Capítulo 3

Desenvolvimento de Aplicações Imperativas para TV Digital no *middleware* Ginga com Java

Raoni Kulesza[1], Carlos Alberto Saibel Santos[2], Tatiana Aires Tavares[1], Manoel Marques Neto[2], Guido Lemos de Souza Filho[1]

[1] Laboratório de Aplicações de Vídeo Digital (LAVID), Departamento de Informática, UFPB
{raoni, tati, guido}@lavid.ufpb.br

[2] Doutorado Multiinsitucioal em Ciência da Computação
Departmento de Ciência da Computação, IM, UFBA
saibel@ufba.br; manoelnetom@gmail.com

Abstract

This chapter presents an introduction to Java applications developing for interactive digital TV based on Ginga, middleware standard of Brazilian System of Digital TV. Java applications in Ginga use a set of APIs based on the JavaDTV specification and brazilian innovations, recently recognized internationally. The chapter presented the features of these solutions through the sample application construction step by step. The chapter also presents techniques and tools to support software engineering to facilitate the development and contribute to the reuse of application code.

Resumo

Este capítulo apresenta uma introdução ao desenvolvimento de aplicações Java para TV digital interativa com base no middleware Ginga, padrão do Sistema Brasileiro de TV Digital. No Ginga as aplicações Java utilizam um conjunto de APIs baseado na especificação JavaDTV e inovações brasileiras, reconhecidas recentemente em âmbito internacional. O capítulo apresentada as funcionalidades dessas soluções através da construção passo a passo de uma aplicação. O capítulo também apresenta técnicas e ferramentas de apoio a engenharia de software que visam facilitar o desenvolvimento e contribuir para o reuso do código das aplicações geradas.

3.1. Introdução

A possibilidade de veiculação de softwares e dados juntamente com os fluxos de áudio e vídeo transmitidos pelas emissoras é uma das maiores inovações trazidas pelos sistemas de TV digital. Esta inovação cria oportunidades para a produção e desenvolvimento de serviços e aplicações dos mais diversos tipos, voltados para esta nova mídia interativa.

Este capítulo explora os desafios e novas oportunidades para o profissional de software advindos da implantação do Sistema Brasileiro de TV Digital (SBTVD ou ISDB-Tb) terrestre[1]. O objetivo do capítulo é atualizar o profissional de software com relação aos novos modelos de produção de conteúdo para a TV digital integrando software e conteúdo audiovisual. Para tanto, primeiramente, a definição do conceito de TV interativa é analisado tendo como base uma série de referências da literatura. Na seção seguinte, discute-se o aparecimento da TV digital no contexto de um processo mais amplo de convergência digital, caracterizado pela fusão de tecnologias e facilidade de produção e acesso de conteúdo. Em seguida, as aplicações interativas são discutidas, procurando caracterizar a parte como o projeto da interatividade impacta na geração desse novo formato de conteúdo televisivo. Chega-se então ao processo de desenvolvimento dos programas interativos e de como estes novos requisitos são importantes para o desenvolvimento de software para a TV. Por fim, a plataforma disponível para o desenvolvimento de aplicações interativas para TV Digital no Brasil é abordada. Em especial, a parte do *middleware* Ginga destinada a execução de programas em Java, denominada Ginga-J, é detalhada e um exemplo de aplicação é apresentado.

3.2. Interatividade na TV

Pode parecer estranho, mas o conceito de TV Interativa (TVi) já existe há mais de 50 anos[2]. Por outro lado, a definição precisa do que é chamado de TVi ainda é uma questão que está longe de ser resolvida, conforme mostra a Tabela 3.1. Mais ainda, termos como TV Digital Interativa (TVDI), Web TV, TV na Internet e outros são muitas vezes utilizados como sinônimos de TVi[3].

A inclusão da interatividade nos programas de TV tornou-se 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 telespectadores. Alguns exemplos recentes de programas líderes de audiência utilizam a colaboração (interação) do telespectador para: (1) por meio de mecanismos de preferência popular, definir os rumos do programa (Big Brother Brasil); (2) produzir e enviar conteúdos a serem integrados aos programas (Quadro Bola Cheia Bola Murcha do Fantástico); ou ainda (3) interagir com participantes de um programa (mensagens com questões para os comentaristas durante uma partida de futebol).

---

[1] O ISDB-Tb, baseado no padrão japonês ISDB-T, tinha sido adotado por Peru, Argentina, Chile, Venezuela, Equador, Costa Rica, Paraguai, Filipinas e Bolívia no momento da escrita deste texto.

[2] O programa "Winky Dink And You", produzido pela CBS-TV na década de 50, é considerado por muitos como o primeiro programa interativo para a TV http://www.toontracker.com/winky/winky.htm

[3] Em http://www.itvdictionary.com/itv.html dezenas de termos associados à TV interativa são listados.

Este novo desenho de conteúdo televisivo tem impactos diretos na forma com a 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.

Tabela 3.1: Definições de TV Interativa encontradas na literatura

Mark Gawlinski [Gawlinski 2003, p. 5]	[...] 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. 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.
Konstantinos Chorianopoulos [Chorianopoulos 2004, p. 9]	[…] é um termo genérico utilizado para todos os sistemas de televisão que oferecem ao consumidor interatividade além da troca de canais e do teletexto.
Jerry Whitaker [Whitaker 2001]	[...] tudo o que permite que um consumidor interaja com o sistema usando um controle remoto ou um teclado para acessar novos e avançados serviços.
Robert Schwalb [Schwalb 2003]	[...] 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.
Karyn Y. Lu [Lu 2005]	[...] qualquer programa de televisão que incorpora conteúdo adicional ou algum tipo de interatividade com o usuário. [...] termo genérico que cobre a convergência da televisão com tecnologias das mídias digitais como computadores, PVRs, jogos eletrônicos, dispositivos móveis e sem fio, possibilitando a interação com o usuário.

A intenção deste capítulo não é apresentar uma definição única e definitiva para o termo TVi, mas apresentar a possibilidade de construir aplicações interativas usando a linguagem de programação Java. Mais especificamente, o capítulo trata de um caso específico de aplicações: aquelas **enviadas aos receptores**[4] **por uma emissora de TV juntamente com seus programas, utilizando o canal que está sob seu controle**.

[4] Os termos receptor, receptor de TV, set-top-box ou terminal de acesso são utilizados indistintamente na sequência do texto.

3.3. Convergência Digital

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, ele pode ser visto 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 nas nos jornais e agências de notícias, na indústria fonográfica entre outras. 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 e os seus provedores de vídeo na Web 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 sócio-econômica 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.

A TV digital, que integra a capacidade de processamento aos receptores de TV, também é um dos produtos gerados neste processo de convergência. Mais ainda, a TV digital surge em um momento no qual o modelo de distribuição de conteúdo por *broadcast* tem que fazer face aos novos modelos de produção e distribuição compartilhada e sob demanda de conteúdos audiovisuais pela Web. Neste cenário, o desafio para a produção de conteúdo audiovisual está na criação de novas aplicações e de formas de distribuição de conteúdo que integrem as características de qualidade visual, simplicidade de acesso, confiabilidade de transmissão e baixo custo, oriundos do modelo da TV tradicional, com o acesso personalizado, busca de informações sob demanda, compartilhamento de conteúdo e comunicação entre grupos do modelo Web. O uso de TVs com acesso direto à Internet (as chamadas TV conectadas) e o desenvolvimento de aplicações do tipo *widgets* (moderadas pelas emissoras, fabricantes ou algum provedor) para acesso a recursos Web não solucionam a questão, por outro lado acirram a competitividade entre os dois mundos. Entretanto, pensar na TV digital como uma plataforma integrada de acesso a serviços, conteúdos e, principalmente, a pessoas, parece ser um caminho mais promissor. A chamada TV Social (apontada pelo *MIT Technology Review* como uma das 10 mais importantes tecnologias emergentes para os próximos anos) aparece como um caminho interessante para o problema, onde a convergência tecnológica transforma a TV num dispositivo central para a integração de indivíduos com interesses semelhantes [Cesar 2009].

A evidente melhoria da qualidade do conteúdo visual com a digitalização do sistema de TV, novos problemas devem ser enfrentadas pelas emissoras na geração de conteúdo interativo para TV digital. No modelo de *broadcast* tradicional, toda a produção do conteúdo fica sob responsabilidade da emissora, a qual integra uma série de elementos (ícones, vinhetas, animações) sobre um conteúdo único audiovisual, que é

transmitido sobre o canal controlado pela emissora. Nesse modelo, o receptor de TV sintonizado no canal da emissora decodifica o sinal e apresenta o conteúdo integrado produzido a uma certa taxa de quadros e de forma sincronizada com o áudio correspondente. 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. 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 SBTVD especifica vários padrões de referência, dentre os quais se destacam o H.264 para a codificação de vídeo, o MPEG-4 para o áudio, além do MPEG-2 System para o transporte (multiplexação) dos fluxos de áudio, vídeo e dados. Dentre os padrões estabelecidos, porém, a camada *middleware* Ginga é, sem dúvida, a mais importante diferença do SBTVD com relação aos outros padrões internacionais.

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 composta por dois subsistemas: (1) o de apresentação, que dá suporte a execução de aplicações declarativas NCL [Soares 2007] e (2) o de execução, que dá suporte a aplicações imperativas Java [Souza Filho 2007], que são o foco deste capítulo.

3.4. Aplicações para TV Digital

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.

3.4.1 Paradigmas Imperativo e Declarativo

O tipo de funcionalidade a ser adicionada a um programa de TV digital 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 linguagens declarativas, imperativas (algumas vezes também chamadas de procedurais) ou ainda, com o uso de uma abordagem híbrida, integrando os dois tipos de linguagens.

