeling of Multimedia Applications in Handbook of software engineering & knowledge engineering*. World Scientific, 2002. 21–53 p. (Handbook of Software Engineering & Knowledge Engineering). ISBN 9789810249748. Disponível em: http://books.google.com.br/books?id=bkGC3TYJKU0C.

[24] CHANG, H.-Y.; WU, S. F.; JOU, Y. F. Real-time protocol analysis for detecting link-state routing protocol attacks. *ACM Trans. Inf. Syst. Secur.*, ACM, New York, NY, USA, v. 4, p. 1–36, February 2001. ISSN 1094-9224. Disponível em: http://doi.acm.org/10.1145/383775.383776.

[25] TERRAZAS, A.; BARLOW, M.; TERRAZAS, D. A.; OSTUNI, J. *Java Media APIs: Cross-Platform Imaging, Media and Visualization*. [S.l.]: Pearson Education, 2002. ISBN 0672320940.

[26] INTERVOZES. *TV Digital: Princípios e Propostas para uma Transição Baseada no Interesse Público*. August 2006. Www.Intervozes.org.br/publicacoes/documentos/TVDigital.pdf.

[27] BROWN, A. W. *Large-Scale, Component Based Development*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000. ISBN 013088720X.

[28] SHIMOMURA, T.; IKEDA, K.; CHEN, Q. L.; LANG, N. S.; TAKAHASHI, M. Visual pivot-table components for web application development. In: *Proceedings of the third conference on IASTED International Conference: Advances in Computer Science and Technology*. Anaheim, CA, USA: ACTA Press, 2007. (ACST'07), p. 90–95. Disponível em: http://dl.acm.org/citation.cfm?id=1322468.1322483.

[29] STEINBERG, D.; BUDINSKY, F.; PATERNOSTRO, M.; MERKS, E.; GRONBACK, R. C.; MILINKOVICH, M. *EMF: Eclipse Modelign Framework*. 2nd. ed. Upper Saddle River, NJ: Addison-Wesley, 2009. (The eclipse series).

[30] KLEPPE, A. G.; WARMER, J.; BAST, W. *MDA Explained: The Model Driven Architecture: Practice and Promise*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003. ISBN 032119442X.

[31] NETO, C. S. S.; SOARES, L.; SOUZA, C. de. Tal-linguagem para autoria de arquétipos de documentos hipermídia. In: *Proceedings of In Webmedia 2010*. Belo Horizonte: SBC, 2010. p. 147–154.

[32] REISMAN, S. Multimedia is dead. *IEEE MultiMedia*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 5, p. 4–5, January 1998. ISSN 1070-986X. Disponível em: http://dl.acm.org/citation.cfm?id=614657.614818.

[33] HIRAKAWA, M. Do software engineers like multimedia? In: *Proceedings of the IEEE International Conference on Multimedia Computing and Systems - Volume 2*. Washington, DC, USA: IEEE Computer Society, 1999. (ICMCS '99), p. 9085–. ISBN 0-7695-0253-9. Disponível em: http://dx.doi.org/10.1109/MMCS.1999.779125.

[34] GONZALEZ, R. Disciplining multimedia. *IEEE MultiMedia*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 7, p. 72–78, July 2000. ISSN 1070-986X. Disponível em: http://dx.doi.org/10.1109/93.879770.

[35] MüHLHäUSER, M.; GECSEI, J. Services, frameworks, and paradigms for distributed multimedia applications. *IEEE MultiMedia*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 3, p. 48–61, September 1996. ISSN 1070-986X. Disponível em: http://dx.doi.org/10.1109/93.556460.

[36] BULTERMAN, D. C. A.; HARDMAN, L. Structured multimedia authoring. *ACM Trans. Multimedia Comput. Commun. Appl.*, ACM, New York, NY, USA, v. 1, p. 89–109, February 2005. ISSN 1551-6857. Disponível em: http://doi.acm.org/10.1145/1047936.1047943.

[37] CHAVES, E. *Multimídia : conceituação, aplicações e tecnologia*. Campinas, SP, Brasil: People Computação Ltda, 1991. ISBN 9788573024876.

[38] MEYER-BOUDNIK, T.; EFFELSBERG, W. Mheg explained. *IEEE MultiMedia*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 2, p. 26–38, March 1995. ISSN 1070-986X. Disponível em: http://dx.doi.org/10.1109/93.368598.

[39] STEINMETZ, R. *Multimedia-Technologie*. Springer, 2000. ISBN 9783540673323. Disponível em: http://books.google.de/books?id=jXRenVdVG_cC.

[40] RAISAMO, R.; PATOMäKI, S.; HASU, M.; PASTO, V. Design and evaluation of a tactile memory game for visually impaired children. *Interact. Comput.*, Elsevier Science Inc., New York, NY, USA, v. 19, p. 196–205, March 2007. ISSN 0953-5438. Disponível em: http://dl.acm.org/citation.cfm?id=1224795.1224828.

[41] CHARBONNEAU, E.; HUGHES, C. E.; LAVIOLA JR., J. J. Vibraudio pose: an investigation of non-visual feedback roles for body controlled video games. In: *Proceedings of the 5th ACM SIGGRAPH Symposium on Video Games*. New York, NY, USA: ACM, 2010. (Sandbox '10), p. 79–84. ISBN 978-1-4503-0097-1. Disponível em: http://doi.acm.org/10.1145/1836135.1836147.

[42] PAREKH, R. *Principles of multimedia*. Tata McGraw-Hill, 2006. ISBN 9780070588332. Disponível em: http://books.google.com.br/books?id=TaNmc2IdNVwC.

[43] STEINMETZ, R.; NAHRSTEDT, K. *Multimedia applications*. Springer, 2004. (X.MEDIA.PUBLISHING Series). ISBN 9783540408499. Disponível em: http://books.google.com.br/books?id=Zw8h5BrEEKMC.

[44] BULTERMAN, D.; ROSSUM, G. V.; LIERE, R. V. A structure for transportable, dynamic multimedia documents. In: *In Proc. of the Summer 1991 USENIX Conference*. [S.l.: s.n.], 1991. p. 137–155.

[45] BOLL, S.; KLAS, W. Zyx-a multimedia document model for reuse and adaptation of multimedia content. *IEEE Trans. on Knowl. and Data Eng.*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 13, p. 361–382, May 2001. ISSN 1041-4347. Disponível em: http://dx.doi.org/10.1109/69.929895.

