UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

INTERFACES DE DESENVOLVIMENTO DE APLICAÇÕES PARA TV DIGITAL BASEADO NO MIDDLEWARE MHP

JOEL ALEXANDRE DARÓS

BLUMENAU
2004

2004/2-25

JOEL ALEXANDRE DARÓS

INTERFACES DE DESENVOLVIMENTO DE APLICAÇÕES PARA TV DIGITAL BASEADO NO MIDDLEWARE MHP

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Mauro Marcelo Mattos

BLUMENAU
2004

INTERFACES DE DESENVOLVIMENTO DE APLICAÇÕES PARA TV DIGITAL BASEADO NO MIDDLEWARE MHP

Por

JOEL ALEXANDRE DARÓS

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Mauro Marcelo Mattos. – Orientador, FURB
Membro:	Prof. Nome do professor, FURB
Membro:	Prof. Nome do professor, FURB

Blumenau, 03 de novembro de 2004

Dedico este trabalho primeiramente a Deus e minha família que tem me ajudado, contribuído e incentivado, me dando forças para acreditar e nunca desistir de meus ideais.

AGRADECIMENTOS

Agradeço a Deus, pela possibilidade e oportunidades que tem me dado. A minha família que sempre deram-me forças e incentivos e por me ensinarem a nunca desistir.

A minha noiva Alessandra pela compreensão e dedicação que tem me dado em vários momentos de dificuldade e alegrias.

Aos meus amigos que direta ou indiretamente contribuíram para meu crescimento pessoal e profissional e pelas rodadas de cerveja que tomamos juntos.

Devo manifestar gratidão ao meu orientador que durante o período de elaboração deste trabalho foi extremamente paciente e esclarecedor quanto as minhas dúvidas, sendo conseqüentemente de essencial importância para a conclusão do mesmo.

RESUMO

A digitalização da televisão em todo seu ciclo de vida vem produzindo transformações fundamentais na experiência de assistir e produzir televisão. Do ponto de vista tecnológico, uma das características mais interessantes é a produção e desenvolvimento de serviços de interatividade para o usuário final. Baseado nestas premissas, o presente trabalho demonstra o estudo detalhado da concepção e funcionamento de um aplicativo de *t-commerce* sobre um sistema de TV digital a fim de demonstrar características e funcionalidades deste modelo de desenvolvimento de sistemas.

Palavras chaves: Sistemas Distribuídos; TV Digital; MHP; Java TV; DVB.

ABSTRACT

The interactive TV comes to producing basic transformations in the experience to watching and produce television. In the technological view, one of the most interesting characteristics is the services production and development for final user. Based in these premises this work it demonstrates the detailed study of the conception and working of a t-commerce application on a digital TV system to demonstrate characteristics and functionalities of this model of systems development.

Key-Words: Distributed systems; Digital TV; MHP; Java TV; DVB.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, o modo como as pessoas assistem, interagem e produzem mídia em televisão passa por um estágio de grande evolução. Ainda que tal fenômeno não seja marcante no Brasil, sistemas de TV com transmissão digital e interativos são realidade nos EUA, Japão e muitos países europeus. Os aprimoramentos, tanto para produtores quanto para espectadores de TV Digital, são numerosos, tais como imagem e som com padrão superior, transmissão em *wide-screen*, mais espaço de freqüência para transmissão, e principalmente interatividade e uma realidade que aproxima ainda mais a TV de outros sistemas de informação, como a Internet.

Segundo Bezerra (2004), poucas tecnologias foram tão aguardadas no ambiente mundial de consumo como a televisão digital. Em um momento inicial, seria o fato tão prometido e esperado da convergência entre duas das mais fantásticas invenções do homem: a televisão e o computador. Seria o nascimento de um super terminal de lazer e serviços *on-line*, interativo e pessoal.

Além de melhoria na qualidade da imagem e som digital, o telespectador teria contato com a mais potencial e interessante característica da transmissão digital: os serviços interativos. Através de canais de retorno, é possível ao telespectador ter uma variedade de serviços como: guias de programação eletrônica, vídeo sob demanda, *TV chat*, *TV banking*, *home shopping*, jogos, entre outros.

O advento dos mecanismos de transmissão de conteúdo televisivo por meio de sinais digitais promete modificar a forma pela qual as pessoas assistirão a programas de televisão nas próximas décadas. O motivo pelo qual esta modificação se processará deve-se à forte tendência em expandir o paradigma de televisão digital para a televisão aberta, atualmente analógica em sua totalidade no Brasil. A migração da televisão analógica para a televisão digital dar-se-á por meio de novos equipamentos que vêm sendo desenvolvidos e até mesmo de equipamentos “adaptadores”, como os chamados *set-top box* (STB), responsáveis por converter sinais digitais em analógicos capazes de serem interpretados pelos aparelhos de TV atuais.

Segundo Morris (2002), atualmente os três principais padrões de televisão digital são: o europeu denominado de *Digital Video Broadcasting* (DVB), o norte-americano *Advanced*

Television Systems Committee (ATSC) e o padrão japonês *Integrated Services Digital Broadcasting* (ISDB).

Dentro do padrão DVB, existe o *middleware Multimedia Home Plataform* (MHP) que define um conjunto de tecnologias para aplicações interativas digitais. O MHP visa adaptar a Internet à televisão digital, provendo assim, conteúdo interativo digital no terminal do usuário. MHP possui dois tipos de aplicações: DVB-HTML e DVB-J, sendo a última a mais popular. Esta é escrita em Java usando um conjunto de *Application Program Interfaces* (API) definidas pelo MHP. As aplicações DVB-J são também chamadas de *Xlets*.

Xlets não são aplicativos Java convencionais. *Xlets* assemelham seu comportamento ao de *applets web*, pois possuem seu ciclo de vida bem definido e controlado pelo gerenciador de aplicações do *middleware*. O *xlet* é um tipo de aplicação construída para ser enviada através de *data broadcasting* e executar de forma interoperável em todos os dispositivos ligados a uma rede de TV digital.

Neste trabalho procura-se mostrar um dos principais padrões abertos para TV interativa e digital em desenvolvimento no mundo. Mais especificamente estudar o *middleware* MHP que compõe o padrão DVB de forma descritiva relacionada à concepção de suas especificações técnicas.

Além disso, serão exibidos aspectos e evolução da arquitetura, principais funcionalidades e aplicações e como é possível construir serviços e aplicativos interativos com as tecnologias disponibilizadas hoje no mercado para TV digital.

Utilizar-se-á o desenvolvimento de uma aplicação *xlet* de comércio eletrônico voltada a esta plataforma, de forma a simular o funcionamento de um serviço sendo executado em um STB, utilizando os recursos da API Java TV de forma que o usuário possa interagir através de um simulador como se estivesse em frente a uma TV digital efetuando uma compra pelo controle remoto de sua TV.

1.1 OBJETIVOS DO TRABALHO

O objetivo do trabalho é construir um aplicativo de comércio eletrônico, denominado *t-commerce* (TAURION, 2004), que seja capaz de interagir sobre um ambiente de simulação de uma TV digital baseado no padrão Java TV (JAVA TV, 2001).

Os objetivos específicos do trabalho são:

a) mostrar os conceitos básicos de desenvolvimento de *software* para plataformas de TV digital, focando principalmente no padrão DVB;
b) desenvolver um protótipo de aplicação baseada no *middleware* MHP, a fim de mostrar suas características técnicas e funcionais.

2 FUNDAMENTAÇÃO TEÓRICA

A seguir explana-se importantes conceitos sobre a tecnologia de TV digital, nos quais este trabalho está fundamentado.

2.1 TV DIGITAL

A televisão é uma das principais fontes de informação, entretenimento e cultura. Após a transmissão em cores, a televisão está sofrendo mais uma nova mudança com a digitalização do sinal. Apesar da produção e armazenamento das informações já serem digitais, a transmissão ainda se dá via sinal analógico. Com a transmissão digital não só haverá uma melhora na qualidade da imagem e do som como também possibilitará a transmissão de vários programas em um único canal. Porém a característica mais interessante da TV digital combinada com um canal de retorno é a possibilidade de criar serviços interativos como vídeo sobre demanda, *pay-per-view*, exibir programas esportivos em vários ângulos, *home shopping*, jogos e *chat*.

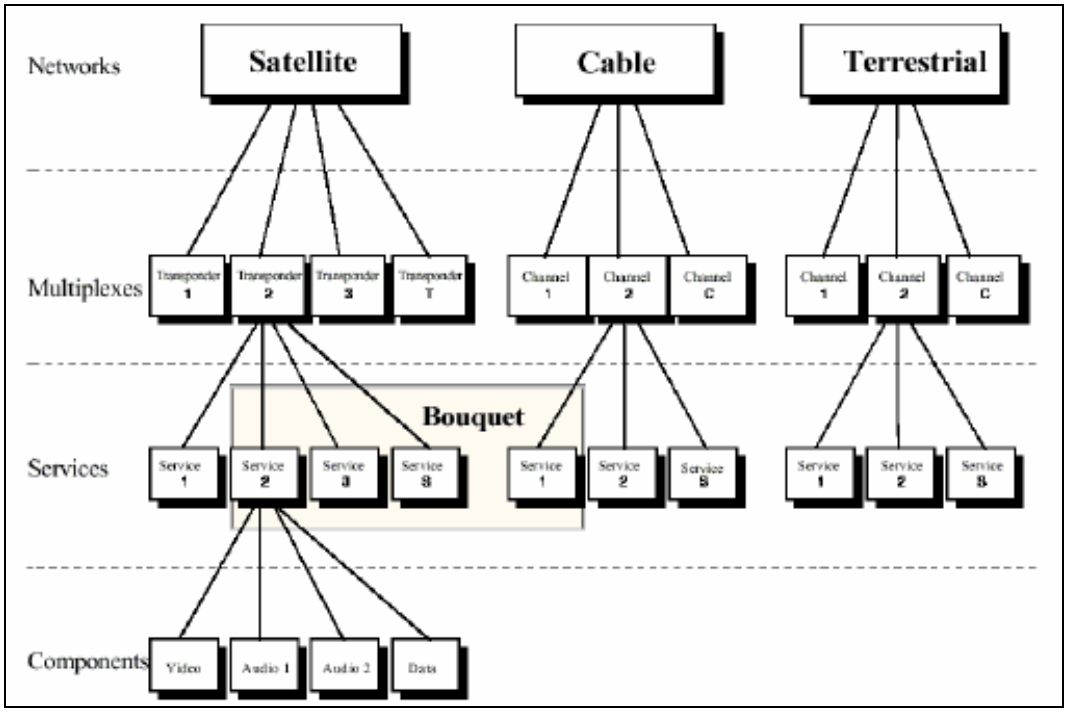
Mudar de TV analógica para digital não é apenas incluir um *chip* no televisor. É uma decisão que envolve muito dinheiro e gente de diversos setores. O governo quer fazer da TV digital um instrumento de inclusão digital, por sua função de educar e informar. As emissoras têm os olhos voltados para as oportunidades de novos negócios. Os fabricantes pretendem comercializar novos produtos digitais, e o consumidor está atraído pelas melhorias de áudio e vídeo.

A TV digital traz novidades positivas. O consumidor vai ter imagem com qualidade de *digital vídeo disc* (DVD), som com qualidade de *compact disc* (CD) e muito mais canais. Para o governo, a TV digital pode servir para incluir a maioria da população na cultura digital, visto que hoje menos de 7% dos lares tem computador, mas quase 90% têm TV, desenvolvendo assim a economia, gerando negócios para os fabricantes e as indústrias de *software*.

2.2 ARQUITETURA DE UM SISTEMA DE TV DIGITAL

A arquitetura geral de um sistema de TV digital introduz uma série de conceitos que apóiam novos modelos de negócios, fundamentais para o sucesso da mudança tecnológica

para a TV digital. A Figura 1 apresenta as principais relações entre estes conceitos, conforme definidos no modelo DVB (DVB, 2004).



Fonte: DVB (2004)

Figura 1 – Arquitetura geral de um sistema de TV digital

No modelo de consumo de serviços de TV digital o elemento de produção de mídia é chamado de evento. Um evento é um agrupamento de *streams* elementares de audio, vídeo e dados com um tempo definido de início e fim, como por exemplo, a primeira parte de uma novela ou o primeiro tempo de uma partida de futebol. Um programa é uma concatenação de um ou mais eventos produzidos por um estúdio, como um capítulo de novela ou um show. Segundo Pawlan (2001), um serviço é uma seqüência de programas controlada por um difusor, que tem por objetivo atingir uma determinada audiência, e que é veiculado em uma determinada faixa de horários. O serviço é a principal unidade de produção e consumo na TV digital. Uma central de produções pode compor um conjunto de serviços produzidos por vários estúdios, formando o que se chama de *bouquet*. O *bouquet* é, portanto, a unidade de distribuição das programações de uma central de produções.