As linguagens declarativas tendem a ser mais intuitivas, à primeira vista e, portanto, mais simples por programadores (também chamados autores, nesse caso), os quais definem: (1) restrições espaciais e temporais (sincronismo) entre as mídias que compõem a aplicação interativa e (2) tratamento de eventos de interação do usuário com aplicação. Uma linguagem declarativa enfatiza a descrição declarativa do problema, ao invés da sua decomposição em uma implementação algorítmica. Elas são linguagens de

mais alto nível de abstração, usualmente ligadas a um domínio ou objetivo específico, onde o programador fornece apenas o conjunto das tarefas a serem realizadas, não estando preocupado com os detalhes de como elas serão executadas [Soares 2007]. A linguagem declarativa utilizada no contexto do SBTVD é a NCL.

As linguagens imperativas tendem a ser mais apropriadas para aplicações genéricas, orientadas a eventos, nas quais o programador possui um maior controle do código, sendo capaz de estabelecer todo o fluxo de controle e execução de seu programa. Neste caso, processador do receptor deve ser informado sobre cada passo a ser executado. Pode-se afirmar que, em linguagens imperativas, o programador possui um maior poder sobre o código, sendo capaz de estabelecer todo o fluxo de controle e execução de seu programa. A linguagem mais usual encontrada nos ambientes imperativos dos sistemas de TV digital é a Java. Uma aplicação para TV desenvolvida nesta linguagem é também conhecida como Xlet.

As aplicações imperativas podem referenciar conteúdos declarativos, ou até construir e iniciar a apresentação de um conteúdo declarativo. Estas aplicações são denominadas de híbridas constituem uma poderosa ferramenta de produção de conteúdo para TV digital, ao unir as vantagens dos paradigmas declarativo e imperativo. Como os dois tipos de aplicações devem coexistir no receptor, todos os sistemas de TV digital propostos no mundo trabalham com a idéia de que o receptor possui um *middleware* que dê suporte a ambos os paradigmas.

3.4.2 Processando Dados no Receptor de TV

As funcionalidades de um programa de TV dependem, dentre outros fatores, dos dados e objetos que irão ser processados pelas aplicações. A depender da sua natureza, os dados e objetos podem ser: (1) enviados pela própria emissora de TV, em broadcast, junto ao conteúdo audiovisual; (2) acessados por meio do canal de interatividade ou (3) acessados, de forma alternativa, a partir de um repositório (através de uma porta USB, por exemplo). A grade de programação de uma emissora (EPG ou *Eletronic Programming Guide*), por não conter dados pessoais do usuário e não ser composta por arquivos extensos é um exemplo típico de aplicação que pode ser enviada junto com os programas de uma emissora. A visualização de mensagens de e-mail, devido ao seu caráter pessoal, seria um exemplo de aplicação com dados enviados por intermédio do canal de interatividade. Outro ponto a ser observado é o fato do canal de interatividade possibilitar que o usuário receba apenas as informações que solicita, enquanto que os dados recebidos pelo canal controlado pela emissora são enviados mesmo sem solicitação do usuário.

A oferta de serviços é dependente da infraestrutura disponível ao usuário. 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 usuários 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.

A transmissão de dados e objetos gerados pelas emissoras e processados nos receptores de TV é feita de acordo com um mecanismo denominado carrossel

3.4.3 Carrossel de Dados

O carrossel de dados e/ou de objetos 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 armazenados e aplicativos a serem disponibilizados aos usuários. Todos os sistemas de TV digital terrestre utilizam o protocolo especificado no padrão DSM-CC (*Digital Storage Media – Command and Control*) [ISO/IEC 13818-6 1998]. 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 ou que partes dos programas ou dados sejam perdidas. Para aplicações que utilizam dados e aplicações provenientes do canal de interatividade (por exemplo, acessar uma aplicação de correio eletrônico pela TV) ou dados 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.

3.4.4. Concepção de Aplicações Interativas

O software transmitido juntamente com o conteúdo audiovisual de um programa de TV[5] é chamado, na TV Interativa, de aplicativo ou aplicação. No entanto, as aplicações interativas têm algumas particularidades em relação às tradicionais aplicações de software [Veiga 2007]:

1. Elas podem fazer parte de um programa de TV, o qual tem formato e contexto próprios;
2. Apesar de serem oferecidas por meio de um *front-end* único (a TV), podem atender a mais de um usuário. De fato, a TV atende, de forma coletiva, a um grupo de usuários, que possuem perfis particulares e cujos anseios em um veículo de comunicação de massa são mapeados em diversidade na grade de programação;
3. Elas devem lidar com interatividade (em diferentes níveis) e uma possível organização não-linear de conteúdo.

O tipo de funcionalidade a ser adicionada a um programa de TV digital 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 estão, normalmente, associadas aos paradigmas declarativo, imperativo (algumas vezes também chamado de procedural) ou híbrido.

As linguagens declarativas tendem a ser mais intuitivas, à primeira vista e, portanto, mais simples por programadores (também chamados autores, nesse caso), os quais definem: (1) restrições espaciais e temporais (sincronismo) entre as mídias que compõem a aplicação interativa e (2) tratamento de eventos de interação do usuário com aplicação. Uma linguagem declarativa enfatiza a descrição declarativa do problema, ao invés da sua decomposição em uma implementação algorítmica. Elas são linguagens de

[5]Refere-se a um programa de TV Convencional, doravante denominado apenas de programa de TV.

mais alto nível de abstração, usualmente ligadas a um domínio ou objetivo específico, onde o programador fornece apenas o conjunto das tarefas a serem realizadas, não estando preocupado com os detalhes de como elas serão executadas [Soares 2007]. A linguagem declarativa utilizada no contexto do SBTVD é a NCL.

As linguagens imperativas tendem a ser mais apropriadas para aplicações genéricas, orientadas a eventos, nas quais o programador possui um maior controle do código, sendo capaz de estabelecer todo o fluxo de controle e execução de seu programa. Como os dois tipos de aplicações devem coexistir num ambiente de TV digital, todos os sistemas proposto trabalham com a idéia de que o receptor possui um *middleware* que dê suporte a ambos os paradigmas. Neste cenário, o computador deve ser informado sobre cada passo a ser executado. Pode-se afirmar que, em linguagens procedurais, o programador possui um maior poder sobre o código, sendo capaz de estabelecer todo o fluxo de controle e execução de seu programa. Entretanto, para isso, ele deve ser bem qualificado e conhecer bem os recursos de implementação. A linguagem mais usual encontrada nos ambientes imperativos de um sistema de TV digital é a Java. Uma aplicação para TV digital desenvolvida nesta linguagem é também conhecida como *Xlet*. As aplicações imperativas podem referenciar conteúdos declarativos, ou até construir e iniciar a apresentação de um conteúdo declarativo. Estas aplicações são denominadas de híbridas constituem uma poderosa ferramenta de produção de conteúdo para TV digital, ao unir as vantagens dos paradigmas declarativo e imperativo.

Dessa forma, para se conceber uma aplicação interativa é necessária a compreensão das diferentes possibilidades de interação deste programa. Nesse sentido, a Figura 3.1 [Silva 2010] sumariza as características dos aplicativos e programas interativos.

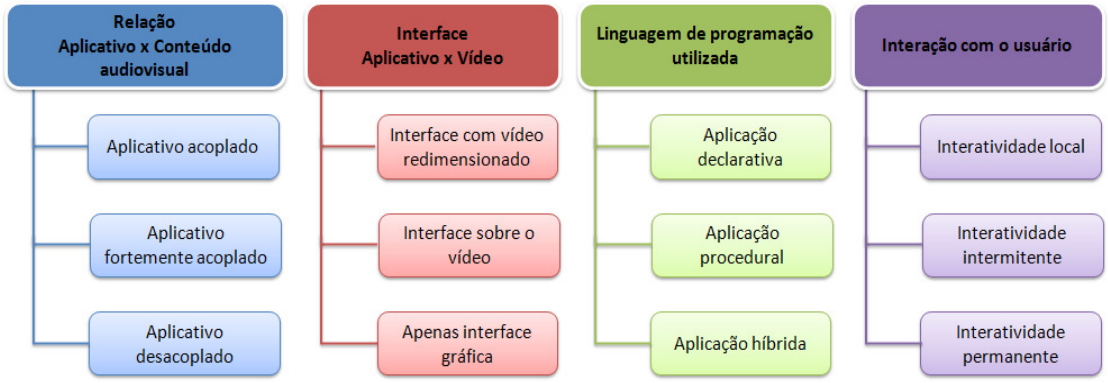

Figura 3.1 - Principais características das aplicações interativas.

Como foi apresentado nas seções anteriores, a interatividade traz uma grande novidade em relação à TV tradicional: a possibilidade vincular uma gama aplicações de computador ao conteúdo televisivo transmitido. Isto traz impactos significativos não só na forma de assistir TV, mas também na forma de produção do seu conteúdo.

Sabe-se que aplicações interativas nada mais são do que softwares; e como tal, precisam passar por um processo que garanta a execução consistente e a estruturação das atividades que compõem o trabalho da equipe de desenvolvimento, além da adequação e qualidade do produto final.

O grande desafio não está na produção das aplicações interativas em si, mas na produção dos programas de TV digital, que agregam mecanismos de interatividade ao programa de TV convencional a partir da vinculação de softwares interativos. Por esta razão, o processo de desenvolvimento de programas para TV digital integra atividades inerentes ao processo de produção de TV e atividades do desenvolvimento de software [Veiga 2007].

3.4.5. Software Tradicional x Software para TV

Em termos computacionais, uma aplicação de TV digital pode ser entendida como um software, especificamente uma aplicação multimídia, através do qual um telespectador pode interagir via controle remoto [Rodrigues 2006]. Isso significa que um telespectador pode receber do difusor, além do vídeo/áudio, softwares que possibilitam que ele interaja com o conteúdo veiculado.