[46] SCHULMEISTER, R. Taxonomy of multimedia component interactivity - a contribution to the current metadata debate. *Studies in Communication Sciences*, Facolta, Lugano, n. Special, p. 61–80, 2003. ISSN 1424-4896.

[47] HOOGEVEEN, M. Toward a theory of the effectiveness of multimedia systems. *International Journal of Human-Computer Interaction*, Lawrence Erlbaum, v. 9, n. 2, p. 151–168, 1997.

[48] SUNDAR, S. S. Theorizing interactivity's effects. *The Information Society*, v. 20, n. 5, p. 385–389, 2004.

[49] DOSPISIL, J.; POLGAR, T. Conceptual modelling in the hypermedia development process. In: *Proceedings of the 1994 computer personnel research conference on Reinventing IS : managing information technology in changing organizations: managing information technology in changing organizations*. New York, NY, USA: ACM, 1994. (SIGCPR '94), p. 97–104. ISBN 0-89791-652-2. Disponível em: http://doi.acm.org/10.1145/186281.186299.

[50] SUTCLIFFE, A.; ZIEGLER, J.; JOHNSON, P.; PROCESSING, I. F. for I. *Designing effective and usable multimedia systems: proceedings of the IFIP Working Group 13.2 Conference on Designing Effective and Usable Multimedia Systems, Stuttgart, Germany, September 1998*. Kluwer Academic Publishers, 1998. (IFIP International Federation for Information Processing Series). ISBN 9780412842702. Disponível em: http://books.google.com.br/books?id=GhhAu2mhnc4C.

[51] ARNDT, T. The evolving role of software engineering in the production of multimedia applications. In: *Proceedings of the IEEE International Conference on Multimedia Computing and Systems - Volume 2*. Washington, DC, USA: IEEE Computer Society, 1999. (ICMCS '99), p. 9079–. ISBN 0-7695-0253-9. Disponível em: http://dx.doi.org/10.1109/MMCS.1999.779124.

[52] ROUT, T. P.; SHERWOOD, C. Software engineering standards and the development of multimedia-based systems. In: *Proceedings of the 4th IEEE International Symposium and Forum on Software Engineering Standards*. Washington, DC, USA: IEEE Computer Society, 1999. p. 192–. ISBN 0-7695-0068-4. Disponível em: http://dl.acm.org/citation.cfm?id=520047.854873.

[53] AEDO, I.; DÍAZ, P. Applying software engineering methods for hypermedia systems. *SIGCSE Bull.*, ACM, New York, NY, USA, v. 33, p. 5–8, June 2001. ISSN 0097-8418. Disponível em: http://doi.acm.org/10.1145/507758.377442.

[54] GOLDMAN, D. B.; CURLESS, B.; SALESIN, D.; SEITZ, S. M. Schematic storyboarding for video visualization and editing. *ACM Trans. Graph.*, ACM, New York, NY, USA, v. 25, p. 862–871, July 2006. ISSN 0730-0301. Disponível em: http://doi.acm.org/10.1145/1141911.1141967.

[55] MUSBURGER, R.; KINDEM, G. *Introduction to Media Production: The Path to Digital Media Production*. Elsevier Science, 2009. ISBN 9780240810829. Disponível em: http://books.google.co.in/books?id=10KzsPD0xMUC.

[56] NACK, F. Capture and transfer of metadata during video production. In: *Proceedings of the ACM workshop on Multimedia for human communication: from capture to convey*. New York, NY, USA: ACM, 2005. (MHC '05), p. 17–20. ISBN 1-59593-247-X. Disponível em: http://doi.acm.org/10.1145/1099376.1099382.

[57] HARDMAN, L.; OBRENOVIć, v.; NACK, F.; KERHERVé, B.; PIERSOL, K. Canonical processes of semantically annotated media production. *Multimedia Systems*, Springer Berlin / Heidelberg, v. 14, n. 6, p. 327–340, dez. 2008. ISSN 0942-4962. Disponível em: http://dx.doi.org/10.1007/s00530-008-0134-0.

[58] CESAR, P.; BULTERMAN, D. C. A.; JANSEN, J.; GEERTS, D.; KNOCHE, H.; SEAGER, W. Fragment, tag, enrich, and send: Enhancing social sharing of video. *ACM Trans. Multimedia Comput. Commun. Appl.*, ACM, New York, NY, USA, v. 5, p. 19:1–19:27, August 2009. ISSN 1551-6857.

[59] LOWE, D. *Hypermedia and the Web: An Engineering Approach*. New York, NY, USA: John Wiley & Sons, Inc., 1999. ISBN 0471983128.

[60] LANG, M.; FITZGERALD, B. New branches, old roots: A study of methods and techniques in web/hypermedia systems design. *IS Management*, v. 23, n. 3, p. 62–74, 2006.

[61] TANNENBAUM, R. S. Theoretical foundations of multimedia. *Ubiquity*, ACM, New York, NY, USA, v. 2000, August 2000. ISSN 1530-2180. Disponível em: http://doi.acm.org/10.1145/347634.347643.

[62] LANG, M. Hypermedia systems development: Do we really need new methods? Cork, Ireland, p. 883–891, June 2002.

[63] SOMMERVILLE, I. *Software Engineering (9th Edition)*. [S.l.]: Pearson Addison Wesley, 2010. ISBN 0137035152.

[64] DIX, A.; FINLAY, J. E.; ABOWD, G. D.; BEALE, R. *Human-Computer Interaction (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2003. ISBN 0130461091.

[65] SHNEIDERMAN, B.; PLAISANT, C.; COHEN, M.; JACOBS, S. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. 5th. ed. USA: Addison-Wesley Publishing Company, 2009. ISBN 9780321537355.

[66] ANDLEIGH, P.; THAKRAR, K. *Multimedia systems design*. Prentice Hall PTR, 1996. ISBN 9780130890955. Disponível em: http://books.google.com/books?id=dZ5QAAAAMAAJ.

[67] RIZVI, S. S.; CHUNG, T.-S. A reliable storage management framework for flash based embedded and multimedia systems experiencing diverse data nature. In: *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication*. New York, NY, USA: ACM, 2011. (ICUIMC '11), p. 128:1–128:4. ISBN 978-1-4503-0571-6. Disponível em: http://doi.acm.org/10.1145/1968613.1968761.