A capacidade de interatividade da TV digital se deve à presença de três elementos: gerador de carrossel, multiplexador e um STB (BROADCAST PAPERS, 2004). O gerador de carrossel é capaz de transformar um conjunto de arquivos de dados em um fluxo elementar, empregando um esquema de transmissão cíclica de dados. O multiplexador é capaz de fundir um ou mais fluxos de dados aos fluxos de áudio e vídeo que compõem os eventos e

programas, os quais por sua vez compõem os serviços consumidos pela audiência. O STB possui capacidade de processamento computacional, sendo capaz de interpretar computacionalmente os fluxos de dados multiplexados. Deste modo o STB executa uma aplicação que exibe na TV uma interface com o usuário. Isto permite à audiência interagir com o programa de TV através do teclado ou controle remoto. Ao entregar à audiência um fluxo de dados localmente computável, a TV digital se torna interativa.

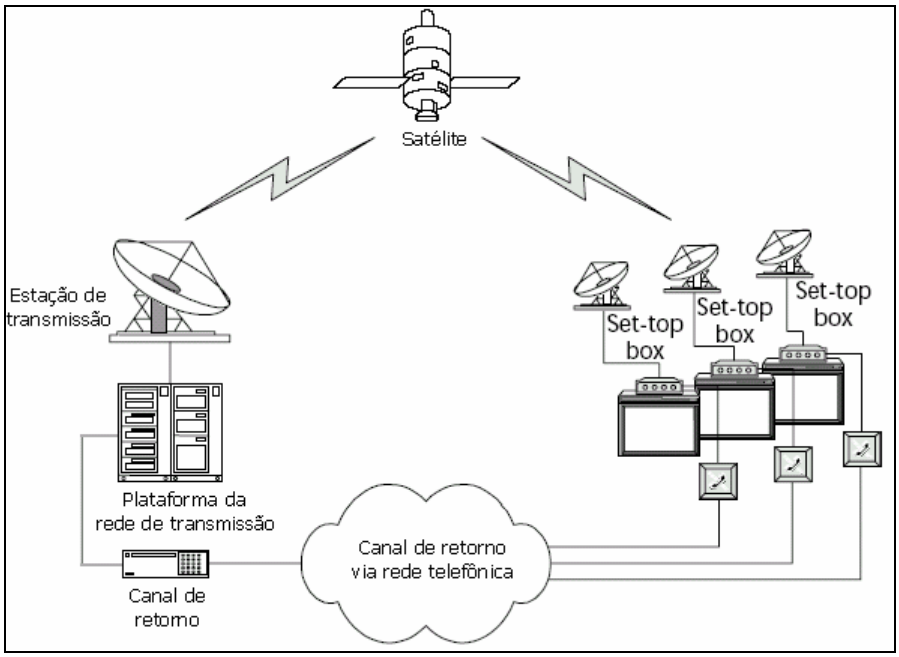
A arquitetura de um sistema digital interativo define a existência de um estúdio especializado, chamado estúdio de dados, que realiza dois processos:

a) produção dados: o que envolve geração de vídeo-texto e páginas HTML;
b) produção aplicações: que envolve desenvolvimento de *software*.

O estúdio de dados disponibiliza seus resultados para o subsistema de transmissão de dados, que contém um gerador de carrossel, responsável por transmitir um fluxo elementar de dados e aplicações a partir dos dados do estúdio. Este fluxo de dados é sincronizado com os fluxos de audio e vídeo. Os fluxos áudio, vídeo e dados são recebidos pelo set-up box. Os dados e programas são interpretados e produzem interações locais, que se dão principalmente através do controle remoto.

Na arquitetura de um sistema de TV digital destaca-se a presença de um provedor de acesso, ao qual o STB conecta-se para enviar e receber dados resultados do processo de interação local. Esta conexão pode ser feita através de um modem ou outro meio alternativo. O provedor de acesso contém um *gateway* para acesso à Internet, e desta forma o STB tem acesso a dados e serviços da Internet. A fim desenvolver um modelo de negócios que produza resultados econômicos mais relevantes, o STB é normalmente direcionado a interagir com um provedor de serviço específico, que oferta um produto ou serviço fortemente relacionado ao conteúdo do evento televisivo produzido na central. As seções que se seguem apresentam maiores detalhes da arquitetura de um STB como também de um gerador de carrossel. Os componentes multiplexador e demultiplexador apóiam praticamente toda a flexibilidade de operação da TV digital e são fundamentais para compreensão da arquitetura deste sistema.

A Figura 2 ilustra a arquitetura do sistema de TV digital DVB via satélite.



Fonte: Andrade Neto (2004, p. 10)

Figura 2 – Componentes de um sistema DVB via satélite.

2.2.1 O Funcionamento de um Set-Top Box

O STB é o elemento que fica na extremidade da arquitetura de um sistema de TV digital. O STB é um pequeno computador dedicado à tarefa de processar fluxos de áudio, vídeo e dados, através da sintonização e demultiplexação do sinal digital. O STB seleciona uma série de *streams* relacionadas a um serviço. As *streams* pertencem a duas categorias: uma relativa a áudio e vídeo e a outra a dados sobre serviços. As *streams* de áudio e vídeo são diretamente enviadas para o controlador de mídias, que as exibirá conforme controles ajustados pelo usuário ou pelas aplicações executadas no STB. Os dados sobre serviço são remetidos ao subsistema de serviços de informação denominado pela sigla SI. Estes fluxos contém informações detalhadas sobre todas as outras *streams* disponíveis para o STB. A partir das informações sobre serviços é possível exibir os dados e executar as aplicações que foram produzidas nos estúdios. A execução de aplicações segue um modelo computacional padronizado, que contém áreas de processo e sistema de arquivos.

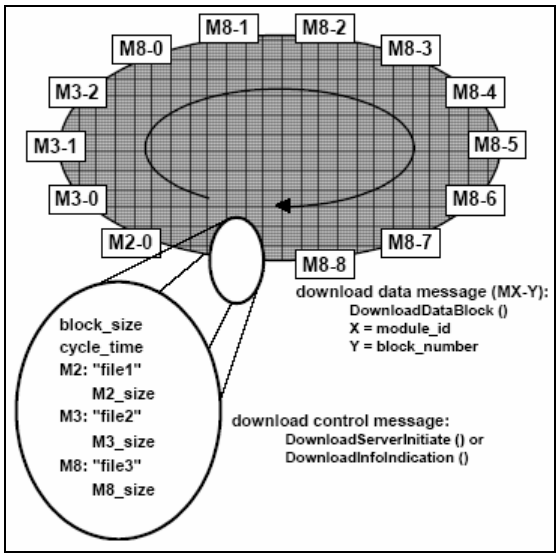
2.2.2 O Gerador de Carrossel

A intenção primária do modelo de carrossel é permitir a instalação dinâmica no STB de uma cópia de um sistema de arquivos produzido no estúdio de dados. Este sistema de arquivos persiste no STB apenas enquanto o serviço está sintonizado. O gerador de carrossel é

o elemento responsável por gerar um *stream* elementar de dados que, recebida pelo STB, produz este efeito. A *stream* carrossel segue o protocolo *Digital Storage Media - Command and Control* (DSM-CC), sub-protocolo DSMCC Object, do padrão MPEG-2. Outras variações do protocolo DSM-CC permitem a transmissão de outros tipos de dados, como fluxos IP e atualização de *firmware*. O nome carrossel se deve ao fato de que os fluxos de dados que geram o sistema de arquivos precisam ser re-transmitidos ciclicamente, a fim de que seja possível a um STB que sintonizou o serviço receber este sistema de arquivos, mesmo após o início da difusão.

O transmissor disponibiliza constantemente de um conjunto distinto de módulos de dados, os repetindo ciclicamente. Para obter dados deste conjunto que ainda não foram recebidos, basta ao receptor aguardar que um novo ciclo do carrossel o transmita.

A Figura 3 ilustra um exemplo de carrossel de dados.



Fonte: Andrade Neto (2004, p. 24)

Figura 3 – Transmissão de um carrossel de dados

Os arquivos a serem transmitidos pela central de produções fazem parte de uma aplicação, que estará sendo transmitida em um fluxo individualmente identificado. Os arquivos são agrupados em módulos e os módulos são divididos em blocos de tamanho fixo, aos quais estão associadas prioridades de re-transmissão. O gerador de carrossel gera continuamente um *stream* contendo os módulos a transmitir, sendo que os módulos de maior prioridade são transmitidos com maior freqüência. O fluxo DSM-CC Object é sincronizado e multiplexado com os outros fluxos que fazem parte do programa a transmitir. Ao serem recebidos pelo STB interativo os arquivos do carrossel podem ter várias finalidades, como

apresentar dados específicos para serem apresentados por um programa de guia de programação eletrônica, conter informações adicionais sobre uma determinada propaganda veiculada, apresentar um teletexto, bem como enviar uma aplicação a ser executada no STB, como um x*let*.

2.2.3 DSM-CC

A especificação DSM-CC define uma série de protocolos que fornecem funções de controle sobre fluxos MPEG, em ambientes de rede heterogêneos (BALABANIAN, 2004). Foi inicialmente concebida como parte da especificação do padrão MPEG-2, consistindo em um protocolo simples cujo objetivo era possibilitar o controle remoto sobre um fluxo MPEG, uma funcionalidade similar a de um aparelho de vídeo-cassete executando suas operações sobre uma fita VHS na rede. Desde então a especificação evoluiu bastante, gerando aplicações na distribuição de serviços audiovisuais interativos, como vídeo sob demanda e TV digital.

Tais aplicações têm sido delineadas pelo *Digital AudioVisual Council* (DAVIC), uma associação sem fins lucrativos de organizações de diversos países, que tem definido especificações de interfaces e protocolos internacionais para as tecnologias emergentes de distribuição de serviços áudio visuais sobre redes. Na especificação DAVIC, DSM-CC é usado como o protocolo padrão de controle sobre sessões de multimídia interativas e os recursos por elas utilizados. O consórcio DVB trabalhou em cooperação com DAVIC, findando por adotar suas especificações no seu padrão de TV digital

2.3 PADRÕES MUNDIAIS DE TV DIGITAL

Um sistema de televisão digital interativa deve adotar e integrar um conjunto de diferentes tecnologias de *hardware* e *software* para implementar suas funcionalidades.

Conjuntamente, estas tecnologias permitem que um sinal eletromagnético, que transporta fluxos de áudio, vídeo, dados e aplicações, possa ser transmitido para o STB e, então, que estes fluxos sejam recebidos, processados e apresentados aos usuários.

Considerando a diversidade de soluções tecnológicas que podem ser adotadas para implementar um sistema de televisão digital interativa, diversos órgãos de padronização

concentraram esforços na especificação de padrões. Como resultado destes esforços, atualmente, existem três padrões mundiais de sistema de televisão digital reconhecidos:

a) DVB - *Digital Video Broadcasting*;
b) ATSC - *Advanced Television Systems Committee*;
c) ISDB - *Integrated Services Digital Broadcasting*.

Estes sistemas adotam diferentes padrões para modulação do sinal de difusão, transporte de fluxos de áudio, vídeo, dados e aplicações, codificação e qualidade de áudio e vídeo e serviços de *middleware*.

A seguir apresentamos uma breve descrição da arquitetura destes principais padrões de sistema de televisão digital, identificando os componentes básicos adotados nestes padrões.

2.3.1 DVB

O projeto DVB (DVB, 2004) é um consórcio iniciado em setembro de 1993 e composto por mais de 300 membros, incluindo fabricantes de equipamentos, operadoras de redes, desenvolvedores de *software* e órgãos de regulamentação de 35 países. O objetivo do projeto DVB é especificar uma família de padrões mundiais para sistemas de televisão digital interativa, incluindo a transmissão do sinal e serviços de dados associados.

A família de padrões especificada pelo projeto DVB caracteriza o padrão de sistema de televisão digital também denominado DVB, que é conhecido como o padrão europeu de televisão digital. O padrão DVB é adotado nos países da União Européia e em outros países como Austrália, Nova Zelândia, Malásia, Hong Kong, Singapura, Índia e África do Sul. A Inglaterra é o país onde a adoção do padrão DVB está mais consolidada, pois já possui mais de um milhão de receptores digitais instalados.

