UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA

PROGRAMA DE MESTRADO EM ENGENHARIA ELÉTRICA

EMTV – EXTENSÃO DE MIDDLEWARE PARA TV DIGITAL COM COMPONENTES DE SOFTWARE PARA EDUCAÇÃO

Juliano Rodrigues Costa

Manaus
2008

UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA

PROGRAMA DE MESTRADO EM ENGENHARIA ELÉTRICA

JULIANO RODRIGUES COSTA

EMTV – EXTENSÃO DE MIDDLEWARE PARA TV DIGITAL COM COMPONENTES DE SOFTWARE PARA EDUCAÇÃO

Dissertação apresentada ao Programa de Mestrado em Engenharia Elétrica da Universidade Federal do Amazonas, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr.-Ing. Vicente Ferreira de Lucena Júnior

Manaus
2008

Catalogação Biblioteca Central da Universidade Federal do Amazonas

C8372 Costa, Juliano Rodrigues.

EMTV – extensão de middleware para TV digital com componentes de software para educação / Juliano Rodrigues Costa.-- Manaus: UFAM, 2008.

130 f. : il.; 30 cm.

Orientador(a) : Prof. Dr. Vicente Ferreira de Lucena Junior
Dissertação (Programa de Mestrado em Engenharia Elétrica). Universidade Federal do Amazonas.

1.Software - Desenvolvimento. 2.Televisão digital. I. Título
CDU 621.397(043.3)
CDD 621.3881

JULIANO RODRIGUES COSTA

EMTV – EXTENSÃO DE MIDDLEWARE PARA TV DIGITAL BRASILEIRA BASEADA EM COMPONENTES DE SOFTWARE

Dissertação apresentada ao Programa de Mestrado em Engenharia Elétrica da Universidade Federal do Amazonas, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica.

Aprovado em 30 de Janeiro de 2008.

Banca Examinadora

Prof. Dr. Carlos Mauricio Seródio Figueiredo
Fundação Centro de Análise, Pesquisa e Inovação Tecnológica

Prof. Dr. João Edgar Chaves Filho
Universidade Federal do Amazonas

Prof.Dr. Raimundo Da Silva Barreto
Universidade Federal do Amazonas

Prof. Dr.-Ing. Vicente Ferreira de Lucena Júnior
Universidade Federal do Amazonas

Dedico esse trabalho ao meu pai, à minha saudosa madrinha, à minha amada esposa Andréia e principalmente ao meu filho Luiz Eduardo, pois qualquer esforço meu tem o objetivo de lhe garantir um futuro melhor.

AGRADECIMENTOS

Em primeiro lugar à força espiritual tão necessária para superação das dificuldades do dia a dia.

Ao meu orientador Vicente por todo auxílio e suporte prestado durante o curso.

À minha esposa Andréia pela paciência e apoio desde o momento em que decidi pela realização desse trabalho.

Ao meu pai pelo apoio durante toda a minha vida.

Aos amigos Edgard e Aracélis e, Daniel e Michele por terem facilitado minha vida em momentos importantes na conclusão desse trabalho.

Aos colegas do laboratório de TV digital da UFAM, Elidelson, Douglas, Orlewilson e principalmente ao professor Waldir que me auxiliaram de alguma maneira durante a realização das minhas experiências.

Ao CETELI, à SUFRAMA, à CAPES e aos professores do colegiado do mestrado em engenharia elétrica pelas suas colaborações no oferecimento desse curso.

Aos colegas de trabalho do Genius Instituto de Tecnologia, especialmente ao Ivandro Sanches e ao Reinaldo Bernardi por terem me concedido as recomendações necessárias para entrada nesse curso.

