UNIVERSIDADE ESTADUAL PAULISTA
"Júlio de Mesquita Filho"

Pós-Graduação em Ciência da Computação

Leandro Jekimim Goulart

Estudo de Caso de uma extensão de Middlewares de TV Digital Interativa para suporte a Aplicações Residentes Não-Nativas

UNESP

2009

Leandro Jekimim Goulart

Estudo de Caso de uma extensão de Middlewares de TV Digital Interativa para suporte a Aplicações Residentes Não-Nativas

Orientador: Prof. Dr. Marcos Antônio Cavenaghi

Dissertação de Mestrado elaborada junto ao Programa de Pós-Graduação em Ciência da Computação – Área de Concentração em Sistemas de Computação, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

UNESP

2009

Goulart, Leandro Jekimim.
Estudo de caso de uma extensão de Middlewares de TV digital interativa para suporte a aplicações residentes não-nativas / Leandro Jekimim Goulart. - São José do Rio Preto : [s.n.], 2009.
127 f. : il. ; 30 cm.

Orientador: Marcos Antônio Cavenaghi
Co-orientador: Eduardo Martins Morgado
Dissertação (mestrado) - Universidade Estadual Paulista, Instituto de Biociências, Letras e Ciências Exatas

1. Televisão digital. 2. TV digital. 3. Middleware (Programa de computador) 4. DVB. 5. OpenMHP (Programa de computador) 6. Interatividade. I. Cavenaghi, Marcos Antônio. II. Morgado, Eduardo Martins. III. Universidade Estadual Paulista, Instituto de Biociências, Letras e Ciências Exatas. IV. Título.

CDU – 004.4

Ficha catalográfica elaborada pela Biblioteca do IBILCE
Campus de São José do Rio Preto - UNESP

AGRADECIMENTOS

Agradeço ao meu orientador Prof. Marcos Cavenaghi que confiou no meu potencial, me guiou e caminhou comigo no desenvolvimento trabalho.

À minha família, em especial meus pais, que foram bastante compreensivos com minha ausência em finais de semana, feriados e outras datas especiais, pois sempre compartilharam comigo o sonho de alcançar mais este título.

Ao LTIA pelo apoio incondicional neste trabalho através de infra-estrutura cedida, cooperação técnica e também recursos humanos de seus qualificados pesquisadores. Agradeço também a estes amigos e colegas de trabalho pela compreensão, paciência e apoio, pois compartilham comigo o mesmo sonho e vivem na busca pela pesquisa e inovação.

Aos amigos do Programa de Pós-Graduação em Ciência da Computação, Evandro, Bruno, Everaldo, Marcelo Fornazin, Giovani, Marcelo Freitas, César, Igarashi e Bárbara, pelos grupos de estudo e pelas preciosas colaborações prestadas.

Ao Prof. Morgado, Prof. Nilceu e Profa. Roberta pelo apoio na realização desse trabalho.

Aos membros da banca, da qualificação e defesa – Prof. Marcos Cavenaghi, Prof. João Ângelo Martini, Prof. Sementille e Prof. Morgado – que trouxeram importantes contribuições à qualidade deste trabalho, tornando-o muito mais organizado e consistente.

Aos amigos que me apoiaram e motivaram nas incontáveis vezes que me ouviram dizer que o trabalho estava atrasado.

E a todos que de alguma maneira colaboraram para o desenvolvimento desse trabalho.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ACAP - Advanced Common Application Platform
AIT - Application Information Table
API – Application Program Interface
ARIB - Association of Radio Industries and Businesses
ATSC - Advanced Television Systems Committee (US)
AV – Audio e Vídeo
BD – Banco de Dados
BML - Broadcast Markup Language (DIBEG)
CA - Conditional Access
DASE - DigitalTV Application Software Environment (ATSC)
DAVIC - Digital Audio Visual Council
DB - Database
DIBEG - DIgital Broadcasting Experts Group (Japan)
DSM-CC - Digital Storage Media Command & Control (MPEG)
DVB - Digital Video Broadcasting (Euro)
EIT - Event Information Table [PID 0x12] (DVB)
EPG - Electronic Program Guide
FDC - Forward Data Channel
FEC - Forward Error Correction
GEM - Globally Executable MHP
HAVi - Home Audio Video Interoperability
HD – High Definition
HDTV – High Definition Television
IPX – Internetwork Packet Exchange
iTV – Interactive Television
JMF – Java Media Framework
MHEG - Multimedia & Hypermedia (Information Coding) Expert Group
MHP - Multimedia Home Platform (DVB)
MPE - Multi-Protocol Encapsulation
MPTS - Multiple Program Transport Stream
NIT - Network Information Table [PID 0x10] (DVB)
NPT - Normal Play Time
OC - Object Carousel (DSM-CC)
OCAP - OpenCable Application Platform (US)
OSI - Open Systems Interconnection (OSI)
PAT - Program Allocation Table [PID 0] (MPEG2)
PC – Personal Computer (Computador Pessoal)
PCR - Program Clock Reference (MPEG2)
PES - Packetised Elementary Stream (MPEG2)
PID - Packet ID [0-8191]
PMT - Program Map Table (MPEG2)
PSI - Program Specific Information
QAM - Quadrature Amplitude Modulation
QPSK - Quadrature Phase-Shifting Key

RF – Radio Freqüência
SCTE - Society of Cable Telecommunications Engineers (US)
SDT - Service Description Table [PID 0x11] (DVB)
SDTV – Standard Definition Television
SI - Service Information i.e. non PSI
SSU - System Software Update (DVB)
STB - Set Top Box
TS - Transport Stream
VCT - Virtual Channel Table
VOD - Video On Demand
VPS - Video-recorder Programming Service
XML – eXtended Markup Language

LISTA DE FIGURAS

LISTAGENS

LISTA DE TABELAS

Resumo

Ao longo dos últimos anos diversos sistemas de TV Digital foram desenvolvidos em todo o mundo, com destaque para o sistema de TV Digital norte-americano (ATSC), europeu (DVB), japonês (ISDB) e brasileiro (SBTVD). Todos estes sistemas diferem entre si em algum aspecto, mas todos apresentem muitas semelhanças conceituais e estruturais, o que permitiu a construção de plataformas de aplicações e serviços interativos comuns ou interoperáveis entre estes sistemas. Em todos estes sistemas, ambientes de aplicação foram desenvolvidos com o objetivo de oferecer serviços interativos através das redes e receptores de TV Digital. Diversos tipos de aplicações são definidos, com variações sobre o acoplamento ou não com o conteúdo audiovisual, a origem da aplicação (nativa ou do broadcast), entre outros aspectos. Este trabalho traz uma nova proposta, a de aplicações residentes não-nativas, um grupo de aplicações que adicionadas e removidas por usuários, executando no topo das APIs disponibilizadas pelo middleware, mas não associadas aos canais de TV Digital. É discutido então uma proposta para desenvolvimento destas extensões, o cenário escolhido para o estudo de caso, composto por implementação de referência de middleware (OpenMHP) e aplicação (Quizlet), o estudo de caso com os dados coletados, discussão e novas propostas. Os resultados são realizados com base na compatibilidade desta implementação de referência com as aplicações MHP existentes.

PALAVRAS-CHAVE: 1. Televisão digital. 2. TV digital. 3. Middleware (Programa de computador) 4. DVB. 5. OpenMHP (Programa de computador) 6. Interatividade

1. INTRODUÇÃO

Nas últimas décadas a televisão se tornou um eletrodoméstico muito popular. Seja para uso informativo, educação ou entretenimento, este importante aparelho está presente nas casas mais do que qualquer outro eletrodoméstico.

Para que este cenário fosse possível, foram necessários muitos anos de pesquisa na padronização e criação de uma infra-estrutura analógica de difusão de sinais de áudio e vídeo que poderiam ser recebidos pelos aparelhos de TV. Estes investimentos permitiram criar um mercado bilionário de equipamentos de difusão, eletrodomésticos e computadores, em geral relacionados à indústria de semicondutores.

Como evolução dos sistemas de televisão existentes, que nasceram monocromáticos e baseados em equipamentos eletromecânicos, bem como para melhorar a eficiência dos sistemas analógicos, diversos comitês lançaram seus programas para o desenvolvimento de um sistema de televisão digital. Este novo sistema poderia trazer maior tolerância à distorção e à interferência de sinais de áudio e vídeo, permitiria ainda maior eficiência no uso da banda, para a transmissão de conteúdo em alta definição ou a transmissão de múltiplos programas no mesmo canal. Além disso, um sistema digital pode ser integrado mais facilmente a outros sistemas de comunicação, possibilitando a oferta de serviços multimídia interativos.

Estes programas iniciaram suas atividades em meados da década de 1980, quando a possibilidade de difusão digital de televisão era considerada uma realidade muito longe de ser alcançada. Porém, através dos comitês e das entidades de padronização foi possível convergir a indústria para a pesquisa e desenvolvimento dos componentes necessários para a TV Digital.

Neste contexto, o presente trabalho procura discutir os principais aspectos dos sistemas de TV digital existentes, bem como de suas plataformas de aplicações e serviços interativos comuns ou interoperáveis. Apesar de estabelecidas e funcionais, estas plataformas têm sido pouco utilizados no mundo todo, não realizando seu potencial previsto. Diversos motivos levaram a tal situação, como por exemplo a falta de audiência de um lado e a falta de programas interativos de outro. Também se cita como motivo o modelo de negócio das operadoras fortemente baseado em *advertising*, a existência de um mercado misto entre vertical e horizontal com padrões abertos.

1.1. Objetivos e Organização da Dissertação

O presente trabalho propõe, portanto, um modelo de *middleware*, aderente às especificações interoperáveis de plataformas interativas, e que permita ampliar o uso de receptores de TV digital através da introdução de aplicações residentes não-nativas que executem no topo das APIs interoperáveis do *middleware*. A proposta possui uma implementação de referência realizada sobre o ambiente OpenMHP, desenhado para executar aplicações MHP em PC para auxiliar desenvolvedores de aplicações interativas. A avaliação da proposta é realizada com base em um estudo de caso e com testes focados na compatibilidade desta implementação de referência com as aplicações MHP existentes.

O presente estudo envolve conceitos de Sistemas de TV Digital e seus componentes, dentre eles codificação de áudio, vídeo e dados, *middleware*, padrões e aplicativos. O Capítulo 2 apresenta e descreve os conceitos e técnicas abordados neste estudo. A seção 0 descreve conceitos de Sistemas de TV Digital e apresenta seus principais componentes, a seção 2.2 descreve padrões de compressão e codificação multimídia, a seção 2.3 introduz o conceito de *middleware* para plataformas interativas

de TV Digital e a seção 2.4 descreve os principais padrões de Sistemas de TV Digital e padrões de *Middlewares* para TV Digital existente atualmente. Concluindo a revisão bibliográfica, a seção 2.5 descreve os conceitos básicos relacionados aos aplicativos para TV Digital, ou Xlets.

Na seção 3 a proposta é discutida tendo como motivação as seções 3.1. Na seção 3.2 a proposta de *middleware* é definida e na seção 3.4 discute-se o método utilizado no desenvolvimento deste trabalho.

Todo o desenvolvimento realizado para implementação da proposta é detalhado na seção 3.4, seguida na seção 3.5 e 3.6 pela coleta de dados, com a respectiva apresentação, detalhamento e discussão. A seção 3.7 encerra o trabalho com a discussão final e conclusões sobre o trabalho.

2. REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta uma revisão bibliográfica com os conceitos sobre TV Digital e Sistemas Multimídia utilizados no Capítulo 3.

2.1. Sistemas de TV Digital

A implementação de um Sistema de TV Digital, em depreciação dos antigos sistemas analógicos traz vantagens como qualidade superior de áudio e vídeo e o aumento da capacidade de novos serviços. Segundo (JONES *et al*, 2006) as principais vantagens destes sistemas em comparação com sistemas analógicos são:

- **Televisão de alta definição (HDTV)** com maior resolução e imagens *widescreen*;
- **Áudio/Som** surround com canais 5.1;

- ***Multicasting***: transmissão de múltiplos programas em definição *standard* (SDTV) e/ou um programa em alta definição (HDTV) no mesmo canal de *broadcast*;
- **Guia de Programação Eletrônica (EPG)**;
- ***Broadcast* de dados**, onde vídeo, áudio e dados podem ser multiplexados juntos para formar um programa (dados podem ser aplicativos).

Collins (COLLINS, 2001) ainda cita como vantagens a **flexibilidade** e **otimização no uso do espectro de radio**. Sâmia *et al* (SÂMIA *et al*, 2004) destaca a possibilidade de uso de **serviços interativos** se houver disponível um canal de retorno.

Contudo, por muitos anos, os Sistemas de TV Analógicos já haviam migrado muitos de seus componentes de produção e distribuição para sub-sistemas digitais. Apesar disso, a introdução da TV Digital requer mudanças para estações de broadcast (emissoras), para equipes de TI e de *advertising*. Jones *et al* (JONES *et al*, 2006) discute diversas mudanças nas cadeias produtivas considerando aspectos como produção de conteúdo, infraestrutura de equipamentos, codificação de áudio e vídeo, multiprogramação, datacasting, entre outros aspectos pertinentes à toda a rede que compõe um Sistema de TV Digital.

Atualmente podemos definir a TV Digital como sendo:

> "um sistema de radiodifusão televisiva que transmite sinais digitais, em lugar dos atuais, analógicos. É um sistema mais eficiente, no que diz respeito à recepção dos sinais, pois, na transmissão analógica cerca de 50% dos pontos de resolução de uma imagem se perdem e, portanto, apenas metade deles

> são recebidos nos lares. Já a transmissão digital permite que a íntegra do sinal transmitido pelas emissoras seja recebido pelos televisores domésticos" ... "Essa nova tecnologia, pelo fato de ser digital, permite a interatividade do sistema com o telespectador, que passa da passividade a uma atitude ativa frente às transmissões." (MELO, 2000, p.7)

Um modelo de sistema de TV digital é composto de vários módulos, em geral desenvolvidos por diferentes grupos, porém operando de forma integrada. Em um dos protótipos de uma aliança na Europa vários módulos em diferentes localidades compunham um sistema completo: codificador de vídeo, decodificador de vídeo, o subsistema de áudio multicanal, o fluxo de transporte e o sistema de transmissão (WU *et al*, 2006).

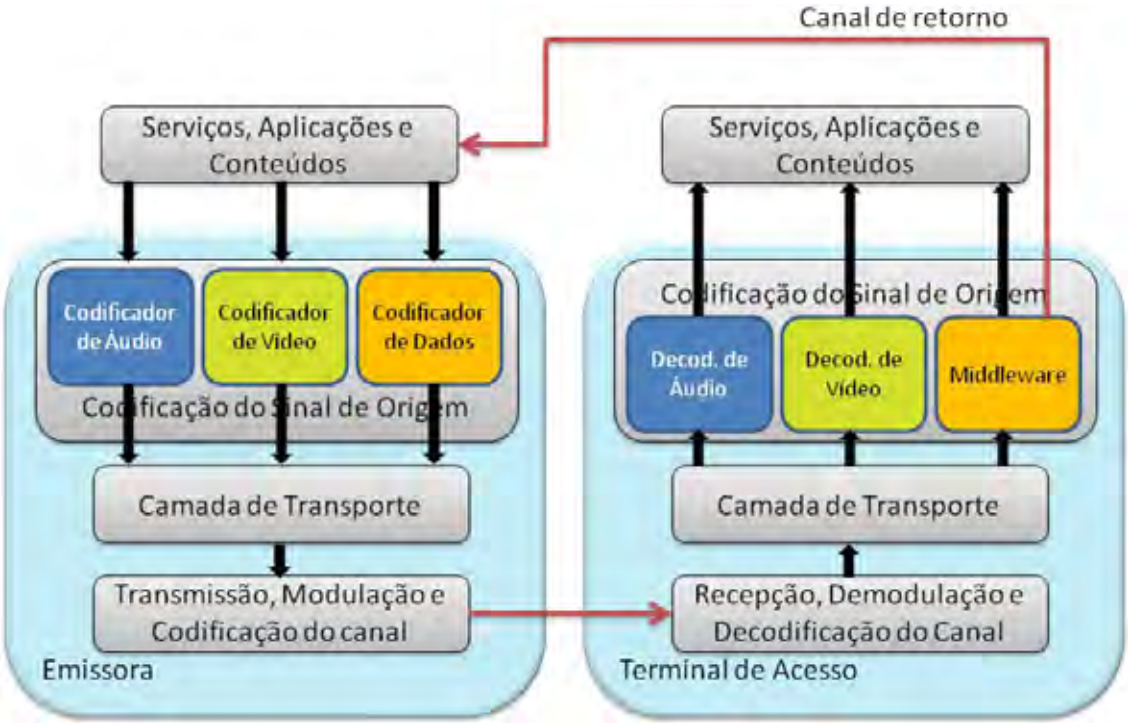

Figura 1 – Sistema de TV Digital segundo (SOL & PINTO, 2007)

O diagrama na Figura 1 representa um Sistema de TV Digital (SOL & PINTO, 2007). Os conteúdos recebidos pelo operador das várias fontes tais como o vídeo local, cabo e canais de satélite, passam por um sistema de emissão digital, onde são preparados para envio aos receptores (BARBOSA & VALENTE, 2002). Este processo inicia-se com a aquisição do Áudio, Vídeo e Dados que são codificados, formando a Camada de Transporte (ou *Transport Stream*). O *Transport Stream* é então modulado e codificado segundo o canal do serviço, e enviado como broadcast ao receptor (Terminal de Acesso). No terminal de acesso, após a demodulação e decodificação do canal, o *Transport Stream* é então decodificado e o Áudio, Vídeo e Dados são tratados com seus filtros e decodificadores específicos, podendo então ser apresentados no dispositivo de saída (ou usado para controle, quando os dados tiverem esta finalidade). A comunicação para serviços interativos é realizada através do canal de retorno.

A função da codificação do canal é, entre outras funcionalidades, modificar o fluxo de dados de entrada da camada de transporte adicionando códigos de correção de erros, codificando para garantia de que o espectro de dados seja uniforme (por todo o canal de 6 MHz, mesmo quando o fluxo de transporte não o oferecer de forma constante), inserindo segmentos repetitivos de sincronização e adicionando outras informações pelas quais o receptor pode detectar e corrigir erros de transmissão (Multipercursos, Ecos, Efeito Doppler – mobilidade, Rajadas de Ruídos, Ruído Térmico ou Branco, Distorções Lineares, entre outros). A importância desta etapa se dá pois uma vez transmitido, não há formas de solicitar repetição de um sinal, de forma que os possíveis erros e ruídos de transmissão, causados por defeitos introduzidos pelo canal de transmissão, precisam ser corrigidos na Set-Top Box (SOL & PINTO, 2007; MOURA, 2006; COLLINS, 2001).

Se na Figura 1 de Sol & Pinto (SOL & PINTO, 2007) é possível observar todo o ciclo de um Sistema de TV Digital, na Figura 2 de Sâmia *et al* (SÂMIA *et al,* 2004), é possível analisar mais detalhes dos componentes do sub-sistema de Broadcasting, em especial o sub-sistema de transmissão RF composto pelo codificador do canal e o modulador, capaz de transmitir para os meios Terrestre, Cabo e Satélite.

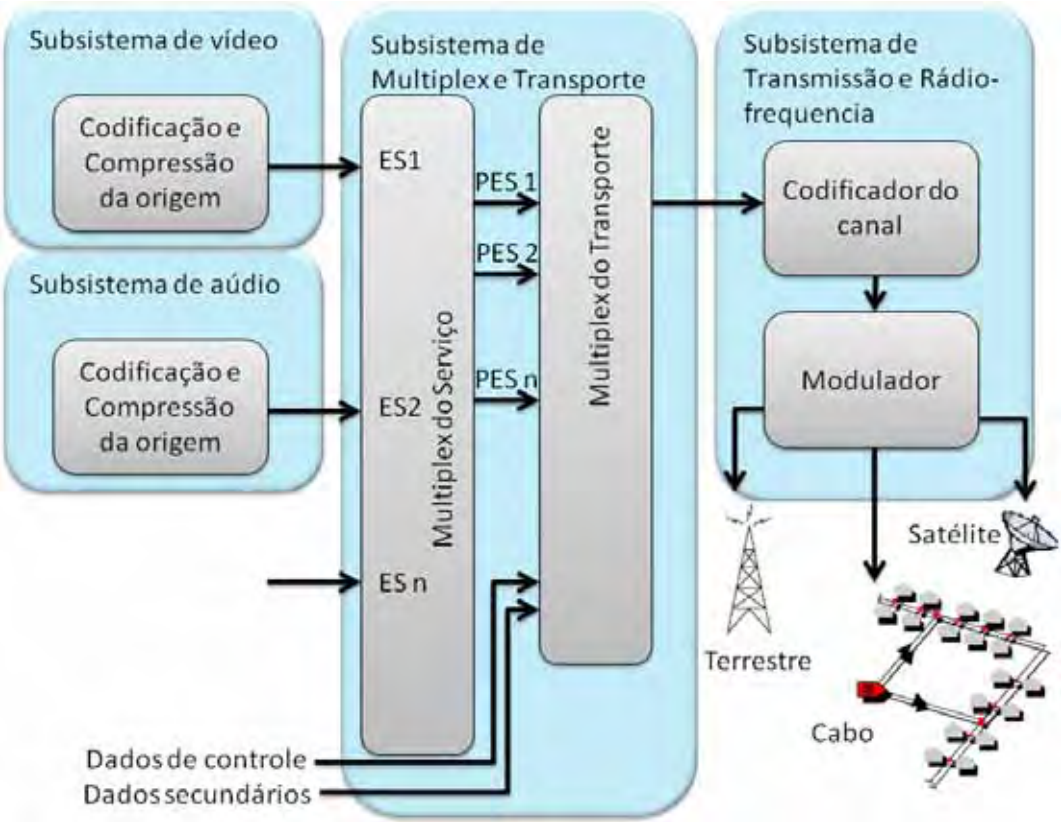

Figura 2 – Um modelo de broadcast de TV Digital segundo Sâmia (SÂMIA *et al*, 2004)

Com fins de ilustrar um sistema de TV Digital abaixo temos o exemplo de um sistema modularizado em racks desenvolvido na América do Norte. Da esquerda para a direita podemos encontramos o rack de compressão de áudio, pré-processamento de vídeo, estimativa de movimento, codificação MPEG, codificação dos pacotes de transporte, moduladores QAM, simulador de canal RF, decodificador QAM,

decodificação dos pacotes de transporte, decodificação MPEG e pós-processamento de vídeo (WU *et al*, 2006).

Figura 3 - Sistema modular de TV Digital desenvolvido na América do Norte (WU *et al.*, 2006)

2.2. Sub-sistema de codificação e compressão do *source*

A codificação do source tem por objetivo auxiliar no uso eficiente da banda de rede disponível, maximizar compressão (permitindo também maximizar a quantidade de canais e a definição do áudio e vídeo) e minimizar a perda de informações e o efeito de erros durante a transmissão do vídeo digitalizado (SOL & PINTO, 2007; BARBOSA & VALENTE, 2002; SIMPSON & GREENFIELD, 2007). Para isto são utilizados diversos padrões de compressão, como o JPEG (DCT), MPEG, Dolby AC-3 ou SMPTE VC-1.

Na Tabela 1 (SOL & PINTO, 2007; SIMPSON & GREENFIELD, 2007; JONES, G. A., 2006) é possível comparar a banda necessária para transmissão de alguns conteúdos não compactados com a largura de banda máxima das principais redes de computadores existentes, corporativas ou domésticas. Nela observa-se que a

transmissão de vídeos não compactados supera a largura de banda máxima das principais tecnologias de redes:

Tabela 1- Largura de banda máxima para algumas tecnologias de redes e banda necessária para transmissão de conteúdos multimídia não-compactados

Tecnologia	Largura de Banda (Mbps)
Áudio estéreo (dois canais)	3
Vídeo 640 x 480 x 24 bits * 24 fps	168,75
Vídeo HD	1.500
Rede cabeada (100BaseT)	100
802.11b/g/n	11/54/100
xDSL	8
WiMax (802.16)	70

Apesar de melhorar a eficiência no uso das redes, a compressão introduz um atraso no sinal de vídeo ou áudio, tanto no estádio de compressão quanto no estágio de descompressão, devido ao processamento necessário para remover as porções semelhantes em frames adjacentes no sinal de entrada (SIMPSON & GREENFIELD, 2007). De forma geral, os benefícios alcançados com a compressão certamente superam as desvantagens, sendo a alternativa mais adequada para sistemas de televisão digital.

Um sistema de compressão é composto por codificadores e multiplexadores. Os codificadores são mecanismos utilizados para digitalizar, comprimir e realizar o *scrambling* em uma gama de áudio, vídeo e canais de dados. Depois de codificado e comprimido, o fluxo é transmitido para o multiplexador, onde serão combinadas as saídas de vários codificadores em conjunto com a informação de segurança, de programas, e outros dados em um único fluxo digital (BARBOSA & VALENTE, 2002), conforme se detalha na próxima seção.

2.2.1. MPEG (Moving Picture Expert Group)

O Moving Pictures Experts Group, um grupo de trabalho da ISO/IEC, tem desenvolvido alguns dos principais sistemas de compressão de vídeo e áudio sob a sigla MPEG (MPEG, 2007; BARBOSA & VALENTE, 2002). Dentre os padrões mais conhecidos estão o MPEG-1, MPEG-2, MPEG-4, MPEG-7 (um meio padronizado de descrever conteúdo audiovisual) e o MPEG-21 (padrões para descrever autoria e direitos autorais em conteúdos). Juntos, estes padrões definem a sintaxe para a codificação da fonte do vídeo e áudio, além do empacotamento e multiplexação do vídeo, áudio, sinais de dados para os sistemas ATSC-DVB, DVB-T, e ISDB-T (COLLINS, 2001). Além da televisão de alta definição, outros meios, como o DVD e a transmissão de vídeo pela internet também fazem uso destes padrões (MPEG, 2007; SIMPSON & GREENFIELD, 2007).

Tabela 2 - Finalidade dos padrões MPEG (MPEG, 2007)

Padrão	Especifica	Exemplo
MPEG-1	Vídeo/Áudio (inclui o MPEG-1 Audio Layer 3 = MP3)	Video CD, câmeras *low-end* e vídeo pela web
MPEG-2	*Stream* de programas	Conteúdo VCR a Full HD HDTV
MPEG-4	AV + objetos, HD	Conteúdo HD com AVC
MPEG-7	Metadados XML	
MPEG-21	Framework multimídia	

O padrão MPEG-2 define a camada de Transport Stream, Multiplex e Sincronização utilizada na maioria dos Sistemas de TV Digital atualmente (CRINON *et al*, 2006). Quando analisado um sistema genérico de TV Digital, como na Figura 2, observa-se que a etapa de codificação é a etapa que precede a camada de transporte, responsável por fornecer o fluxo de dados para o sistema de transmissão RF. A camada de transmissão desempenha um papel importante uma vez que não há proteção contra

erros na camada de transporte. Dessa forma, códigos de correção de erros são fornecidos na camada de transmissão (COLLINS, 2001).

O formato MPEG-2 é o padrão mais utilizado dentre os padrões MPEG, sendo comumente implementado diretamente em semicondutores. Ele é usado em uma grande variedade de aplicativos, dentre eles os ambientes de transmissão HDTV europeu, japonês e norte-americano e codificação de DVDs. Também permite multiplexação de fluxos de áudio e vídeo, permitindo o uso de transmissão multicanal. Suporta 5 canais de áudio (som surround) (SOL & PINTO, 2007) e o padrão Advanced Audio Coding (AAC) (SIMPSON & GREENFIELD, 2007).

Este padrão de codificação define diversos "perfis", permitindo que os formatos dos conteúdos variem progressivamente as suas *bit rates*, atingindo qualidade de VCR a full HDTV (COLLINS, 2001). Testes indicam que o vídeo com qualidade de estúdio pode ser alcançado com um *bit rate* de cerca de 9 Mbps. Vídeo com qualidade de consumo pode ser atingido com *bit rates* variando de 2,5 a 6 Mbps, dependendo do conteúdo do vídeo (COLLINS, 2001).

Lançado em 2000, o padrão MPEG-4 incorpora diversas novas tecnologias para compressão de vídeo. Porém somente após o lançamento do padrão MPEG–4 Advanced Video Coding (AVC) podem ser percebidas melhoras na performance para compressão de seqüências naturais de vídeo ao vivo (SIMPSON & GREENFIELD, 2007).

A codificação da fonte de vídeo no MPEG se dá em várias etapas, entre elas temos ilustrado na Figura 4 a Redundância no Tempo, removida pelas técnicas de *Motion Compensation* e na Figura 5 a Redundância Espacial, removida com a Transformada Discreta do Cosseno (SÂMIA *et al.*, 2004).

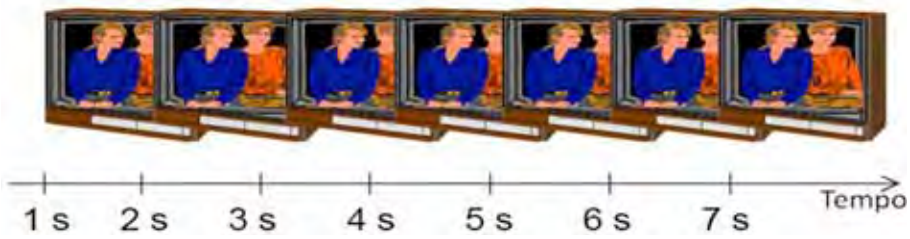

Figura 4 – Redundância no Tempo (SÂMIA *et al.*, 2004)

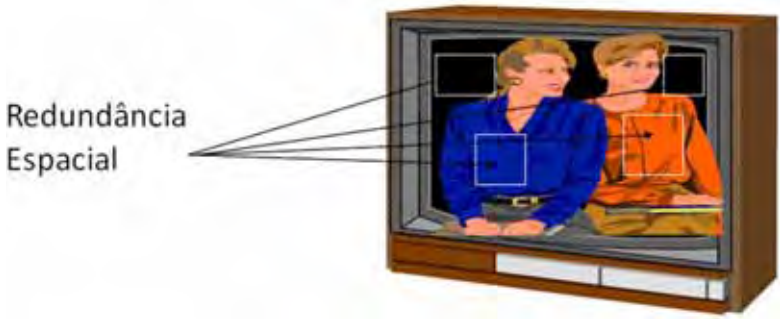

Figura 5 – Redundância Espacial (SÂMIA *et al.*, 2004)

Além de ter grande penetração nos PCs através dos aplicativos reprodutores de multimídia, o padrão MPEG-4 foi adotado na especificação do Sistema Brasileiro de TV Digital (NBR 15601, 2007) e atualmente tem potencial para substituir o sistema MPEG-2 em uma grande variedade de aplicações. Porém, uma das desvantagens do MPEG-4 é que os *decoders* são mais complexos, estimado em 2.5 a 4 vezes mais complexo do que os *decoders* MPEG-2 (SIMPSON & GREENFIELD, 2007). Isto implica em hardwares mais caros e maior demanda por recursos de processamento em reprodutores, como PCs, Set-Top Boxes ou Celulares.