O padrão DVB é formado por um conjunto de documentos que definem os diversos padrões adotados, incluindo aqueles relacionados à transmissão, transporte, codificação e *middleware*.

O padrão DVB permite diversas configurações para a camada de transmissão, cada configuração apresentando uma diferente relação capacidade/robustez. Os principais padrões de transmissão adotados pelo DVB são: DVB-T: (transmissão terrestre por radiodifusão);

DVB-C (transmissão via cabo); DVB-S (transmissão via satélite); DVB-MC (transmissão via microondas operando em freqüências de até 10GHz); e DVB-MS (transmissão via microondas operando em freqüências acima de 10GHz).

Os padrões DVB-T, DVB-C, DVB-S, DVB-MC e DVB-MS adotam diferentes esquemas de modulação. O DVB-T pode operar em canais de 6, 7 ou 8 MHz e adota a modulação *Coded Orthogonal Frequency Division Multiplexing* (COFDM), cuja taxa de transmissão pode variar entre 5 e 31,7 Mbps, dependendo dos parâmetros utilizados na codificação e modulação do sinal.

O DVB-T suporta seis modos de transmissão com resoluções que variam de 1080 à 240 linhas, podendo ser usado para sistemas de alta definição e sistemas móveis de baixa definição. No entanto, alguns estudos apontam que o funcionamento não é satisfatório quando ocorrem transmissões simultâneas para sistemas de alta definição e sistemas móveis.

Na Europa, em um primeiro momento, está sendo utilizada a resolução padrão *standard definition television* (SDTV), inicialmente em formato de tela 4:3. Considerando a largura de banda do canal, a transmissão SDTV permite a difusão de até seis programas simultaneamente. Portanto, o modelo de negócios dos países europeus privilegiou a oferta diversificada de programas e serviços na transmissão terrestre. Por outro lado, a Austrália optou por combinar programas em alta definição e programas em definição padrão.

Nas camadas de transporte e codificação, o padrão DVB é um sistema fundamentalmente baseado no MPEG-2. Portanto, os padrões de transporte e codificação adotados pelo DVB são baseados nas recomendações MPEG-2.

2.3.2 ATSC

O comitê ATSC (ATSC, 2004) é uma organização de padronização americana iniciada em 1982 e composta por cerca de 170 membros, incluindo fabricantes de equipamentos, operadores de redes, desenvolvedores de *software* e órgãos de regulamentação. O objetivo do comitê ATSC é especificar padrões para televisão digital.

O conjunto de padrões especificado pelo comitê ATSC caracteriza o padrão de sistema de televisão digital que é conhecido como o padrão americano. Em funcionamento nos

Estados Unidos desde novembro de 1998, o padrão ATSC também já foi adotado pelo Canadá, Coréia do Sul, Taiwan e Argentina. Nestes dois últimos países, existe uma sinalização que deverá ocorrer uma revisão do padrão de sistema de televisão digital a ser adotado.

Da mesma forma que o padrão DVB, o padrão ATSC é formado por um conjunto de documentos que definem os diversos padrões adotados, incluindo aqueles relacionados à transmissão, transporte, codificação e *middleware*.

O padrão ATSC permite diversas configurações para a camada de transmissão, definindo diferentes esquemas de modulação para transmissão terrestre, via cabo e via satélite. Na radiodifusão terrestre, o padrão ATSC pode operar com canais de 6, 7 ou 8 MHz e utiliza a modulação 8-VSB, cuja taxa de transmissão é de 19,8 Mbps. Em função deste esquema de modulação, um sistema ATSC apresenta problemas na recepção por antenas internas e não permite a recepção móvel.

O padrão ATSC prevê diversos modos de transmissão com diferentes níveis de resolução da imagem e formatos de tela. No entanto, o modelo de negócios americano foi direcionado para a televisão de alta definição. Em função do alto custo dos aparelhos de televisão de alta definição, o sistema americano de televisão digital ainda possui uma baixa adesão dos usuários.

Na camada de codificação, o sinal de áudio é codificado usando o padrão Dolby AC-3 (ATSC, 2004) e o sinal de vídeo é codificado usando o padrão MPEG-2 com qualidade *high definition television* (HDTV). Na camada de transporte, da mesma forma que o padrão DVB, o padrão ATSC multiplexa e demultiplexa os fluxos elementares de áudio, vídeo e dados (aplicações) usando a recomendação MPEG-2.

2.3.3 ISDB

O padrão ISDB (ISDB, 1999) foi especificado em 1999 no Japão pelo grupo *Digital Broadcasting Experts Group* (DiBEG), criado em 1997 e composto por várias empresas e operadoras de televisão (DiBEG, 2004). O objetivo do grupo DiBEG é promover e especificar o sistema de difusão terrestre de televisão digital japonês.

O padrão ISDB é também conhecido como o padrão japonês de televisão digital. Até o momento, o padrão ISDB foi adotado apenas no Japão, porém é amplamente divulgado que o mesmo é um sistema que reúne o maior conjunto de facilidades: alta definição, transmissão de dados e recepção móvel e portátil.

Da mesma forma que os padrões DVB e ATSC, o padrão ISDB é formado por um conjunto de documentos que definem os diversos padrões adotados, incluindo aqueles relacionados à transmissão, transporte, codificação e *middleware*.

O padrão ISDB permite diversas configurações para a camada de transmissão, definindo diferentes esquemas de modulação para transmissão terrestre, via cabo e via satélite. Na radiodifusão terrestre, a especificação *Integrated Services Digital Broadcasting – Terrestrial* (ISDB-T), pode operar com canais de 6, 7 ou 8 MHz e, da mesma forma que o padrão DVB, utiliza a modulação *Coded Orthogonal Frequency Division Multiplexing* (COFDM), mas com algumas variações, alcançando uma taxa de transmissão que varia entre 3,65 e 23,23 Mbps.

O ISDB é projetado para suportar sistemas hierárquicos com múltiplos níveis, podendo ser usado, por exemplo, para prover simultaneamente recepção de baixa taxa de dados sob condições móveis excepcionalmente difíceis, taxa de dados intermediária para recepção estática e alta taxa de dados para boas condições de recepção.

Embora seja baseado no sistema de transmissão europeu, o ISDB-T é superior ao DVB-T quanto à imunidade a interferências, permitindo a convivência da televisão de alta definição com a recepção móvel.

Na camada de codificação, o sinal de áudio é codificado usando o padrão *Advanced Audio Coding* (AAC), o sinal de vídeo é codificado pelo padrão MPEG-2 com qualidade de alta definição. Vale ressaltar que, em função da flexibilidade do sistema, o sinal de vídeo pode ser codificado usando diferentes níveis de resolução.

2.4 MIDDLEWARES

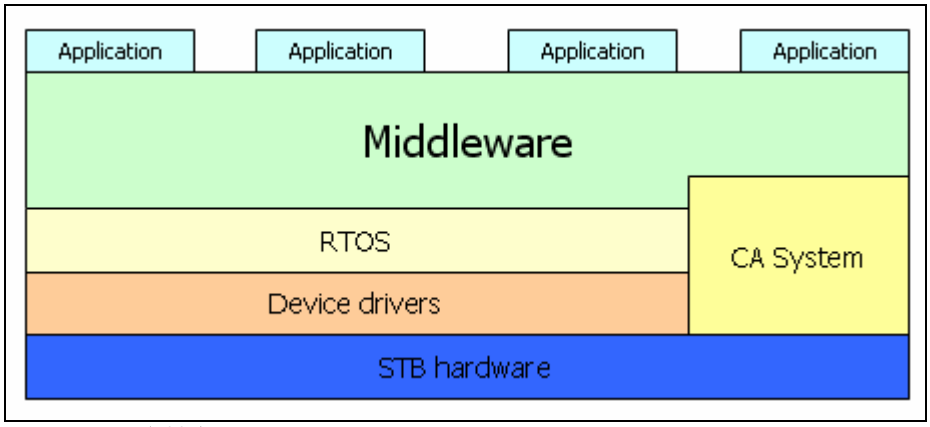
As tecnologias de TV digital permitem a fabricação de STB com diferentes arquiteturas de *hardware*, cujas capacidades de processamento, armazenamento e

comunicação são bastante variáveis. Além disso, estes diversos dispositivos também podem adotar diferentes sistemas operacionais.

Neste cenário de *hardware* e *software* heterogêneos, os desenvolvedores de aplicações devem escrever diferentes versões dos programas para cada combinação de *hardware* e sistema operacional dos diversos tipos de STB. Conseqüentemente, a heterogeneidade das plataformas torna o desenvolvimento de aplicações para televisão digital uma atividade ineficiente e de custo elevado, que pode inviabilizar sua adoção em larga escala.

Para tornar mais eficiente o desenvolvimento de aplicações, bem como reduzir os custos associados, favorecendo assim a consolidação da TV digital, os fabricantes e provedores de conteúdo perceberam que a solução é adotar mecanismos que tornem portáveis as aplicações e os serviços nos diversos tipos de STB.

Neste sentido, para atender ao requisito de portabilidade, os STBs devem prover às aplicações uma API genérica, padronizada e bem definida. Esta API deve abstrair as especificidades de *hardware* e *software* dos diversos tipos de dispositivos receptores. (Figura 4)



Fonte: Morris (2002)

Figura 4 – O middleware dentro de uma arquitetura de TV digital

2.4.1 MHP

MHP (MHP, 2004) é um padrão aberto para TV digital definido pelo DVB que apresenta um conjunto de tecnologias para implementar serviços digitais multimídia interativos. O principal objetivo do MHP é permitir o crescimento de mercados para televisão digital e serviços multimídia onde existe uma acirrada competição entre fornecedores de

conteúdo e fabricantes de receptores. Além disso, explorar a crescente convergência entre transmissão *broadcast*, Internet e consumo eletrônico.

A proposta do padrão aberto MHP é finalizar com o fato do usuário ser obrigado, atualmente, a "alugar" um STB de determinada emissora *broadcast* uma vez que os conteúdos apenas são reconhecidos e apresentados em um receptor cujas aplicações conhecem a especificação do *middleware*, o chamado mercado vertical. Existindo um padrão de *middleware* aberto, todas as transmissoras poderiam desenvolver seus conteúdos baseados neste padrão, sendo o fato semelhante aos fabricantes de receptores que poderiam ter conhecimento do que deveriam implementar para suportar as aplicações, e por conseguinte permitir ao usuário receber sinais de qualquer transmissora *broadcast*.

O padrão MHP define também protocolos de transporte, ciclos de vida das aplicações, modelos de sinalização, modelos de segurança, *plug-ins*, entre outros elementos. Também são especificados os dois tipos de aplicações suportadas pelo MHP, sendo definidas como aplicações DVB-HTML, onde a informação é apresentada através de conteúdo *hipermídia*, e aplicações DVB-J, que são representadas através de conteúdo compilado na linguagem Java.

2.4.2 AS PRINCIPAIS APIs QUE COMPÕEM UM MIDDLEWARE

Embora os diversos padrões de *middleware* não sejam compatíveis entre si, estes padrões adotam versões modificadas, reduzidas ou estendidas de determinadas APIs.

Dentre estas APIs, pode-se destacar: *Digital Audio-Visual Council* (DAVIC), (DAVIC, 1999), *Home Audio Vídeo Interoperability* (HAVi), (HAVi, 2001) e Java TV (JAVA TV, 2001). Em função da ampla adoção destas APIs em grande parte dos *middewares* atuais, apresentamos uma breve descrição de suas principais funcionalidades.

2.4.2.1 DAVIC

DAVIC é uma associação que representa diversos setores da indústria audiovisual, que foi iniciada em 1994, mas extinta após 5 anos de atividade, conforme já previsto no seu estatuto. O principal objetivo da associação DAVIC foi especificar um padrão da indústria para interoperabilidade fim-a-fim de informações audiovisual digital interativa e por difusão.

Para obter esta interoperabilidade, as especificações DAVIC (DAVIC, 1999) definem interfaces em diversos pontos de referência, onde cada interface é definida por um conjunto de fluxos de informações e protocolos. As especificações DAVIC define formatos de conteúdos para diversos tipos de objetos (fonte, texto, hipertexto, áudio e vídeo) e também incluem facilidades para controlar a língua adotada no áudio e na legenda.