Resumo

Este trabalho apresenta uma plataforma capaz de gerar aplicações interativas para serem executadas em sistemas de televisão digital. Ele parte de uma contextualização acerca das tecnologias existentes localizando o modelo adotado no Brasil e as dificuldades envolvidas no processo de desenvolvimento de aplicações. Tendo a necessidade de focalizar sobre um tipo específico de problema, identifica que, aplicações que coletam as respostas dos usuários mediante perguntas que lhe forem feitas são de extrema importância por servirem simultaneamente a propósitos de entretenimento e educação. Esse tipo de aplicação é comumente chamado de *Quiz*. Tendo esse propósito em vista, parte-se para enumeração das características consideradas essenciais para esse tipo de aplicação. Essas características são levadas em consideração para a definição de um formato de arquivo de configuração, baseado em XML cuja flexibilidade fornece à plataforma a capacidade de gerar novas aplicações com diferentes funcionalidades de maneira simplificada. Faz parte deste trabalho, apresentar métodos de edição e validação de tais arquivos de configuração bem como a descrição detalhada da arquitetura de *software* da solução proposta para atender aos requisitos apresentados. Para qualificação dos artefatos produzidos são apresentados dados comparativos entre a plataforma proposta e soluções semelhantes além da apresentação e descrição de uma experiência na qual a plataforma desenvolvida é executada em um sistema real de televisão digital. O resultado obtido na experiência demonstra que a plataforma contribui de forma bastante positiva para distribuição gratuita de aplicações para televisão digital por provedores de conteúdo sem que sejam necessários grandes conhecimentos em programação.

Palavras-chave: TV Digital, plataforma, aplicações, componentes, *Quiz*.

Abstract

This study introduces a software framework which permits to generate interactive applications to be executed over digital television systems. It starts from a theoretical contextualization around existing technology focusing on the Brazilian model and on difficulties inherent to software development process for embedded systems. With the need for focus on a specific type of problem this work identifies that applications which gather information from users based on their opinions or specific knowledge are extremely useful applications as can be used for entertainment and education purposes. This type of application is commonly referred as *Quiz* and, its essential features are fully described and taken into consideration for the specification of a configuration file format based on XML which is used to describe applications appearance and behavior. This external configuration file is composed by independent blocks representing visual or functional components which can be easily rearranged resulting in another application. It is also part of this work propose methods for editing and validating configuration files as well as provide a detailed description of the framework architecture which contemplates all requisites presented before. In order to qualify the produced software artifacts this work presents comparative data among the framework and other similar solutions and also describe in details an experience performed to demonstrate the solution working on a real digital TV system. The final result is a functional framework which can definitely contribute, especially to Brazilian content television providers, to easily deploy interactive applications over digital TV systems without require advanced programming skills.

Keywords: Digital TV, application, framework, components, quiz.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

Lista de Siglas

ACAP	*Advanced Common Application Platform*
API	*Application Programming Interface*
APP	*Application*
ARIB	*Association of Radio Industries and Businesses*
ASCII	*American Standard Code for Information Interchange*
ATSC	*Advanced Television Systems Committee*
AWT	*Abstract Window Toolkit*
BST-OFDM	*Band Segmented Transmission-OFDM*
COFDM	*Coded Orthogonal Frequency Division Modulation*
DASE	*DTV Application Software Environment*
DAVIC	*Digital Audio Visual Council*
DSM-CC	*Digital Storage Media-Command and Control*
DSP	*Digital Signal Processor*
DTD	*Document Type Definition*
DVB	*Digital Video Broadcast*
DVB SI	*DVB Service Information*
DVD	*Digital Video Disk*
ECMA	*European Computer Manufacturers Association*
EDTV	*Enhanced Definition Television*
EMTV	Extensão De Middleware Para Televisão Digital Brasileira
EPG	*Electronic Programming Guide*
ES	*Elementary Stream*
FCC	*Federal Communications Commission*
FDM	*Frequency-Division Multiplexing*
FM	*Frequency Modulation*
GBPS	*Giga Bits Per Second*
GEM	*Globally Executable MHP*
GIF	*Graphics Interchange Format*
GIFT	*General Import Format Template*
HAVI	*Home Audio Video Interoperability*
HDTV	*High Definition Television*
HTTP	*HyperText Transfer Protocol*
HW	*Hardware*
IPTV	*Internet Protocol Television*
ISDB	*Integrated Services of Digital Broadcasting Terrestrial*
ISDTV	International System for Digital TV
ISO/IEC	*International Organization for Standardization/International Electro technical Commission*
ITU-R	*International Telecommunications Union, Radio communication sector*
J2ME	*Java 2 Micro Edition*
JMF	*Java Media Framework*
JPG	*Joint Photographic Experts Group*
LDTV	*Low Definition Television*
LFE	*Low Frequency Enhancement*