Para a compressão de áudio, o padrão MPEG (em todas as suas versões) oferece três camadas (I, II e III), além de um novo padrão de compressão de áudio (não presente no MPEG-1), chamado Advanced Audio Coding (AAC) (SIMPSON & GREENFIELD, 2007). Estes métodos de compressão baseiam-se no mascaramento de sons low-level em

freqüências próximas, através da codificação destes elementos em taxas mais baixas. Outra técnica utilizada é a eliminação dos componentes de áudio que não são audíveis. Como resultado, tem-se qualidade de áudio aproximada com a de um CD (Compact Disk) em uma taxa reativamente baixa (COLLINS, 2001).

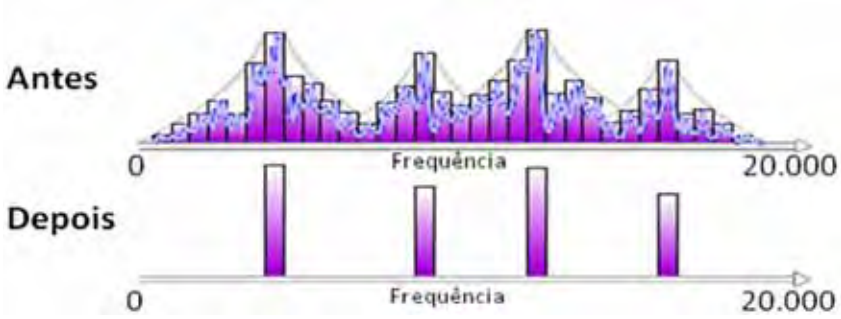

Figura 6 – Remoção de componentes não audíveis e mascaramento de sons low-level em freqüências próximas (SÂMIA *et al.*, 2004)

O sistema de compressão de áudio mais simples é o MPEG audio layer I. A Layer I pode atingir uma taxa de compressão de 4:1, atingindo qualidade comparável com CDs quando codificado a 384 kbps (SIMPSON & GREENFIELD, 2007). O MPEG Audio Layer II consegue atingir maior precisão e ainda eliminar redundâncias na codificação da Layer I, chegando a atingir compressão de até 8:1, o que significa que a qualidade de CD pode ser atingida com 192 kbps (SIMPSON & GREENFIELD, 2007).

O MPEG Audio Layer III aplica outras técnicas que a tornam mais eficiente. A Layer III possui um modo chamado *joint stereo* que se aproveita das semelhanças de áudio entre os canais esquerdo e direito, retirando informações que estão em ambos os canais para utilizar mais bytes em outras informações. Ele também usa uma codificação de tamanho variável para empacotar mais eficientemente os coeficientes do áudio comprimido nos fluxos de saída. Como resultado, o codificador Layer III consegue comprimir em taxas de 12:1, atingindo a qualidade de CD em um fluxo a 128 kbps (SIMPSON & GREENFIELD, 2007).

O MPEG AAC, que está disponível somente no MPEG-2 e MPEG-4 atinge excelentes qualidades de som surround a partir de AAC a 192 kbps (SIMPSON & GREENFIELD, 2007).

O formato e protocolo do fluxo de transporte são baseados em um pacote de tamanho fixo definido e otimizado para entrega em televisão digital. Fluxos de bits básicos dos codificadores de áudio, vídeo e dados são empacotados e multiplexados para formar o fluxo de bit de transporte. Complementarmente, a recuperação do fluxo de bits elementar é feita pelo receptor (COLLINS, 2001).

Nos sistemas de televisão digital, o fluxo de transporte composto por áudio, vídeo e dados empacotados (todos multiplexados) é projetado para acomodar um único programa HDTV ou vários programas SDTV. Em ambos os casos múltiplas fontes de dados são multiplexadas em dois níveis distintos, conforme ilustrado na Figura 7. No primeiro nível, o fluxo de bits do programa é formado pela multiplexação dos fluxos elementares de um ou mais fontes, contendo informações de *timing* para garantir que a decodificação será realizada na seqüência correta (COLLINS, 2001).

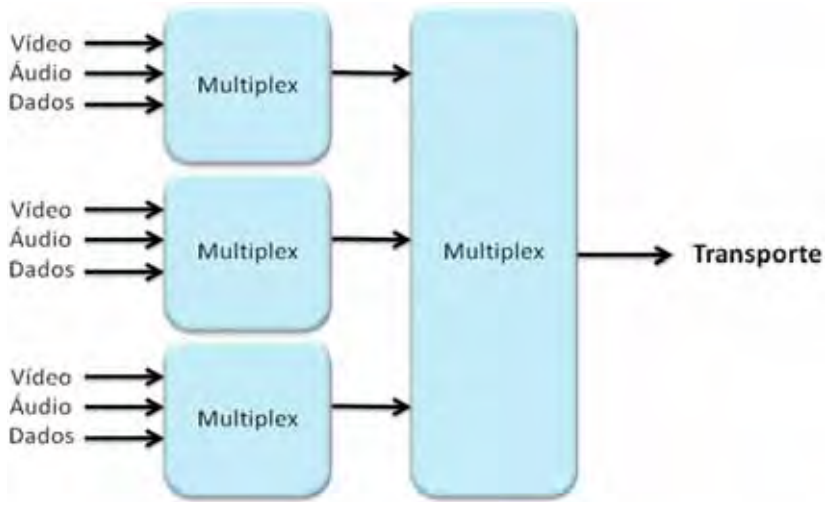

Figura 7 – Multiplexação no padrão MPEG-2 (COLLINS, 2001)

Isto é possível pois o MPEG (a partir do MPEG-2) possui dois métodos de multiplexação (SÂMIA *et al*., 2004):

- *Program Stream*: apenas um programa é multiplexado. Um programa consiste de um ou mais *Elementary Streams* (é um conjunto de bytes - fluxo de dados - de um tipo de dado específico: áudio, vídeo, dados);
- *Transport Stream*: um ou mais programas podem ser multiplexados juntos.

Um programa típico pode incluir vídeo, diversos canais de áudio, e múltiplos fluxos de dados. No segundo nível de multiplexação, diversos programas são combinados para formar um sistema de programas. O conteúdo do fluxo de transporte pode variar dinamicamente dependendo do conteúdo da informação dos programas de origem (COLLINS, 2001).

Em alguns sistemas de TV Digital também se encontra em uso o Dolby AC-3 Audio, também conhecido como Dolby Digital. Este padrão oferece boa qualidade e taxa de compressão, sendo aprovado para uso em DVDs e no sistema de televisão digital dos Estados Unidos. Ele está presente em algumas versões do MPEG-4 e é usado em sistemas de televisão por satélite (SIMPSON & GREENFIELD, 2007).

2.2.1.1. Digital Storage Media Command and Control (DSM-CC) e o Carrossel de Objetos

Em se tratando de um Sistema de TV Digital, uma importante especificação é a do DSM-CC, um toolkit para desenvolver *control channels* associados com fluxos MPEG, através do Data Download Protocol e do Protocolo do Carrossel de Objetos (CRINON *et al*, 2006).

O Protocolo do Carrossel de Objetos possui um papel fundamental no desenvolvimento dos sistemas de TV Interativa, pois ele atua como uma ponte entre o mecanismo de transmissão tradicional do MPEG-2 (baseado em sistemas) e o

middleware interativo. Mais especificamente, o Carrossel de Objetos é um sistema de "arquivos na nuvem" que define uma organização hierárquica entre os módulos de dados transmitidos em um *Transport Stream*, formando um conjunto de objetos representando diretórios e objetos ligados entre si através de um mecanismo de referência. Como resultado, uma aplicação interativa executando em um receptor digital pode usar o Carrossel de Objetos para navegar através de grandes conjuntos de dados (CRINON *et al*, 2006).

De fato, o Protocolo do Carrossel de Objetos permite usar a carga de módulos (pacotes) de dados enviados via *broadcast* para representar uma coleção de nomes de diretórios, nomes de arquivos ou arquivos. A Figura 8 ilustra o Protocolo do Carrossel de Dados com a transmissão periódica de módulos de dados carregados nas seções do MPEG-2. Dependendo do tempo de acesso pelo *datacasting* ou aplicação interativa e a robustez contra erros de transmissão, os módulos de dados podem ser repetidos muitas vezes em um mesmo período do carrossel. Por exemplo, na Figura 8, a taxa de repetição r_i, i = 1, 2, 3, 4, 5 para os módulos m_i, m = 1, 2, 3, 4, 5, é respectivamente 4, 4, 4, 2, 1 (CRINON *et al*, 2006).

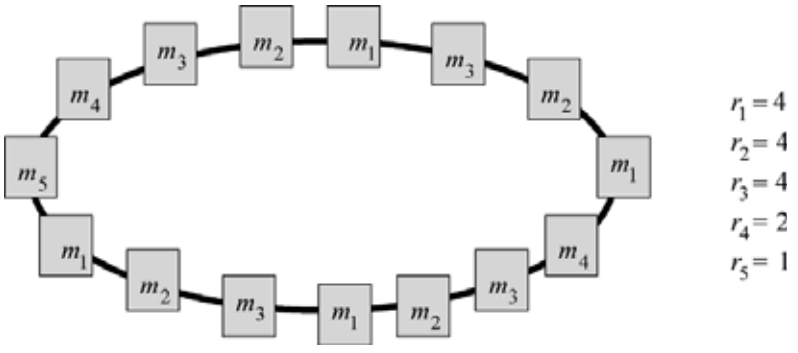

Figura 8 – Carrossel de objetos com múltiplos módulos de dados (CRINON *et al*, 2006)

2.3. Middlewares dos receptores de TV Digital

Além dos subsistemas de transmissão, modulação, codificação, detecção de erros, acesso condicional, interatividade e outros citados anteriormente, os Sistemas de TV Digital definem uma extensão que oferece um conjunto comum e genérico de Application Programming Interfaces (APIs) que permitem que aplicações interoperáveis sejam carregadas das redes de broadcast e executadas nos receptores de qualquer fabricante. Esta API comum fornece uma interface independente de plataforma entre as aplicações de diferentes provedores e as implementações específicas de hardware e software de fabricantes. Isto permite que qualquer fornecedor de conteúdo digital alcance todos os tipos de terminais (PIESING, 2006).

Segundo Sol & Pinto (SOL & PINTO, 2007), independente do sistema para o qual foram desenhadas, tem-se como requisito da arquitetura dos *middleware* dos receptores de TV Digital:

- Robustez e confiabilidade: os espectadores de televisão não são familiarizados com o "reiniciar" das suas Set-top Boxes;
- Capacidade para processar simultaneamente múltiplas tarefas, como processar a chegada de fluxos MPEG e validar mensagens de segurança;
- Incluir normas para interoperabilidade, que permita sua utilização em computadores e Internet onde quer que seja possível.

Figura 9 - Ambiente operacional de um receptor (SOL & PINTO, 2007)

A Figura 9 permite observar as camadas existentes em um ambiente operacional de um receptor de TV Digital (Set-Top Box) e compará-lo com o ambiente encontrado em Computadores Pessoais (PCs). Na base atemos o hardware tradicional, que pode também incluir alguns dos decodificadores necessários em um sistema de TV Digital. Sobre o hardware encontram-se os Drivers e o Real-Time Operating System. A camada de adaptação permite a utilização de uma mesma implementação de *middleware* em diversas plataformas de hardware. Acima dela encontra-se o *middleware*, que expõe diversas interfaces às aplicações.

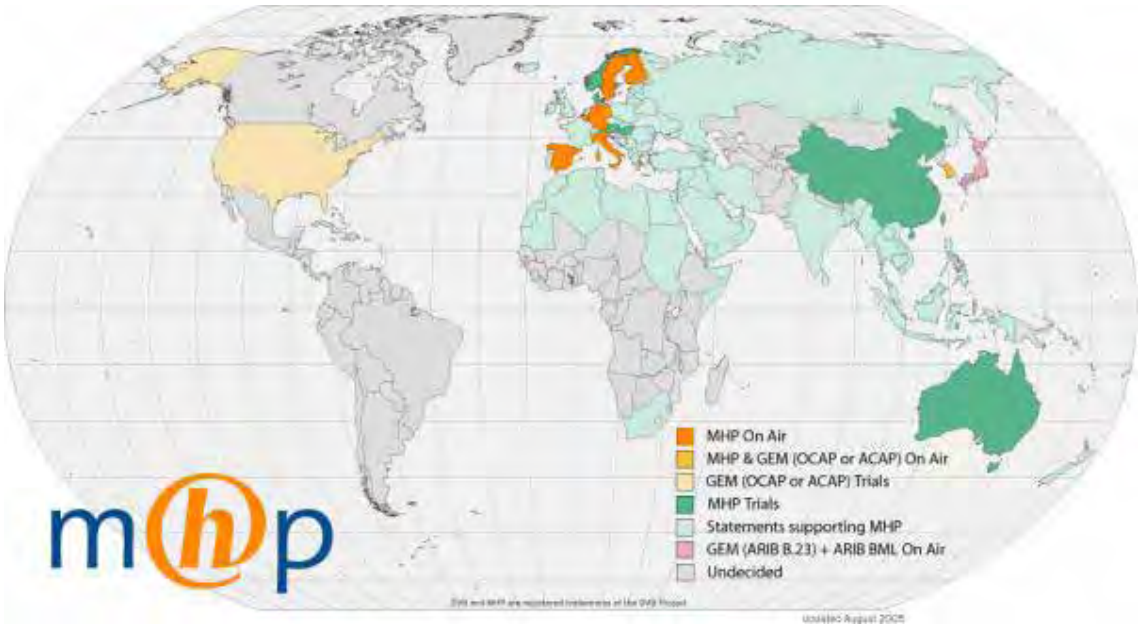

Figura 10 – Mapa de Adoção de *middlewares* - 2005 (MHP-INTERACTIVE, 2008)

Diversas especificações de *middleware* são utilizadas para diferentes sistemas de TV Digital. A Figura 10, produzida pelo projeto MHP e publicada por Steve Morris (MHP-INTERACTIVE, 2008) detalha a adoção dos diferentes *middlewares* no mundo todo em 2005. Cada especificação atua em um perfil diferente, entre aplicações procedurais ou declarativas, *low-end* ou *high-end*. As principais são:

- Para o Sistema de TV Digital ATSC: DASE (Digital TV Applications Software Environment), OCAP (OpenCable Applications Platform) e ACAP (Advanced Common Application Platform)
- Para o Sistema de TV Digital DVB: MHP (Multimedia Home Platform) e MHEG (Multimedia and Hypermedia information coding Expert Group)
- Para o Sistema de TV Digital ISDB: ARIB-AE (Association of Radio Industries and Businesses) e BML (Broadcast Markup Language – ARIB STD-B23)
- Para o Sistema de TV Digital SBTVD: Ginga-NCL e Ginga-J
- Ainda podemos citar o GEM (Globally Executable MHP) como uma plataforma comum a outras especificações de *middleware*, sendo considerada uma harmonização das especificações existentes. A seção 2.4.2.2 que detalha o sistema MHP também traz mais detalhes sobre o GEM.

Middlewares para TV Digital são soluções dirigidas para o que se chama de "mercado horizontal". Este termo implica que um consumidor pode comprar um receptor, que seja compatível com um padrão, e usá-lo para executar todas as aplicações disponíveis, independente de onde esta aplicação se origine. Considerando a enorme complexidade destas especificações, tem-se claro que tal interoperabilidade é difícil de alcançar. As ferramentas-chave para a interoperabilidade na prática são a especificação propriamente dita, suítes de testes e os programas de certificação (REIMERS, 2006).

2.4. Padrões de Sistemas de TV Digital

Atualmente quatro padrões concorrentes são utilizados em sistemas de TV Digital: DVB (Europa), ATSC (América do Norte), ISDB (Japão) e SBTVD (Brasil). Cada grupo de padrões baseou-se na experiência e necessidades específicas de uma

região, como tomadas de eletricidade, voltagem, freqüência, experiência com sistemas analógicos, sistemas de banda larga, entre outros (SOL & PINTO, 2007).

Estes padrões, cujo surgimento dos três primeiros foi contemporâneo (WU *et al*, 2006), são definidos por entidades nacionais (como a ABNT no Brasil) ou internacionais (como a ETSI no padrão europeu DVB), compostas por especialistas de governos, universidades e consórcios de indústrias que fazem parte do ecossistema de TV Analógica ou Digital, desde a produção de transmissores, moduladores, codificadores, set-top boxes, *middlewares* e mesmo emissoras. (COLLINS, 2001). Dentre outras atividades estas entidades são responsáveis pelo design, análises, testes e avaliações dos padrões e das normas originadas por eles (COLLINS, 2001).

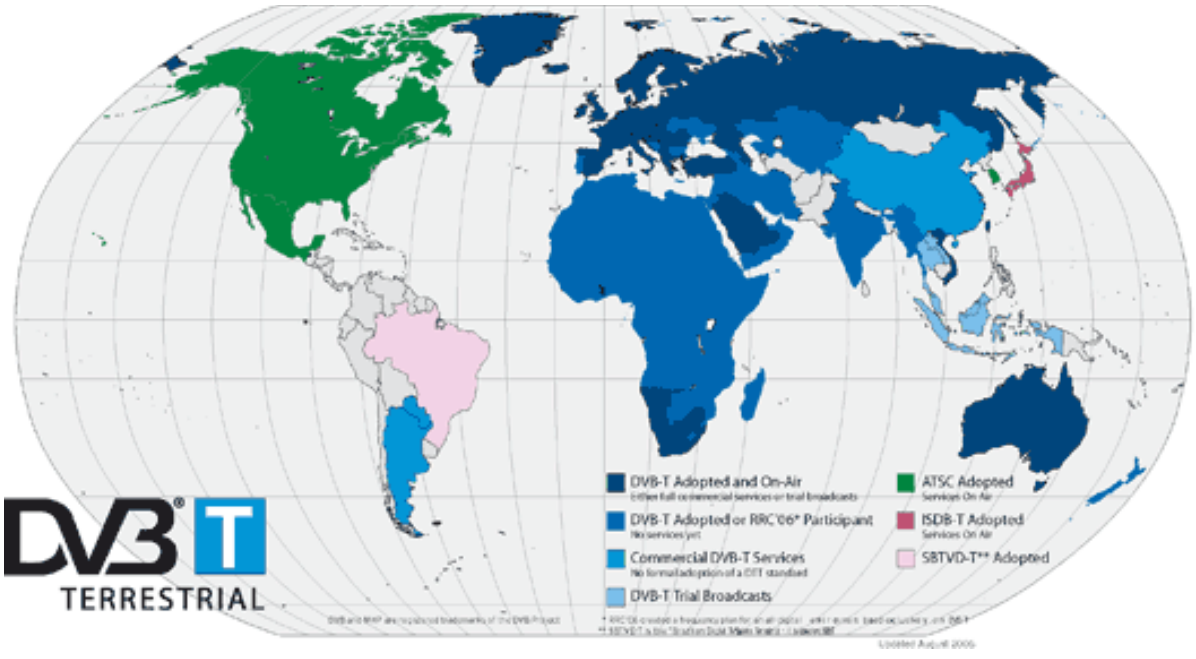

Figura 11 – Mapa de Adoção dos sistemas de TV Digital em Agosto de 2006 (MHP-INTERACTIVE, 2008)

A Figura 11 apresenta um mapa que indica o estado atual de adoção dos sistemas de TV Digital, e a Tabela 3 oferece um comparativo entre os sistemas de TV Digital citados até então.

Tabela 3 – Comparação entre os sistemas de TV digital

Padrão	Uso (princip.)	Largura de banda do canal	Codificação	Modulação	*Bit rate* (máx)	*Middleware*
ATSC	América do Norte	6 MHz	MPEG-2 e Dolby AC-3	8-VSB 16-VSB	19 Mbps	DASE, ACAP
DVB-T	Europa	5, 6, 7 ou 8 MHz	MPEG-2	COFDM, QPSK, 16-QAM, 64-QAM	32 Mbps	MHP, MHEG
ISDB-T	Japão	6 MHz	MPEG-2	BST-OFDM, QPSK, DQPSK, 16-QAM, 64-QAM	19 Mbps	ARIB BML
SBTVD-T	Brasil	6 MHz	MPEG-2 e MPEG-4 (H.264)	BST-OFDM, QPSK, DQPSK, 16-QAM, 64-QAM	19 Mbps	Ginga-J, Ginga-NCL

2.4.1. Advanced Television Systems Committee (ATSC)

A série de padrões que definem o sistema de TV Digital terrestre ATSC foi desenvolvida pela organização norte-americana de normatização de TV Digital homônima - Advanced Television Systems Committee (ATSC) (WU *et al*, 2006). Estes padrões foram inicialmente adotados pelos Estados Unidos (U.S. Federal Communications Commission - FCC), Canadá, e Coréia do Sul (COLLINS, 2001), Taiwan (MOURA, 2006) e por fim sendo definidos como padrão no México em 2004 (JONES, 2006).

Existe ainda o padrão norte-americano de TV Digital por cabo, em uso atualmente nos Estados Unidos, que foi desenvolvido separadamente com base no trabalho feito pelo Cable Television Laboratories (CableLabs) e codificado pela Society of Cable Telecommunications Engineers (SCTE) (WU *et al*, 2006).

Resultado de diversos anos de pesquisa, o ATSC inclui diversos subsistemas para produção, codificação, transporte, transmissão e recepção de vídeo e áudio digital de alta qualidade, bem como dados, por sistemas *over-the-air* ou por cabo (COLLINS, 2001). O sinal digital de input para o sistema de transmissão ATSC é um fluxo de

transporte MPEG-2, serial e síncrono, a uma taxa constante de 19,39 Mbps (carga útil de 19,2895 Mbps). Como modulação o ATSC utiliza o Vestigial Sideband de oito níveis (8-VSB) (COLLINS, 2001).

Com o objetivo de garantir a interoperabilidade na troca de dados (*datacasting*), foi definido o padrão ATSC Data Broadcast, que descreve formatos, protocolos e procedimentos de *signaling* e anúncio de informações, que permitem ao *middleware* conhecer a presença de serviços de dados. Estes formatos e protocolos, que podem ser visualizados na forma de camadas na Figura 12 são baseados na estrutura do *Transport Stream* do MPEG-2 (CRINON *et al*, 2006).

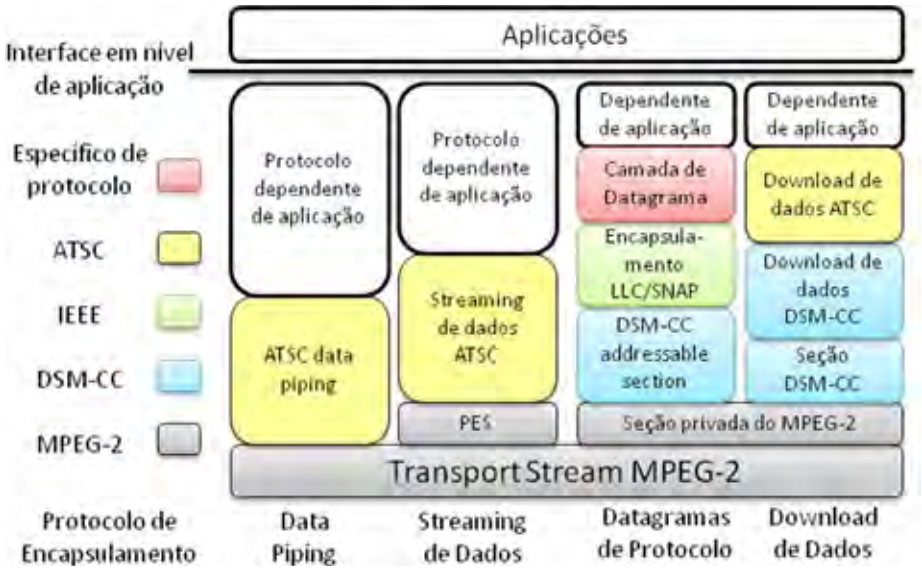

Figura 12 - Pilha de protocolos para o ATSC (CRINON et al, 2006)

Conforme detalhado por Crinon *et al* (CRINON *et al*, 2006), cada coluna da Figura 12 representa uma família de protocolos com uma finalidade específica:

- *Data Piping*: dependente de aplicação, empacotado diretamente no *Transport Stream* do MPEG-2. Utilizados para requisitos não atendidos pelos protocolos padrão;

- *Data Streaming*: é utilizado para dados que utilizam o sistema de timing do MPEG-2 (clock de referência do programa – PCR - e marcação de tempo de apresentação - PTS). Os dados são empacotados em Packetized Elementary Streams (PES);
- Datagramas de Protocolo: informações dos protocolos de rede da família IP, IPX e OSI, através de *data units* encapsuladas DSM-CC *addressable sections*. Os cabeçalhos Logical Link Control/Subnetwork Access Point (LLC/SNAP) são usados dentro das *addressable sections* para identificar o protocolo;
- Encapsulamento de Download de Dados: transporte de arquivos baseado no DSM-CC em modo carrossel.

O ATSC define tabelas para indicar ao middleware onde encontrar os dados de controle. A tabela Virtual Channel Table (VCT) traz uma lista de canais virtuais e a Program Map Table (PMT) traz a tabela de programas disponíveis. A tabela PMT é transmitida pelo menos uma vez a cada 400 ms (CRINON *et al*, 2006). Estas tabelas podem ainda apontar para outras, como a Data Service Table (DST), utilizada para encontrar os serviços de dados no *Transport Stream*. Ela pode ter entradas para uma ou mais aplicações, que por sua vez podem apontar para seus recursos (CRINON *et al*, 2006).

O ATSC define também um protocolo conhecido como Program and System Information Protocol (PSIP), responsável pela publicação do tipo de aplicação, permitindo aos receptores do Guia de Programação Eletrônico (EPG) a informação de quando um *virtual channel* é um canal exclusivo para serviços de dados, quando é um canal de áudio e dados ou um canal de vídeo, áudio e dados (CRINON *et al*, 2006).

2.4.1.1. DTV Application Software Environment (DASE) e Advanced Common Application Platform (ACAP)

O DASE (desenvolvido entre 1996 e 2003) e o ACAP são padrões de *middlewares* e protocolos de dados (assíncronos e síncronos) para permitir o desenvolvimento, distribuição e gerenciamento de aplicações com *datacasting* e *iTV* sobre uma infra-estrutura unificada.

Figura 13 – Recursos disponíveis para uma aplicação interativa (CRINON et al, 2006)

A Figura 13 ilustra os recursos disponíveis para uma aplicação executando sobre o DASE e também a seqüência de execução deste middleware. O primeiro estágio (na base do diagrama, da esquerda para a direita) é responsável pela sintonização. O segundo estágio é relacionado ao *Transport Stream*, sendo responsável por isolar (demultiplexar) os fluxos específicos. O terceiro estágio (que pode ser utilizado pelos itens em azul, ao centro do diagrama) isola os programas específicos, ou tabelas específicas dentro de programas específicos. Os estágios subseqüentes na base do diagrama são relacionados ao conteúdo. A decriptação recupera o fluxo de mídia e a decodificação trata diretamente o fluxo de mídia.

A especificação do DASE suporta quatro camadas de apresentação que podem apresentar os conteúdos sobrepostos na tela. A primeira camada (de traz para frente) armazena a imagem de fundo, a segunda é usada pelo vídeo, a terceira para gráficos e a quarta para legendas (CRINON *et al*, 2006).

A Figura 13 também ilustra os componentes que suportam os serviços e aplicações. As aplicações podem criar filtros que realizam o *parse* dos *metadados* do serviço para isolar os serviços de interesse. A aplicação pode então, tendo permissão para tal:

- Criar preferências; requisitar preferências; configurar opções padrão da plataforma (como a localidade para os textos);
- Selecionar a linguagem; configurar a linguagem padrão para o dispositivo de áudio e o dispositivo de captura;
- Configurar o hardware de transporte; registrar interesse em eventos de transporte, por exemplo, a alocação ou liberação dos recursos de transporte;
- Configurar o hardware de mídia; controlar os fluxos de mídia; registrar interesse em eventos de mídia, por exemplo, mudanças no formato de vídeo;
- Configurar o canal de retorno; criar *sockets* e trocar datagramas;
- Criar filtros de seção; extrair tabelas específicas no *Transport Stream*;
- Criar filtros de serviço; navegar nos metadados do serviço; registrar interesse em eventos, como a introdução, troca ou remoção de um serviço;
- Seleção de serviço; resolver o nome do serviço para o endereço do serviço; controlar a execução do serviço; registrar interesse em eventos, como a transição da máquina de estado de execução;
- Criar filtros para isolar as aplicações anexas ao serviço e então requisitar à plataforma a executar a aplicação (a plataforma extrai a aplicação do

Carrossel de Objetos, cria o contexto no qual a aplicação executa e executa a aplicação). Navegar pelas aplicações disponíveis; controlar a execução de aplicação; registrar interesse em eventos de aplicação, como a transição da máquina de estado de execução;

- Navegar pelo carrossel de objetos; carregar objetos específicos do carrossel de objetos; registrar interesse em eventos do carrossel de objetos, como a inclusão, troca ou remoção de objetos;
- Acessar o armazenamento persistente;
- Criar *widgets*; configurar a posição, ordem e foco do widget; compor imagens na camada para gráficos; registrar interesse em eventos de interação com o dispositivo, como o status do foco;

Para lidar com a distância até a tela, o conjunto de *widgets* padrão é especial. O *middleware* provê eventos de controle remoto bem como eventos de teclado. O *middleware* permite às aplicações interpor filtros para isolar eventos específicos. A aplicação não precisa ter o foco para receber eventos de interação com o dispositivo. Se a aplicação é um guia de programação, por exemplo, a aplicação pode escutar pelos eventos de controle remoto e, ao receber e perceber estes eventos, ativar seus *widgets* (CRINON *et al*, 2006).