[68] BLAKOWSKI, G.; STEINMETZ, R. A Media Synchronization Survey: Reference Model, Specification, and Case Studies. *IEEE Journal on Selected Areas in Communications*, v. 14, n. 1, p. 5–35, 1996.

[69] BLAIR, G.; BLAIR, L.; CHETWYND, A.; BOWMAN, H. *Formal Specifications of Distributed Multimedia Systems*. Bristol, PA, USA: Taylor & Francis, Inc., 1997. ISBN 1857286774.

[70] COURTIAT, J.-P.; OLIVEIRA, R. C. D. Proving temporal consistency in a new multimedia synchronization model. In: *Proceedings of the fourth ACM international conference on Multimedia*. New York, NY, USA: ACM, 1996. (MULTIMEDIA '96), p. 141–152. ISBN 0-89791-871-1. Disponível em: http://doi.acm.org/10.1145/244130.244178.

[71] HUSSAIN, Z.; SLANY, W.; HOLZINGER, A. Current state of agile user-centered design: A survey. In: *Proceedings of the 5th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society on HCI and Usability for e-Inclusion*. Berlin, Heidelberg: Springer-Verlag, 2009. (USAB '09), p. 416–427. ISBN 978-3-642-10307-0. Disponível em: http://dx.doi.org/10.1007/978-3-642-10308-7-30.

[72] SHAHZAD, S. Learning from experience: The analysis of an extreme programming process. In: *Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations*. Washington, DC, USA: IEEE Computer Society, 2009. p. 1405–1410. ISBN 978-0-7695-3596-8. Disponível em: http://dl.acm.org/citation.cfm?id=1588296.1588848.

[73] JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. *The unified software development process*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999. ISBN 0-201-57169-2.

[74] MEDVIDOVIC, N.; ROSENBLUM, D. S.; REDMILES, D. F.; ROBBINS, J. E. Modeling software architectures in the unified modeling language. *ACM Trans. Softw. Eng. Methodol.*, ACM, New York, NY, USA, v. 11, p. 2–57, January 2002. ISSN 1049-331X. Disponível em: http://doi.acm.org/10.1145/504087.504088.

[75] CHAM, T. *Advances in multimedia modeling: 13th International Multimedia Modeling Conference, MMM 2007, Singapore, January 9-12, 2007 : proceedings*. Springer, 2007. (Lecture notes in computer science). ISBN 9783540694212. Disponível em: http://books.google.com.br/books?id=WutBD4rxo7AC.

[76] HUSSMANN, H.; MEIXNER, G.; ZUEHLKE, D. *Model-Driven Development of Advanced User Interfaces*. Springer, 2011. (Studies in Computational Intelligence). ISBN 9783642145612. Disponível em: http://books.google.com.br/books?id=GADoPzJCAGsC.

[77] BONASIO, V. *Televisão:Manual de Pradução & Direção*. Leitura, 2002. 408 p. ISBN 9788573584783. Disponível em: http://books.google.com.br/books?id=_crGcQAACAAJ.

[78] LEITE, L. E. C.; FILHO, G. L. de S.; MEIRA, S. R. de L.; ARúJO, P. C. T. de; LIMA, J. F. de A.; FILHO, S. M. A component model proposal for embedded systems and its use to add reconfiguration capabilities to the flextv middleware. In: *WebMedia '06*. New York, NY, USA: ACM, 2006. p. 203–212. ISBN 85-7669-100-0.

[79] SCHWALB, E. itv handbook: technologies & standards. *Comput. Entertain.*, ACM, New York, NY, USA, v. 2, p. 17–17, abr. 2004. ISSN 1544-3574. Disponível em: http://doi.acm.org/10.1145/1008213.1008241.

[80] WIEGAND, T.; SULLIVAN, G. J.; BJONTEGAARD, G.; LUTHRA, A. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 13, n. 7, p. 560–576, jul. 2003. ISSN 1051-8215. Disponível em: http://dx.doi.org/10.1109/TCSVT.2003.815165.

[81] MARPE, D.; WIEGAND, T.; SULLIVAN, G. J. The H.264/MPEG4 Advanced Video Coding Standard and its Applications. *IEEE Communications*, v. 44, n. 8, p. 134–143, ago. 2006. Disponível em: http://iphome.hhi.de/marpe/download/commag06.pdf.

[82] JEON, J. H.; KIM, H. S.; BOO, G. N.; SONG, J. S.; LEE, E. W.; PARK, H. W. Real-time mpeg-2 video codec system using multiple digital signal processors. *Multimedia Tools Appl.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 11, p. 197–214, June 2000. ISSN 1380-7501. Disponível em: http://dl.acm.org/citation.cfm?id=597026.597154.

[83] SOARES, L. F. G.; RODRIGUES, R. F.; MORENO, M. F. Ginga-ncl: the declarative environment of the brazilian digital tv system. In: *Journal of the Brazilian Computer Society, v. 12*. [S.l.]: SBC, 2007. p. 37–46.

[84] FILHO, G. L. de S.; LEITE, L. E. C.; BATISTA, C. E. C. F. Ginga-j: The procedural middleware for the brazilian digital tv system. *J. Braz. Comp. Soc.*, v. 13, n. 1, p. 47–56, 2007.

[85] ITU-T. *Recommendation J.200: Worldwide common core - Application environment for digital interactive television services*. jun. 2011. Disponível em: http://www.itu.int/en/Pages/default.aspx.

[86] STANDARDIZATION, I. O. for; COMMISSION, I. E. *Information technology – Generic coding of moving pictures and associated audio information –: Extensions for DSM-CC. TECHNICAL CORRIGENDUM 2. Extensions pour DSM-CC. RECTIFICATIF TECHNIQUE 2*. ISO/IEC, 2002. (International standard, pt. 6). Disponível em: http://books.google.com.br/books?id=pU4_PwAACAAJ.

[87] NETO, M. C. M.; SANTOS, C. A. An event-based model for interactive live tv shows. In: *MM '08: Proceeding of the 16th ACM international conference on Multimedia*. New York, NY, USA: ACM, 2008. p. 845–848. ISBN 978-1-60558-303-7.

[88] AMOR, M.; FUENTES, L.; PINTO, M. A survey of multimedia software engineering. *Journal of Universal Computer Science*, v. 10, n. 4, p. 473–498, apr 2004.

