PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 03/08

TV Digital Para Dispositivos Portáteis - Middlewares

Vitor Medina Cruz
Marcio Ferreira Moreno
Luiz Fernando Gomes Soares

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 03/08 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena Janeiro/2008

TV Digital Para Dispositivos Portáteis- Middlewares.

Vitor Medina Cruz, Marcio Ferreira Moreno, Luiz Fernando Gomes Soares

{vcruz, mfmoreno, lfgs}@inf.puc-rio.br

Abstract. *This document is a study about the Digital Television technologies, mainly about the available declarative middlewares, and its use on portable devices.*

Keywords: portable, middleware, Ginga, NCL, DTV, IPTV.

Resumo. Este documento é um estudo sobre as tecnologias de TV Digital, principalmente no que diz respeito aos *middlewares* declarativos, e a sua aplicação em dispositivos portáveis.

Palavras-chave: portável, *middleware*, Ginga, NCL, TVD, IPTV.

Sumário

1 Introdução

O uso da TV Digital cresce no mundo e traz diversas vantagens tanto para os provedores quanto para os telespectadores. No Brasil, em especial, ela também representa uma forma de promover a inclusão digital. Oferecer TV Digital nos dispositivos portáteis é uma nova forma de se prover TV e é o foco deste estudo.

Uma vez tendo a infra-estrutura necessária para tornar disponível a TV Digital convencional, bem como tendo o seu conteúdo pronto, oferecer esse serviço para os dispositivos portáteis e atender a esse nicho é uma boa vantagem técnica e comercial. Entretanto, os dispositivos portáteis possuem características diferentes daquelas encontradas nos aparelhos convencionais usados na TV Digital. Essas diferenças fazem com que algumas vezes o próprio conteúdo a ser oferecido necessite ser moldado.

Existem padrões que definem a infra-estrutura necessária para a disponibilização da TV Digital e como os seus conteúdos devem ser criados. Nesses padrões são descritas as formas e os formatos de transmissão, as codificações e, entre outras coisas, os *middlewares* de TV Digital. O conceito de *middleware* será explicado em capítulos subseqüentes, mas é nele que são definidas as linguagens usadas na criação do conteúdo digital. Para permitir a transmissão da TV Digital para dispositivos portáteis, tanto os padrões quanto os seus *middlewares* precisam observar e se adequar às características diferenciadas desses aparelhos. Além disso, é importante oferecer uma forma simples de adaptação ao conteúdo existente, de forma a possibilitar sua exibição em um dispositivo portátil.

Neste estudo será feita uma análise da aplicação da TV Digital nos dispositivos portáteis, principalmente no que diz respeito aos principais *middlewares* declarativos usados. O objetivo é compará-los com o *middleware* declarativo Ginga-NCL (SBTVD, 2007), que também será abordado.

Um outro ponto abordado trata das plataformas de desenvolvimento encontradas para dispositivos portáteis. O objetivo é determinar aquela, ou aquelas, que melhor se adequa ao desenvolvimento de um *middleware* de TV Digital.

A organização deste documento é a seguinte: o Capítulo 2 discorre sobre a TV Digital, suas vantagens e tecnologias envolvidas. O Capítulo 3 apresenta um conjunto de características comuns aos dispositivos portáteis. No Capítulo 4, são descritos os principais sistemas operacionais embarcados nesses dispositivos. O Capítulo 5 faz um estudo sobre as principais plataformas de desenvolvimento dos sistemas operacionais avaliados como adequados para o objetivo proposto. O Capítulo 6 realiza uma análise dos principais *middlewares* de TV Digital. Por fim, o Capítulo 7 trás a conclusão deste estudo.

2 A TV Digital

Em um sistema de televisão existem, de forma simplificada, dois conjuntos de atores: os provedores de conteúdo e os telespectadores. Os primeiros desenvolvem os programas televisivos, ou conteúdos, que deverão ser entregues aos telespectadores. Apesar desse conteúdo já ser produzido digitalmente, ele ainda é, em muitos lugares, transmitido de forma analógica dos provedores para os telespectadores. A mudança da transmissão analógica para digital já foi feita em alguns países, e está sendo

implementada no Brasil. Isso muda a forma como a TV é feita e abre novos horizontes, tanto para os provedores quanto para os telespectadores.

A TV Digital oferece diversas vantagens. A primeira delas, melhor percebida pelo telespectador, é a qualidade da imagem. Dada a natureza discreta do conteúdo digital, é possível realizar uma recuperação muito precisa a partir do sinal recebido. Sendo assim, quando esse procedimento é feito corretamente, o conteúdo recebido é o mais próximo possível daquele que foi transmitido. Isso já não acontece quando o conteúdo é analógico, ou seja, de natureza contínua. Nesse caso, imprecisões que ocorrem durante a transmissão geram falhas que causam perdas na qualidade da imagem. De forma similar, a qualidade do áudio também sofre melhoras e mais canais de áudio também podem ser oferecidos. Isso possibilita o uso de tecnologias, como o *surround*, ou a escolha de línguas diferentes durante a exibição de um determinado programa. Uma outra vantagem da TV Digital é permitir que o conteúdo transmitido seja facilmente armazenado e com a mesma qualidade do sinal transmitido. Assim, o telespectador poderia, por exemplo, pausar ou solicitar um *replay*, entre outras funções de PVR (*Personal Video Recorder*) (Weber e Tom , 2006) , mesmo em transmissões feitas ao vivo.

Os provedores de conteúdo, por outro lado, encontram uma facilidade maior no controle e gerenciamento da informação, devido à sua natureza discreta. É mais fácil, também, compor o conteúdo e adicionar, por exemplo, propagandas. O modelo de propaganda também tende a mudar, uma vez que elas poderão ser mais personalizadas e lançadas em momentos mais oportunos, dependendo até da escolha do telespectador. Uma outra vantagem é o uso otimizado das redes de comunicação. No Brasil, por exemplo, um canal de *broadcast* é usado para a transmissão analógica de um único programa. Com a TV Digital, o mesmo canal será capaz de transmitir, além do áudio e dos dados, vários canais de vídeo com qualidade padrão (SDTV), ou até mesmo de alta qualidade (HDTV).

Por fim, essa nova forma de se prover TV é, além de tudo, uma maneira de oferecer conteúdo personalizado e interatividade para os telespectadores.

A TV Digital pode ser provida trazendo-se tecnologias computacionais para o âmbito da TV ou levando as tecnologias de TV para o ambiente computacional. Um exemplo do primeiro caso é o que tem acontecido atualmente com a TV aberta do Brasil. Ao incorporar as tecnologias digitais nesse modelo, estaremos promovendo uma distribuição de TV Digital que permite interatividade e a personalização do conteúdo. Entretanto, a questão da oferta do conteúdo, que corresponde ao vídeo principal de um programa, sob demanda não é facilmente realizada. Isso ocorre por causa da natureza do serviço oferecido, onde a programação é transmitida em *broadcast* para todos receptores. É preciso ter uma nova infra-estrutura que ofereça um canal de retorno que possibilite ao telespectador se comunicar com o provedor de conteúdo. Assim, a oferta de conteúdo individualizado seria possível. A personalização do conteúdo permite também que uma emissora conheça melhor os seus consumidores.

A IPTV é um exemplo do caso onde a tecnologia de TV é levada para o ambiente computacional. A sigla IPTV quer dizer TV *Over* IP, e significa que a TV é distribuída sobre o protocolo da Internet, o IP. Isso não quer dizer que a TV é oferecida pela Internet, método que é chamado de Internet TV. Na verdade, os serviços de IPTV existentes são oferecidos em redes fechadas, parecido com o que é feito nas TVs por assinatura. Essas redes possuem tanto um canal de *downstream*, por onde as informações são recebidas, quanto um de *upstream*, por onde informações são enviadas para o provedor. A IPTV encontra-se, portanto, mais próximo do modelo da Internet, e

isso permite que serviços, como o vídeo sob demanda, sejam mais facilmente oferecidos. Os serviços comumente encontrados para IPTV são:

- VoD (*Video on Demand*), onde o consumidor tem acesso a um acervo de vídeos que podem ser escolhidos e apresentados sob sua demanda, na hora em que lhe for mais conveniente.
- PVR (*Personal Video Recorder*), que permite realizar operações de vídeo gravado sobre *streams*, como pausa, *fast-forward* e *rewind*.
- *Broadcast* convencional, similar ao que é feito pela TV convencional, onde o telespectador escolhe o canal que deseja assistir.
- ESG (*Electronic Service Guide*), um serviço de guia para conteúdos digitais. Com ele, o telespectador pode navegar pelo conjunto de programações e serviços oferecidos e escolher o que mais lhe agradar. Ele pode selecionar um canal convencional ou resolver comprar um vídeo pré-armazenado para assistir.

A IPTV oferece mais recursos por estar mais próxima do modelo da Internet, mas existem algumas desvantagens a serem ressaltadas. A primeira é a complexidade da rede envolvida. Sendo baseada no protocolo IP, serviços bidirecionais de *unicast* são providos mais naturalmente. Entretanto, o serviço de *Broadcast* de conteúdo, bem como o de VoD, não são facilmente oferecidos pela rede da IPTV. Para tanto, é preciso que a rede seja capaz de oferecer facilidades de *multicast*, onde a transmissão é feita de um para n endereços IPs registrados em um grupo, chamado de grupo *multicast*. Assim, quando o telespectador escolhe um canal para assistir, o seu IP é adicionado ao grupo *multicast* daquele canal e, a partir de então, o conteúdo passa a ser recebido. Tudo isso aumenta a complexidade da rede (Weber e Tom , 2006: 38).

Um outro problema da IPTV é o consumo de banda nos serviços sob demanda, como o VoD. Mecanismos devem ser criados para que esse consumo seja reduzido e o serviço se torne viável. Serviços de VoD já prevêem isso e possuem formas de lidar com o problema. Uma tecnologia interessante a ser utilizada nesse caso é o P2P. Nela, terminais encontrados nas casas dos telespectadores podem ser usados como *seeds* para o conteúdo (Weber e Tom , 2006: 66-68). Assim, o provedor pode enviar um vídeo para um conjunto de terminais e esses, por sua vez, transmitirem para outros usando a técnica P2P, liberando a banda do provedor.

Uma outra questão que merece atenção é a da proteção de conteúdo contra cópia não autorizada, que está relacionada tanto a IPTV quanto a TV Digital em geral. Conteúdos digitais são mais facilmente copiados e replicados, com a mesma qualidade original, do que conteúdos analógicos. Isso trás problemas de direitos autorais e é um dos principais fatores que preocupam os provedores de conteúdo na transição do sistema analógico para o digital. Um ambiente de TV Digital, principalmente um de IPTV ou Internet TV, pode oferecer algum grau de segurança ao conteúdo transmitido. Para atingir esse objetivo, foram criados os sistemas de DRM (*Digital Rights Management*). Esses sistemas envolvem criptografia de conteúdo com controle de acesso por chaves, onde regras são definidas a fim de determinar quem, e sob quais circunstâncias, pode ter uma chave que acesse um determinado conteúdo. Existem ainda outras regras que definem, por exemplo, se um conteúdo pode ser replicado, sob quais circunstâncias e por quantas vezes. Um sistema DRM deve ser suficientemente abstrato para se adequar a situações diversas que exijam diferentes regras.

Existem mais no que tange a questão da segurança do conteúdo. É preciso que a indústria dos dispositivos receptores esteja a par dessas técnicas e desenvolvam o

hardware necessário para torná-las viáveis. Um dispositivo precisa manter as informações seguras e proteger as chaves de segurança. É preciso, também, impedir que cópias não autorizadas sejam feitas. Isso envolve técnicas que impedem que um conteúdo seja copiado para um CD, DVD ou até mesmo para um VCR. Técnicas desse tipo são: HDCP (*High-bandwidth Digital Content Protection*), DTCP (*Digital Transmission Content Protection*), CPRM (*Content Protection for Recordable Media*), VCPS (*Video Content Protection System*), *Macrovision system*, CGSM-A (*Copy Generation Management System-Analog*) e *Watermarks* (Weber e Tom , 2006: 233-240). A falta de segurança de conteúdo é um problema em qualquer sistema de TV digital, sendo mais acentuada no caso da IPTV e mais ainda na Internet TV.

À medida que esses problemas vão sendo solucionados, a TV Digital vai sendo mais e mais utilizada. Nos EUA, a TV já era provida via cabo, e, por isso, a realidade da IPTV está mais próxima nesse país. No Brasil, em especial, a TV Digital será oferecida em âmbito nacional por radiodifusão. A IPTV ficará mais atrativa quando mais recursos estiverem disponíveis por um preço mais barato. Provavelmente, em um futuro próximo, a TV estará prontamente disponível pela Internet.

Muito embora sendo considerável o avanço da TV Digital, atualmente a sua realidade ainda não alcançou, em sua plenitude, o mundo dos dispositivos portáteis. A maioria dos esforços na implementação desse recurso para esse novo contexto iniciou-se há pouco tempo. Um bom exemplo é encontrado no Japão que, apesar de bem avançado em termos de TV Digital, só deu início à migração desse serviço para os dispositivos portáteis em meados de 2006. Assim como no início do uso da TV Digital, essa migração deve trazer novos horizontes, tanto para os provedores quanto para os telespectadores.

3 Características dos Dispositivos Portáteis

Os dispositivos portáteis possuem características específicas que precisam ser observadas quando do desenvolvimento de aplicativos. Entre elas pode-se citar:

- Uso de bateria, que exige um consumo moderado de energia. Aplicações nesse escopo podem ser sumariamente terminadas, e precisam se adequar ao fato. Um outro ponto é que o consumo de energia gerado pela aplicação precisa ser baixo.
- Limite de processamento e de memória.
- Mobilidade, com processo de *handoff*.
- Tamanho de tela pequeno.
- A conversa, no caso do celular, como funcionalidade principal.. Qualquer outra atividade fica em segundo plano.
- Teclado limitado.
- Limite maior de banda.

Qualquer *middleware* direcionado a esses dispositivos precisa levar em consideração essas características. De forma semelhante, os sistemas operacionais desenvolvidos para essa plataforma precisam ser diferentes dos convencionais, que, de forma geral, não atendem aos requisitos acima descritos. Os próximos capítulos se destinam ao estudo desses dois elementos.

4 Sistemas Operacionais

Este capítulo apresenta os principais sistemas operacionais existentes para as plataformas de dispositivos embarcados, sendo os portáteis os mais relevantes para o caso específico deste estudo. O objetivo é verificar quais deles são adequados para receber uma implementação de um *middleware* de TV Digital.

Segundo (Canalys Reseach, 2007), em pesquisa realizada no final de 2006, o sistema operacional mais usado atualmente em dispositivos portáteis na Europa é o Symbian, seguido da versão mobile do sistema Windows e, em seguida, pelo sistema operacional dos aparelhos BlackBerrys vendidos pela RIM (*Research In Motion*). Outros sistemas importantes são o Linux e o PalmOS. O primeiro porque a sua participação no mercado, de acordo com a (Canalys Reseach, 2007), tem crescido e por ser *open-source*, e o segundo por ser amplamente conhecido.

4.1 Symbian

O sistema operacional Symbian vem sendo desenvolvido pela empresa de mesmo nome, a Symbian Ltd, fundada em 1993, e tem a Nokia, Ericsson, Siemens AG, Panasonic, Samsung e a Sony Ericsson como suas acionistas; todas indústrias de dispositivos portáteis. O desenvolvimento do sistema foi feito especificamente para esses aparelhos. Apesar de estar presente, na grande maioria das vezes, em dispositivos de grande porte, como os *smartphones*, o Symbian OS pode ser usado em aparelhos mais simples. Um exemplo é o Raku-Raku PHONE *Simple*, encontrado em (Symbian Limited) na seção "*Symbian Phones*".

Dentre as principais vantagens do SymbianOS em relação aos outros sistemas destaca-se o seu foco. Desde a sua concepção, o sistema em questão já previa as características diferenciais encontradas nos dispositivos portáteis. O principal foco no desenvolvimento do Symbian foi criar um gerenciamento que evitasse o uso excessivo e desnecessário de memória ou a ocorrência de vazamentos, o que ocorre quando um espaço de memória não mais utilizado continua alocado. Um outro fator levado em consideração é o consumo de energia, que, segundo (Symbian OS - Wikipedia, the free encyclopedia), é tratado com o uso de um recurso, denominado *Active Object*. Aplicações Symbian podem ser desenvolvidas em Symbian C++, Java, Python, .NET, Ruby, Perl, Open Programming Language (OPL) e Adobe Flash For Development (Symbian Limited, 2007b). Symbian C++ e Java, no entanto, são as que possuem maior suporte da comunidade Symbian, com ferramentas de desenvolvimento, exemplos, APIs adicionais e informações. Isso se dá pelo fato da primeira ser a linguagem nativa do sistema e pela difusão da segunda no mercado.

Outra vantagem interessante do Symbian é a sua experiência no mercado, além de estar associado às maiores industrias de celulares do mundo. O sistema em questão cobre a maior parte do mercado, o que, muitas vezes, gera um interesse maior na criação de aplicativos para a sua plataforma. Contribuem para esse aspecto, também, o fato da sua API de desenvolvimento ser gratuita, a existência de uma comunidade ativa de programadores disponível na (Symbian Developer Network) e a disponibilidade de muitas ferramentas úteis para o desenvolvimento, sendo algumas gratuitas.

No que tange a TV Digital, Symbian possui uma versão, a 9.5, que oferece suporte a esse tipo de aplicação, aceitando, atualmente, dois padrões de modulação/transmissão de TV Digital, o DVB-H e o ISDB-T (Symbian Limited).

Uma desvantagem que poderia ser destacada vem do seu próprio sucesso. Por ser muito popular e oferecer ferramentas poderosas de desenvolvimento, o sistema acaba atraindo aqueles que desenvolvem programas maliciosos. Assim, algumas empresas preferem uma plataforma mais fechada e, conseqüentemente, mais segura. Atualmente, um dos esforços é no sentido de adicionar mais segurança ao Symbian OS. Um outro ponto que pode ser considerado como uma desvantagem são os custos associados às licenças necessárias para o uso do sistema em questão. Segundo (Symbian Limited, 2007a), o custo por unidade embarcada com o Symbian OS é de, aproximadamente, US$4,4. Esse é um motivo pelo qual esse sistema operacional é mais encontrado em dispositivos de grande porte, pois esse custo extra inviabiliza seu uso em dispositivos mais simples, que possuem apelo pelo preço. O desenvolvimento de aplicações também pode ser custoso. A ferramenta usada para esse objetivo mais conhecida é o Carbide C++. Existem quatro licenças para esse produto, uma delas gratuita, que podem ser usadas para desenvolver aplicações comerciais. A versão gratuita é muito limitada e, para desenvolver aplicações mais complexas, pode ser necessário adquirir uma versão com mais recursos, sendo que a mais barata custa 299 € (Forum Nokia - Carbide Development Tools for Symbian OS C++, 2007).

4.2 Windows

O Windows possui duas versões que atendem ao mercado dos dispositivos embarcados de pequeno porte, o Windows CE e o Mobile.

O Windows CE é um sistema componentizado do tipo *Hard Real Time* (Hall, 2007). A sua versão 6.0 possui cerca de setecentos componentes que podem ser usados para montar um sistema operacional que atenda aos requisitos específicos de um ambiente. Por exemplo, uma empresa poderia construir um sistema para um celular com poucos recursos e um outro para um handheld com recursos gráficos avançados. O Windows CE oferece flexibilidade para que os dois casos em questão sejam contemplados. Isso significa que cada dispositivo com um sistema Windows CE instalado pode possuir um conjunto de tecnologias diferentes. A escolha da componentização realizada implica no tamanho da "imagem do sistema", ou seja, ela pode ser maior ou menor dependendo dessa escolha.

O Windows Mobile é baseado no Windows CE, onde a escolha dos componentes foi feita de forma a atender especificamente o mercado de dispositivos portáteis/móveis. Nele é oferecida uma API genérica, ou seja, uma aplicação desenvolvida para esse sistema funciona em qualquer dispositivo que o Windows Mobile estiver instalado (Hall, 2007).

Existe ainda o Windows XP Embedded (Microsoft Windows XP Embedded: Home page). No entanto, essa versão não se destina aos dispositivos portáteis, foco deste estudo.

Por ser um sistema muito conhecido e amplamente utilizado em *desktops*, o Windows é bem recebido pelos usuários dos dispositivos portáteis. Além disso, as ferramentas utilizadas para o desenvolvimento de aplicações são as mesmas da sua versão *desktop*, como o Visual Studio ou o .NET. Existe, inclusive, uma ferramenta totalmente gratuita, o Microsoft eMbedded Visual C++, que foi concebida especificamente para o desenvolvimento visando dispositivos embarcados. Entretanto, o pacote de desenvolvimento recomendado pela Microsoft para o Windows Mobile é o Windows Mobile 6 Developer Resource Kit. Nesse pacote recomendado, encontra-se o Visual Studio 5, ferramenta de desenvolvimento da Microsoft, que deve ser comprada.

É aqui que começam os problemas de custos atrelados a plataforma Windows. O custo de uma licença do Visual Studio é de US$299. A licença do Windows CE na sua versão mais básica é de US$ 3 por dispositivos. Os valores dos custos do Windows Mobile não são divulgados, sendo assim, o valor do Windows CE serve de base. Além desses problemas de custo, os ambientes portáteis com o sistema operacional Windows instalado tendem a terem os mesmos problemas encontrados na sua versão *desktop*, como a necessidade constante de reinicialização, vazamento e uso desnecessário de memória.

4.3 RIM

BlackBerry é o Produto vendido pela RIM. O seu sistema operacional é proprietário e é feito dedicadamente para os *handhelds* e outros dispositivos portáteis que a RIM produz.

A RIM não disponibiliza o seu SO para que outras empresas utilizem, sendo assim, apenas os hardwares fabricado pela própria RIM o possuem. O desenvolvimento nesse sistema pode ser feito na linguagem Java com o uso de um pacote de desenvolvimento disponibilizado pela empresa em questão. Uma outra opção é utilizar o BlackBerry MDS Studio, que é uma aplicação de desenvolvimento visual. Ambas as opções são gratuitas.

Os aparelhos da BlackBerry não possuem boa interação com outros tipos de dispositivos. Algumas aplicações, como e-mail, por exemplo, funcionam apenas com um conjunto específico de dispositivos de outras marcas.

4.4 Linux

Devido ao crescimento do mercado dos dispositivos portáteis, esforços para a disponibilização de versões do sistema operacional Linux que atendam a esse mercado têm sido feitos. O resultado é o surgimento de uma grande quantidade de distribuições comerciais, gratuitas e até mesmo com o código fonte aberto.

O Linux é um sistema flexível que oferece uma grande diversidade de métodos de instalação e, por conta disso, pode funcionar em muitos sistemas heterogêneos. Uma grande vantagem do seu uso nos dispositivos portáteis é que, se for configurado corretamente, o consumo de recursos pode ser reduzido em muito. O porte de um código da plataforma fixa para a portátil é, aparentemente, direto. Ou seja, uma aplicação desenvolvida para o ambiente fixo deve poder ser portada diretamente para uma distribuição Linux portátil que possua todas as bibliotecas requeridas. Outra vantagem é que o desenvolvimento de aplicações pode ser feito sem custos, uma vez que as muitas ferramentas gratuitas disponíveis no ambiente fixo podem ser usadas.

Apesar dessas vantagens, a configuração do sistema é, freqüentemente, uma tarefa bastante complicada, e encontrar *drivers* para cada hardware específico dos dispositivos portáteis pode não ser fácil. Além disso, as distribuições atuais não são completamente compatíveis entre si, o que significa que uma aplicação pode funcionar em uma delas e em outra não, como pode ser visto em (Small Biz Resource) e em (Mace, 2006). Uma outra questão é que distribuições mais estáveis são pagas. Entretanto, uma versão Open Source (Mobile Linux Internet Project, 2007) desenvolvida para dispositivos Intel já parece estar bem estável. Outros projetos Open Source vem surgindo, como da Ubuntu (Ubuntu Mobile and Embedded Edition, 2007).

Com o crescimento do uso do Linux em ambientes portáteis, é possível que mais distribuições gratuitas apareçam.

Algumas distribuições Linux Mobile são: GPE Phone Edition, OSDL Mobile Linux, MontaVista, A-la-mobile, Mozi dentre outros, que podem ser encontrados em (Heuser, 2007).

A comunidade Linux está, atualmente, realizando esforços para acelerar a adoção do seu sistema operacional no mundo dos dispositivos portáteis, uma vez que a sua participação ainda não é muito grande.

4.5 PalmOS

O PalmOS tem tentado se manter no mercado, muito embora não tenha conseguido se sobressair em relação aos seus concorrentes (Palm OS, 2007).

As últimas versões desse sistema foram a PalmOS Garnet e a PalmOS Cobalt, sendo que a última nem chegou a ser lançada comercialmente. A primeira versão foi feita para dispositivos com menos recursos de *hardware*, enquanto que a segunda foi desenvolvida para aparelhos mais modernos, como os *smartphones* (Palm OS, 2007).

O desenvolvimento de ambas as versões foi paralisado e um novo desenvolvimemto começou a ser realizado, o da ACCESS Linux Platform (ACL). Ele será baseado em um *kernel* Linux e foi lançado para pré-venda em fevereiro de 2007. Essa é uma nova tentativa de alavancar o sistema e fazer frente à concorrência.

Para ilustrar o problema que o PalmOS tem passado, até mesmo alguns aparelhos da Palm não tem usado a sua variante SO e, ao invés disso, utilizam o Windows Mobile, como é o caso do `Treo 700w e do Treo750`. A expectativa agora fica no lançamento do ACL e da sua aceitação.

4.6 Análise Comparativa

Para este estudo em questão, são interessantes plataformas abertas, que sejam estáveis, de baixo custo, em uso hoje e que possam representar uma opção viável de mercado no futuro.

O PalmOS está tentando se alavancar no mercado e não possui nenhuma versão sendo relevantemente utilizada. O BlackBerry por outro lado, possui plataforma fechada, tem poucas opções de desenvolvimento e, em comparação ao Symbian e ao Windows Mobile, possui uma participação pequena no mercado, sem tendências aparentes de crescimento. Por esses motivos, esses dois sistemas foram descartados.