2.4.2. Digital Video Broadcasting (DVB)

O DVB, uma série de padrões que correspondem ao Sistema de TV Digital (vídeo, áudio e serviços de dados) europeu, foi desenvolvido pela organização européia DVB e em 1991 padronizado pelo European Telecommunication Standard Institute (ETSI) (BARBOSA & VALENTE, 2002; REIMERS, 2006). Dentre estes padrões encontram-se:

- DVB-S: utiliza satélites para transmissão de televisão digital
- DVB-S2: a nova geração do sistema por satélites
- DVB-C: utiliza sistemas de cabo digital para transmissão de televisão digital
- DVB-T: utilizado em sistemas ambientes terrestres para transmissão de televisão digital
- DVB-MC/S: utiliza sistemas de distribuição de vídeo multiponto com microondas
- DVB-SI: define as estruturas de dados que acompanham o sinal da televisão digital
- DVB-CA: define os padrões de segurança da televisão digital
- DVB-CI: define uma interface comum para o sistema de segurança da televisão digital
- DVB-I: apresentação da ITV
- DVB-H: utilizado em Handhelds

Lançado oficialmente em 1993, estes padrões foram adotados para entrega de sinal por satélite, cabo e terrestre através de broadcasting, mas hoje também trabalham sobre redes IP. Atualmente este sistema é adotado pelos 15 membros da União Européia, além de Austrália e Nova Zelândia. Hoje participam da construção da norma mais 200 membros de mais de 30 países (BARBOSA & VALENTE, 2002; COLLINS, 2001).

Segundo (REIMERS, 2006), o DVB foi projetado para atingir (e o fez com sucesso) os seguintes objetivos:

- Transmissão de imagens em alta definição (HDTV), mesmo através de redes de *broadcasting* terrestres;
- Permitir transmissão de programas em definição *standard* (SDTV) usando canais com menor banda, permitindo um aumento do número de programas oferecidos com a alocação existente de canais de transmissão;
- Permitir broadcast de conteúdo para receptores de TV portáteis de baixo custo, equipados com antenas receptoras embutidas, garantindo recepção estável de diversos programas de televisão;

- Receber sinal de televisão estável em veículos em movimento (trens, ônibus ou carros), em alta qualidade, mesmo em altas velocidades;
- Suportar o broadcast de programas de radio e permitir a transmissão de dados para propósitos de entretenimento e de negócios;
- Suportar a utilização de métodos seguros para serviços pagos, garantindo que acesso não autorizado a tais serviços seja extremamente difícil, senão impossível;
- Suportar serviços interativos por completo, trazendo padrões para os canais de interação entre o expectador e a rede da operadora ou o provedor de conteúdo;
- Prover uma plataforma de software interoperável para serviços avançados, como interatividade e mesmo acesso à internet em receptores de TV;
- Como uma técnica de transmissão de dados, o DVB precisa garantir outras características de tecnologias digitais, como a estabilidade de recepção dentro de uma área de cobertura bem definida, a possibilidade de simples distribuição sobre redes de telecomunicação como um serviço dentre outros.

O DVB-T é projetado para realizar a transmissão sobre os canais existentes de 7 ou 8 MHz, é capaz de entregar informações digitais a taxas de 4,98 a 31,67 Mbps. Ele é similar em muitos aspectos com os padrões de DTV dos Estados Unidos e do Japão (por exemplo, o uso do MPEG-2, camada de transporte e na codificação do canal) (REIMERS, 2006), porém existem também diferenças importantes e significantes tanto na codificação do canal quanto na modulação.

Uma delas está no tipo de modulação utilizada: o sinal de saída do bloco modulador é um sinal Coded Orthogonal Frequency-Division Multiplex (COFDM), que procura atender aos requisitos das estações e redes de broadcast européias (LADEBUSCH & LISS, 2006; COLLINS, 2001). O modulador pode ser dividido em duas tarefas principais: codificação do canal e modulação propriamente dita. As funções

executadas no codificador do canal incluem dispersão de energia ou randomização dos dados, codificação R/S, intercalação, codificação Trellis e intercalação novamente. As funções do modulador incluem mapeamento, adaptação do quadro e inserção do sinal *pilot* para sincronia (COLLINS, 2001; LADEBUSCH & LISS, 2006; REIMERS, 2006).

Assim como em outros sistemas de TV digital, o sinal digital de entrada é um fluxo de transporte síncrono MPEG-2 (LADEBUSCH & LISS, 2006). Porém, Reimers (REIMERS, 2006) discute que com o tempo, uma maior variedade de formatos de áudio foi requisitada pelas emissoras e passou a ser suportado pelo DVB, como o Dolby AC-3, DTS e o AAC (MPEG-4).

Em geral, os fluxos primários do DVB-T são compostos de dados (como teletexto, legendas, TV Anytime, pacotes IP e serviços baseados no carrossel de dados), vídeo e áudio. Os serviços DVB são tipicamente compostos de uma grade variedade de programas carregados por diversos canais de transmissão. Para facilitar o acesso aos serviços pelos usuários (ser capaz de sintonizar em um canal e navegar dentre os programas) foi definido uma série de tabelas, conhecido como Service Information (SI) ou DVB-SI (REIMERS, 2006; LADEBUSCH & LISS, 2006), são elas:

- Network Information Table (NIT): descreve parâmetros de modulação e freqüências alternativas;
- Service Description Table (SDT): traduz números de programas em nomes de serviços, podendo conter também informações como linguagem ou a relação entre largura e altura (ex: 16:9);
- Event Information Table (EIT): traz o nome do evento, hora de início, duração, classificação do conteúdo e outras descrições. Esta tabela é bastante útil para implementar o Electronic Program Guide (EPG).

Na Figura 14 de Reimers (REIMERS, 2006) é possível visualizar como o Service Information é encapsulado dentro de um contêiner de dados DVB. Além do SI, encontra-se nesta figura o PSI (Program Specific Information), usado para gerenciar o DVB e enviar quaisquer outros dados relativos às aplicações, mantendo a organização deste contêiner.

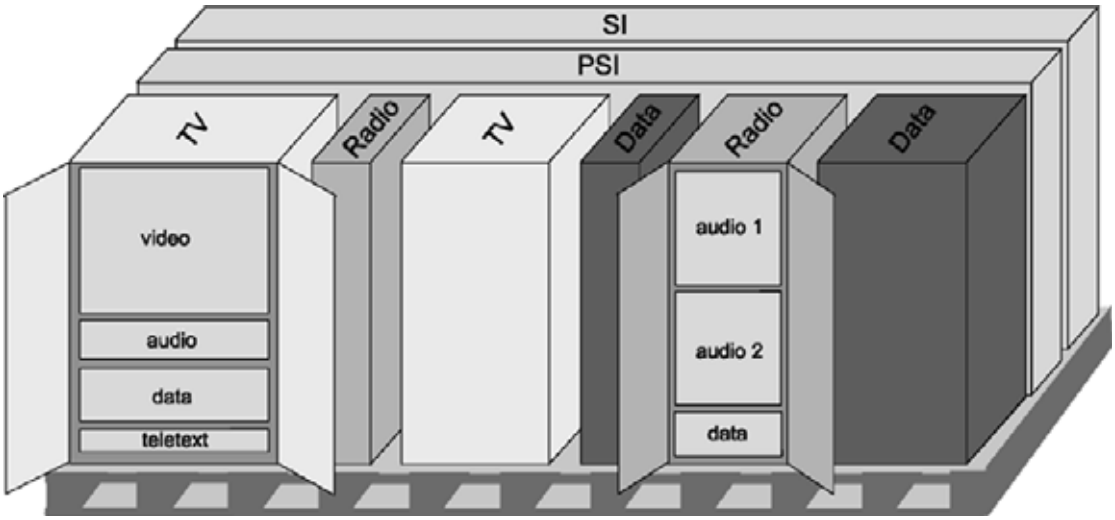

Figura 14 - Componentes dentro do container de dados DVB (REIMERS, 2006)

A Figura 14 pode ser visualizada como uma representação visual dos dados que podem estar multiplexados no *Transport Stream*. Independente do tipo de serviço que o dado digital representa, ele pode ser inserido dentro de um *Transport Stream* MPEG-2. Assim como no ATSC, dependendo da necessidade da aplicação provisões adicionais precisam ser feitas para assegurar a transmissão correta e para garantir que o receptor entenda o serviço transmitido. Em conseqüência, quatro tipos de transmissão foram definidos no DVB: *data piping*, *data streaming*, uso do carrossel de dados ou objetos e *multiprotocol encapsulation* (REIMERS, 2006).

Por várias razões, os fabricantes de receptores DVB desejam poder atualizar o software *middeware* de seus produtos (em especial aqueles que já estiverem nas casas

dos telespectadores). O DVB fornece então uma especificação para atualização de software conhecida como System Software Update (SSU).

2.4.2.1. Multimedia Home Platform (DVB-MHP)

Como apenas a especificação da plataforma de transporte não é suficiente para o desenvolvimento de serviços e para a sua interoperabilidade dos terminais, as organizações-membro do Consórcio DVB passaram a especificar uma solução técnica para o terminal do usuário: o Multimedia Home Platform (MHP) (REIMERS, 2006).

Conforme detalhado na Figura 15, a especificação do MHP oferece um conjunto consistente de recursos e funções necessárias para três diferentes perfis de aplicações (MHP WHITE PAPER, 2007; PIESING, 2006):

- Enhanced Broadcast: serviços de broadcast unidirecional, ou que necessitam apenas interatividade local;
- Interactive Broadcast: serviços interativos adicionais, através de um canal de retorno (bidirecional). É definido como um superset do Enhanced Broadcast;
- Internet Access: fornece um conjunto de recursos de software necessários para comunicação global através da internet.

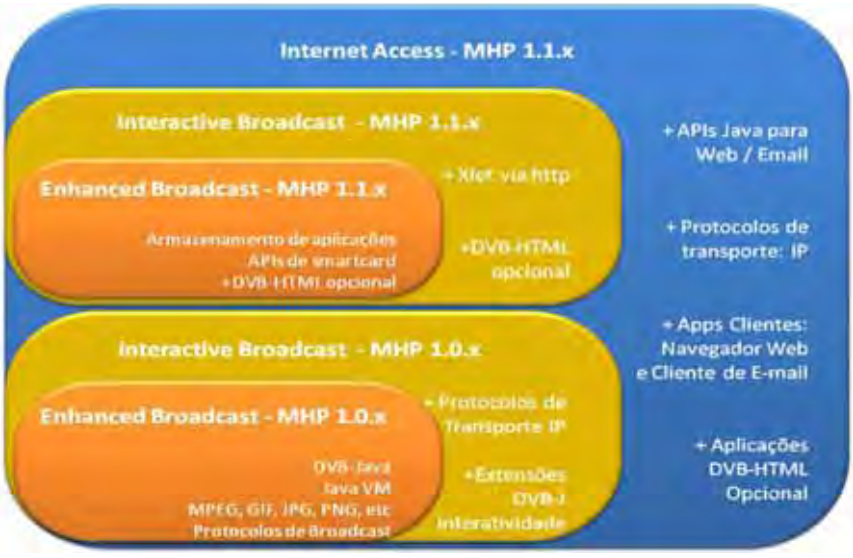

Figura 15 - DVB-MHP Perfis 1, 2 e 3 (MHP WHITE PAPER, 2007)

Uma forma de entender o MHP é descrita na Figura 16, que traz uma visão simplificada de um *middleware* MHP. A arquitetura básica do MHP começa com os recursos de hardware (na base da pilha - dispositivos de I/O, CPU, memória e sistema gráfico), o sistema básico (system software), o qual o fabricante irá escolher e também oferece o processamento MPEG.

Figura 16 – Arquitetura básica do MHP (REIMERS, 2006)

Sobre o sistema básico o MHP define como *core* da plataforma DVB-J uma Java Virtual Machine (JVM), em conformidade com a especificação fornecida pela Sun Microsystems, que fornece várias interfaces de programação, algumas das quais são baseadas em elementos da PersonalJava (apenas na especificação do MHP 1.0, sendo posteriormente a PersonalJava substituída pelo seu sucessor chamado de "Personal Basis Profile", definido pela Java Community Process - JCP). Outras APIs foram desenvolvidas para atender requisitos específicos da TV Digital, como o Digital Audio Visual Council (DAVIC), a Home Audio Visual Initiative (HAVi) – que define o conjunto de *widgets* da interface com o usuário - e a especificação da Sun Microsystems para televisão, conhecida como JavaTV (PIESING, 2006). Esta camada também inclui Gerenciador de Aplicações (também chamado de *navigator*). O conjunto total de APIs

cria a API MHP na qual aplicações interoperáveis (que são transmitidas para o terminal MHP) podem ser executadas (REIMERS, 2006; MHP WHITE PAPER, 2007).

O MHP é um sistema bastante complexo, e na prática inclui muitos outros aspectos do que os apresentados na Figura 16. Entre estes aspectos estão os protocolos de transporte, tanto para o *broadcast* quanto para o canal de interatividade. Existem formatos de conteúdo estático e dinâmico que pode ser parte de uma aplicação MHP. O MHP utiliza um modelo de ciclo de vida de aplicações que define os seus vários estados. Existe o protocolo de sinalização, incluindo a Application Information Table (AIT) que informa ao receptor as aplicações disponíveis em casa serviço. Um *framework* seguro de áudio também é definido, uma vez que aplicações executando em um receptor pode potencialmente danificar o dispositivo. Um modelo de referência gráfico também é utilizado, com o qual é possível garantir um *look-and-feel* consistente entre as várias aplicações executando em terminais de vários fabricantes (REIMERS, 2006).

2.4.2.2. Globally Executable MHP (GEM)

Quando lançado em 2002 como um padrão aberto, o MHP despertou grande interesse mundial e se tornou referência para diversos outros padrões. Porém, como a pilha de protocolos do DVB não é usada por completo nestes outros países (que desenvolveram outros protocolos na camada de transporte para finalidades semelhantes) a adoção global do MHP necessita versões modificada. Sendo assim, o Projeto DVB criou uma especificação conhecida como Globally Executable MHP (GEM), que cria uma plataforma de software que pode ser incorporada em outros sistemas de TV Digital pelo mundo, como sistemas de TV a cabo (OCAP nos Estados Unidos), terrestre (ACAP nos Estados Unidos, ARIB STD B-23 no Japão e Ginga-J no Brasil). O GEM define os equivalentes funcionais para os requisitos do DVB para sistemas não-DVB

(REIMERS, 2006). O Ginga, em sua parte procedural (Ginga-J) também é baseado no GEM.

2.4.2.3. OpenMHP

OpenMHP é um projeto criado em 2002 na Finlândia com o objetivo de incentivar o desenvolvimento de serviços e aplicações baseados na especificação DVB-MHP. Ele é um projeto open source e gratuito para que desenvolvedores, pequenas empresas, universidades e outras organizações pudessem desenvolver trabalhos relacionados ao DVB-MHP (PAKARINEN et al, 2004).

Este projeto pode ser definido como um conjunto de bibliotecas para fornecer classes compatíveis com MHP para aplicações interativas em um sistema de TV Digital. Ele possui uma biblioteca MHP, implementada em Java e independente de plataforma, implementações parciais das APIs do HAVi, DAVIC e DVB-MHP e uma biblioteca chamada de Adaptation Layer responsável por gerenciar requisições de aplicações para o nível de hardware e vice-versa. Os módulos da Adaptation Layer são listados na Figura 17.

Figura 17 – Camada de Adaptação do OpenMHP

Para executar o OpenMHP é necessário um ambiente com o Java Runtime Environment (JRE) compatível com a versão 1.4 ou superior, a implementação de referência das APIs do JavaTV, e o Java Media Framework 2.1 (JMF 2.1) - ambos são desenvolvidos e disponibilizados pela Sun Microsystems. Para compilar este ambiente,

é preciso adicionalmente o Java Development Kit compatível com a versão 1.4 ou superior.

No OpenMHP, o Service Information não foi implementado. Referências para as aplicações são armazenadas no seu application database, que atua como um servidor que pode receber requisições e devolver as aplicações ou realizar alterações (incluir, alterar parâmetros, remover).

Existe suporte a mecanismos de eventos, inclusive com a implementação de um emulador de controle remoto.

O OpenMHP segue o esquema de sobreposição de superfícies gráficas definidas no MHP, separadas em camadas. No OpenMHP são chamadas de BackgroundLayer, VideoLayer, GraphicsLayer, SubtitleLayer e ExtraLayer. A BackgroundLayer é usada para exibir I-frames do MPEG-2 como imagem de background. A VideoLayer exibe o fluxo de vídeo, a GraphicsLayer apresenta o gráfico das aplicações e a SubtitleLayer exibe as legendas. A ExtraLayer pode ser usada para exibir, por exemplo, o logo da emissora do canal, uma animação ou o cursor do sistema. Todas estas camadas são combinadas em uma única superfície e adicionadas a um objeto MHPDevice.

O MHPDevice é uma abstração de uma Set-Top Box e pode ser reconfigurado, buscando abstrair diferentes dispositivos receptores, alterando o esquema de cores, resolução e *aspect ratio*, por exemplo.

A manipulação de vídeos no OpenMHP (na VideoLayer, simulando a decodificação de vídeo que seria feita por um hardware específico em uma Set Top Box) é feita utilizando-se a JMF 2.1, que disponibiliza diversos codecs. Porém, uma das desvantagens do uso da JMF nesta camada é que o uso do objeto Player para execução de arquivos de mídia impede que sejam renderizados outros componentes visuais sobre esta camada, impedindo que as camadas GraphicsLayer, SubtitleLayer e ExtraLayer

sejam apresentadas com seus respectivos componentes. Uma alternativa seria no uso do objeto Processor, que pode ser usado para implementar seus próprios filtros para o vídeo antes da apresentação, podendo tratar da sobreposição de camadas com soluções "*hard coded*". Porém estas soluções exigiriam profundo conhecimento nos codecs disponíveis na JMF, bem como do tratamento e eventos para objetos desenhados manualmente sobre os frames do vídeo, de forma que esta solução foi descartada e para efeito deste trabalho não há prejuízos.

Na Figura 18 está um *screenshot* de uma aplicação MHP executando sobre a OpenMHP. À esquerda encontra-se a janela Output com informações de debug e trace fornecidas pelas diversas APIs do OpenMHP, em especial do ClassLoader. Ao centro está o MHPDevice, responsável por apresentar o conteúdo. E na janela à direita encontra-se o emulador de controle remoto que envia seus comandos ao servidor de eventos.

Figura 18 – Aplicação HelloWorld executado sobre o OpenMHP

No MHPDevice (janela ao centro da Figura 18) é possível observar a camada BackgroundLayer (neste caso uma imagem fixa, com diversas faixas verticais com

cores distintas), a GrahicsLayer (com a aplicação HelloWorld) e a SubtitleLayer (com as legendas, na parte inferior apresentando o texto “what is that?”).

2.4.3. Integrated Services Digital Broadcasting (ISDB)

No Japão, o Digital Broadcasting Experts Group (DiBEG) desenvolveu em 1999 um conjunto de especificações baseado no DVB para broadcast digital de televisão, áudio e serviços de dados, através de sinais por satélite, cabo ou terrestre. Posteriormente estas especificações foram padronizadas pela Association of Radio Industries and Business (ARIB) e pela Japan Cable Television Engineering Association (JCTEA).

Denominado Integrated Services Digital Broadcasting (ISDB), este conjunto de especificações define um sistema de transmissão básico que fornece o broadcast digital de televisão, incluindo a modulação e a codificação do canal (MOURA, 2006). Algumas destas especificações são listadas abaixo (HORI & DEWA, 2006; YOSHIMURA, 2006):

- ARIB TR-B13: guias operacionais para broadcast de som digital terrestre;
- ARIB TR-B14: guias operacionais para broadcast de televisão digital terrestre;
- ARIB TR-B15: guias operacionais para broadcast por satélite;
- ARIB TR-B26: guias operacionais para broadcast de som por satélite;
- ARIB STD-B24: consiste de três partes (codificação de dados, esquema de codificação multimídia baseado em XML e especificação de transmissão de dados;
- ARIB STD-B25: especificação do sistema de acesso condicional para broadcast digital;
- ARIB STD-B24: Data Coding and Transmission Specification for Digital Broadcasting;
- ARIB STD-B21: receptor para broadcast digital;
- ARIB STD-B10: serviços de informações para sistema de broadcast digital.

Segundo Moura e Uehara, o sistema de TV digital ISDB possui como característica (MOURA, 2006; UEHARA, 2006):

- Alta definição;

- Mobilidade (acesso em dispositivos móveis, como celulares, ou conteúdos HDTV para veículos em movimento);
- Portabilidade;
- *Datacasting* (serviços de dados);
- Interatividade;
- Flexibilidade (transmissão hierárquica e recepção parcial).

O padrão de transmissão para televisão digital terrestre é similar em muitos aspectos com o padrão DVB-T, e neste conjunto é chamado de ISDB-T. Em comum com os demais padrões mundiais, o sinal de entrada digital para o sistema de transmissão ISDB-T é um transporte MPEG-2 composto de pacotes de dados de 187 bytes mais um byte de sincronização, que é responsável pela codificação e multiplexação dos fluxos elementares. O sinal de áudio é codificado pelo AAC (SAKURAI, 2006). A carga útil pode incluir pacotes codificados de vídeo digital, áudio digital, texto, gráficos, e dados (COLLINS, 2001).

Uma diferença-chave com relação ao DVB-T é o uso da band-segmented transmission–OFDM (BST-OFDM). Este é uma abordagem para segmentação de dados que permite que a banda seja alocada para vários serviços, incluindo dados, radio, televisão em definição padrão (SDTV), e televisão em alta definição (HDTV) de maneira mais flexível, serviços móveis e portáteis (COLLINS, 2001), permitindo que sejam transmitidos dados, imagem e som, com tipos de modulação e taxas de transmissão diferentes (MOURA, 2006). Esta abordagem traz de fato uma transmissão hierárquica e a possibilidade de recepção parcial do conteúdo transmitido (UEHARA, 2006).

Outras diferenças estão na utilização dos canais de 7 ou 8 MHz, podendo entregar informação digital a uma taxa de dados de 3,561 a 30,980 Mbps (COLLINS, 2001). Existem diferenças também na implementação na codificação do canal, onde cada grupo de segmentos pode receber uma modulação, codificação para correção de

erros e intercalador de tempo, de acordo com o propósito de cada serviço. Esta nova abordagem no sistema ISDB permite maior flexibilidade no ajuste dos parâmetros (COLLINS, 2001; ASAMI & SASAKI, 2006).

Os muitos portadores modulados separados podem empregar os mesmos três padrões de constelação fornecidos no padrão DVB-T: QPSK, 16 QAM ou 64 QAM. Adicionalmente, é disponibilizado o Differential Quadrature-Phase Shift Keying (DQPSK) (COLLINS, 2001).

No sistema ISDB é possível realizar o download do software do receptor através do canal de broadcast e executar a atualização de versão do software do receptor. A tabela Software Download Trigger Table (SDTT) inclui o nome do fabricante, tipo do produto, identificação do serviço, e o agendamento de download. O receptor memoriza a tabela SDTT durante a recepção de TV, porém é somente quando o usuário desliga o seu receptor que ele automaticamente muda o canal para o especificado e espera pelos dados do download. Quando o receptor acha os dados que precisa, ele realiza o download para a sua memória, e após a checagem com sucesso ele sobrescreve seu próprio software com o novo (SAKURAI, 2006).

2.4.3.1. Broadcast Markup Language (BML) e Application Execution Engine Platform (ARIB-AE)

Para broadcast de dados, o ISDB utiliza o Broadcast Markup Language (BML), uma máquina de apresentação declarativa com programas baseados em XML. Depois de padronizado, o BML se tornou o ARIB STD B24 "Data Coding and Transmission Specification for Digital Broadcasting".

A especificação do BML é dividida em três partes: Codificação de dados, Esquema de codificação de multimídia XML e Especificação de transmissão de dados. Os requisites considerados no design do BML foram:

- Harmonização com conteúdos Web;
- Extensibilidade;
- Apresentação de conteúdo rico;
- Apresentação única (interoperabilidade).

A pilha de software necessária em um receptor ISDB pode ser visualizada na Figura 19. Ela consiste de cinco camadas: camada de *device drivers*, sistema operacional de tempo real, camada baixa do *middleware*, camada alta do *middleware* e camada de aplicação. Neste modelo, a camada do *middleware* é dividida em duas camadas, a camada baixa está conectada aos *device drivers* e a camada alta é responsável por fornecer as interfaces de programação (APIs) às aplicações (SAKURAI, 2006).

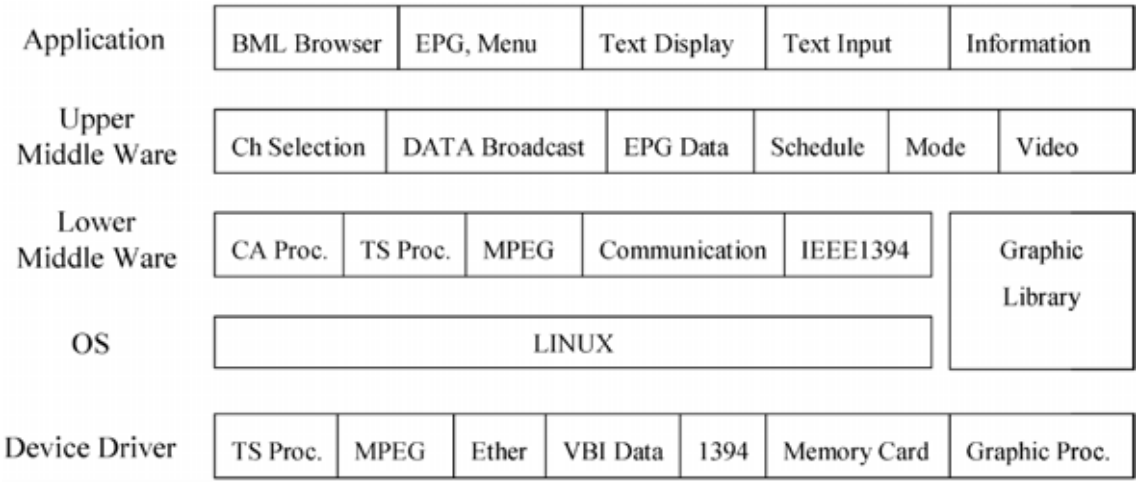

Figura 19 – Pilha de software para um receptor ISDB (SAKURAI, 2006)

No *middleware* tem-se o processo de seleção do canal, onde o módulo de processamento do *Transport Stream* (TS proc.) adquire o sinal de áudio e vídeo e os dados SI. Em combinação com o módulo de processamento do Acesso Condicional (CA Proc.) e o módulo MPEG, o sinal de áudio e vídeo é decodificado no sinal *base-band*. O

módulo de comunicação pode operar através do modem e de TCP/IP (SAKURAI, 2006).

Na camada de aplicações temos a renderização dos menus e o EPG. Dos dados do EPG é produzida a tabela de programa e exibida na tela da TV. Para o broadcast terrestre, a aquisição de dados do EPG é feita enquanto o receptor encontra-se em modo *stand by*. O browser BML consiste de um *core* de um browser, um bloco *parser* e um bloco de processamento de ECMAScript (SAKURAI, 2006).

O BML é totalmente baseado no XML e utiliza um draft inicial do XHTML 1.0. Alguns subsets do CSS 1 e 2 são suportados para controle do layout da tela e controle da apresentação, bem como o ECMAScript para controle dos conteúdos (SAKURAI, 2006; HORI & DEWA, 2006). O Document Object Model (DOM) é usado como interface entre o script e o conjunto de *tags*. Outras extensões às tecnologias citadas anteriormente são utilizadas no BML para se adequar ao cenário de broadcast de aplicações (HORI & DEWA, 2006).

Para a transmissão dos documentos BML e mídias por ele referenciadas, utiliza-se o DSM-CC. O sistema de carrossel de dados é mais fácil de manipular e a recepção destes dados necessita de menor overhead do que o sistema de carrossel de objetos. Cada módulo transmitido pelo carrossel de dados é ajustado de acordo com o formato de entidade definido no HTTP/1.1. Esta entidade consiste de um corpo de entidade para armazenar um documento BML e mídia e uma entidade cabeçalho para descrever os *metadados*, como o tamanho do arquivo. O Multipurpose Internet Mail Extension (MIME) é usado para empacotar documentos multi-BML ou um conjunto de mídias em um módulo. Ao contrário do MIME definido na RFC 2045, não é necessário converter dados binários em texto neste formato de entidades, sendo possível armazenar dados como imagens sem alterar seu formato (HORI & DEWA, 2006).

Além do BML, o ISDB adicionou em 2003 o suporte para conteúdos e aplicações procedurais através do ARIB STD B-23, o "Application Execution Engine Platform", também conhecido como ARIB-AE. O ARIB-AE foi amplamente baseado no sistema DVB, uma vez que procura a interoperabilidade e a portabilidade internacional em aplicações futuras. A *engine* de execução de aplicações (também conhecida como ARIB-J) foi estabelecida como uma extensão da especificação do MHP, mais especificamente na especificação de harmonização de *middlewares* denominada GEM (ARIB STD-B23, 2004).

2.4.4. Sistema Brasileiro De Televisão Digital Terrestre (SBTVD)

No Brasil, o desenvolvimento de um sistema de TV Digital procurou aproveitar as experiências dos sistemas já formulados por todo o mundo. Desta forma, diversas instituições de P&D, Universidades e centros de pesquisa brasileiros foram qualificados pela FINEP, através da chamada pública MC/MCT/FINEP/FUNTTEL - 01/2004 a apresentar propostas de projetos nas áreas e temas definidos como prioritários pelo Sistema Brasileiro de Televisão Digital. Juntamente com estas organizações, empresas (emissoras, empresas de TI, telecomunicações) e membros do governo federal formou-se o consórcio responsável pela definição do Sistema Brasileiro de TV Digital Terrestre (ARQUITETURA DE REFERÊNCIA, 2006; SOUZA FILHO *et al*, 2007).