LGPL	*Lesser General Public License*
LIB	*Software Library*
LUA	*A Programming Language defined at www.lua.org*
MBPS	*Mega Bits Per Second*
MP3	*MPEG-1/2 Layer III*
MPEG	*Moving Pictures Expert Group*
MPEG NBC	*Moving Pictures Expert Group Non-Backward Compatible*
Moodle	*Multi-user Object Oriented distributed learning environment*
MUSE	*Multiple Sub-Nyquist Sampling Encoding*
NCL	*Nexted Context Language*
NTSC	*National Television System Committee*
OCAP	*OpenCable Application Platform*
PAL	*Phase Alternation Line*
PAT	*Program Association Table*
PNG	*Portable Network Graphic*
PMT	*Program Map Table*
PS	*Program Stream*
PSI	*Program Specific Information*
PVR	*Personal Vídeo Recorder*
QAM	*Quadrature Amplitude Modulation*
QPSK	*Quadrature Phase Shift Keying*
SAP	*Second Audio Program*
SDTV	*Standard Definition Television*
SECAM	*Sequencial Couleur Avec Mémoire*
STB	*Set-Top Box*
SW	*Software*
TCP/IP	*Transport Control Protocol/Internet Protocol*
TP	*Transport Packet*
TS	*Transport Stream*
TVD	*TV Digital*
TVD	*TV Digital Interativa*
URL	*Uniform Resource Locator*
VSB	*Vestigial Side Band*
XML	*eXtensible Markup Language*
WWW	*World Wide Web*

1. Introdução

Estamos passando, no Brasil e no mundo, por um momento de intensas disputas e negociações entre empresas, entidades de pesquisa e governos em razão da padronização dos novos sistemas abertos de televisão utilizando tecnologia digital em substituição dos sistemas analógicos que vêm sendo utilizados a mais de 50 anos. O motivo para tais discussões, encontradas em várias referências tais como [2], [6], [7], [8] e [9], é bastante justificável dado que, conforme MACHADO (2000) [10], "desde que foi inaugurada, há meio século, a televisão não deixou de crescer em importância, a ponto de firmar-se hoje como meio de comunicação de maior influência nos costumes e opinião pública". Em países desenvolvidos existem pesquisas que indicam a existência de um número maior de aparelhos de televisão do que o número de pessoas. Por exemplo, nos EUA a conclusão do Instituto Nielsen Media Research [1] é que em média existem 2,73 televisores numa típica residência americana, contra 2,55 pessoas. Essa média tende a ser alcançada mesmo em países em desenvolvimento como é o caso do Brasil onde a estimativa é que existam aparelhos de televisão em pelo menos 95,7% dos lares, conforme dados encontrados em [2] e [3]. Levando-se em consideração que uma porcentagem muito menor da população, cerca de 15% no mundo [4], possui computadores pessoais e que, ainda assim, a Internet [5] é considerada uma das maiores revoluções tecnológicas dos últimos tempos, podemos esperar que a TV Digital, sendo baseada no princípio da convergência digital de conteúdos, venha a se tornar uma revolução ainda maior.