No caso da plataforma Windows, o Windows Mobile é o mais interessante para o caso de estudo em questão, pois foi feito especificamente para dispositivos portáteis. Windows Mobile dispõe das ferramentas típicas do seu ambiente *desktop* e uma versão gratuita de uma ferramenta de desenvolvimento específica para os dispositivos portáteis. Comparando com o Symbian OS, esse último se destaca em relação ao suporte oferecido. Symbian OS possui uma série de ferramentas de desenvolvimento, algumas até mesmo gratuitas, e um suporte bem amplo. No Windows, o suporte parece ser menos especializado pelo fato desse sistema não ser tão amplamente utilizado como o outro. Em termos de custo, os dois se quivalem, pois, em ambos, ferramentas de desenvolvimento com mais recursos são muito caras e licenças para o uso dos sistemas são cobradas. Um outro ponto, e esse deafavorecendo o Symbian OS, são os vírus. Por ser mais disseminado, Symbian OS tem sido um alvo maior de

códigos maliciosos. Tem-se tentado contornar esse problema com o lançamento de versões mais seguras da plataforma. Por outro lado, o Windows é bem conhecido por possuir vírus direcionados à sua plataforma, e é de se esperar que o seu crescimento no mercado atraia os desenvolvedores desse tipo de software. O maior problema do Windows Mobile é que ele possui os problemas típicos da sua versão *desktop*, como os vazamentos de memória e as reinicializações do sistema. Isso complica a questão do seu uso na implementação de um *middleware* de TV Digital, que precisa ser robusta. A conclusão é que o Windows Mobile, ainda que com as suas qualidades, não alcança o nível do Symbian OS. Apesar dos custos atrelados a sua plataforma, o Symbian OS representa uma boa opção para o desenvolvimento de um *middleware* de TV Digital. Ainda mais porque foi o único dos sistemas estudados a ter lançada uma versão que aceita recepção de TV Digital.

No caso do Linux Mobile, considerando o seu crescimento, espera-se que mais distribuições gratuitas e estáveis sejam lançadas no futuro. Um bom indicador de que a plataforma Linux venha a manter a sua tendência de crescimento no mundo portátil é o fato do novo SO da Palm ser baseado nela. Pensando nesse futuro e considerando a grande quantidade de ferramentas gratuitas disponíveis, o Linux Mobile ganha na questão do custo em relação aos outros sistemas operacionais. A plataforma em questão tende a ser robusta, otimizada e possui pouquíssimos casos de problemas com vírus. O porte de aplicações do ambiente fixo para o portátil, aparentemente, é direto, o que facilita o desenvolvimento. Dada as projeções para a plataforma Linux Mobile e das suas vantagem inerentes, assume-se, neste estudo, que esse sistema também parece ser adequado para o desenvolvimento de um *middleware* de TV digital.

Sendo assim, esta Seção conclui que o Symbian OS e o Linux Mobile, atualmente, parecem constituir as melhores opções de sistemas operacionais para o objetivo em estudo. Dessa forma, o próximo capítulo será dedicado ao estudo das suas plataformas de desenvolvimento mais relevantes.

5 Plataformas de Desenvolvimento

O objetivo desta Seção é apontar as plataformas Linux e Symbian consideradas mais adequadas para o desenvolvimento de uma implementação de um *middleware* de TV Digital.

No Linux, a plataforma de desenvolvimento utiliza a linguagem C++. Algumas distribuições oferecem ferramentas de desenvolvimento pagas, entretanto, qualquer ferramenta gratuita disponível para o Linux na sua versão fixa pode ser usada. Como pode ser visto em (Marcio 2006), a plataforma Linux e a linguagem C++ são apropriados para o desenvolvimento de um *middleware* de TV Digital.

Dentre as diversas linguagens oferecidas pelo Symbian OS, a Java e o Symbian C++ são as mais usadas.

Java é uma linguagem muito usada pelo fato de ser simples, oferecer portabilidade e estar disponível em uma grande variedade de plataformas de sistemas operacionais. JavaME . (Java ME Technology) é uma plataforma Java concebida especificamente para funcionar em dispositivos portáteis/móveis, e disseminou-se não só no Symbian OS, mas em todo o mundo dos dispositivos portáteis/móveis

Symbian C++ é a linguagem nativa da plataforma do sistema operacional Symbian OS e é baseada na linguagem C++. É muito utilizada na plataforma em questão por oferecer desempenho otimizado e um maior acesso aos recursos do sistema.

Nas próximas três Seções serão feitas analises de ambas as plataformas Symbian citadas, a fim de apontar a mais adequada para o objetivo proposto. As Seções 5.1 e 5.2 apresentam, respectivamente, as arquiteturas das plataformas Symbian OS e JavaME. A Seção 5.3 realiza uma comparação das arquiteturas.

Por fim, a Seção 5.4 tece algumas considerações finais em torno das plataformas do Symbian OS e do Linux.

5.1 Plataforma Symbian OS

Como já mencionado, Symbian C++ é a linguagem de programação nativa usada na plataforma do sistema operacional Symbian.

A arquitetura do sistema segue o conceito de *Subsystems*, ou subsistemas, que são módulos que encapsulam funcionalidades relacionadas, como pode observado na Figura 1.

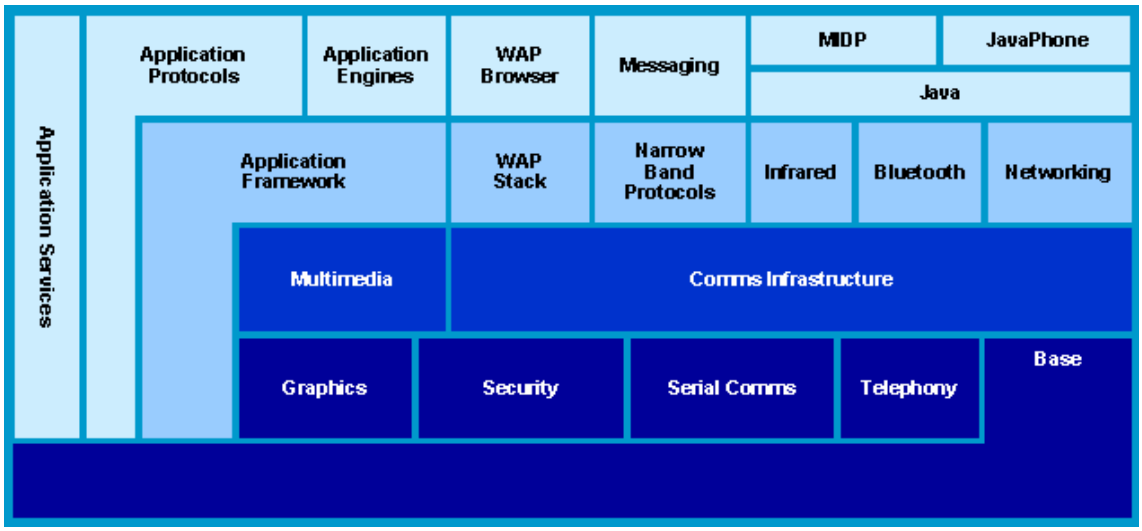

Figura 1: *Subsystems*, retirado de (Symbian Limited, 2005)

Cada subsistema é subdivido em componentes. Sendo um sistema modular, o Symbian OS pode ser moldado de acordo com a escolha desses componentes. Isso significa que dois dispositivos podem ter dois sistemas operacionais completamente diferentes, mesmo sendo baseados na mesma versão do Symbian OS. Entretanto, isso normalmente não acontece, como veremos mais adiante. A escolha dos componentes, obviamente, influencia nas APIs disponíveis para desenvolvimento. Portanto, os desenvolvedores devem ficar atentos ao fato de que a sua aplicação pode não funcionar em determinados dispositivos. Isso vai depender de quais APIs foram utilizadas.

Para facilitar a portabilidade, é feita uma flexibilização desse sistema, onde os seus componentes são controlados e categorizados. Em (Jode e Turfus, 2004) podemos encontrar a seguinte categorização:

- *Common Symbian*: Representa o núcleo do Symbian OS. Todas as APIs referentes aos componentes dessa categoria devem estar disponíveis em todos os dispositivos. O código desenvolvido apenas com essas APIs deve compilar e o

seu binário deve funcionar em qualquer dispositivo. A implementação dessas APIs é feita pela Symbian.

- *Common Replaceable*: Também precisa estar presente em todos os dispositivos, mas, diferentemente da *Common Symbian*, as suas APIs precisam ser implementadas pelo fabricante do dispositivo.
- *Optional Symbian*: São componentes opcionais, implementados pela Symbian. Se um código for desenvolvido com o uso de uma API de um componente opcional, esse deve compilar e o seu binário deve executar corretamente apenas nos aparelhos que possuírem esse componente.
- *Optional Replaceable*: São componentes opcionais da mesma forma que os do *Optional Symbian*. Também são implementados pela Symbian, mas, nesse caso, os fabricantes podem optar por desenvolver implementações próprias desses componentes. Sendo assim, é possível decidir entre usar as APIs Symbian ou desenvolver as suas próprias.

Os componentes que fazem parte de cada categoria podem ser vistos com detalhes em (Symbian OS System Definition - Detailed View). Em (Jode e Turfus, 2004: 6) podemos observar que, na prática, as duas primeiras categorias estão presentes em todos os dispositivos com o Symbian OS instalado, salvo duas exceções específicas. Uma está relacionada à apresentação de Fontes *True Type* e a outra ao suporte de servidores de Telefonia. Na prática, a terceira categoria também é completamente suportada por todos os aparelhos. Apenas a última é que apresenta componentes alterados ou não utilizados. Entretanto, isso não é verdade para maioria dos casos, de acordo com (Jode e Turfus, 2004: 8).

A programação em Symbian C++ apresenta alguns problemas de portabilidade, mas, de forma geral, o código desenvolvido tende a executar corretamente em qualquer dispositivo. Isso é verdade principalmente se as APIs utilizadas se enquadrarem nas categorias *Common Symbian* ou *Common Replaceable*, e até mesmo na *Optional Symbian*. O problema da portabilidade se restringe muito mais à UI (*User Interface*) oferecida pelos dispositivos. Como são muitas, e bem distintas, existe uma coletânea de APIs para cada uma delas. O sítio de desenvolvimento da Symbian, (Symbian Developer Network), oferece um SDK (*Software Development Kit*) diferente para cada UI disponível no mercado. Para atender a mais de um nicho de dispositivos, um desenvolvedor deve adaptar o seu código para cada SDK das diferentes UIs.

Um outro problema de portabilidade é entre as diferentes versões do Symbian OS. Mudanças substanciais do sistema foram feitas a partir da versão 9.1, o que causou uma perda de compatibilidade com versões mais antigas, como a 8.0. Em (Mark Shackman) são detalhados os procedimentos necessários para que se altere um código de uma versão mais antiga a fim de que seja possível compilá-lo em versões 9.1 ou superiores. Ou seja, versões mais recentes mantêm compatibilidade com versões inferiores até a versão 9.1 apenas.

O último problema de portabilidade é que um código Symbian C++ funcionará apenas na plataforma Symbian.

As APIs do sistema em questão foram desenvolvidas de tal forma a otimizar o consumo de processamento e de memória dos dispositivos. Por conta disso, nenhuma biblioteca C++ como, por exemplo, a STL (*Standard Template Library*), é utilizada. Ao invés disso, novas APIs são oferecidas com funcionalidades similares daquelas

especificadas no padrão ISO C++ (ISO/IEC 14882:2002). A diferença, como mencionado, é que elas são especialmente otimizadas para a execução em dispositivos portáteis. Apesar disso, esforços para permitir a portabilidade de código ANSI C (ISO/IEC 9899:1999) e da STL estão sendo realizados.

Como o Symbian C++ é baseado em C++, o programador tem em mãos uma linguagem que oferece total controle sobre a sua aplicação. Sendo assim, programas desenvolvidos em Symbian C++ são altamente otimizados e possuem desempenho elevado. Por outro lado, C++ é uma linguagem complexa, que exige um tempo de desenvolvimento elevado antes que uma aplicação possa ser distribuída. Nesse sentido, Symbian C++ também não é diferente.

Por fim, sendo Symbian C++ uma linguagem nativa, os desenvolvedores podem fazer uso de todos os recursos disponibilizados pelo sistema como, por exemplo, as funções nativas. Isso possibilita que aplicações mais poderosas possam ser desenvolvidas.

5.2 JavaME

JavaME (*Micro* ou *Mobile Edition*) é um subconjunto da J2SE (Java Technology, 2007) e é uma plataforma desenvolvida para atender aos requisitos dos dispositivos portáteis e móveis de pequeno a grande porte. Essa plataforma implementa a linguagem Java e possui uma arquitetura modular de três camadas: *Virtual Machine*, *Configuration* e *Profile*. A primeira camada é a máquina virtual, que interpreta e executa o *bytecode* das aplicações. A segunda determina quais APIs estarão disponíveis de acordo com o tipo de dispositivo e os recursos mínimos que o mesmo possui. A última especifica APIs que se relacionam às características mais específicas dos aparelhos, como a existência ou não de um teclado. A Figura 2 mostra as diferentes plataformas Java e as suas modularizações.

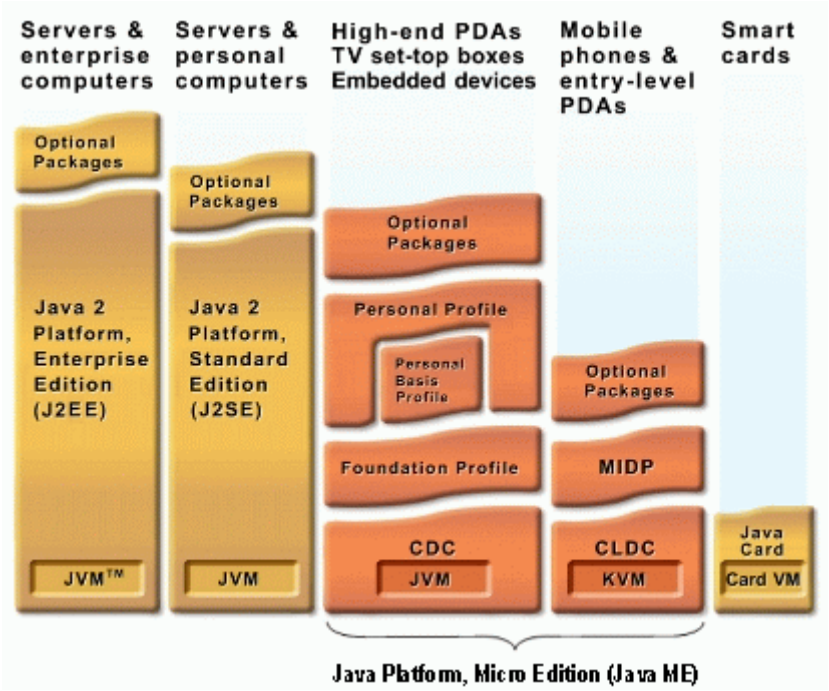

Figura 2: Diferentes Plataformas Java. Retirado de (Java ME Technology)

Podemos ver, pela figura, todas as plataformas Java. A J2SE é um subconjunto da J2EE, a JavaME é um subconjunto da J2SE, e a Java Card é o menor dos subconjuntos

da plataforma Java. Podemos observar, também, as duas *Configurations* disponibilizadas pela JavaME. Uma é a CDC (*Connected Device Configuration*) e a outra é a CLDC (*Connected Limited Device Configuration*). As *Configurations* são os componentes mais importantes na plataforma JavaME, e são elas que determinam como a máquina virtual e os *Profiles* serão especificados.

A CDC se destina a dispositivos mais avançados, com mais recursos, como os receptores de TV, comumente chamados de set-top boxes, e PDAs de última geração. Para oferecer suporte ao CDC, um dispositivo precisa ter, tipicamente, um microprocessador de 32-bits, 2 MB de RAM e 2.5 de ROM disponíveis para o ambiente Java (adaptado de Connected Device Configuration Overview, 2007). Os pacotes usados nessa *Configuration* podem ser vistos em (Martin de Jode, 2004: 17-18).

A outra *Configuration*, a CLDC, foi desenvolvida para aparelhos mais restritos, com menos memória e poder de processamento, como os celulares convencionais. Para oferecer suporte à CLDC é preciso ter um processador de 16 ou 32 bits com freqüência mínima de 16Mhz, pelo menos 160 KB de memória em disco rígido para a Máquina Virtual Java e bibliotecas, 192 KB de RAM, baixo consumo de bateria e acesso primário de rede (Connected Limited Device Configuration (CLDC), 2007). Os pacotes usados nessa *Configuration* podem ser vistos em (Martin de Jode, 2004: 17 a 18). Nela, é definida uma série de restrições relativas à máquina virtual J2SE, que foram impostas com o objetivo de atender às limitações dos dispositivos em questão. Entre essas restrições estão:

- *Class Loaders* (Deitel e Deitel, 2004: 62-64) definidas por usuários são proibidas.
- Não existem grupos de *threads* , *deamon threads*, *Finalisation* e nem reflexão (Deitel e Deitel, 2004: 768 e 413).
- Na versão da CLDC 1.0, não é oferecido suporte à *float point*, nem à referência fraca. Na versão 1.1, ambas passam a ser suportadas.
- O tratamento de erro é limitado. Apenas três classes de erros são oferecidas: *Error*, *OutOfMemory*, e *VirtualMachineError*.

Com o objetivo de promover segurança sem fazer uso das políticas de segurança custosas do J2SE, a plataforma JavaME implementa um ambiente restrito de operação, chamado de *sandbox*. As aplicações são executadas dentro desse ambiente e ficam completamente separadas das execuções de outras que, por ventura, estejam em execução. Qualquer ação tomada deve ser feita por meio do uso das APIs disponibilizadas pelo CLDC. Funções nativas não podem ser acessadas, a não ser indiretamente por meio de uma dessas APIs. Também não é possível realizar o *download* e instalação de bibliotecas que possuam funcionalidades nativas. Isso significa que não existe acesso direto aos recursos do hardware nessa *Configuration*, além dos oferecidos pelas APIs.

Conforme mencionado na arquitetura da plataforma JavaME, abaixo das C*onfigurations* encontram-se as máquinas virtuais. No caso da CLDC, a máquina virtual relacionada é a KVM. O "K" diz respeito ao seu tamanho, que se encontra na faixa dos *kilobytes*. Ela implementa todas as restrições definidas pela *Configuration* em questão. Já no caso da CDC, é oferecida uma máquina virtual Java completa, otimizada para ambientes com recursos limitados.

Acima da CDC ou da CLDC temos os *Profiles*, que determinam o tipo de dispositivo em que a plataforma se encontra. Através dos *Profiles* pode-se determinar se um aparelho é um PDA ou um smartphone, que, para todos os efeitos, podem ser definidos em uma mesma *Configuration*. O CDC oferece três deles, o *Foundation Profile*, o *Personal Basis Profile* e o *Personal Profile*. O primeiro é o mais básico, e não oferece suporte a GUI (*Graphical User Interface*). O segundo é mediano e é um super-conjunto do *Foundation Profile*, que oferece algumas adições como, por exemplo, suporte ao uso de *Xlets* (Java Technology). O *Personal Profile* possui como adição suporte completo ao AWT e Applet (Deitel e Deitel, 2004: 605 e 135). Esses *Profiles* possuem pacotes derivados do J2SE. O conjunto de pacotes usados por cada *Profile* pode ser visto em (Martin de Jode, 2004: 19-20).

O único *Profile* oferecido pelo CLDC, ou seja, que está relacionado a ele, é o MIDP (*Mobile Information Device Profile*), que possui as versões 1.0 e 2.0. O MIDP 1.0 exige que o dispositivo tenha, ao menos, resolução de tela de 96x54 *pixels* em monocromático e algum dispositivo de entrada, como um teclado. Oferece APIs para controlar o ciclo de vida da aplicação, acesso a rede, persistência e interface com o usuário. A versão 2.0 trouxe melhorias, dentre elas:

- Segurança na conexão de rede com o uso do protocolo HTTPS.
- Suporte a mídias de áudio.
- Melhorias gerais nas GUIs, principalmente nos objetos Form e Item.
- Avanços na Game API com a inserção de controle de camadas.
- Representação de imagens em matrizes.

Maiores detalhes sobre o MIDP 2.0 podem ser vistos em (What's New in MIDP 2.0, 2002).

Além dos pacotes derivados do J2SE, o java.lang, java.io e o java.util; existem outros derivados do CLDC, que oferecem as funcionalidades padrões do MIDP 2.0. Essas funcionalidades são organizadas da seguinte forma:

- *Mobile User Interface*, que possui componentes padrões para criação de interface.
- *Multimedia and Game Functionality*, que oferece APIs para manipulação de imagens, controle de animações de jogos e captura do uso da interface humana (i.e. teclado).
- *Extensive Connectivity*, que permite às aplicações se comunicarem usando HTTP, HTTPS, datagramas, *sockets* e portas seriais. Recursos de SMS também são suportados.
- *Over-the-Air Provisioning* define como as aplicações podem ser descobertas, instaladas, atualizadas ou removidas de um determinado dispositivo via interface *wireless*.
- *Persistent Storage* oferece gerenciamento de uma base de dados simples. Uma base de dados sempre está relacionada e é acessada exclusivamente por uma aplicação JavaME instalada. Se uma aplicação for removida do aparelho, a sua base de dados relacionada também será.
- *End-to-End Security* proporciona segurança na execução das aplicações instaladas. Faz isso determinando domínios de segurança e definindo restrições de acesso às funcionalidades privilegiadas.

Virtual Machine, *Configuration* e *Profiles* são as camadas que definem tudo o que uma implementação JavaME deve possuir. As duas últimas determinam as APIs padrões disponíveis no ambiente. A fim de estender essas funcionalidades, alguns pacotes opcionais são definidos, como serviços de Web e de mídia, que podem ser vistos com mais detalhes em (Java ME Technology, 2007), (Java ME Technology APIs & Docs, 2007) e em (Martin de Jode, 2004). Um exemplo, que estende uma implementação JavaME na *Configuration* CLDC, é a MMAPI (*Móbile Media API*), que oferece funcionalidades multimídia, como apresentação de vídeos WMV (*Windows Media Video*). Já no caso da CDC, um exemplo seria o JDBC, que é usado para acessar bancos de dados.

O desenvolvimento das aplicações depende, portanto, da escolha de uma dessas plataformas e dos pacotes opcionais disponíveis. "Na prática, as funcionalidades oferecidas pelo CLDC, com os *profiles* associados e mais os pacotes opcionais, estão próximas das oferecidas pelo CDC" (Martin de Jode, 2004: 5). Como conseqüência, atualmente, a CLDC é comumente utilizada em *smartphones* e em outros dispositivos mais avançados. Em (Java ME Device Table) é apresentada uma lista, feita pela Sun, de todos os dispositivos que utilizam JavaME. Nessa lista, é possível notar que a maioria dos dispositivos é baseada na CLDC, e que poucos são baseados na CDC. Não existe nenhum estudo que verifique se dispositivos portáteis com suporte a TV Digital utilizam mais a CLDC ou a CDC. Também não é possível certificar-se disso através do sítio da Sun. Foram encontrados, entretanto, alguns aparelhos que ofereciam suporte à recepção e ao tratamento do sinal da TV Digital, e todos ofereciam apenas o CLDC, como é o caso do Nokia N92. Assim, é possível perceber que, na prática, a plataforma JavaME utilizada em dispositivos portáteis possui sua arquitetura composta pelo CLDC, KVM e MIDP.

Aplicações CLDC são implementadas, necessariamente, com o uso de MIDlets, que funcionam de forma similar aos Xlets e possuem restrição de 64 Kb, em alguns casos chegando ao extremo de apenas 30 Kb, no consumo de memória. Como discutido anteriormente, funções nativas do dispositivo não podem ser utilizadas. Por exemplo, não é permitido o uso da GUI nativa. A criação da interface é feita exclusivamente com o uso das APIs disponibilizadas pelo MIDP, e isso vale para qualquer outro caso. "O problema é que essas APIs são muito limitadas e acabam não oferecendo suporte ao desenvolvimento de aplicações mais poderosas" (Drewry).

Dessa forma, pode-se dizer que a plataforma JavaME, que faz uso do CLDC e do MIDP, foi concebida para permitir apenas o desenvolvimento de aplicações menos complexas. Existem esforços para a disponibilização de APIs que dêem mais liberdade aos desenvolvedores. Enquanto elas não surgem, algumas implementações proprietárias dessa plataforma vêm com funcionalidades adicionais, que tem como objetivo possibilitar o desenvolvimento de aplicações mais complexas. Porém, isso condiciona a uma perda na portabilidade, que já pode ser notada atualmente, mesmo com o uso exclusivo da plataforma padrão. Isso porque, como existe uma série de módulos opcionais na plataforma padrão, muitos dispositivos vêm com suporte a diferentes funcionalidades. Atrelado a isso, existem uma série de implementações da KVM, cada uma com uma interpretação diferente do padrão. Isso faz com que as aplicações tenham comportamentos diferentes dependendo de onde são executadas, degradando a portabilidade. Além disso, a plataforma está sujeita a *bugs*, e como a JavaME vem instalada no *firmware* dos dispositivos, raramente um *upgrade* que os corrija pode ser feito. Esses problemas podem ser vistos em (Borg, 2006). Para

completar, os mesmos problemas referentes à UI encontrados no Symbian OS são vistos na JavaME. Assim, uma aplicação JavaME precisa ser testada em todos os dispositivos, ou em nichos de dispositivos, que possa vir a ser executada, como é recomendado em (Aiglstorfer, 2006).

Em relação ao suporte às funcionalidades de TV Digital para dispositivos portáteis, a comunidade Java já está desenvolvendo um conjunto de APIs, a JSR 272, que oferece isso. Atualmente ela se encontra na fase de desenvolvimento do *Proposed Final Draft*, que será usado para implementar os testes para prova de conceito e de conformidade. Antes que os primeiros dispositivos capazes de oferecer suporte à JSR 272 apareçam no mercado, o processo de especificação dessa JSR precisará ser finalizado, bem como suas implementações comerciais.

5.3 Comparação

Mesmo oferecendo uma linguagem simples, e mesmo que uma variedade grande de dispositivos portáteis a possua, na prática, a plataforma JavaME não oferece a portabilidade esperada. Sendo assim, o conceito "Write One, Run Anywhere" fica apenas na teoria. Isso constitui um problema grave, pois é a principal vantagem que qualquer plataforma Java deveria oferecer. Em troca de uma degradação de desempenho, o *time-to-market* de uma aplicação deveria ser reduzido e a mesma deveria ter portabilidade.

Por outro lado, a plataforma do Symbian OS não cria nenhuma ilusão acerca da sua portabilidade. Nele é oferecida uma documentação abrangente para ajudar na manutenção desse aspecto. A organização das APIs visa tornar o código mais portável. No que diz respeito à sua linguagem de programação, a Symbian C++, também existe um grande esforço para tornar fácil o porte de aplicações implementadas em ANSI C. Em segundo plano, projetos de implementação da STL para Symbian também estão sendo desenvolvidos, o que pode, em breve, possibilitar o porte de códigos C++ para o Symbian OS. O maior problema de portabilidade da plataforma em questão, que também é encontrado na JavaME, é o da UI.

Apesar disso, o tamanho pequeno das aplicações JavaME permite a sua disponibilização via OTA (Over-The-Air *provisioning*), o que, na prática, não é possível com as aplicações desenvolvidas em Symbian C++, que possuem até mesmo alguns megabytes. (Fórum Nokia, 2003). Essa facilidade, entretanto, é irrelevante, quando se trata de um *middleware* de TV Digital, que dificilmente será distribuído dessa forma. O mais razoável é o dispositivo vir pré-instalado com essa aplicação.