Posteriormente, estes grupos foram formados para que se pudessem traçar modelos de referência e avaliar o impacto dos modelos de negócios no Brasil. A especificação final ficou conhecida como SBTVD e traz influencias principalmente do sistema de TV Digital japonês. A especificação técnica de referência do SBTVD é dividida em:

- Sistema de transmissão: responsável por receber o fluxo de dados, codificar o canal, adicionar a proteção contra erros, prepará-lo para envio por RF, configurar os segmentos e fazer a modulação de acordo com a transmissão hierárquica. O espectro de freqüência possui largura de 5,7 MHz, a mesma das freqüências utilizadas no sistema analógico no Brasil (NBR 15601, 2008);
- Codificação de vídeo, áudio e multiplexação: Para vídeo é suportado o ITU-T H.264, baseado no MPEG-4 Part 10 ou AVC (Advanced Video Coding), na razão de aspecto 4:3 ou 16:9 e em SD ou HD. Para codificação do áudio é utilizado o MPEG-4 AAC (multicanal ou estéreo). O *Transport Stream* é baseado no MPEG-2 (NBR 15602-1, 2008; NBR 15602-2, 2008; NBR 15602-3, 2008);
- Multiplexação e serviços de informação (SI): especifica as tabelas de serviço de informação, conhecidas por tabelas SI. Elas permitem a configuração automática do receptor para que este possa demultiplexar e decodificar as várias transmissões de programas existentes em uma multiplexação (NBR 15603-1, 2008; NBR 15603-2, 2008; NBR 15603-3, 2008);
- Receptores: define as configurações básicas para receptores, condições de ambiente e segurança, unidades receptoras do sinal de TV Digital terrestre e busca de canais, decodificação dos fluxos, EPG, controle de acesso, acessibilidade, canal de interatividade e atualização de software (NBR 15604, 2008);
- Codificação de dados e especificações de transmissão para radiodifusão digital: especifica a arquitetura do *middleware* para a televisão interativa do SBTVD (de acordo com a ITU Recommendation J.200, J.201 e J.202 – o Globally Executable MHP, GEM 1.1). Possui dois importantes componentes: a máquina

de execução (*execution engine*, procedural, baseado em uma máquina virtual Java conhecido como Ginga-J) e a máquina de apresentação (*presentation engine*, declarativo, conhecido como Ginga-NCL). Define ainda a pilha de protocolos, processo de apresentação e transmissão de dados (carrossel de dados, carrossel de objetos, outros métodos) (NBR 15606-1, 2008; NBR 15606-2, 2008; NBR 15606-3, 2008; NBR 15606-5, 2008);

- Canal de interatividade: especifica os protocolos, interfaces físicas e interfaces de software (NBR 15607-1, 2008).

É importante citar que na especificação Codificação de dados e especificações de transmissão para radiodifusão digital do SBTVD, a Parte 4, "Ginga-J - Ambiente para a execução de aplicações procedurais" ainda não se encontra publicada, estando até a presente data de conclusão deste trabalho em fase de projeto/elaboração.

No SBTVD, assim como no ISDB, utiliza-se um *Transport Stream* MPEG-2, e a modulação gera um sinal OFDM dividido em 13 segmentos, cada segmento ocupando 1/14 da largura do canal de televisão. Também é utilizada a transmissão hierárquica para trazer flexibilidade, recepção parcial que suporte a recepção fixa, móvel e portátil. O esquema de modulação permite o uso de DQPSK, QPSK, 16QAM e 64QAM (NBR 15601, 2008).

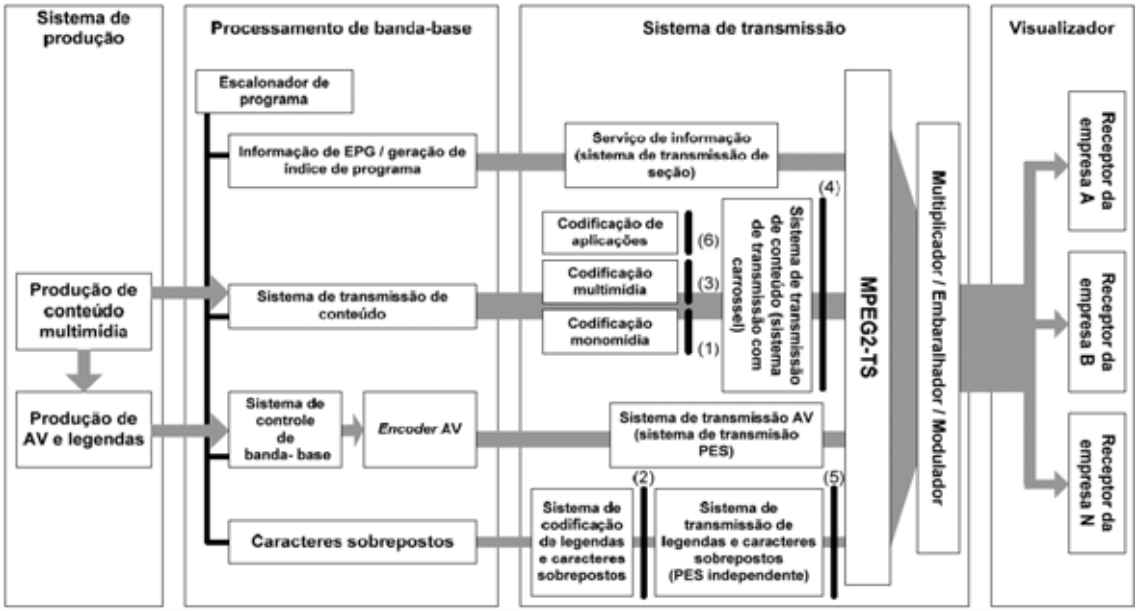

Figura 20 – Estrutura do Sistema Brasileiro de TV Digital Terrestre (NBR 15606-1, 2007)

A Figura 20 ilustra a estrutura de difusão de dados do SBTVD, em especial o fluxo de difusão, iniciando-se no sistema de produção (conteúdos), no processamento da banda-base, no sistema de transmissão (hierárquica, com *Transport Stream* MPEG-2) e no visualizador (com garantia de interoperabilidade entre os receptores dos fabricantes).

No SBTVD, vídeo, áudio e todos os serviços de dados devem ser multiplexados no *Tranport Stream* especificado pelo sistema MPEG2, que deve ser transmitido sobre uma onda de rádio. O canal de interatividade deve ser disponibilizado por uma rede independente desta pilha de protocolos.

A pilha de protocolos utilizada na difusão digital (baseada nas especificações ARIB) podem ser visualizadas na Figura 23.

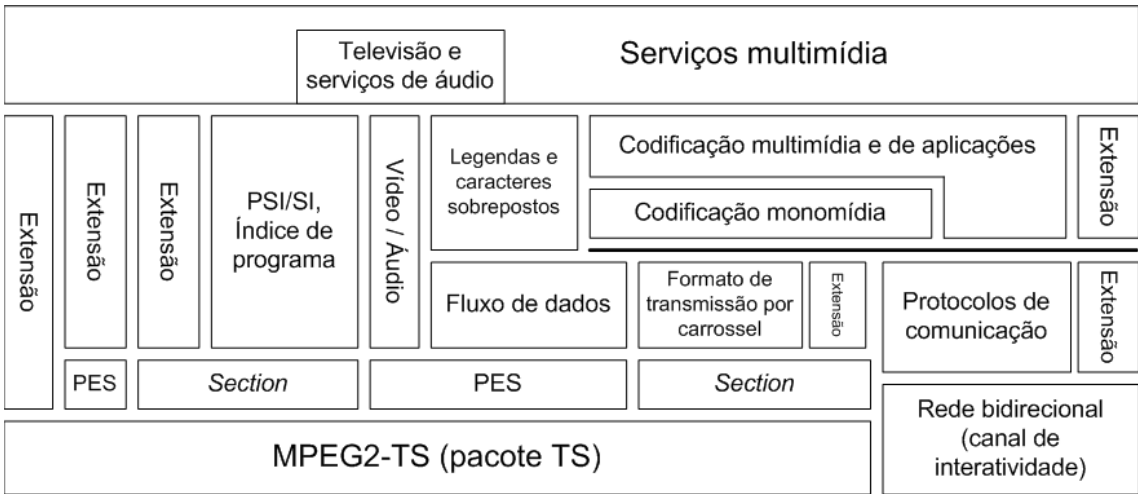

Figura 21 – Pilha de protocolos do SBTVD (NBR 15606-1, 2007)

A Função de apresentação deve garantir que os serviços multimídia sejam reproduzidos de acordo com as intenções do produtor de conteúdo, em todos os receptores. A função de apresentação deve ser designada baseando-se na representação lógica da tela da televisão, sendo esta composta por cinco camadas: camada de vídeo, camada de imagem estática, camada de seleção vídeo/imagem, camada de texto e gráficos e camada de legendas. Esta estrutura lógica de camadas pode ser visualizada na Figura 22.

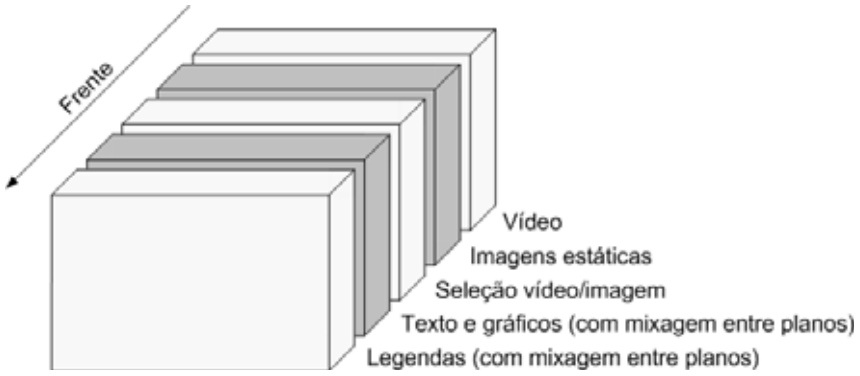

Figura 22 - Estrutura de camadas para a apresentação (NBR 15606-1, 2007)

2.4.4.1. Ginga-NCL e Ginga-J

Ginga é o nome do *middleware* atualmente adotado no sistema digital terrestre do Brasil (SBTVD). Aplicações que executarão sobre o Ginga são classificadas em duas categorias:

- Procedural, escrita em linguagem Java, executa sobre o ambiente Ginga-J (subsistema lógico do Ginga responsável pelas APIs Java). A especificação do GEM, adotada pelo padrão de *middleware* japonês (ISDB ARIB B-23) e pelo norte-americano (ATSC ACAP e OCAP) também suporta o Ginga, em sua parte procedural (Ginga-J).
- Declarativo, escrita em linguagem NCL (Nested Context Language), executa sobre o ambiente Ginga-NCL (Ginga-NCL é um subsistema lógico através do qual é possível processar e apresentar documentos NCL).

Uma aplicação declarativa é aquela onde o tipo do conteúdo da entidade inicial é declarativo. Por outro lado, uma aplicação procedural é aquela cujo tipo do conteúdo da entidade inicial é procedural. Uma aplicação declarativa pura é aquela na qual o conteúdo de todas as entidades é do tipo declarativo. Uma aplicação procedural pura é aquela na qual o conteúdo de todas as entidades é do tipo procedural. Uma aplicação híbrida é aquela cujo conjunto de entidades possui tanto conteúdo do tipo declarativo quanto procedural. Uma aplicação Ginga não necessita ser puramente declarativa ou procedural (NBR 15606-1, 2007; SOUZA FILHO *et al*, 2006).

Em particular, as aplicações declarativas freqüentemente fazem uso de scripts, cujo conteúdo é de natureza procedural. Além disso, uma aplicação declarativa pode fazer referência a um código Java TV Xlet embutido. Da mesma forma, uma aplicação procedural pode fazer referência a uma aplicação declarativa, contendo, por exemplo, conteúdo gráfico, ou pode construir e iniciar a apresentação de aplicações com conteúdo

declarativo. Portanto, ambos os tipos de aplicação Ginga podem utilizar as facilidades dos ambientes de aplicação declarativo e procedural (NBR 15606-1, 2007).

Um componente-chave do Ginga-NCL é a máquina de interpretação do conteúdo declarativo (formatador NCL). Outros módulos importantes são o exibidor (user agent) XHTML, que inclui interpretadores CSS e ECMAScript, e a máquina de apresentação Lua, que é responsável pela interpretação dos scripts Lua. Já no Ginga-J um componente-chave do ambiente de aplicação procedural é a máquina de execução do conteúdo procedural, composta por uma máquina virtual Java (NBR 15606-1, 2007).

Decodificadores de conteúdo comuns servem tanto às aplicações procedurais quanto às declarativas que necessitam decodificar e apresentar tipos comuns de conteúdo como PNG, JPEG, MPEG e outros formatos. O núcleo comum ginga (Ginga Common Core) é composto pelos decodificadores de conteúdo comuns e por procedimentos para obter conteúdos transportados em *Transport Streams* MPEG-2 e através do canal de interatividade. O núcleo comum ginga também deve obrigatoriamente suportar o modelo conceitual de exibição (SOUZA FILHO *et al*, 2006).

O ambiente de aplicações do *middleware* (ilustrado na Figura 23) deve ser composto pelos seguintes elementos arquiteturais (NBR 15606-1, 2008):

- Máquina de apresentação (NBR 15606-2, 2008), que disponibiliza para as aplicações engines e APIs XHTML, NCL e LUA-NCL;
- Máquina de execução (ambiente procedural, ou gerenciador de Xlets), definida sobre uma Java Virtual Machine (JVM);
- Ponte: mecanismo para aplicações que permite o mapeamento bidirecional entre as API Java e os objetos e métodos do DOM, ECMAScript e LUAScript;

- Monitor do ciclo de vida de aplicação: aplicação ou recurso do sistema operacional para controle do estado do software. Sua função inclui a gerência de todo o ciclo de vida da aplicação, incluindo a inicialização, término e controle. O monitor do ciclo de vida de aplicações deve estar de acordo com o ambiente procedural;
- Aplicações: podem ser escritas para a máquina de apresentação, para máquina de execução ou para ambas as máquinas;
- Outras mídias: incluem fluxos de mídia como áudio e dados ou monomídias como imagens estáticas e texto;
- Software nativo: inclui software legado ou softwares escritos usando API adicionais com funcionalidades (não especificados no SBTVD).

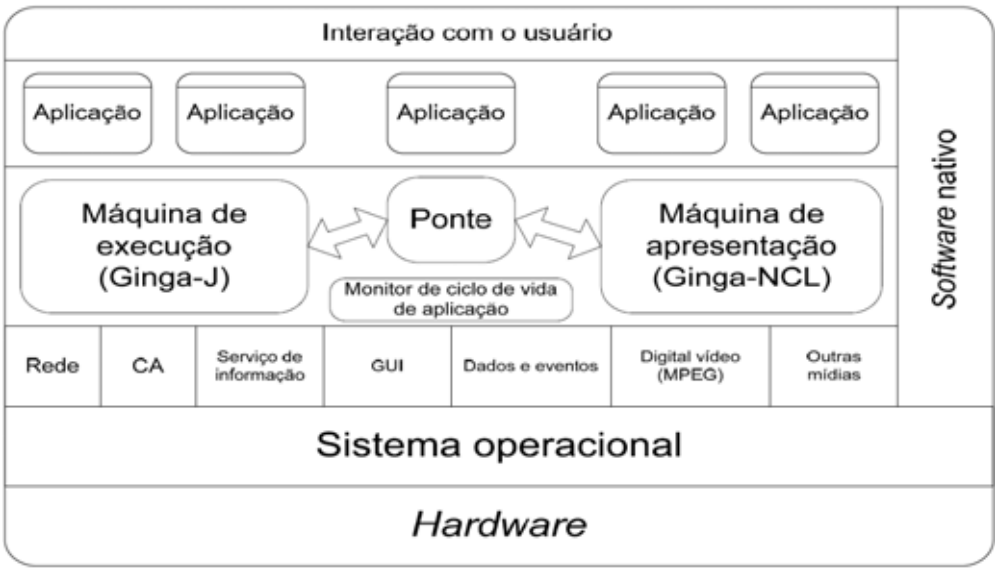

Figura 23 - Estrutura do ambiente de aplicações (NBR 15606-1, 2008)

A Figura 23 mostra a pilha de software do Ginga-J e seu ambiente de execução. Nesta figura, a pilha de software reside sobre um Host Device, que é um PC, uma Set-Top Box, um celular entre outros dispositivos. O modelo do Ginga-J prevê um hardware, recursos, software de sistema e aplicações (SOUZA FILHO *et al*, 2006).

Aplicações nativas podem ser implementadas usando as funcionalidades padronizadas fornecidas pelo Sistema Operacional, pelo *middleware* Ginga ou pelo Ginga-J. Aplicações recebidas pelo broadcast devem usar o conjunto de APIs do Ginga-J. Em geral, o Ginga atua de forma independente de aplicações nativas (SOUZA FILHO *et al*, 2006).

O Ginga-J acessa os fluxos de vídeo, áudio e dados, e fornece meios de acesso a estes fluxos pelas aplicações através de sua API.

A especificação do Ginga-J ainda prevê suporte a comunicação com dispositivos usando Bluetooth, Wi-Fi, Infravermelho, Powerline, Ethernet ou qualquer outra rede presente em uma Home Area Network. Aplicações podem fornecer suporte a interação multi-usuário, uma vez que o receptor de T pode ser acessado por diferentes dispositivos simultaneamente (SOUZA FILHO *et al*, 2006).

Alguns requisitos importantes identificados no contexto brasileiro não eram suportados pela definição internacional harmonizada de *middleware* (GEM). Com o objetivo de completar a especificação brasileira e manter a compatibilidade com a API do GEM, o Ginga desenvolveu três conjunto de APIs chamadas de Verde, Amarelo e Azul (Figura 24). Nesta figura o SBTVD é chamado de ISDTV (SOUZA FILHO *et al*, 2006).

As APIs Amarelas foram extensões propostas para o SBTVD que podem ser implementadas em outros sistemas através do uso de um adaptador de software ou usando as APIs Verde. As APIs Azuis são aquelas que não são compatíveis com as APIs do GEM. Desta forma, aplicações que fazem uso somente das APIs Verdes podem ser executados nos *middlewares* Ginga, MHP, OCAP, ACAP e B.23. Aplicações que usam as APIs Verdes e Amarelas podem ser executadas nos *middlewares* MHP, ACAP, OCAP e B.23, somente se os adaptadores de software necessário forem transmitidos e

executados juntamente com a aplicação. Aplicações que usam as APIs Azuis podem ser executadas somente nos *middlewares* Ginga (SOUZA FILHO *et al*, 2006).

As APIs Verdes são compostas pelos pacotes Sun JavaTV, DAVIC, HAVi e DVB, todos inclusos na especificação do GEM. As APIs Amarelas são compostas pela API do JMF 2.1, que é necessário para o desenvolvimento de aplicações avançadas. Uma extensão da API de apresentação do GEM, com funcionalidades para suportar as especificações de fluxo de vídeo definidas no padrão Ginga-J. Estende-se também a API do canal de retorno do GEM, que permite o envio de mensagens assíncronas. Outra extensão para a API do Service Information do ARIB B-23. As APIs Azuis são compostas pela API de Integração do Dispositivo, que permite ao receptor comunicar-se com outros dispositivos através de interfaces compatíveis, uma API Multi-usuário (que utiliza a API de Integração do Dispositivo) para permitir a interação de múltiplos usuários simultaneamente, uma API de ponte com o NCL, que permite o desenvolvimento de aplicações Java que contenham aplicações NCL (SOUZA FILHO *et al*, 2006).

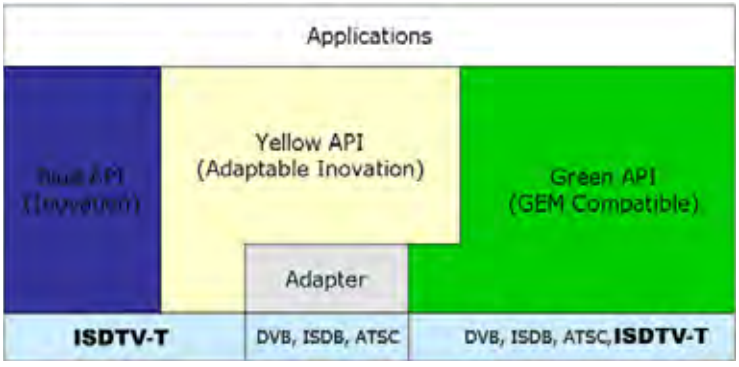

Figura 24 – APIs Azul, Amarela e Verde do Ginga-J

Do lado declarativo tem-se o Ginga- NCL. Ambientes declarativos são igualmente importantes em sistemas de TV Digital, e como exemplo tem-se o DVB-

HTML, ACAP-X e o BML (no ARIB). Convém ainda citar que no ARIB o ambiente de execução baseado no GEM ainda não foi de fato implementado (SOARES *et al*, 2007).

Uma das vantagens de um ambiente declarativo é que ele oferece maior nível de abstração do que linguagens como Java. Além disso eles oferecem uma rígida separação entre conteúdo e estrutura, formas de separação entre a mídia e os dispositivos de apresentação e acesso ao carrossel de dados e objetos (SOARES *et al*, 2007).

2.5. Aplicativos para TV Digital

Nesta seção traz uma revisão da literatura sobre os conceitos introdutórios de uma aplicação para TV Digital desenvolvida sobre a plataforma Java TV (conhecida como Xlet), como o ciclo de vida, interfaces e o diagrama de estados, e em seguida apresenta uma definição das principais classificações utilizadas para aplicações para TV Digital, porém de forma mais ampla, baseados ou não na plataforma Java TV.

2.5.1. Conceitos básicos de um Xlet

As aplicações para TV Digital baseadas na plataforma JavaTV, mais conhecidas como Xlet, possuem algumas diferenças significativas das aplicações Java tradicionais para PC. A primeira grande diferença se dá no ciclo de vida das aplicações, que em ambientes de TV Digital é controlado por uma entidade externa à aplicação, neste caso o Gerenciador de Aplicações (*Application Manager*) (MHP-INTERACTIVE, 2008). Tal diferença acontece pois a ativação e finalização de aplicações estão associada ao contexto do serviço sendo apresentado pelo receptor.

Uma forma de compreender tal cenário é avaliar os métodos que todo aplicativo deve implementar, pois os herdarão da classe Xlet:

```
public interface Xlet {

    public void initXlet(XletContext ctx) throws XletStateChangeException;

    public void startXlet() throws XletStateChangeException;

    public void pauseXlet();

    public void destroyXlet(boolean unconditional)

                throws XletStateChangeException;
}
```

Figura 25 – Interface Xlet

Uma classe que implemente um Xlet será instanciada pelo Gerenciador de Aplicações e entrará no estado LOADED. Quando a aplicação for iniciada (por intervenção do usuário, ou pelo sinal de AUTOSTART da emissora), o Gerenciador de Aplicações faz uma chamada ao método *initXlet* para que o Xlet possa preparar para ser iniciado (executar preload, instanciar variáveis, acessar recursos escassos), e envia nesta chamada um objeto *XletContext*. Neste momento a aplicação encontra-se em estado PAUSED. Em seguida o Gerenciador de Aplicações executa o método startXlet, que indica ao Xlet para iniciar sua execução, apresentar seus elementos gráficos, interação com o usuário, entre outros aspectos que uma Xlet pode conter, e também a movendo para o estado STARTED. Durante qualquer momento da execução, o Gerenciador de Aplicações pode executar o método *pauseXlet*, indicando para a aplicação mover para o estado PAUSED e liberar recursos neste período, e o método *destroyXlet*, indicando o término da execução deste Xlet (estado DESTROYED). A Figura 26 ilustra esta transição de estados de acordo com os métodos da interface do Xlet:

Figura 26 – Ciclo de vida de um Xlet

Através deste novo modelo de ciclo de vida, outra diferença significativa é que apenas uma instância da máquina virtual é utilizada para execução de todas as Xlets, com o objetivo de economia de recursos. Por este motivo, adaptações são necessárias para que uma aplicação Xlet não tenha acesso a recursos que a permitam finalizar a instância da máquina virtual e para que ela não interfira no funcionamento das demais aplicações em execução, o que é feito através do uso de ClassLoaders customizados.

Figura 27 – Ambiente de execução e interação através de um XletContext

A Figura 27 detalha o papel de um *XletContext*, que é responsável por ser a interface entre o Xlet e o ambiente de execução. Através dele um Xlet pode notificar ao Gerenciador de Aplicações que automaticamente entrou em estado PAUSED (*XletContext.notifyPaused()*). Ela também pode requisitar voltar ao estado STARTED (*XletContext.requestResume()*), e neste caso o *XletContext* notifica o Gerenciador de

Aplicações, que é responsável pela chamada do método *startXlet* do Xlet para retorná-lo ao estado solicitado.

Através de um *XletContext*, um Xlet ainda pode ter acesso a diversas propriedades do ambiente, como por exemplo o *ServiceContext*, através do método XletContext. getXletProperty ("javax.tv.xlet.service_context"). Um *ServiceContext* pode ser usado para manipular os fluxos de mídia de um serviço, para iniciar ou parar um serviço ou alterar o serviço que está sendo apresentado pelo receptor, entre outras funcionalidades.

A Figura 28 apresenta todas as interfaces citadas de um *XletContext*, e que portanto podem ser consumidas pelo Xlet.

```
public interface XletContext {
    public static final String ARGS = "javax.tv.xlet.args";

    public void notifyDestroyed();

    public void notifyPaused();

    public void resumeRequest();

    public Object getXletProperty(String key);
}
```

Figura 28 – Interface de um XletContext

2.5.2. Principais classificações de aplicativos para TV Digital

Em um ambiente de TV Digital, cada aplicativo MHP está amarrado a um serviço específico, de forma que este aplicativo não pode existe sem o serviço que o originou. Cada serviço possui uma tabela com os aplicativos associados (também conhecida como Application Information Table - AIT). Esta estrutura permite que um serviço determine qual o conjunto de aplicações que deverão ser executadas durante sua

apresentação e também garante que nenhum outro aplicativo esteja executando, permitindo maior controle sobre o funcionamento esperado destes aplicativos (MHP-INTERACTIVE, 2008).

A dinâmica da troca de serviços em um receptor de TV Digital prevê que na troca de serviço, todas as aplicações executando no serviço atual que não estiverem listadas na AIT do novo serviço serão encerradas, enquanto aplicações que estiverem listadas em ambos os serviços serão executadas sem interrupção (não serão carregadas novas instâncias ou cópias desta aplicação). As aplicações que estão listadas na AIT deste novo serviço serão carregadas (LOADED) e iniciadas (STARTED) conforme especificado na AIT (que poderá indicar que elas sejam executadas imediatamente após a leitura da AIT, que elas estejam disponíveis para que o usuário a inicie, ou ainda que aguarde um sinal da emissora para sincronização com algum outro evento) (MHP-INTERACTIVE, 2008).

Abaixo está listada um sistema de classificação dos principais tipos de aplicações interativas, independente de uso de linguagem procedural ou declarativa, proposta por (MORRIS & SMITH-CHAIGNEAU, 2005; MHP-INTERACTIVE, 2008; CRINON *et al*, 2006):

- *Service-bound applications*: são aplicações transmitidas cujo escopo está amarrado a um serviço específico. Elas são transmitidas todas as vezes que necessitarem ser executadas, podendo ser iniciadas pelo usuário ou automaticamente. Estas aplicações são encerradas pelo Gerenciador de Aplicações imediatamente após a troca de serviço;
- *Unbound* ou *External applications*: são aplicações que são transmitidas como parte de um serviço, mas não estão estritamente associadas com o serviço que a transmitiu. Elas também podem ser iniciadas pelo usuário ou

automaticamente, mas a continuidade de sua execução após a troca de serviço é condicionada à sua presença na AIT do novo serviço com o mesmo identificador. Um exemplo deste tipo de aplicações são os *news tickers*;

- *Stored applications*: são aplicações que foram transmitidas pela emissora com este indicador para que elas sejam armazenadas localmente no receptor, juntamente com sua entrada na AIT, e portanto carregadas do armazenamento local quando necessário executá-las. Isto permite um carregamento mais rápido. O receptor pode negar o armazenamento de algumas aplicações e também excluí-las quando lhe faltar espaço. Exemplos de aplicações que podem utilizar o indicador *stored* são EPG ou *news ticker*;
- *Native* ou *Resident applications*: são aplicações instaladas junto ao receptor ao invés de serem transmitidas pela emissora, ou seja, são fornecidas pelo fabricante do receptor. Exemplos de aplicações são aquelas introduzidas pelo Internet Access Profile do MHP, como Web Browser e Cliente de E-mail.
    - É importante destacar também que o termo *Native* e *Resident* não são sinônimos: enquanto *Native* refere-se ao fato de serem fornecidas pré-instaladas pelo fabricante do receptor, *Resident* refere-se ao fato de as aplicações persistirem no receptor, sem poder se removida ou alterada. Os termos tradicionalmente definem o mesmo conjunto de aplicações pois todas as aplicações *Native* são também aplicações *Resident*, ou seja, aplicações que persistem (residem) no receptor, e por outro lado, segundo as especificações atuais, para ser uma aplicação *Resident* ela deve ser *Native*;

- *System applications*: é um sub-conjunto das *native applications*, e normalmente possuem funcionalidades intrínsecas ao receptor, não associadas com o usuário;

Na classificação apresentada nesta seção considera-se que a aplicação em algum momento foi transmitida pelo *broadcast* (mesmo que depois tenha sido armazenada) ou já se encontra nativamente na plataforma, sendo este fato comum pois estas são as formas de carregamento de aplicações em uma plataforma de TV Digital (CRINON *et al*, 2006).

Um aspecto não citado até então é o acoplamento desta aplicação com um programa (CRINON *et al*, 2006), pois mesmo sendo service-bound, uma aplicação pode não possuir nenhuma sincronia ou exibir conteúdo não relacionado ao serviço atual.

Apesar de existirem aplicações consideradas *Native* ou *Resident*, as mesas são projetadas juntamente com o middleware para evitar qualquer impacto ou interferência em uma aplicação de um determinado serviço. Outro fato importante discutido por Steven Morris em (MHP-INTERACTIVE, 2008) é que não existe uma aplicação completamente *Standalone*, que possa ser iniciada e finalizada a qualquer momento sem restrições. Para suportar aplicações *Native* ou *Resident* os middlewares precisam impor tratamentos especiais e um conjunto de restrições, como por exemplo parar o serviço atual antes de ser iniciada a aplicação *Native* ou *Resident*. Por ser dependente da plataforma, não é possível ter uma regra comum às aplicações *Native* ou *Resident* executam sobre o topo da pilha de APIs de um middleware.

Aplicações do tipo *Native* ou *Resident* e *System* são tipicamente encontradas em todos os padrões de *middleware* e podem variar bastante de acordo com a implementação utilizada. Um *Service Information*, por exemplo, será implementado

como uma aplicação do tipo *Native* ou *Resident*, podendo usar a linguagem da plataforma ou pode ser uma aplicação Java (esta decisão depende de parâmetros de performance, pode exemplo a freqüência de acesso à base de dados do SI ou a latência para acessar objetos nativos em Java) (MORRIS & SMITH-CHAIGNEAU, 2005).