[89] BEDERSON, B. B.; GROSJEAN, J.; MEYER, J. Toolkit design for interactive structured graphics. *IEEE Trans. Softw. Eng.*, IEEE Press, Piscataway, NJ, USA, v. 30, p. 535–546, August 2004. ISSN 0098-5589. Disponível em: http://dl.acm.org/citation.cfm?id=1018036.1018385.

[90] SOPER, M. E. *Unleashing Microsoft Windows Vista Media Center*. [S.l.]: Que Publishing Company, 2008. ISBN 0789736713, 9780789736710.

[91] MILLER, F. P.; VANDOME, A. F.; MCBREWSTER, J. *DirectShow: Open source, DirectX Media Objects, Media Foundation, DirectX plugin, DirectX Video Acceleration, Multimedia framework,*

Application programming interface, Microsoft, Software developer. [S.l.]: Alpha Press, 2009. ISBN 6130256736, 9786130256739.

[92] PINTO, M.; AMOR, M.; FUENTES, L.; TROYA, J. M. Analyzing architectural evolution issues of multimedia frameworks. *Multimedia Tools Appl.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 22, p. 31–51, January 2004. ISSN 1380-7501. Disponível em: http://dl.acm.org/citation.cfm?id=960204.960230.

[93] QU, C.; NEJDL, W. Constructing a web-based asynchronous and synchronous collaboration environment using webdav and lotus sametime. In: *Proceedings of the 29th annual ACM SIGUCCS conference on User services*. New York, NY, USA: ACM, 2001. (SIGUCCS '01), p. 142–149. ISBN 1-58113-382-0. Disponível em: http://doi.acm.org/10.1145/500956.500991.

[94] CASAMAYOR, A.; AMANDI, A.; CAMPO, M. Intelligent assistance for teachers in collaborative e-learning environments. *Comput. Educ.*, Elsevier Science Ltd., Oxford, UK, UK, v. 53, p. 1147–1154, December 2009. ISSN 0360-1315. Disponível em: http://dl.acm.org/citation.cfm?id=1613342.1613556.

[95] TAPADIYA, P. *Com+ Programming: A Practical Guide Using Visual C++ and Atl*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000. ISBN 0130886742.

[96] SHERROD, A. *Ultimate Game Programming With DirectX*. Rockland, MA, USA: Charles River Media, Inc., 2006. ISBN 1584504587.

[97] LUNA, F. D. *Introduction to 3D Game Programming with Direct 3D 10: A Shader Approach*. Plano, TX, USA: Wordware Publishing Inc., 2008. ISBN 1598220535, 9781598220537.

[98] BULTERMAN, D.; JANSEN, J.; CESAR, P.; MULLENDER, S.; DEMEGLIO, M.; QUINT, J.; VUORIMAA, P.; CRUZ-LARA, S.; KAWAMURA, H.; WECK, D.; HYCHE, E.; PAÑEDA, X. G.; MELENDI, D.; MICHEL, T.; ZUCKER, D. F. *Synchronized Multimedia Integration Language (SMIL 3.0)*. December 2008. World Wide Web Consortium, Recommendation REC-SMIL3-20081201.

[99] MARCHISIO, C.; MARCHISIO, P. Mediatouch: A native authoring tool for mheg-5 applications. *Multimedia Tools Appl.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 14, p. 5–22, May 2001. ISSN 1380-7501. Disponível em: http://dl.acm.org/citation.cfm?id=597033.597192.

[100] SOARES, L. F.; G.; RODRIGUEZ, N. L. R.; CASANOVA, M. A. Nested composite nodes and version control in an open hypermedia system. *Inf. Syst.*, Elsevier Science Ltd., Oxford, UK, UK, v. 20, p. 501–519, September 1995. ISSN 0306-4379. Disponível em: http://dl.acm.org/citation.cfm?id=214228.214233.

[101] COSTA, R. M. de R.; MORENO, M. F.; RODRIGUES, R. F.; SOARES, L. F. G. Live editing of hypermedia documents. In: *Proceedings of the 2006 ACM symposium on Document engineering*. New York, NY, USA: ACM, 2006. (DocEng '06), p. 165–172. ISBN 1-59593-515-0. Disponível em: http://doi.acm.org/10.1145/1166160.1166202.

[102] MASCARENHAS, F.; IERUSALIMSCHY, R. Efficient compilation of lua for the clr. In: *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*. New York, NY, USA: ACM, 2008. p. 217–221. ISBN 978-1-59593-753-7.

[103] SANT'ANNA, F.; SALLES, C. Programando em ncl 3.0. In: ____. [S.l.: s.n.], 2009. cap. 18-Programando com Objetos NCLua.

[104] ARNDT, T.; KATZ, E. Visual software tools for multimedia authoring. *J. Vis. Lang. Comput.*, Academic Press, Inc., Orlando, FL, USA, v. 21, p. 184–191, June 2010. ISSN 1045-926X. Disponível em: http://dx.doi.org/10.1016/j.jvlc.2009.12.005.

[105] BRITTON, C.; JONES, S.; MYERS, M.; SHARIF, M. A survey of current practice in the development of multimedia systems. *Information & Software Technology*, v. 39, n. 10, p. 695–705, 1997.

[106] HANNINGTON, A.; REED, K. Towards a taxonomy for guiding multimedia application development. In: *Proceedings of the Ninth Asia-Pacific Software Engineering Conference*. Washington, DC, USA: IEEE Computer Society, 2002. (APSEC '02), p. 97–. ISBN 0-7695-1850-8. Disponível em: http://dl.acm.org/citation.cfm?id=785409.785852.

[107] HANNINGTON, A.; REED, K. Factors in multimedia project and process management–australian survey findings. In: *Proceedings of the 2007 Australian Software Engineering Conference*. Washington, DC, USA: IEEE Computer Society, 2007. p. 379–388. ISBN 0-7695-2778-7. Disponível em: http://dl.acm.org/citation.cfm?id=1249255.1250600.

[108] WILLRICH, R.; SAQUI-SANNES, P. D.; SéNAC, P.; DIAZ, M. Multimedia authoring with hierarchical timed stream petri nets and java. *Multimedia Tools Appl.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 16, p. 7–27, January 2002. ISSN 1380-7501. Disponível em: http://dl.acm.org/citation.cfm?id=597039.597221.