Já no que diz respeito à facilidade de programação, Java é muito mais simples. Isso leva a um outro aspecto, o do desempenho, que em Symbian C++ é muito melhor. Essa é uma relação típica entre as linguagens nativas e a linguagem Java. A facilidade de programação em Java retira uma parte do controle que programador tem sobre a aplicação, e isso o impossibilita de otimizar a aplicação ao máximo. No caso das linguagens nativas, e do Symbian C++ propriamente dito, o programador tem controle total do código, podendo, então, realizar otimizações muito mais refinadas. Além disso, um outro fator que colabora para o desempenho alcançado em Symbian C++ é que as APIs do Symbian OS são otimizadas. Por fim, o fato dos programadores terem acesso direto aos recursos nativos também contribui muito para o desempenho. Isso não é possível em Java, uma vez que a linguagem acessa apenas as APIs da plataforma JavaME sem ter acesso direto aos recursos nativos.

Por fim, o último aspecto analisado é o poder de expressão oferecido pelas duas. Symbian C++ é uma linguagem nativa, portanto o seu poder de expressão é elevado. O desenvolvedor pode criar desde uma aplicação simples até uma mais complexa. Java, por outro lado, é muito restrita. Os seus esforços de padronização ainda não conseguiram desenvolver APIs que atendam ao desenvolvimento de aplicações mais especializadas. O desenvolvimento para TV Digital, em especial, vai depender, ainda, da finalização da JSR 272. Quando isso acontecer, novas implementações precisarão ser criadas e novos dispositivos terão que ser lançados com a JSR 272. Só então é que aplicações de TV Digital vão se tornar mais viáveis no JavaME.

Em resumo, segundo (Forum Nokia, 2003), "J2ME é ideal para aplicações que podem ser implementadas em um contexto de memória limitado, onde performance não é um fator crítico, e que precisem ser distribuídas amplamente". Por outro lado "aplicações grandes e mais complexas, que precisem interagir com outros dispositivos e que precisem de muito processamento ou memória" são mais apropriadas para serem desenvolvidas em Symbian C++ na plataforma do Symbian OS.

5.4 Conclusão

No caso do sistema operacional Symbian, temos a plataforma JavaME e a plataforma nativa do sistema, que implementam, respectivamente, as linguagens Java e Symbian C++. No que diz respeito à facilidade de programação, pode-se avaliar a linguagem Symbian C++ como inferior à Java. Entretanto, Symbian C++ possui um desempenho melhor. É importante observar que estamos analisando as plataformas frente a uma aplicação (o middleware) que será implementado uma única vez, e não quanto a seu suporte a aplicações em geral, assim, a facilidade de programação perde sua relevância e a importância do desempenho deve ser salientada. O problema da portabilidade é encontrado nas plataformas de ambas as linguagens.

Em (Lars Kirkhus e Anders R. Sveen, 2003: 33) foi feita uma comparação teórica entre várias plataformas que tinha como objetivo escolher a ideal para o desenvolvimento de um *framework*. A escolhida acabou sendo a Java para JavaME, ficando a Symbian C++ para o Symbian OS com a segunda colocação. Em (Lars Kirkhus e Anders R. Sveen, 2004) é utilizada a conclusão de (Lars Kirkhus e Anders R. Sveen, 2003) para o desenvolvimento do *framework*. O resultado é que o desenvolvimento teve que ser prorrogado devido a problemas de portabilidade do JavaME. Para todos os efeitos, portar um código Symbian C++ na plataforma Symbian OS não parece ser mais complicado do que portar um código Java na plataforma JavaME.

É importante ressaltar que JavaME está disponível em uma variedade bem maior de dispositivos, e não só na plataforma Symbian. A sua programação facilitada, bem como sua abrangência, ainda constituem uma vantagem muito grande. Sobram as questões de desempenho e limitações. Apesar do desempenho ser melhor com o uso do Symbian C++, isso não constitui, necessariamente, um fator determinante na sua escolha para aplicações em geral, embora seja importante no caso particular de um *middleware* embarcado. Na verdade, essa questão acaba ficando em segundo plano considerando-se o problema das limitações impostas pela JavaME. Esse é o problema determinante que dificulta o desenvolvimento de um *middleware* declarativo nessa plataforma e na linguagem Java. O que mais conta nesse sentido é a impossibilidade do acesso direto os recursos de *hardware*, pois um *middleware* de TV Digital certamente precisará fazer acessos desse tipo.

Sendo assim, a plataforma do Symbian OS/C++ é a que hoje oferece, na opinião dos autores, um ambiente mais adequado para o desenvolvimento de um *middleware* de TV Digital. Isso porque possui performance assegurada, acesso aos recursos nativos por meio da linguagem Symbian C++, é livre de limitações, a não ser as impostas pelo hardware, e tem boa portabilidade dentro da sua plataforma. Em termos de disponibilidade, embora essa plataforma seja limitada se comparada à JavaME, é a que abrange a maioria do mercado em relação às outras, segundo (Canalys Reseach, 2007).

O desenvolvimento de aplicações no Linux Mobile é feito na sua plataforma nativa usando a linguagem C/C++, que, como já foi mencionado, também é adequada para o desenvolvimento de um *middleware* de TV Digital.

6 Organizações e seus Middlewares

Como sabido, *middleware* é uma camada intermediária entre o hardware/SO e as aplicações. O seu objetivo é esconder os detalhes da plataforma de hardware e sistema operacional para que o desenvolvimento das aplicações seja simplificado. No contexto da TV Digital, o objetivo do *middleware* é permitir que as aplicações descrevam a composição do conteúdo digital sem se preocuparem com outras questões, como o método de transporte utilizado etc.

Neste capítulo, serão apresentados os principais sistemas de TV Digital terrestre, sua aplicação nos dispositivos portáteis e os seus padrões de *middleware*. Também serão apresentadas as implementações desses *middlewares* nos dispositivos portáteis, quando existirem.

Essa seção vai apresentar, também, os esforços de padronização da IPTV, focando a questão dos *middlewares*. Entretanto, o que mais existe hoje são soluções proprietárias de IPTV, e, por causa disso, elas também serão abordadas. Por fim, será feita uma avaliação de como a IPTV está posicionada, atualmente, na questão dos dispositivos portáteis.

6.1 DVB

O DVB é um fórum formado por um conjunto de empresas, criado com o objetivo de definir padrões relacionados à transmissão de TV Digital e serviços de dados. Existe desde 1993 e surgiu em vista da necessidade de uma padronização nessa área.

O grupo criou o DVB-H, adotado oficialmente como padrão em 2004, a fim permitir a transmissão da TV Digital para os dispositivos portáteis. O padrão foi desenvolvido a fim de atender aos requisitos dos Handhelds, mas pode ser usado em outros dispositivos com características semelhantes, como os smartphones.

Existem duas camadas no DVB-H, a de enlace e a física. Na primeira, existe um protocolo de correção de erro, o MPE-FEC, que pode ser utilizado ou não pelo serviço, e uma estratégia de *time-slicing*, que visa reduzir o consumo de energia. Na camada física, são implementadas funcionalidades que melhoram a recepção nos dispositivos portáteis, adicionam flexibilidade na composição da rede e diminuem os efeitos do ruído. Informações mais detalhadas sobre esse padrão de transmissão podem ser encontradas em (ETSI EN 302 304 V1.1.1, 2004).

Atualmente, já existem dispositivos desenvolvidos especificamente para o DVB-H, como o N77 e o N92. Entretanto, o *middleware* usado nesses dispositivos não foi encontrado.

O DVB definiu, ainda, um outro padrão para transmissão de TV Digital, o DVB-IP, que, nesse caso, foi desenvolvido especificamente para IPTV.

6.1.1 MHP (*Multimedia Home Platform*)

MHP é o *middleware* de TV Digital especificado pelo DVB. A idéia por trás do MHP é permitir o seu uso com qualquer serviço de TV Digital existente, seja um especificado pelo próprio DVB, como o DVB-H ou DVB-IP, ou qualquer outro. Também é possível que aplicações de outras especificações sejam incorporadas a ele. A arquitetura do MHP é apresentada na Figura 3.

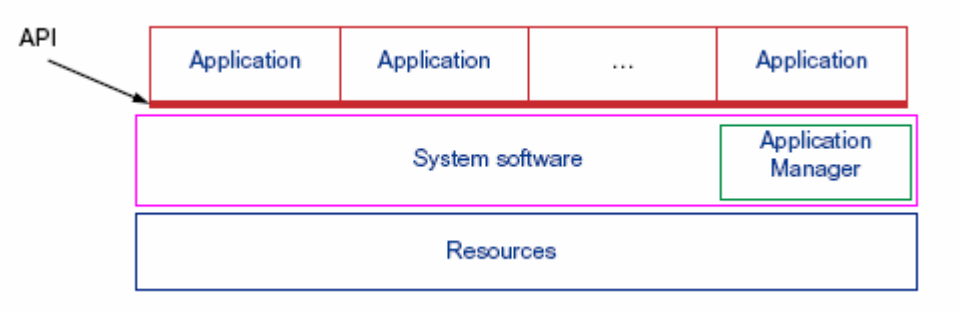

Figura 3: Arquitetura MHP. Retirado de (ETSI TS 102 812 v1.2.1 (2006-03))

Resources representa os recursos, tanto de hardware como de software, disponíveis. A arquitetura não distingue se esses recursos estão distribuídos.

System software é uma camada intermediária entre a aplicação e os recursos.

Application Manager realiza o gerenciamento das aplicações em execução. É nessa camada que se encontra a API do MHP e é por essa API que as aplicações conseguem fazer uso do serviço digital.

Applications são as aplicações de conteúdo interativo que fazem uso dos serviços oferecidos através da API MHP.

Plug-ins podem ser criados a fim de integrar aplicações de outros padrões na arquitetura MHP. Um plug-in não deve interferir no funcionamento de uma implementação padrão do MHP ou ainda em outros plug-ins usados. A Figura 4 ilustra o uso desses elementos.

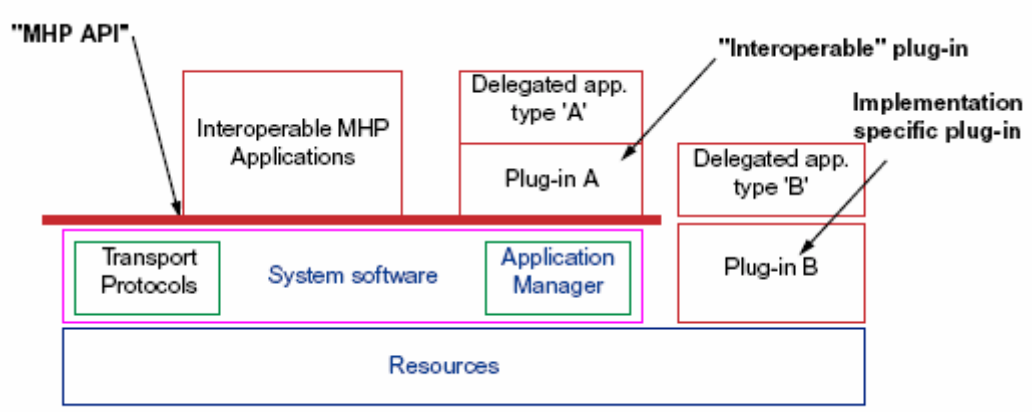

Figura 4: As duas opções no uso de plug-ins. Retirado de (ETSI TS 102 812 v1.2.1 (2006-03))

Como se pode observar, existe duas formas de se implementar um plug-in. A forma "A" constitui-se da criação de uma aplicação MHP, que será responsável pela execução das aplicações portadas. Esse plug-in pode ser visível ou invisível para o MHP. No caso de ser visível, o *Application Manager* fica responsável pelo controle da aplicação, e o plug-in vai precisar implementar uma interface que permita ao *Application Manager* fazer esse controle. Se for invisível, todo o controle da aplicação deve ser feito pelo próprio plug-in. A forma "B" constitui-se de uma implementação específica que ficará responsável pela execução da aplicação portada, e que é adicionada na camada *System software* da arquitetura. Exemplo de plug-in para o MHP é a implementação do ambiente declarativo do *middleware* Ginga, realizada em Java (Rodrigues, 2007).

O MHP define o DVB-HTML e o DVB-J como seus ambientes declarativo e procedural, respectivamente, sendo que o primeiro está embutido no segundo. É importante salientar, entretanto, que a implementação do DVB-HTML é opcional. Uma aplicação DVB-HTML executa sobre um *user agent*, que pode ser adicionado como um plug-in na camada de abstração do MHP. A Figura 5 ilustra isso.

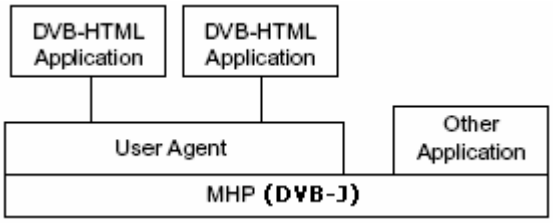

Figura 5: DVB-HTML, DVB-J e o User Agent. Adaptado de (ETSI TS 102 812 v1.2.1 (2006-03))

6.1.1.1 DVB-HTML

O DVB-HTML é uma linguagem baseada nas recomendações do W3C, e usa as tecnologias XHTML, XML, CSS, DOM e ECMAScript. DVB-HTML utiliza apenas um subconjunto dos módulos XHTML. A Figura 6 mostra quais módulos são utilizados.

Modularization	Required
Structure	Yes
Text	Yes
Hypertext	Yes
List	Yes
Applet	
Presentation	Yes
Edit	
Bi-directional Text	Yes
Basic Forms	
Forms	Yes
Basic Tables	Yes
Tables	
Image	Yes
Client side Image map	Yes

Modularization	Required
Server side image map	
Object	Yes
Frames	Yes
Target	Yes
Iframe	Yes
Intrinsic events	
DVB Intrinsic events	Yes
Metainformation Module	Yes
Scripting	Yes
Style sheet	Yes
Style Attribute	Yes
Link	Yes
Base	Yes
Name Identification	
Legacy	

Figura 6: Módulos usados no DVB-HTML. Adaptado de (ETSI TS 102 812 v1.2.1 (2006-03))

A Figura 7 apresenta os tipos de objetos suportados pelo DVB-HTML.

MIME media type	Common name
text/xml	XML
application/xml	
text/css	CSS
text/plain	Monomedia format for text
text/dvb.utf8	
audio/mpeg	Monomedia format for audio clips or Audio
image/jpeg	JPEG
image/png	PNG
image/gif	GIF
image/mpeg	MPEG-2 I-Frames
video/dvb.mpeg.drip	MPEG-2 Video drips
video/mpeg	MPEG video
multipart/dvb.service	Multipart DVB Service
application/dvb.pfr	Downloadable Fonts
application/dvbj	Initial class file of an Xlet
text/ecmascript	ECMAScript

Figura 7: Tipos de objetos suportados pelo DVB-HTML. Adaptado de (ETSI TS 102 812 v1.2.1 (2006-03))

O uso do CSS deve seguir a especificação da W3C do CSS2, vista em (Cascading Style Sheets, Level 2), com alterações especificadas pelo MHP:

- O parâmetro opacity é adicionado para permitir apresentações com transparência.

- No parâmetro *color*, especifica-se que não é garantido o comportamento padrão no uso de uma *system color*.
- *length* usa o *aspect ratio* de um dispositivo para determinar o tamanho de um *pixel*.
- Um shape só pode do tipo *rect*.
- *Angle*, *frequency* e *time* não são requeridos pelo MHP.
- A regra *port* não é requerida, e a *viewport*, que determina visualmente um ponto de entrada de uma aplicação, deve ser usada.

É possível embutir um Xlet no DBV-HTML através da tag "object". Para isso, o *type* do objeto deve ser igual a "application/dvbj". Parâmetros podem ser definidos a fim de serem passados como argumento para o Xlet que será executado.

O ECMAScript *1st edition* é utilizado como *scripting* language no DVB-HTML. É possível acessar as APIs do DVB-J por meio dessa linguagem de *script*, e algumas das suas chamadas retornam elementos que não são suportados pelo ECMAScript. Sendo assim, três tipos de objetos a mais são definidos para a linguagem: JavaArray, que encapsula uma *array* DVB-J; JavaObject, que encapsula um objeto DVB-J que pode ser convertido para uma string, um número ou um valor *booleano*; e JavaClass, que encapsula uma classe DVB-J. Tabelas de conversão dos tipos ECMAScript para os do DVB-J podem ser vistas em (TS 102 812 v1.2.1 (2006-03): 129-130).

Em relação ao DOM, a Figura 8 mostra os itens requeridos pela especificação MHP.

DOM Module		Required
Package	Feature String	
Level 2 Core	Core	Yes
	XML	No
Level 2 Views	Views	Yes
Level 2 Stylesheets	StyleSheets	No
Level 2 CSS stylesheets	CSS	No
	CSS2	Yes
Level 2 Events	Events	Yes
	UIEvents	Yes
	MutationEvents	Yes
DVB-HTML	DVBHTML	Yes
DVB Events	DVBEvents	Yes
DVB Key Events	DVBKeyEvents	Yes
DVB CSS	DVBCSS	Yes
DVB Environment	DVBEnvironment	Yes

Figura 8: Adaptações sobre o DOM. Adatado de (ETSI TS 102 812 v1.2.1 (2006-03))

O MHP define o conceito de *trigger*, que é a forma com a qual o provedor do serviço digital pode se comunicar e influenciar na execução de uma aplicação DVB-HTML. São mensagens enviadas em um canal diferente daquele usado na transmissão do serviço digital. Essas mensagens contêm informação necessária para realizar uma mudança no comportamento da aplicação. O transporte de um *Trigger* é feito via DSM-CC *stream events*.

6.1.1.2 DVB-J

O ambiente procedural do MHP é o DVB-J. Como o uso da linguagem declarativa é opcional, sempre que se fala no uso do MHP como *middleware*, faz-se referência ao DVB-J. Sua linguagem procedural é baseada em Java e, portanto, faz uso dos seus pacotes e define algumas extensões.

O funcionamento de uma aplicação DVB-J é o mesmo de um Xlet, uma vez que a aplicação é uma extensão Xlet. Existem os comandos de initXlet, startXlet e destroyXlet, que são derivados da JavaTV. A Figura 9 ilustra o funcionamento de um Xlet e, conseqüentemente, de uma aplicação procedural MHP.

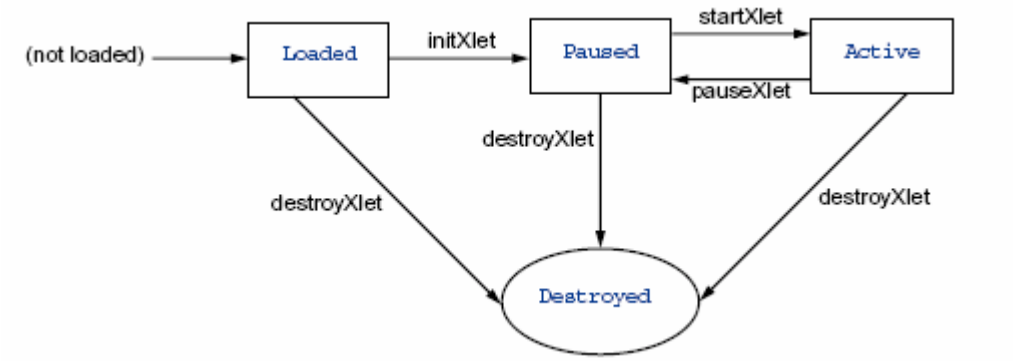

Figura 9: Estados de uma aplicação MHP. Retirado de (TS 102 812 v1.2.1 (2006-03))

Os seguintes pacotes padrões da plataforma Java são suportados com devidas alterações: java.lang, java.lang.reflect, java.util, java.util.zip, java.io, java.net, java.beans, java.math, java.math.BigInteger e java.text. Apenas os pacotes especificados são requeridos, os seus sub-pacotes não. Maiores detalhes sobre as alterações definidas pelo MHP para as classes desses pacotes podem ser vistas em (TS 102 812 v1.2.1 (2006-03)), no Capítulo 11.3.1. O DVB define dois pacotes extras, o org.dvb.lang e o org.dvb.event. Detalhes podem ser vistos na mesma referência anterior, no Capítulo 11.3.2.

DVB-J especifica, também, o uso de APIs derivadas da plataforma Java para a sua máquina de apresentação. A GUI é formada por classes do pacote java.awt, e a interface de usuário é feita com o uso dos pacotes org.havi.ui e org.havi.ui.event.

Mais detalhes sobre o DVB-J podem ser encontrados em (TS 102 812 v1.2.1 (2006-03)) Capítulo 11.

6.1.2 Implementações

Nenhum *middleware* DVB para dispositivos portáteis pode ser encontrado. Foram encontradas apenas implementações em conformidade com o DVB-H, mas usando outras soluções para o desenvolvimento de aplicações. Um exemplo é o produto da Axel, o Salmonstream (Axel Technologies). As suas aplicações são baseadas no ESG (Electronic Service Guide) do DVB (ETSI TS 102 471 V1.2.1 (2006-11)), e no ESG do Java. A Figura 10 ilustra a sua arquitetura. No futuro, a Axel pretende tornar disponível, no Salmonstream, o desenvolvimento de aplicações com as APIs do pacote Java para TV Digital, a JSR 272.

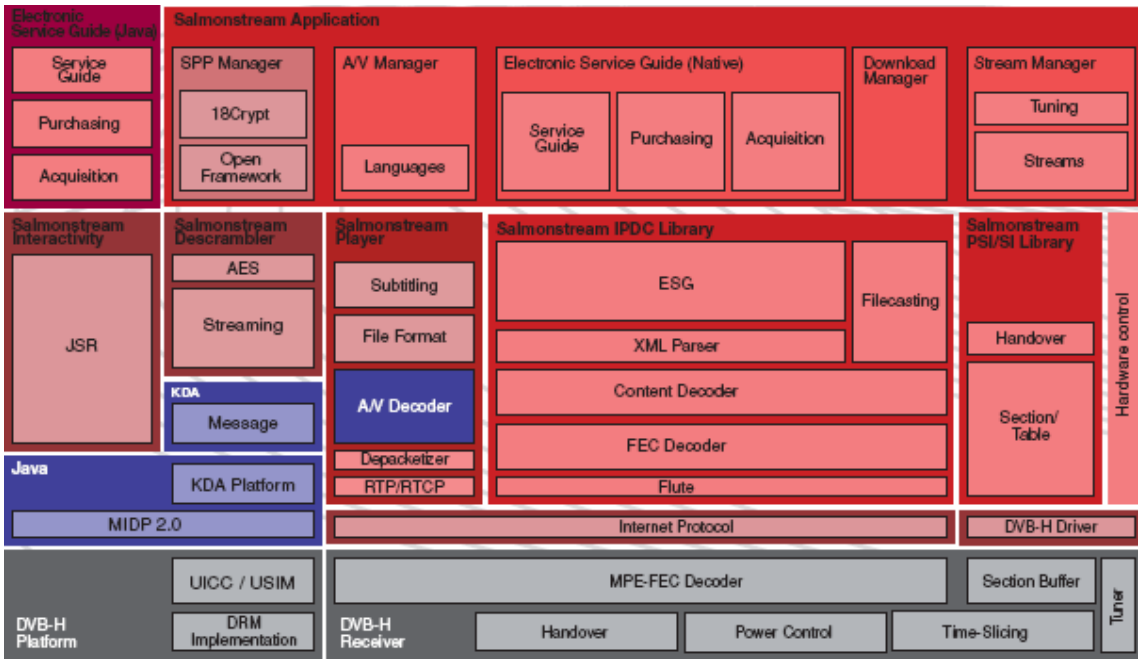

Figura 10: Arquitetura do Salmonstream. Retirado de (Axel Brochure)

Outras soluções com suporte a DVB-H (apenas padrão de transmissão) são:

- Thin Mobile TV Middleware DVB-H, InterVideo's iMobi Mobile DTV e Hantro Mobile TV, que oferecem apenas ESG.
- Cardinal Datacast, que oferece uma linguagem XHTML, possivelmente proprietária, e Java para o desenvolvimento de aplicações. Também existe uma ferramenta de autoria, o Cardinal Professional Studio, que pode ser usada para o mesmo objetivo.
- S3 onHandTV, que oferece ESG e PVR.
- eMuzed Middleware, que não especifica nenhuma forma de se prover aplicações através do seu *middleware*.
- NeoMagic Mobile TV Solution, que oferece apenas uma ferramenta de autoria, o MiniMagic 6+ Development Board.
- AltiCaptor-Mobile, que oferece HTML, Java (CLDC), ESG e o Alticaptor Mobile SDK como ferramenta de autoria.
- SmartVision™ Mobility platform, que oferece suporte a ESG, VoD e PVR.
- IPanel Browser DVB, que oferece suporte a ESG, e-mail e *chat*.

6.2 SBTVD

A PUC-Rio e a Universidade Federal da Paraíba desenvolveram o *middleware* Ginga para o Sistema Brasileiro de TV Digital - SBTVD. A primeira ficou responsável pelo ambiente declarativo do Ginga, enquanto que a segunda ficou responsável pelo ambiente procedural. Ainda não existem implementações comerciais desse *middleware* para dispositivos portáteis, onde só é exigido o ambiente declarativo. Futuramente, produtos com esse *middleware* serão lançados para o sistema de TV Digital Brasileiro.

6.2.1 Ginga

Ginga é o padrão brasileiro de *middleware* para TV Digital. Nele são definidas duas classes de aplicações, as declarativas e as procedurais, chamadas, respectivamente, de Ginga-NCL e Ginga-J.

O uso de ambas as linguagem é obrigatório nos terminais fixos. Apenas o Ginga-NCL é obrigatório para dispositivos portáteis. Para permitir o desenvolvimento de aplicações híbridas, o padrão determina os meios pelos quais entidades declarativas e procedurais podem se comunicar através de uma ponte entre os dois ambientes.

6.2.1.1 Ginga-NCL

A parte declarativa do *middleware* de TV Digital Brasileiro, o Ginga-NCL, utiliza a linguagem NCL (*Nested Context Language*) para descrever apresentações hipermídia de TV Digital. É uma linguagem que tem foco no sincronismo das mídias, na adaptabilidade e no suporte a múltiplos dispositivos de exibição. Uma outra vantagem é que o tamanho das suas aplicações é pequeno, facilitando processo de transmissão da programação de TV Digital.

NCL é uma linguagem XML e, assim como as linguagens XML definidas pelo W3C, é dividida em módulos. Normalmente, existem situações onde o uso completo de uma linguagem não é necessário, ou é até mesmo prejudicial. O caso da programação para dispositivos portáteis é um exemplo disso, onde é melhor usar um *subset* da linguagem que seja ideal ao contexto. Com isso em mente, a NCL especifica *profiles*, que contêm subconjuntos dos módulos da linguagem, usados para atender a diferentes requisitos. Para o contexto deste estudo, o *profile* que interessa é o Basic DTV, perfil mínimo para dispositivos portáteis. Cada módulo NCL se encontra dentro de uma área funcional. A Figura 11 apresenta os módulos do perfil Basic DTV.