Porém, nem todos os tipos de aplicações são suportados em todos os padrões de middleware. No MHP aplicações do tipo *stored* (também conhecidas como *standalone*) e *unbound* são suportadas apenas na versão 1.1. Elas não são parte do MHP 1.0.x e não podem ser usadas nesta versão do sistema. Aplicações *stored* no MHP podem ser do tipo *bound*, portanto controladas pelo sistema de sinalização de aplicações no fluxo recebido, ou aplicações do tipo *unbound* que executa independente do conteúdo do broadcast. Aplicações do tipo *stored* no MHP podem ser rejeitadas pelo *middleware* e usualmente são aplicações do próprio operador da rede, como EPG ou monitor de aplicações (MORRIS & SMITH-CHAIGNEAU, 2005).

O GEM, por sua vez, não suporta aplicações do tipo *stored* e *unbound*.

CRINON *et al* (CRINON *et al*, 2006) cita a existência dos chamados "canais virtuais" no ATSC, composto por aplicações do tipo não-acoplado. É citados também a existência de aplicações do tipo *bounded* e *unbounded*. O DASE possui inclusive um protocolo conhecido como Program and System Information Protocol (PSIP), responsável pela publicação do tipo de aplicação, permitindo aos receptores do Guia de Programação Eletrônico (EPG) a informação de quando um *virtual channel* é um canal exclusivo para serviços de dados, quando é um canal de áudio e dados ou um canal de vídeo, áudio e dados.

Na parte 4 da norma NBR 15606 que trata de "Ginga-J - Ambiente para a execução de aplicações procedurais", ainda em desenvolvimento pela respectiva comissão na ABNT, discute-se a questão de aplicações residentes no seguinte trecho:

> "As aplicações residentes podem ser implementadas usando funções não padronizadas, fornecidas pelo Sistema Operacional do dispositivo de Ginga, ou por uma implementação particular do Ginga. Os aplicativos residentes também podem incorporar funcionalidades providas pelas API padronizadas Ginga-J. Aplicativos transmitidos (Xlets) sempre devem utilizar API padronizadas fornecidas pelo Ginga-J. Em geral, o Ginga é alheio a quaisquer aplicativos residentes. Estas aplicações residentes incluem, mas não se limitam a: closed caption, mensagens do sistema de acesso condicional (Conditional Access – CA), menus do receptor e guias de programação eletrônica (Eletronic Program Guide – EPG) residente. Os aplicativos residentes podem ter prioridade sobre os aplicativos Ginga. Como exemplo, o closed caption e mensagem de emergência devem ter prioridade no Sistema Ginga."

Desta forma a norma citada não define um padrão para conteúdos e aplicações interativas do tipo *native* e para características do *middleware* que as suportem. Esta afirmação citado acima é apresentada de forma semelhante por Souza Filho (SOUZA FILHO *et al*, 2006) e na especificação brasileira NBR-15606-2 (NBR 15606-2, 2008).

Na especificação brasileira NBR 15601-1 (NBR 15606-1, 2008), que trata do Ginga-NCL, apenas cita-se a possibilidade de existência de aplicações nativas, ou outros softwares específicos e de conteúdo. Nesta especificação, entende-se por "software nativo" ou "aplicações nativas" qualquer "software legado" ou "softwares escritos usando API adicionais com funcionalidades". E nesta norma "software legado"

ou “softwares escritos usando API adicionais com funcionalidades” não são especificados.

O Ginga-J, porém permite o uso de aplicações do tipo *unbound* e *stored*, definindo-as como aplicações cujo ciclo de vida não está ligado ao programa de TV, podendo inclusive ser salvas de forma persistente. Diferentemente do MHP, onde estas aplicações são usadas pelos operadores da rede, um dos possíveis usos associados a estas funcionalidades no Ginga-J é a distribuição de aplicações educacionais interativas que podem ser salvas por estudantes e professores para uso posterior (SOUZA FILHO, *et al*, 2007).

Aplicações do tipo *unbound* e *stored* abrem diversas possibilidades para novos conteúdos e formas inovadoras de interação com o usuário. Um desses exemplos é o recém-lançado Yahoo! Connected TV, ilustrado na Figura 29, Figura 30 e Figura 31. Além de *unbound* e *stored* esta aplicação é do tipo *native*. Segundo o Yahoo (YAHOO, 2008), empresas como AT&T, TiVo, Samsung, Sony, LG e Vizio estão oferecendo este aplicativo em seus Set-Top Boxes. Esta aplicação pode permanecer entre os canais e seu objetivo é oferecer uma interface adaptada para a TV que apresente conteúdo existente na web. Desta forma são definidos diversos *widgets*, cada um representando um serviço web existente (YAHOO, 2008).

Figura 29 - Yahoo! Connected TV oferecendo conteúdo jornalístico

Figura 30 - Barra inferior do Yahoo! Connected TV com aplicações do tipo *unbound* traduzindo para TV conteúdo da web

Figura 31 - Menus laterais do Yahoo! Connected TV

2.5.3. Classes importantes em um middleware baseado em MHP e JavaTV

Abaixo temos uma lista com as principais classes e interfaces de um middleware baseado em MHP e JavaTV, bem como um resumo de seu papel, métodos (JAVA TV, 2009):

- *javax.tv.xlet.XletContext*: uma interface fornecida a um Xlet no começo de sua execução, que provê uma forma de acesso às informações do seu ambientes (através do método *getXletProperty*), como acesso ao contexto do serviço;

- *javax.tv.service.selection.ServiceConext*: representa o contexto do serviço sendo apresentado pelo receptor, pode ser utilizado para seleção de novos serviços, bem como para acessar os controladores (*javax.tv.service.selection.ServiceContentHandler*) do serviço atualmente em execução (como controladores para o vídeo e para o background);
- *javax.tv.service.selection.ServiceContentHandler*: representa um mecanismo para apresentação, processamento e controle de poções de um serviço (Áudio, Vídeo, Dados, Aplicações - Xlets) ou de todo um serviço. É representado por classes que o especializam, como por exemplo o *javax.tv.service.selection.ServiceMediaHandler*;
- *org.havi.ui.HScreen*: um middleware que implemente o HAVI deverá possuir uma instância deste objeto para cada *display device* físico que estiver conectado ao receptor (tipicamente um);
- *org.havi.ui.HScreenDevice*: diversos objetos *HScreenDevice* (na verdade suas subclasses) fazem parte de um *HScreen*, representando as diversas camadas (ou superfícies gráficas) neste display, dentre elas a *org.havi.ui.HBackgroundDevice* (que exibe uma imagem estática ou preenche o plano com uma cor sólida), a *org.havi.ui.HVideoDevice* (onde o vídeo é apresentado pelo decodificador) e a *org.havi.ui.HGraphicsDevice* (usada para o gráfico das aplicações);
- *org.havi.ui.HScene*: define o container *top-level* de uma aplicação para TV Digital, semelhante à classe *Frame* presente em aplicações PC;
- *javax.media.Player*: representa um *player* de conteúdo multimídia, possuindo ou não controles associados (*javax.media.Control*) que adicionam funcionalidades para controle e manipulação do conteúdo gerenciado pelo *player*. A classe *javax.tv.service.selection.ServiceMediaHandler* é herdeira desta classe.

3. MIDDLEWARE PARA TV DIGITAL COM SUPORTE A APLICAÇÕES RESIDENTES NÃO-NATIVAS

Este capítulo descreve uma proposta de *middleware* baseado em sistemas de TV Digital existentes e especificações de *middleware* usadas mundialmente (mantendo-se inclusive compatibilidade com ambas), porém expandindo opções de interatividade em um *middleware* de TV Digital através da criação de canais virtuais. Estes canais podem ser associados ou não com o broadcast audiovisual, e permitem a execução de aplicações dissociadas ao canal de broadcast, que podem tornar-se aplicações residentes.

Apesar de previsto por diversos autores, este cenário encontra-se pouco explorado e não especificado nas principais normas de TV Digital encontradas mundialmente, incluindo as normas brasileiras. Esta proposta traz uma implementação de referência e discute as adaptações necessárias na camada de *middleware*, a compatibilidade deste canal virtual com as aplicações interativas compatíveis com as especificações de *middleware* existentes e também discute formas de distribuição padronizadas para aplicações destes canais.

3.1. Motivação: novas oportunidades para a TV Digital no Brasil

Juntamente com a alta resolução gráfica, a multiprogramação e o uso eficiente do espectro de rádio, o datacasting é um dos principais motivadores na implantação de um sistema de TV Digital. Esta forma de transmissão permite a difusão de dados em

conjunto com vídeo e áudio e, aliado com melhorias importantes na arquitetura de receptores que fazem uso de frame buffers e novas lógicas de captura para demultiplexação e parsing de fluxos de dados digitais, tornou possível a TV Interativa (WU *et al*, 2006; HORI & DEWA, 2006).

A principal motivação reside no fato de haver novas oportunidades de emissoras complementarem seus serviços de áudio e vídeo com novos tipos de interatividade e informação. Emissoras públicas enxergavam como um meio de promover sua missão de prestação de serviço público, enquanto emissoras privadas (com fins comerciais) enxergavam novas fontes de receita (CRINON *et al*, 2006).

Porém, Crinon *et al* (CRINON *et al*, 2006) relata que o sucesso da interatividade tem sido mais destacado nas emissoras públicas, pois estas captam recursos e financiamento para promover sua missão de prestação de serviço público (educação, informação, outros serviços), e fazem uso de datacasting para estas finalidades, possuindo portanto menor compromisso com o retorno financeiro do investimento, mas buscando mais reputação e em geral buscando atingir melhor sua missão. Novos tipos de serviços a levam indiretamente a conseguir mais fundos. Enquanto isso emissoras privadas possuem modelos de negócios fortemente baseados em advertising, e no serviço que prestam precisam preocupar-se com a relação entre o contratante (tradicionalmente de advertising) e o serviço que oferecem (audiência), ou seja, mostrar uma relação direta entre o serviço que oferecem e àqueles que pagam. Portanto o foco destas emissoras quando olham para serviços interativos é nos consumidores, suportadas pelo advertising. Porém elas tendem a retardar o investimento em programas de TV com uso de datacasting pois serviços interativos podem reduzir audiência (espectadores estarão consumindo os serviços), interferindo no fluxo de receita atual destas emissoras, e também enquanto não considerar haver número suficiente de

consumidores com receptores capazes de processar tais dados, e por outro lado os consumidores são relutantes em pagar por receptores enquanto não há programação suficiente com uso de datacasting.

Além das amplas possibilidades de uso do datacasting associado a um programa e suportado por advertising, Crinon *et al* (CRINON *et al*, 2006) discute também que a geração de receitas adicionais por parte das emissoras privadas com o uso de datacasting não associado à programação, onde emissoras privadas ofereceriam sua banda de transmissão para terceiros utilizarem como canal de distribuição de dados ou aplicações não relacionadas com a programação. Este modelo exige das emissoras novos modelos de negócios, novos canais de vendas e novas competências de marketing, o que pode demorar um tempo para se desenvolver.

Tais preocupações também se tornaram presente quando as emissoras passaram a iniciar projetos de portais Web, que poderiam reduzir a audiência de seu broadcast e reduzir suas receitas. Da mesma forma, espera-se que gradualmente modelos de negócios inovadores possam surgir como casos de sucesso, tornando-se referência às demais emissoras. Crinon *et al* (CRINON *et al*, 2006) ainda cita outros fatores que tendem a um aumento de projetos que utilizem datacasting, como o custo decrescente dos receptores, amadurecimento do potencial desta tecnologia por parte das emissoras e empreendedores e as aplicações direcionadas para o mercado corporativo (onde o custo tem menor impacto). No Brasil, iniciativas do Governo Federal começam a surgir para garantir público e demanda de serviços interativos, como na Portaria Interministerial 237 (PORTARIA INTERMINISTERIAL 237, 2008) que institui no parágrafo 4 que:

> "A partir de 1º- de janeiro de 2010, pelo menos, 5% (cinco por cento) da produção total de aparelhos celulares incentivados, por empresa, deverão ter capacidade de recepção de sinais de TV

> digital compatíveis com as especificações e normas do Sistema Brasileiro de TV Digital Terrestre - SBTVD, inclusive com o middleware GINGA-NCL, de acordo com norma técnica nacional (NBR) aplicável."

Zuffo (ZUFFO, 2006) expõe que o potencial para as emissoras reavaliarem e aprimorarem seus modelos de negócios está na individualização do expectador e na flexibilidade trazida pela TV Digital nos aspectos: Qualidade (DTV), Definição HDTV, Múltiplos Programas, Mobilidade (mTV), Transmissão de Dados, Interativa (iTV) e Portabilidade. Todos são componentes novos em um cenário de TV Digital, que podem guiar novas ofertas em modelos de negócios inovadores e de sucesso.

Conforme detalhado por Srivastava (SRIVASTAVA, 2002), na tentativa de alavancar estes modelos de negócios inovadores na busca de uma "killer application" interativa para a TV Digital (ou "killer service" interativo, em uma adaptação livre) que tivessem a capacidade de criar a audiência que justifique investimentos das emissoras e produtoras de conteúdo, algumas empresas norte-americanas (ACTV, Liberty Livewire, Motorola, OpenTV, e Universal Electronics) contrataram a Boyd Consulting para uma pesquisa de mercado onde buscaram entender o consumidor e sua percepção sobre TV interativa. Alguns resultados coletados foram:

- 76% gostariam que um handheld pudesse ser usado para interagir com a TV;
- 52% gostariam de informações jornalísticas, sobre tempo e esportes em uma TV interativa;
- 44% gostariam de ter um guia da TV;
- 38% gostariam de poder acessar informações sobre os bastidores dos programas;
- 37% gostariam de acessar e-mails na TV;
- 32% gostariam de jogos e quizzes;
- 47% dos usuários de TV analógica disseram que a TV interativa poderia fazê-los se atualizar;
- 57% disseram que se a TV interativa tivesse um handheld também, eles poderiam fazer a atualização para este serviço.

Os resultados, segundo Srivastava (SRIVASTAVA, 2002) são fortemente influenciados pela TV não-interativa em vigor (programas jornalísticos e de esporte) e por aplicações de internet consideradas “killer” (e-mail), porém revelam uma tendência do uso de plataformas de TV Digital com mecanismos de entrada mais sofisticados do que controle remoto, neste caso os handhelds, e no uso de receptores de TV Digital como plataforma de aplicações standalone, possivelmente não ligadas aos programas sendo recebidos.

São citadas na pesquisa de 2002 aplicações standalone como e-mails, jogos, quizzes, conteúdos (atualização / educação). Em um cenário atual possivelmente analisa-se que aplicações como redes sociais, como YouTube e Flickr, estariam entre as aplicações desejadas por 23% dos consumidores norte-americanos (PARKS ASSOCIATES, 2009). Ainda são citados por Zuffo (ZUFFO, 2006) aplicações e serviços como: acesso à Internet, serviços de utilidade pública como governo eletrônico, saúde (consultas ao SUS, por exemplo), serviços bancários (pagamento de contas, consultas a saldo), educação (pesquisas na internet e trabalhos cooperativos). A maioria destes serviços poderá ser ofertada na forma de cartões pré-pagos que eventualmente poderão ser distribuídos gratuitamente em aplicações de relevância social.

Este cenário ainda converge com os objetivos do Governo Federal ao implantar um sistema de TV Digital no Brasil, destacado entre outros como propiciar a criação de rede universal de educação à distância e propiciar a inclusão social por meio de acesso a tecnologias digitais (DECRETO 4901, 2003).

É neste aspecto da inclusão social através da inclusão digital que Zuffo (ZUFFO, 2006) considera ser a grande oportunidade de um sistema de TV Digital, uma vez que este sistema representa para o consumidor final a o estabelecimento da maior rede de faixa larga digital popular e gratuita, com capacidade unidirecional e compartilhada de

descida de informação multimídia praticamente comparável a internet II em todos os lares brasileiros (entre 20Mbits/s a 30Mbits/s ou 100 a 1000 vezes mais rápido que qualquer modem hoje utilizado).

E em se tratando de inclusão digital, segundo Souza *et al* (Souza Filho *et al*, 2007) o mercado brasileiro traz um cenário bastante peculiar com um baixo número de pessoas que possuem acesso à internet (32,1 milhões de pessoas). Já o número de pessoas que possuem um telefone celular é de cerca de 79, 5 milhões, consideravelmente maior e mais próximo (porém ainda distante) das residências brasileiras que possuem aparelhos de TV atualmente (correspondem a 91% do total de residências no Brasil). A Figura 32 ilustra este cenário:

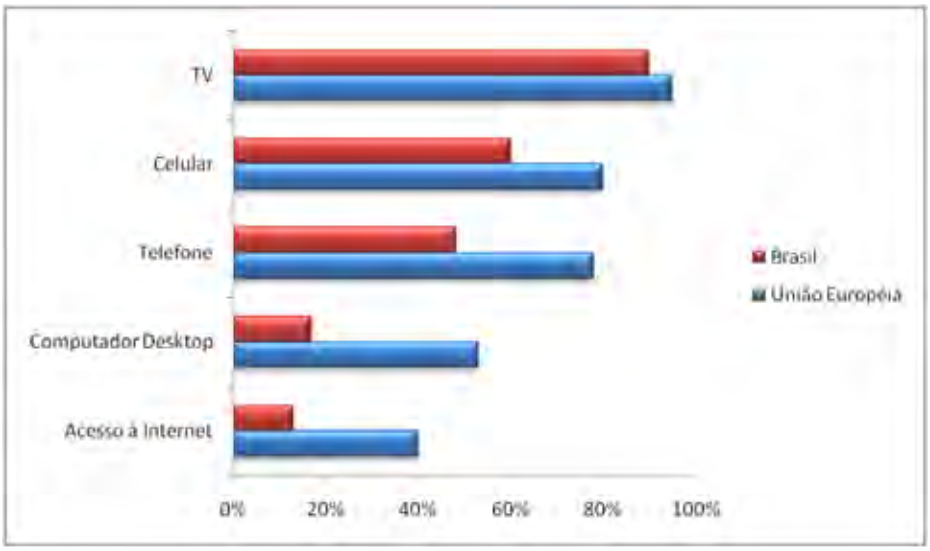

Figura 32 - Alcance de alguns serviços de comunicação no Brasil e na Europa (SOUZA FILHO *et al*, 2007)

Portanto, tendo estas considerações de mercado, no cenário brasileiro e internacional, é proposto na próxima seção um conjunto de extensões para middlewares de TV Digital, com base na possibilidade de oferta de novos serviços interativos que explorem as capacidades dos receptores, oferecendo serviços associados ou não a programas, ou em canais que podem funcionar como canais virtuais (sem conteúdo

audiovisual associado), atendendo algumas das expectativas dos consumidores apontados na pesquisa descrita em (SRIVASTAVA, 2002) e trazendo estímulos à adoção, conforme discutido por (CRINON *et al*, 2006). A proposta trazida neste trabalho também traz a possibilidade de atender os objetivos listados em (DECRETO 5820, 2006) e discutidos em (ZUFFO, 2006).

3.2. Proposta de *middleware* para TV Digital com suporte a aplicações residentes não-nativas

O presente trabalho busca avaliar a possibilidade de desenvolvimento de um middleware aderente às normais tradicionais de TV Digital e que adicionalmente possua extensões para suporte a aplicações residentes, não-nativas e instaláveis (ou seja, aplicações que atuem como residentes e que sejam adicionadas, removidas e executadas por comandos efetuados pelo usuário, sem ter sido pré-instalada pelo fabricante do receptor) e compatíveis com as APIs oferecidas às aplicações tradicionais de TV Digital.

Este trabalho provou ser possível tal desenvolvimento conforme as características e termos citados, e sendo assim será executada avaliação através de Estudo de Caso, conforme descrito na seção 3.3 (que detalha o método que será utilizado para desenvolvimento e validação desta proposta apresentada).

A primeira parte desta proposta defende que o funcionamento tradicional de um middleware, bem como as aplicações tradicionais, não sejam afetados pelas extensões propostas. Em outras palavras, o funcionamento de funções comuns como o sintonizador, o seletor de serviços, ciclo de vida das aplicações, gerenciador de aplicações, superfícies gráficas, controle de mídia, entre outras características de um

middleware, não sejam afetadas e que as aplicações já existentes continuem sendo compatíveis com o middleware.

O segundo ponto é a definição de aplicações residentes e não-nativas. Conforme já citado anteriormente, este é um tipo de aplicação não suportada nas especificações de TV Digital, e portanto não suportada por middlewares de TV Digital. O objetivo ao suportar este novo tipo de aplicação é potencializar as utilidades de um receptor de TV Digital, oferecer novas fontes de aplicações e conteúdos interativos, e expandindo a capacidade em atender as oportunidades discutidas na seção 3.1.

O suporte a aplicações residentes e não-nativas permite a execução de aplicações que sejam independentes dos serviços recebidos pelo receptor, e também independentes do conjunto de aplicações nativas, podendo assim ser instaláveis. Esta nova característica não é discutida em detalhes neste trabalho, que busca tratar de forma transparente a origem da aplicação (seja através de dispositivo de armazenamento em massa, rede IP, internet, entre outros), mas seu objetivo é criar mecanismos padronizados para, através da intervenção direta de um usuário (telespectador), adicionar, remover e atualizar uma aplicação residente e não-nativa de um receptor de TV Digital.

A última característica citada é a capacidade de execução sobre as APIs existentes para as aplicações tradicionais para TV Digital. Tal característica tem como objetivo o reaproveitamento de funcionalidades existentes em receptores de TV Digital (e todo o processo de avaliação, teste, especificação e normatização já executado em diversos países), a aderência às características destes receptores (como uso de controle remoto, gerenciamento de recursos escassos, entre outros), o reaproveitamento de ferramentas de autoria de conteúdo e de aplicações, bem como de bibliotecas existentes, e permitir maior portabilidade entre aplicações tradicionais e aplicações residentes e

não-nativas, assumindo um conjunto de recursos de hardware tradicionalmente encontrados em receptores de TV Digital.

Como conseqüência às características definidas anteriormente, temos que as aplicações suportadas por esta extensão são consideradas aplicações não-acopladas a um programa. Outras características importantes para as extensões propostas são listadas abaixo:

- Permitir que aplicações acessem dispositivos de armazenamento externo;
- Oferecer às aplicações mecanismos de acesso a protocolos de rede;
- Possuir um mecanismo de atualização de aplicações residentes;
- Ofereça novas opções para *deployment* de aplicações residentes;
- Oferecer mecanismos de execução segura, oferecendo às aplicações acesso somente as APIs previamente definidas;
- Ofereça às aplicações acesso a um ou mais canais de retorno;
- Ofereça às aplicações acesso ao carrossel de dados, ao carrossel de objetos e demais informações dos serviços.

Destaca-se como vantagens no uso desta extensão proposta a flexibilidade e independência de emissoras ou dos operadores da rede para a transmissão do arquivo ou serviço interativo, que podem ainda ser adicionadas e removidas do receptor de TV Digital quando forem necessárias. Além disso, elas podem ser armazenadas e acessadas pelo usuário sempre que necessário.

Cenários que ilustram o potencial destas aplicações estão na apresentação de conteúdo em sala de aula, onde a aplicação residente pode ser o conteúdo em si (empacotado) ou um visualizador de conteúdos trazidos por um professor para a sala de aula. Neste caso, um receptor de TV Digital ligado a uma TV pode oferecer conteúdos e formas de interação bastante diferentes daqueles tradicionalmente usados em

microcomputadores, além de eventualmente ser mais acessíveis para uso neste cenário (um dispositivo por sala de aula).

Não foi encontrado na literatura outras propostas que suportem aplicações de forma semelhante à proposta, que traz desafios adicionais ao desenho do middleware e suas interfaces, em permitir que aplicações residentes não-nativas e aplicações tradicionais utilizem o mesmo middleware e o mesmo conjunto de APIs consistentemente e harmoniosamente.

3.3. Método utilizado

O método utilizado para desenvolvimento, coleta e avaliação dos resultados deste trabalho foi um estudo de caso sobre a proposta de middleware na seção anterior. O objetivo do estudo de caso é verificar a teoria proposta, que defende que é possível realizar adaptações em um middleware de TV Digital de forma que seja possível o suporte a esta nova classe de aplicações, denominadas aplicações residentes, sem perder conformidade com as especificações que normatizam as implementações padrão de middlewares de TV Digital.

O estudo de caso utilizou uma implementação de referência de um middleware de TV Digital, bem como uma aplicação que consuma uma grande variedade de APIs deste middleware, como por exemplo as APIs listadas abaixo citadas em (MHP-INTERACTIVE, 2008):

- Superfícies Gráficas (*display devices*);
- Elementos da *Graphical User Interface* (GUI), dentre eles o conjunto de widgets e componentes *top-level* (containers);
- Apresentação de Texto;
- Legendas (*subtitles*);

- Entrada de dados do usuário;
- Controle de mídia audiovisual e referênciamento de conteúdo;
- Acesso às superfícies gráficas;
- Sincronização de conteúdo;
- Contexto do serviço e seleção de serviço;
- Acesso a arquivos;
- Filtros de seção;
- Sintonização (*tuning*);
- Gerenciamento de recursos escassos;
- Manipulação de aplicações;
- Canal de retorno;
- Comunicação Inter-Xlet;
- Acesso Condicional.

O middleware e a aplicação de estudo representam uma parte do contexto sobre o qual será aplicado o estudo. Para completar o contexto serão realizadas as adaptações necessárias no middleware para implementação do suporte a aplicações residentes. Sendo assim, compõe o contexto de estudo:

- O middleware com as adaptações iniciais para suporte a aplicações residentes
- A aplicação de estudo executando como uma aplicação residente neste middleware
- A aplicação de estudo executando em um canal no modelo tradicional de TV Digital.

As adaptações inicias no middleware se concentram na criação de um novo serviço, representado como o último serviço (em ordem numérica crescente) disponível na rede acessada pelo middleware, podendo a rede ser emulada. A criação de um novo serviço, exclusivo para aplicações residentes, prevê inicialmente apenas as adaptações abaixo:

- Alteração no seletor de serviço (acessado através de API ou do controle remoto) indicando que a tabela de serviços possui um serviço adicional;
- Um aplicação capaz de exibir um menu de aplicações residentes disponíveis, bem como de iniciá-las (executá-las);
- Uma nova layer, dentre as superfícies gráficas disponíveis no middleware, para receber o menu que será exibido no serviço de aplicações residentes. Esta layer tem como objetivo não interferir na apresentação de aplicações residentes que estejam usando a layer gráfica (usada pelas aplicações);
- Emulação das tabelas com informações de aplicação dos serviços para que, quando o serviço corrente for o serviço de aplicações residentes, a tabela apresente informações referentes às aplicações residentes disponíveis.

Esta implementação deverá ter foco em manter a compatibilidade com aplicações e serviços tradicionais de TV Digital. Uma das técnicas importantes neste processo é realizar um tratamento redundante serviços tradicionais e as novas aplicações residentes não-nativas, para que não haja um sub-conjunto de APIs que leve à perca de interoperabilidade.

A coleta de dados se dará determinando quais, dentre as APIs utilizadas por esta aplicação, são conflitantes ou incompatíveis com um middleware que suporte concomitantemente o modelo de funcionamento do canal de aplicações residentes e com

o modelo de funcionamento dos canais tradicionais. Este levantamento se dará com execuções consecutivas do middleware com as adaptações e da aplicação, testes de caixa-preta e depuração de código. Também será realizada a análise das APIs disponibilizadas pelo middleware, através de informações sobre desenvolvimento de aplicações para TV Digital coletadas após a revisão da literatura e contidas em artigos científicos, documentos técnicos, revistas especializadas, portais e páginas da internet.

Após a coleta destes dados, será realizada uma análise de código-fonte do middleware e da aplicação, com o objetivo de identificar as características significantes de eventos identificados. Tal análise levará a identificar detalhes dos comportamentos inadequados nesta aplicação em um dos serviços ou em ambos, para então poder discutir tais comportamentos e as respectivas soluções que sustentem a teoria proposta.

Tais conflitos serão destacados e posteriormente, conforme discussão sobre as incompatibilidades encontradas, serão propostas e implementadas soluções que permitam o suporte à aplicação estudadas nos serviços tradicionais e no serviço de aplicações residentes, sem que seja necessário alterações no código-fonte desta aplicação.

Estas soluções implementadas compõe o conjunto final de adaptações necessárias ao middleware. Cada solução implementada possui foco em uma ou mais funcionalidades que necessitarem de adaptação, sendo que as mesmas serão implementadas seqüencialmente, porém de forma iterativa alternando com coleta de dados de compatibilidade que foquem apenas na funcionalidade previamente adaptada com a solução implementada.

3.4. Desenvolvimento do Estudo de Caso

Os itens a seguir detalham o desenvolvimento deste projeto. Primeiramente são apresentadas as considerações de arquitetura sobre as implementações que serão realizadas. Em seguida, é discutido um cenário para este estudo de caso, apresentando as opções disponíveis e as escolhas realizadas.

Por ser uma implementação voltada para desenvolvedores validarem suas aplicações MHP, o OpenMHP não possui alguma funcionalidades padrão de um middleware, como um sistema de canais, e também não oferece por completo as tabelas de serviços ou aplicações. Sendo assim, nesta etapa foi realizada a implementação no OpenMHP de algumas funcionalidades como suporte a múltiplos serviços.

Em seguida são detalhadas as etapas de desenvolvimento para suporte a aplicações residentes, os dados coletados neste estudo de caso, com as respectivas discussões e propostas de solução e os testes de compatibilidade.

3.4.1. Considerações de arquitetura do middleware

A implementação desta proposta começa por desafios arquiteturais no middleware. O primeiro grande desafio é definir em qual nível e bem como qual estrutura lógica deverá controlar as aplicações residentes não-nativas, o que define qual API é responsável diretamente pela manipulação destes aplicativos e portanto indicando o ponto central do esforço de adaptação.