De acordo com [Marques Neto 2008], os softwares para TV digital podem ser categorizados em:

1. Softwares que não têm relação com a semântica do conteúdo de áudio e vídeo apresentados, por exemplo, *e-mail* e *TV-Banking*.
2. Softwares que têm relação com a semântica do conteúdo de áudio e vídeo apresentados, mas sem restrições fortes de sincronização, por exemplo, a cotação das ações da bolsa durante um programa de economia.
3. Softwares que têm relação com a semântica do conteúdo de áudio e vídeo apresentados exibidos de forma sincronizada, por exemplo: anúncios interativos de produtos exibidos em momentos específicos de uma transmissão. Esta última categoria pode ainda ser subdividida em:

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

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

O processo de construção de softwares da primeira categoria não se diferencia dos processos tradicionais. Assim, a única diferença de um software de TV digital para os demais está no tratamento dado aos requisitos não-funcionais tais como, plataforma de execução (receptor de TV), dispositivos de entrada (controle remoto, *smartphone*, etc).

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

1. Os softwares podem são partes do conteúdo dos programas de TV e esses, por sua vez, têm formato e contexto próprios;
2. O resultado da execução pode ser gerar elementos na tela TV, que é um espaço de uso tradicionalmente coletivo ou em dispositivos separados, como no caso do Ginga [Soares 2009].

3. Eles exigem infra-estrutura de transmissão e componentes de software/hardware adequados para o seu funcionamento.
4. Eles modificam os programas de TV tradicionais de forma a capacitá-los para lidar com diferentes níveis de interatividade e com uma organização do conteúdo não-linear.

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

Com a necessidade de se produzir aplicações para 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 (computação, design, etc.). A introdução de componentes interativos (softwares) nos programas de TV passa a envolver tanto profissionais de produção de TV (diretores, produtores, editores e etc.), quanto aqueles ligados à produção de software (programadores, analistas, etc.). O desafio neste caso é aliar ao processo de modelagem de componentes para o ambiente de TV, as técnicas tradicionalmente usadas na engenharia de software para solução de questões como, por exemplo, o reuso e a estruturação de requisitos.

Alguns trabalhos tratam a questão da modelagem de componentes interativos para o ambiente de TV e apontam que a rota mais curta para resolver esse desafio é adaptar algumas das abordagens usadas em modelos de processo da engenharia de software às restrições do ambiente de TV. A próxima seção, apresenta um modelo de processo de software denominado StoryToCode, que aparece como uma alternativa a este problema [Marques Neto 2009].

3.4.6. O Modelo de Processo StoryToCode

O StoryToCode é um modelo de processo de software que permite a especificação e construção de componentes para uso em programas de TV digital e em outros contextos. Para este processo, o termo contexto refere-se à plataforma de execução do software (TV, smartphone, Web, etc.). A dinâmica do modelo consiste em partir de um Storyboard/Cenário e usar conceitos de 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 algumas visões sistêmicas (estrutura, eventos, etc.) a fim de poder reusar os artefatos gerados em outros contextos.

A escolha do Storyboard/Cenário como ponto de partida assume as seguintes hipóteses: (1) os Storyboards/Cenários são artefatos que sempre estão presentes no processo de especificação de um programa interativo e (2) o uso dos Storyboards/Cenários facilita o entendimento e a execução de todo o processo de desenvolvimento de um programa interativo feito por uma equipe multidisciplinar. Em um ambiente real de produção de programas interativos, essa equipe poderia ser

formada por analistas de sistemas, programadores, designers, editores e diretores de TV entre outros.

A arquitetura do StoryToCode é inspirada na MDA [Perovich 2009] e está dividida em três partes relacionadas entre si, conforme a Figura 3.2:

1. Storyboard/Cenário, que é considerado na MDA como *Computational Independent Model* (CIM);
2. Diagrama de Elementos (*Components*), o qual, a depender do nível de detalhamento, pode ser chamado na MDA de *Plataform Independent Model* (PIM) ou de *Plataform Specific Model* (PSM) e;
3. Geração de Código (*Code*) para uma plataforma específica. 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.

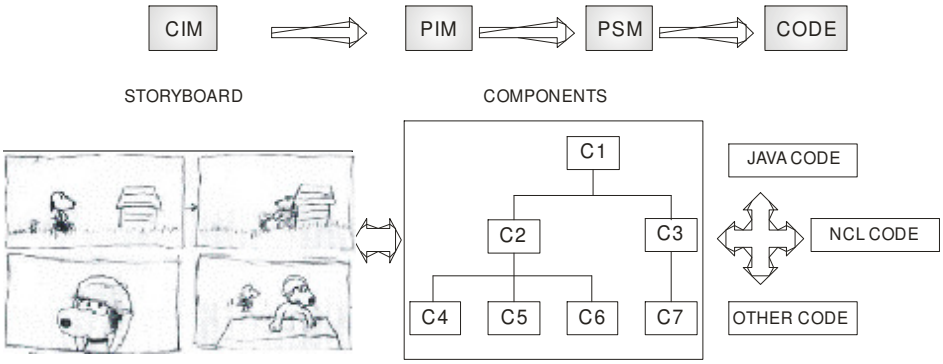

Figura 3.2 - Arquitetura do Modelo StoryToCode

Os Storyboards/Cenários presentes na primeira parte do StoryToCode são produzidos pela equipe de TV e usados como repositório para definição dos requisitos. Esses requisitos deverão ser atendidos na geração do conjunto de elementos de uma aplicação. Na versão atual do StoryToCode, essa geração não é feita de forma automatizada conforme ilustrado na Figura 3.3. Assim, cabe a equipe de projeto de software usar os Storyboards/Cenários para extrair os requisitos e gerar uma representação abstrata do conjunto de elementos contidos nas cenas de um programa, com as suas relações e os seus eventos de interatividade. Esse conjunto de elementos está representado pela segunda parte do StoryToCode.

O conjunto de elementos do StoryToCode caracterizam o *Platform Independent Model* da MDA. 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 em outros modelos (*Platform Specific Model*) 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 (Elementos do StoryToCode), representados através da notação UML e do formato XMI [Yang 2009]. Esses elementos devem ser estendidos para se adequar aos requisitos específicos de um Storyboard/Cenário.

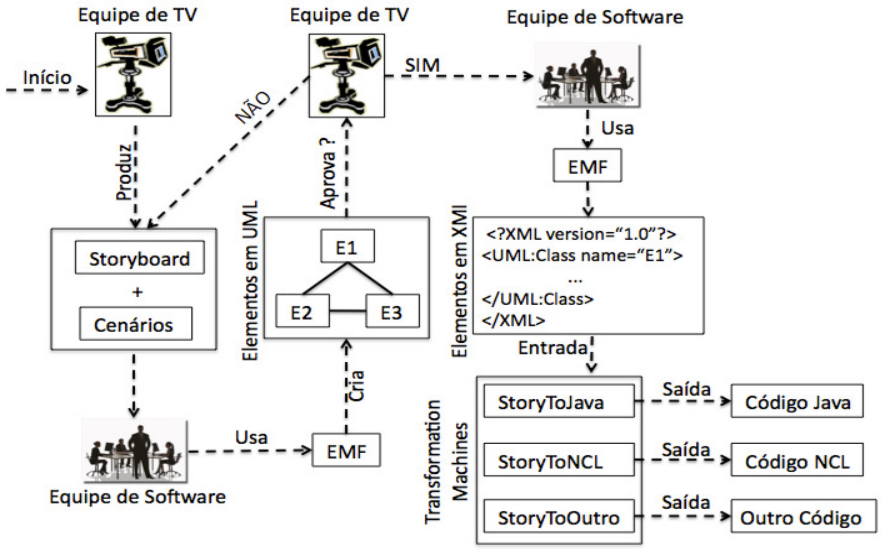

Figura 3.3 – Dinâmica do Processo de Geração de Código no StoryToCode.

O StoryToCode define que cada um dos elementos do modelo deve ser especializado para conter sua própria lista de características a fim de poder abstrair corretamente um Storyboard/Cenário. A transformação inicial dos elementos (especialização) antes da geração de código caracteriza o conceito de *Platform Specific Model* da MDA. Essa capacidade de extensão do modelo permite adequá-lo a qualquer especificação tanto no que diz respeito à estruturação das mídias, quanto dos componentes de software dos Storyboards/Cenários de um programa interativo. A possibilidade de descrever cada um dos elementos que compõem um Storyboard/Cenário com riqueza de detalhes é que permite o uso das transformações para ir reduzindo o nível de abstração até que o código de uma aplicação para um contexto específico seja obtido. Para permitir a especificação/ extensão dos elementos em UML e também em XMI, o StoryToCode usa o arcabouço denominado de *Eclipse Modeling Framework* (EMF) [Steinberg 2009]. Esse arcabouço contém um *parser* XML que permite gerar código XMI a partir de especificações visuais feitas em UML.

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. A Figura 3.3 exibe toda a dinâmica do processo de geração de código a partir de Storyboards/Cenários no modelo StoryToCode. O objetivo específico da terceira 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* (um *parser*). Ele recebe

como entrada o conjunto de elementos (em XMI) e gera como saída o código para uma plataforma específica.

O primeiro passo dessa etapa do StoryToCode, é usar o EMF na tarefa de obter automaticamente os dados de entrada (código XMI dos elementos representados em UML) para um *transformation machine*. O uso do formato XMI, que é independente de plataforma, é um requisito de entrada para o processo de geração de código no StoryToCode. Assim, depois de gerar o arquivo XMI e de carregá-lo na memória, a equipe de software escolhe uma das instâncias disponíveis de *transformation machine* para gerar automaticamente o 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, um único conjunto de elementos pode ser transformado em código para plataformas diferentes. Para isso deve existir uma instância do *transformation machine* específica para cada plataforma.

As principais vantagens do uso do StoryToCode são:

- Permitir o projeto e implementação de uma aplicação (conjunto de componentes interativos), independente do contexto, levando em consideração as particularidades de um programa interativo de TV;
- Diminuir a responsabilidade do gerador de conteúdo através da descentralização das etapas de produção que estão fora do seu universo de trabalho original: a especificação e implementação de um artefato de software;
- Permitir a participação de outros atores (analistas de sistemas, programadores, designers e etc.) no processo produtivo de um programa interativo de TV;
- Diminuir do esforço despendido durante o processo de produção dos componentes interativos.
- Permitir o reuso dos componentes de software, criados para TV, em outros ambientes.
- Permitir o reuso de componentes de software, criados em outros domínios, no ambiente de TV.