Áreas Funcionais	Módulos
Structure	Structure
Layout	Layout
Components	Media
	Context
Interfaces	MediaContentAnchor
	CompositeNodeInterface
	PropertyAnchor
	SwitchInterface
Presentation Specification	Descriptor
Linking	Linking
Connectors	CausalConnectorFunctionality
	ConnectorBase
Presentation Control	TestRule
	TestRuleUse
	ContentControl
	DescriptorControl
Timing	Timing
Reuse	Import
	EntityReuse
	ExtendedEntityReuse
Navigational Key	KeyNavigation

Figura 11: Módulos da NCL no perfil Básico.

Structure é o módulo que define a estrutura básica de um documento NCL. Nele são definidos os nós "ncl", "head" e "body". "ncl" é o nó pai de todos os documentos NCL, "head" e "body" são seus filhos diretos e todos os outros elementos dos outros módulos são filhos diretos desses últimos dois. O módulo *Structure* faz parte área funcional *Structure*.

O módulo *Layout* define dois elementos usados para criar regiões de apresentação de objetos multimídia. Essas regiões são definidas para determinado dispositivo de saída. O primeiro dos elementos é o "regionBase", que determina, através do seu atributo *device*, em qual dispositivo as regiões serão criadas. O seu nó filho, o "region", é o segundo elemento desse módulo. É por ele que se define uma região no dispositivo referenciado pelo "regionBase". Os atributos *left*, *right*, *top*, *bottom* e zindex de um "region" servem para determinar o tamanho e a posição da região no dispositivo. As *regions* podem ser aninhadas, ou seja, uma pode estar contida na outra. Isso determina, analogamente, se uma região faz parte de uma outra. Esse módulo é o único integrante da área funcional *Layout*.

Media é o módulo que define o elemento "media", e esse, por sua vez, referencia um objeto de mídia que poderá ser usado na apresentação. Esse elemento possui, além do id, o atributo src, que é uma URL para o objeto de mídia, e o *type*, que determina o tipo desse objeto. Esse último serve apenas de guia e é opcional, pois é possível obtê-lo através da extensão do arquivo apontado pelo src. Um outro atributo importante é o *descriptor*, que informa um descritor para o objeto de mídia associado. Descritores serão abordados mais adiante. O módulo *Context* especifica o elemento "context", que serve para agregar elementos em um único contexto. Isso serve para facilitar a estruturação do documento NCL. Um elemento desse tipo pode ter, inclusive, outros *contexts* aninhados recursivamente. Esses dois módulos fazem parte da área funcional *Components*.

A próxima área funcional abordada é a *Interfaces*. Nela existem quatro módulos que possuem elementos usados na criação das interfaces dos nós NCL. Essas interfaces são usadas nos relacionamentos entre os nós. Via de regra, nenhum nó NCL pode ser acessado sem ser por uma das suas interfaces. O primeiro módulo é o MediaContentAnchor, que define o elemento "área" usado para representar porções espaciais, temporais ou espaço-temporais de uma mídia. Além do atributo id, existem outros, como o begin, end e coords; que especificam as porções referenciadas do objeto de mídia. O segundo módulo é o CompositeNodeInterface, que define o elemento "port" usado para referenciar um nó contido dentro de um "context". Apenas através de um "port" é que um elemento de um "context" pode ser acessado de fora. Além do "port", esse módulo especifica dois atributos, o component e o interface, que referenciam, respectivamente, um nó e uma das suas interfaces. O terceiro módulo é o PropertyAnchor, que define o elemento "property", usado para referenciar propriedades de um objeto NCL (mídia ou composição). Para alterar o tamanho de um vídeo, por exemplo, é preciso criar uma interface do tipo "property" para os seus atributos *width* e *height*. Esse elemento define um id, um *name*, que é o nome da propriedade referenciada, e um *value*, que é o valor que devera ser atribuído à propriedade. O último módulo é o *SwitchInterface*, que é usado para mapear as alternativas de um nó do tipo *switch*. Dois elementos são introduzidos, o primeiro é o "mapping" que realiza o mapeamento das opções através dos atributos component e interface. Esses dois possuem funcionamento idêntico aos do elemento "port". O segundo é o "switchPort", que mantém uma coleção de "mappings".

Descriptor é o único módulo da área funcional PresentationSpecification e define os elementos "descriptor", "descriptorParam" e "descriptorBase". O "descriptorBase" deve ser criado no *head* do documento e serve para armazenar uma coleção de elementos "descriptor". Esse segundo, por sua vez, é usado para criar um modelo de representação para os objetos de mídias definidos. Esse elemento é que define a forma de apresentação das mídias. Para esse objetivo, existe uma série de atributos que podem ser usados e que são vistos com detalhes em (SBTVD, 2007: 54). Parâmetros também podem ser definidos em um "descriptor", com o mesmo objetivo das propriedades. Isso é feito através do elemento "descriptorParam", que possui dois atributos, o *name*, usado para indicar o nome da propriedade, e o *value*, que indica o seu conteúdo.

Linking é o único módulo da área funcional *Linking* e define dois elementos, o "link" e o "bind", usados para relacionar nós (objetos) NCL. Uma relação pode ser de vários tipos diferentes, por isso o "link" possui um atributo para um elemento que determina o tipo dessa relação. Esse elemento é o "*connector*", que define uma relação criando papéis que serão assumidos por nós. Por exemplo, um "connector" pode criar dois papéis, de *start* e *onEnd*, e definir que o nó que assumir o papel de *start* será iniciado logo após o término da apresentação do outro que assumiu o papel de *onEnd*. O "connector", entretanto, não define quais elementos estão associados a quais papéis. É por isso que o "link" possui o parâmetro "bind", que faz exatamente essa associação.

A área funcional *Connector* contém módulos onde são definidos alguns dos elementos mais importantes e mais complexos da NCL, os conectores. Esses elementos são totalmente independentes de quaisquer outros existentes na linguagem NCL, e servem para definir as relações dos relacionamentos criados no documento. Documentos NCL podem importar bases de conectores. Isso pode ser feito através do elemento "connectorBase", que deve ser definido no *head* do documento NCL. Existem muitos detalhes acerca dos conectores que os tornam um recurso muito poderoso da linguagem NCL, mas que também os fazem mais difíceis de serem criados por usuários não especialistas, daí a importância da importação de bases de conectores pré-criadas. Maiores detalhes podem ser encontrados em (SBTVD, 2007), no Capítulo 7.2.8.

A área funcional *Presentation Control* possui quatro módulos, que tem como objetivo proporcionarem os meios de se definir alternativas em uma apresentação. O módulo Test Rule define elementos que serão usados para definir as regras das alternativas. O elemento "rule" é a regra propriamente dita, que possui um operador e dois operandos. Por exemplo, uma regra pode ter a variável "nacionalidade" e o valor "português" como os seus dois operandos, e o operador "eq". Essa regra vai ser avaliada como verdadeira se a variável "nacionalidade" possuir o seu valor igual (***eq****ual*) a "português". Outros elementos são "compositeRule", que permite a definição de regras compostas, e o "ruleBase", que agrega todos os elementos do tipo "rule" e "compositeRule". Todos esses elementos devem ser definidos no *head* do documento. O módulo TestRuleUse especifica o elemento bindRule, que associa as regras com as alternativas. A primeira alternativa que possuir uma regra verdadeira será apresentada. O módulo ContentControl define o elemento "swicth", que vai permitir a criação de nós alternativos associados a regras. Um outro elemento definido pelo mesmo módulo é o "defaultComponent", que determina a apresentação de um elemento padrão para o caso de nenhuma regra ser verdadeira. Por fim, o módulo DescriptorControl determina os elementos "descriptorSwicth", que agrega um conjunto de *descriptors* alternativos para um nó semelhantemente ao que um "switch"

faz, e o "defaultDescriptor", que funciona de forma similar ao "defaultComponent", só que para descritores.

A área funcional *Timing* possui o módulo *Timing* que especifica atributos usados na definição de propriedades temporais de objetos do documento hipermídia. Esses atributos podem ser usados nos elementos do tipo "descriptor".

A área funcional *Reuse* possui três módulos que permitem o reuso de componentes NCL. O módulo *Import* permite o reuso de bases de elementos já criadas. Isso é feito através do elemento "importBase", que possui dois atributos, documentURI, que referencia um documento NCL possuidor da base importada, e alias, que é um apelido usado no documento corrente para referenciar a base importada. O módulo EntityReuse possibilita aos elementos "media", "context", "body" e "swicth" serem reusados através do atributo refer. O último módulo é o ExtendedEntityReuse que permite o uso do atributo newInstance em elementos reusados.

Por fim, a área funcional Navigation Key possui um módulo com o mesmo nome e oferece extensões usadas para opções de movimento de foco.

EM NCL um documento HTML é um tipo de elemento de mídia. De forma semelhante, linguagens imperativas podem ser adicionadas e usadas como nós de mídia. Em específico, a Ginga-NCL deve oferecer suporte a duas linguagens procedurais pela especificação, Lua e Java. Lua é a linguagem de script da NCL e a Java deve seguir as especificações do Ginga-J. Para dispositivos portáteis, apenas o suporte à linguagem Lua é obrigatório.

6.2.1.2 Ginga-J

O Ginga-J é o ambiente *procedural* do *middleware* Ginga. Na sua versão atual, é uma extensão do GEM (*Globally Executable MHP*).

GEM determina uma série de APIs que devem ser usadas. Dentre elas, encontram-se as provenientes de pacotes da JavaTV, DAVIC, HAVi e as definidas pelo próprio GEM. O Ginga-J especifica que os seguintes pacotes desses padrões são requeridos (SBTVD, 2006):

- JavaTV
    - java.awt
    - java.awt.event
    - java.awt.image
    - java.beans
    - java.io
    - java.lang
    - java.lang.reflect
    - java.net
    - java.security
    - java.security.cert
    - java.util

    - java.util.zip
    - javax.media
    - javax.media.protocol
    - javax.tv.graphics
    - javax.tv.locator
    - javax.tv.media
    - javax.tv.media.protocol
    - javax.tv.net
    - javax.tv.service
    - javax.tv.service.guide
    - javax.tv.service.navigation
    - javax.tv.service.selection
    - javax.tv.service.transport
    - javax.tv.util
    - javax.tv.xlet
    - java.math
    - java.rmi
    - java.security.spec
    - javax.net
    - javax.net.ssl
    - javax.security.cert
- DAVIC
    - org.davic.media
    - org.davic.resources
    - org.davic.mpeg
    - org.davic.mpeg.sections
    - org.davic.net
    - org.davic.net.dvb
    - org.davic.net.tuning
- HAVi
    - org.havi.ui
    - org.havi.ui.event
- Extensões GEM
    - org.dvb.application
    - org.dvb.dsmcc

- org.dvb.event
- org.dvb.io.ixc
- org.dvb.io.persistent
- org.dvb.lang
- org.dvb.media
- org.dvb.net
- org.dvb.net.tuning
- org.dvb.net.rc
- org.dvb.test
- org.dvb.ui
- org.dvb.user

O Ginga-J também define extensões para as especificações GEM. Sobre a API org.isdtv.net.tuning são adicionadas funcionalidades de sintonização. Ao invés da JMF 1.0, Ginga-J especifica o uso da JMF 2.1. O Pacote org.sbtvd.media é uma extensão sobre o org.dvb.media que oferece suporte a *streams* de vídeo especificados pelo Ginga-J. De forma semelhante, o org.sbtvd.net.rc é uma extensão sobre o org.dvb.net.rc que possibilita o uso do canal de retorno para o envio assíncrono de informações.

Ginga-J possui algumas especificações próprias. A primeira é a org.sbtvd.ui.event, que permite interações múltiplas de múltiplos usuários. Por exemplo, dois usuários poderiam interagir com uma mesma aplicação usando dispositivos de interação diferentes. A segunda é a org.sbtvd.interactiondevices, que possibilita à aplicação saber quais dispositivos podem ser usados pelo Ginga-J. Por fim, org.sbtvd.bridge define a ponte entre aplicações declarativas e procedurais.

6.3 ATSC

O ATSC definiu o ACAP como o padrão americano para distribuição de serviços de TV Digital, mas não é considerado adequado para o ambiente portátil (MBMS White Paper, Agosto 2004).

O problema que impossibilita o uso desse padrão em dispositivos portáteis é a utilização do 8-VSB como técnica de modulação. Como pode ser visto em (DTV Guide Transmission), o 8-VSB não atende os requisitos mínimos necessário para a transmissão sem o uso de cabos, própria para o caso dos dispositivos portáteis. Por conta disso, não foi encontrada nenhuma implementação para dispositivos portáteis do *middleware* ATSC.

6.4 ARIB

A ARIB (*Association of Radio Industries and Businesses*) realiza pesquisa e traz soluções para questões ligadas à área de radiodifusão. Seu objetivo é promover novas tecnologias e popularizar o uso de serviços providos por radiodifusão no Japão.

O ISDB-T é o padrão para TV Digital Terrestre definido por essa organização. A TV Digital em Dispositivos portáteis também já existe na prática no Japão com o uso o

ISDB-T, que já dispunha dos requisitos necessários para isso, como é observado em (MBMS White Paper, http://www.tzi.de/~dku/pub/iqpc-mobiletv-kutscher-japan.pdf).

O padrão de TV Digital Japonês possui uma adaptação do seu uso, descrita em (B24 Appendix 5), para serviços de TV Digital portátil. Já existem alguns dispositivos portáteis, chamados de *One Seg devices*, que oferecem suporte ao ISDB-T. Exemplos são: D903iTV, P903iTV, e o SH903iTV, todos da DoCoMo. O *middleware* de TV Digital do ISDB-T é baseado na linguagem BML (*Broadcast Markup Language*), que é uma linguagem declarativa baseada em XHTML 1.0 e que tem foco na interação do usuário.

6.4.1 BML

A estruturação da BML é feita por módulos de acordo com a Figura 12.

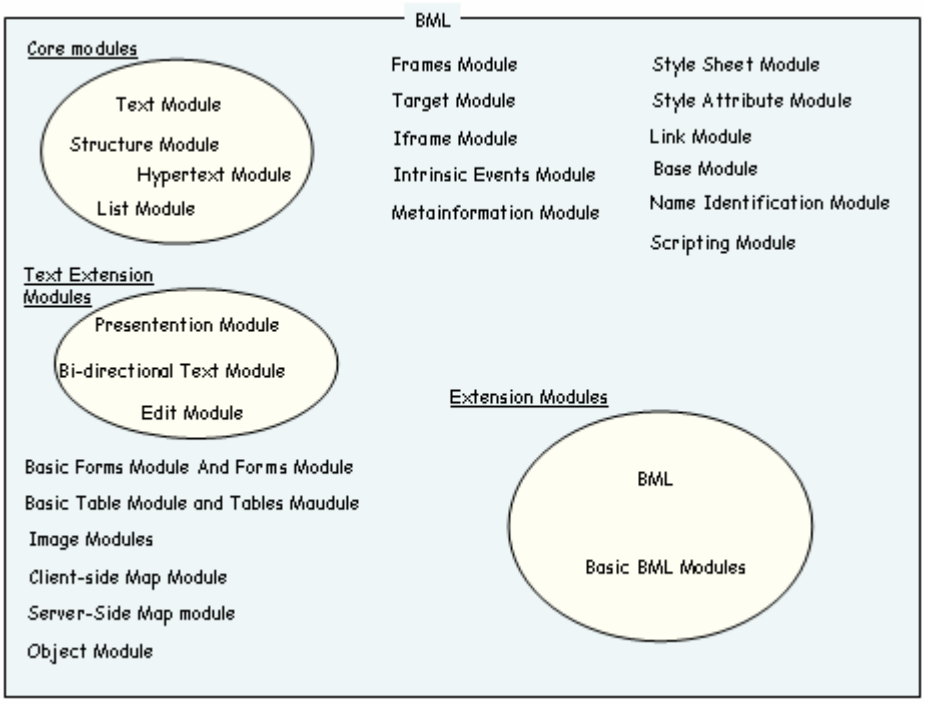

Figura 12: Módulos do BML.

Os módulos BML são brevemente comentados a seguir. No final, uma tabela vai apresentar aqueles que devem ser suportados nos serviços oferecidos para dispositivos portáteis.

O *Structure Module* define elementos que indicam a estrutura base do documento, como o "body" ou o "html". O *Text Module* define elementos de apresentação de texto, como o "h1". O *Hypertext Module* possui apenas o elemento "a", usado para especificar *links*. O *List Module* define elementos para criação de listas, como o "dl" e o "ol". Esses quatro Módulos compõem o *Core Modules*.

O *Presentation Module* possui elementos, como o "b" e o "hr", que servem para estilizar o texto contido em uma apresentação BML. O *Edit Module* possui os elementos "del" e "ins", usados para editar um documento BML. O *Bi-directional Text Module* define o elemento "bdo", que controla a direção em que um texto é apresentado. Esses três módulos compõem o *Text Extention Modules*.

Extension Modules possui dois módulos, o *BML Module* e *Basic BML Module*, e ambos definem características adicionais à linguagem.

Basic Forms Module and Forms Module oferecem elementos usados em procedimentos de coleta de dados, que são feitos comumente em cadastros. Exemplos são o "input" e o "textarea". *Basic Table Module and Tables Module* oferecem elementos usados para a criação de tabelas, como o "table". A inserção de imagens é possível através da *tag* "img", pertencente ao *Image Module*. O *Client-side Map Module* oferece suporte ao mapeamento de imagens presentes no cliente, enquanto que o *Serverside Map Module* faz o mesmo para imagens presentes no servidor.

Object Module, *Frame Module* e *Iframe Module* definem elementos para inserir, respectivamente, objetos genéricos, frames e iframes nas apresentações.

O *Target Module* possui o elemento "target" que é usado para informar onde um determinado recurso será apresentado. Eventos, como o "onclick", são lidados por elementos contidos no *Intrinsic Events Module*. Meta-informação pode ser adicionada com o uso da *tag* "meta" pertencente ao *Metainformation Module*.

O *Scripting Module* propicia o controle e a descrição do comportamento de *scripts*. O *Style Sheet Module* oferece elementos para definir *style sheets* e o *Style Attribute Module* define o atributo "style". O *Link Module* permite referenciar elementos externos através do uso da tag "link". O elemento "base", especificado pelo *Base Module*, permite definir o base URI. *Name Identification Module* especifica o atributo "name", que, atualmente, foi substituído pelo atributo "id".

A Figura 13 sintetiza os módulos suportados e os não suportados, representados pelos símbolos "+" e "-" respectivamente:

Módulo	Suportado
Structure Module	+
Text Module	+
Hypertext Module	+
List Module	+
Presentation Module	+
Edit Module	-
Bi-directional Text Module	-
BML Module	-
Basic BML Module	+
Basic Forms Module	+
Forms Module	-
Basic Table Module	+
Tables Module	-
Image Module	+
Client-side Map Module	-
Serverside Map Module	-
Object Module	+
Frame Module	-
Iframe Module	-
Target Module	-
Intrinsic Events Module	+
Metainformation Module	+
Scripting Module	+
Style Sheet Module	+
Style Attribute Module	+
Link Module	+
Base Module	+
Name Identification Module	-

Figura 13: Síntese dos módulos suportados pela BML.

Nos módulos suportados, existem diversos elementos que não devem ser usados. A tabela com essas restrições pode ser vista em (B24 Appendix 5: 773). Isso modela a especificação existente para atender às restrições de hardware oferecidas pelos dispositivos portáteis. Entretanto, caso a implementação ou as restrições de um determinado sistema sejam diferentes do previsto, é possível limitar ainda mais ou estender as funcionalidades suportadas.

Assim como os módulos, atributos, elementos CSS, interface DOM e objetos de *browser* também são restringidos. As listas completas das restrições estão, respectivamente, em (B24 Appendix 5: 783), (B24 Appendix 5: 789), (B24 Appendix 5: 809) e (B24 Appendix 5: 833).

A linguagem de script oferecida pela BML é baseada no ECMAScript definido em (ECMA-262). O seu relacionamento com a interface DOM está de acordo com o Apendix E do DOM *level* 1 *Specifications*.

Uma classificação de acesso sobre os *scripts* é feita a fim de manter a segurança e a integridade das aplicações. São definidas duas classes, a "A" e a "B". A primeira classifica *scripts* que tem acesso completo e a segunda que possuem acesso limitado. Nesse último caso, não é permitido o uso de funções que possam prejudicar o receptor, como as de sistema e as de gerência. A linguagem de script usada na BML também possui uma série de restrições para a execução em dispositivos portáteis. Restrições quanto ao uso de memória são descritas na Figura 14.

Item	Maximum value	Remarks
Length of a symbol name character string	255 bytes	
Function arguments	255	
Local variables	255	
Total length of all character strings	65536 bytes	The total length of strings (including evaluated values of string equations, string constants, string variables) and symbol names.
Instances of objects	256	Total number of instances of Object, Number, String, Boolean, Array, Date, BinaryTable and, Function.
Properties of one instance of one object	256	Maximum number of instances of Object, Number, String, Boolean, Array, Date, BinaryTable or, Function. For Array, the number of properties that correspond subscripts are excluded.
Elements of one array	1024	
Nest levels with function for invoking	16	Including functions invoked through event handlers.
Total number of properties of all objects	4096	The total number of properties to which the "properties of one instance of one object" restriction applied. Including number of properties of activation object and arguments object. Excluding built-in properties of global object (built-in functions, built-in objects, extended functions for broadcasting, extended objects for broadcasting) and properties of host object.
Global variables	64	
Function definitions	64	Any function is defined globally. Excluding event handlers.
Work memory for ECMAScript	8192 steps	Based on the computing method defined in ANNEX C, APPENDIX 2.

Figura 14: Restrições de memória para os scripts BML. Retirado de (Apêndice 5)

Outras restrições são:

- Precisão de no máximo 32 bits para números.
- Proibição do uso de *floats*.
- Não é permitido o uso de objetos estáticos com funções matemáticas.
- Restrições para a conversão dinâmica (*Casting*) de tipo.
- A interpretação de *scripts* em *run-time* não deve ser suportada.
- *strings* devem ser codificados usando-se a codificação EUC-JP ou Shift-JIS e uso de Unicode é restringido

- Operações com data devem ser feitas usando-se a faixa de 31/03/1900 até 28/02/2100.

Maiores detalhes das restrições sobre os scripts e as suas conseqüências podem ser vistos em (Appendice 5: 856-861).

6.4.2 Implementações

O Middleware FSDTV foi desenvolvido no Japão para o ISDB-T tanto para dispositivos portáteis quanto fixos. Nos dispositivos portáteis, a exibição das aplicações pode ser feita com o uso do NetFront DTV Solution, que é um *browser* desenvolvido pela Access especificamente para oferecer TV Digital em dispositivos portáteis. Sua plataforma tem suporte a BML ou HTML 4.0, bem como ECMAScript. NetFront ainda oferece uma ferramenta para a criação de conteúdo, o NetFront Browser DTV Profile BML/HTML Edition SDK, mas o FSDTV não especifica se isso vem incluso no seu pacote. A arquitetura do *browser* pode ser vista na Figura 15 e na Figura 16.

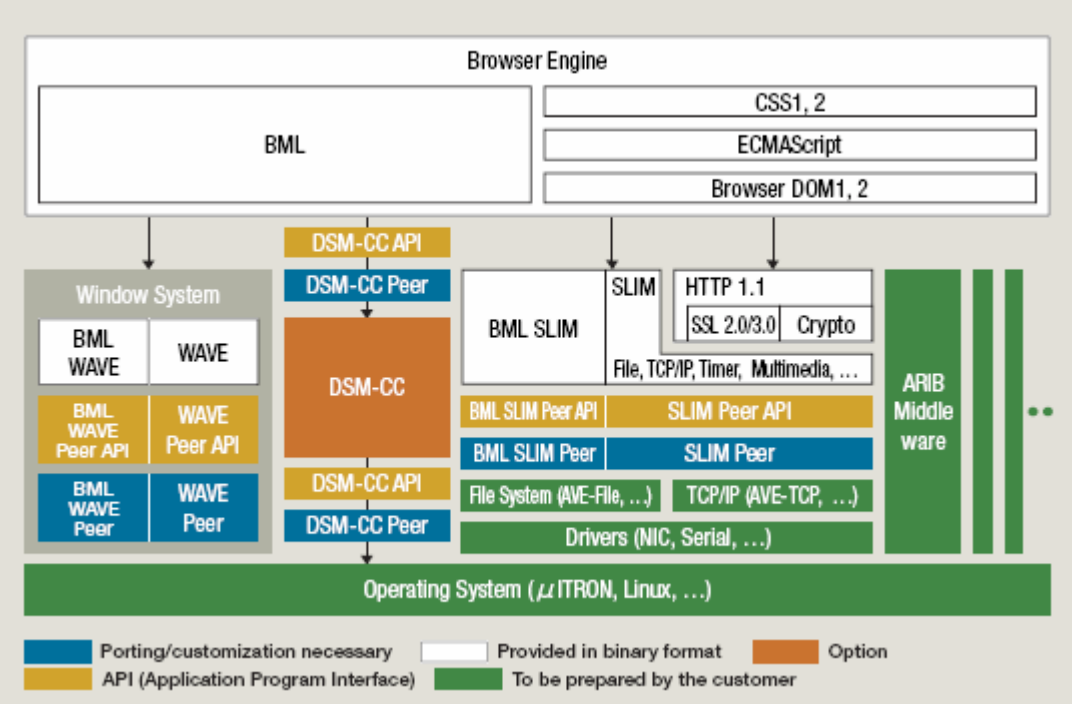

Figura 15: Arquitetura NetFront/BML Retirado de (NetFront DTV Brochure)

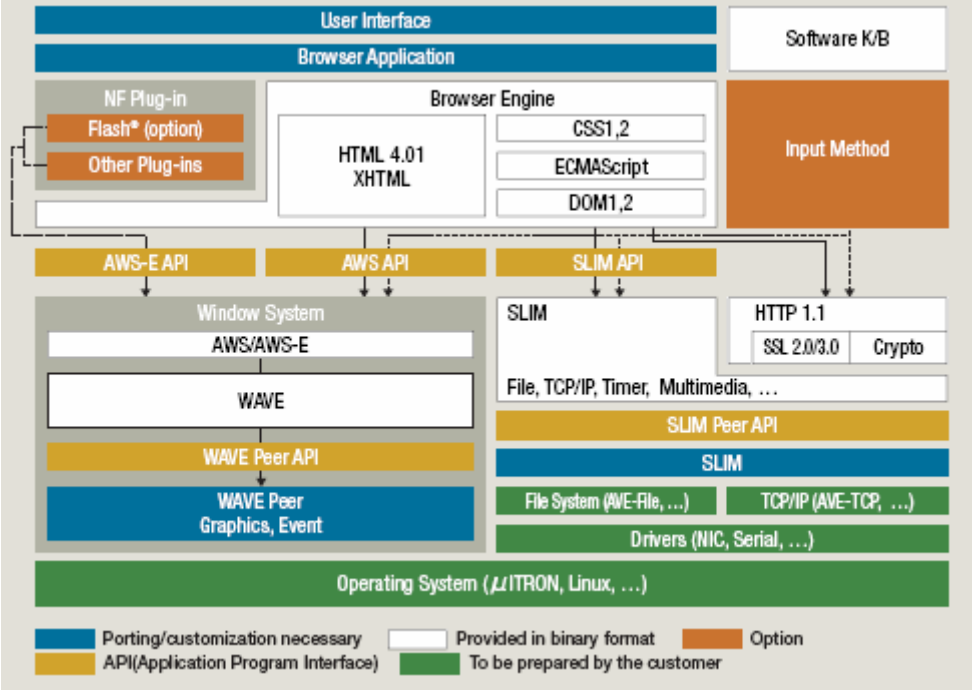

Figura 16: Arquitetura NetFront/HTML Retirado de (NetFront DTV Brochure)

Outros produtos que possuem suporte ao ISDB-T foram encontrados, mas eles não oferecem o *middleware* definido pela ARIB. Ao invés disso, apresentam outras formas de se prover o conteúdo. Esses produtos são:

- InterVideo's iMobi Mobile DTV, que oferece apenas ESG.
- NeoMagic Mobile TV Solution, que oferece apenas uma ferramenta de autoria, o MiniMagic 6+ Development Board.
- SmartVision™ Mobility platform, que oferece suporte a ESG, VoD e PVR.