O apelo central da TV digital é que a transmissão dos sinais captados pelos aparelhos deixe de ser analógica, mais sujeita a ruídos e pouco eficiente quanto ao aproveitamento da banda, e passe a ser digital, em que são empregadas modernas técnicas para detecção e remoção de ruídos e, a banda disponível é bem melhor aproveitada graças ao emprego de métodos para compressão de informações digitais. Com tudo isso, a tecnologia de TV Digital possibilita transmitir, na mesma largura de banda utilizada pelos sistemas analógicos, um conteúdo bem maior de informações ao receptor, proporcionando qualidade de áudio e vídeo superiores em comparação ao sinal analógico além de permitir a transmissão de um sinal adicional, chamado sinal de dados, unidirecional tal como os sinais de áudio e vídeo.

O conteúdo complementar do sinal de dados pode ser utilizado nos aparelhos receptores de forma inovadora, pois a princípio, pode conter documentos, imagens, sons e principalmente códigos que ao serem interpretados podem gerar aplicativos para interação com o usuário com os mais diferentes propósitos [12]. Isso expande os horizontes das geradoras de conteúdo que podem oferecer aos seus espectadores uma enorme quantidade de produtos, serviços, informação e entretenimento.

A TVD permite que o usuário não apenas se beneficie dos recursos do sinal digital, por exemplo, seleção de diferentes ângulos de uma mesma cena e seleção de informações adicionais sobre as cenas exibidas como também oferece novas possibilidades em torno de outra inovação da TV digital, que é a utilização de um canal de retorno. O canal de retorno normalmente é estabelecido através de uma rede de comunicação de dados TCP/IP [12] e serve para envio de informações fornecidas pelos usuários [13] ou para recebimento de dados para gerar aplicativos. É nesse último caso que fica mais amplamente caracterizada a convergência da tecnologia digital. O aparelho de televisão finalmente terá o potencial de exibir praticamente todo tipo de conteúdo, tal como a Internet.

A contribuição desse trabalho se aterá a esse terceiro componente do sinal de TV digital, constituído pelo canal de dados e pelo canal de retorno, e de que forma ele pode ser utilizado para geração de aplicações interativas especialmente aquelas que trazem algum benefício social por servirem como ferramentas complementares para a educação.

1.1 Motivação

A situação do Brasil nesse momento é bastante propícia para a realização de trabalhos nessa área do conhecimento. O país concretizou suas escolhas tecnológicas e decidiu que as transmissões digitais fossem iniciadas ainda em 2007. Foi bastante noticiada durante todo aquele ano, a relevância do assunto para o governo, empresários e para a sociedade em geral tendo em vista a movimentação econômica com a venda de novos equipamentos e serviços. O governo brasileiro concedeu empréstimos e benefícios para aquisição de equipamentos, serviços e também para formação de mão de obra qualificada em contrapartida ao compromisso dos beneficiários de implantar e democratizar a TVD no país. Apesar da existência de diversos recursos, criados principalmente em países estrangeiros, a novidade da tecnologia no país ainda traz a expectativa de que muitas inovações propostas possam contribuir para evitar a total dependência tecnológica do país acerca da questão.

Tendo em vista esse contexto é possível identificar que uma contribuição possível é abordar a questão do desenvolvimento de aplicativos de *software* que serão oferecidos para interação entre o sistema e os usuários. Através do *software* interativo, temas de interesse social, como a educação à distância, poderão avançar e contribuir positivamente para que o país tenha uma sociedade mais justa. Entretanto a atividade de desenvolvimento de *software* para TVD não é uma tarefa trivial, pois exige mão de obra especializada além de ferramentas que normalmente não têm preços acessíveis para instituições de ensino, que são aquelas geralmente à frente da questão da educação à distância.

As principais motivações para esse trabalho são: a importância do tema para o país nesse momento; a necessidade de formação de mão de obra especializada sobre o assunto, minimizando a depedência sobre tecnologias estrangeiras; a potencialidade do *software* interativo para o processo educativo e a necessidade de serem oferecidas soluções sem custo e que facilitem o desenvolvimento desse tipo de *software* contribuindo assim para democratizar o acesso à tecnologia ao mesmo tempo em que presta um serviço útil à sociedade.