Após a análise detalhada das questões ligadas à concepção de aplicações interativas para TV digital apresentada nas primeiras seções, a sequência do texto irá tratar da infraestrutura disponível no receptor de TV para a execução dessas aplicações.

3.5. O *middleware* Ginga

O Ginga é a especificação de *middleware* do SBTVD, resultado da integração das propostas FlexTV [Leite 2005] e MAESTRO [Soares 2006], desenvolvidas por consórcios liderados pela UFPB e PUC-Rio no projeto SBTVD , respectivamente.

O FlexTV, proposta inicial de *middleware* procedural do SBTVD, incluía um conjunto de APIs compatíveis com outros padrões além de funcionalidades inovadoras, como a possibilidade de comunicação com múltiplos dispositivos, permitindo que

diferentes usuários pudessem interagir com uma mesma aplicação interativa a partir de dispositivos remotos. Já o MAESTRO foi a proposta inicial de *middleware* declarativo do SBTVD. O foco era oferecer facilidade do sincronismo espaço-temporal entre objetos multimídia, utiliza a linguagem declarativa NCL e agregar as funcionalidades da linguagem de script da linguagem Lua.

O Ginga integrou estas duas soluções, chamadas de Ginga-J [Souza Filho 2007] e Ginga-NCL [Soares 2007], tomando por base as recomendações internacionais da ITU [ITU J200 2001]. 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 3.4. O conteúdo imperativo é executado por uma Máquina Virtual Java (JVM).

Figura 3.4 – Visão geral do *middleware* Ginga

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 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 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.

3.6. Ginga-J: O Ambiente Imperativo do Ginga

3.6.1. Java e TV Digital

Java foi criada pela *Sun Microsystems* e define um conjunto de tecnologias, que inclui desde a linguagem de programação, um conjunto de classes organizadas em bibliotecas (a API de Java), um formato de arquivo para as classes (arquivo *class*) e uma máquina virtual, até os ambientes de execução e desenvolvimento, também chamada de plataforma Java

Java pode ser utilizada para se criar vários tipos de aplicativos, desde aplicações *Desktop* até aplicações designadas para serem controladas pelo software que as executa, tais como *Applets* que são carregados pela Web e executados dentro de um browser, *Servlets*, que são designados para serem executados dentro de um servidor de aplicações Web; *Midlets*, designados para serem executados dentro de dispositivos móveis. Para o ambiente de TVD existem os *Xlets*, que são aplicações que são executadas por um *middleware* em terminais de acesso que suportam a API JavaTV.

Uma das propriedades fundamentais que torna Java apropriada como linguagem de programação para TV Digital é a habilidade de construir códigos móveis que podem rodar em máquinas pequenas. A Figura 3.5 ilustra este cenário. O código fonte de uma aplicação para TV Digital geralmente é produzido na emissora usando um ambiente de desenvolvimento integrado (do inglês, IDE *Integrated Development Environment*), o qual muitas vezes torna possível gerar códigos de forma rápida. Tipicamente, a IDE possibilita aos desenvolvedores compilar o código em arquivos de classes móveis. Esses arquivos são transportados para os terminais de acesso utilizando os protocolos de difusão, os quais usam uma máquina virtual Java, especificamente um *ClassLoader*, no *middleware* para executá-los. Uma vez instanciado, o código fonte das aplicações é capaz de acessar classes Java de APIs da JVM ou então acessar código nativo através de *Java Native Interface* (JNI), que dá acesso aos recursos de hardware da plataforma.

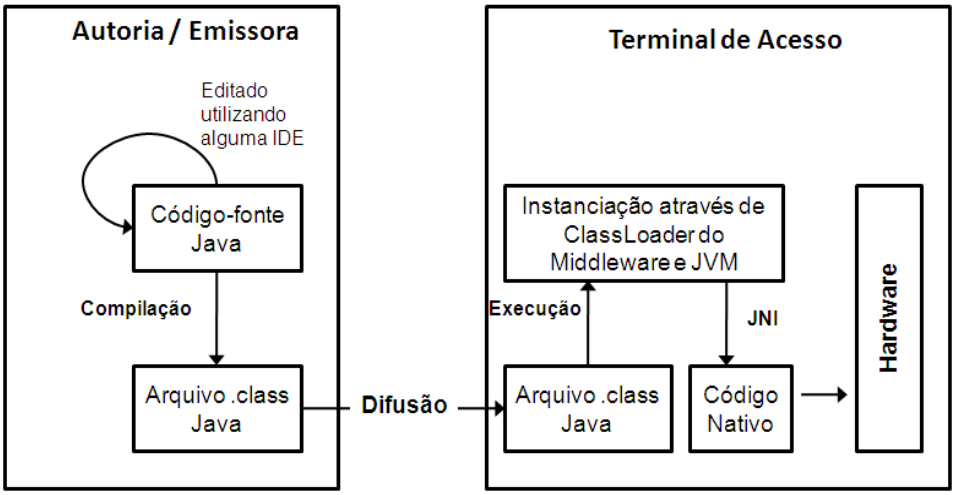

Figura 3.5 – Modelo de programação Java para TV Digital

3.6.2. Visão geral do Ginga-J

Como todo ambiente Java, o Ginga-J é composto por um conjunto de APIs, conforme mostra a Figura 3.6. Tais APIs são definidas para atender funcionalidades necessárias para o desenvolvimento de aplicativos para TVD, desde a manipulação de dados multimídia até protocolos de acesso. Sua especificação é formada por uma adaptação da API de acesso a informação de serviço do padrão japonês [ARIB STD-B.23 2006], pela especificação Java DTV [JavaDTV API 2009], que inclui a API JavaTV, além de um conjunto de APIs adicionais de extensão ou inovação.

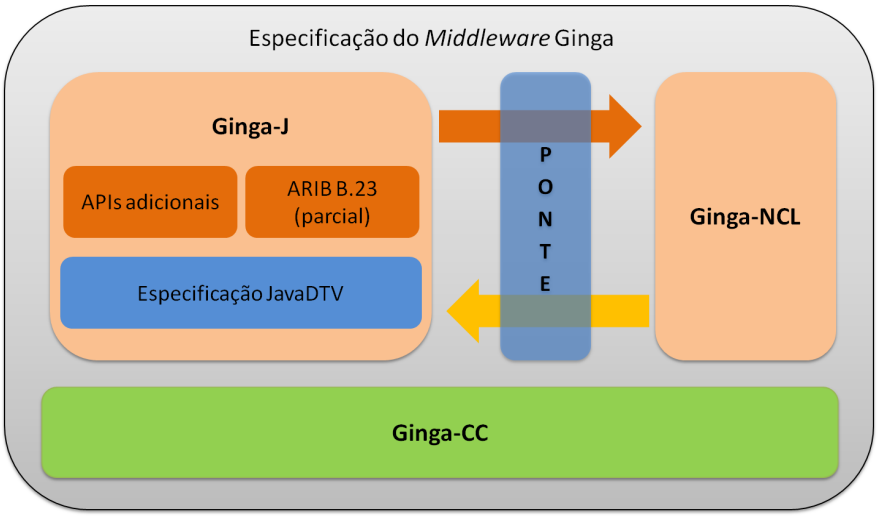

Figura 3.6 – Visão geral do Ginga-J

As APIs adicionais incluem um conjunto de classes disponíveis para a ponte entre os aplicativos NCL e Java, funcionalidades adicionais para sintonia de canais, envio de mensagens assíncronas pelo canal de interatividade e integração de dispositivos externos ao *middleware*, viabilizando a interação simultânea de múltiplos usuários e dispositivos em aplicações de TVD [Silva 2007, Silva 2008].

A especificação Java DTV é uma plataforma aberta e interoperável que permite a implementação de serviços interativos com a linguagem Java, tendo sido inserida recentemente ao conjunto de APIs do Ginga-J. Funcionalmente, a JavaDTV substitui a coleção de APIs definidas inicialmente para o Ginga-J [Souza Filho 2007] e utilizadas pelo padrão GEM (*Globally Executable MHP*) [ETSI TS 102 819], tais como DVB (*Digital Video Broadcast*), DAVIC (*Digital Audio Video Council*) e HAVi (*Home Audio Video Interoperability*). O objetivo da JavaDTV é fornecer uma solução livre de *royalties* para permitir a fabricação de aparelhos de TVD e/ou conversores e desenvolvimento de aplicações com um custo mais acessível. Uma diferença importante da Java DTV em relação ao GEM, é a API LWUIT *(LightWeight User Interface Toolkit*), responsável por definir elementos gráficos, extensões gráficas para TV digital, gerenciadores de *layout* e eventos do usuário.

Adicionalmente, o Ginga-J é composto pela API JavaTV e pelo ambiente de execução Java para sistemas embarcados (JavaME), incluindo a plataforma CDC

(*Connected Device Configuration*), e as APIs dos perfis: FP (*Foundation Profile*) e PBP (*Personal Basis Profile*), conforme mostrado na Figura 3.7.