6.5 GMIT

GMIT é uma empresa de consultoria relacionada ao padrão DVB-H, e é a principal provedora do HisTV. Outras empresas que suportam o HisTV são: BenQ Móbile, T-Systems Media Broadcast, Neva Media e Siemens. HisTV é um *framework* de aplicação usado como *middleware* de TV Digital para dispositivos portáteis. Maiores detalhes são vistos na próxima subseção.

6.5.1 HisTV

HisTV é direcionado para apresentação de conteúdo televisivo em Handhelds. A sua primeira versão surgiu em 2005 com o nome MiTV. Em 2006, o primeiro cliente HisTV foi apresentado. Não existe ainda uma especificação fechada para esse *framework*, que ainda está em fase de *drafting*. A última versão do documento foi lançada em 7. Nov.2006. A Figura 17 exemplifica a sua arquitetura.

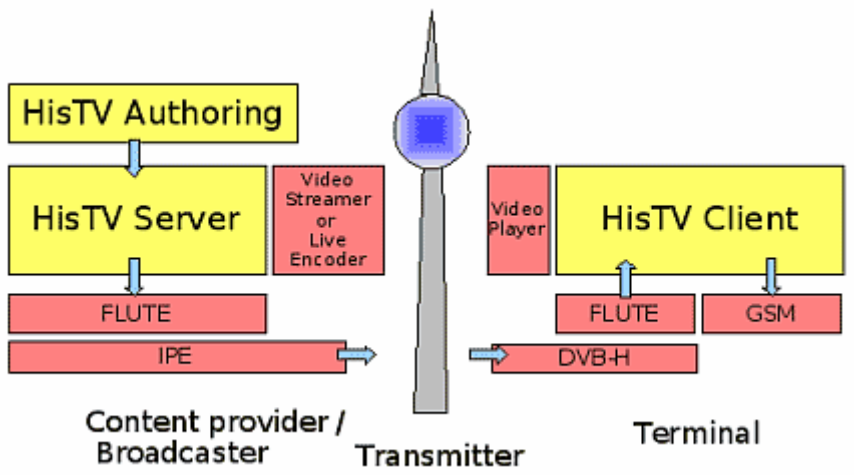

Figura 17: Arquitetura do HisTV Retirado de (HisTV Main)

No HisTV, como podemos observar pela figura, a transmissão é feita via um canal de broadcast definido pelo padrão DVB-H, o FLUTE (**RFC 3926, 2004**). O canal de retorno é implementado com o uso de mensagens SMS ou via protocolo HTTP (Skrodzki, 2006). Devem ser usados *timestamps* no formato NTP (RFC 1305) no fluxo de vídeo, que são transportados com base no protocolo RTP/RTCP (RFC 1889). Todas essas tecnologias estão incorporadas no HisTV, enquanto que exibidores de mídias, como o componente "*video player*" presente na figura, precisam ser fornecidos. As setas indicam componentes incorporados, enquanto que os outros são os que precisam ser fornecidos, e é isso que define o HisTV como um *framework*. A autoria, por sua vez, é totalmente independente das tecnologias utilizadas. Isso significa que o desenvolvimento de uma aplicação de TV Digital pode ser feito sem o conhecimento da tecnologia usada pelo serviço de transporte, dando ao HisTV a sua característica de *middleware*. O padrão, na verdade, define apenas uma linguagem de representação, enquanto que as características de transporte incorporadas são apenas referenciadas.

Para entender como deve ser feito o desenvolvimento de uma aplicação HisTV, é melhor separá-lo em três etapas: a construção do *layout* da tela, o projeto da máquina de estados e a produção dos objetos.

A tela do dispositivo é subdividida em 255 campos, chamados de ifields. Todo o funcionamento da aplicação está relacionado a eles. Cada um representa uma porção maior ou menor da tela. A sua disposição pode ser feita de qualquer forma desejada, e ifields menores podem estar contidos em ifields maiores. Contudo, existem os ifields especiais, que possuem tamanhos e disposição na tela limitados.

Existem dois *layouts* de tela permitidos, *landscape* (320x204) e *portrait* (240x320). A Figura 18 demonstra a organização de uma tela com o uso dos ifields no modo *landscape*.

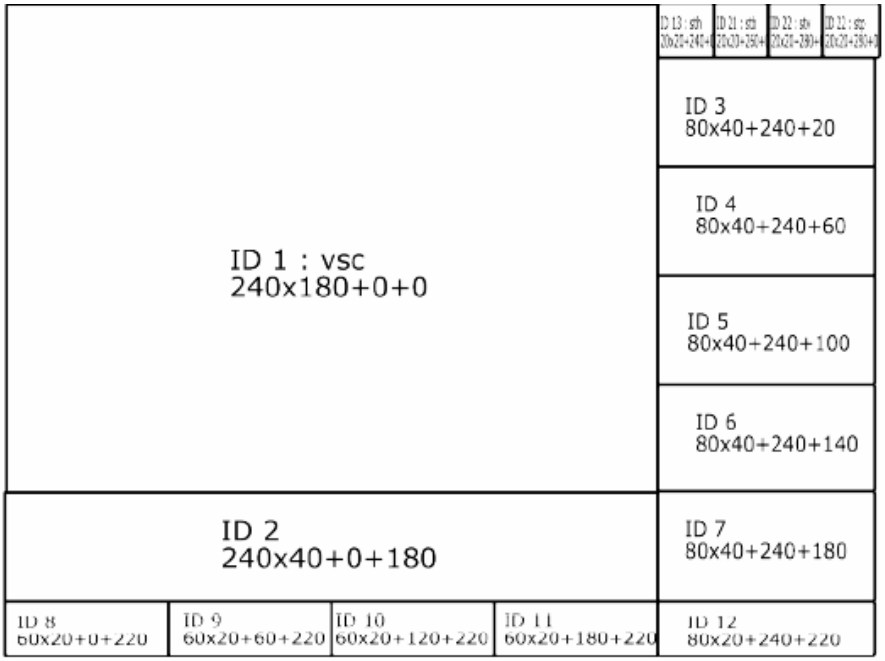

Figura 18: Tela de Handheld organizada com Ifields. Retirado de (Skrodzki, 2006).

O conteúdo de um ifield é chamado de HisTV Object, que é um objeto da aplicação e é um arquivo. Por exemplo, o arquivo "lápis.jpg", que possui uma imagem, pode ser associado a um ifield, tornando-se, assim, um HisTV Object. Essa associação consiste na alteração do nome do arquivo, fato que será explicado mais a frente. O conjunto de todos os ifields, com os seus respectivos conteúdos, compõem uma cena.

É definida uma série de ifields especiais, que, como já foi mencionado anteriormente, podem possuir limitação de tamanho e disposição na tela. Exemplos são: *Video Screen*, que mantém o *aspect ratio* do vídeo, *Sidebar List*, que mostra uma lista de elementos com scroll, e *Embedded Browser*, que abre um *browser* no ifield. Cada um deles possui uma identificação especial composta de três caracteres. As identificações daqueles mostrados como exemplo são: vsc, sls e emb. A lista completa desses campos e suas descrições pode ser vista em (Skrodzki, 2006: 14-15).

Uma aplicação HisTV é baseada em cenas controladas por uma máquina de estados. O conteúdo de um ifield pode ser configurado diferentemente para cada estado, podendo, assim, compor cenas diferentes em cada um deles. Um problema, segundo (Skrodzki, 2006), é que implementar esta máquina é uma tarefa complicada e, portanto, deve ser feita com cuidado.

As interações são feitas com um clique no ifield e podem ser de dois tipos: que alteram o estado da máquina de estados ou que enviam uma mensagem pelo canal de interatividade. As interações podem lançar mão do uso de contadores para mudar o estado de uma apresentação. Por exemplo, um contador pode armazenar o número de cliques feitos em um ifield, se esse valor ultrapassar um N qualquer, o estado muda.

A sincronização é sempre feita entre os ifields e um vídeo. Para tal, define-se um *timestamp* nos objetos dos ifields, que são comparados com aqueles inseridos no vídeo. Os objetos que possuírem o seu tempo maior ou igual ao do vídeo sincronizado são apresentados. A idéia consiste em apresentar o conteúdo do campo até que o seu *timestamp* seja igual ao do vídeo. É importante salientar que o conteúdo de um ifield apresentado se mantém na tela, ou seja, apenas a substituição por um outro conteúdo

limpa o antigo. Um mesmo ifield pode estar sincronizado com um vídeo em tempos diferentes, exemplo: (ifield 1; tempo = 10s; conteúdo = ”A”) e (ifield 1; tempo = 15s; Conteúdo = ”B”). Os campos ifield que possuírem tempos mais recentes são apresentados primeiro. Sendo assim, “A” será apresentado até que o vídeo alcance dez segundos, e, depois que isso ocorra, “B” será apresentado até que o vídeo alcance quinze segundos.

Unindo os conhecimentos apresentados até agora, podemos entender como o conteúdo pode ser formado a fim de conceber a aplicação. Cada objeto no HisTV possui um nome composto da seguinte forma: <id>_<starttime>_<state>_<extension>, onde <id> é o id do ifield, <starttime> é o timestamp do objeto, <state> é o estado do qual esse objeto é apresentado no ifield e <extension> é o tipo do objeto. É importante salientar que o objeto e o arquivo, nesse contexto, são a mesma coisa. O desenho dos ifields na tela é feito com uso de objetos do tipo .lal e .lap, usados para os *layouts landscape* e *portraint* respectivamente. Sendo assim, o objeto de nome 1_0000000000000002_3.lal define a posição e tamanho do ifield 1, com *timestamp* 2, no estado 3 em modo *landscape*. Para todo objeto do tipo .lal, deve sempre existir um outro .lap equivalente. Dessa forma, a aplicação funcionará em ambos os *layouts*, *landscape* ou *portraint*. Dentro do arquivo .lal temos uma linha no formato x,y,width, height;<special_type>. A posição do ifield é determinada pelos campos x,y,width e height, e o <special_type> especifica o seu tipo especial. <special_type> pode ser vazio, indicando que o ifield não é de um tipo especial, ou uma das identificações de tipo especial, que já foram mencionadas. Por exemplo, “0,0,50,50;vsc” define um ifield especial do tipo Vídeo Screen com tamanho cinqüenta. O arquivo .lap possui apenas o conjunto de campos que definem tamanho e posição, o tipo especial é obtido do seu par .lal. Existe uma série de outros objetos que determinam o conteúdo do ifield. Um objeto 1_0000000000000002_3.txt, por exemplo, preenche, com o seu conteúdo textual, o ifield de *layout* especificado pelo par 1_0000000000000002_3.lal e 1_0000000000000002_3.lap. Existem uma série de objetos permitidos no HisTV, como texto (.txt), fonte (.fcl) ou cor (.col), que são especificados em (Skrodzki, 2006).

Como podemos ter um ifield apresentado em diversos estados e, para cada conjunto (estado, ifield) podemos ter um starttime diferente, o máximo de duplas descritoras de layout .lal e .lap é a combinação desses itens. Dessa forma, se uma aplicação possuir cinco estados, dez ifields e dois starttimes, precisaremos de, no máximo, cem duplas desses arquivos descritores de *layout*. Isso não está claro em (Skrodzki, 2006), mas, dado funcionamento descrito, é o que se pode concluir. Existe um tipo de objeto, o State List (stl), que informa quais estados são válidos para uma determinada aplicação. Todos os objetos que possuírem estados inválidos são ignorados. Obviamente, é inviável para um ser humano criar uma estrutura nesta sintaxe (linguagem) de transferência definida. A idéia é que a ferramenta de autoria a crie automaticamente. Nenhuma linguagem é descrita em (Skrodzki, 2006), mas é mencionada a existência de um *script* Perl, chamado de histv_asfa.pl, que transforma um documento CSV (*Coma Separated Value*) na estrutura de uma aplicação HisTV. CSV é um arquivo em um formato onde as informações são separadas por vírgulas, e é freqüentemente utilizado em planilhas eletrônicas. Entretanto, nada é dito sobre como esse documento CSV deve ser criado. Com esse ambiente de autoria, o HisTV parece estar mais próximo de se ser um *middleware* declarativo do que procedural, apesar disso não estar claro nas documentações disponíveis.

6.5.2 Implementações

Segundo o seu site oficial (HisTV Main), a T Systems e a Neva Media possuem serviços baseados no HisTV, enquanto que a Benq e a Siemens possuem serviços baseados na sua versão mais antiga, o MiTV. Essas empresas desenvolvem soluções baseadas no DVB-H e, apesar de não fazerem referência direta à utilização do HisTV, podemos observar que os seus produtos possuem o layout oferecido por ele. Pode-se perceber isso na Figura 19, que é um exemplo de um produto da Siemens.

Figura 19: Produto Siemens utilizando tecnologia HisTV.

6.6 LASeR Interest Group

Seus membros são a Siemens, a Streamezzo, a @net&TV e a ETRI e esse grupo tem como o objetivo o desenvolvimento de um padrão de descrição de cenas, o LASeR, e um formato de agregação, o SAF, para dispositivos portáteis. Antes deles, 3GPP, MPEG e OMA tentaram implementar o MPEG4-BIFS (*Binary Format for Scene*) e um somatório do SVG, DOM, CSS e EcmastScript, a fim de estender o uso de conteúdos multimídia em dispositivos portáteis. Eles chegaram a conclusão que, dessa forma, não era possível atender a todos os requisitos necessários, foi daí que o padrão LASeR e SAF nasceu. Com os dois foi possível oferecer uma arquitetura completa, que compreende a transmissão do conteúdo multimídia e a criação de aplicações interativas voltadas para os dispositivos portáteis. Esse padrão está sendo definido como MPEG4 *part* 20 pelo ISO/IEC/JTC1/SC29/WG11 (LASeR). O seu desenvolvimento iniciou-se em 2004, o último *draft* foi lançado em 2005 e agora está em fase de emenda para depois se transformar em um padrão internacional. A Figura 20 mostra a sua arquitetura.

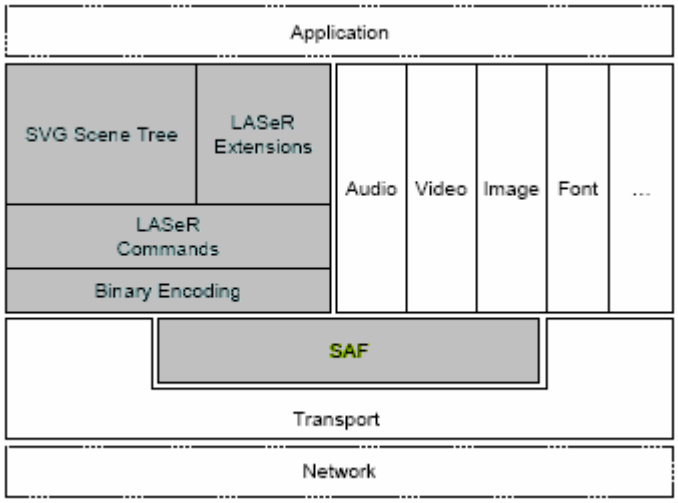

Figura 20: Arquitetura LASeR e SAF. Retirado de (ISO 14496-20)

SAF (*Simple Aggregation Format*), definido pelo LASeR, é usado para transportar os dados, como o áudio e o vídeo. Ele foi criado pela necessidade de se ter uma solução mais leve e simples para o transporte, como é argumentado em (ISO 14496-20).

LASeR (*Lightweight Application Scene Representation*), como o próprio nome já diz, é o padrão que define a representação por cenas de uma aplicação multimídia, direcionada especificamente para o ambiente portátil. Oferece, para esse fim, uma linguagem declarativa. O uso do LASeR é totalmente independente do tipo de transmissão feita, entretanto, o padrão recomenda o uso do SAF para esse objetivo. Dessa forma, LASeR é uma proposta de padrão de *middleware* declarativo para dispositivos portáteis e será estudado de forma mais aprofundada a seguir . Nenhuma implementação sua foi encontrada.

6.6.1 LASeR

LASeR oferece um formato binário para codificação e atualização de cenas 2D. Esse formato é um *superset* do SVG *(Scalable Vector Graphics*) Tiny, que é apropriado para dispositivos portáteis. O conceito por trás do SVG é oferecer uma maneira de escalar tanto a apresentação, possibilitando a sua execução em dispositivos com mais ou menos recursos, quanto o uso de recursos gráficos. SVG é uma linguagem declarativa padrão W3C.

É possível representar três tipos de objetos gráficos diferentes: formas geométricas desenhadas a partir de vetores, objetos multimídia (vídeo, áudio etc), e texto. O SVG é extensível, ou seja, as suas funcionalidades podem ser complementadas. De forma a atender aos seus requisitos, o LASeR realiza a sua própria extensão desse padrão, oferecendo mais funcionalidades que o original.

Um documento SGV é um XML que possui a *tag root* "sgv". Todas as outras *tags* existentes ficam contidas nela. Apenas uma tag "sgv" pode existir em um documento, enquanto que outras *tags* podem ser duplicadas.

Para criar estruturas, usa-se o elemento "g", que cria grupos de elementos. Um grupo formado por "g" pode estar contido em um outro recursivamente. Isso permite aninhamentos de grupos e elementos, possibilitando a criação de estruturas que

poderão ser usadas em animações ou reusadas em outros documentos. Um elemento que não está contido em um "g" é considerado um grupo de um único elemento.

O elemento "defs" funciona de forma similar ao "g", mas, no seu caso, os elementos contidos nele não são renderizados. O propósito é que ele agrupe elementos que serão referenciados pelo mesmo ou por outro documento.

Para elementos gráficos e grupos do tipo "g", é possível definir um título e uma descrição, "title" e "desc" respectivamente. O primeiro é usado como uma *hint* e o segundo não é renderizado. Esses dois elementos funcionam como comentários dentro de um documento SGV e, assim, facilitam a sua leitura e estruturação.

A *tag* "image" renderiza, na posição da tela definida pelos atributos x e y, uma imagem carregada do arquivo especificado pelo atributo "xlink:href".

Para criar apresentações com longa duração, é importante o uso do "discard", que exclui objetos da apresentação, liberando, assim, recursos. O atributo *begin* é definido a fim de determinar quanto tempo deve-se esperar para que o recurso referenciado seja liberado. O atributo xlink:href, por sua vez, faz referência ao objeto que será liberado. Se nenhuma referência for indicada, "<discard>" referenciará o seu objeto pai na hierarquia DOM.

O reuso é feito usando-se o elemento "use". O seu funcionamento dá impressão de que um clone completo do elemento referenciado pelo atributo xlink:href foi criado, embora isso não seja verdade. Para determinar as coordenadas onde o elemento reusado deve ser apresentado, outros dois atributos são definidos: o x e o y.

Diferentes visualizações de um mesmo documento podem ser oferecidas via o uso do elemento "switch" e dos seus atributos: requiredFeatures, requiredExtensions, systemLanguage, requiredFomats e requiredFonts. Cada filho do elemento "<switch>" possui um conjunto desses atributos definidos, que são verificados no momento da renderização. O primeiro filho que for avaliado como verdadeiro para todos os atributos é renderizado. Um nó que possuir "systemLanguage=en", por exemplo, só será apresentado caso a linguagem do sistema for o inglês.

SGV determina uma série de formas geométricas padrões que podem ser utilizadas para a criação das apresentações, são elas: retângulos, círculos, elipses, linhas, polilinhas (conjunto de linhas) e polígonos; representados, respectivamente, pelas tags "rect", "circle", "ellipse", "line", "polyline" e "polygon".

A apresentação de textos é feita usando-se as *tags* "text" e "textarea". É possível definir as coordenadas da sua apresentação da mesma forma como é feito com gráficos ou qualquer outro elemento do SGV. "text" apresenta uma única linha de texto e não realiza quebra de linha, automática ou não. Para esse tipo de facilidade, deve-se usar o "textarea". Além das coordenadas, é possível definir fontes, orientação do texto e outros atributos que modelam a forma como o texto aparecerá na tela. Também é possível definir se um determinado campo de texto pode ser editado pelo usuário. As fontes são codificadas separadamente das cenas, e todos os elementos definidos pelo SGV para esse objetivo não são suportados no LASeR. (ISO 14496-20) sugere o ISO/IEC 14496-18 para a realização desta codificação.

Três tipos diferentes de objetos de mídia são suportados: áudio, vídeo e animação, que são representados, respectivamente, pelas *tags* "audio", "video" e "animation". Esses objetos de mídia possuem o seu próprio *timeline* embutido. "áudio" referencia, através do atributo xlink:href, um arquivo de áudio do tipo definido pelo atributo "type". De forma similiar, "video" faz referência a um arquivo de vídeo, a diferença é

que atributos visuais podem ser definidos, como as coordenadas. Qualquer tipo de manipulação com o vídeo, como animações ou mudanças no seu tamanho, deve ser feita com cuidado, pois tratam-se de aplicações desenvolvidas para dispositivos portáteis, onde os recursos são limitados.

A *tag* "animate" referencia e aplica uma animação em um elemento qualquer. Essa referência é feita de forma similar àquela da *tag* "use", mas, ao contrário dela, o objeto é completamente duplicado.

Os objetos são iniciados e encerrados nos tempos apropriados seguindo o modelo temporal do Smil (Smil 2.1, 2007), que é adaptado pelo LASeR (ISO 14496-20: 7). Basicamente, todos os objetos possuem uma duração (dur) e tempos de *begin* e *end*. A duração é representada em segundos. Os valores do *begin* e do *end* podem ser representados por valores temporais, bem como por eventos, ou uma associação dos dois. Exemplo: "begin= 19s; objeto1.end", onde o início do objeto será no décimo nono segundo ou quando o objeto1 terminar, o que acontecer primeiro.

A Interatividade também é suportada pelo SGV, e pode ser feita via cliques, teclas pressionadas ou *hyperlinks* selecionados. Tais interações podem ativar *browsers*, eventos, *scripts*, ou até mesmo animações. Existe uma série de eventos que podem ser tratados e que são descritos em (ISO 14496-20: 9), que foram adaptadas de (SVG Tiny, 2005b). Alguns dos mais comuns são: "click", "begin", "end" e "mouse down".

A animação possui o mesmo funcionamento que é especificado no Smil, mas com extensões feitas pelo SVG. Portanto, os tipos de animação suportados são: "animate", "set", "animateMotion" e "animateColor", que são procedentes do Smil, e "animateTransform", "path", "mpath", "KeyPoints", "rotate" e "discard", que são extensões. O "animateTransform" permite realizar animações sobre atributos do tipo *transform*. O "discard" já foi explicado anteriormente, e as outras extensões são atributos adicionais que podem ser usados no "animateMotion" a fim de oferecer animações extras, como a de rotação.

LASeR define cabeçalhos para realizar a configuração da codificação de uma cena. Uma cena pode ser particionada em segmentos incrementáveis. Assim, uma partição pode ter a definição de uma cena e as outras partições podem complementá-la. O LASeRHeader possui atributos que especificam o conjunto completo de uma cena, enquanto que o LASeRUnitHeader tem atributos relativos à uma unidade, ou segmento, de uma determinada cena.

É possível, também, realizar alterações em uma cena através dos seguintes comandos: *NewScene*, *RefreshScene*, *Insert*,*Delete*,*Replace*, *Add*, SendEvent, *Save*, *Restore* e *Clean*.

NewScene cria uma cena nova de tempo igual a zero e aborta a execução de qualquer outra que esteja sendo apresentada no momento. *RefreshScene* atualiza uma cena que está sendo apresentada. *Insert* insere um novo elemento na referência indicada pelo seu atributo "ref". *Delete* exclui um elemento referenciado pelo seu atributo "ref". *Replace* substitui um elemento ou um atributo de um elemento, onde "ref" indica o elemento substituído ou aquele que terá um dos seus atributos alterados. A *tag* "attribute" indica o nome do atributo a ser alterado, e o seu novo valor estará contido na *tag* "value" do *Replace* ou no valor de um atributo de qualquer outro elemento do documento. Esse elemento é referenciado pela *tag* "operandElementId" e o seu atributo é indicado pela *tag* "operandAttributeName". *Add* tem o comportamento igual ao do *Replace*, mas cria ao invés de realizar substituições. SendEvent envia um evento para

algum elemento referenciado por "ref". *Save*, *Restore* e *Clean* salva, recupera e remove, respectivamente, informações persistentes de uma cena.