Além disso, as especificações DAVIC definem diversas APIs relacionadas a informações de serviços, filtragem de informações, notificação de modificações nos recursos, sintonização de canais de transporte e controle de acesso:

a) API s*ervice information*: provê às aplicações uma interface de alto nível para acessar informações de serviços presentes em fluxos MPEG-2. Esta API define métodos para acessar todas as informações presentes nas tabelas de serviços, permitindo, por exemplo, que um guia de programação eletrônico possa identificar o escalonamento dos programas de cada serviço;
b) API MPEG-2 s*ection filter*: permite as aplicações identificarem a ocorrência de determinados padrões nos dados mantidos em seções privadas MPEG-2;
c) API *resource notification*: define um mecanismo padrão para aplicações registrarem interesse em determinados recursos e serem notificadas de mudanças nestes recursos;
d) API *tuning*: especifica uma interface de alto nível para fisicamente sintonizar os diferentes fluxos de transporte;
e) API *conditional access*: provê uma interface básica para o sistema de controle de acesso. Por exemplo, esta API permite a aplicação verificar se o usuário possui direito de acesso a um dado serviço ou evento;
f) API DSM-CC *user-to-network*: define mecanismos para que as aplicações possam controlar as sessões DSM-CC.

Para apresentação da saída gráfica, as especificações DAVIC adotam um subconjunto do pacote *Abstract Window Toolkit* (AWT) de interface com o usuário da API Java. Para apresentar fluxos de áudio e vídeo, as especificações DAVIC adotam o *Java Media Framework* (JMF), (JMF, 1999) e definem algumas extensões para características específicas de televisão digital. Por exemplo, as especificações definem APIs para sincronizar aplicações em um instante específico de tempo de um determinado conteúdo e gerenciar eventos incluídos no conteúdo ou início da apresentação de uma determinada mídia.

2.4.2.2 HAVi

HAVi é uma iniciativa das oito maiores companhias de produtos eletrônicos que especifica um padrão para interconexão e interoperação de dispositivos de áudio e vídeo digital. A especificação HAVi (2001) permite que os dispositivos de áudio e vídeo da rede possam interagir entre si, como também define mecanismos que permitem que as funcionalidades de um dispositivo possam ser remotamente usadas e controladas a partir de outro dispositivo.

A especificação HAVi é independente de plataforma e linguagem de programação, podendo ser implementada em qualquer linguagem para qualquer processador e sistema operacional. Desta forma, a especificação HAVi permite que os fabricantes projetem dispositivos interoperáveis e os desenvolvedores de aplicações possam escrever aplicações Java para estes dispositivos usando a API provida pelo HAVi.

No contexto de televisão digital interativa, o STB pode ser conectado em uma rede HAVi, podendo compartilhar seus recursos com outros dispositivos e usar os recursos de outros dispositivos para compor aplicações mais sofisticadas. Por exemplo, um STB pode gerar um menu completo que permite ao usuário acessar funcionalidades de qualquer dispositivo ou uma combinação de dispositivos HAVi, usando somente o controle remoto da televisão e apresentando o sistema de forma consistente para o usuário.

Como outro exemplo, um STB pode automaticamente programar o aparelho de vídeo cassete a partir das informações obtidas do guia de programação eletrônico.

A especificação HAVi define uma arquitetura de *software* distribuída cujos elementos de *software* asseguram a interoperabilidade e implementam serviços básicos tais como: gerenciamento da rede, comunicação entre dispositivos e gerenciamento da interface com os usuários. HAVi define um conjunto serviços distribuídos que suportam APIs Java padronizadas, permitindo que aplicações distribuídas possam transparentemente acessar os serviços através da rede. Para assegurar a interoperabilidade, todos os elementos de *software* se comunicam usando um mecanismo de passagem de mensagem que adota formatos de mensagens e protocolos padronizados pelo HAVi.

2.4.2.3 JAVA TV

Java TV é uma API que estende a plataforma Java. Foi desenvolvida pela Sun Microsystems para prover acesso e funcionalidades a um receptor de televisão digital. Suas principais funcionalidades incluem: fluxo de áudio e vídeo, acesso condicional aos dados nos canais de transmissão, controle do sintonizador de canais, sincronização da mídia para permitir que conteúdo interativo seja sincronizado com o vídeo e o áudio do programa, gerenciamento do ciclo de vida das aplicações, para permitir que as aplicações coexistam com conteúdo televisivo como propagandas, por exemplo, entre outras.

O principal objetivo de Java TV é viabilizar o desenvolvimento de aplicações interativas portáveis, que são independentes da tecnologia de rede de difusão.

Java TV consiste de uma máquina virtual *Java Virtual Machine* (JVM) e várias bibliotecas de códigos reusáveis e específicos para televisão digital interativa. A JVM é hospedada e executada no próprio STB. Java TV foi desenvolvido sobre o ambiente *Java 2 Micro Edition* (J2ME), que foi projetado para operar em dispositivos com restrições de recursos. Neste contexto, determinadas características da plataforma Java foram excluídas, pois são consideradas desnecessárias ou inadequadas para tais ambientes. No entanto, o J2ME não define funcionalidades específicas de televisão. Tais funcionalidades são incluídas no Java TV.

Java TV permite níveis avançados de interatividade, gráficos de qualidade e processamento local no próprio STB. Estas facilidades oferecem um amplo espectro de possibilidades para os desenvolvedores de conteúdo, mesmo na ausência de um canal de retorno. Por exemplo, guias eletrônicos de programação podem oferecer uma visão geral da programação disponível, permitindo a mudança para o canal desejado pelo usuário. Através de mecanismos de sincronização, aplicações específicas podem ser associadas a um determinado programa de televisão. Além disso, aplicações isoladas podem executar de forma independente do programa de televisão.

Em Java TV, programas de televisão tradicionais e interativos são caracterizados como um conjunto de serviços individuais. Um serviço é uma coleção de conteúdo para apresentação em um STB. Por exemplo, um serviço pode representar um programa de televisão convencional, com áudio e vídeo sincronizados, ou um programa de televisão interativa, que contém áudio, vídeo, dados e aplicações associadas.

Cada serviço Java TV é caracterizado por um conjunto de informações que descrevem o conteúdo do serviço denominado *Service Information* (SI). As informações sobre os serviços disponíveis são armazenadas em uma base de dados de informações de serviços denominado de s*ervice information database*.

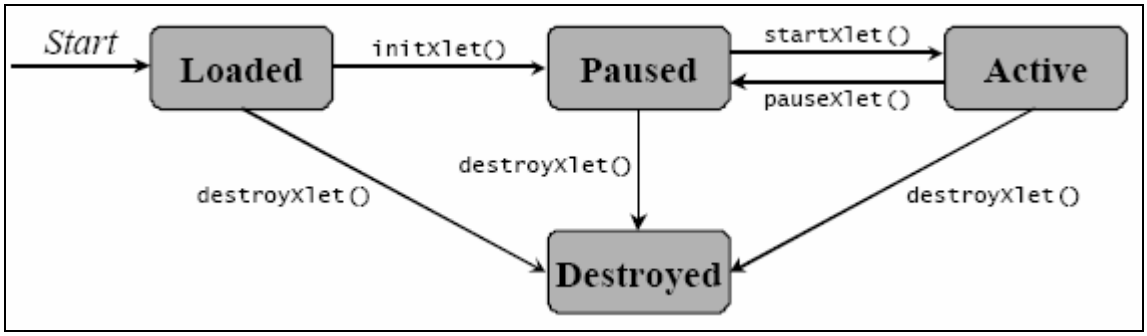
A API Java TV provê uma abstração que permite aplicações obterem informações sobre os diversos serviços disponíveis de forma independente do *hardware* e dos protocolos adotados. Desta forma, uma aplicação pode ser reusada em uma variedade de ambientes.

Java TV define vários pacotes que suportam um conjunto de facilidades para selecionar serviços, obter informações dos serviços, filtrar informações de serviços, controlar a apresentação dos serviços, acessar informações que são entregues através do canal de difusão e gerenciar o ciclo de vida das aplicações. As informações dos serviços podem ser acessadas através de filtros que encontram apenas os serviços de interesse da aplicação. A API Java TV usa o JMF para tratar os fluxos digitais que são recebidos pelo STB, definindo fontes de dados e manipuladores de conteúdo.

Uma aplicação Java TV é denominada x*let*. *Xlet*s não precisam estar previamente armazenados no STB, pois podem ser enviados pelo canal de difusão quando necessários.

Ou seja, o modelo x*let* é baseado na transferência de código executável pelo canal de difusão para o STB e posterior carga e execução do mesmo, de forma automática ou manual. Um x*let* é bastante similar a um *applet* na *web* ou *midlet* em celulares e outros dispositivos móveis.

O ciclo de vida de um x*let* é composto por 4 estados: *loaded*, *paused*, *active* e *destroyed* cujos métodos são ativados para sinalizar mudanças de estado da aplicação. Todo x*let* deve implementar a interface javax.tv.xlet. A Figura 5 ilustra o ciclo de vida de um x*let*, identificando os estados e os métodos suportados por sua interface.



Fonte: Bezerra (2004, p. 12)

Figura 5 – Ciclo de vida de um x*let*

Para gerenciar o ciclo de vida das aplicações (x*let*s), Java TV define o conceito de um gerente de aplicação (*application manager*). O estado de um x*let* pode ser mudado pelo gerente de aplicação ou pelo próprio x*let*. Para tal, métodos da interface x*let* devem ser ativados pelo gerente de aplicação ou pelo próprio x*let*. Neste último caso, o próprio x*let* notifica o gerente de aplicação sobre a transição de estado via um mecanismo de *callback*, que é configurado durante o processo de inicialização do mesmo. Desta forma, o estado de um x*let* é sempre conhecido pelo gerente de aplicação.

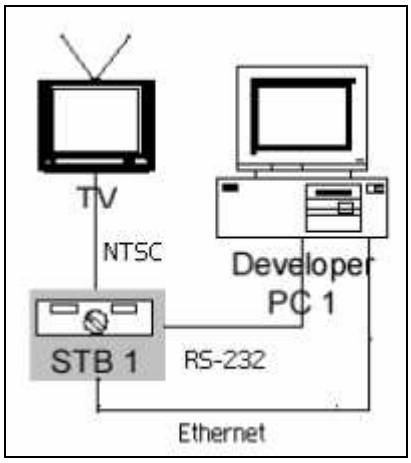
Inicialmente, o x*let* é instanciado pelo gerente de aplicação usando o método *new*. Após a instanciação, o x*let* encontra-se no estado *loaded*. Em seguida, pode ser inicializado pelo gerente de aplicação usando o método *initXlet*. No processo de inicialização, o gerente de aplicação passa para o x*let* um objeto *XletContext* que define o contexto de execução do x*let*. Através deste objeto, o *xlet* pode obter propriedades do ambiente de execução e notificar o gerente de aplicação sobre mudanças de estados via o mecanismo de *callback*.

Após a inicialização, o x*let* encontra-se no estado *paused*. Neste estado, não pode manter ou usar nenhum recurso compartilhado. O x*let* no estado *paused* pode ser ativado usando o método *startXlet*. Após a ativação, encontra-se no estado *active*. Neste estado, o x*let* ativa suas funcionalidades e provê seus serviços. O *xlet* no estado *active* pode voltar ao estado *paused* usando o método *pauseXlet*. Em qualquer estado, um *xlet* pode ser destruído usando o método *destroyXlet*. Após ser destruído, libera todos os recursos e finaliza a execução.

Java TV tem sido amplamente adotado por organizações de padronização, tornando-o um forte candidato a padrão mundial para conteúdo de televisão digital interativa. Atualmente diversas implementações de *middleware* tem adotado o modelo Java TV, com ligeiras diferenças entre si.

2.5 PLATAFORMA DE DESENVOLVIMENTO DVB-J/MHP

A Figura 6 apresenta uma plataforma simplificada para desenvolvimento DVB-J/MHP, composta basicamente por três elementos de *hardware*: TV, computador e um STB. O computador é usado para codificação e simulação de x*let*s. A carga de x*let*s é feita manualmente, com o uso de um *software loader* que carrega o x*let* no STB através de um cabo serial. O canal de retorno do STB é feito através da saída *ethernet*.



Fonte: Pawlan (2001)

Figura 6 – Plataforma para desenvolvimento DVD-J/MHP

2.6 AMBIENTES DE SIMULAÇÃO

Para se entender toda a cadeia de distribuição de dados e sua relação com a disponibilização de serviços interativos em sistemas de TV digital é necessário ir além de apenas compreender os mecanismos que permitem o transporte de dados em meio aos sinais da transmissão. É necessário um ambiente de simulação onde todos os conceitos podem ser especificados e testados.