[109] COURTIAT, J.-P.; OLIVEIRA, R. C. d. A reachability analysis of rt-lotos specifications. In: *Proceedings of the IFIP TC6 Eighth International Conference on Formal Description Techniques VIII*. London, UK, UK: Chapman & Hall, Ltd., 1996. p. 117–124. ISBN 0-412-73270-X. Disponível em: http://dl.acm.org/citation.cfm?id=646214.681526.

[110] BEEKMAN, G. *HyperCard 2.3 in a Hurry: The Fast Track to Multimedia*. Belmont, CA, USA: Wadsworth Publ. Co., 1996. ISBN 053451300X.

[111] BEEKMAN, G. *Hypercard Script Language Guide: The Hypertalk Language*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1990. ISBN 0201570815.

[112] KELLOGG, O.; BHATNAGAR, V. *Macromedia Authorware 6 with Cdrom*. [S.l.]: Macromedia Press, 2001. ISBN 0201774267.

[113] BUCHANAN, M. C.; ZELLWEGER, P. T. Automatic temporal layout mechanisms revisited. *ACM Trans. Multimedia Comput. Commun. Appl.*, ACM, New York, NY, USA, v. 1, p. 60–88, February 2005. ISSN 1551-6857. Disponível em: http://doi.acm.org/10.1145/1047936.1047942.

[114] EUN, S.; NO, E. S.; KIM, H. C.; YOON, H.; MAENG, S. R. Eventor: an authoring system for interactive multimedia applications. *Multimedia Syst.*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 2, p. 129–140, September 1994. ISSN 0942-4962. Disponível em: http://dl.acm.org/citation.cfm?id=197999.198002.

[115] MILLER, D.; MACDOWALL, G. *Inside Macromedia Director 5 with Lingo for MacIntosh with Cdrom*. Thousand Oaks, CA, USA: New Riders Publishing, 1996. ISBN 1562055674.

[116] MOOCK, C. *Action Script for Flash MX: the Definitive Guide 2e*. 2nd. ed. Sebastopol, CA, USA: O'Reilly & Associates, Inc., 2002. ISBN 059600396X.

[117] LOPEZ-NATAREN, C.; VISO-GUROVICH, E. An ecmascript compiler for the .net framework. In: *ENC '05: Proceedings of the Sixth Mexican International Conference on Computer Science*. Washington, DC, USA: IEEE Computer Society, 2005. p. 235–239. ISBN 0-7695-2454-0.

[118] PIMENTEL, M. G. C.; GOULARTE, R.; CATTELAN, R. G.; SANTOS, F. S.; TEIXEIRA, C. Ubiquitous interactive video editing via multimodal annotations. In: *Proceedings of the 6th European conference on Changing Television Environments*. Berlin, Heidelberg: Springer-Verlag, 2008. (EUROITV '08), p. 72–81. ISBN 978-3-540-69477-9.

[119] TEIXEIRA, C. A. C.; FREITAS, G. B.; PIMENTEL, M. d. G. Distributed discrimination of media moments and media intervals: a watch-and-comment approach. In: *Proceedings of the 2010 ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2010. (SAC '10), p. 1929–1935. ISBN 978-1-60558-639-7. Disponível em: http://doi.acm.org/10.1145/1774088.1774496.

[120] TEIXEIRA, C.; MELO, E.; FREITAS, G.; SANTOS, C.; PIMENTEL, M. d. Discrimination of media moments and media intervals: sticker-based watch-and-comment annotation. *Multimedia Tools and Applications*, Springer Netherlands, p. 1–22, 2011. ISSN 1380-7501. 10.1007/s11042-011-0846-6. Disponível em: http://dx.doi.org/10.1007/s11042-011-0846-6.

[121] MARTINS, D.; PIMENTEL, M. da G. C. End-user ubiquitous multimedia production: Process and case studies. In: *Ubi-Media Computing (U-Media), 2011 4th International Conference on*. [S.l.: s.n.], 2011. p. 197 –202.

[122] SOARES, L. F. G.; RODRIGUES, R. F.; COSTA, R. M. de R. Automatic building of frameworks for processing xml documents. In: *WebMedia '06: Proceedings of the 12th Brazilian symposium on Multimedia and the web*. New York, NY, USA: ACM, 2006. p. 118–127. ISBN 85-7669-100-0.

[123] FRAUNHOFER. *JAME Author*. jun. 2010. Disponível em: http://www.broadcastpapers.com/whitepapers/IMKJame.pdf.

[124] ICAREUS. *iTV Suite*. jun. 2010. Disponível em: http://www.icareus.com/web/guest/itv-suite-author

[125] BULTERMAN, D. C. A.; HARDMAN, L.; JANSEN, J.; MULLENDER, K. S.; RUTLEDGE, L. Grins: a graphical interface for creating and playing smil documents. *Readings in multimedia computing and networking*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, p. 817–827, 2001.

[126] DELTOUR, R.; ROISIN, C. The limsee3 multimedia authoring model. In: *Proceedings of the 2006 ACM symposium on Document engineering*. New York, NY, USA: ACM, 2006. (DocEng '06), p. 173–175. ISBN 1-59593-515-0. Disponível em: http://doi.acm.org/10.1145/1166160.1166203.

[127] NETO, M. C. M.; SANTOS, C. A. S. Storytocode: a new model for specification of convergent interactive digital tv applications. *J. Braz. Comp. Soc.*, v. 16, n. 4, p. 215–227, 2010.

[128] JONES, J. Dvb-mhp/java tv™data transport mechanisms. In: *CRPIT '02: Proceedings of the Fortieth International Conference on Tools Pacific*. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2002. p. 115–121. ISBN 0-909925-88-7.

[129] SILVA, H. V. O.; RODRIGUES, R. F.; SOARES, L. F. G.; SAADE, D. C. M. Ncl 2.0: integrating new concepts to xml modular languages. In: *DocEng '04*. New York, NY, USA: ACM, 2004. p. 188–197. ISBN 1-58113-938-1.

[130] PEMBERTON, S. *XHTML 1.0: The Extensible HyperText Markup Language (Second Edition)*. August 2002. World Wide Web Consortium, Recommendation REC-xhtml1-20020801.

[131] ANICIC, D.; FODOR, P.; STUHMER, R.; STOJANOVIC, N. Event-driven approach for logic-based complex event processing. In: *Computational Science and Engineering, 2009. CSE 09. International Conference on*. [S.l.: s.n.], 2009. v. 1, p. 56 –63.