LASeR também define uma linguagem de *script* embutida, igualmente baseada no SGV. O script é escrito entre as *tags* "<script>" e "</script>". A *tag* "<script>" possui um parâmetro "*type*" que define a linguagem de *script* usada. No LASeR, são dois os tipos de linguagens disponíveis, a "application/ecmascript" e a "application/laserscript". O primeiro tem o funcionamento definido pelo SVG, que é similar ao de qualquer *browser* HTML. Por exemplo, um *script* pode ser usado em qualquer um dos eventos definidos na aplicação. A Figura 21 ilustra isso:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="6cm" height="5cm" viewBox="0 0 600 500"
     xmlns="http://www.w3.org/2000/svg" version="1.1">
  <desc>Example script01 - invoke an ECMAScript function from an onclick event
  </desc>
  <!-- ECMAScript to change the radius with each click -->
  <script type="text/ecmascript"> <![CDATA[
    function circle_click(evt) {
      var circle = evt.target;
      var currentRadius = circle.getAttribute("r");
      if (currentRadius == 100)
        circle.setAttribute("r", currentRadius*2);
      else
        circle.setAttribute("r", currentRadius*0.5);
    }
  ]]> </script>
  <!-- Outline the drawing area with a blue line -->
  <rect x="1" y="1" width="598" height="498" fill="none" stroke="blue"/>
  <!-- Act on each click event -->
  <circle onclick="circle_click(evt)" cx="300" cy="225" r="100"
          fill="red"/>
  <text x="300" y="480"
        font-family="Verdana" font-size="35" text-anchor="middle">
    Click on circle to change its size
  </text>
</svg>
```

Figura 21: Exemplo do uso de script no LASeR. Retirado de (Scripting - SVG 1.1)

O outro tipo, o "application/laserscript", define *scripts* que serão usados nas atualizações de cenas.

Dois atributos extras, além do "*type*", são especificados para os *scripts*: o "*begin*", usado para determinar o momento em que o *script* deve entrar em funcionamento, e o "*enabled*", de valor booleano, que determina se o *script* é passível de ativação ou não.

6.7 3GPP

Sigla para *3rd Generation Partnership Project*, essa organização é o resultado de um acordo de colaboração firmado entre empresas de padronização em telecomunicações ao redor do mundo. Foi fundada em 1998 e seus integrantes, atualmente, são: ARIB, CCSA, ETSI, ATIS, TTA e TTC. O padrão MBMS (*Mobile Broadcast/Multicast Service*) de

transmissão de TV Digital para dispositivos portáteis foi desenvolvido por essa organização.

O MBMS é adequado para transmissão de conteúdos leves, sendo possível prover alguns serviços de *streaming* em tempo real. Entretanto, para *streamings* pesados em redes complexas e densas, outras soluções se mostram mais adequadas, como o DVB-H, que é sugerido para esse fim em (MBMS White Paper).

A primeira versão do MBMS tornou-se disponível em 2001 e o último release foi disponibilizado no final de 2006. Espera-se que em 2008 surjam os seus primeiros terminais MBMS.

O serviço usado pelo padrão é do tipo IP *datacast* (IPDC) e pode ser oferecido usando GSM ou UMTS. A sua arquitetura permite tanto o uso de Broadcast como o de *Multicast*, sendo que o segundo tem o objetivo de oferecer serviços pagos, enquanto que o primeiro ofereceria serviços gratuitos. É definido, ainda, um serviço de transmissão de dados a uma taxa muito baixa de bits, que pode ser utilizado para o envio de mensagens instantâneas.

O controle da apresentação de uma aplicação deverá ser feito usando uma padronização do DIMS (*Dynamic and Interactive Multimedia Scenes*) (3GPP Feature), que está especificada em **(**3GPP TS 26.142 v1.0.0 (2006-11)). Em outras palavras, o *middleware* do MBMS para TV Digital é o DIMS, definido pelo 3GPP.

6.7.1 DIMS

DIMS define um conceito onde a representação das aplicações multimídia é feita sob a forma de cenas interativas. Sempre que for usado o termo DIMS, nesta seção, estará sendo referenciado aquele especificado pelo 3GPP no documento (3GPP TS 26.142 v1.0.0 (2006-11)). Como DIMS encontra-se em versão de rascunho, pouca coisa ainda está definida e alterações substanciais podem ocorrer ao longo do tempo.

A arquitetura proposta pelo padrão pode ser observada na Figura 22, que foi retirada de (3GPP TS 26.142 v1.0.0 (2006-11)).

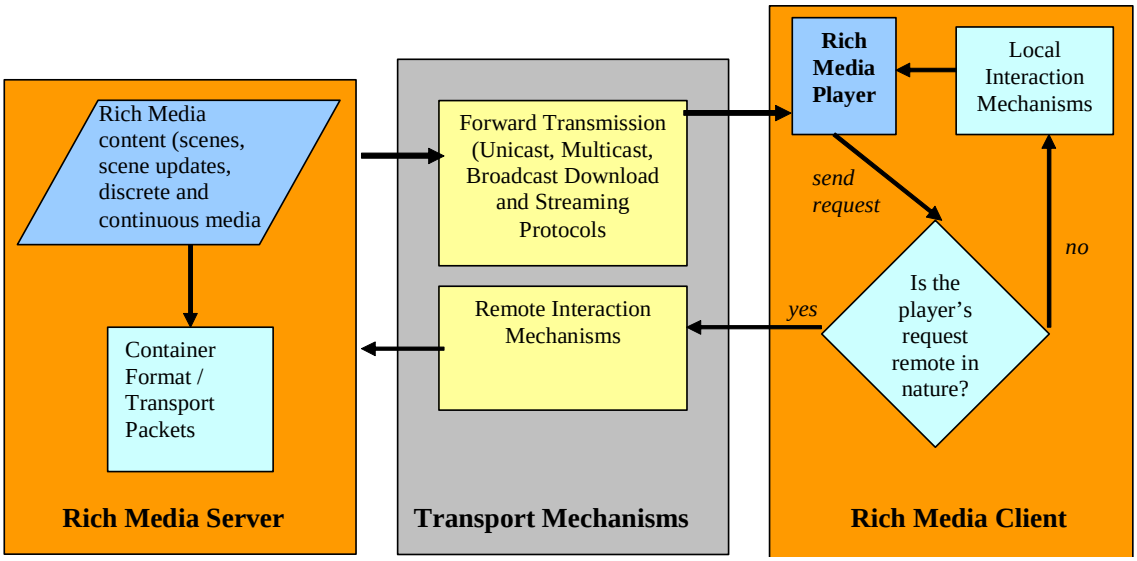

Figura 22: Arquitetura geral do sistema de mídia. Retirado de (3GPP TS 26.142 v1.0.0 (2006-11))

O conteúdo hipermídia é composto por cenas, atualizações de cenas e por mídias discretas ou contínuas, como as imagens e os vídeos, respectivamente. A composição é feita no servidor e transmitida para o cliente. Pode-se observar na figura que é possível realizar interações locais no cliente, bem como remotas. O primeiro tipo de interação é feita com o uso de eventos DOM , SVG e XML. O segundo tipo utiliza um dentre três tipos diferentes de mecanismos: abrindo uma URL, estabelecendo conexão via *socket* com o servidor ou usando métodos de getURL ou postURL via http.

As cenas podem ser atualizadas ao longo do tempo de forma parcial ou completa. Isso permite que uma nova cena seja montada aos poucos ou de uma só vez.

A descrição de uma cena é feita usando-se SGV Tiny (SVG Tiny, 2005a) com extensões próprias e comandos retirados de (LASeR: 9). Apenas três extensões foram definidas em (3GPP TS 26.142 v1.0.0 (2006-11)): Alinhamento de gráficos, *Full Screen Video*, para definir o vídeo em tela cheia, e *Full Screen svg*, para indicar que a cena sgv deve ser renderizada em tela cheia. A Seção 5.5.1 descreve a representação de cenas com o SGV e os comandos do LASeR.

DIMS também oferece a possibilidade de se criar *scripts* em seus documentos hipermídia. A linguagem de *script* utilizada é aquela especificada em (SVG Tiny, 2007a), que pode ser usada em conjunto com a ECMAScript Mobile Profile, mas nenhuma informação adicional ou mais aprofundada sobre esse assunto é dada em (3GPP TS 26.142 v1.0.0 (2006-11)).

6.8 DMB

Um projeto nacional desenvolvido pela Coréia do Sul tentou implantar o padrão de transmissão DAB (Eureca 147 Digital Audio Broadcast) em seu país. Esse padrão estava, na época, sendo utilizado por países da Europa, como Alemanha, Inglaterra e Espanha. Durante esse processo de implantação, o padrão foi aperfeiçoado para suportar vídeo e outras mídias. O resultado disso foi o DMB (*Digital Multimedia Broadcast*), que foi implantado na Coréia do Sul em 2005. Existem duas versões do DMB: o S-DMB, para transmissão via satélite, e o T-DMB, para transmissão terrestre.

O padrão DMB teve o seu primeiro *draft* em 2002, que foi apresentado em 2003 e firmado em junho de 2005 (DMB-PORTAL).

A arquitetura do DMB é composta de três camadas: compressão de conteúdo, sincronização e transporte. MPEG4 BIFS é usado como linguagem descritora de cenas, a sincronização é feita com o uso do MPEG4 SL e o transporte foi implementado com o uso do MPEG2 TS. Essas três tecnologias são aplicadas, respectivamente, nas três camadas descritas. Dessa forma, o *middleware* do DMB é o MPEG4 BIFS.

6.8.1 MPEG4 BIFS

BIFS é usado com ressalvas, apenas o seu core *profile* é utilizado e a descrição gráfica das cenas é feita apenas com nós que realizam tratamento 2D. Nós de tratamento 3D não devem ser suportados. A linguagem define também restrições em relação aos descritores oferecidos pelo BIFS. Descritores são elementos que relacionam os objetos de uma cena com os *streams*. OD, IOD, ES Descriptor, DecoderConfigDescriptor e SLConfigDescriptor são descritores que devem ser utilizados, enquanto que IPI Descriptor Pointer, IPMP Descriptor Pointer e IPMP Descriptor não.

O OD, ou Object Descriptor, é usado para identificar, univocamente, um objeto de mídia de uma cena. O ES_Descriptor descreve as características e requerimentos de um *stream* elementar. O DecoderConfigDescriptor determina o tipo da decodificadora requerida por um *stream*. O SLConfigDescriptor não pode ser encontrado na especificação. A documentação completa dos descritores pode ser encontrada em (ISO/IEC 14496: 134). A estrutura de uma cena BIFS, que deve seguir o que é descrito em ISO/IEC DIS 14772-1: 1997 (ISO/IEC 14496: 134) e é exemplificada na Figura 23 e na Figura 24:

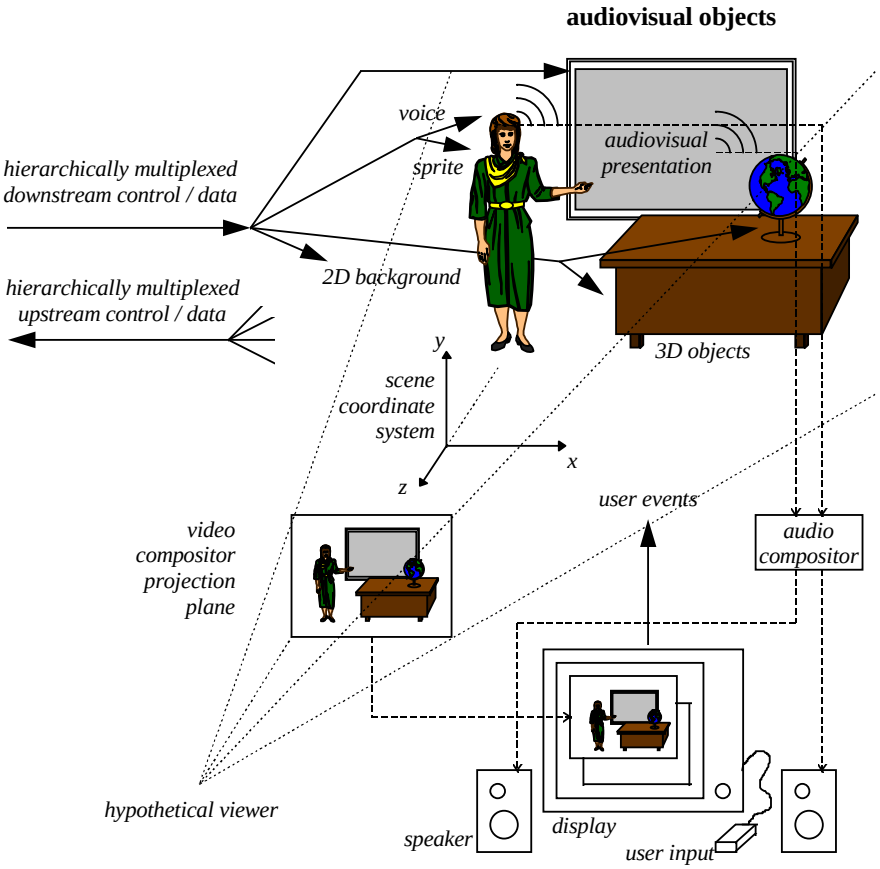

Figura 23: Um exemplo de uma cena multimídia. Retirado de (ISO/IEC 14496-)

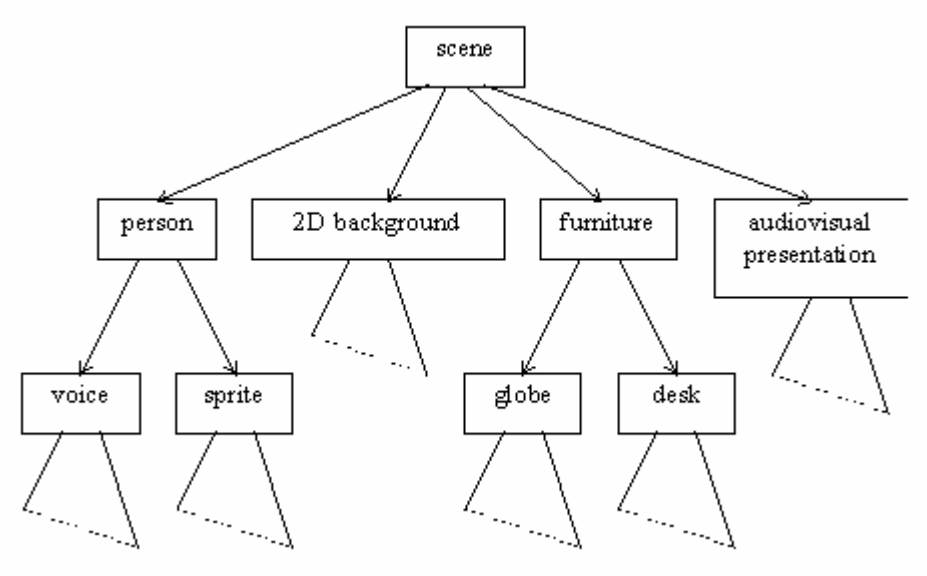

Figura 24: Exemplo da estrutura lógica da cena. Retirado de (ISO/IEC 14496)

Os nós são dispostos de forma hierárquica e possuem uma relação espaço temporal definida pelo seu nó ancestral. Os nós do MPEG4 BIFS usados pelo DMB são

- Background2D, que define um *background* para a cena;
- *Circle*, que desenha um circulo;
- Coorsdinate2D, que define um conjunto de coordenadas 2D usadas em nós geométricos;
- Curve2D, que desenha um polígono;
- DiscSensor, que permite a um objeto realizar rotações sobre o seu eixo;
- *Form*, que permite que um conjunto de objetos seja apresentado com o mesmo espaçamento entre si;
- Group2D é um nó que representa uma cena e agrupa outros nós BIFS;
- Image2D, que adiciona uma imagem 2D em uma cena;
- IndexedFaceSet2D representa uma forma que é uma junção de polígonos em pontos especificados pelo seu parâmetro "coord";
- Inline2D inclui uma cena 2D externa na atual;
- *Layout* alinha os seus nós filhos de acordo com uma série de modos de alinhamento, como letfToRight e *justify;*
- LineProperties define parâmetros para a renderização de linhas 2D;
- Material2D define parâmetros de renderização de formas geométricas 2D;
- VideoObject2D inclui um vídeo na cena;
- PlaneSensor2D permite que um objeto seja arrastado na tela, se o dispositivo permitir;
- Rectangle desenha um retângulo na cena;

- ShadowProperties define parâmetros de sombreamento para alguns tipos de objetos geométricos;
- Switch2D apresenta um ou nenhum dos nós de uma lista de nós;
- Transform2D permite que os seus nós filhos possam sofrer rotação, mudança de tamanho e translação.

Outros nós usados no DMB, mas que não foram descritos pela especificação, são o Position2DInterpolator e o Proximity2DSensor. Uma descrição mais detalhada desses nós de elementos 2D pode ser encontrada em (ISO/IEC 14496: 67).

6.8.2 Implementações

Um dos *middlewares* DBM achados é o da Thin Multimedia (thin multimedia), o TIM's T-DMB *middleware*. Suas aplicações podem ser acionadas com o uso do EPG (ETSI TS 102 818 V1.3.1 (2005-09)/ ETSI TS 102 371 V1.2.1 (2005-09)). A sua Arquitetura pode ser observada na Figura 25.

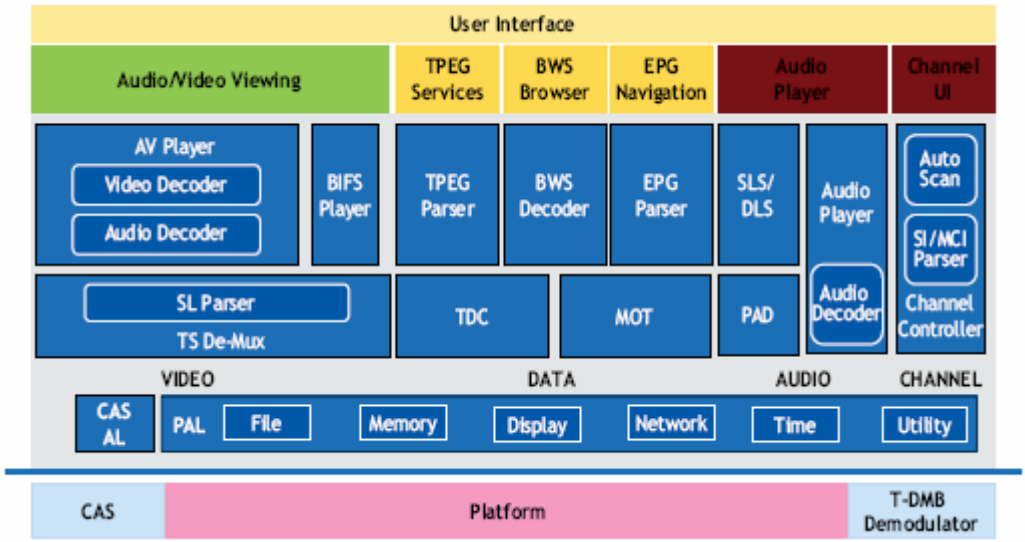

Figura 25: Arquitetura do TMI's T-DMB *middleware* Retirado de (T-DMB Middleware Overview)

Outros produtos que oferecem suporte a T-DMB são:

- InterVideo's iMobi Mobile DTV e FACTUM EPG Library que oferecem apenas ESG.
- S3 onHandTV, que oferece ESG e PVR.
- NeoMagic Mobile TV Solution, que oferece apenas uma ferramenta de autoria, o MiniMagic 6+ Development Board.
- AltiCaptor-Mobile, que oferece HTML, Java(CLDC), ESG e o Alticaptor Mobile SDK como ferramenta de autoria.

6.9 OMA

Open Mobile Alliance foi fundada em 2002, e tem como principal objetivo oferecer soluções para serviços portáteis. Para isso, desenvolve padrões que atendam aos requisitos desse mercado. A OMA também trabalha com outros órgãos de padronização e tem como objetivo ser um ponto de referência nessa área.

O Browsing and Content Working Group (BAC) é responsável por preparar especificações que permitam a criação e o uso de serviços de dados pelos dispositivos portáteis. O OMA-BCAST, por sua vez, é um dos seus subprojetos, e disponibiliza especificações para a prestação de serviços necessários para a TV Digital.

A necessidade de uma padronização nessa área foi observada pela OMA em 2003. No final de 2004, o primeiro *draft* do BCAST foi disponibilizado. O último documento foi lançado em 2007, entretanto, nenhum release do BCAST foi feito pela OMA ainda.

A sua arquitetura possui um conjunto de entidades lógicas relacionadas entre si, como pode ser observado na Figura 26.

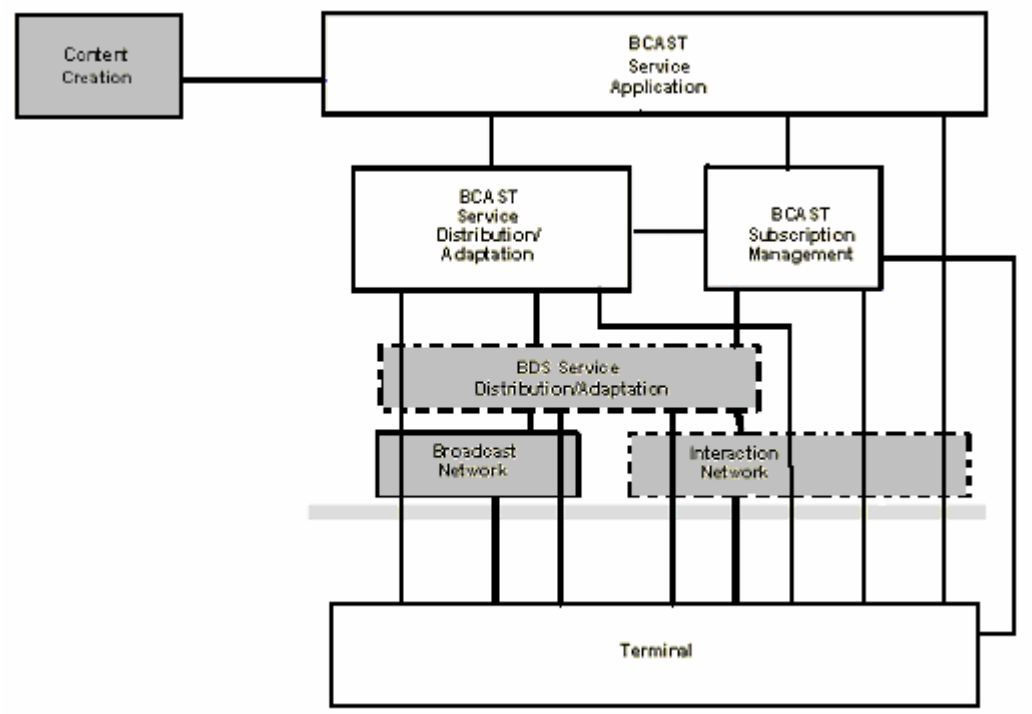

Figura 26: Relacionamento das Entidades lógicas do OMA-BCAST. Adaptado de (Mobile Broadcast Services Architecture)

As entidades preenchidas em branco são aquelas definidas pelo BCAST. As que estão preenchidas em cinza são aquelas que não são especificadas pelo BCAST. As que possuem o contorno pontilhado são entidades opcionais, enquanto que as outras são obrigatórias. As linhas representam os relacionamentos entre esses objetos.

As entidades descritas a seguir são aquelas que fazem parte do escopo da especificação em questão:

- *BCAST Service Application* dispõe o serviço oferecido, propriamente dito, como *streamings* de áudio ou vídeo e *downloads* de filmes. Nele também é definido qualquer tipo de interação relacionada ao serviço.

- *BCAST Service Distribution Adaptation* realiza a distribuição do serviço oferecido pela entidade descrita anteriormente. Faz isso realizando adaptações como, por exemplo, de segurança, adicionando criptografia ou alguma outra técnica.
- *BCAST Subscription Management* oferece mecanismos que permitem a assinatura de serviços e o pagamento dos mesmos, quando for o caso.
- Terminal é o dispositivo que deve receber e ser capaz de apresentar um serviço BCAST. O terminal também pode oferecer, fugindo do escopo do BCAST, suporte a canal de interatividade.

As entidades a seguir estão fora do escopo desta especificação:

- *Content Creation* oferece o conteúdo dos serviços. Uma vez que esta é uma entidade que se encontra fora do escopo desta especificação, o BCAST, portanto, não oferece forma de descrever conteúdos de apresentações multimídia.
- *BDS Service Distribution/Adaptation*. Esse serviço pode ser oferecido opcionalmente. (Mobile Broadcast Services Architecture)
- *Broadcast Network* dispõe a transmissão de conteúdos via um canal de broadcast. (adaptado de Mobile Broadcast Services Architecture).
- Interaction Network suporta o canal de interatividade. Esse serviço pode ser oferecido opcionalmente**.**

Existem nove áreas funcionais distribuídas entre as entidades lógicas da arquitetura do BCAST: *Service Guide*, *File Distribution*, *Stream Distribution*, *Service Protection*, *Content Protection*, *Service Interaction*, *Service Provisioning*, *Terminal Notification* e *Terminal Provisioning*.

Service Guide oferece informações sobre o conteúdo disponível para broadcast. *File Distribution* é responsável por distribuir um arquivo, ou um conjunto deles, para os terminais. Stream Distribution distribui *streams* de áudio ou vídeo. Service Protection protege o acesso ao serviço, permitindo ou não o seu uso, enquanto que o Content Protection protege o conteúdo obtido através do serviço. Service Interaction oferece comunicação ponto a ponto do terminal com o provedor de serviço. Service Provisioning é responsável pela assinatura de serviços pagos. Terminal Notification informa ao terminal sobre um evento lançado por um serviço. Por fim, o Terminal Provisioning gerencia os parâmetros de codificação do terminal.

Como a parte de criação de conteúdo não é definida pelo padrão em questão, nenhum *middleware* próprio foi especificado. BCAST nem mesmo recomenda um outro padrão de *middleware* a ser usado.

É importante lembrar que o OMA-BCAST ainda está em fase de *drafting*, portanto algumas coisas ainda podem mudar ou serem acrescentadas. Apesar disso, a Nokia possui dispositivos que se baseiam no OMA BCAST, como pode ser visto em (Mobile TV Forum).

6.10 Open IPTV Forum

Fundado em 13 de Março de 2007 pelas empresas AT&T Inc., Ericsson, France Telecom, Panasonic, Philips, Samsung, Siemens Networks, Sony, e Telecom Itália, o

fórum tem como objetivo disponibilizar padrões abertos para IPTV. A princípio, apenas as empresas fundadoras participarão dele, mas, no futuro, qualquer uma poderá.

Diferentemente de outros órgãos de padronização, o Open IPTV Forum visa criar padrões em todas as áreas de uma solução IPTV. O objetivo é preencher os espaços em branco dos esforços já produzidos. Sendo assim, o fórum pretende trabalhar em conjunto com os outros órgãos de padronização existentes.

O fórum tem como objetivo tornar esses padrões disponíveis em uma grande variedade de dispositivos, inclusive nos portáteis. Pelo cronograma, as primeiras *releases* de especificações deverão estar prontas em meados de 2008. (Open IPTV Forum Whitepaper, 2007). Não foi feita nenhuma menção sobre a definição de algum *middleware*.

6.11 ATIS IIF

ATIS (*Alliance for Telecommunications Industry Solution*) é uma organização que tem como objetivo oferecer padrões para áreas relacionadas a comunicações e visa especificamente o mercado americano. IIF é a sigla para o IPTV Interoperability Forum da ATIS, criado em 2005 para realizar padronizações na área de IPTV.

As empresas que participam do Fórum são ADTRAN, Alcatel-Lucent, Amdocs, AT&T, British Telecom, Cisco Systems, Digital Fountain, Hitachi Telecom, Huawei Technologies, IneoQuest Technologies, Irdeto Inc., Leapstone Systems Inc., LG Electronics, Motorola, Myrio/Siemens e muitas outras. Nesse grupo existem várias empresas que desenvolvem soluções de IPTV, como a Myrio e a Huawei.

Os esforços de padronização abrangem as seguintes áreas: Arquitetura, DRM, Interoperabilidade e Teste, Metadata e Métricas de QoS. Informações mais detalhadas sobre essas áreas, acesso às especificações ou qualquer outra informação mais específica só estão disponíveis para membros.

6.12 Middlewares Proprietários

Essa seção é muito significativa para este estudo porque os processos de padronização em IPTV ainda se encontram na sua juventude.

Informações detalhadas sobre os produtos são de difícil acesso, uma vez que as empresas negam-se a provê-las. Em alguns casos, algumas informações não são disponibilizadas nem mesmo para clientes.

Algumas das soluções oferecem suporte para alguns padrões de TV Digital, como DVB, e até mesmo aos seus *middlewares*. Mas em nenhum desses casos o produto era adequado para os dispositivos portáteis. De forma geral, nenhuma das soluções de IPTV serve para ser usada nesse tipo de dispositivo, entretanto, elas serão apresentadas a fim de se ter uma noção do cenário vigente.

6.12.1 Adtec Middleware Application Server

Desenvolvido pela Adtec, esse produto precisa de 512MB de memória RAM e 1.5 GHz de processador. Suporta os sistemas operacionais Windows XP, OSX Jaguar e Linux com kernel 2.6 ou superior, e é direcionado para ambientes com dispositivos fixos.

A solução da Adtec oferece suporte a ESG e a VoD. Os clientes podem escolher dentre os *browsers* IE 6.0, Firefox, Safari ou Opera para a exibição das suas apresentações. Sendo assim, as aplicações podem ser construídas com uso de HTML e JavaScript. Nenhuma ferramenta de autoria ou qualquer API é disponibilizada.

6.12.2 Alticaptor

Desenvolvido pela Alticast, o Alticaptor suporta IPTV e os padrões de TV Digital DVB-MHP 1.0.3 e OCAP 1.0. Requer 8Mb de memória para o seu *middleware*, 16Mb para o MHP e 32Mb para o OCAP. Suporta os seguintes sistemas operacionais: Linux, Nucleos+, OS20, pSOS, PowerTV e WinCE. Funciona em dispositivos fixos.