Para tal necessidade, criaram-se os ambientes de simulação que visam por em prática os conceitos estudados em um ambiente que possa simular um *hardware* geralmente de alto custo e que demandaria grandes investimentos.

Os ambientes de simulação em sua maioria encontram-se em fases iniciais de desenvolvimento, portanto ainda não são totalmente funcionais e capazes de simular todos os recursos de um STB verdadeiro.

2.6.1 O AMBIENTE DE SIMULAÇÃO *XLET*VIEW

O *Xlet*View é um projeto *open-source*, disponível em Sveden (2004), de uma plataforma para testes de aplicativos MHP que possui um gerenciador de aplicações capaz de iniciar e controlar o ciclo de vida de um x*let*. Os x*let*s em estado de execução aparecem em uma tela de TV simulada, também estando disponível um simulador de controle remoto para simulações de interação. O *Xlet*View permite obter um aprendizado inicial do processo de desenvolvimento de aplicações para TV digital.

2.6.2 O AMBIENTE DE SIMULAÇÃO ESPIAL

O simulador Espial disponível em Espial (2004) contém um subconjunto mínimo do MHP, suficiente para a construção de x*let*s simples. Uma versão de avaliação do simulador é fornecida sem custo.

O simulador *iTV Reference* da Espial mostra-se bastante estável para o teste e homologação de aplicações para TV digital. Possui uma vasta documentação e exemplos práticos de sua utilização. Possui ainda suporte a diversos componentes e APIs visuais que facilitam o desenvolvimento de aplicações para este ambiente.

A plataforma Espial permite a simulação da carga e execução de *Xlet*s bem como dos arquivos no carrossel. Em relação a funcionalidade, este ambiente é muito similar a um STB real, sua estrutura de arquivos é definida através de pastas que representam características reais de um sistema de TV digital.

A pasta *channels* simula as informações associadas aos canais (serviços) que são sintonizados pelo usuário através de números de um a nove. Na pasta de cada serviço existe um arquivo em formato texto, no qual estão definidas propriedades do x*let* a ser executado automaticamente quando o usuário selecionar o serviço. No simulador eles são montados a partir da área raiz do sistema de arquivos. A pasta *carousel*, contida na pasta de cada um dos serviços, emula os arquivos que são distribuídos no carrossel.

As pastas *jmfstub* e *lib* (Figura 10) contem as bibliotecas de apoio à compilação e emulação MHP.

A pasta *persistent* emula o sistema de arquivos persistente do STB, possivelmente armazenado numa memória *flash* ou em um disco rígido do STB real.

2.7 TRABALHOS CORRELATOS

Peng (2004) descreve o desenvolvimento de um protótipo de uma aplicação baseada em um gerenciador de simulação utilizando um subconjunto das APIs de um ambiente MHP. Em seu trabalho, Peng inicia com uma introdução sobre as tecnologias disponíveis atualmente para o desenvolvimento de aplicações para TV digital, definindo os tipos de aplicações suportadas e suas arquiteturas. O autor comenta também sobre as limitações de interação que

o usuário tem sobre o aplicativo, visto que a única forma de comunicação entre *software* e usuário é através do controle remoto. Aborda em alguns aspectos a utilização do modelo de comunicação utilizando um canal de retorno onde o usuário interage enviando dados de retorno ao *broadcaster*, possibilitando assim comunicação entre STB e *broadcaster*.

Bezerra (2004) exibe em detalhes dois dos principais padrões abertos para TV interativa em desenvolvimento no mundo, mais especificamente estuda de forma comparativa os *middlewares* que compoem os padrões ATSC e DVB. Demonstra alguns aspectos relacionados às definições dos *middlewares*, seu reconhecimento e padronização pelos órgãos internacionais, a adoção mundial de suas implementações e onde esses padrões atuam. Além disso, destaca aspectos técnicos como as especificações, que hoje são o ponto de partida mais importante para implementação das APIs que compõe os *middlewares*, aspectos e evoluções de arquitetura, principais funcionalidades, aplicações, e como é possível construir serviços interativos com as tecnologias disponibilizadas hoje no mercado em ambas as plataformas para a TV Digital.

No trabalho de graduação de Andrade Neto (2004) é feito um estudo a respeito das técnicas de *data broadcasting* nas plataformas de TV Digital dos principais padrões de sistemas de televisão digital existentes. O processo de *data broadcasting* corresponde à transmissão de dados integrados com o áudio e vídeo, permitindo o envio de texto e outros tipos de mídia associados à programação, além de aplicações a serem executadas nos receptores. Este trabalho de graduação é fruto da criação de um grupo de pesquisa do Centro de Informática, que faz parte do processo de inserção da Universidade Federal de Pernambuco (UFPE) no contexto de contribuição para a definição do padrão nacional de TV Digital.

3 DESENVOLVIMENTO DO TRABALHO

De acordo com o que foi discutido na introdução, e apresentado como objetivo deste trabalho, foi desenvolvido um protótipo de uma aplicação *t-commerce* que simula uma compra virtual de objetos fictícios através de um simulador de TV digital visando demonstrar o funcionamento e arquitetura de uma aplicação para esta plataforma. O presente capítulo trata da especificação e implementação deste protótipo.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Abaixo são enumerados os principais requisitos a serem obedecidos no desenvolvimento do protótipo, classificados como funcionais e não funcionais.

3.1.1 REQUISITOS FUNCIONAIS

Os requisitos funcionais descrevem o que o sistema faz, isto é, as funções necessárias para atender os objetivos do sistema.

O protótipo do *software* a ser desenvolvido deve:

a) permitir ao usuário acesso ao sistema através do controle remoto de forma que o mesmo possa interagir com o *software*, efetuando compras em um sistema de comércio eletrônico;
b) permitir que a cada produto adicionado ao carrinho de compras o mesmo seja contabilizado em um totalizador, onde, ao final da compra, o usuário possa saber quanto comprou;
c) todo produto deverá ter informações breves e legíveis, ilustradas com fotos;
d) o sistema deve permitir que todas as informações relacionadas aos produtos sejam cadastradas em um arquivo de forma que o simulador as recupere através de um *data corousel*, exatamente como acontece em uma aplicação real;
e) o sistema deve ser capaz de rodar um vídeo por baixo da aplicação, simulando uma transmissão de TV;
f) a aplicação deve manter sempre visível uma área reservada ao vídeo que simula a transmissão da TV, para que o usuário, enquanto estiver fazendo uso do aplicativo continue assistindo a TV;

g) no canto inferior direito da aplicação deverá existir um pequeno botão, que simulará o botão de ativação manual da interação. Este aparecerá quando houver um x*let* disponível no *data carousel*.

3.1.2 REQUISITOS NÃO FUNCIONAIS

Os requisitos não funcionais dizem respeito à qualidade do sistema e correspondem as restrições que se colocam sobre como o sistema deve realizar seus requisitos funcionais.

O protótipo a ser desenvolvido deverá obedecer os seguintes requisitos não funcionais:

a) o sistema deverá permitir facilidade de uso, através de uma interface simples e auto explicativa, demonstrando na tela do simulador as opções de interação que o usuário tem sobre o sistema;
b) mesmo sendo executado em um simulador, o sistema deve prover boa performance através de tempos de respostas curtos;
c) o ambiente de desenvolvimento utilizado para o construção do protótipo será a ferramenta JCreator LE, fazendo uso da linguagem Java e das APIs que se fizerem necessárias.

3.2 ESPECIFICAÇÃO E IMPLEMENTAÇÃO

Nesta seção é apresentada a especificação das classes que representam o sistema através de diagramas ilustrados com exemplos de implementação destas classes.

Na Figura 7 tem-se o diagrama de classes que representa a completa modelagem do sistema. Esta modelagem é baseada no padrão UML de representação de classes para um sistema orientado a objetos.

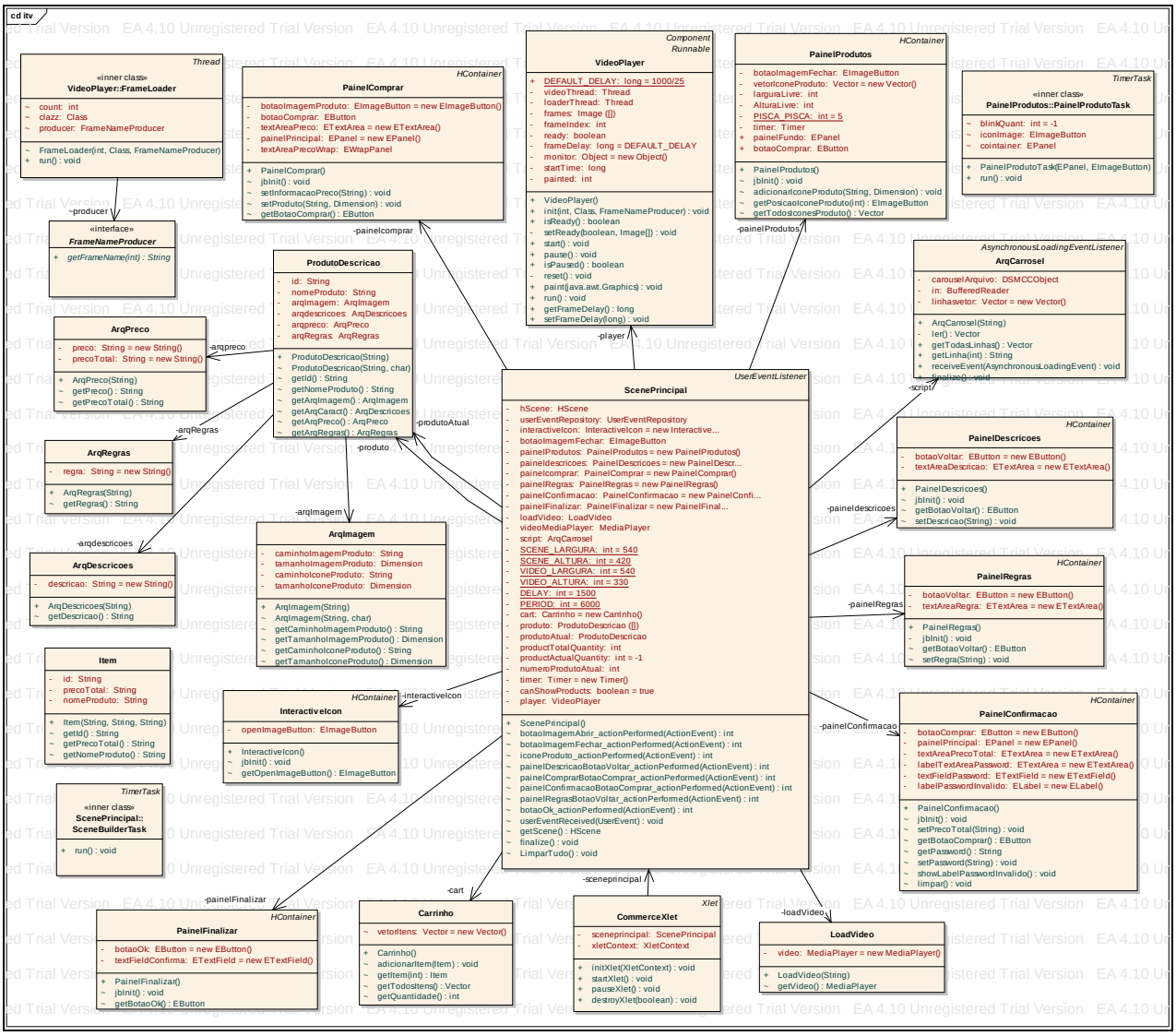


Figura 7 – Diagrama de classes do sistema

O simulador Espial será o responsável por fazer com que a aplicação desenvolvida possa ser simulada em um ambiente virtual.

Para o correto funcionamento do simulador é necessário que o computador que servirá como base de testes possua a máquina virtual Java devidamente instalada.

A tela que aparece na Figura 8 indicada pelo ítem 2 emula o funcionamento do controle remoto. Observa-se que não existe teclado, e que em geral o usuário espera usar apenas as setas de navegação na região inferior do controle remoto para interagir com a aplicação. No item 3 da Figura 8 tem-se a parte do console do simulador. Através dele que pode-se acompanhar o funcionamento da aplicação, que também é utilizado com uma espécie de *debuger* para depurações de erros.