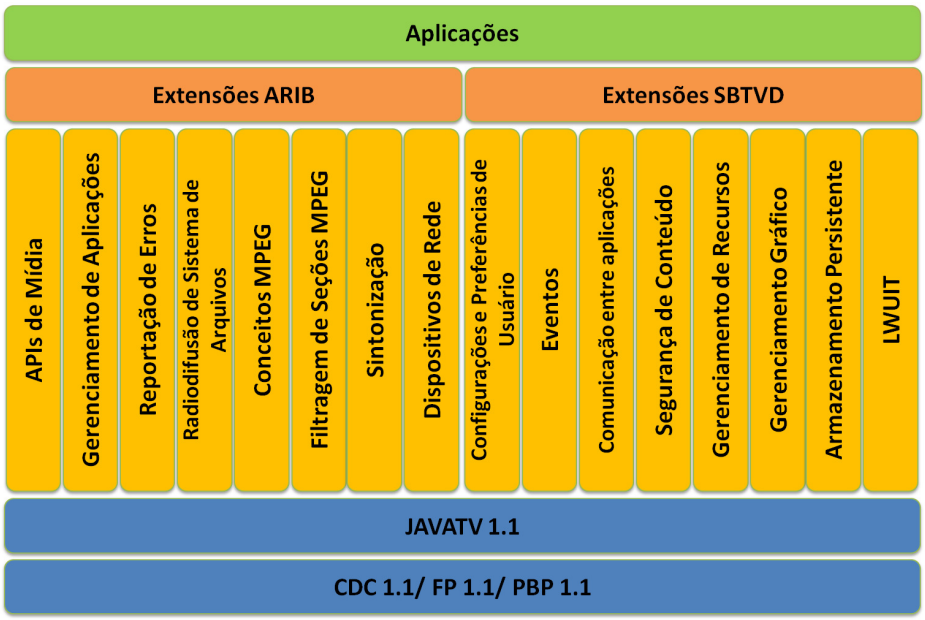

Figura 3.7 – Conjunto de APIs Ginga-J

3.6.3. Pacotes do Ginga-J

A norma Ginga-J [NBR 15606-4 2010] define uma plataforma para execução de aplicações Java para TV Digital. Assim como outras tecnologias Java, uma forma de estudar este ambiente é através das suas APIs, representadas por conjuntos de pacotes. Tais conjuntos são divididos em 7 (sete) partes:

1. Pacotes da plataforma básica Java – representa as funcionalidades de um ambiente Java básico um sistema embarcado baseado nos ambientes CDC [CDC 1.1 2008], FP [FP 1.1 2008] e PBP [PBP 1.1 2008].
2. Pacotes da especificação JSSE 1.0.1 – implementa funcionalidades opcionais de segurança para a plataforma básica de Java para TV Digital, como por exemplo protocolos de transporte seguro [JSSE 1.0.1 2006].
3. Pacotes da especificação JCE 1.0 – implementa outras funcionalidades opcionais de segurança para a plataforma básica de Java para TV Digital, especificamente para operações de criptografia [JCE 1.0.1 2006].
4. Pacotes da especificação SATSA 1.0.1 – permite comunicação com dispositivos externos (normalmente *smartcards*) utilizando o protocolo APDU (do inglês, *Application Protocol Data Unit*) [SATSA 1.0.1 2007].
5. Pacotes da especificação JavaTV 1.1 – implementa o modelo de gerenciamento de aplicações, funcionalidades específicas para TV Digital num grau de abstração maior, além de incluir a API JMF *(Java Media Framewor)* [JavaTV 1.1 2008].

6. Pacotes da especificação JavaDTV 1.3 – extende os pacotes do JavaTV 1.1 para implementar funcionalidades específicas de TV Digital adicionais ou de menor grau de abstração. Também contém os pacotes de APIs gráficas do LWUIT (componentes gráficos, tratamento de eventos do usuário e gerenciador de layout) [JavaDTV 1.3 2009].
7. Pacotes específicos Ginga-J – contém pacotes que implementam funcionalidades exclusivas do sistema brasileiro (controle de planos gráficos, ou que foram herdadas do sistema japonês (acesso a informações de serviço dependente de protocolo) [NBR 15606-4 2010].

As informações mais técnicas sobre cada pacote dessas especificações podem ser encontradas nas referências. A norma Ginga-J [NBR 15606-4 2010] também possui várias descrições desses pacotes.

3.6.4. Emulador Ginga-J

O Emulador Ginga-J[6] é um projeto de código aberto lançado pelo LAVID (Laboratório de Aplicações de Vídeo Digital) da UFPB. Ele consiste num ambiente para execução de aplicações Java para TVD que já é compatível com a nova especificação de APIs do Ginga-J. Por exemplo, esta ferramenta já implementa elementos gráficos e gerenciadores de layout da API LWUIT e vários pacotes da API JavaDTV.

Desenvolvido exclusivamente na linguagem Java, o objetivo do emulador é oferecer um ambiente de execução de aplicações mais simples de instalar e usar num computador pessoal. Porém, devido essa característica a ferramenta apresenta algumas limitações para representar todas as funcionalidades disponíveis num terminal de acesso real. A principal está relacionada a exibição de vídeo e áudio com formatos da alta resolução ou taxa de dados, uma vez que o processamento das mídias é realizada exclusivamente através de software. Outra restrição está relacionada ao uso dos protocolos das redes de difusão, como sinalização de aplicações, seleção de fluxos elementares, acesso a informações de serviço e carrossel de dados, já que a execução das aplicações é realizada localmente. Entretanto, através do emulador é possível realizar prototipação rápida da interface gráfica do usuário para aplicações de TVD, permitindo testar aspectos de disposição, cores, forma, navegação e interação com o usuário.

A arquitetura do Emulador Ginga-J é baseada no Xletview[7], que implementa APIs do GEM. Foram reutilizados os seguintes elementos: (1) interface gráfica com o usuário; (2) gerenciador de aplicações; (3) tratamento de eventos do usuário; (4) modelo das camadas gráficas de um terminal de acesso.

O elemento (1) da arquitetura, ilustrado na Figura 3.8, consiste de um ambiente com: (a) uma área para execução das aplicações que emula uma visor de TV; (b) um controle remoto virtual; (c) menu para acesso as funcionalidades do gerenciador de aplicações.

---

[6] Emulador Ginga-J– Disponível em: http://dev.openginga.org/projects/gingaj-emulator

[7] XletView – Disponível em: http://www.xletview.org

Para o elemento (2) foi possível reaproveitar todo o código, pois o Ginga-J utiliza o mesmo modelo de aplicações (JavaTV Xlets) do GEM.

Para os elementos (3) e (4) da arquitetura, o reuso de código foi possível pois o XletView utiliza as bibliotecas AWT e Swing do ambiente gráfico do *Java Standard Edition* (JSE). No caso, é utilizado mecanismo de tratamento de eventos de eventos do AWT para captura e notificações de eventos do usuário (através de cliques no controle remoto virtual ou pressionamento de teclas do teclado do computador) e a classe ***java.swing.JLayedPlane*** para representar as camadas de planos gráficos de um terminal de acesso.

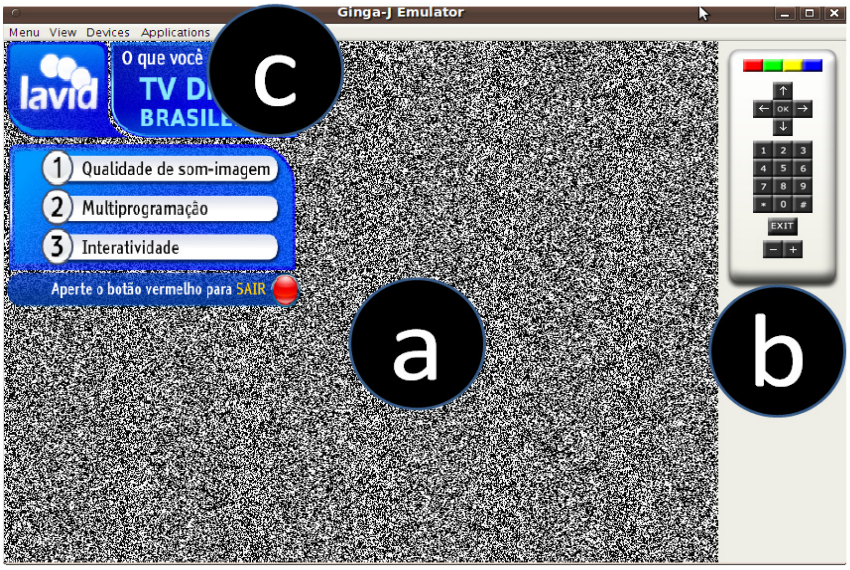

Figura 3.8 – Interface gráfica do Emulador Ginga-J

Outra estratégia de desenvolvimento empregada foi a substituição de artefatos (classes e interfaces) do *XletView* aderentes ao GEM por elementos de equivalência funcional aderentes ao JavaDTV no Emulador Ginga-J. Em outras palavras, algumas classes/interfaces do Ginga-J apenas mudam de nome de classe e assinaturas de métodos em relação ao GEM, tornado possível a reutilização de métodos privados e trechos de código de método. Nesse processo a maior dificuldade foi retirar dependências (através de herança, composição, agregação, referência direta etc.) entre classes do GEM e pacotes internos do XletView. Contudo, para alguns pacotes do JavaDTV foi necessária um implementação sem nenhum reuso do código, como por exemplo as APIs do LWUIT, que não possui equivalência funcional com HAVi (utilizado no *XletView*).

Como requisitos adicionais implementados no Emulador Ginga-J podemos citar: (1) melhor integração da camada gráfica de vídeo com a camada gráfica da aplicação (diminuição do efeito de *flicker*); (2) capacidade de reprodução de múltiplos arquivos de áudio e vídeos nas aplicações; (3) possibilidade de configurar e trocar os vídeos da camada gráfica de vídeo que representam um canal de TV e (4) integração com dispositivos externos. Este último requisito permite o desenvolvimento de aplicações

utilizando a API de Integração de Dispositivos[8][Silva 2007] e é relevante, uma vez que não está presente em nenhum outro ambiente de execução de aplicações para TVD disponível atualmente.

3.6.5. Exemplo de Aplicação Ginga-J

Esta seção procura explorar um exemplo da construção de uma aplicação imperativa utilizando o Ginga-J. A aplicação consiste num serviço que permite obter informações adicionais do conteúdo exibido no programa (similar aos menus interativos disponíveis em DVDs). A figuras 3.9 e 3.10 ilustram a tela inicial (lado esquerdo) e a tela com o menu de opções da aplicação (lado direito), respectivamente.