Essa solução oferece suporte a ESG, VoD, PVR e Pay-per-View. Possui *browsers* proprietários integrados, o NetFront e o Infraware Embider, e agrega também um *player* Flash. Sendo assim, suas aplicações podem ser desenvolvidas em HTML/Javascript, Flash ou Java (MHP ou OCAP). Sua arquitetura pode ser vista em detalhes na Figura 27.

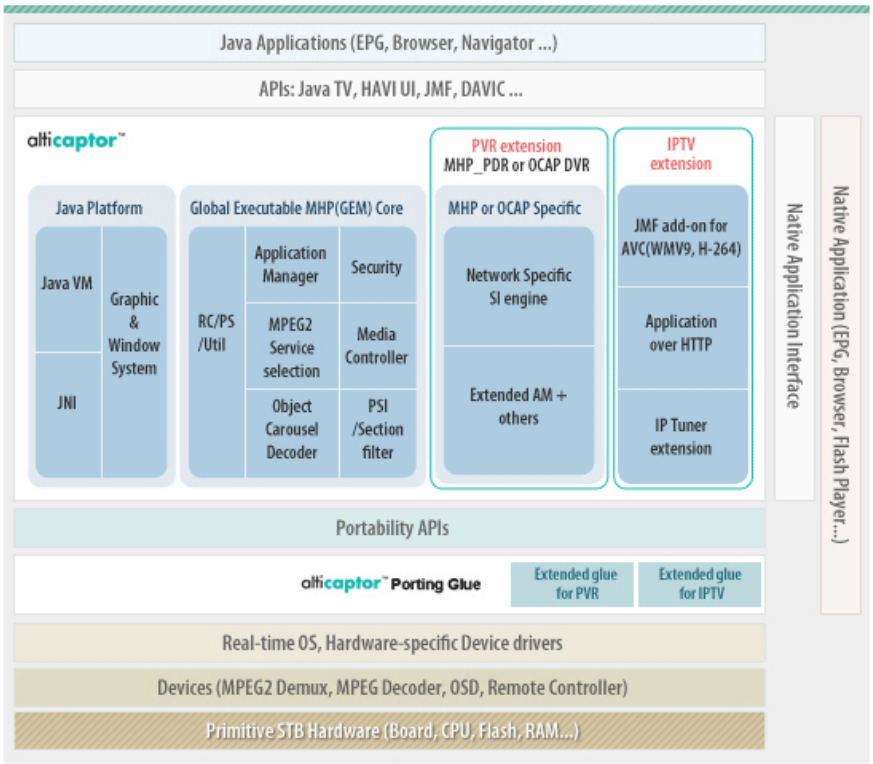

Figura 27: Arquitetura do Alticaptor. Retirado de (http://www.alticast.com/solutions/middleware.html).

Nenhuma API ou ferramenta de autoria é disponibilizada para esse produto.

6.12.3 ANT Galio Client

A sua desenvolvedora é a Ant, que se encontra no mercado há quatorze anos. São poucas as informações liberadas do produto.

O Ant Galio Client suporta IPTV e as suas aplicações podem ser desenvolvidas em HTML, CSS e JavaScript. A solução possui um *browser* próprio, o Ant Fresco Browser,

que diz ser leve e multiplataforma. Existe pouca informação acerca de serviços como ESG e VoD, mas aparentemente eles não são suportados.

A Ant afirma, também, que o Ant Pureplay, um outro produto, permite que o conteúdo digital recebido pelo Ant Galio seja visualizado em dispositivos portáteis. Pode-se, então, com esse software, criar uma rede *wireless* dentro de casa e, assim, transferir conteúdo digital de um dispositivo para outro. Dessa forma, parece ser possível uma transmissão limitada, dentro da própria casa, de TV para dispositivos portáteis. Entretanto, dentre os tipos de dispositivos portáteis suportados, não foram citados celulares ou PDAs, e sim aparelhos maiores, como DVD *players*. Não fica claro, com as informações disponíveis, se o Ant Pureplay é integrado ou não ao Ant Gálio Client.

6.12.4 Bstream Middleware

Bstream foi desenvolvido pela Broadstream Communications e oferece versões cliente e servidor do seu produto a fim de prover serviços de IPTV. A versão servidora oferece recursos para manutenção dos serviços oferecidos, enquanto que a versão cliente os apresenta. Funciona no Windows CE e foi desenvolvido para dispositivos fixos.

O Bstream Middleware suporta ESG, Pay-per-View e VoD, além de permitir acesso à Internet . Oferece suporte ao Internet Explorer 6.0 e a ao Flash Player 6.0, que podem ser usados para apresentação de conteúdos. A Broadstream Communications da pouca informação acerca do seu produto.

6.12.5 Dreamgallery

Foi desenvolvido pela Dreampack e foi resultado de uma pesquisa com usuários finais criada pelo governo Sueco. Sendo assim, Dreamgallery tem como foco o usuário final, afirmando ter, portanto, uma interface amigável e eficiente.

Nenhum tipo de requisito do produto é informado. Afirma-se, contudo, que o seu funcionamento é flexível e que pode ser portado para qualquer STB. Está disponível para dispositivos fixos apenas.

São oferecidos EPG, VoD, e-mail, navegação pela Internet e jogos. As aplicações são customizáveis. Existe uma ferramenta, a Dreamgallery Portal Generator, que permite a criação de portais de TV sem a necessidade de conhecimentos em linguagens de programação. Nenhuma API é disponibilizada, as aplicações podem ser criadas apenas com a ferramenta citada.

6.12.6 Envivio 4Front IPTV Middleware

Sua desenvolvedora é a Envivio, especializada em soluções de IP baseadas no sistema MPEG4 vídeo. Nenhum requisito é informado, mas os sistemas operacionais suportados são o Windows e Linux, e pode operar sobre qualquer player MPEG4-BIFS. O produto é adequado para dispositivos fixos.

Os serviços suportados são EPG, VoD e PVR. As aplicações são construídas usando-se o MPEG4-BIFS, que já foi descrito neste documento, com a ajuda da Envivio 4Mation MPEG-4 Authoring Environment. Essa ferramenta de autoria possui *templates* que ajudam na criação do serviço ou aplicação, e permite também que novos *templates* sejam criados.

6.12.7 Evo Client

A Espial é uma empresa especializada em desenvolver aplicações para IPTV. O seu *middleware* é o Evo Client, que, a fim de facilitar o seu porte e a criação de conteúdos, oferece uma arquitetura aberta. Afirma-se ser portável inclusive para dispositivos portáteis, mas não detalha essa característica. Não são dadas informações acerca dos requisitos exigidos.

Os serviços oferecidos são ESG, VoD, PVR, Pay-per-View e acesso a Internet. Oferece o Evo Future-Proof Framework, que possui APIs e ambientes de desenvolvimento que podem ser usados na criação de novos serviços e aplicações.

O Evo Portal é uma plataforma que permite a criação de portais com o uso de HTML e Javascript.

O Evo Browser permite a execução de aplicações que utilizam padrões de Internet do W3C, como o HTML 4, CSS2 e Javascript 1.5. Além disso, outros padrões podem ser usados, como o ARIB BML, MHP e OCAP. Flash também é oferecido através de plugins. O Evo Browser possui uma ferramenta de autoria, o Evo Browser Content Development Kit, que é capaz de criar aplicações em qualquer um dos padrões citados.

O Evo Client, por ser aberto, pode ser integrado com outras soluções, mas a Espial oferece o Evo Server, que pode ser vendido pré-integrado ao Evo Client.

6.12.8 fs/cdn

Desenvolvido pela Conklin-Intracon, o fc/cdn permite a transmissão de TV sobre o protocolo IP para dispositivos fixos. Possui um sistema DRM embutido e afirma ser independente de hardware e de infra-estrutura.

Os serviços oferecidos são ESG, VoD, Pay-per-View, e-mail e acesso a Internet. Não existe nenhuma API ou ferramenta de autoria que permita a criação de aplicações. Ao invés disso, a Conklin-Intracon oferece serviços de criação de ESG e aplicações para os seus clientes.

6.12.9 IPANEL Browser IPTV

Sua produtora é a EIS (*Embedded Internet Solution*), que oferece produtos e serviços de TV digital em broadcast e por IPTV. Os requisitos exigem no máximo 300 Kb de armazenamento, 1Mb de memória Flash e 2,5 Mb de memória RAM e é independente de plataforma.

Os serviços oferecidos são ESG, acesso a Internet e e-mail. Sua plataforma aceita várias ferramentas típicas da Internet, como Flash, DHTML, XML e J2ME. Entretanto, não é claro se essas ferramentas podem ser usadas na criação de aplicações de TV Digital.

6.12.10 iViewTV

iViewTV foi desenvolvido pela TutSystems, recentemente comprada pela Motorola, que vende apenas a sua solução completa de IPTV. Nenhuma informação acerca dos requisitos ou da portabilidade desse *middleware* pode ser encontrada, mas a empresa porta a sua solução de IPTV, que funciona em dispositivos fixos.

Os seus serviços oferecidos são EPG, VoD, PVR, email e a possibilidade de jogos on-line. A solução oferece também formas de se desenvolver aplicações com o uso de HTML e Javascript. Existem tags do iViewTV que estendem as funcionalidades padrões do HTML. Não é informado se existe alguma ferramenta de autoria ou se o HTML e o Javascript seguem alguma padronização, como a do W3C.

6.12.11 Kasenna PortalTV

Desenvolvido pela Kasenna, o Kasenna Portal TV é uma solução aberta, que oferece APIs e soluções padronizadas para o desenvolvimento e, por conta disso, afirma ser facilmente portável para qualquer STB disponível. São desconhecidos os requisitos mínimos exigidos pela ferramenta, que funciona em dispositivos fixos.

Os serviços oferecidos por esse produto são ESG, VoD, PVR e Pay-per-View. Possui uma ferramenta de desenvolvimento, a Kasenna Living Roon SDK, que possibilita a criação de novos serviços e aplicações. Utiliza, para esse objetivo, padrões de HTML e Javascript, bem como Java. A Figura 28 ilustra a sua arquitetura.

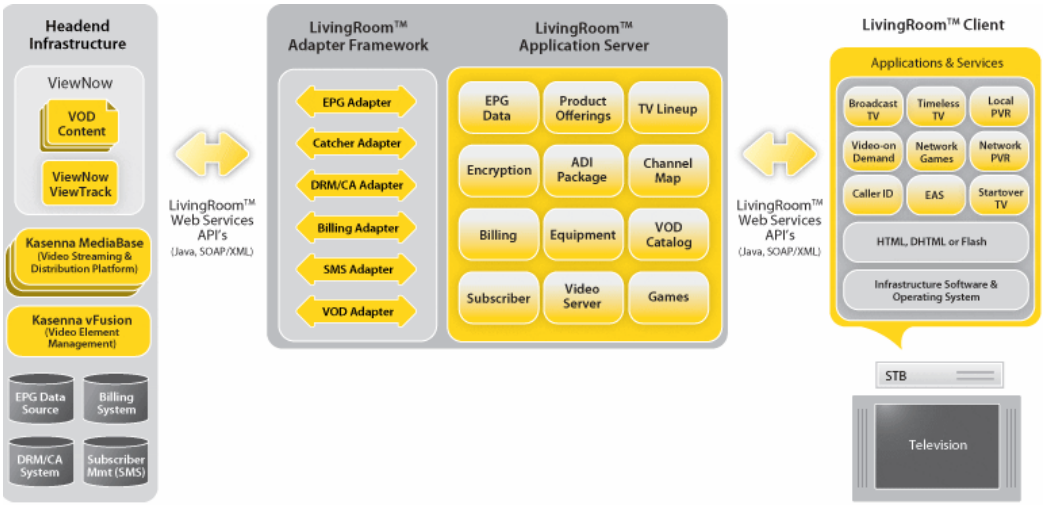

Figura 28: Arquitetura do Kasenna PortalTV. Retirado de (http://www.kasenna.com/solutions/telco/index.php)

6.12.12 Lucent's MiView TV

Alcatel.Lucent é o resultado da fusão das empresas Alcatel e Lucent. A primeira possuía um acordo com a Microsoft para o desenvolvimento da sua solução de IPTV, que será apresentada em capítulos subseqüentes. Pelo acordo, a Alcatel interromperia o desenvolvimento do seu *middleware*, deixando o seu produto integrado com a solução Microsoft. Entretanto, a Alcatel foi comprada pela Lucent e esse novo *middleware* foi desenvolvido.

O Lucent's MiView TV funciona nos sistemas operacionais Windows 2003 e Linux Redhat AS 4.0. Os seus requisitos mínimos não são informados, pois variam de acordo com o tipo de serviço oferecido. O produto destina-se a dispositivos fixos.

Os seus recursos oferecidos são ESG, Pay-per-View, VoD e PVR. As aplicações são baseadas em padrões da Internet para HTML e Javascript, mas pouca informação é dada acerca disso. Nenhuma ferramenta de autoria ou qualquer SDK foi encontrado.

6.12.13 MediaHighway

Desenvolvido pela NDS, empresa especializada em TV por assinatura, o MediaHighway possui duas versões, a *Core* e a *Advanced*. Ambas são desenvolvidas para dispositivos fixos e não são informados os seus requisitos mínimos. Afirma-se, em ambas as versões, que o seu porte é fácil, mas maiores detalhes sobre isso não puderam ser encontrados. Além de IPTV, esse *middleware* aceita Broadcast baseado no DVB.

Os serviços oferecidos pelas duas são: ESG, VoD e PVR. Também em ambas existe suporte a aplicações HTML 4.0 com Javascript 1.3. A versão *Advanced*, entretanto, suporta o DVB/MHP e o OCAP, bem como a uma API Java especialmente desenvolvida para o *middleware*.

A arquitetura de ambas as versões, *Core* e *Advanced* respectivamente, podem ser vistas na Figura 29 e na Figura 30.

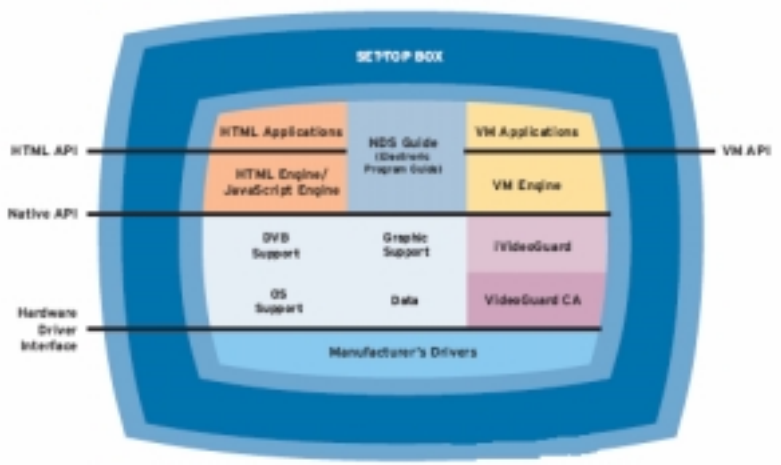

Figura 29: Arquitetura do MediaHighway Core. Retirado de (http://www.nds.com/middleware/mediahighway_core.html)

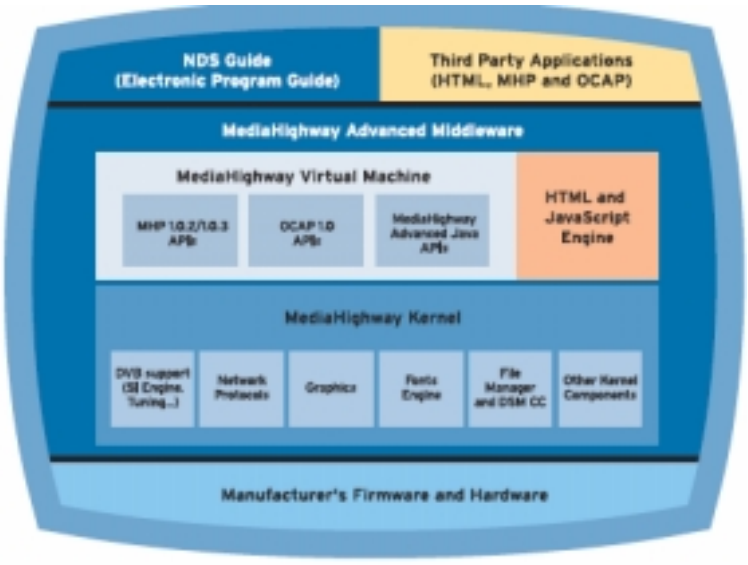

Figura 30: Arquitetura do MediaHighway Advanced. Retirado de (http://www.nds.com/middleware/mediahighway_advanced.html)

A NDS oferece, também, uma ferramenta de desenvolvimento para os seus *middlewares*, a MediaHighway Development Kit, que inclui versões para o *Core* e para o *Advanced*.

6.12.14 Microsoft/Alcatel TV

A Microsoft e a Alcatel uniram-se a fim de oferecer uma solução de IPTV. Com a recente compra da Alcatel pela Lucent, ambas oferecem esse tipo de solução, mas a Microsoft vende a sua completa, ou seja, as partes, como o *middleware*, não são vendidas separadamente. Não se sabe como a parceria entre as duas vai ficar.

Pouca informação acerca da solução da Microsoft pode ser encontrada, mas ela é desenvolvida para dispositivos fixos que operam com o Windows CE. Nenhuma informação acerca dos requisitos é dada. Os serviços oferecidos são ESG, VoD, PVR e Pay-per-View utilizando o Windows Media 9 Series para transmissão e apresentação de mídias. Aparentemente, é possível realizar o desenvolvimento de aplicações com o uso do .NET.

6.12.15 Minerva iTVManager

Desenvolvido pela Minerva Networks, esse *middleware* possui uma arquitetura aberta e disponibiliza APIs de desenvolvimento. Não são informados os seus requisitos mínimos nem mesmo em que tipo de plataforma ou sistema operacional ele funciona.

Os serviços oferecidos são ESG, VoD, PVR, Pay-per-View e acesso a Internet. Como foi dito, a plataforma é aberta e as APIs para o desenvolvimento das aplicações são disponibilizadas. A possibilidade do uso de HTML e Javascript para esse mesmo objetivo não é clara, mas parece que é possível. A Minerva disponibiliza o *Multimedia Service Delivery Platform* (MSDP) com o seu *middleware* que permite a fácil criação de aplicações com o uso das APIs e de ferramentas que vem integradas ao MSDP.

6.12.16 Myrio Interactive

Myrio, comprada em 2005 pela Siemens, desenvolveu o Myrio Interactive, que é uma parte do SUPASS Home Entertainment, uma solução de IPTV feita pela Myrio/Siemens em conjunto com a Nokia.

Esse *middleware* de IPTV foi feito para dispositivos fixos e foi desenvolvido em Java, o que possibilita ao produto executar em uma variedade grande de dispositivos sem grandes esforços de porte. Como os outros produtos já analisados, o Myrio Interactive não informa os seus requisitos mínimos requeridos.

Os seus serviços oferecidos são VoD, PVR , Pay-per-View e *Browser* HTML Walled Garden Internet. Possui uma interface própria para a interação com o usuário que pode ser customizada com o uso da Myrio Interactive Creator.

O mais interessante é que esse *middleware* parece permitir a transmissão de IPTV tanto para dispositivos fixos como para portáteis. Entretanto, maiores informações sobre isso não estão disponíveis.

6.12.17 OpenTV Middleware

OpenTV desenvolveu um *middleware* que suporta IPTV e Broadcast baseado em DVB. É aberto e é independente de plataforma ou de sistema operacional, exigindo apenas 2Mb de memória Ram. O produto foi desenvolvido para dispositivos fixos apenas.

Seus serviços oferecidos são ESG, VoD, PVR e Pay-per-View. O que mais chama a atenção no produto são as diferentes linguagens que podem ser usadas no desenvolvimento das aplicações. É possível usar HTML 4.0 com Javascript 1.3, Flash, Java/MHP e ainda é disponibilizada um tipo de máquina virtual C que suporta o OpenTV Software Developer's Kit (SDK), desenvolvido especialmente pela OpenTV. O suporte a essas linguagem é obtido através do uso de pacotes opcionais do OpenTV, ou seja, para poder criar aplicações Java/MHP é preciso ter integrado o pacote MHP. Assim, é possível customizar o *middleware* para atender as necessidades requeridas. Os pacotes existentes são: HTML, Flash e MHP. Um quarto pacote é o de PVR, que não é um serviço padrão da ferramenta, ou seja, precisa ser adicionado, se requerido. A Figura 31 ilustra a arquitetura do produto.

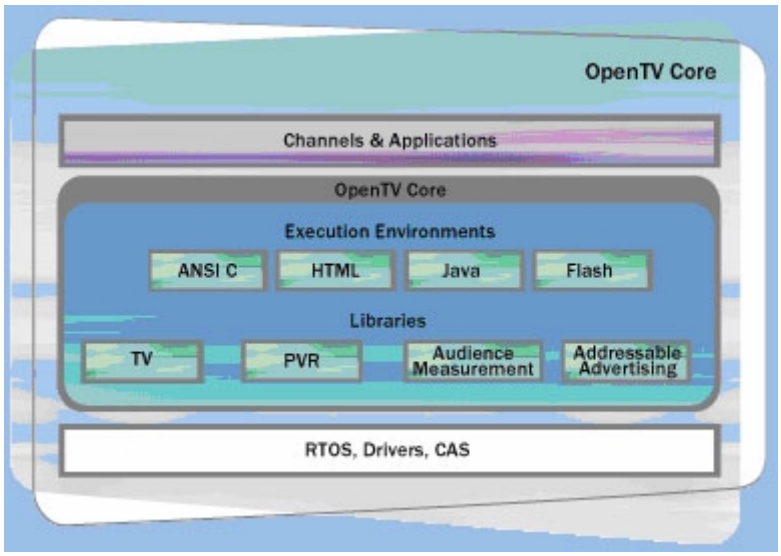

Figura 31: Arquitetura do OpenTV Middleware Retirado de (http://www.opentv.com/products/middleware/mw_core.html)

6.12.18 Orca's RiGHTv

Orca é a empresa que desenvolveu o Orca's RightTv, um *middleware* feito para transmissão IPTV em dispositivos fixos. Seu produto é dividido em quatro pacotes: XVOD, XBIP, XPVR e o Core.

O Core é a plataforma de execução dos serviços digitais, oferecendo meios para a criação das aplicações. Todos os outros pacotes e aplicações externas são executados sobre essa plataforma. O XVOD permite a criação e o gerenciamento de serviços de VoD. O XBIP possibilita o uso de *multicasting* para a transmissão dos canais de TV. O XPVR permite que serviços de PVR sejam oferecidos.

Os requisitos mínimos da plataforma não são informados, bem como nenhuma questão acerca da sua portabilidade. As aplicações podem ser criadas com o uso de HTML e Javascript, através da ferramenta RiGHTv SUI SDK, que vem junto com o *middleware*.

6.12.19 ORTIKON ACE® IPTV Middleware

Desenvolvido pela Oikon, ele é um middleware de IPTV para dispositivos fixos. Assim como a maioria dos softwares proprietários, nenhuma informação sobre s requisitos mínimos é dada. A empresa afirma que o seu produto é independente de hardware e de plataforma.

Seus serviços oferecidos são ESG, VoD, PVR, Pay-per-View, e-mail e servidor de jogos. As aplicações podem ser feitas em flash e, com a ajuda da ACE SDK, em HTML 4.01, XHTML 1.0, Javascript 1.3 e Javascript extensions for enhanced TV.

6.12.20 Osmosys MHP 1.1 ou OCAP 1.0

O *middleware* da Osmosys possui arquitetura aberta e versões para MHP ou OCAP. Foi desenvolvido para transmitir IPTV para dispositivos fixos e não possui qualquer informação sobre requisitos mínimos ou sobre a portabilidade da sua plataforma. O produto foi desenvolvido em C e em Java.

Oferece apenas serviço de PVR. Pay-per-View é suportado apenas para a versão MHP do produto. Na versão OCAP, as aplicações podem ser feitas em HTML, Java script ou OCAP. Na versão MHP, além do HTML e do Javascript, é possível usar a DVB-HTML e o MHP, ou DVB-J. Para auxiliar no desenvolvimento, a Osmosys oferece o Osmosys SDK, que possibilita a criação de aplicações em qualquer uma das linguagens descritas. As arquiteturas da versão MHP e OCAP, respectivamente, podem ser vistas na Figura 32 e na Figura 33.

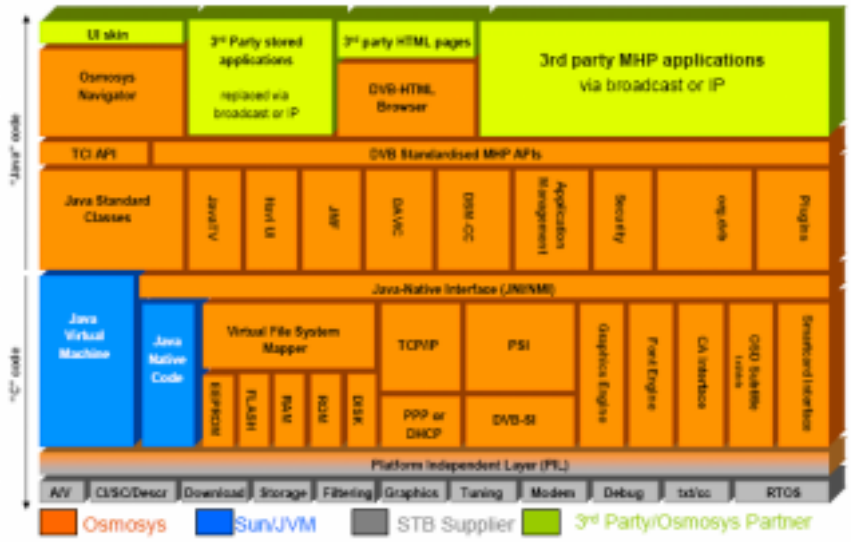

Figura 32: Arquitetura do Osmosys MHP. Retirado de (http://www.osmosys.tv/products/products.htm)

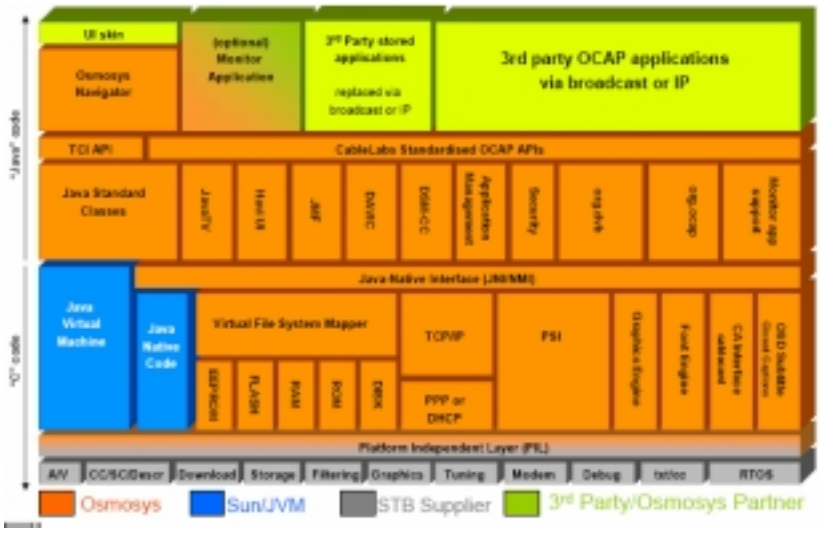

Figura 33: Arquitetura do Osmosys OCAP. Retirado de (http://www.osmosys.tv/products/products.htm)

6.12.21 RollingStream

Desenvolvido pela UTStarcom, especializada em criar infra-estruturas IP para transmissão de conteúdo digital, o RollingStream é uma solução completa de IPTV que não é vendida em partes. Quase nenhuma informação é dada acerca do *middleware* utilizado. Não existe nenhuma forma de criar aplicações e os serviços oferecidos são ESG, VoD e PVR apenas. A transmissão é feita apenas para aparelhos fixos, ou seja, não existe suporte para dispositivos portáteis.

6.12.22 SeaChange TV Navigator

TV Navigator, desenvolvido pela SeaChange, é um *middleware* destinado à recepção de IPTV via cabo, satélite ou radiodifusão em território fora dos EUA. Funciona em dispositivos fixos, não possui requisitos mínimos e permite que aplicações de terceiros sejam usadas.

Os serviços oferecidos são ESG, VoD, PVR, servidor de jogos, acesso a Internet e e-mail. O produto oferece formas de se criar aplicações com duas ferramentas de desenvolvimento, o SeaChange Multiverse e o SeaChange Universal. O primeiro permite que novos serviços de vídeos sejam criados e o segundo possibilita que aplicações variadas sejam desenvolvidas. Afirma-se que padrões da Internet podem ser utilizados, mas foram encontradas apenas citações para HTML e Javascript sem qualquer menção das suas versões.

6.12.23 SmartVision TV

Pouca informação acerca deste *middleware* é dada pela Grass Valley, sua produtora. O SmartVision TV funciona em dispositivos fixos e oferece serviços de VoD e PVR. Nenhuma forma de se criar aplicações foi encontrada. Nenhuma outra informação relevante a este estudo foi encontrada.

6.12.24 ViewCore

Assim como no item anterior, pouca informação do ViewCore é oferecida pela sua desenvolvedora, a Infogate. O produto foi feito para dispositivos fixos e oferece serviços de ESG, VoD, PVR e Pay-per-View. Não é possível desenvolver aplicações para esse *middleware* e nenhuma outra informação relevante a este estudo foi encontrada.

6.13 Análise Comparativa

Embora IPTV seja considerada uma ótima opção para o futuro, padrões abertos para essa tecnologia ainda não são encontrados. Existem apenas aqueles que estão sendo desenvolvidos por consórcios e que só estão disponíveis para membros, como é o caso da ATIS ISS. A DLNA cria padrões de IPTV e, segundo ela, abrange também os dispositivos portáteis, mas informações para não membros do consórcio não são liberadas; no que diz respeito aos *middlewares* em específico, nenhum padrão foi definido. O Open TV Forum pode vir a desenvolver algum middleware, mas os seus primeiros esforços não são para a criação de um *middleware*. Não se sabe, também, se alguma tentativa futura nesse sentido vai abranger o contexto dos dispositivos portáteis. Hoje, o que mais se vê são soluções proprietárias, altamente especializadas e, freqüentemente, completamente fechadas. Mesmo nessas soluções, são poucas as menções do seu uso em dispositivos portáteis e, quando ocorrem, são muito superficiais. O que se observa é que a IPTV, principalmente no que diz respeito a padronização, ainda se encontra na sua infância. Parece ainda mais jovem se olharmos da perspectiva da sua aplicação em dispositivos portáteis, principalmente sabendo que a TV Digital já é transmitida, hoje, para celulares e em outros aparelhos do tipo.