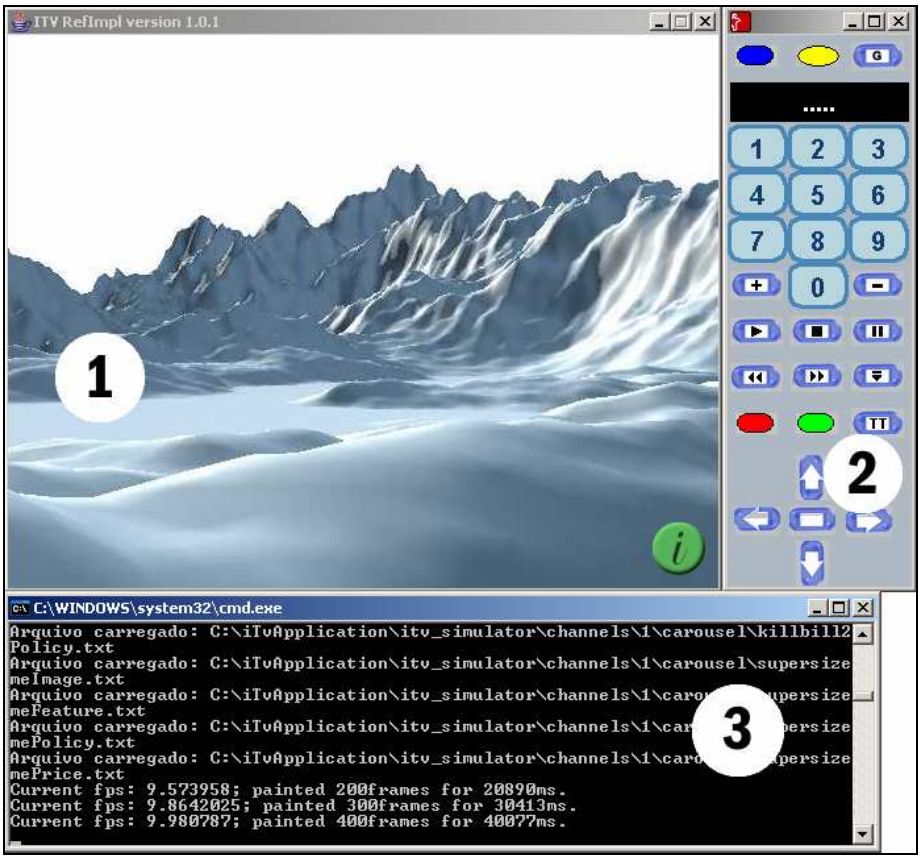


Figura 8 – O emulador iTv Simulator da Espial

O ítem 1 da Figura 8 emula o funcionamento da tela de TV que será denominada neste trabalho de monitor. Por limitações do simulador *iTv Simulator*, o mesmo não é capaz de reproduzir vídeos nesta área, apenas imagens estáticas. Visando suprir esta limitação, criou-se um *plugin* chamado de *frameloader* baseado na implementação de Dogadaylo (2001) afim de tornar mais real a simulação. Desta forma sempre que o simulador for executado o *frameloader* carregará na área monitor uma seqüência de imagens que simula uma transmissão real de TV, conforme ilustrado no Quadro 1.

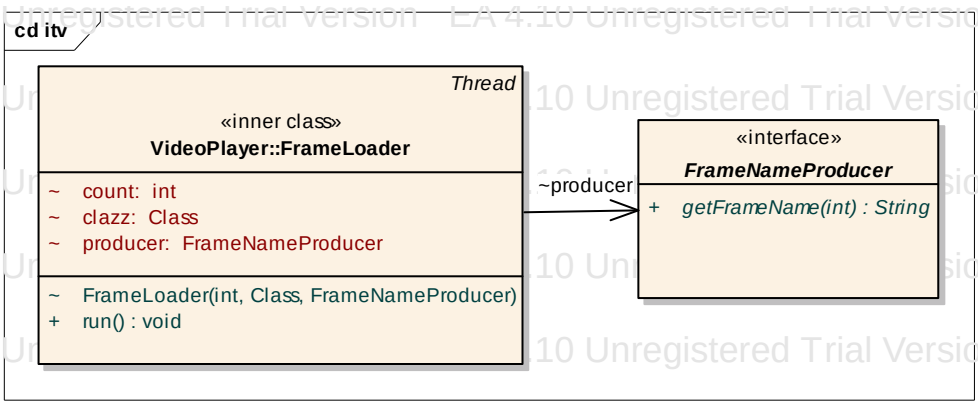


Figura 9 – Representação da classe frameloader

```
    //thread frameloader que tem como objetivo montar um filme
    //através de uma sequencia de imagens
    private class FrameLoader extends Thread {
        int count;
        Class clazz;
        FrameNameProducer producer;

        FrameLoader(int count, Class clazz, FrameNameProducer producer)
        {
            this.count = count;
            this.clazz = clazz;
            this.producer = producer;
        }

        // construtor da thread
        public void run() {
            long time = System.currentTimeMillis();
            System.out.println("Iniciando frame loader");
            Thread t = Thread.currentThread();
            setReady(false, null);

            //vetor das imagens
            Image[] temp = new Image[count];

            // loop que realiza a leitura dos arquivos para montagem do filme
            for (int i = 0; i <  count && t == loaderThread; i++) {
                String fName = producer.getFrameName(i);
                System.out.println("Carregando o frame " + i + ": " + fName);
                try {
                    temp[i] = ImageUtils.getImageResource(clazz, fName);
                    // se não localizou o arquivo da imagem
                    if (temp[i] == null)
                        System.err.println("Não foi possivel localizar a imagem: " + fName);
                } catch (Exception e) {
                    //mostrando o erro
                    System.err.println(e);
                }
            }

            System.out.println("Checando o estado as imagens...");
            // checagem para verificação do filme
            for (int i = 0; i <  count && t == loaderThread; i++) {
                if (temp[i] != null)
                    ImageUtils.ensureImageLoad(temp[i]);
            }

            //se tudo ok.
            if (t == loaderThread) {
                setReady(true, temp);
            }
            time = System.currentTimeMillis() - time;
            System.out.println("Carga completa em " + time + "ms.");
        }
    }
```

Fonte: Adaptado de Dogadaylo (2001)

Quadro 1 – Classe frameloader

A *frameloader* é uma *thread* da classe VideoPlayer (Figura 9) que é executada durante o carregamento inicial da aplicação pelo STB. Sua lógica implementa através do método *run* o carregamento de imagens através de um vetor. A partir do momento que estas imagens são carregadas, um novo processo é executado que tem como função fazer com que estas imagens sejam montadas e agrupadas através de um filme que fica rodando em *loop* por baixo da aplicação.

A plataforma Espial permite a simulação da carga e execução de *xlet*s e bem como dos arquivos no carrossel. A estrutura de arquivos do simulador é apresentada na Figura 10.

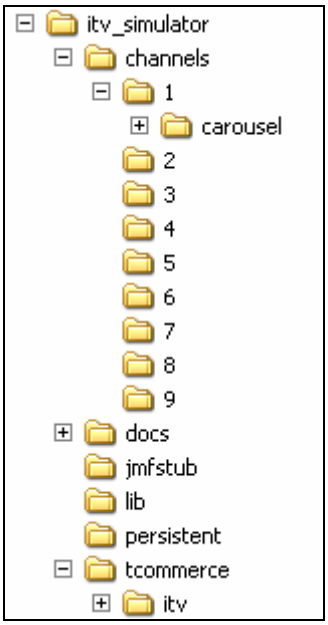


Figura 10 – Estrutura de arquivos do simulador

A pasta *channels* simula as informações associadas aos canais (serviços) que são sintonizados pelo usuário, em número de um a nove. Na pasta de cada serviço existe um arquivo chamado *channel.properties*, em formato texto, no qual define-se os valores de dois atributos: *TVImage* e *Xlet* conforme ilustra o Quadro 2.

```
TVImage=background.png
Xlet=tcommerce.itv.CommerceXlet
```

Quadro 2 – Exemplo de um arquivo channel.properties

O atributo TVImage define o nome do arquivo que contém uma imagem qualquer a ser apresentada no fundo da aplicação quando o serviço for selecionado. Neste aplicativo esta imagem não terá nenhuma função, visto que ao invés de uma imagem estática estar-se-á utilizando um simulador de vídeo conforme definido anteriormente. O atributo *xlet* é o nome do *xlet* a ser executado automaticamente quando o usuário selecionar o serviço. Em um STB real o *xlet* só é executado automaticamente se o *xlet* e STB estiverem configurados no modo *autostart*.

A pasta *carousel*, contida na pasta de cada um dos serviços, emula os arquivos que são distribuídos no carrossel. Nesta aplicação cria-se uma estrutura de arquivos para representar os dados recebidos através do carrossel de dados, distribuídos conforme descrito a seguir.

Arquivo script.txt conterá informações referentes aos produtos disponíveis no carrossel de dados. Este arquivo obedece uma formalização de modo a padronizar o cadastramento dos produtos. Cada campo é separado por vírgula para denotar ao aplicativo *xlet* o início e fim do campo. Este arquivo obedece respectivamente à definição: id do produto, nome do produto, nome do arquivo que contém informações sobre a imagem do produto, nome do arquivo que contém as características e descrição do mesmo, nome do arquivo que contém informações sobre preço e finalmente, o nome do arquivo com informações referentes às regras de venda. O Quadro 3 demonstra um exemplo do arquivo scripts.txt.

```
ID#00002, Eu Robo, euroboImagem.txt, euroboCaract.txt, euroboPreco.txt, euroboRegras.txt
```

Quadro 3 – Estrutura do arquivo scripts.txt

Os arquivos euroboImagem.txt, euroboCaract.txt, euroboPreco.txt e euroboRegras.txt referenciado em script.txt seguem a mesma definição e formatação do exemplo anterior.

A classe ArqCarrosel é responsável pelo carregamento das informações contidas no arquivo de carrossel para a aplicação. Esta classe (Figura 11) possui um método construtor que é executado na instanciação da classe conforme especificado no Quadro 4, que tem como função ler o arquivo do carrossel definido no parâmetro *path* deste método alimentando assim um vetor de dados que conterá todos os dados relacionados aos produtos.

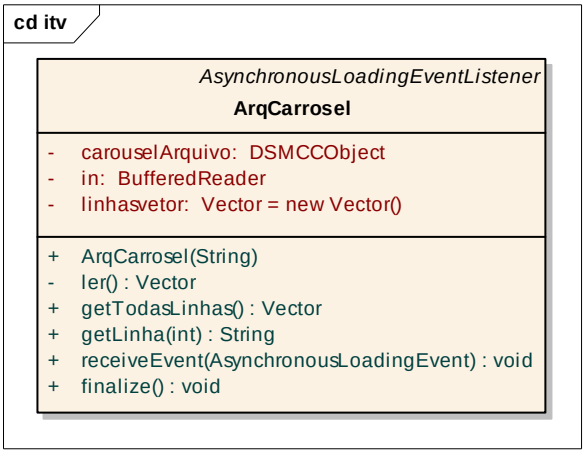


Figura 11 – Modelagem da classe ArqCarrosel

```
  //construtor da classe
  public ArqCarrosel(String path) {
    try {
      System.out.println("Abrindo arquivo...");
      carouselArquivo = new DSMCCObject(path);
      carouselArquivo.asynchronousLoad(this);

      in=new BufferedReader(new FileReader(carouselArquivo));

      linhasvetor = read();

    } catch (InvalidPathNameException e) {
```

```
      System.out.println("Caminho inválido!");
    } catch (IOException er) {
      System.out.println("IOException");
    }
  }
```

Quadro 4 – Método ArqCarrosel, construtor da classe

No canto inferior direito do monitor existe um pequeno botão com um símbolo ‘i’, que simula o botão de ativação manual da interação, que aparece quando há um *xlet* disponível no carrossel de dados (Figura 12).

Figura 12 – Botão que simula a ativação manual da interação

O *xlet* disponível no carrossel de dados é informado no arquivo *channel.properties* já comentado anteriormente. A partir da inicialização do simulador o sistema executa o método *InitXlet* (Quadro 5), conforme definido em Java TV (2001), da classe *CommerceXlet* (Figura 13). Este método quando é executado realiza a montagem das *scenes* para criação da aplicação definido através da instanciação da classe *ScenePrincipal*.