[132] NETO, M. C. M.; SANTOS, C. A. An approach based on events for treating the late tuning problem in interactive live tv shows. In: *Proceedings of the 1st ACM international workshop on Events in multimedia*. New York, NY, USA: ACM, 2009. (EiMM '09), p. 41–48. ISBN 978-1-60558-754-7. Disponível em: http://doi.acm.org/10.1145/1631024.1631034.

[133] HUNTER, J. *JDOM*. jun. 2011. Disponível em: http://http://www.jdom.org/.

[134] NETO, M. C. M.; PASSOS, C.; SANTOS, C. A. S. Tecnologias para processamento de documentos xml: Uma abordagem java. In: *ERI '03: III Escola Regional de Informática*. Porto Alegre, RS, Brazil: SBC, 2003. p. 63–94.

[135] TAVARES, T. A.; SANTOS, C. A. S.; ASSIS, T. Rocha de; PINHO, C. Braga Bittencourt de; CARVALHO, G. Mariniello de; COSTA, C. Santana da. *A TV Digital Interativa como Ferramenta de Apoio à Educação Infantil*. [S.l.]: SBC, 2007. 31–44 p. (2, v. 14). ISBN 14145685.

[136] BACHMAYER, S.; LUGMAYR, A.; KOTSIS, G. Convergence of collaborative web approaches and interactive tv program formats. *Information Systems Journal*, v. 6, n. 1, p. 74–94, 2010. Disponível em: http://www.emeraldinsight.com/10.1108/17440081011034493.

[137] COPPENS, T.; HANDEKYN, K.; VANPARIJS, F. Amigotv: A social tv experience through triple-play convergence. *Alcatel Technology white paper*, 2005. Disponível em: http://www.telecomreview.ca/eic/site/tprp-gecrt.nsf/vwapj/AmigoTV.pdf/FILE/AmigoTV.pdf.

[138] WILLIAMS, C. M.; WAGES, R. Video conducting the olympic games 2008: the itv field trial of the eu-ist project live. In: *Proceedings of the 3rd international conference on Digital Interactive Media in Entertainment and Arts*. New York, NY, USA: ACM, 2008. (DIMEA '08), p. 436–440. ISBN 978-1-60558-248-1. Disponível em: http://doi.acm.org/10.1145/1413634.1413711.

[139] LEE, C.-H. J.; CHANG, C.; CHUNG, H.; DICKIE, C.; SELKER, T. Emotionally reactive television. In: *Proceedings of the 12th international conference on Intelligent user interfaces*. New York, NY, USA: ACM, 2007. (IUI '07), p. 329–332. ISBN 1-59593-481-2. Disponível em: http://doi.acm.org/10.1145/1216295.1216359.

[140] URSU, M. F.; THOMAS, M.; KEGEL, I.; WILLIAMS, D.; TUOMOLA, M.; LINDSTEDT, I.; WRIGHT, T.; LEURDIJK, A.; ZSOMBORI, V.; SUSSNER, J.; MYRESTAM, U.; HALL, N. Interactive tv narratives: Opportunities, progress, and challenges. *ACM Trans. Multimedia Comput. Commun. Appl.*, ACM, New York, NY, USA, v. 4, p. 25:1–25:39, November 2008. ISSN 1551-6857. Disponível em: http://doi.acm.org/10.1145/1412196.1412198.

[141] FERREIRA, T. P.; KULESZA, R.; FILHO, G. L. S. itv visual design: Uma ferramenta para desenvolvimento visual de aplicacões java para o padrão brasileiro de tv digital. In: SBC (Ed.). *Proceedings: SBQS 201o, Curitiba, SC, Brasil*. Porto Alegre, RS, Brasil: SBC, 2010. p. 1–8.

[142] KASKALIS, T. H.; TZIDAMIS, T. D.; MARGARITIS, K. *Multimedia Authoring Tools: The Quest for an Educational Package*. 2007. 135–162 p.

[143] YANG, X.-m.; GU, P.; DAI, H. Mapping approach for model transformation of mda based on xmi/xml platform. In: *Proceedings of the 2009 First International Workshop on Education Technology and Computer Science - Volume 02*. Washington, DC, USA: IEEE Computer Society, 2009. p. 1016–1019. ISBN 978-0-7695-3557-9. Disponível em: http://dl.acm.org/citation.cfm?id=1545020.1546537.

APÊNDICES

Apêndice

A

Este apêndice apresenta o storyboard e a descrição dos cenários do projeto Árbaro produzida pelos membros da equipe do projeto responsáveis pela criação artística.

STORYBOARD

A.1 DESCRIÇÃO DOS CENÁRIOS

A aplicação representa o plantio e desenvolvimento de uma árvore para reflorestamento, iniciando com a atividade de escolha do personagem temático (fogo, terra, água e ar) e da semente. Será desenvolvida e apresentada ao usuário em etapas: 1) plantio da semente, 2) crescimento de flores e frutos e 3) árvore adulta. O usuário terá que completar uma etapa antes de passar à próxima.

Cada semente plantada é considerada uma planta ativa do usuário na aplicação. Durante o crescimento das plantas ativas, até que cheguem à idade adulta, o usuário será responsável pelos cuidados necessários à sua saúde. Ao completar o desenvolvimento e atingir a terceira etapa, a planta deixa de estar ativa, ou seja, não necessitará mais de cuidados diários, pois será transferida ao bosque de reflorestamento. O usuário poderá rever todas as plantas desenvolvidas por ele e que não estejam mais ativas ao visitar o bosque de reflorestamento.

Durante o tempo em que estiver cuidando de uma planta o jogo deverá oferecer ao usuário dois tipos de interação: 1) Participar de mini-Jogos interativos (Quiz, Memória etc.) ganhando pontos caso obtenha sucesso; e 2) Assistir apresentações interativas com temas relacionados ao meio ambiente como respiração de plantas / fotossíntese, papel de insetos na polinização, pragas agrícolas etc.

As opções de sementes devem ser exemplos de árvores do mundo real (jacarandá, ipê etc.) e o crescimento dessas plantas deve manter uma relação de proporcionalidade com o tempo real. O jogo deve contar com 3 opções de áudio para : 1) abertura e escolha de sementes 2) cuidando da planta e 3) tipos de interação.