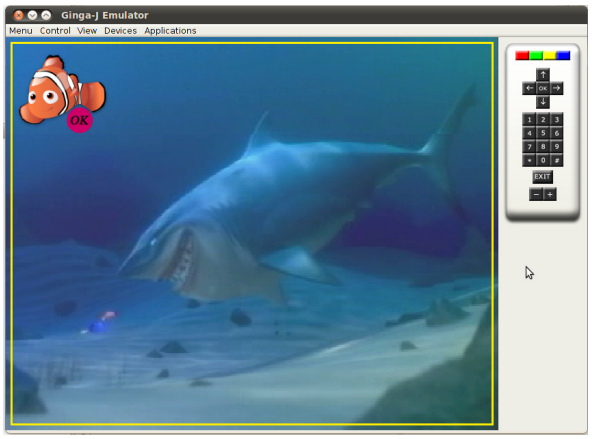

Figura 3.9 – "Tela Inicial"

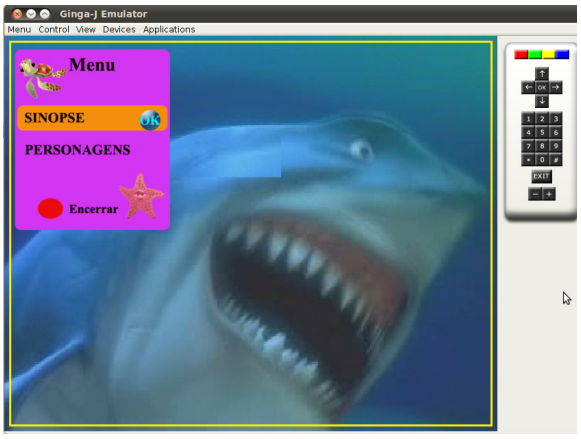

Figura 3.10 – "Tela Menu 1"

A seguir são descritos os passos definidos para o desenvolvimento da aplicação. O objetivo é explicar de forma geral o desenvolvimento de uma aplicação Ginga-J, o código-fonte e a documentação técnica completa do exemplo estão disponíveis na página do projeto do Emulador Ginga-J (ver nota de rodapé 6 na página 18).

Passo 1- Definição de um Xlet

Todas as aplicações Ginga-J devem conter uma classe implementando a interface ***javax.microedition.xlet.Xlet*** (ver PBP 1.1 2008), que deve ser instanciada de acordo com as definições de sinalização de aplicação (ver NBR 15606-3:2007, 12.16). Caso contrário, a classe (o objeto que representa a instância da aplicação) será ignorada quando enviado para um terminal de acesso. Dessa forma, é possível executar a aplicação num ambiente orientado a serviços e mantidas por um gerenciador de aplicações do *middleware*, que garante cada aplicação acesse seu ambiente de execução através de um contexto de serviço (representado por uma instância da classe ***javax.microedition.xlet.ServiceContext***).

Considerando que a aplicação define uma classe que implementa ***javax.microedition.xlet.Xlet***, esta classe deve conter no mínimo 4 (quatro) métodos da interface que permite que a plataforma gerencie seu ciclo de vida e envie informações

[8] É importante lembrar que a API de *Interaction Devices* ainda está em discussão como proposta para o Ginga-J no Fórum SBTVD e ainda não aprovada até o momento de escrita desse texto.

ou eventos de mudanças. O trecho de código a seguir a estrutura inicial de qualquer aplicação Ginga-J: uma classe que implementa 1 (uma) interface e contém obrigatoriamente 4 (quatro) métodos.

```
import javax.microedition.xlet.*;

public class Xlet1 implements Xlet{
   private XletContext context = null;

   public void initXlet(XletContext xletContext)
        throws XletStateChangeException {}

   public void startXlet() throws XletStateChangeException {}

   public void pauseXlet(){}

   public void destroyXlet(boolean flag) throws XletStateChangeException {       }
}
```

A partir daí é necessário implementar cada método (***initXlet, startXlet, pauseXlet e destroyXlet***) de acordo com o seu papel no ciclo de vida da aplicação. Detalhes sobre o papel de cada método podem ser encontrados na norma Ginga-J ou na API JavaTV.

Passo 2- Configuração dos planos gráficos

As aplicações Ginga-J conseguem obter acesso de forma genérica aos planos gráficos oferecidos pelo terminal de acesso, para configuração e exibição de conteúdo de acordo com um modelo de camadas na tela do dispositivo. De acordo com a NBR 15606-1 (2010), a organização dos planos ou camadas gráficas é feita conforme a Figura 3.11.

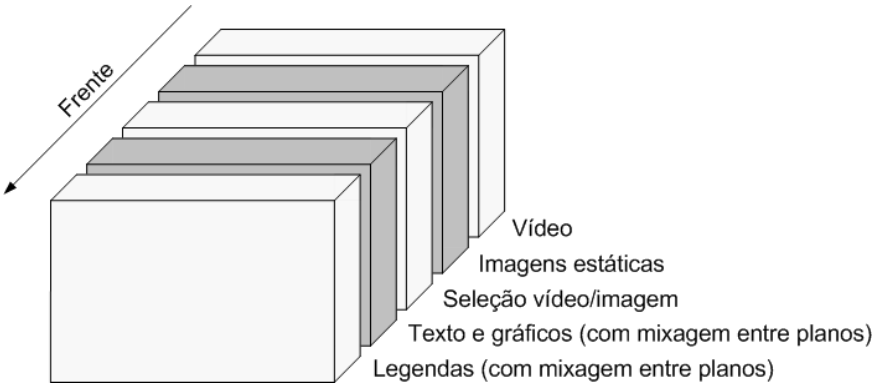

Figura 3.11 – Estrutura de camadas gráficas de um terminal de acesso.
Fonte: [NBR 15606-1 2010]

Com exceção do plano de legendas, todos os planos são acessíveis por aplicações Ginga-J, sendo uma característica nativa de um terminal de acesso. Portanto, existem 4 (quatro) planos sobre os quais uma aplicação Java pode operar:

1. Plano[0]: Plano de texto e gráficos;
2. Plano[1]: Plano de seleção vídeo/imagem;

3. Plano[2]: Plano de imagens estáticas;
4. Plano[3]: Plano de vídeo.

Para cada um destes planos, é permitido obter suas características e efetuar operações gráficas sobre eles. O Ginga-J, através de pacotes da JavaDTV e pacotes específicos SBTVD, define abstrações (classes e interfaces) que implementam este modelo gráfico do terminal de acesso. A Figura 3.12 resume esses elementos:

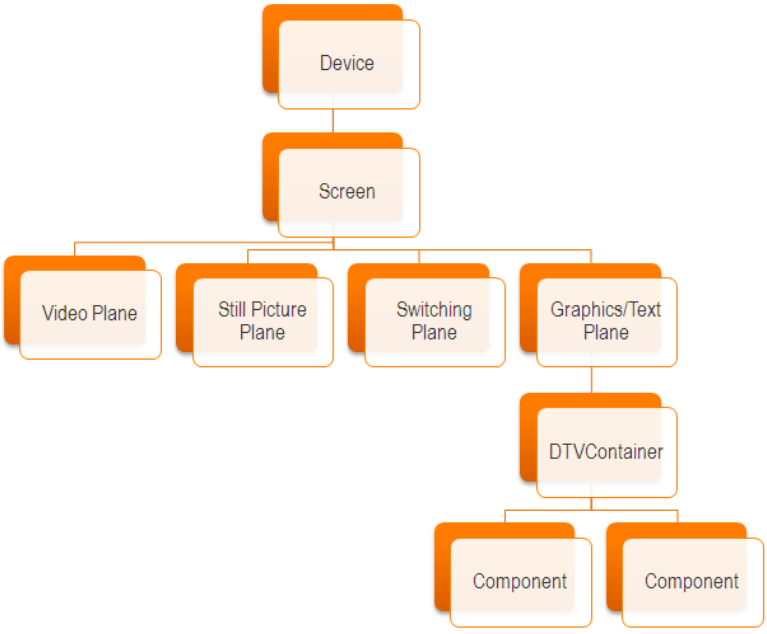

Figura 3.12 – Hierarquia de abstrações representando o modelo gráfico no Ginga-J

A classe ***com.sun.dtv.ui.Device*** (representa o dispositivo no qual o Ginga está instalado) dá acesso a uma ou mais classes ***com.sun.dtv.ui.Screen*** (representa cada tela disponível para exibição dos gráficos da aplicação). Esta classe permite acessar todos os planos do terminal, retornando instancias da classe ***com.sun.dtv.ui.Plane*** com os respectivos identificadores específicos (***Video Plane, Still Picture Plane, Switching Plane e Graphics/Text Plane***). Para cada plano é possível verificar suas características (***com.sun.dtv.ui.Capabilities***) e obter uma classe ***com.sun.dtv.ui.DTVContainer*** que é o componente que suporta todos os tipos de ***com.sun.dtv.lwuit.component*** e operações gráficas definidas na API do LWUIT (por exemplo, ***com.sun.dtv.lwuit.Form***).