1.2 Objetivos e contribuição

O principal objetivo a ser atingido com esse trabalho é especificar e desenvolver uma plataforma para geração de aplicações interativas para serem executadas em sistemas de TVD levando-se em consideração os seguintes aspectos:

- A plataforma visará a geração de aplicações interativas para TVD brasileira.
- Será uma plataforma sem custo voltada para provedores de conteúdo que possuam pouco ou nenhum conhecimento em programação.
- Deverá fazer parte dos objetivos da plataforma gerar aplicações gráficas com boa aparência visual.

- A plataforma deverá oferecer mecanismos que lhe permita ser reutilizada para produzir aplicações com conteúdo, comportamento e aparência diferentes.
- As aplicações geradas pela plataforma serão, ao menos inicialmente, voltadas para fins educativos devido ao caráter social envolvido nessa escolha. Nesse aspecto também é um obtivo desse trabalho identificar um tipo de aplição interativa que seja relevante em processos educativos.
- A plataforma terá que ser funcionar em um sistema real de TVD por isso terá que levar em consideração a necessidade de minimizar tamanho físico em *bytes,* consumo de memória e de processamento.
- Esse trabalho deverá fornecer as ferramentas e métodos necessários para facilitar a utilização da plataforma produzida.

São esperadas duas contribuições na conclusão deste trabalho:

- Que a plataforma desenvolvida possa ser realmente útil para geração, rápida e fácil, de aplicações interativas com propósito educativo para serem executadas em sistemas de TVD.
- Que essa plataforma sirva como base para realização de trabalhos futuros nessa área na UFAM ou em outras instituições que por ventura se interessarem.

1.3 Metodologia

Inicialmente foi realizado um estudo do estado da arte das tecnologias existentes no mundo para sistemas de televisão digital abertos. O objetivo foi tomar conhecimento das maneiras como as aplicações interativas podem ser geradas, distribuídas e executadas nesses sistemas. Isso incluiu tomar conhecimento das características de cada padrão tendo esclarecido suas diferenças e as implicações das escolhas brasileiras. Essa parte inicial permitiu identificar as dificuldades envolvidas na construção de aplicações, quais bibliotecas de *software* são utilizadas e quais ferramentas para desenvolvimento e depuração estavam disponíveis para possibilitar a realização desse trabalho.

Em seguida, foram empregados esforços para definição das melhores escolhas técnicas a serem empregadas. Ainda nessa fase, apesar da utilização de uma ferramenta de emulação de um sistema real, foram gerados protótipos que foram utilizados para provar conceitos e adquirir proficiência no uso das interfaces de *software*. Esses produtos, apesar de terem sido importantes, não são relacionados como parte da plataforma.

Posteriormente foi necessário identificar, entre um grande número de possibilidades, uma família de aplicações que, de acordo com os objetivos básicos do estudo, tivessem utilidade dentro um contexto educativo. Um estudo acerca dessa questão apontou que aplicações tipo *Quiz* não só se enquadram em processos educativos como também se prestam a outras aplicações o que torna o foco nesse tipo de aplicação bastante conveniente . A identificação da relevância dos *Quizzes* permitiu a realização de duas atividades subseqüentes:

- Identificar todas as características obrigatórias e desejáveis que atendessem aos objetivos deste estudo.

- Fornecer informações necessárias para definir como flexibilizar o funcionamento da plataforma. Essa tarefa foi auxiliada por um rápido estudo sobre os métodos utilizados por outras plataformas com propósitos semelhantes, mesmo que se destinassem a execução em sistemas diferentes, por exemplo, na Internet.