Antes de iniciar o jogo o usuário deve se cadastrar e criar o login/senha no sistema.

Figura A.1. Storyboard do projeto Árbaro

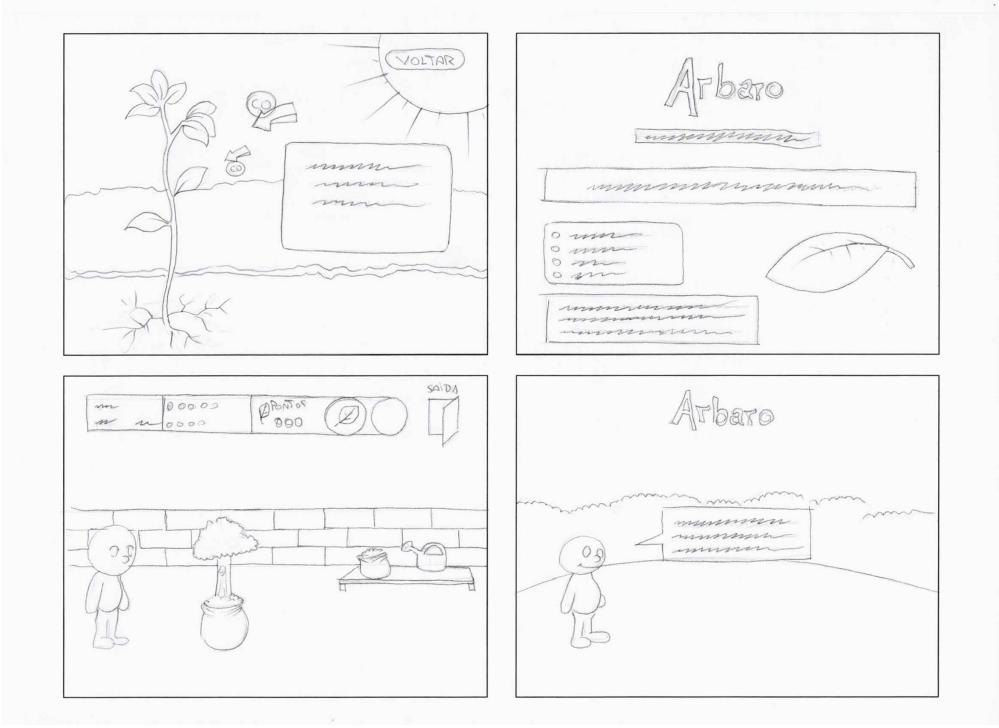

Figura A.2. Storyboard do projeto Árbaro

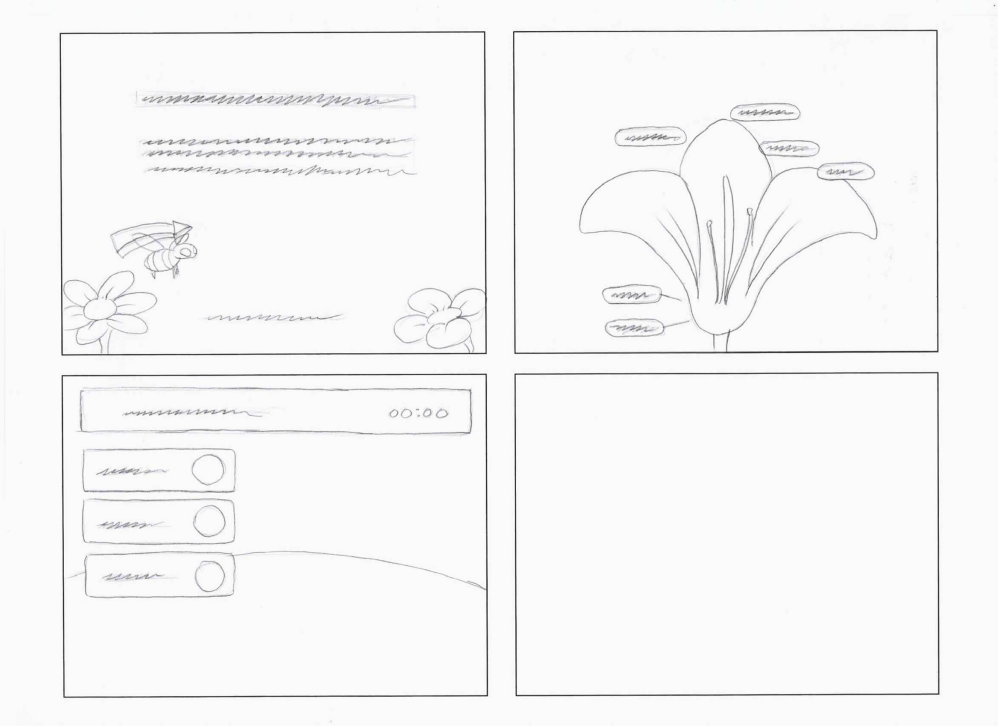

Figura A.3. Storyboard do projeto Árbaro

Apêndice

B

Este apêndice apresenta a lista de requisitos funcionais e não funcionais do projeto Árbaro produzida pelos membros da equipe do projeto responsáveis pelo desenvolvimento dos componentes interativos (software).

REQUISITOS

B.1 REQUISITOS FUNCIONAIS

B.1.1 Ativar uma planta

A fim de ativar uma nova planta o usuário deverá escolher, entre um conjunto de opções, qual tipo de árvore deseja plantar. Nesta funcionalidade, deverão estar disponíveis informações relevantes para a escolha, tais quais: tempo de germinação da semente, tempo de crescimento da árvore, necessidades especiais e uso de cada árvore etc. A semente já deverá possuir uma identidade visual, que a personalize.

B.1.2 Visualizar o estágio e histórico de crescimento da planta

Após entrar na aplicação, deverá sempre estar disponível ao usuário os elementos visuais representando as suas plantas em crescimento. A imagem deverá indicar claramente o estado atual da planta em termos de forma, tamanho e idade. Deverá ser possível acessar indicadores de saúde da planta no seu estágio atual, tais quais: nutrição, nível de água no organismo, existência de doenças, bem como o seu histórico de crescimento (ocorrências ao longo do tempo).

B.1.3 Preparar o solo

Após a escolha da semente, a aplicação deverá permitir que o usuário prepare o solo para o seu plantio. Nesta funcionalidade, deverão estar disponíveis informações relevantes à tarefa, tais quais: técnicas de adubação e plantio, alternativas orgânicas a fertilizantes e agrotóxicos etc.