A classe ***ResourceComponents*** é apenas uma classe auxiliar (pertence a própria aplicação e não faz parte da especificação Ginga-J) para encapsular as classes que permitem configurar (***Device, Screen, Plane*** e ***Capabilities***) e desenhar elementos gráficos na tela do terminal do acesso (***Form*** e ***DTVContainer***), conforme ilustrado no trecho de código a seguir.

```
public void initXlet() throws XletStateChangeException {

               ResourceComponents.init();
               Device device = Device.getInstance();
               Screen currentScreen = device.getDefaultScreen();

               ResourceComponents.manager.addUserInputEventListener(this,
                    ResourceComponents.anyColoredKeyTyped);

               Plane[] planes = currentScreen.getAllPlanes();

               for(int i=0; i<planes.length; i++) {
                      Capabilities cap = planes[i].getCapabilities();
                         if (cap.isGraphicsRenderingSupported()) {
                                    ResourceComponents.plane = planes[i];
                                    ResourceComponents.planeSetup =
                                       ResourceComponents.plane.getCurrentSetup();
                                    break;
                         }
               }
               ResourceComponents.form = new Form();
               ResourceComponents.dtvcontainer =
               DTVContainer.getDTVContainer(ResourceComponents.plane);
               ResourceComponents.dtvcontainer.setLayout(null);
               ResourceComponents.dtvcontainer.addComponent(ResourceComponents.form);
               ResourceComponents.dtvcontainer.setVisible(true); (…)
```

Passo 3 – Reserva e configuração de recursos limitados

A API JavaDTV fornece, através do pacote ***com.sun.dtv.resources***, um *framework* básico para gerenciamento de recursos limitados em terminais de acesso. Consequentemente, para acessar qualquer recurso limitado é necessário fazer uma solicitação de reserva previamente e após o seu uso ou configuração realizar uma liberação. É importante lembrar que a aplicação também deve possuir a permissão para acessar determinados recursos.

As seguintes classes são recursos limitados, ou seja implementam a interface ***com.sun.dtv.resources ScarceResource*** no Ginga-J: filtros de seções MPEG-2 (***com.sun.dtv.filtering.DataSectionFilterCollection***), eventos de entrada do usuário (***com.sun.dtv.ui.event.UserInputEvent, com.sun.dtv.ui.event.KeyEvent, com.sun.dtv.ui.event.MouseEvent, com.sun.dtv.ui.event.RemoteControlEven***), dispositivos de rede (***com.sun.dtv.net.NetworkDevice***), telas (***com.sun.dtv.ui.Screen***) e sintonizador (***com.sun.dtv.tuner.Tuner***).

O trecho de código abaixo exemplifica a reserva da escuta de eventos de qualquer tecla colorida do controle remoto (através da chamada do método ***reserve***()). No exemplo, ***anyColoredKeyTyped*** é uma instância da classe ***com.sun.dtv.ui.event.RemoteControlEvent***.

```
ResourceComponents.anyColoredKeyTyped =
               new RemoteControlEvent(null, java.awt.event.KeyEvent.KEY_TYPED,0, 0,
                    RemoteControlEvent.VK_COLORED, KeyEvent.CHAR_UNDEFINED);
try {
       ResourceComponents.anyColoredKeyTyped.reserve(true, -1, null);
} catch (…)
```

Passo 4 – Tratamento de Eventos de Entrada do Usuário

Depois de configurar as camadas gráficas para desenho dos elementos da interface gráfica do usuário e realizar a reserva de recursos limitados do terminal, o próximo passo é definir o esquema de navegação de aplicação através de tratamento de eventos de entrada do usuário.

No Ginga-J o mecanismo de tratamento de eventos de entrada do usuário é oferecido através de componentes específicos de TVD (pacote ***com.sun.dtv.ui.event***) e componentes mais gerais providos pela LWUIT (pacote ***com.sun.dtv.lwuit.events***). A classe que trata os eventos gerados por componente gráficos é a ***com.sun.dtv.ui.event.UserInputEventManager***.

Uma boa prática é definir uma classe para cada "tela" da aplicação e nela encapsular o tratamento dos eventos do usuário que vai definir o comportamento da aplicação de acordo com o evento de entrada do usuário (por exemplo, ir para a próxima "tela" ou voltar para a "tela" anterior).

Na sequência do texto trechos de código da tela inicial da aplicação exemplo são ilustrados. As 5 (cinco) primeiras linhas do trecho de código servem para a criação de um objeto ***UserInputEventManager*** a partir de uma referência para uma ***Screen*** (***currentScreen***) e o cadastro (método ***addUserInputEventListener)*** do objeto corrente (o objeto da Tela Inicial) como ***Listener*** (implementa a interface ***com.sun.dtv.ui.event.UserInputEventListener***) do ***UserInputEventManager*** e configurado para receber apenas eventos das teclas coloridas (já explicado no passo 3 deste seção). Já o método ***userInputEventReceived*** é definido na interface do ***Listener*** e é chamado quando algum evento é disparado em componentes gráficos da aplicação. No exemplo é verificado se o evento gerado é de uma tecla de ENTER. Caso afirmativo, a "Tela" atual é apagada e se abre uma nova "Tela" (representado pela classe ***Menu1***).

```
(…)
ResourceComponents.manager =
       UserInputEventManager.getUserInputEventManager(currentScreen);
ResourceComponents.manager.addUserInputEventListener((UserInputEventListener) this,
ResourceComponents.anyColoredKeyTyped);

(…)

public void userInputEventReceived(UserInputEvent inputEvent) {
        com.sun.dtv.ui.event.KeyEvent e = (com.sun.dtv.ui.event.KeyEvent) inputEvent;
        int type=e.getID();
        int code = e.getKeyCode();

        if (type == KeyEvent.KEY_PRESSED) {
              if (code == KeyEvent.VK_ENTER) {
                       this.clear();
                        Menu1 menu1 = new Menu1();
                        menu1.init();
               }
        }
}
```

Passo 5 – Desenho da interface gráfica com o usuário

A etapa final da construção da aplicação consiste em desenhar os elementos gráficos da aplicação nas "Telas" de navegação.

O Ginga-J, através do pacote ***com.sun.dtv.lwuit***, define o mecanismo de composição de elementos gráficos (***com.sun.dtv.lwuit.Componen***t e ***com.sun.dtv.lwuit.Container***) da API LWUIT, que segue um modelo idêntico ao das APIs AWT/Swing. Entretanto, ao contrário do AWT/Swing, não é empregado um sistema de janelas de tela cheia. Neste caso é utilizado o modelo de planos gráficos apresentado no passo 2 desta seção. As seguintes interfaces e classes devem ser empregadas seguindo a especificação JavaDTV:

Tabela 2 – Elementos do pacote com.sun.dtv.lwuit
Fonte: [NBR 15606-4 2010]

Classes	Descrição
Classe ***MediaComponent***	Possibilita a inserção e controle de conteúdo de mídia rica
Classe ***StaticAnimation***	Uma imagem capaz de animação
Classe ***Graphics***	Abstrai a plataforma de contexto gráfico, permitindo a portabilidade entre dispositivos.
Classe ***Container***	Implementa o padrão de projeto ***Composite*** para ***com.sun.dtv.lwuit.Component*** de um modo que permite arranjar e relacionar ***components*** utilizando um arquitetura de gerenciadores de layout plugáveis
Classe ***ComboBox***	Elemento gráfico que representa uma lista que permite apenas uma seleção por vez através da escolha do usuário
Classe ***Font***	Uma simples abstração de fontes de plataforma e de biblioteca que permite o uso de fontes que não são suportadas pelo dispositivo
Interface ***Painter***	Esta interface pode ser usada para desenhar em componentes de fundo de tela
Classe ***RadioButton***	Tipo específico de ***com.sun.dtv.lwuit.Button*** que mantém um estado de seleção exclusivamente de um ***com.sun.dtv.lwuit.ButtonGroup***.
Classe ***Calendar***	Possibilita a seleção de valores de data e hora
Classe ***TabbedPane***	Permite ao usuário alternar entre um grupo de componentes clicando em um guia com um determinado título e/ou ícone
Classe ***Command***	Ação referente aos "*soft buttons*" e menu do dispositivo, similar à abstração de ações do Swing
Classe ***CheckBox***	Botão que pode ser marcado ou desmarcado e ao mesmo tempo exibir seu estado para o usuário
Classe ***Form***	Componente de alto nível que é classe base para interfaces gráficas com o usuário da LWUIT. O contêiner é dividido em três partes: ***Title*** (barra de títulos localizada normalmente na parte superior), ***ContentPane*** (espaço central para disposição de elementos entre ***Title*** e ***MenuBar***) e ***MenuBar*** (barra de menu localizada normalmente na parte inferior)
Classe ***Dialog***	Um tipo de ***Form*** que ocupa uma parte da tela e aparece como uma entidade modal para o desenvolvedor
Classe ***Image***	Abstração da plataforma que trata imagens, permitindo manipulá-

	las como objetos uniformes
Classe ***TextField***	Componente para recebimento de entrada de texto de usuário que utiliza uma API mais leve, sem utilizar o suporte nativo para texto do dispositivo
Classe ***ButtonGroup***	Esta classe é utilizada para criar um escopo de múltipla exclusão para um conjunto de ***RadioButtons***
Classe ***Label***	Permite exibir *labels* e imagens com diferentes opções de alinhamento, também funciona como classe base para alinhamento da disposição de componentes
Classe ***Button***	Componente base para outros elementos gráficos que são clicáveis
Classe ***List***	Um conjunto (lista) de elementos que são criados utilizando uma ***ListCellRenderer*** e são extraídos através da ***ListModel***
Classe ***Component***	Classe base para todos os elementos gráficos da LWUIT. Utiliza o padrão de projeto ***Composite*** de forma semelhante à relação de ***Container*** e ***Component*** do AWT
Classe ***TextArea***	Componente gráfico que permite entrada de textos com múltiplas linhas editáveis, também permite exibir e editar o texto
Classe ***AWTComponent***	Estende a classe ***com.sun.dtv.lwuit.Component*** como uma variante especial que delega as ações de renderização para ***java.awt.Component***

A classe que implementa a "Tela" Menu1 da aplicação exemplo é mostrada no trecho de código a seguir. Ela é construída e configurada uma classe ***com.sun.dtv.lwuit.Label*** que possui uma ícone representado por uma instancia da classe ***com.sun.dtv.lwuit.Image***. Por fim, o ***labelMenu1*** é adicionado a uma instância de ***com.sun.dtv.lwuit.Form*** que por sua vez é exibido na tela (***método repaint()***).

```
import com.sun.dtv.lwuit.Form;
import com.sun.dtv.lwuit.Image;
import com.sun.dtv.lwuit.Label;
(…)
        Label labelMenu1;

        FileInputStream menu1 = new FileInputStream( new File(Imagens.menu1) );

                                labelMenu1 = new Label();
                                labelMenu1.setIcon( Image.createImage(menu1) );

                labelMenu1.setX(0);
                labelMenu1.setY(0);
                labelMenu1.setSize(new Dimension(Imagens.WIDTH ,Imagens.HEIGTH));

                form.removeAll();
                form.addComponent(ResourceComponents.labelMenu1);
                form.repaint();
(…)
```

3.7. Comentários Finais

Este capítulo descreveu uma introdução ao desenvolvimento de aplicações imperativas para TV Digital utilizando o Ginga. Foram apresentados conceitos de TV Interativa, convergência digital, os principais componentes de sistema de TV Digital, tipos aplicações, processo e metodologias de desenvolvimento de software na área, o *middleware* Ginga, o Emulador Ginga-J e, na última seção, a construção passo a passo

de uma aplicação Java para TV Digital. O objetivo não foi esgotar o assunto, mas levantar questões importantes e divulgar a tecnologia.