O caminho aparentemente mais lógico é utilizar a API de gerenciamento de aplicações para controlar as aplicações residentes não-nativas, porém a concepção de uso desta API e o contexto onde ela atua pode dificultar a inserção de aplicações residentes não-nativas diretamente neste nível. Tradicionalmente, as emissoras e operadores da rede criam um conjunto de elementos de um serviços e os transmitem com garantia do comportamento esperado, pois o ambiente de execução irá conter

apenas estes elementos do serviço (aplicações, dados e conteúdo audiovisual) e eles serão certamente interoperáveis entre si. Devido a este cenário, se a API de gerenciamento de aplicações atuar neste contexto iniciando e encerrando aplicações residentes não-nativas concomitantemente com os elementos do serviço atual, a quantidade de situações imprevistas e alterações que estas aplicações poderão realizar no contexto irão aumentar consideravelmente, reduzindo o controle do ambiente para o transmissor do serviço, e conseqüentemente a capacidade de os elementos do serviço serem interoperáveis. Esta abordagem levaria a incompatibilidades e comportamentos inesperados no serviço que estiver sendo apresentado, e potencialmente a aplicações menos confiáveis.

No modelo tradicional definido pela API de seleção de serviço, as aplicações estão associadas com um serviço, mesmo que ela seja o único elemento presente neste serviço (ou seja, mesmo que não existe nenhum stream de áudio ou vídeo). No modelo tradicional temos também que os serviços sempre estão presentes em um contexto (ServiceContext). Desta forma o caminho mais lógico para controlar aplicações residentes não-nativas é através deste mesmo modelo (com as devidas extensões e adaptações), através da API de seleção de serviço.

Com o uso da API de seleção de serviço é possível retomar o controle do ambiente garantindo o isolamento necessário, uma vez que o que acontece dentro de um serviço é independente dos demais, e garantindo a alocação dos recursos escassos de um receptor. Nesta abordagem, cada aplicação residente não-nativa é tratada como um serviço adicional que contém apenas uma aplicação *bounded*, tendo como contexto um conjunto mais restrito de interfaces disponíveis, uma vez que em um serviço de uma aplicação residente não-nativa não haveria elementos como o conteúdo audiovisual

apresentado pelo decoder (apesar de ser possível haver conteúdo audiovisual apresentado pela própria aplicação na *layer* gráfica).

Com esta abordagem, iniciar uma aplicação residente não-nativa corresponderá a realizar a troca de serviços, ou seja, interromper o fluxo de conteúdo audiovisual do *broadcast*, encerrar as aplicações do serviço atual, e iniciar as aplicações do novo serviço (neste caso a aplicação residente não-nativa), uma vez que o novo serviço não possuirá conteúdo audiovisual. Com esta abordagem é possível também referenciar aplicações residentes não-nativas, de forma que elas possam ser iniciadas aplicações (tradicionais ou não) apenas selecionando o serviço correspondente ou as encerrando parando o serviço.

Também se destaca que o modelo de suporte a aplicações residentes não-nativas baseado em serviços pode ser útil mesmo para aplicações nativas residentes, podendo este trabalho ser usado como referência em futuras implementações de middlewares tradicionais. Neste caso é possível garantir o isolamento apropriado, a alocação de recursos, e a manutenção da interoperabilidade e previsibilidade no comportamento dos serviços do *broadcast*.

3.4.2. Seleção de uma implementação de referência para middleware de TV Digital

Diante dos objetivos já traçados, a seleção do caso depende da escolha de uma implementação de referência de um middleware de TV Digital, de uma ou mais aplicações e do conjunto de adaptações necessárias. Tal escolha requer um cuidado muito grande, pois não se trata de uma mera escolha visual ou preceptiva, devendo estar apoiada na proposta deste trabalho.

Sendo assim, a escolha de um middleware iniciou-se com o levantamento das opções disponíveis. Em um primeiro momento foram selecionados middlewares que ofereciam suporte a aplicações procedurais baseadas na linguagem Java, especificamente na plataforma JavaTV ou GEM, uma vez que estas plataformas podem oferecer ao estudo de caso uma opção mais abrangente, devido à disponibilidade de códigos-fonte, ferramentas e exemplos, permitindo uma visão mais ampla daquilo que se quer estudar. Middlewares com suporte a aplicações declarativas (DVB-HTML e ARIB-BML) possuem menor adoção e até o momento nenhuma implementação de referência open source conhecida.

Sendo assim, após revisão da literatura, as opções encontradas e avaliadas foram as seguintes:

- Sun Java TV Reference Implementation 1.1.1;
- XleTView 0.3.6;
- OpenMHP 1.0.4;
- JTVOS 1.0.1;
- MHP4Free;
- Espial DVB-MHP-RefImpl.

Como primeiro critério, não foram consideradas as implementações proprietárias, devido ao alto custo de licenciamento e o modelo de licenciamento em lote. Dentre as implementações proprietárias encontradas estão: Osmosys MHP4Win, IRT-RI, Alticast AltiCaptor, NDS MediaHighway e OpenTV

Sendo assim, as implementações citadas na lista são implementações cujo código-fonte pode ser avaliado sem custo de licença.

O segundo critério foi baseado na quantidade de documentação disponível. Um fator contribuinte para este critério foi considerar quais implementações foram descontinuadas. Neste caso, encontrou-se que todas as implementações restantes foram descontinuadas e não possuíam atualização recente (em tempo menor que 1 ano), e em outros casos o website oficial do grupo mantenedor também não se encontra mais disponível.

Como terceiro critério, considera-se a abrangência da implementação, tendo como preferência aquelas que suportem um maior conjunto de APIs (HAVi, DAVIC, JavaTV, DVB-MHP, entre outras). E por fim, considera-se como quarto critério a compatibilidade com um ambiente real e aderência às especificações das APIs implementadas no comportamento e nas suas interfaces.

Seguido este conjunto de critérios, foram inicialmente desconsideradas (em ordem):

- MHP4Free: descontinuada, website não acessível, documentação em Alemão;
- Espial DVB-MHP-RefImpl: implementação bastante parcial (APIs JavaTV e HAVi, mas nenhuma API MHP), descontinuada, website não acessível;
- JTVOS 1.0.1: documentação escassa, parte em italiano;
- Sun Java TV Reference Implementation 1.1.1: está presente nas demais implementações citadas (diretamente ou com uma implementação equivalente). Além disso, a Sun Java TV Reference Implementation 1.1.1 não permite que seu código seja alterado sem autorização da Sun Microsystems, o que impede a execução deste trabalho;

Desta forma, duas implementações se destacaram: XleTView 0.3.6 e OpenMHP 1.0.4. Ambas possuem boa documentação e um volume semelhante de funcionalidades. Porém a principal diferença está no fato de o XleTView ter foco em ser um emulador para execução de Xlets MHP em PCs, enquanto o OpenMHP busca ser uma implementação de referência aberta desenvolvida em PC para middlewares MHP, o que leva o comportamento do OpenMHP a ser mais parecido com o comportamento real de Set-Top-Boxes. Além disso, o XleTView reimplementa todas as classes JavaTV com um nome de pacote próprio iniciado por xjavax (a API JavaTV desenvolvida pela Sun Microsystems não é utilizada), o que leva as aplicações a apresentarem comportamentos diferentes da JavaTV e também obriga efetuar modificações nas aplicações (para importar classes do pacote xjavax e não javax), podendo causar problemas de compatibilidade.

Desta forma, a implementação de referência escolhida foi a OpenMHP 1.0.4, por ser open source, possuir documentação disponível (em língua inglesa, além de alguma documentação técnica e papers em português), por possuir grande abrangência (implementando – parcialmente – as APIs HAVi, DAVIC, DVB-MHP, e utilizando a própria implementação de referência da Sun da API JavaTV) e maior compatibilidade com implementações reais do MHP.

É importante destacar que o OpenMHP, assim como as demais implementações avaliadas, são implementações incompletas de um middleware executando em um PC, usadas como referência para uma implementação real executando em um receptor, o que significa que algumas funcionalidades de um receptor não estão disponíveis, em especial aquelas relacionadas a sintonização de canais, APIs referente à decodificação MPEG, entre outras. Desta forma, não é garantido que as aplicações e o middleware tenham comportamento idêntico a receptores reais, porém considera-se esta

implementação de referência bastante completa, oferecendo um conjunto grande de APIs, dentre elas Classloader, Serviço, Carrosséis, Gerenciadores de Aplicações, HAVi, DAVIC, DVB-MHP, JavaTV, entre outras e também APIs de sistema que tratam de runtime, camada de adaptação e configurações específicas do ambiente PC, possuindo portanto uma abrangência suficiente para o desenvolvimento do estudo de caso.

3.4.3. Seleção de uma aplicação para TV Digital

Tendo os objetivos traçados e a implementação de referência do middleware definida (OpenMHP 1.0.4), a próxima etapa para definir o cenário do caso trata da escolha de uma aplicação que poderá ser tratada como uma aplicação tradicional e uma aplicação residente não-nativa. Deforma semelhante à escolha da implementação de referência do middleware, a escolha da aplicação não se trata de uma mera escolha visual ou preceptiva. Para que fosse possível realizar uma escolha que permitisse o desenvolvimento do estudo de caso, focalizando nas extensões para aplicações residentes não-nativas, foram adotados os seguintes critérios:

- Disponibilidade do código-fonte: é necessário que se tenha acesso ao código-fonte, bem como seja possível alterá-lo, para permitir que sejam feitas modificações necessárias, testes, e também depuração de código para diagnóstico de problemas e coleta de dados durante o estudo de caso;
- Compatibilidade: as aplicações selecionadas deverão ser compatíveis com o conjunto de funcionalidades oferecidas pelo middleware;
- Abrangência: para se conseguir uma visão mais ampla daquilo que se quer estudar busca-se aplicações que utilizem o maior número de funcionalidades disponíveis no middleware;
- Compreensão do código-fonte ou disponibilidade de documentação.

Após uma busca em literaturas, identificou-se existe baixa disponibilidade de aplicações que atendesse aos critérios descritos anteriormente nesta seção, em especial a abrangência, uma vez que as aplicações encontradas que atendessem os demais critérios muitas vezes buscavam expor o funcionamento de uma funcionalidade, ou seja, eram apenas amostras de alguma funcionalidade que uma aplicação MHP pode utilizar.

Diante deste cenário, foi selecionada a aplicação Quizlet, uma aplicação open source desenvolvida pelo mesmo grupo que mantém o OpenMHP, e portanto garantindo total compatibilidade e disponibilidade de código-fonte.

Quizlet é uma aplicação (mais especificamente, é um jogo) quiz que apresenta ao usuário uma seqüência de questões selecionadas randomicamente, recebe as respostas e calcula uma pontuação para cada resposta correta, exibindo o resultado ao final da seqüência de questões. Com este escopo, o Quizlet explora um conjunto de funções básicas úteis em quase todos os tipos de aplicação.

Ela é composta por quatro partes, sendo cada uma representada por uma classe:

- WelcomePage: Tela de abertura (Welcome screen);
- QuestionPage: Tela de ajuda (Help screen) com instruções de uso;
- ResultPage: Tela onde as questões são apresentadas
- HelpPage: Tela para apresentação dos resultados;

A aplicação é implementada como uma máquina de estados, sendo cada tela um estado, e um único *listener* na classe principal responsável pelo tratamento da entrada de dados do usuário de acordo com o estado atual.

A aplicação concentra o processamento principal no início e fim do processo, garantindo a responsividade durante a interação com o usuário.

A aplicação Quizlet realiza modificações no background (acesso às superfícies gráficas), interrompe a execução do conteúdo audiovisual (controle de mídia),

interrompe a exibição de legendas (acesso ao contexto do serviço) e inicia a apresentação de suas telas cujo screenshot é apresentado abaixo:

Figura 33 – Tela de abertura da aplicação Quizlet

Figura 34 – Tela de questões da aplicação Quizlet

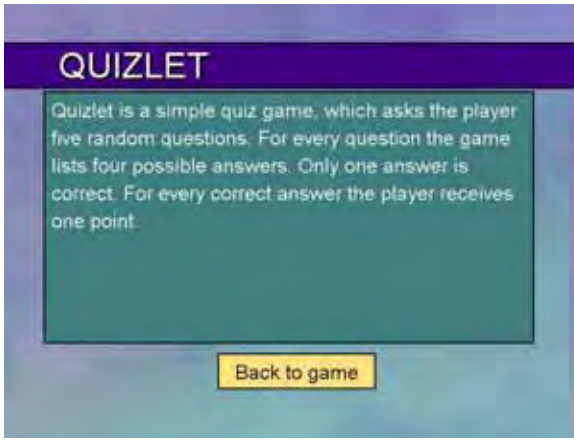

Figura 35 – Tela de ajuda

Figura 36 – Tela de resultados

Outras classes são implementadas na aplicação Quizlet para manipulação de textos, apresentação de menus, manipulação da imagem de background, acesso ao arquivo de questões e seleção randômica, e manipulação da camada de apresentação de vídeo do dispositivo MHP, ocultando-a.

3.4.4. Desenvolvimento de um emulador de serviços no OpenMHP

Uma das funcionalidades parcialmente implementadas no OpenMHP são as funcionalidades relacionadas ao *Service Information*. O status de implementação oferece suporte às classes de contexto (exceto para troca de serviços), à manipulação de localizadores de conteúdo, acesso a *handlers* (controles de conteúdos do contexto), e outros recursos baseados no pacote javax.tv.service.selection. Porém no OpenMHP foram implementadas apenas pequenas porções das classes definidas no pacote org.dvb.si, do qual fazem parte classes como SIDatabase e SIService responsáveis pelo preenchimento, recuperação e manipulação de informações do banco de dados de serviços (através da classe org.dvb.si.SIDatabase).

Desta forma, observa-se que o OpenMHP apresenta funcionalidades restritas para manipulação de serviços. Dentre as principais restrições podemos destacar a ausência de implementação para o método de seleção de serviço da classe *ServiceContext*, o que impede que novos serviços sejam iniciados ou que haja troca de serviço dentro de um contexto.

Por outro lado, o OpenMHP possui mais funcionalidades do que um gerenciador de aplicações. Na verdade, ele possui uma classe responsável pela emulação dos dados de um *Service Information*, e a utiliza como parâmetro para ajustar previamente qual serviço será exibido após a inicialização do middleware, e não oferecendo funcionalidades para que o serviço seja trocado durante sua execução. A classe citada é a org.openmhp.si.SIDataEmulation, responsável por recuperar a lista de serviços, simulando uma consulta às tabelas de *Service Information* da rede do receptor. Porém, como no ambiente emulado não existe uma rede de TV Digital disponível, esta classe recupera estas informações de um arquivo-texto, chamado services.txt, que possui os dados em formato de uma tabela e o qual apresentamos abaixo:

```
# finland
original_network_id =0x20F6

#network_n = network number, region, network_name_descriptor, id (hex)
network_1 = 2, Turku, Digita_Turku, 0x3302

#service_n  = logical channel num, transportid (hex), serviceid (hex), si type, name, provider name
service_1=  1,   0x1001,   0x0011,  1,   YLE TV 1,          YLE
service_2=  2,   0x1001,   0x0021,  1 ,  YLE TV 2 ,              YLE
service_3=  3,   0x2001,   0x0031,  1 ,  MTV3 D ,           MTV Oy
service_4=  4,   0x3001,   0x0041,  1 ,  Nelonen ,          Ruutunelonen
service_5=  5,   0x1001,   0x0051,  1 ,  YLE FST ,          YLE
service_6=  6,   0x2001,   0x0061,  1 ,  Subtv ,            Citytv Oy
service_7=  10,  0x2001,   0x00A1,  1 ,  Urheilukanava,     Suomen Urheilutelevisio
service_8=  12,  0x1001,   0x00C1,  1 ,  YLE24 ,            YLE
service_9=  13,  0x1001,   0x00D1,  1 ,  YLE Teema ,             YLE
service_10= 16,  0x3001,   0x0111,  1 ,  Nelonen Plus,      Ruutunelonen
service_11= 22,  0x3001,   0x0161,  1 ,  Estradi,           Various (Administrated by Digita Oy)
service_12= 26,  0x3001,   0x01A1,  1 ,  CANAL+,                 CANAL+
service_13= 27,  0x3001,   0x01B1,  1 ,  CANAL+ KULTA ,     CANAL+
service_14= 28,  0x3001,   0x01C1,  1 ,  CANAL+ SININEN,    CANAL+
service_15= 29,  0x3001,   0x01D1,  1 ,  CANAL+ Infokanava, CANAL+
```

Listagem 1 – Tabela de Serviços

Uma vez que o OpenMHP suporta parcialmente as APIs de manipulação de serviços, foram necessárias extensões no middleware para que funcionalidades de seleção de serviço, acesso ao banco de dados de serviços, navegação entre serviços por controle remoto e gerenciamento do contexto de troca de serviço pudessem ser utilizadas. Para o presente trabalho tais funcionalidades são fundamentais, uma vez que a estratégia para suporte a aplicações residentes não-nativas considera cada aplicação residente não-nativa como um serviço independente, tornando indispensável a realização de testes referentes a carregamento e descarregamento de serviços, execução e encerramento de aplicações, troca de serviços dentro de um contexto, bem como localização e seleção de novos serviços.

O desenvolvimento desta extensão iniciou-se com uma revisão das classes pertencentes aos pacotes responsáveis pela manipulação de serviços (dentre eles javax.tv.service, javax.tv.service.selection, org.dvb.si, org.openmhp.service e org.openmhp.si) com o objetivo de planejar esta extensão e garantir efeitos inesperados não seriam adicionados nas demais classes do middleware.

A primeira parte desta extensão teve foco em permitir a seleção de um novo serviço, garantindo o encerramento do anterior, a troca de serviço no contexto e a inicialização do próximo serviço. O método responsável por tais funcionalidades é o método select da interface javax.tv.service.selection.ServiceContext. É importante destacar que a classe ServiceContext pode ser recuperada por uma aplicação como uma das propriedades retornadas pelo XletContext.

Uma abordagem bastante utilizada na API JavaTV é o uso de Factories para garantir a coerência nas classes de sistema. Isto acontece com a ServiceContext, pois a classe responsável por instanciá-la é a classe ServiceContextFactory, garantindo que não haja múltiplas instâncias para um mesmo contexto ou objetos ServiceContext que não correspondam a um contexto.

Outro artifício utilizado são interfaces (como a ServiceContext) e classes abstratas (como a ServiceContextFactory) para classes que são dependentes de plataforma. Ao utilizar deste artifício o middleware garante a interface comum, permite que as aplicações as utilizem com garantia do comportamento esperado, mas sem poder instanciá-las diretamente. Na prática, existem outras classes que estendem tais classes abstratas e implementam tais interfaces, mas cujo nome não é conhecido pelos desenvolvedores de aplicações. E como elas utilizam os recursos de herança e polimorfismo da plataforma Java, elas é que são na prática enviadas às aplicações, na forma das classes que elas herdaram ou implementaram (conforme esperado pelas aplicações).

Considerando portanto estes dois artifícios, encontra-se no OpenMHP dois pacotes que contém implementações dos Factories, das classes abstratas e das interfaces dos dois pacotes citados acima. Estas classes encontram-se nos pacotes org.openmhp.service e org.openmhp.si, e não podem ser instanciadas diretamente pelas

aplicações. Seguindo esta organização, a implementação realizada do método select foi realizado na classe org.openmhp.service.ServiceContextImpl, uma classe herdeira de javax.tv.service.selection.ServiceContext e a classe que de fato é acessada pelas aplicações.

A interface definida para o método select prevê um parâmetro do tipo javax.tv.service.Service. O comportamento esperado é que o novo serviço substitua o serviço sendo atualmente apresentado, preservando algum componente que esteja presente nos dois serviços.

A implementação realizada no método citado executa as seguintes instruções:

1. Verifica a validade do serviço recebido, procurando-o na tabela de serviços disponibilizada pelo SIDataEmulator;
2. Sendo válido, o serviço atual é interrompido e seu estado padrão é reposto com as seguintes etapas:
    a. Todas as aplicações não presentes no novo serviço são interrompidas;
    b. Todos os ServiceContentHandler (ex: controle de mídia, legendas) são interrompidos;
    c. Todos os elementos adicionados no background são removidos (ex: imagens ou cor fixa);
    d. Todas as layers são ajustadas para se tornarem visíveis (a aplicação pode ter ocultado alguma layer);
    e. O background é ajustado para uma imagem padrão do middleware (tradicionalmente o logotipo do fabricante, mas neste caso é uma imagem com uma seqüência vertical de padrões de cores).
3. Em seguida o novo serviço é carregado com as seguintes etapas:
    a. Inicialização de todos os ServiceContentHandler;

b. Inicialização das aplicações que fazem parte deste serviço;
c. O nome e o ID do novo serviço é exibido na Extra Layer, a superfície gráfica definida pelo OpenMHP sobre todas as demais.

Tendo implementado o método de seleção de serviços, foi necessário implementar uma nova classe que soubesse como navegar entre os serviços, acessando o SIDataEmulation. Foi então implementada a classe org.openmhp.si.ServiceSelectEmulation, cuja funcionalidade concentra-se em recuperar o serviço que estiver sendo apresentado, verificar (através de seu localizador) sua posição na tabela de serviços (SIDataEmulation) e assim ter uma referência ao serviços localizados na posição anterior e posterior ao serviço atual. Estas funcionalidades são disponibilizadas através dos métodos serviceUp() e serviceDown(), que percorrem circularmente a tabela de serviços.

Apesar de identificar os serviços adjacentes ao atual, o ServiceSelectEmulation não é responsável por manter o estado de um serviço, ou mesmo de manter uma referência ao serviço atual, uma vez que um novo serviço poderá ser selecionado e iniciado sem a intervenção desta classe. Para seu funcionamento correto, a classe sempre recupera o serviço atual em execução no ServiceContext (a classe ServiceSelectEmulation prevê a existência de apenas um ServiceContext, fato que pode não acontecer em uma implementação real, mas que por definição acontece no OpenMHP), e assim localizando os serviços adjacentes. Outra característica importante é que o ServiceSelectEmulation é uma classe interna ao middleware, de forma que não poderá ser instanciada por aplicações.

Por fim, para a emulação da navegação entre os serviços foi necessário adicionar tratamentos adicionais no tratamento de eventos do controle remoto disponibilizado

pelo OpenMHP. O controle remoto é representado pela classe EmulatorRemote, que define o frame com os objetos visuais e também atua como listener dos botões do controle remoto. Este listener atua principalmente acionando o gerenciador (servidor) de eventos, que é responsável por transferir os eventos diretamente para as aplicações.

As alterações foram implementados imediatamente após a notificação de novo evento para o gerenciador de eventos, capturando eventos do controle remoto referentes a mudança de canal (representado pelos botões P+ e P- no controle remoto do OpenMHP na Figura 37), e representados internamente respectivamente pelas constantes:

- VK_CHANNEL_UP: responsável por realizar uma chamada ao método serviceUp da classe ServiceSelectEmulation;
- VK_CHANNEL_DOWM: responsável por realizar uma chamada ao método serviceDown da classe ServiceSelectEmulation;

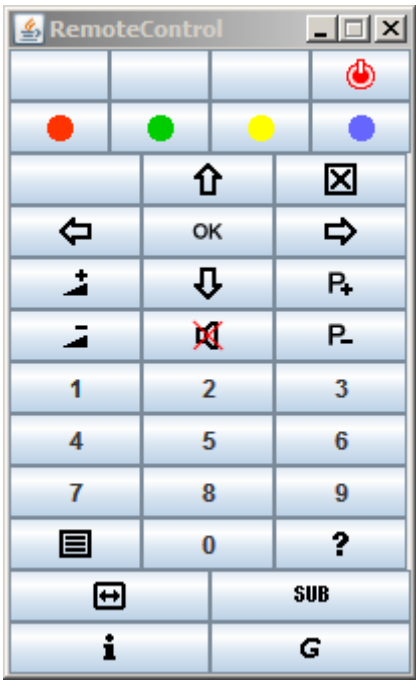

Figura 37 – Controle remoto do OpenMHP

Até este marco do trabalho não foi adicionado nenhum tratamento adicional para suporte a aplicações residentes não-nativas.

3.4.4.1. Ajustando o acesso à Application Information Table (AIT)

Em um sistema de TV Digital cada serviço inclui uma lista com suas aplicações associadas, conhecida como Application Information Table, ou AIT. Qualquer aplicação que não estiver listada não AIT de um serviço não poderá ser iniciada, e será encerrada na troca de serviço.

Além de definir a lista de aplicações disponíveis, a AIT possui outros atributos como por exemplo o seu nome, descrição, informações para seu *download* e posterior inicialização (quando deve ser iniciada, qual a classe principal, o path e o diretório da aplicação).

No status atual de implementação do OpenMHP, uma grande parcela das classes de controle do ciclo de vida das aplicações (Application Lifecycle API) são implementadas, mas restrições são encontradas devido à arquitetura e modelo de funcionamento original do OpenMHP. Como o OpenMHP não efetuava a troca de serviços em tempo de execução (ele iniciava em um serviço e lá permanecia até o fim de sua execução), ele implementava uma tabela de aplicações única para qualquer serviço que fosse inicializado, contendo todos os projetos e aplicações registrados no OpenMHP. Na realidade esta tabela era representada por um diretório (chamado Projects), sendo cada arquivo dentro deste diretório considerado um registro da AIT. Abaixo são fornecidos dois exemplos destes registros (arquivos) com seus respectivos atributos:

```
#Wed Jul 08 18:21:26 BRT 2009
debug=true
```

```
#Wed Jul 08 19:09:17 BRT 2009
debug=true
```

```
xletclass=ColorDemo
orgid=17
todo=true
trace=true
subtitles=true
fixme=true
projectname=Demo
error=true
appid=12
classloader=true
path=C\:\\openmhp104_bin\\
apps\\ColorDemo
```

Listagem 2 – Exemplo dos atributos no arquivo C:\openmhp104_bin\Projects\ ColorDemo

```
xletclass=HelloWorld
orgid=33
todo=true
trace=true
subtitles=true
fixme=true
projectname=HelloWorld
error=true
appid=9
classloader=true
path=C\:\\openmhp104_bin\\
apps\\HelloWorld
```

Listagem 3 – Exemplo dos atributos no arquivo C:\openmhp104_bin\Projects\ HelloWorld

A classe org.openmhp.application.AppsDatabaseImpl, que estende a classe org.dvb.application.AppsDatabase é responsável por criar a tabela de aplicações com seus atributos. Ela utiliza um objeto org.openmhp.system.ProjectHandler que é capaz de ler o diretório e identificar os arquivos dentro dele que representem entradas na tabela de aplicações. Desta forma, a classe AppsDatabaseImpl preenche duas tabelas separadas, representadas pela classe org.openmhp.application.AppAttributesImpl (que implementa a interface org.dvb.application.AppAttributes) e pela classe org.openmhp.application.AppProxyImpl (que implementa a interface org.dvb.application.AppProxy). A existência destas duas tabelas aparentemente redundantes acontece pois, conforme previsto pela API de manipulação de serviços do DVB, elas permitem um isolamento entre acessos que desejam consultar apenas atributos de aplicações (AppAttibutes) e acessos que desejam carregar, iniciar, pausar ou encerrar uma aplicação (AppProxy).

Para solicitar a referência a uma das entradas destas tabelas, pode-se usar o ID da aplicação ou então um filtro, como por exemplo através das classes:

- org.dvb.application.CurrentServiceFilter ou
- org.dvb.application.RunningApplicationsFilter.

Porém, como no OpenMHP inicialmente era previsto somente a execução de um serviço, esta classe não prevê a reconstrução da tabela que seria disparada durante a troca de serviço. Como a AIT é um componente importante para a troca de serviços, tornou-se necessário implementar a reconstrução do banco de dados, permitindo que a troca de serviços execute o encerramento e carregamento correto de aplicações.

O primeiro passo foi ajustar a AIT, criando tabelas individuais emuladas para cada serviço. A tabela de cada serviço foi colocada em um arquivos-texto e adicionada ao diretório AIT. Para associar a tabela ao serviço, foi utilizado o identificador do serviço como nome do arquivo, acrescentando-se a extensão txt. Observe que além de criar diversas tabelas, outra diferença do modelo originalmente usado na OpenMHP é no modelo implementado cada arquivo permite várias entradas (pois cada arquivo representa uma AIT), enquanto no modelo anterior cada arquivo representava uma única entrada, sendo o conjunto de arquivos o que formava a AIT.

Para permitir maior compatibilidade com a AIT definida no MHP, também foram definidos novos atributos, que podem ser observados na listagem abaixo, que representa o arquivo 33.txt:

```
#app_n  = app index, app type, org id, app id, name, description, provider name,
#icon path, app version, service bound, control code, xlet class, classpath, directory

app_1=  1, DVB-J, 0x0021, 0x0027, Hello World, Sample Application, YLE, , 0.4.8, ,
BOUND, AUTOSTART, HelloWorld, C\:\\openmhp104_bin\\apps\\HelloWorld,
C\:\\openmhp104_bin\\apps\\HelloWorld
```

Listagem 4 – Application Information Table do Serviço nº 33

A listagem do arquivo acima representa a AIT do service cujo identificador é o número decimal 33. Nela temos apenas uma entrada, identificada pelo app_1. Esta entrada possui diversos atributos, colocados seqüencialmente e separados pelo caractere vírgula. Observe que alguns atributos existentes na AIT original do OpenMHP não foram mantidos, são eles: debug, todo, trace, subtitles, fixme, error. Estes atributos representam apenas informações relevantes ao ambiente de emulação e portanto foram definidos de forma constante no middleware, podendo ser retirado da AIT.

Em seguida, foi realizada uma adaptação na classe AppsDatabaseImpl para que, ao invés de trazer os dados do ProjectHandler (que busca os dados da pasta Projects), fosse carregado o arquivo correspondente ao serviço atual na pasta AIT. Este recarregamento foi realizado pelo método reloadDatabase na classe AppsDatabaseImpl. A identificação do serviço atual (para localização do arquivo correto) se dá pelo acesso ao singleton da classe ServiceContextImpl, a através da qual é possível requisitar o serviço atual (getService). Tendo o serviço, extrai-se o seu Locator (getLocator), e através do Locator recupera-se o identificador do serviço. O método reloadDatabase só realiza o recarregamento da AIT se houver de fato mudança de serviço no ServiceContext, uma vez que é armazenado o Locator do último serviço para o qual se carregou uma AIT.

A troca de serviço então pode ser complementada com novas funcionalidades além das descritas na seção 3.4.4. A partir da construção dinâmica de AITs é possível realizar a troca de serviços com o carregamento de aplicações baseado na sua própria AIT, verificando-se ainda a existência de aplicações *unbound*. O processo adicionado é descrito abaixo:

1. É examinado o conjunto de aplicações em execução e todas as aplicações assinaladas como *bound* são encerradas (AppProxyImpl.stop);
2. É realizado o recarregamento da AIT para o novo serviço. Antes desta etapa deve ser alterado o atributo currentService da classe ServiceContextImpl, para que a classe AppDatabaseImpl possa recuperar o id do serviço correto;
3. Com o novo AIT verifica-se todas as aplicações em execução (STARTED e PAUSED através do método AppProxyImpl.getState()) e aquelas que não estiverem presentes no novo AIT são encerradas. Assim restaram em execução as aplicações *unbound* que estão presentes em ambos os serviços;
4. São examinadas as aplicações presentes no novo serviço e todas as aplicações assinaladas como AUTOSTART são carregadas e iniciadas (desde que estejam em estado NOT_LOADED – anterior a LOADED –, uma vez que uma aplicação *unbound* pode ter sido iniciada pelo serviço anterior).