As atividades listadas acima permitiram a modelagem da arquitetura que foi utilizada durante toda a fase seguinte que foi a implementação da plataforma. É importante esclarecer que, ainda nessa fase, foi utilizada uma ferramenta para emulação de uma plataforma de televisão digital. Durante o percurso de implementação foram definidos métodos e ferramentas que facilitassem o uso da plataforma.

Tendo a plataforma desejada totalmente funcional partiu-se para realização de experimentos no laboratório de televisão digital da UFAM onde foi possível depurar, detectar e efetuar as alterações necessárias para que a plataforma funcionasse em um sistema real.

A etapa final consistiu em qualificar a plataforma desenvolvida, buscando compará-la com outros projetos semelhantes, sendo possível apontar as vantagens e desvantagens obtidas com seu uso.

1.4 Organização dos capítulos

O restante do texto está organizado de acordo com a metodologia apresentada. O capítulo 2 faz uma breve análise dos conceitos básicos utilizados em sistemas de televisão, desde suas origens até os padrões de transmissão de TVD aberta, demonstrando o que torna essa tecnologia tão atrativa e inovadora. No capítulo 3 o enfoque será dado sobre o *middleware* que é um componente de sistemas de TVD que possibilita a conversão de informações fornecidas através do canal de dados em aplicações interativas. Esse capítulo apresenta conceitos e fundamentos dos principais *middleware* abertos em uso no mundo na atualidade, o que permite identificar qual, dentre estes *middleware*, possui maiores vantagens para ser utilizado no desenvolvimento da plataforma que se deseja obter. O capítulo 4 faz um rápido estudo sobre aplicações interativas, aborda conceitos importantes e procura justificar qual tipo de aplicação tem maior utilidade em processos educativos. O tipo de aplicação identificada é completamente caracterizada ainda no capítulo 4 o que será útil na definição da plataforma. O capítulo 5 apresenta a proposta conceitual da plataforma se valendo das conclusões tiradas nos capítulos 3 e 4 e dos conceitos fornecidos no capítulo 2. O capítulo 6 define detalhes finais da arquitetura e parte para a descrição da implementação do software proposto. O capítulo 7 traz aspectos relativos ao uso da plataforma, os experimentos realizados para a validação do funcionamento do sistema conforme as especificações e finalmente apresenta comparações com sistemas com funcionalidades semelhantes. O capítulo 8 faz uma revisão geral dos objetivos e do que foi obtido além de registrar os pontos de melhorias e dificuladades encontradas no desenvolvimento desse trabalho. Esse trabalho se encerra com a apresentação da bibliografia e de alguns anexos que fornecem algumas informações adicionais relevantes.

2 Conceitos Iniciais

Esse capítulo apresenta conceitos necessários para o entendimento das tecnologias envolvidas em sistemas de TVD. Para tanto parte da apresentação do princípio básico de funcionamento dos sistemas analógicos e da evolução histórica até os sistemas digitais. Segue uma introdução dos conceitos mais importantes na tecnologia digital como MPEG-2 e MPEG-4, as técnicas de modulação para transmissão de sinais de TVD disponíveis na atualidade, suas vantagens e desvantagens. Todos os conceitos são então utilizados para a caracterização dos três principais padrões abertos de TVD mais a solução adotada no Brasil.

2.1 Origens dos sistemas de televisão

Os sistemas de televisão começaram a ser desenvolvidos a partir do final da década de 20 do século passado através de experiências inovadoras, como a do engenheiro escocês John Logie Baird, que demonstrou em Londres um sistema completo de transmissão e recepção de um sinal composto de áudio e imagem em preto e branco [17]. Este sistema esse baseava na caputra, transmissão, recepção e reprodução de sons e imagens em preto e branco, ou seja tons de cinza. Alguns conceitos adotados até os dias atuais foram apresentados nesse momento [15]:

- Define-se que o menor elemento distinguível em uma imagem é chamado *pixel* e corresponde a um ponto na imagem. A distância entre dois pixels consecutivos é chamada de "*dot*" e essa definição está intimamente ligada à resolução da imagem, ou seja, a qualidade da imagem visualizada.
- Utiliza-se uma câmera com capacidade para capturar cenas estáticas, como se fossem fotografias, com velocidade superior à taxa de persistência do olho humano.
- As imagens capturadas pela câmera têm formatos retangulares e tamanhos limitados em altura e largura. Sendo assim, é importante compreender que cada imagem é formada por um conjunto de linhas de *pixels*.
- A câmera é também capaz de registrar, seguindo as características da curva de luminância do olho humano, a intensidade luminosa de cada *pixel* na imagem. Quanto mais claro, ou seja, mais próximo do branco, maior o valor da intensidade luminosa do *pixel*. A câmera realiza essa varredura *pixel* por *pixel* de cada linha da imagem.
- Os valores de intensidade luminosa de cada *pixel* de cada linha da imagem são transcritos em um sinal elétrico que pode ser modulado e transmitido.
- No lado do receptor, que é o aparelho de televisão preto e branco, é necessário que os sinais elétricos correspondentes a cada *pixel* da imagem sejam impressos na tela. Para isso utiliza-se um tubo de raios catódicos em que o ângulo de emissão é dependente de uma variável no tempo e a intensidade do feixe é controlada pela amplitude do sinal elétrico de entrada, representando a intensidade luminosa de cada *pixel* da imagem.
- Todas as linhas são impressas na tela com velocidade muito superior à taxa de persistência do olho humano de modo que seja criada a sensação de movimento que existia de fato no momento da criação da cena. Taxas de 30 imagens, ou quadros, por segundo são suficientes para garantir essa sensação.

- Nesse sistema o sinal de áudio é transmitido separadamente, porém utilizando freqüências de portadoras próximas ao do sinal de vídeo através dos métodos já utilizados nos sistemas de rádio.

A Figura 1, proveniente de [15], ilustra o sinal elétrico resultante da varredura de uma linha de *pixels* de uma imagem. Nela, também estão ilustrados outros sinais necessários para a sincronização do sinal com o processo de regeneração do sinal no lado do receptor. Isso é necessário para garantir que um televisor ao ser ligado num momento aleatório da transmissão seja capaz de sincronizar a posição do *pixel* na tela. Entretanto essa característica só foi introduzida alguns anos depois como um processo de inovação das transmissões. No início, havia a necessidade de um ajuste manual do usuário. Outra característica existente desde o início dos sistemas de televisão [15] é realizar primeiro a varredura das linhas pares e em seguida das linhas ímpares, constituindo os chamados campos, de modo que uma imagem completa é formada por um campo par e um campo ímpar. Esse mecanismo é chamado de varredura entrelaçada, ou intercalada [15] e [17], e é utilizado até os dias atuais.