Como foi observado no decorrer no capítulo, o desenvolvimento de software para TV Digital traz consigo vários desafios e oportunidades para os desenvolvedores de software. O mercado caracterizado por produtores de conteúdo audiovisual e produtores de software devera endereçar esses novos desafios.

Em contraste a maioria dos softwares para computador, as aplicações interativas precisam de um conjunto de componentes complexos que precisam trabalhar sob uma série de circunstâncias específicas, de modo que estes softwares precisam de novos modelos de processo de desenvolvimento e novas ferramentas para endereçar suas novas necessidades.

Nesse sentido, se torna prioritário um esforço de desenvolvimento de processos e ferramentas da Engenharia de Software aplicados ao contexto de aplicações interativas. [Gawlinski 2007] caracteriza essa problemática na seguinte afirmação:

> *O problema com as aplicações de televisão interativa é que elas têm toda a complexidade das aplicações de computadores, mas estão rodando em uma plataforma que normalmente não tem problemas técnicos ou de usabilidade. Os aparelhos de televisão dos telespectadores não falham, o controle remoto sempre funciona (a menos que a bateria acabe) e, ao contrário da internet, a imagem não carrega mais lentamente nos horários de pico de visualização. Qualquer serviço de televisão interativa que cause problemas com o aparelho de televisão ou faça coisas inesperadas provavelmente não será tolerado pelos telespectadores.*

Este capítulo abordou os pontos-chave para construção de aplicações interativas na TV Digital brasileira, bem como, mostrar as oportunidades para o profissional de software nesse contexto. É possível afirmar que este pode ser visto como um novo nicho de atuação para o profissional de software e suas competências especialmente nas iniciativas de modelos de processo, ferramentas e técnicas de desenvolvimento de software.

3.8. Referências

ABNT NBR 15606-2 (2007) "Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 2: Ginga-NCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações". *NBR 15606-2.*

ABNT NBR 15606-4 (2010) "Televisão digital terrestre - Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 4: Ginga-J - Ambiente para a execução de aplicações procedurais". *NBR 15606-4*.

ABNT NBR 15606-5 (2008) "Televisão digital terrestre — Codificação de dados e especificações de transmissão para radiodifusão digital Parte 5: Ginga-NCL para receptores portáteis – Linguagem de aplicação XML para codificação de aplicações". *NBR 15606-5.*

ARIB STD-B23 (2006) “Application Execution Engine Platform for Digital Broadcasting”. *ARIB Standard B23*.

CDC 1.1 (2008), “*Connected Device Configuration 1.1 - JSR 218*”. Sun Microsystems, disponível em http://jcp.org/en/jsr/detail?id=218

CESAR P; GEERTS D.; CHORIANOPOULOS K. (2009). “*Social Interactive Television: Immersive Shared Experiences and Perspectives*”. IGI Global, 1 ed, 2009.

CHORIANOPOULOS, K. (2004). “Virtual Television Channels: Conceptual Model, User Interface Design and Affective Usability Evaluation”. *Phd Thesis*. Dep. of Management Science and Technology Athens, University of Economics and Business. 2004. p. 180.

CPqD. (2006). “Modelo de referência: Sistema Brasileiro de Televisão Digital Terrestre.

ETSI TS 102 819 V1.3.1 (2005) “Digital Video Broadcasting (DVB): Globally Executable MHP (GEM) Specification 1.0.2”. European Telecommunications Standards Institute, *TS 102 819 V1.3.1*

FP 1.1 (2008) “*Foundation Profile 1.1 - JSR 219*”. Sun Microsystems, disponível em http://jcp.org/en/jsr/detail?id=219

Gawlinski, M. (2003). “*Interactive television production*”. Oxford: Focal Press, 2003.

ISO/IEC 13818-6 (1998) “Information Technology – Generic coding of moving pictures and associated audio information – Part 6: Extensions for DSMCC”. *ISO/IEC 13818-6*

ITU-T Recommendation J.200 (2001) “Worldwide common core – Application environment for digital interactive television services”. *ITU-T J.200*.

JavaDTV API (2009) “*Java DTV API 1.3 Specification*”. Sun Microsystems, disponível em URL: http://java.sun.com/javame/technology/javatv/

JavaTV API (2008) “*Java TV Specification 1.1 - JSR 927*”. Sun Microsystems. Disponível em URL: http://jcp.org/en/jsr/detail?id=927

JCE 1.0.1 (2006) “*Java Cryptography Extension - Security Optional Package Specification v1.0.1 – JSR 219*”. Sun Microsystems. Disponível em URL: http://jcp.org/en/jsr/detail?id=219

JSSE 1.0.1 (2006) “*Java Secure Socket Extension – Security Optional Package Specification v1.0.1 – JSR 219*”. Sun Microsystems. Disponível em URL http://jcp.org/en/jsr/detail?id=219

Leite, L. E. C., et al. (2005). “FlexTV – Uma Proposta de Arquitetura de middleware para o Sistema Brasileiro de TV Digital”. *Revista de Engenharia de Computação e Sistemas Digitais*. 2005, Vol. 2, pp. 29-50.

Lu, K. Y. (2005). “Interaction Design Principles for Interactive Television”. *Msc. Thesis*, Master of Science in Information Design and Technology. Georgia Institute of Technology. 2005. 202p.

Marques Neto M. C.; Santos, C. A. S. (2008) "An event-based model for interactive live TV shows". *In*: Proc. 16th ACM international conference on Multimedia, pages 845–848, New York, NY, USA, 2008. ACM.

Marques Neto M. C.; Santos, C. A. S. (2009). "StoryToCode: Um Modelo baseado em componentes para especificação de aplicações de TV Digital e Interativa convergentes". *In*: XV WebMedia, 2009, Fortaleza. SBC, 2009. v. 1. p. 59-66.

PBP 1.1 (2008), "Personal Basis Profile 1.1 – JSR 217", disponível em *URL*: http://jcp.org/en/jsr/detail?id=217

Perovich, D.; Bastarrica, M. C.; Rojas, C. (2009). "Model-Driven approach to Software Architecture design". *In*: ICSE Workshop on Sharing and Reusing Architectural Knowledge (May 16 - 16, 2009). IEEE, Washington, DC, 1-8.

Rodrigues R. F.; Soares L. F. (2006) "Produção de conteúdo declarativo para TV Digital". In: SEMISH, 2006.

SATSA 1.0.1 (2007) "Security and Trust Services API for J2ME – JSR 177". Sun Microsystems, disponível em: http://jcp.org/en/jsr/detail?id=177

Schwalb, E. M. (2003). "*iTV Handbook: Technologies and Standards*". Prentice Hall PRT, 2003.

Silva, L. D. N. (2008) "Uma Proposta de API para Desenvolvimento de Aplicações Multiusuário e Multidispositivo para TV Digital Utilizando o middleware Ginga", dissertação de mestrado, Universidade Federal da Paraíba, 2008.

Silva, L. D. N., et al. (2007). "Suporte para desenvolvimento de aplicações multiusuário e multidispositivo para TV Digital com Ginga". *T&C Amazônia Magazine*. N. 12, 2007, pp. 75-84

SILVA, F. P. R. (2010) "Xtation: um ambiente de execução e teste de aplicações interativas para o middleware Openginga", *Dissertação de mestrado*, Universidade Federal da Paraíba, 2010.

Soares, L. F. G. (2006). "MAESTRO: The Declarative middleware Proposal for the SBTVD". *In*: Proc. 4th European Interactive TV Conference. 2006

Soares, L. F. G., et al. (2007) "Ginga-NCL: the Declarative Environment of the Brazilian Digital TV System". *Journal of the Brazilian Computer Society*. 2007, Vol. v12, pp. 37-46.

Soares, L. F. G; Moreno, M. F.; Sant'anna, F. F. G. (2009) "Ginga-NCL: Suporte a Múltiplos Dispositivos". *In*: Simpósio Brasileiro de Sistemas Multimídia e Hipermídia, 2009, Fortaleza. p. 43-50.

Souza Filho, G. L. et al. (2007) "Ginga-J: The Procedural middleware for the Brazilian Digital TV System". *Journal of the Brazilian Computer Society*. 2007, Vol. v12, pp. 47-56.

Steinberg, D. et al. (2009) "*EMF: Eclipse Modeling Framework 2.0*". 2nd ed. Addison-Wesley Professional.

Veiga, E. G.; Tavares, T. A. (2007). "Um Modelo de Processo para o Desenvolvimento de Programas para TV Digital e Interativa baseado em Metodologias Ágeis". *In*: 1o. Workshop em Desenvolvimento Rápido de Aplicações, 2007.

Whitaker, J. 2001. "*Interactive Television Demystified*", McGraw-Hill, 2001.

Yang, X.; Gu, P.; Dai, H. (2009). "Mapping Approach for Model Transformation of MDA Based on XMI/XML Platform". *In*: First Int. Work. on Education Technology and Computer Science – V.2. ETCS. IEEE, Washington, DC, Mar. 2009. 1016-1019.