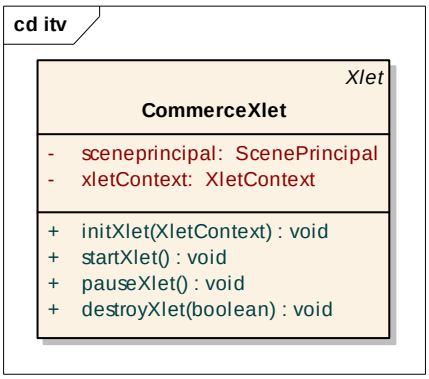


Figura 13 – Modelagem da classe CommerceXlet

```
//Chamado quando o Xlet é inicializado.
  Public void initXlet (XletContext context) {
    System.out.println("Inicializando o Xlet (" + context + ")");

    XletContext = context;

    // instancia de SceneBuilder que monta o Xlet principal
    sceneprincipal = new ScenePrincipal();

  }
```

Quadro 5 – Método initXlet

A classe *ScenePrincipal* possui em sua especificação todo o código necessário para criação da aplicação, montagem de todas as *scenes* secundárias e eventos que são disparados ao serem clicados os botões do programa. Ela é equivalente ao programa principal da aplicação.

Na *scene* principal (Figura 14) são criados (instanciados) todos os *scenes* secundários que representam cada tela do sistema. A cada interação do usuário, como o clique de um botão por exemplo, o *ScenePrincipal* encarrega-se de instanciar a classe do *scene* secundário, redimensionando o monitor, adicionando os elementos que compõe a tela fazendo assim com que todos os componentes correspondentes sejam apresentados ao usuário.

Cada *scene* secundário, denominado aqui de painel, é uma classe estendida de *Hcontainer* (HAVI, 2001) que possui métodos e instância de outros componentes. Um painel basicamente é composto de objetos como: imagens, botões, caixas de texto que juntos formam um módulo do aplicativo.

O sistema proposto trabalha com um *ScenePrincipal* e seis secundários. São eles: comprar, produtos, descrições, regras, confirmação e finalização.

Os *scenes* secundários são responsáveis por apresentar informações específicas para cada módulo do sistema. Dividindo-se desta forma melhora-se a legibilidade do programa facilitando manutenções e futuras implementações.

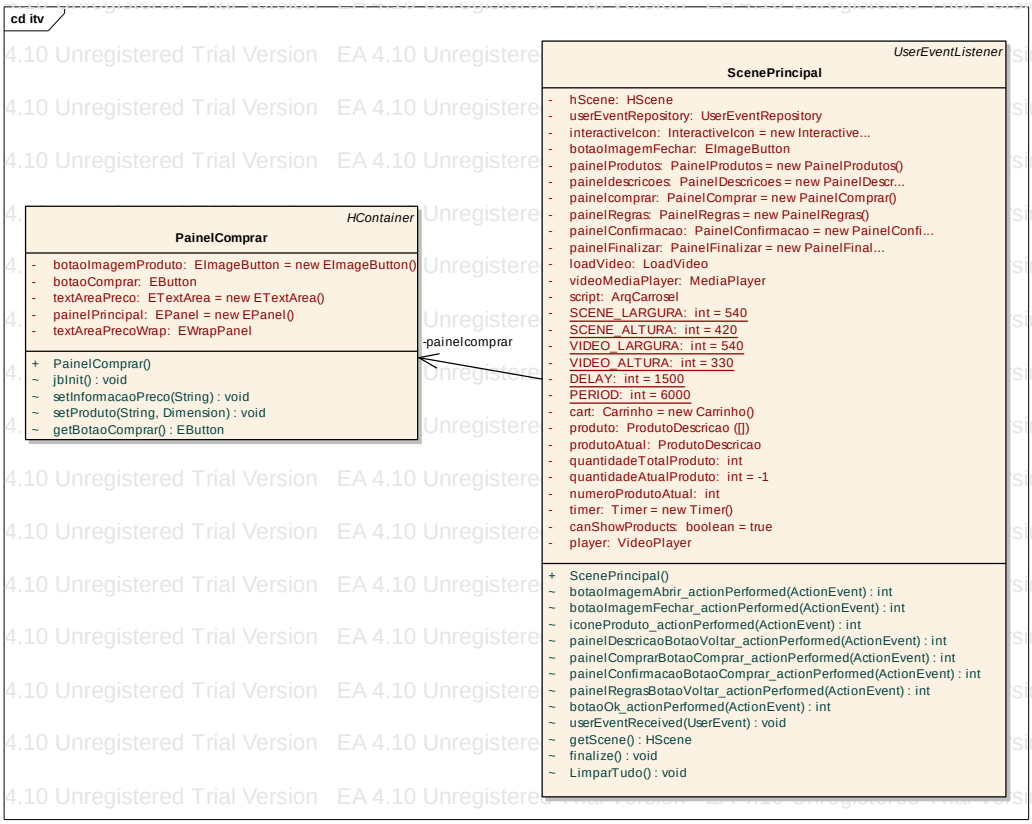


Figura 14 – Modelagem das classes ScenePrincipal e PainelComprar

O Quadro 6 ilustra o evento *iconeProduto_actionPerformed* que é chamado toda vez que o usuário, através de uma interação com o sistema, seleciona em determinado produto para obter mais informações sobre o mesmo.

```
  void iconeProduto_actionPerformed(ActionEvent e) {
    LimparTudo();

    if (e != null) {
      Component  c = (Component) e.getSource();
      numeroProdutoAtual = Integer.parseInt(c.getName());
    }

    //redimensionando a area do monitor
    player.setSize(VIDEO_LARGURA - 130, VIDEO_ALTURA - 5);

    produtoAtual = produto[numeroProdutoAtual];

    //buscando valor do preço do produto
    String preco = produtoAtual.getArqPreco().getPreco();

    //buscando descrição do produto
    String caract = produtoAtual.getArqCaract().getDescricao();

    paineldescricoes.setDescricao(caract);
    painelcomprar.setInformacaoPreco(preco);
    painelcomprar.setProduto(produtoAtual.getArqImagem().getCaminhoImagemProduto(),
    produtoAtual.getArqImagem().getTamanhoImagemProduto());


    //adicionando os camponentes ao scene
    hScene.add(botaoImagemFechar);
    hScene.add(paineldescricoes);
    hScene.add(painelcomprar);
    hScene.add(player);

    hScene.validate();
    hScene.repaint();
  }
```

Quadro 6 – Evento iconeProduto_actionPerformed

No exemplo acima percebe-se que o sistema realiza primeiramente busca de informações sobre os produtos, como preço e descrições. Logo em seguida estas informações são setadas para a classe correspondente alimentando as propriedades do painel secundário comprar. Por fim, todo os elementos que compõem o módulo comprar são adicionados a *scene* através do método *add* da classe *hScene*.

Todos os demais módulos funcionam praticamente da mesma forma, alterando-se apenas alguns parâmetros que são particulares de cada painel.

Um evento também poderá ser ativado ao clique de algum botão do controle remoto. O Quadro 7 demonstra um exemplo de uma interação do usuário com o sistema através do pressionamento da tecla *guide* do controle remoto.

```
Public void userEventReceived(UserEvent e) {
    System.out.println ("userEventReceived " + e.getCode() + ", " + e.getFamily() + ", " +
e.getKeyChar());
    // Evento botão Guide pressionado
    if (e.getCode() == HRcEvent.VK_GUIDE) {
      System.out.println("O botao guide foi pressionado");
    }
  }//end method
```

Quadro 7 – Demonstração de um evento disparado pelo controle remoto do simulador

O parâmetro “e” do método *userEventReceived* possui uma propriedade *getCode* que determina o código do botão que foi pressionado no controle remoto.

3.2.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

As APIs e *softwares* utilizados neste trabalho foram desenvolvidos em Java, portanto no protótipo desenvolvido também foi utilizado como linguagem de programação a linguagem Java, em sua versão SDK 1.4. Para simulação do protótipo foi utilizado o ambiente iTV Simulator da Espial. O ambiente de desenvolvimento utilizado para a implementação das classes foi o JCreator LE, atendendo as necessidades de implementação do trabalho. Para fazer a modelagem em UML das classes foi utilizada a ferramenta Enterprise Architect, versão 4.1 disponibilizando todos os recursos necessários para a modelagem do sistema.

3.2.2 OPERACIONALIDADE DA IMPLEMENTAÇÃO

Afim de exemplificar a utilização da aplicação utilizaremos produtos fictícios para um *t-commerce* de filmes para DVD. Os filmes, como também todas as informações pertinentes aos mesmos estarão presentes no *data carrossel* enviado pela transmissora do serviço.

O canal um será utilizado para simulação do aplicativo. Dentro deste canal tem-se um arquivo chamado *channel.properties* que contém o nome do *xlet* a ser executado conforme o Quadro 8 que representa o nome da classe principal do programa.

```
      Xlet=tcommerce.itv.CommerceXlet
```

Quadro 8 – Arquivo de configuração channel.properties

Na pasta *carrousel* tem-se os arquivos com informações sobre os produtos a serem comercializados que simulam dados enviados pelo *broadcaster* do serviço para a aplicação. O

arquivo principal, scritpt.txt (Quadro 9) possui informações sobre cada um dos produtos anunciados.

```
ID#00001, Eternal Sunshine, eternal_sunshineImagem.txt, eternal_sunshineCaract.txt,
eternal_sunshinePreco.txt, eternal_sunshineRegras.txt

ID#00002, Eu Robo, euroboImagem.txt, euroboCaract.txt, euroboPreco.txt, euroboRegras.txt

ID#00003, Kill Bill 2,  killbill2Imagem.txt,   killbill2Caract.txt,   killbill2Preco.txt,
killbill2Regras.txt

ID#00004, Super Size Me, supersizemeImagem.txt, supersizemeCaract.txt, supersizemePreco.txt,
supersizemeRegras.txt
```

Quadro 9 – Arquivo script.txt

Cada arquivo referenciado no arquivo script.txt possui características e informações sobre o produto especificado.

O usuário no momento que estiver assistindo seu canal favorito desviará sua atenção para um ícone que aparece no canto inferior da tela de sua televisão (Figura 15) chamando sua atenção para os serviços que estão disponíveis naquele momento.

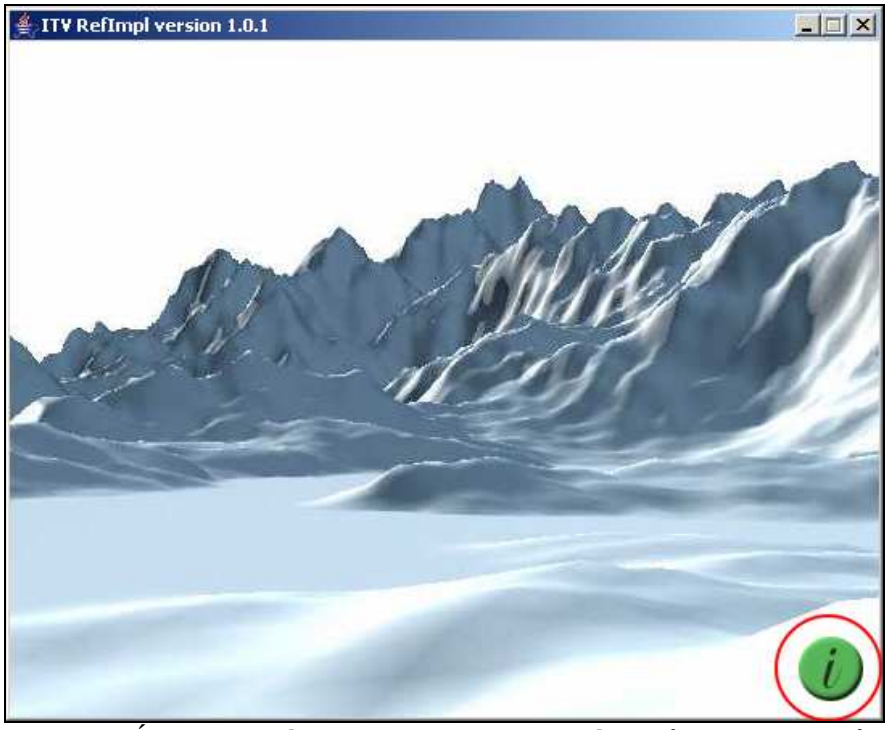


Figura 15 – Ícone que indica que existem serviços disponíveis para o usuário

O usuário ativa o sistema pressionando o botão “i”, o qual dá inicio ao carregamento da aplicação pelo STB. A partir deste momento todas informações disponíveis no carrossel de dados são trazidas à tela e o programa é inicializado. Durante a execução do aplicativo a tela da TV é reduzida, para que o mesmo possa continuar assistindo a programação da TV enquanto efetua a compra, dando espaço ao programa que mostra na tela do usuário ícones com as capas do filmes disponíveis para venda (Figura 16).