B.1.4 Acompanhar o período de germinação

Ao término da preparação e plantio, o usuário deverá ser obrigado a esperar um tempo pré-definido para a germinação. A aplicação deverá controlar que, durante este período, sejam executadas as tarefas de molhar o solo e conferir as condições de temperatura.

B.1.5 Regar a planta

A qualquer momento durante o plantio e crescimento da planta ativa, a aplicação deverá permitir que o usuário possa regá-la, indicando a quantidade de água que deseja utilizar.

B.1.6 Adubar a planta

A qualquer momento durante o plantio e crescimento da planta ativa, a aplicação deverá permitir que o usuário possa adubar o solo, indicando que tipo de substância que deseja utilizar.

B.1.7 Informar aparecimento do broto

Ao final do período de germinação, caso todas as atividades tenham sido cumpridas com sucesso, o sistema deverá informar ao usuário o aparecimento do broto, além de modificar o estado de semente para broto, permitindo, assim, o acompanhamento do crescimento da árvore.

B.1.8 Informar aparecimento de folhas

Durante o crescimento da árvore, após um determinado período pré-definido de tempo, o sistema deverá informar ao usuário o aparecimento das primeiras folhas da árvore plantada, bem como modificar o elemento visual da árvore plantada. Nesta atividade, a aplicação deverá focar o processo de fotossíntese, explicando como ela ocorre, bem como o processo de transpiração das folhas.

B.1.9 Informar crescimento das raízes

Durante o crescimento da árvore, em intervalos de tempo pré-definidos, o sistema deverá informar ao usuário sobre o progresso do crescimento das raízes, bem como modificar o elemento visual da árvore plantada.

B.1.10 Informar crescimento do caule

Durante o crescimento da árvore, após um determinado período pré-definido de tempo, o sistema deverá informar ao usuário sobre o progresso do crescimento do caule, bem como modificar o elemento visual da árvore plantada.

B.1.11 Informar chegada à segunda fase

Após um determinado período pré-definido de tempo, o sistema deverá informar ao usuário sobre a chegada da árvore à segunda etapa, em que a árvore produzirá frutos e flores, explicando suas principais características.

B.1.12 Gerar e informar ocorrências de ameaças à arvore

Na segunda etapa da aplicação (Flores e Frutos), a aplicação deverá gerar, de maneira aleatória, ocorrências pré-definidas que ameacem a vida da planta, tais quais: mudanças no clima, ação do homem, surgimento de insetos parasitas etc. Estas ocorrências deverão ser informadas ao usuário, com conteúdo relativo às causas, conseqüências a possíveis ações relacionadas à ocorrência.

B.1.13 Permitir ações de resposta às ocorrências

A aplicação deverá permitir ao usuário cuja planta se encontre ameaçada por alguma ocorrência a execução de ações pré-definidas para solucionar o problema.

B.1.14 Gerar e informar ocorrências benéficas

Na segunda etapa da aplicação (Flores e Frutos), a aplicação deverá gerar, de maneira aleatória, ocorrências pré-definidas benéficas tais quais: visitas de pássaros, polinização de outras árvores, sombra para homens etc. Estas ocorrências deverão ser informadas ao usuário.

B.1.15 Informar aparecimento de flores e frutos

Após um determinado período pré-definido de tempo, o sistema deverá informar ao usuário sobre o aparecimento de flores e frutos

B.1.16 Informar chegada à fase de árvore adulta

Após um determinado período pré-definido de tempo, o sistema deverá informar ao usuário sobre a chegada da árvore à terceira etapa, fase adulta. O usuário deverá receber um certificado de reflorestamento, como um pergaminho e uma mensagem parabenizando pela iniciativa.

Ao chegar à idade adulta, a planta deixa de estar ativa para o usuário, ou seja, seu elemento visual não fica mais visível durante a aplicação nem serão geradas novas ocorrências para ela. No entanto, ela deverá ficar armazenada em local acessível a qualquer momento para que o usuário possa consultar suas condições de vida e rever a história do seu crescimento.

B.1.17 Visitar o bosque de reflorestamento

A qualquer momento, o usuário poderá visitar o bosque de reflorestamento, visualizando o estado e o histórico de crescimento das árvores adultas que ele plantou. Nesta funcionalidade, a aplicação deverá apresentar conteúdo relacionado a reflorestamento e suas implicações.

B.2 REQUISITOS NÃO-FUNCIONAIS

B.2.1 Utilizar recursos audiovisuais

A aplicação deverá utilizar elementos sonoros e reforço audiovisual, com desenhos criativos e coloridos, em todas as interações com os usuários.

B.2.2 Administrar o credenciamento do usuário

A aplicação deverá solicitar o cadastro do usuário através de senha e sempre exigir o seu credenciamento na entrada.

B.2.3 Conduzir o usuário através de um personagem

Haverá um personagem responsável por interagir com o usuário e por conduzi-lo o em todas as etapas da aplicação.

B.2.4 Tratar o usuário de maneira pessoal

Em todas as interações com o usuário, a aplicação deverá usar um tratamento pessoal com o usuário, sempre utilizando seu nome de acesso.

B.2.5 Disponibilizar conteúdo didático explicativo

Ao longo do processo de plantio da árvore, a qualquer momento, o usuário poderá consultar o conteúdo didático relativo a qualquer fase ou etapa do plantio, independente de onde ele se encontre no processo.

B.2.6 Apresentar todo conteúdo de forma interativa

Toda a aplicação deverá ser concebida para que o usuário sempre tenha oportunidades de interação com as tarefas sendo executadas,

B.2.7 Manter proporcionalidade da unidade de tempo

A aplicação terá uma unidade de tempo própria que deverá ser proporcional ao tempo real. A taxa de conversão deverá ser sempre a mesma em todas as atividades, a fim de manter uma correlação com o mundo real.

B.2.8 Plataforma de funcionamento

A aplicação deverá funcionar dentro de navegador web, em dispositivo móvel (JME) e para o ambiente de TV Digital Interativa.

Apêndice

C

Este apêndice os principais screenshots do projeto Arbaro produzidos pelos membros da equipe do projeto responsáveis pela criação artística.

STORYBOARD

Figura C.1. *screenshots* do projeto Arbaro