O principal desafio na implementação desta seqüência é o fato de haver a perda da AIT antiga no recarregamento da nova AIT. Como solução, foi delegado à classe AppsDatabaseImpl a etapa 3 na lista acima. Neste caso, antes do recarregamento é feita uma cópia do vetor de objetos AppProxyImpl (que possui a referência às aplicações da AIT) e após o carregamento da nova AIT em um novo vetor ambos são comparados. Quando houver aplicações *unbound* que estiverem presentes nos dois serviços considerados no processo (portanto nos dois vetores), não será criado um novo objeto AppProxyImpl, transferindo-o apenas e mantendo-se a referência do objeto criado para o serviço anterior, garantindo a manutenção do estado da aplicação.

De forma semelhante, após a transferência dos objetos AppProxyImpl para o novo vetor, aquelas aplicações que restarem no vetor antigo com o estado STARTED ou PAUSED são encerradas.

Na Figura 38 e na Figura 39 é possível observar o resultado da troca de serviço, pressionando-se o botão P+ (inicialmente o serviço 33 e em seguida o serviço 81):

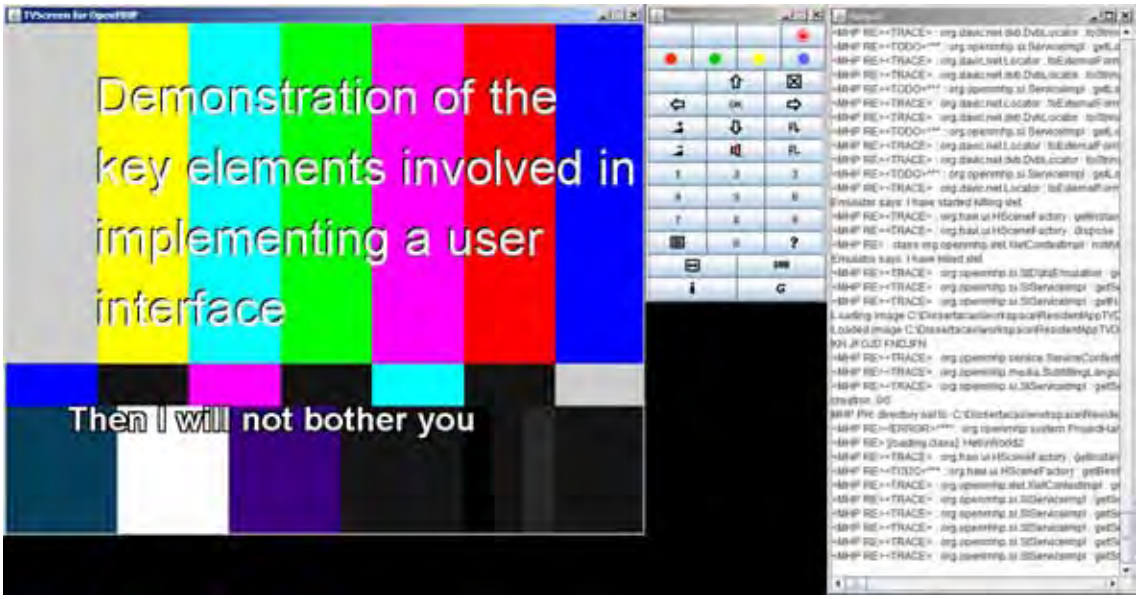

Figura 38 – Apresentação do serviço (id 33) com sua aplicação e legenda

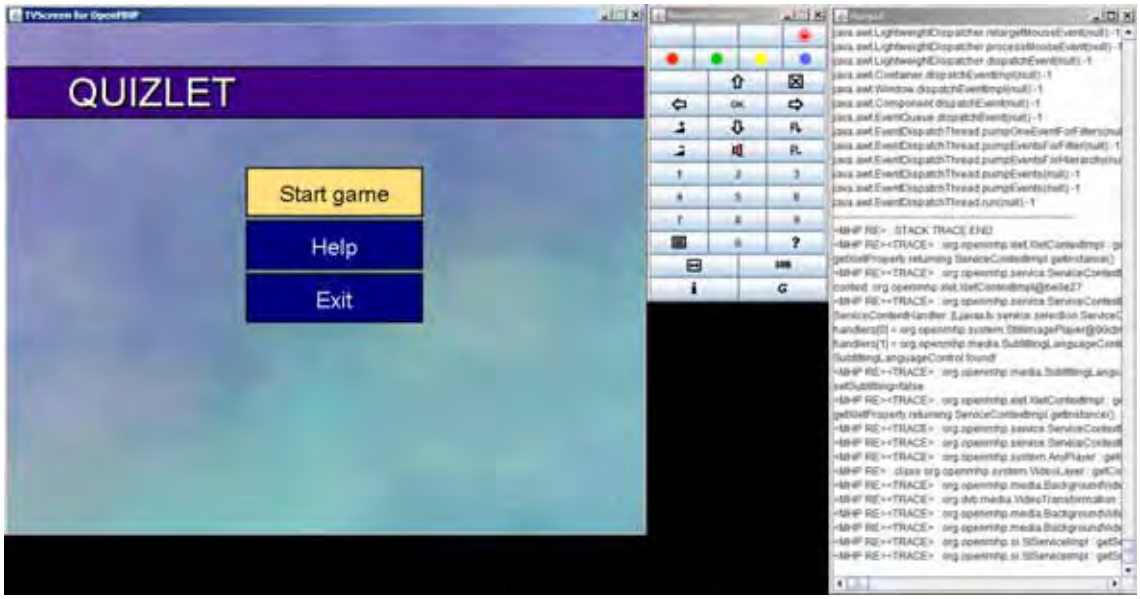

Figura 39 – Apresentação do serviço (id 81) com sua aplicação

3.4.4.2. Exibindo o nome e identificador do serviço na Extra Layer

Um tratamento adicional que foi necessário para a seleção e navegação entre os serviços foi a exibição do seu nome e identificador, com o objetivo de trazer uma referência visual (no dispositivo MHP) ao serviço sendo apresentado.

Conforme já apresentado, o OpenMHP possui diversas superfícies gráficas sobrepostas, cada qual com a sua função. Seus arquitetos, porém, planejaram preventivamente uma camada excedente, chamada de Extra Layer, posicionada sobre todas as demais (inclusive sobre a Graphics Layer – aplicações – e sobre a Subtitle Layer – legenda), e que até então não possuía nenhuma função no OpenMHP. Sendo assim, tendo como objetivo garantir o isolamento necessário do middleware e não interferir o funcionamento de background, player de vídeo/imagens, aplicações ou legendas, utilizou-se a Extra Layer para exibir as informações adicionais.

A Extra Layer é adicionada diretamente na classe MHPDevice, responsável por ser uma abstração do dispositivo físico MHP, e que por ser uma implementação em PC é representada por um Container AWT que adiciona cada superfície (layer) a si mesma como componentes-filhos, permitindo que o sistema gráfico do AWT renderize as camadas corretamente.

A troca das informações na Extra Layer foi realizada no método select da classe ServiceContext onde, após a troca de serviço no contexto, é realizada chamadas à métodos da Extra Layer, enviando o nome e identificador do novo serviço, sendo o Extra Layer responsável por desenhar estas informações na tela sempre que for necessário redesenhar sua superfície gráfica.

O resultado pode ser observado abaixo com os serviços 33 e 81:

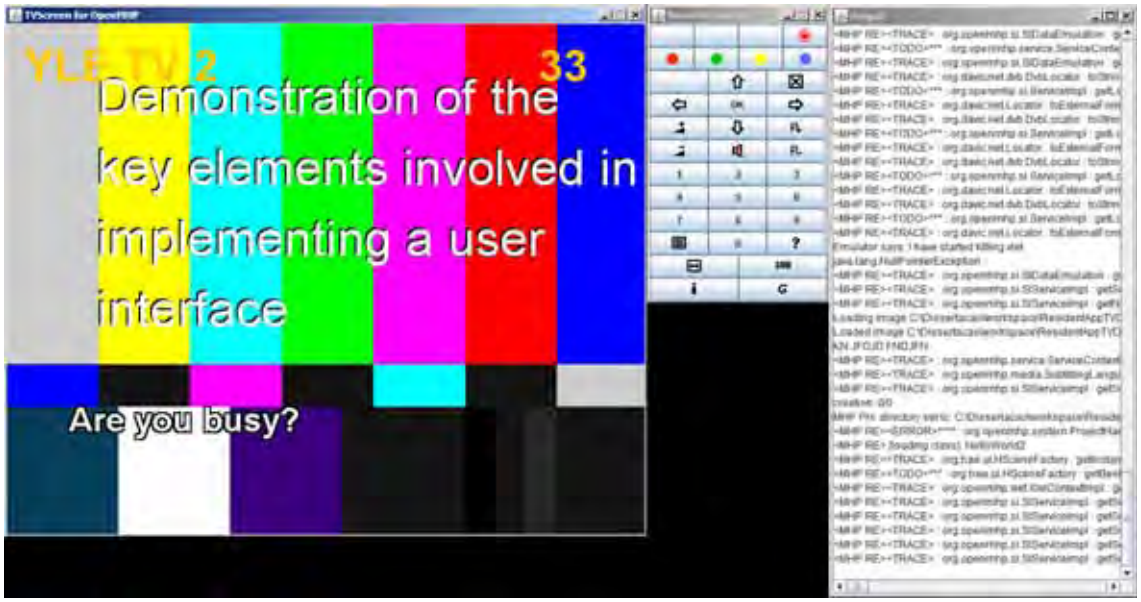

Figura 40 – Serviço 33 com nome e identificador na Extra Layer

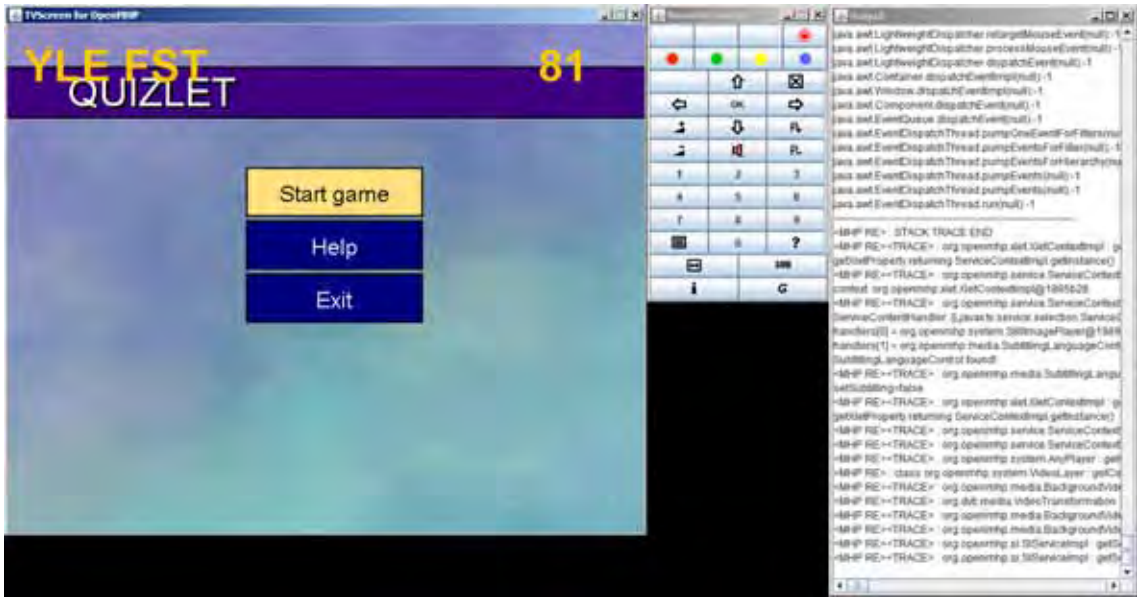

Figura 41 – Serviço 81 com nome e identificador na Extra Layer

3.4.5. Implementando o suporte à aplicações residentes

Tendo realizado as implementações, adaptações e extensões detalhadas nas seções anteriores, iniciou-se a implementação do suporte à aplicações residentes no middleware OpenMHP. Conforme especificado na seção 3.4.1, a arquitetura definida

para esta nova extensão prevê que cada aplicação residente não-nativa execute dentro de um ServiceContext, exatamente como um serviço, sendo sua inicialização executada pelas mesmas classes e rotinas de seleção de serviços.

Para então implementar esta arquitetura e incluir o suporte à aplicações residentes não-nativas no OpenMHP, foram executadas as seguintes etapas:

1. A primeira parte da implementação foi a criação de uma tabela de aplicações residentes não-nativas que se encontram instaladas (persistidas até ação do usuário que a remova) no middleware. De forma semelhante à lista de serviços e às diversas AITs, a lista de aplicações residentes não-nativas foi definida em um arquivo-texto (residentapp.txt). Por definir aplicações, seu conteúdo é bastante semelhante às AITs:

```
#app_n  = app index, app type, org id, app id, name, description, provider name, icon path, app version, control code, xlet class, classpath, directory
app_1=  1, DVB-J, 0x0021, 0x0027, Hello World, Sample Application, YLE, , 0.4.8, , AUTOSTART, HelloWorld, ,
app_2=  2, DVB-J, 0x0031, 0x0009, Quizlet, Case study app, OpenMHP team, , 1.0.4, , AUTOSTART, Quizlet.Quizlet, ,
```

Listagem 5 – Tabela de aplicações residentes não-nativas

2. O segundo passo para esta implementação foi criar um novo Locator para aplicações residentes. Este novo Locator tem como objetivo permitir o referenciamento de conteúdo para serviços de aplicações não-nativas. O uso de locators tradicionais do DVB (iniciados por dvb://) não seria suficiente para trazer a identificação correta para os novos serviços, uma vez que as demais APIs precisam prever a existência desses novos serviço e oferecer um comportamento diferente quando for consumida por aplicações pertencentes a estes serviços. Desta forma, foi criada uma classe org.openmhp.resident.ResidentAppLocator, que estende a classe

org.davic.net.dvb.DvbLocator e reimplementa todos os seus métodos para que na construção e manipulação da URL do locator seja inserido e considerado com o prefixo “resapp://” ao invés do prefixo “dvb://”;

3. De forma semelhante ao item anterior, foi criada uma nova classe que é a representação de um serviço de aplicação residente não-nativa. Esta classe é identificada por org.openmhp.resident.ResidentAppServiceImpl e estende a classe org.openmhp.si.ServiceImpl. Seu conteúdo é idêntico à classe antecessora, porém o uso de DvbLocator é substituído por ResidentAppLocator;
    a. A classe ResidentAppServiceImpl é responsável também por conter os dados necessários para carregamento da aplicação. Tais dados serão utilizados nas etapas seguintes para construção de uma AIT emulada.
4. O próximo passo consistiu na criação de um bando de dados de aplicações residentes, similar à classe SIDataEmulation. Esta nova classe, denominada org.openmhp.resident.ResidentAppDataEmulation, é responsável por executar o carregamento e parse do arquivo residentapp.txt, preenchendo o vetor com a lista de aplicações residentes.
    a. Um diferencial desta classe é a existência de um novo método público (reload) que recarrega a tabela do arquivo-texto, permitindo que aplicações residentes não-nativas sejam adicionadas ou removidas do arquivo-texto em tempo de execução (simulando o processo de instalação/desinstalação);
    b. Ao criar dois tratamentos para AITs de serviços tradicionais e o AIT emulado de um serviço de aplicação residente não-nativa, também se garante o escopo destas aplicações residentes não-nativas, uma vez que

elas não poderão ser carregadas por outras aplicações através da API do Gerenciador de Aplicações.

5. Também foi necessário prever a existência destes novos serviços na classe AppDatabaseImpl. Nesta classe é realizado o carregamento da AIT de cada serviço e também feito o gerenciamento da transferência de aplicações *unbound* comum a ambos os serviços de uma transição. A adaptação realizada ajustou o método reloadDatabase para verificar inicialmente qual o tipo do serviço corrente. Se for identificado um serviço DVB, mantém-se o mesmo tratamento. Caso o serviço corrente seja uma aplicação residente não-nativa, o vetor AppAttributesImpl e o vetor AppProxyImpl recebem apenas um elemento (uma entrada), que representa a aplicação residente não-nativa em questão (pois de fato esta será a única aplicação presente neste serviço).
    a. Outro tratamento realizado é o trecho referente à transferência de aplicações *unbound* também possui um duplo tratamento, uma vez que quando a transição acontece de um serviço de aplicação residente não-nativa, ou para um serviço de aplicação residente não-nativa, não pode acontecer transferência de aplicações *unbound*, sendo necessário encerrar todas as aplicações do serviço anterior. Isto garante o isolamento necessário e a não-interferência de aplicações residentes não-nativas nas aplicações tradicionais, e vice-versa;
    b. Adicionalmente, para garantir o isolamento necessário, é ajustado na AIT simulada o parâmetro *bound* e o código de controle AUTOSTART para todas as aplicações residentes não-nativas;
    c. Uma vez que a classe ResidentAppServiceImpl possui os dados necessários para carregamento da aplicação, não é necessária uma nova

consulta à classe ResidentAppDataEmulation (que carrega os dados do arquivo residentapp.txt), basta apenas buscar o serviço corrente no ServiceContext.

6. No método select da classe ServiceContextImpl as adaptações ocorreram de forma a limitar o contexto do serviço para que a aplicação não tivesse acesso a elementos de um serviço tradicional que não estariam presentes em um serviço de aplicação residente não-nativa. Desta forma, tarefas como *reset* do Background e troca de aplicações (a AIT é transparente para a ServiceContext, uma vez que ela é emulada) mantiveram-se. Porém nas seguintes tarefas foi necessário um tratamento especial para aplicações residentes não-nativas:
    a. Video Layer: como não haveria a presença de conteúdo na Video Layer, esta camada foi apenas oculta;
    b. O nome e o identificador do novo serviço foram buscados na própria classe ResidentAppServiceImpl fornecida como parâmetro, enquanto nos serviços DVB estes dados eram buscados no banco de dados SIDataEmulation;
    c. Utilizando os media handlers que não possuem conteúdo em serviço de aplicações residentes não-nativas, interrompe-se o ServiceMediaHandler (método stop) e o SubtitlingLanguageControl (setSubtitling (false)).
7. Outro tratamento importante realizado na classe ServiceContextImpl foi no método getServiceContentHandlers. Adicionalmente às funções executadas, este método também passou a ser responsável por limitar os ServiceContentHandler devolvidos quando o serviço corrente for um serviço de aplicação residente não-nativa. Desta forma, ServiceContentHandler's como o ServiceMediaHandler e o

SubtitlingLanguageControl não são enviados a aplicações residentes não-nativas.

   a. Este tratamento complementa o tratamento realizado no método select desta mesma classe. A Video Layer, por exemplo, é apenas oculta (não é removida do dispositivo MHP), e poderia tornar-se visível novamente caso a aplicação tivesse acesso ao ServiceMediaHandler.

8. Por fim, é realizado o tratamento na classe ServiceSelectEmulation, para que seja previsto a existência de aplicações residentes não-nativas e que também elas estejam presentes na navegação entre serviços, ou seja, na navegação com os botões P+ e P- do controle remoto o usuário poderá iniciar os serviços de aplicações residentes não-nativas logo após os serviços tradicionais.

   a. Desta forma, a classe ServiceSelectEmulation deverá no seu carregamento verificar a quantidade de serviços tradicionais e a cada iteração (channelUp e channelDown) verificar a quantidade de serviços de aplicações residentes não-nativas (pois esta quantidade pode ser alterada em tempo de execução).

   b. De forma semelhante aos serviços tradicionais, onde é possível determinar a seqüência dos serviços analisando-se o vetor presente na classe SIDataEmulation, é também possível determinar a ordem dos serviços de aplicações residentes não-nativas através do vetor mantido pela classe ResidentAppDataEmulation.

   c. Na chamada para o método changeService(int destinationServiceIndex), primeiro determina-se se o serviço é um serviço DVB (SIDataEmulation) ou um serviço de aplicação residente não-nativa (ResidentAppDataEmulation) ou ainda um serviço inexistente (menor

que 1 ou maior que a quantidade de serviços DVB somada à quantidade de serviço de aplicação residente não-nativa – neste caso é selecionado o primeiro ou o último serviço da lista). Se for um serviço DVB, recupera-se o ServiceImpl correspondente da classe SIDataEmulation. Se for um serviço de aplicação residente não-nativa, recupera-se um ResidentAppServiceImpl da classe ResidentAppDataEmulation. É então feita uma chamada ao método select, do ServiceContext com o novo serviço como parâmetro.

Com esta seqüência de implementações, foi então implementado o suporte a aplicações residentes não-nativas no middleware através de serviços adicionas independentes. Desta forma, temos o cenário de estudo completo para a coleta de dados.

3.4.6. *Launcher* de aplicações residentes - XletChannel

Para permitir os testes deste trabalho foi implementada uma adaptação adicional que permitisse a listagem das aplicações residentes não-nativas e, de acordo com os comandos do usuário, carregar o serviço correspondente.

A abordagem para esta adaptação foi criar uma aplicação nativa (portanto residente) que, assim como as aplicações residentes não-nativas, pudesse executar em um serviço exclusivo. Desta forma, a adaptação foi realizada de forma semelhante à realizada para as aplicações residentes não-nativas, porém sem permitir escalabilidade, pois a tabela foi acrescentada no próprio código da aplicação.

Na seqüência de serviços, esta aplicação nativa foi posicionada no serviço imediatamente posterior aos serviços tradicionais presentes na SIDataEmulation (e todos os serviços de aplicações residentes não-nativas foram deslocados em uma unidade).

Esta abordagem foi utilizada com o objetivo de garantir o isolamento desta aplicação nativa, para que não houvesse interferência no modo de funcionamento das aplicações tradicionais e também das aplicações residentes não-nativas.

Esta aplicação, chamada de XletChannel, ainda possui uma característica importante. Ela é um Xlet, executa sobre as APIs disponíveis para as Xlets, e segue o ciclo de vida de um Xlet, mas por ser uma aplicação nativa sua compilação foi realizada diretamente com o código do middleware, permitindo que ela também reconhecesse funções pertencentes ao middleware como a ResidentAppDataEmulation, que possui a lista de serviços. Porém, para que ela pudesse instanciar estas classes foi necessário uma adaptação no ClassLoader para que, quando o serviço fosse de uma aplicação nativa, a lista de classes permitidas (allowedClasses.txt) fosse maior, abrangendo todo o middleware. Já as funcionalidades para seleção de serviço em um ServiceContext está disponível para todas as aplicações através da Service API.

O resultado foi a aplicação abaixo, executando como o 16º serviço da lista, entre os serviços tradicionais e os serviços de aplicações residentes não-nativas.

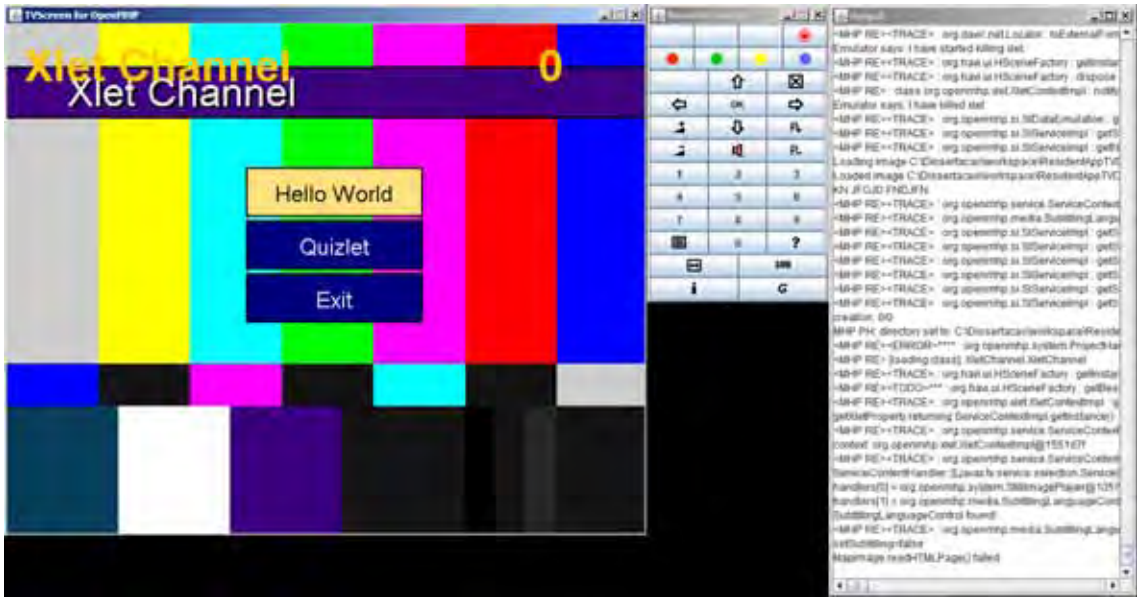

Figura 42 – XletChannel listando aplicações residentes não-nativas disponíveis

3.5. Coleta de dados e discussão

A partir das implementações descritas nas seções anteriores, tem-se o ambiente completo para o estudo de caso. Primeiramente foi realizada revisão sobre as tabelas com o objetivo definir para o estudo de caso um conjunto de serviços tradicionais e um conjunto de aplicações residentes não-nativas da qual o aplicativo Quizlet fizesse parte.

Nesta etapa busca-se determinando quais, dentre as APIs utilizadas por esta aplicação, são conflitantes ou incompatíveis com um middleware que suporte concomitantemente o modelo de funcionamento do canal de aplicações residentes não-nativas e com o modelo de funcionamento dos canais tradicionais.

As execuções consecutivas, testes de caixa-preta, depuração de código bem como análise das APIs realizadas nesta etapa gerou um volume de dados que são listados, detalhados e discutidos nas próximas seções juntamente com novas propostas para extensões ao middleware que permitam manter seu estado de funcionamento correto.

As figuras abaixo ilustram a execução do aplicativo Quizlet como aplicação residente não-nativa e o ambiente configurado na IDE Eclipse para depuração de código-fonte e análise de APIs:

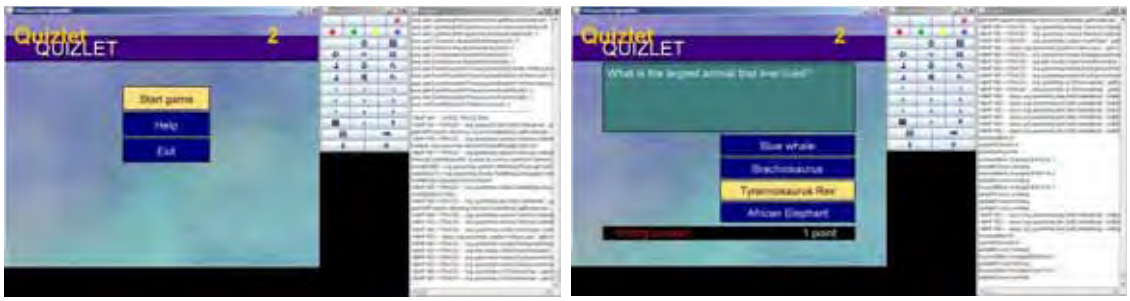

Figura 43 – Coleta de dados - Quizlet executando como aplicação residente não-nativa

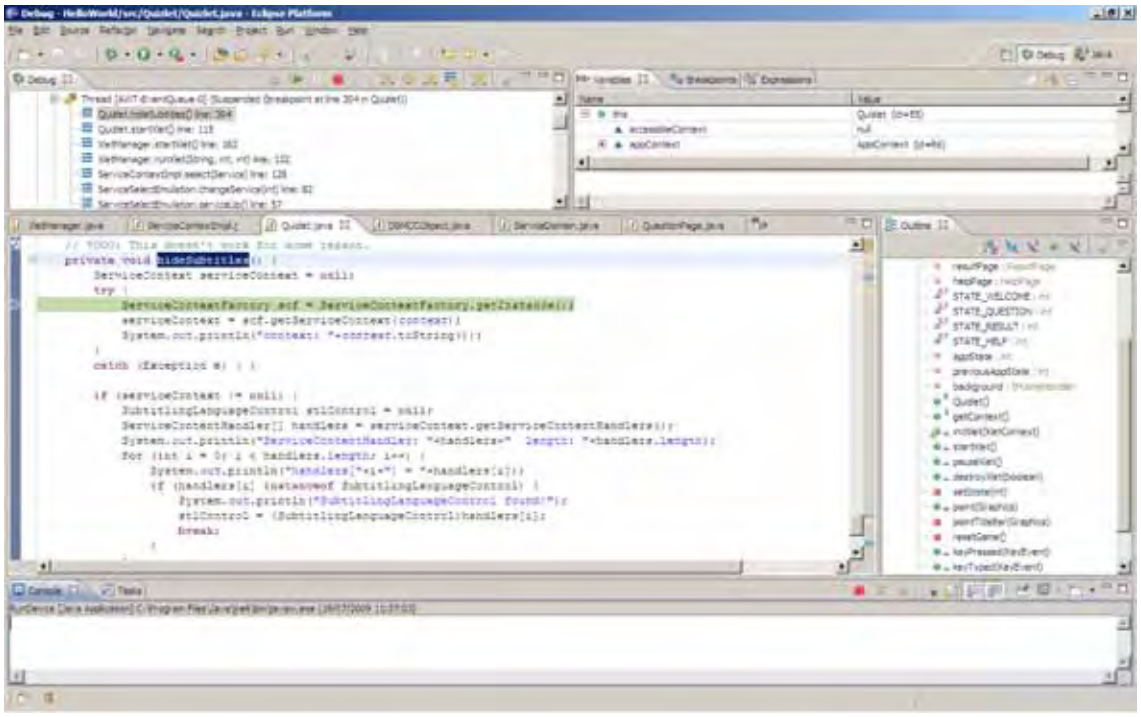

Figura 44 – IDE Eclipse utilizada para a análise de APIs e depuração durante a coleta de dados

3.5.1. API Gráfica

Quanto à API gráfica, foi identificado uma característica comum aos serviços de aplicações residentes não-nativas é a ausência de alguns *handlers* que dariam acesso às superfícies gráficas (como por exemplo a VideoLayer), ou a recursos como o subtitle. Estes handlers são acessados através do método getServiceContentHandlers da classe ServiceContext. Tradicionalmente aplicações acessam a estes recursos com o objetivo de efetuar configurações visuais para a melhor apresentação de seu gráfico.