Figura 16 – Produtos disponíveis para compra

A partir do momento que o usuário selecionar o filme que deseja adquirir o mesmo passará para uma tela onde visualizará mais informações sobre o produto (Figura 17).

Figura 17 – Informações sobre o produto escolhido

Nesta tela o usuário terá opções para adquirir o filme selecionado clicando no botão comprar, voltar ao menu principal, ou encerrar a aplicação clicando sobre o ícone de interatividade.

Caso o mesmo tenha interesse na compra, o botão comprar será clicado e a partir daí será apresentada uma tela com informações sobre a compra e políticas de venda da empresa, como também um campo de senha, conforme ilustrado na Figura 18. O campo da senha serve como uma forma de garantir quem está realizando a compra e para onde o produto será enviado.

Para este modelo de aplicação assume-se que o usuário já deverá ter um cadastro junto à empresa que presta o serviço de TV digital, a qual fornecerá uma senha a cada usuário.

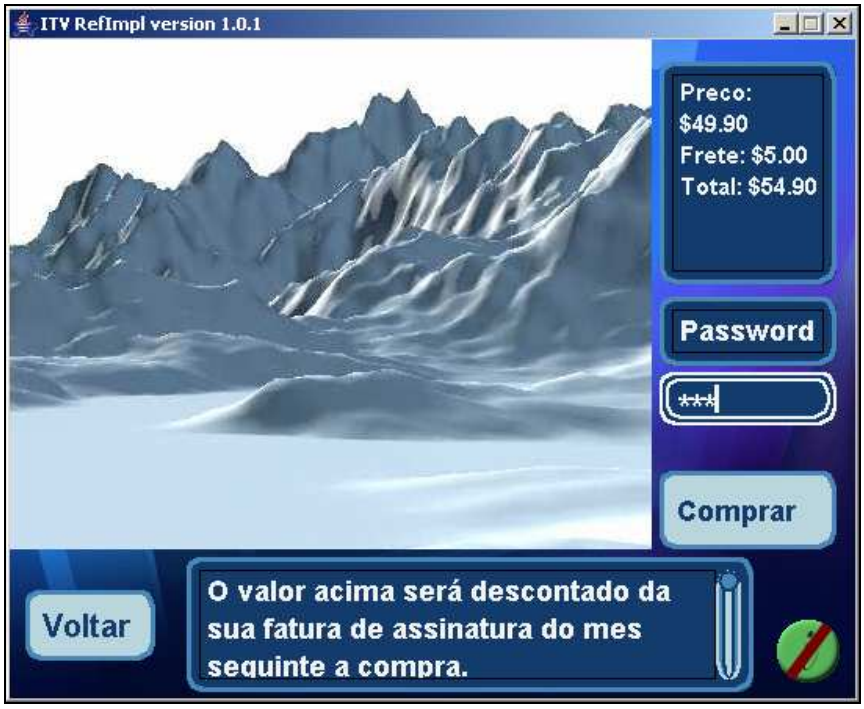


Figura 18 – Confirmando a compra do produto

Realizada a compra, ao final do processo o sistema emite uma mensagem de agradecimento ao usuário indicando que o processo de compra foi concluído (Figura 19).

Figura 19 – Compra realizada com sucesso

3.3 RESULTADOS E DISCUSSÃO

Do ponto de vista tecnológico, o desenvolvimento de uma aplicação fora dos conceitos tradicionais de programação proporcionou uma visão de um novo mercado de desenvolvimento de sistemas ainda em ascensão no Brasil.

Os trabalhos de Andrade Neto (2004) e Bezerra (2004) trouxeram uma boa fundamentação para um conhecimento da tecnologia. Diferente do que foi proposto neste trabalho, Andrade Neto (2004) focou-se principalmente no padrão ATSC sobre o *middleware* DASE, demonstrando através da utilização de um simulador para este padrão um exemplo do funcionamento de um aplicativo já desenvolvido. O objetivo do autor ao confrontar dois dos maiores e mais divulgados padrões de TV digital foi propiciar uma visão geral do funcionamento e das arquiteturas existentes hoje. Peng (2002) procurou focar seu conteúdo em uma espécie de manual de introdução da tecnologia. Infelizmente não foi possível ter acesso ao protótipo desenvolvido para uma análise comparativa mais crítica por indisponibilidade do mesmo pelo autor. Todos os autores relacionados procuram concentrar-se mais no ponto de vista teórico do que prático. As implementações realizadas serviram apenas para demonstrar exemplos e conceitos comentados durante o trabalho, infelizmente

nenhuma delas foi especificada e programada como uma aplicação para o mundo real com algum propósito comercial.

Como resultado, chegou-se ao desenvolvimento de um aplicativo com aproximadamente duas mil linhas que simula um ambiente real de comércio eletrônico para tv digital, fazendo uso de diversos conceitos e técnicas estudadas durante o desenvolvimento do trabalho. Apesar das dificuldades em se desenvolver um *software* em um ambiente não visual, escassez de recursos e ferramentas visuais que facilitariam o desenvolvimento de uma aplicação mais bem apresentável visualmente, o protótipo funcionou de acordo com as expectativas demonstrando-se bastante eficiente para o que foi proposto.

4 CONCLUSÕES

O trabalho cumpriu seu propósito no sentido de fornecer uma preparação básica sobre os conceitos básicos da tecnologia de TV digital e os aspectos de desenvolvimento de aplicações para esse tipo de sistema pelo qual foi proposto.

A API Java TV é o primeiro passo que a Sun está dando em relação à TV digital rumo a um modelo sólido, seguro e principalmente independente de plataforma. Provendo uma série de funcionalidades, o Java TV foi estendido por diversos órgãos internacionais de padronização de TV digital, como MHP por exemplo, adicionando novos recursos e novas APIs. O Java TV serviu como alicerce para criação de muitos padrões existentes hoje, e tem uma relação muito forte com o *middleware* MHP. Infelizmente sua documentação é extremamente escassa, falta bibliografia, suporte, e principalmente exemplos práticos de seu funcionamento.

Os ambientes de simulação são muito importantes para a validação afim de por em prática os conceitos estudados. Através deles consegue-se simular o funcionamento quase que real da aplicação desenvolvida sem a necessidade da aquisição de caros equipamentos de *hardware*. Os atuais simuladores infelizmente ainda não implementam a especificação do MHP em sua totalidade. Algumas funcionalidades por serem extremamente complexas ainda não foram adicionadas, prejudicando em partes uma expansão maior nos itens estudados e um atraso na estabilização de um ambiente que viabilizasse o maior número de recursos possíveis. O simulador XletView, tido como um dos projetos mais difundidos e ativos no momento, e com maior número de recursos implementados da especificação MHP, ainda é muito deficiente em documentação. O simulador da Espial mostrou-se o mais simples e prático ambiente de testes para o *middleware* MHP. Além de possuir uma ampla documentação sobre seu funcionamento, o mesmo possui uma infinidade de APIs já desenvolvidas e documentadas, facilitando assim o desenvolvimento de aplicações simuladas neste ambiente.

Fica evidente o grande potencial de interatividade e diversificação de conteúdo que propõe a TV digital. Cumprindo o objetivo do trabalho, conclui-se que todas estas tecnologias e padrões são de extrema importância no desenvolvimento de aplicações e contribuem objetivamente para o estudo e entendimento de como funciona o *background* do

desenvolvimento, concepção e funcionamento de um aplicativo baseado em um padrão de *middleware* para TV digital.

4.1 EXTENSÕES

São propostos aqui alguns temas que foram identificados durante o desenvolvimento deste trabalho como temas interessantes de aprofundamento:

a) fazer efetivamente com que a aplicação utilize o canal de retorno especificado em MHP (2004) de forma que a operação de compra seja concluída e os seus dados armazenados em um servidor;
b) adicionar ao protótipo novos recursos como a simulação de um vídeo real por baixo da aplicação, melhorias na forma visual de apresentação dos produtos, sincronização da aplicação e vídeo transmitido, permitindo que uma determinada aplicação inicialize-se automaticamente quando um determinado programa entrar no ar, ao quais foram restringidos pela falta de implementação por parte do simulador utilizado, testando-o em novas versões e em outros ambientes que venham a surgir;
c) analisar a parte de segurança das transações entre STB e provedor.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDRADE NETO, Fernando da Cunha**. Entendendo data brodacasting em plataformas de TV digital**. 2004. 33 f**.** Trabalho de Graduação (Graduação em Ciências da Computação) - Centro de Informática, Universidade Federal de Pernambuco, Pernambuco.

ATSC ADVANCED TELEVISION SYSTEMS COMMITTEE. **ATSC**. [S.l], [2004?]. Disponível em: <http://www.atsc.org/>. Acesso em: 01 ago. 2004.

BALABANIAN Victor. **An introduction to digital storage media command and control**. [S.l.], [2004?]. Disponível em: <http://www.chiariglione.org/mpeg/events&tutorials/dsmcc/dsmcc.htm>. Acesso em: 15 out. 2004.

BEZERRA, Edmo Sérgio Ribeiro**. Estudo dos middlewares MHP e DASE para TV digital**. 2004. 52 f**.** Trabalho de Graduação (Graduação em Ciências da Computação) - Centro de Informática, Universidade Federal de Pernambuco, Pernambuco.

BROADCAST PAPERS. **Interactive TV and datacasting papers**. [S.l.], 2004. Disponível em: <http://www.broadcastpapers.com/data/data.htm>. Acesso em: 10 set. 2004.

DAVIC. **DAVIC 1.4.1 specification part 9:** information representation. [S.l.], 1999. Disponível em: <http://www.davic.org>. Acesso em: 10 set. 2004.

DIBEG DIGITAL BROADCASTING EXPERTS GROUP. **Digital broadcasting experts group**. [S.l.], 2004. Disponível em: <http://www.dibeg.org>. Acesso em: 12 out. 2004.

DOGADAYLO, Dmytro. **Frameproducer**. [S.l.], [2001]. Disponível em: < http://www.javadevices.org/dtcourse/index.html>. Acesso em: 19 set. 2004.

DVB DIGITAL VÍDEO BROADCASTING. **DVB project**. [S.l.], [2004?]. Disponível em: <http://www.dvg.org>. Acesso em: 05 ago. 2004.

ESPIAL. **ITV reference plataform**. [S.l.], [2002?]. Disponível em: <http://www.espial.com>. Acesso em: 05 ago. 2004.

HAVI. **HAVi v1.1 - home audio video interoperability**. [S.l], 2001. Disponível em: <http://www.havi.org>. Acesso em: 03 jun. 2004.

ISDB INTEGRATED SERVICES DIGITAL BROADCASTING. **Specification of channel coding, framing structure and modulation**. [S.l.], 1999. Disponível em: <http://www.isdb.org>. Acesso em: 25 set. 2004.

JAVA TV. **API specification**. [S.l], 2001. Disponível em: <http://java.sun.com/products/javaTV/>. Acesso em: 01 ago. 2004.

JMF JAVA MEDIA FRAMEWORK. **JMF 2.0 - java media framework 2.0 API guide**. [S.l.], 1999. Dipononível em: <http://java.sun.com/products/java-media/jmf>. Acesso em: 12 out. 2004.

MHP MULTIMEDIA HOME PLATAFORM. **DVB project**. [S.l.], [2004?]. Disponível em: <http://www.mhp.org>. Acesso em: 05 ago. 2004.

MORRIS, Steven. **The interactive TV web**. [S.l.], [2002?]**.** Disponível em: <http://www.mhp-interactive.org>. Acesso em: 10 jul. 2004.

PAWLAN, Mônica. **Introduction to digital TV applications programming**. [S.l], fev. 2001. Disponível em: <http://java.sun.com/developer/technicalArticles/javaTV/apiintro/index.html>. Acesso em: 04 ago. 2004.

PENG, C. **Digital television applications**. 2002. 117 f. Tese de Doutorado (Telecommunications Software and Multimedia Laboratory) - Departamento de Ciências da Computação e Engenharia, Helsinki University of Technology, Finlandia.

SVEDEN, Martin. **Xletview sourceforge project**. [S.l.], [2004?]. Disponível em: <http://xletview.sourceforge.net/>. Acesso em: 25 jul. 2004.

TAURION, Cezar. **T-commerce**. [S.l.], set. 2001. Disponível em: <http://www.timaster.com.br/revista/artigos/main_artigo.asp?codigo=427>. Acesso em: 03 out. 2004.