Por outro lado, a TV Digital por radiodifusão terrestre, por cabo ou satélite possui padrões abertos e implementados para dispositivos portáteis.

O padrão de TV Digital OMA-BCAST, desenvolvido pela OMA, encontra-se em fase de *drafingt* e, apesar de ser direcionado para prover serviços em dispositivos portáteis, não possui nenhuma especificação de *middleware*.

O MBMS, desenvolvido pela 3GPP, especifica o DIMS como o seu *middleware*. Entretanto, tanto as especificações do MBMS, quanto as do DIMS, ainda são *drafts* e possuem pouco conteúdo. Também não existe, atualmente, nenhuma implementação prática desse padrão e o lançamento dos seus primeiros terminais só está previsto para 2008.

O padrão definido pela ATSC não é adequado para o âmbito dos dispositivos portáteis e o seu *middleware* não possui nenhuma implementação nesse ambiente. O T-DMB, por sua vez, possibilita também a transmissão em dispositivos portáteis. Entretanto, usa o MPEG4-BIFS, que não é apropriado para a composição de conteúdo digital em alto nível de abstração. Esforços de padronização têm sido feitos para criar uma outra linguagem mais adequada.

O LASeR surgiu em vista da dificuldade do uso do BIFS, e o seu *middleware* deriva do SGV Tiny. É feito exclusivamente para o âmbito dos dispositivos portáteis, e, apesar de ser um *draft* e de não ter sido aceito como um padrão ainda, parece estar mais maduro do que os outros já citados. Todavia, nenhuma implementação sua foi encontrada.

O padrão DVB definiu o DVB-H para transmissão digital em dispositivos portáteis. Para dispositivos fixos, o *middleware* MHP foi definido pelo DVB. Entretanto, nenhum

middleware, nem tão pouco um *subset* do MHP, foi definido pelo DVB a fim de atender ao mercado dos dispositivos portáteis. Implementações do DVB-H podem ser encontradas, mas não com o uso do MHP, que utiliza especificações J2SE na sua linguagem procedural, a DVB-J, que não são apropriadas para os dispositivos portáteis. Como já foi visto neste estudo, as aplicações nesse contexto precisam ser mais enxutas e, por isso, utilizam uma versão Java diferenciada, a JavaME. Além disso, a linguagem declarativa especificada pelo padrão, a DVB-HTML, é baseada em HTML, que não é muito eficiente, e executa, na maioria das vezes, em uma camada acima do DVB-J, comprometendo ainda mais sua performance. O *middleware* MHP, portanto, não parece ser adequado para ser executado em aparelhos portáteis.

O HisTV é um padrão de *middleware* para TV Digital desenvolvido especialmente para dispositivos portáteis. A sua especificação ainda se encontra em fase de *drafting*, mas existem implementações sendo executas em dispositivos portáteis no mercado. O HisTV oferece uma interface padrão fixa para as aplicações. Para um domínio de aplicações, isso é uma vantagem, pois acelera o processo de desenvolvimento. Por outro lado, aplicações com interfaces diferenciadas não podem ser desenvolvidas. A linguagem usada no HisTV é complexa e pouca informação é dada acerca do uso de documentos CSV para representá-la. Mas o maior problema é que a linguagem requer que o desenvolvedor seja capaz de conceber a sua aplicação como uma máquina de estados. Isso certamente não é intuitivo, e usuários menos especializados não serão capazes de desenvolver aplicações HisTV sem a ajuda de uma boa ferramenta de autoria.

BML, do padrão Japonês, foi desenvolvida para a criação de conteúdo digital tanto para dispositivos fixos quanto portáteis. Para isso, determinou-se uma linguagem padrão e um *subset* dela para aplicações portáteis. Existem implementações dessa tecnologia no Japão. Baseada em XHTML, BML é uma linguagem com foco na apresentação do documento e na interação com o usuário. Como XHTML é uma linguagem conhecida, programar em BML requer uma curva baixa de aprendizagem.

O Ginga-NCL foi desenvolvido de forma modular, possuindo *profiles* tanto para dispositivos fixos quanto para portáteis. Por enquanto, existem apenas implementações protótipos para dispositivos portáteis. A NCL, se comparada a BML, é mais apropriada para o desenvolvimento de aplicações hipermídia e, principalmente, de TV Digital. Isso porque tem foco no sincronismo e adaptabilidade, fatores importantes no desenvolvimento de aplicações de TV Digital.

7 Conclusão

Nesse estudo foi feita uma avaliação do suporte à TV Digital em dispositivos portáteis, principalmente no que diz respeito aos seus *middlewares* declarativos. Com esse objetivo em mente, apresentou-se os requisitos limitantes dos dispositivos portáteis, os sistemas operacionais por eles usados, seus ambientes de desenvolvimento e os *middlewares* de TV Digital existentes. Com isso, pôde-se ter uma idéia do cenário atual dessa área.

No que diz respeito ao sistema operacional e a plataforma de desenvolvimento, seções anteriores concluíram que o Symbian OS e o Linux Mobile, juntamente com as suas linguagens nativas, parecem constituir as melhores opções para se fazer uma implementação de um *middleware* de TV Digital.

Quanto aos *middlewares* para IPTV, não existe ainda nenhum padrão, e as implementações proprietárias, de forma geral, oferecem apenas aplicações básicas, como o ESG. No que diz respeito aos dispositivos portáteis, existe menos definição ainda. Assim, nenhuma solução de IPTV, se adequa, ainda, a apresentação de aplicações em dispositivos portáteis.

No caso da TV Digital terrestre, o padrão americano não possui relevância para este trabalho, pois não é apropriado para transmissão *broadcast* em dispositivos portáteis. Isso tende a permanecer assim no futuro, uma vez que os americanos estudam a adoção do DVB-H. O DIMS ainda tem pouca coisa especificada e o uso do BIFS não se mostra adequado para o contexto em questão. Como alternativa ao uso do BIFS, existe o LASeR, que parece interessante, mas não possui implementação e ainda é um *draft*. O HisTV por outro lado, já foi implementado, mas é muito complicado, apesar de ser adequado para dispositivos portáteis. Um outro fator é que o seu domínio de aplicações é limitado pela sua interface fixa. Isso pode ser uma vantagem em inúmeros cenários, mas este estudo está interessado em aplicações com domínios mais amplos. O MHP é um padrão amplamente utilizado, mas, mesmo assim, nenhuma implementação sua para dispositivos portáteis, em específico, foi encontrada. Também não foi feita nenhuma alteração, nem nenhum *profile* específico do MHP foi criado para esse contexto. Além disso, o MHP parece ser muito pesado para ser inserido em um dispositivo portátil. Portanto, o MHP também não parece ser adequado.

A BML e o Ginga-NCL parecem, atualmente, ser os mais apropriados para servirem de *middleware* em dispositivos portáteis. Ambos oferecem *profiles* diferenciados, visando especificamente esse contexto. A BML já possui implementação comercial. O Ginga-NCL possui apenas implementações protótipo. Comparando as duas linguagens dos *middlewares*, a NCL, que foi feita especificamente para representar documentos hipermídia com foco na sincronização, é superior a BML, que é baseada em XHTML e que tem foco declarativo apenas na interatividade. Pela conclusão deste estudo, Symbian OS e Linux Mobile parem constituir as melhores opções para uma primeira implementação de referência do Ginga-NCL.

Referências Bibliográficas

Rodney Aiglstorfer. **Troubleshooting Java Platform, Micro Edition - Tips from the Pros - TS-5927**. 2006.

Anders Borg. **Java still not write once, run anywhere.** Disponível em: http://www.abiro.com/news/2006/06/java-me-still-not-write-once-run.html. jun. de 2006. Ultimo acesso em abril de 2007.

H.M. Deitel, P.J. Deitel. **Java Como Programar 4ª Edição**. Bookman, 2004.

John Drewry. **Will J2ME Deliver?** Disponível em: http://www.wirelessdevnet.com/articles/j2me_orative/index.html. Último acesso em abril de 2007.

Canalys Reseach. **Canalys research release: 64 million smart phones shipped worldwide in 2006**. Disponível em: http://www.canalys.com/pr/2007/r2007024.htm. 2007. Último acesso em mar. de 2007.

Canalys Reseach. **Canalys research release: EMEA smart mobile device market growth rises to 11.7%.** Disponível em:

http://www.canalys.com/pr/2006/r2006102.htm. 2006. Último acesso em mar. de 2007.

Marcio Ferreira Moreno. **Um Middleware Declarativo para Sistemas de TV Digital Interativa**. Dissertação de mestrado, Abril de 2006.

Forum Nokia. **J2ME & Symbian OS: A Plataform Comparision**. Version 1.0, 2003.

Mike Hall. **Mikehall's Embedded WEblog : Windows Mobile and Windows Embedded CE - what's the difference ?.** Disponível em: http://blogs.msdn.com/mikehall/archive/2007/01/17/windows-mobile-and-windows-embedded-ce-what-s-the-difference.aspx. 2007. Último acesso em mar. de 2007.

Werner Heuser. **Linux on the Road.** Disponível em: http://tuxmobil.org/Mobile-Guide/index.html. 2005. Último acesso em mar. de 2007.

Werner Heuser. **TuxMobil: Linux Distributions and Kernels for Mobile SmartPhones.** Disponível em: http://tuxmobil.org/mobile_phone_linux_distributions.html. 2007. Último acesso em mar. de 2007.

Martin de Jode, Colin Turfus. **Symbian OS System Definition Version: 1.2.4.** Symbian Developer Network, set. 2004.

Lars Kirkhus e Anders R. Sveen. **An examination of mobile devices for spontaneous collaboration.** 2003.

Michael Mace. **Mobile Opportunity.** Disponível em: http://mobileopportunity.blogspot.com/2006/11/symbian-unloads-uiq-and-mobile-apps_09.html. 2006. Último acesso em set. de 2007.

Rafael Ferreira Rodrigues. **Tese de Mestrado; PUC-Rio - Departamento de Informática; "Ambiente Declarativo para Sistemas que Implementam o GEM".** set. de 2007.

Anders Rene Sveen e Lars Kirkhus. **MOWAHS - Mobile Collaboration Framework.** 2004.

Mark Shackman. **Migrating Applications to Symbian OS v9.1 Version 1.1.**

Skrodzki. **Handheld Interactive Sincronized TV (HisTV)**. nov. 2006.

Symbian Limited. **Symbian OS 9.1 Manual.** Junho de 2005.

Symbian Limited. **34.6m Symbian smartphones shipped in H107.** Disponível em: http://www.symbian.com/news/pr/2007/pr20079333.html. 2007a. Último acesso em set. de 2007.

Symbian Limited. **Using Symbian OS GETTING STARTED, 2007 2nd edition.** 2-6 Boundary Row, London SE1 8HP, UK, jan. de 2007b.

Symbian Limited. **Symbian OS: the open mobile operating system.** Disponível em: http://www.symbian.com . Último acesso em set. de 2007.

Joseph W. Weber, Tom Newberry. **IPTV Crash Course**. Paperback, Novembro de 2006.

Martin de Jode, Jonathan Allin, Darren Holland, Alan Newman

e Colin Turfus. **Programing on Java 2 Micro Edition on Symbian OS**. Wiley, 2004.

3GPP Feature. Disponível em: http://www.3gpp.org/specs/WorkItem-info/WI--34032.htm. Último acesso em fevereiro de 2007.

3GPP Specification: 22.146. Disponível em: http://www.3gpp.org/ftp/Specs/html-info/22146.htm

3GPP TS 22.146 V8.1.0 - 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Service; Stage 1 (Release 8). set. 2006.

A/101 – ATSC Standard: Advanced Common Application Platform (ACAP). ago. 2005

A/100-1 - DTV APPLICATION SOFTWARE ENVIRONMENT LEVEL 1 (DASE-1) PART 1: INTRODUCTION, ARCHITECTURE, AND COMMON FACILITIES. mar. 2003

A/100-2 - DTV APPLICATION SOFTWARE ENVIRONMENT LEVEL 1 (DASE-1) PART 2: DECLARATIVE APPLICATIONS AND ENVIRONMENT. mar. 2003.

A/100-3 - DTV APPLICATION SOFTWARE ENVIRONMENT LEVEL 1 (DASE-1). mar. 2003.

Applications for Mobile Information Devices - Helpful Hints for Application Developers and User Interface Designers using the Mobile Information Device Profile - A White Paper. 2000.

ATIS: IIF - IPTV Interoperability Forum. Disponível em: http://www.atis.org/iif/. Último acesso em janeiro de 2008.

Axel Technologies. Disponível em: http://www.axel.fi/products.htm. Último acesso em março de 2007.

Axel Brochure - S A LMO N S T R E AM MO B I L E - T V MI D D L EWA R E

B24 Appendix 5 – Operational Guidelines for Implementing Extended Services for Mobile Receiving System. fev. 2004.

Cascading Style Sheets, Level 2. Disponível em: http://www.w3.org/TR/REC-CSS2/. maio 1998. Último acesso em Maio de 2007.

Connected Device Configuration Overview. Disponível em: http://java.sun.com/products/cdc/overview.html. Último acesso em abril de 2007.

Connected Limited Device Configuration (CLDC); JSR 30, JSR 139 Overview. Disponível em: http://java.sun.com/products/cldc/overview.html. Último acesso em abril de 2007.

DIMS. **3GPP TS 26.142 v1.0.0 – Dynamic and Interactive Multimedia Scenes (Release 7).** nov. 2006.

DMB-PORTAL. Disonível em: http://eng.t-dmb.org/. Último acesso em fevereiro de 2007.

DVB – Digital Video Broadcasting. Disponível em: http://www.dvb.org/index.xml. Ultímo acesso em ago. 2007.

DVB-H: Global Mobile TV. Disponível em: http://www.dvb-h.org/. Último acesso em ago. 2007.

DTV Guide Transmission. Disponível em: http://dtvfacts.com/latest/250/as-digital-tv-reception-controversy-dims-e-vsb-gets-another-look/. maio 2006. Último acesso em fevereiro de 2007.

Draft ETSI TS 102 371 V1.2.1 - Digital Audio Broadcasting (DAB); Transportation and Binary Encoding Specification for Electronic Programme Guide (EPG). set 2005.

Draft ETSI TS 102 818 V1.3.1 - Digital Audio Broadcasting (DAB); XML Specification for Electronic Programme Guide (EPG). set. 2005.

Dynamic and Interactive Multimedia Scenes in Mobile Broadcast Environments. Disponível em: http://www.lnt.e-technik.tu-muenchen.de/mainframe/DBdata/diplomanden_einzeln.php?language=de&id=619. Último acesso em fevereiro de 2007.

ECMA-262 - ECMAScript Language Specification. 3rd edition. Dec. 1999.

ETSI TS 102 427 V1.1.1 - Digital Audio Broadcasting (DAB); Data Broadcasting - MPEG-2 TS streaming. Jul 2005.

ETSI TS 102 428 V1.1.1 - Digital Audio Broadcasting (DAB); DMB video service; User Application Specification. jun. 2005.

ETSI TS 102 471 V1.2.1 - Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Electronic Service Guide (ESG). nov. 2006.

ETSI TS 102 812 v1.2.1 – Digital Video Broadcasting (DVB); Multemidia Home Plataform (MHP) Specification 1.1.1. mar. de 2006

ETSI EN 302 304 V1.1.1 - Digital Video Broadcasting (DVB); Transmission System for Handheld Terminals (DVB-H). Nov. de 2004

HisTV Main. Disponível em: http://www.histv.org/. Último acesso em março de 2007.

Interactivity. Disponível em: http://www.w3.org/TR/2005/WD-SVGMobile12-20050413/interact.html#Introduction. abr. 2005. Último acesso em fevereiro de 2007.

IPTV. Disponível em: http://www.openiptvforum.org/. Último acesso em janeiro de 2008.

ISO/IEC 14882:2002. 2002.

ISO/IEC 9899:1999. 1999.

ISO/IEC 14496-1 - CODING OF MOVING PICTURES AND AUDIO, MPEG4 Systems. nov. 1997.

ISO 14496-20 Lightweight Application Scene. Representation (LASeR) and Simple Aggregation Format (SAF). Jun. 2006.

Forum Nokia - Carbide Development Tools for Symbian OS C++. Disponível em: http://www.forum.nokia.com/main/resources/tools_and_sdks/carbide_cpp/. Último acesso em set. de 2007.

Java ME Device Table. Disponível em: http://developers.sun.com/techtopics/mobility/device/device;jsessionid=01962C3A641B9BDD882D3AC598D10A93. Último acesso em maio de 2007.

Java ME Technology. Disponível em: http://java.sun.com/javame/technology/index.jsp. Último acesso em abril de 2007.

Java ME Technology APIs & Docs. Disponível em: http://java.sun.com/javame/reference/apis.jsp#opkg. Último acesso em abril de 2007.

Java Technology. Disponível em: http://java.sun.com/. Último acesso em Setembro de 2007.

JSR 272. Disponível em: http://jcp.org/en/jsr/detail?id=272. Último acesso em maio de 2007.

LASeR. Disponível em: http://www.mpeg-laser.org/html/techSection_technicalOverview.htm

MBMS White Paper. August 2004

Microsoft Windows Mobile. Disponível em: http://www.microsoft.com/windowsmobile/default.mspx. Último acesso em set. de 2007.

Microsoft Windows XP Embedded: Home page. Disponível em: http://www.microsoft.com/brasil/embedded/default.mspx. Último acesso em mar. de 2007.

Mobile Broadcast Services Architecture – Draft Version. 2007.

Mobile Linux Internet Project. Disponível em: http://www.moblin.org/. Último acesso em set. de 2007.

NeoMagic Digital TV Brochure.

NetFront DTV Brochure. 2007.

Nokia - Mobile TV Forum - Home. Disponível em: http://www.mobiletv.nokia.com/. Último acesso em março de 2007.

OMA Moile Broadcast Services. Disponível em: http://www.tml.tkk.fi/Studies/T-111.590/2005/lectures/paila.pdf. feb. 2005. Último acesso em janeiro de 2008.

Open IPTV Forum Whitepaper. nov. de 2007.

Palm OS - Wikipedia, the free encyclopedia. Disponível em: http://en.wikipedia.org/wiki/Palm_OS. Último acesso em mar. de 2007.

RFC 1305. Disponível em: http://www.ietf.org/rfc/rfc1305.txt. mar. 1992. Último acesso em março de 2007.

RFC 1889. Disponível em: http://www.ietf.org/rfc/rfc1889.txt. jan 1996. Último acesso em março de 2007.

RFC 3926 - FLUTE - File Delivery over Unidirectional Transport. Disponível em: http://www.rfc-archive.org/getrfc.php?rfc=3926. Último acesso em março de 2007. out. 2004.

Research In Motion. Disponível em: http://www.rim.com/. Último acesso em mar. de 2007.

SBTVD. **SBTVD - 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.** 2007

SBTVD**. SBTVD N06 - Volume 4 – GINGA-J: Environment for the execution of procedural applications.** 2006.

Scalable Vector Graphics (SVG) Tiny 1.2 Specification. Disponível em: http://www.w3.org/TR/2005/WD-SVGMobile12-20050413/. apr. 2005. Último acesso em fevereiro de 2007.

Scripting - SVG 1.1. Disponível em: http://www.w3.org/TR/SVG/script.html#ScriptingLanguage. jan. 2003. Último acesso em maio de 2007.

Small Biz Resource. Disponível em: http://www.smallbizresource.com/document.asp?doc_id=115099&page_number=1. Último acesso em mar. de 2007.

Smil 2.1 – Timing and Syncronization. Disponível em: http://www.w3.org/TR/2005/REC-SMIL2-20051213/smil-timing.html. dec. 2005. Último acesso em fevereiro de 2007

Symbian Developer Network. Disponível em: http://developer.symbian.com/home.action. Último acesso em mar. de 2007.

Symbian OS System Definition - Detailed View. Disponível em: http://developer.symbian.com/main/downloads/papers/SymbOS_cat/SymbianOS_cat.html. Último acesso em abr. de 2007.

Symbian OS - Wikipedia, the free encyclopedia. Disponível em: http://en.wikipedia.org/wiki/Symbian_OS. Último acesso em mar. de 2007.

T-111.590 Research Seminar on Digital Media P. Disponível em: http://www.tml.tkk.fi/Opinnot/Tik-111.590/. 2005. Último acesso em fevereiro de 2007.

T-DMB Middleware Solution for Mobile TV - Product Overview. out. 2006.

thinDVB-HTM Overview. nov. 2006

thin multimedia, inc. - Converging Multimedia for the Mobile World. Disponível em: http://www.thinmultimedia.com/thin_dvbh_tdmb.html. Último acesso em março de 2007.

Ubuntu Mobile and Embedded Edition. Disponível em: https://lists.ubuntu.com/archives/ubuntu-devel-announce/2007-May/000289.html. Último acesso em set. de 2007.

What's New in MIDP 2.0, 2002. Disponível em: http://developers.sun.com/techtopics/mobility/midp/articles/midp20/. Último acesso em abril de 2007.