Sobre este aspecto é necessário apenas destacar que as aplicações devem considerar portanto que estes elementos podem não estar acessíveis e medidas secundárias devem ser tomadas para configuração somente das layers as quais a aplicação tem acesso, como a Graphics Layer. Cabe também ressaltar que, apesar de ser incomum, mesmo para aplicações tradicionais pode existir situações onde algum dos

handlers não são disponibilizados, seja por restrição do middleware ou por restrição do serviço, de forma que aplicações sempre devem prever a ausência de tais controladores.

De forma semelhante foi necessário o tratamento das classes do pacote HAVi. A classe HScreen possui a seguinte interface:

```
public class HScreen
{
        public static HScreen[] getHScreens();
        public static HScreen getDefaultHScreen();

        public HVideoDevice[] getHVideoDevices();
        public HGraphicsDevice[] getHGraphicsDevices();

        public HVideoDevice getDefaultHVideoDevice();
        public HGraphicsDevice getDefaultHGraphicsDevice();
        public HBackgroundDevice getDefaultHBackgroundDevice();

        public HScreenConfiguration[]
          getCoherentScreenConfigurations();

        public boolean setCoherentScreenConfigurations(
          HScreenConfiguration[] hsca);
}
```

Listagem 6 – Interface da classe HScreen

O tratamento foi realizado nos métodos getDefaultHVideoDevice e getHVideoDevices, de forma que eles retornem sempre o valor nulo quando o serviço corrente for o serviço de aplicações residentes não-nativa, uma vez que não é exibido conteúdo na Video Layer.

Não foi identificada nenhuma incompatibilidade, observação ou tratamento especial em relação às classes HScene, HContainer, HComponent e suas derivadas, bem como no MHP Widget set.

3.5.2. Service Selection API

Uma vez que todo o suporte a aplicações residentes é feito através da Service Selection API, e todos os tratamentos identificados foram suportados, não foi identificada nenhuma situação especial no estudo de caso que necessitasse de ajustes implementação realizada até então.

Destaca-se que esta API tornou-se central na arquitetura proposta, onde aplicações residentes são referenciadas como serviços (através de Locators) e devem ser carregadas usando a Service Selection API, seja por classes do middleware (como a emulação de navegação) como por outras aplicações.

3.5.3. Locators

Apesar de não observado nenhum comportamento inesperado durante o estudo de caso, e não encontrada nenhuma evidência direta durante a análise de APIs, acredita-se que o uso de uma nova classe de Locators para serviços de aplicações residentes não-nativas pode trazer incompatibilidades com algumas rotinas da API service selection. Isto acontece pois os conteúdos que são referenciados por Locators são tradicionalmente Transport Streams, Serviços ou partes de um serviço, (todos identificados pelo prefixo dvb://) ou Image Drips (referenciados pelo prefixo dripfeed://).

3.5.4. Ciclo de vida da aplicação

Apesar de seguir o modelo e o ciclo de vida de Xlets, identificou-se no estudo de caso que, apesar de serem tratadas como serviço, as aplicações residentes não-nativas deverão possuir tendência em suas funcionalidades à buscar copiar o modelo standalone

de PCs. Desta forma uma nova melhoria foi realizada de forma sugestiva, sem haver de fato a necessidade de implementá-las por questões de compatibilidade.

Esta funcionalidade é implementada na classe org.openmhp.system.XletManager, utilizada pela AppProxyImpl, e responsável por ser o *Xlet launcher*, e atende apenas às aplicações residentes não-nativas. De forma assíncrona, após a chamada ao método killXlet, aguarda-se o estado DESTROYED para a Xlet e em seguida utiliza a Service Selection API para selecionar o serviço referente ao XletChannel. Dessa forma, sempre que uma aplicação residente não-nativa for encerrada, automaticamente o serviço correspondente ao XletChannel é selecionado, uma vez que no serviço da aplicação residente não-nativa nenhuma outra interação será feita e nenhum outro conteúdo será apresentado.

3.5.5. Media Control

No controle de mídia, verificou-se que a principal restrição já havia sido prevista, que se trata de ocultar os *handlers* de fluxos de mídia que não estão presentes em um serviço de aplicação residente não-nativa. Também foi verificado que o uso de novos players para exibição de mídia na Graphics Layer (um vídeo ou áudio armazenado localmente), através da JMF, apresentou funcionamento correto.

Observou-se também que quando uma aplicação é associada um service, ela poderá utilizar um recurso para sincronizar seu comportamento com as ações dos demais elementos do serviço (por exemplo o fluxo audiovisual). Neste caso o acesso ao *clock* referente à mídia não é realizado através dos *handlers*. Para isso é utilizado o DSM-CC Normal Play Time (NPT), um contador que indica o “tempo” da mídia corrente (mas não baseado no sistema tradicional de unidades – hora, minuto e segundo). Para realizar a sincronia as aplicações devem, solicitar a lista de eventos

disponíveis na classe DSMCCStreamEvent (utilizando-se o método getEventList), e inscrever-se para escutar quando o evento for acionado, disparando o gatilho que notifica o listener.

Apesar de ser possível realizar um filtro no método getEventList, para que ele não retornasse nenhum evento referente ao NPT, a abordagem utilizada manteve toda a lista de eventos referentes ao DSM-CC e considerou que em serviços de aplicações residentes não-nativas, a ausência do broadcast faria levaria a nenhum destes eventos ser acionado. Mesmo que aplicações insiram listeners para os eventos, nenhum gatilho é disparado.

3.5.6. DSM-CC

Conforme discutido na seção anterior, o acesso ao Carrossel de Dados e Objetos foi mantido, garantindo-se porém que com a ausência de broadcast nenhum objeto poderia ser instanciado. Desta forma, permite-se instanciar o ServiceDomain (carrossel), mas quanto a aplicação realiza a chamada ao método *attach* com o locator do objeto que ele pretende recuperar do carrossel (DSMCCObject), é lançada uma exceção do tipo DSMCCException uma vez que este objeto não existe.

Na prática, o OpenMHP instancia arquivos quando são referenciados objetos DSMCCObject, porém ele considera um único carrossel de dados para todas as aplicações. Desta forma, foi necessário realizar uma verificação para que nenhum arquivo pudesse ser carregado quando o serviço corrente for um serviço de aplicação residente não-nativa.

3.5.7. Persistent Storage

O acesso ao persistent storage é permitido para serviços de aplicações residentes não-nativas, oferecendo a possibilidade de salvar o seu estado. Neste ponto é importante observar que o *persistent storage* é organizado utilizando o identificador da organização e o identificador da aplicação. Neste caso, quando houver aplicações residentes não-nativas com o mesmos identificadores de aplicações residentes não-nativas, elas compartilharão um mesmo *persistent storage*.

3.5.8. Event Manager

Uma vez que as aplicações residentes não-nativas são tratadas como aplicações pertencentes a serviços, não houve mudanças necessárias no Event Manager, todos os eventos do usuário (controle remoto) foram gerados corretamente e encaminhados (notificados) à aplicação.

3.5.9. Application Manager API

Outra API que não requereu nenhuma nova alteração foi a API de Application Manager, pois ela usa como referência a AIT do serviço (AppDatabase), que em serviços de aplicação residente não-nativa possui apenas uma entrada, que é a própria aplicação residente não-nativa.

3.5.10. Tuning, Return Channel e Comunicação Inter-Xlet

Não foram identificadas alterações necessárias nestas APIs.

3.6. Teste de compatibilidade

Uma vez que o objetivo deste trabalho é avaliar a possibilidade de desenvolvimento de um middleware aderente às normais tradicionais de TV Digital e que adicionalmente possua extensões para suporte a aplicações residentes não-nativas, foi necessário a execução de testes e coleta de dados de aplicações tradicionais executando sobre serviços tradicionais. Através deste teste foi possível validar que o middleware continua compatível com o modelo tradicional de TV Digital concomitantemente com o suporte às aplicações residentes não-nativas.

Foram utilizados conjuntos de aplicações de acordo com as APIs às quais fazem acesso. Estes grupos serão considerados experimentos que serão executados sobre a implementação original do OpenMHP e da implementação final deste trabalho. Neste teste serão coletados os problemas de compatibilidade e serão relatadas novas descobertas. Para este teste foi usado o seguinte contexto:

1. O conjunto de aplicações:
    a. HelloWorld2: Aplicação sample que demonstra como inserir os conteúdos principais nos objetos visuais HAVi;

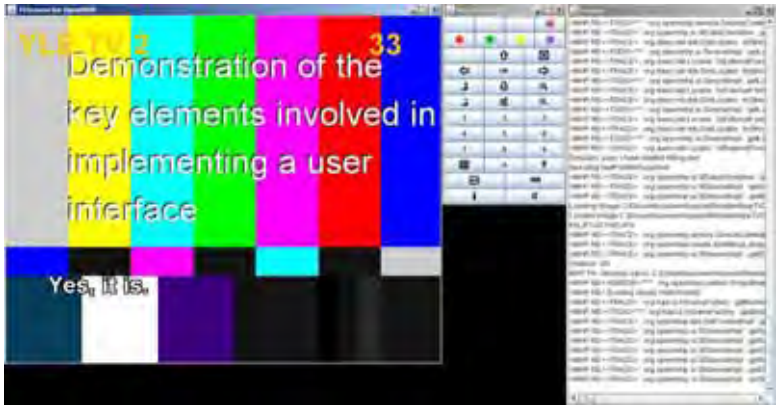

Figura 45 – Exemplo de construção de interface HAVi

    b. HelloWorld: Aplicação que manipula os principais *handlers* de conteúdo e interrompe a execução de conteúdo na Video Layer;

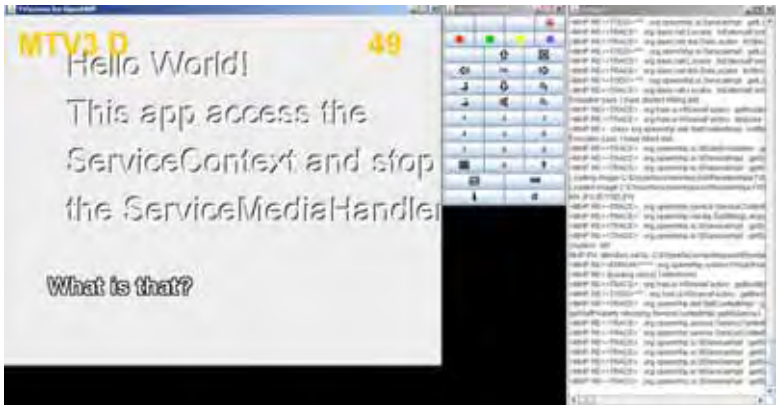

Figura 46 – Aplicação que manipula o *content handler* da Video Layer

c. HelloWorld: Aplicação que manipula os principais *handlers* de conteúdo e interrompe a apresentação de legenda na Subtitle Layer;

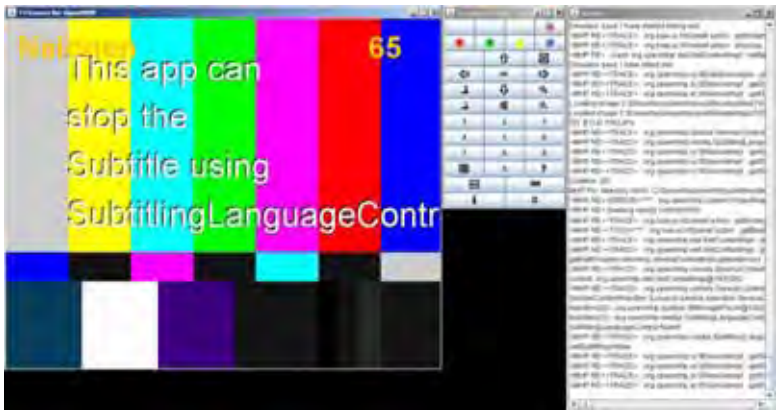

Figura 47 – Aplicação que manipula o *content handler* da Subtitle Layer

d. Quizlet

e. Quadra: um jogo de tetris open source desenvolvido

Figura 48 – Quadra: um jogo e tetris open source

2. O arquivo services.txt foi definido com o seguinte conteúdo:

```
# finland
original_network_id =0x20F6

#network_n = network number, region, network_name_descriptor, id (hex)
network_1 = 2, Turku, Digita_Turku, 0x3302

#service_n  = logical channel num, transportid (hex), serviceid (hex), si type, name, provider name
service_1=  1,   0x1001,   0x0011,  1,   YLE TV 1,          YLE
service_2=  2,   0x1001,   0x0021,  1 ,  YLE TV 2 ,              YLE
service_3=  3,   0x2001,   0x0031,  1 ,  MTV3 D ,           MTV Oy
service_4=  4,   0x3001,   0x0041,  1 ,  Nelonen ,          Ruutunelonen
service_5=  5,   0x1001,   0x0051,  1 ,  YLE FST ,          YLE
service_6=  6,   0x2001,   0x0061,  1 ,  Subtv ,            Citytv Oy
service_7=  10,  0x2001,   0x00A1,  1 ,  Urheilukanava,     Suomen Urheilutelevisio
service_8=  12,  0x1001,   0x00C1,  1 ,  YLE24 ,            YLE
service_9=  13,  0x1001,   0x00D1,  1 ,  YLE Teema ,             YLE
service_10= 16,  0x3001,   0x0111,  1 ,  Nelonen Plus,      Ruutunelonen
service_11= 22,  0x3001,   0x0161,  1 ,  Estradi,           Various (Administrated by Digita Oy)
service_12= 26,  0x3001,   0x01A1,  1 ,  CANAL+,                 CANAL+
service_13= 27,  0x3001,   0x01B1,  1 ,  CANAL+ KULTA ,     CANAL+
service_14= 28,  0x3001,   0x01C1,  1 ,  CANAL+ SININEN,    CANAL+
service_15= 29,  0x3001,   0x01D1,  1 ,  CANAL+ Infokanava,      CANAL+
```

Listagem 7 – Arquivo services.txt

3. Os arquivos no diretório AIT são:
    a. 33.txt

```
#app_n  = app index, app type, org id, app id, name, description, provider name, icon path, app version, service bound, control code, xlet class, classpath, directory
app_1=  1, DVB-J, 0x0021, 0x0027, Hello World 2, Sample Application, YLE, , 0.4.8, , BOUND, AUTOSTART, HelloWorld2, ,
```

Listagem 8 – Arquivo 33.txt no diretório AIT

    b. 49.txt

```
#app_n  = app index, app type, org id, app id, name, description, provider name, icon path, app version, service bound, control code, xlet class, classpath, directory
app_1=  1, DVB-J, 0x0005, 0x0001, Hello World, Sample Application, MTV3, , , , BOUND, AUTOSTART, HelloWorld, ,
```

Listagem 9 – Arquivo 49.txt no diretório AIT

    c. 65.txt

```
#app_n  = app index, app type, org id, app id, name, description, provider name, icon path, app version, service bound, control code, xlet class, classpath, directory
app_1=  1, DVB-J, 0x0006, 0x0001, Hello World, Sample Application, Nelonen, , , , BOUND, AUTOSTART, HelloWorld3, ,
```

Listagem 10 – Arquivo 65.txt no diretório AIT

d. 81.txt

```
#app_n  = app index, app type, org id, app id, name, description, provider name, icon path, app version, service bound, control code, xlet class, classpath, directory
app_1=  1, DVB-J, 0x0007, 0x0001, Quizlet, Quiz Game, YLE, , , , BOUND, AUTOSTART, Quizlet.Quizlet, ,
```

Listagem 11 – Arquivo 81.txt no diretório AIT

e. 97.txt

```
#app_n  = app index, app type, org id, app id, name, description, provider name, icon path, app version, service bound, control code, xlet class, classpath, directory
app_1=  1, DVB-J, 0x0008, 0x0001, Quadra, Tetris Game, Subtv, , , , BOUND, AUTOSTART, Quadra, ,
```

Listagem 12 – Arquivo 97.txt no diretório AIT

4. O arquivo residentapp.txt foi definido com o seguinte conteúdo:

```
#app_n  = app index, app type, org id, app id, name, description, provider name, icon path, app version, control code, xlet class, classpath, directory
app_1=  1, DVB-J, 0x0021, 0x0027, Hello World, Sample Application, YLE, , 0.4.8, , AUTOSTART, HelloWorld, ,
app_2=  2, DVB-J, 0x0031, 0x0009, Quizlet, Case study app, OpenMHP team, , 1.0.4, , AUTOSTART, Quizlet.Quizlet, ,
```

Listagem 13 – Arquivo residentapp.txt

Com este cenário, foram realizadas execuções consecutivas e análise do comportamento das aplicações em um middleware Open MHP original (portanto sem suporte a aplicações residentes não-nativas) e em um middleware com todas as adaptações e extensões propostas neste trabalho. É importante destacar que o middleware original precisa de configurações adicionais para iniciar diretamente em um serviço e executando uma aplicação.

Durante os testes não foram observadas nas aplicações incompatibilidades geradas pelas adaptações e extensões desenvolvidas, exceto pequenos bugs gerados pelos novos códigos (que, quando identificados, eram solucionados e os testes reiniciados). Sendo assim, determinou-se que o comportamento das aplicações tradicionais em serviços tradicionais manteve-se inalterado e nenhuma nova descoberta foi realizada sobre este estudo de caso.

3.7. Resultados e Conclusão

A presente dissertação descreveu os conceitos necessários sobre TV Digital em todo o mundo, as considerações metodológicas, o processo de desenvolvimento da proposta realizada e a coleta de dados para verificação da teoria apresentada. Dentro da metodologia proposta, tem-se neste trabalho o processo utilizado para o desenvolvimento de um middleware para TV Digital (baseado em uma implementação de referência) que oferecesse suporte a uma nova classe de aplicações, além de detalhar os ajustes preliminares realizados na implementação de referência escolhida, o OpenMHP.

Com o cenário definido e as extensões propostas implementadas, a coleta de dados trouxe novas descobertas que foram discutidas e soluções complementares foram propostas, sempre buscando cumprir o objetivo inicialmente definido.

Com o cenário completo implementado após com os tratamentos realizados após a coleta de dados pode-se finalmente verificar a teoria proposta dentro do cenário definido para o estudo de caso. Identificou-se, portanto que no cenário proposto é possível realizar as adaptações propostas, de forma a suportar a nova classe de aplicações residentes não-nativas, mantendo compatibilidade do middleware com as aplicações e serviços tradicionais.

Cabe também uma consideração metodológica, de que o processo adotado é um estudo de caso, sendo possível a realização deste mesmo processo com casos ainda mais abrangentes, em especial com um conjunto de aplicações que permita verificar uma maior variedade funcionalidades, rotinas, classes e APIs dentro do middleware OpenMHP, podendo relatar novas descobertas e possíveis incompatibilidades não detectadas neste trabalho.

Outro aspecto importante é a portabilidade de aplicações entre os dois ambientes, pois uma vez que o ambiente de execução para aplicações residentes não-nativas é mais restrito elas também poderão ser utilizadas nos serviços tradicionais. Por outro lado, as aplicações tradicionais também poderão ser executadas como aplicações residentes não-nativa, obviamente considerando-se, desde que estas aplicações não assumam a existência de todas as propriedades e estejam preparadas para que suas algumas de suas requisições não sejam contempladas.

Recomenda-se ainda, que este tipo de implementação pode ser traduzida em normas e harmonizada com as normas já existentes para middlewares de TV Digital, como a especificação do MHP e do SBTVD.

Além de verificada a praticalidade da proposta deste trabalho, considera-se que esta abordagem para uso de aplicações baseadas em serviços pode ser reaproveitada para as demais aplicações nativas. Esta abordagem favorece o compartilhamento de recursos escassos (uma vez que ocorre a troca de serviço e os recursos alocados pelo serviço anterior são desalocados), não interfere no funcionamento do conjunto de aplicações dos serviços e suas aplicações e também não interfere a apresentação de conteúdo, uma vez que a sobreposição de conteúdo poderia interferir na receita das emissoras.

Este conjunto de extensões, se praticado nas Set-Top Boxes comerciais, poderão definir uma nova dinâmica para apresentação de conteúdo e acesso a serviços diversos, com um novo modelo de distribuição de aplicações.

Ainda cabe uma observação sobre temas importantes para um ambiente de TV Digital, poderão ser considerados em trabalhos futuros como segurança, gerenciamento de recursos escassos, distribuição e instalação de aplicações, interoperabilidade entre diversos ServiceContext, migração deste modelo para aplicações nativas ou a execução

de um trabalho semelhante sobre uma implementação de referência de um middleware brasileiro, quando houver material suficiente para construir um estudo de caso.

REFERÊNCIAS BIBLIOGRÁFICAS

ARIB STD-B23; Application Execution Engine Platform For Digital Broad Casting, Association of Radio Industries and Businesses (ARIB), 2004.

ARQUITETURA DE REFERÊNCIA; Arquitetura de Referência – Sistema Brasileiro de Televisão Digital Terrestre - SBTVD, FUNTTEL , 2006.

ASAMI H., SASAKI M.; Outline of ISDB Systems. Proceedings of the IEEE, vol. 94, nº 1, 2006, pp. 248-250.

BARBOSA, A. M. J. P., VALENTE, S. A. L. P.; Set-Top Boxes para Televisão Digital, Faculdade de Engenharia da Universidade do Porto, 2002.

COLLINS, G. W; Fundamentals of Digital Television Transmission. John Wiley & Sons, 1ª edição, 2001.

CRINON, R. J., BHAT, D., CATAPANO, D., THOMAS, G., LOO, J. T. V., BANG, G.; Data Broadcasting and Interactive Television. Proceedings of the IEEE, vol. 94, nº 1, 2006, pp 102-118.

DECRETO 4901; Decreto nº 4.901, de 26 de Novembro de 2003. "Institui o Sistema Brasileiro de Televisão Digital - SBTVD, e dá outras providências", Presidência da República, Casa Civil – Diário Oficial da República Federativa do Brasil, Brasília, 27 de Novembro de 2003.

DECRETO 5820; Decreto nº 5.820, de 29 de junho de 2006. “Dispõe sobre a Implantação do SBTVD-T (Sistema Brasileiro de Televisão Digital – Terrestre), e dá outras providências”, Presidência da República, Casa Civil - Diário Oficial da República Federativa do Brasil, Brasília, 30 de junho 2006.

HORI A., DEWA Y.; Japanese Datacasting Coding Scheme BML. Proceedings of the IEEE, vol. 94, nº 1, 2006, pp. 312-317.

JAVA TV; Java TV API 1.1 (JSR-927) Specification, disponível em http://java.sun.com/javame/reference/apis/jsr927/, acessado em 22 de Janeiro de 2009.

JONES, G. A., DEFILIPPIS, J. M., HOFFMANN, H., WILLIAMS, E. A.; Digital Television Station and Network Implementation. Proceedings of the IEEE, vol. 94, nº 1, 2006, pp. 22-36.

MELO, P. R. S., Rios, E. C. S. D., Gutierrez, R. M. V.; TV DIGITAL: DESAFIO OU OPORTUNIDADE, Rio de Janeiro, BNDES, 2000.

MHP WHITE PAPER; Multimedia Home Platform White Paper, disponível em http://www.dvb.org/documents/white-papers/WP02%20(MHP).pdf, acessado em 20 de Novembro de 007.

MHP-INTERACTIVE; Your source for MHP, OCAP, ACAP, and JavaTV information, disponível em http://www.mhp-interactive.org/, acessado em 12 de Dezembro de 2008.

MORRIS, S.; SMITH-CHAIGNEAU, A.; Interactive TV Standards – A Guide to MHP, OCAP and JavaTV; Focal Press. ISBN: 0-240-80666-2, 2005.

MOURA, A. P.; TV Digital: Convergência e Perspectiva. Dissertação (Mestrado), Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP), Bauru, Brasil, 2006.

MPEG; Página do Moving Picture Experts Group. Disponível em http://www.chiariglione.org/mpeg/, acessado em 20 de Setembro de 2007.

NBR 15601; ABNT NBR 15601 - Televisão digital terrestre - Sistema de transmissão, Associação Brasileira de Normas Técnicas (ABNT), 2008.

NBR 15602-1; ABNT NBR 15602-1 - Televisão digital terrestre - Codificação de vídeo, áudio e multiplexação - Parte 1: Codificação de vídeo, Associação Brasileira de Normas Técnicas (ABNT), 2008.

NBR 15602-2; ABNT NBR 15602-2 - Televisão digital terrestre - Codificação de vídeo, áudio e multiplexação - Parte 2: Codificação de áudio, Associação Brasileira de Normas Técnicas (ABNT), 2008.

NBR 15602-3; ABNT NBR 15602-3 - Televisão digital terrestre - Codificação de vídeo, áudio e multiplexação - Parte 3: Sistemas de multiplexação de sinais, Associação Brasileira de Normas Técnicas (ABNT), 2008.

NBR 15603-1; ABNT NBR 15603-1 - Televisão digital terrestre — Multiplexação e serviços de informação (SI) Parte 1: SI do sistema de radiodifusão, Associação Brasileira de Normas Técnicas (ABNT), 2008.

NBR 15603-2; ABNT NBR 15603-2 - Televisão digital terrestre — Multiplexação e serviços de informação (SI) Parte 2: Estrutura de dados e definições da informação básica de SI, Associação Brasileira de Normas Técnicas (ABNT), 2008.

NBR 15603-3; ABNT NBR 15603-3 - Televisão digital terrestre — Multiplexação e serviços de informação (SI) Parte 3: Sintaxes e definições de informação estendida do SI, Associação Brasileira de Normas Técnicas (ABNT), 2008.

NBR 15604; ABNT NBR 15604 - Televisão digital terrestre — Receptores, Associação Brasileira de Normas Técnicas (ABNT), 2008.

NBR 15606-1; ABNT NBR 15606-1 - Televisão digital terrestre — Codificação de dados e especificações de transmissão para radiodifusão digital Parte 1: Codificação de dados, Associação Brasileira de Normas Técnicas (ABNT), 2008.

NBR 15606-2; ABNT NBR 15606-2 - 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, Associação Brasileira de Normas Técnicas (ABNT), 2008.

NBR 15606-3; ABNT NBR 15606-3 - Televisão digital terrestre — Codificação de dados e especificações de transmissão para radiodifusão digital Parte 3: Especificação de transmissão de dados, Associação Brasileira de Normas Técnicas (ABNT), 2008.

NBR 15606-5; ABNT NBR 15606-5 - 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, Associação Brasileira de Normas Técnicas (ABNT), 2008.

NBR 15607-1; ABNT NBR 15607-1 - Televisão digital terrestre – Canal de interatividade Parte 1: Protocolos, interfaces físicas e interfaces de software, Associação Brasileira de Normas Técnicas (ABNT), 2008.

PAKARINEN, T.; HAGSTRÖM, N.; KOISTILA, P.; BJÖRKQVIST, J.; NYKÄNEN, P.; SAIKAMÄKI, A.; A Guide to the OpenMHP environment, ArviD-publications 2004.

PARKS ASSOCIATES; Social Media & User-Generated Content, disponível em http://newsroom.parksassociates.com/article_display.cfm?article_id=5140, acessado em 10 de Março de 2009.

PIESING, J.; The DVB Multimedia Home Platform (MHP) and Related Specifications. Proceedings of the IEEE, vol. 94, nº 1, 2006, pp.237-247.

PORTARIA INTERMINISTERIAL 237; Portaria Interministerial nº 237, de 29 de Dezembro de 2008. “Processo Produtivo Básico para os produto Terminal Portátil de

Telefonia Celular", Ministério de Estado, Ministério do Desenvolvimento, Indústria e Comércio Exterior e Ministério da Ciência e Tecnologia - Diário Oficial da República Federativa do Brasil, 30 de Dezembro de 2008.

REIMERS, U. H.; DVB - The Family of International Standards for Digital Video Broadcasting. Proceedings of the IEEE, vol. 94, nº 1, 2006, pp.173-182.

SAKURAI, M.; Digital Television Receiver for ISDB. Proceedings of the IEEE, vol. 94, nº 1, 2006, pp.323-326.

SÂMIA, Â. P. M., Mendes, L. L., GUIMARÃES, D. A., Fasolo, S. A.; Digital TV Systems and Standards, I International Workshop on Telecommunications: Proceedings of the Conference, Santa Rita do Sapucaí, FINATEL, 2004.

SIMPSON W., GREENFIELD H.; IPTV and Internet Video, National Association of Broadcasters, Focal Press – Elsevier, 1ª edição, 2007.

SOARES, L. F. G.; RODRIGUES, R. F.; MORENO, M. F.; Ginga-NCL: the Declarative Environment of the Brazilian Digital TV System, Journal of the Brazilian Computer Society, 2007, pp.37-46.

SOL, A. PINTO, P. S.; Digital TV The Software Components, 20th Brazilian Symposium on Computer Graphics and Image Processing (Sibgrapi), 2007.

SOUZA FILHO, G. L., LEITE, L. E. C., BATISTA, C. E. C. F.; Ginga-J: The Procedural Middleware for the Brazilian Digital TV System. Journal of the Brazilian

Computer Society. No. 4, Vol. 13. p.47-56. ISSN: 0104-6500. Porto Alegre, RS, 2007, pp.47-56.

SRIVASTAVA, H. O.; Interactive TV Technology and Markets, Artech House; ISBN 1-58053-321-3, 2002.

UEHARA, M.; Application of MPEG-2 Systems to Terrestrial ISDB (ISDB-T). Proceedings of the IEEE, vol. 94, nº 1, 2006, pp.261-268.

WU Y., HIRAKAWA S., REIMERS U. H., WHITAKER J.; Overview of Digital Television Development Worldwide. Proceedings of the IEEE, vol. 94, nº 1, 2006, pp.8-21.

YAHOO; Yahoo! Connected TV: The best of the Internet on your TV, disponível em http://connectedtv.yahoo.com/, acessado em 13 de Fevereiro de 2009.

YOSHIMURA T.; Conditional Access System for Digital Broadcasting in Japan. Proceedings of the IEEE, vol. 94, nº 1, 2006, pp.318-322.

ZUFFO, M. K.; TV Digital Aberta no Brasil - Políticas Estruturais para um Modelo Nacional, Escola Politécnica, USP, São Paulo, 2006.