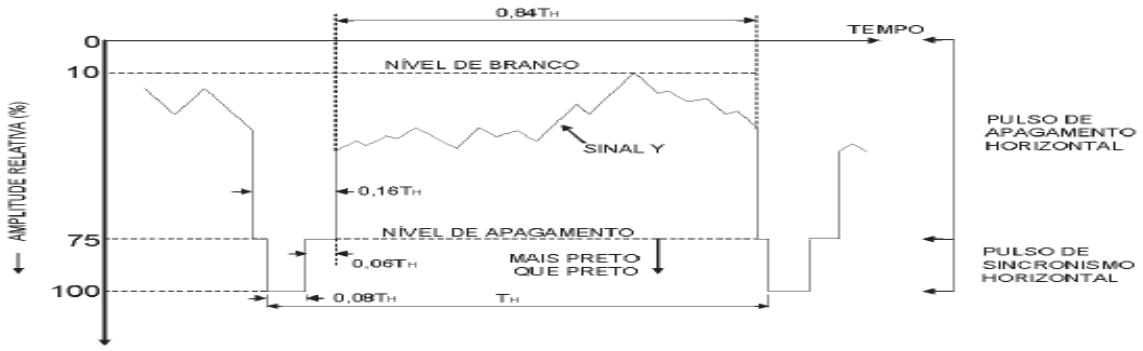

Figura 1 – Aspecto de um sinal de vídeo em P&B durante uma varredura horizontal

As transmissões comerciais de sinais de televisão em preto e branco iniciaram-se nos Estados Unidos em 1947 com um padrão chamado de padrão "M" que também foi adotado pelo Brasil, quando do lançamento da primeira estação de televisão do país em 1952 [15]. Ele estabelecia a transmissão de imagens com relação de aspecto de 4:3, com 525 linhas e 30 quadros por segundo, conforme encontrado em [12], [15] e [17].

Em 1954 também nos Estados Unidos foi definido o primeiro padrão para transmissão de sinais de televisão em cores que recebeu o nome do comitê que o definiu *National Television System Committee* (NTSC). Esse sistema se baseia na existência no tubo de imagens que o *pixel* possua elementos químicos capazes de emitir luz no espectro do vermelho, verde e azul e que sendo bombardeados com diferentes intensidades formam as cores dentro espectro visível ao olho humano. Foi necessário adaptar as câmeras para obtenção da informação das intensidades de cores de cada *pixel* de imagem e também adaptar o sistema de transmissão tendo a dificuldade adicional de se manter compatível com os sistemas em preto e branco já bastante difundidos. A solução encontrada foi transmitir o sinal "Y" utilizado pelos receptores preto e branco seguindo a Equação 1, que leva em consideração a relação do número de cones existentes no olho para captação de cada tipo de cor. A utilização desse artifício permitiu que os televisores em preto e branco continuassem funcionando alheios à modificação e os aparelhos coloridos poderiam extrair as informações de cores através de manipulações matemáticas com outros dois sinais

adicionais, (R-Y) e (B-Y), denominados sinais diferença e que não interferiam nos receptores em preto e branco.

$$Y = 0{,}30R+0{,}59G+0{,}11B$$

Equação 1- Y: Sinal de Luminância; R,G,B: Nível de vermelho, verde e azul

Naquela época o sistema de cores definido pelo NTSC era considerado deficiente na fidelidade das cores e isso motivou o surgimento dos padrões alemão *Phase Alternation Line* (PAL) e *Sequencial Couleur Avec Mémoire* (SECAM) ambos em 1967 e considerados tecnicamente melhores que o NTSC. Esses formatos, assim como o americano, eram compatíveis com os sistemas em preto e branco adotados em seus países respectivamente "G" e "L" que diferem do padrão "M" basicamente por utilizarem 625 linhas e apresentarem 50 imagens por segundo.

Características	PAL M	PAL B-G-H	SECAM B-G-H	NTSC M
N. de Linhas de um Quadro	525	625	625	525
N. de Quadros por seg.	30	25	25	30
Freqüência Horizontal	15.734 kHz	15.625 kHz	15.625 kHz	15.734 kHz
Freqüência Vertical	60 Hz	50 Hz	50 Hz	60 Hz
Modulação de Vídeo	AM-VSB	AM-VSB	AM-VSB	AM-VSB
Largura faixa sinal vídeo	4.2 MHz	5.0 MHz	5.0 MHz	4.2 MHz
Freq. Sub-portadora de áudio	4.5 MHz (FM)	5.5 MHz (FM)	5.5 MHz (AM)	4.5 MHz (FM)
Freq. Sub-portadora de cor	3.575611 MHz.	4.433618 MHz	4.433618 MHz	3.579545 MHz
Largura de faixa do canal	6 MHz	8 MHz (G-H) 7 MHz (B)	8 MHz (G-H) 7 MHz (B)	6 MHz

Tabela 1 – Principais características dos sistemas analógicos de televisão aberta

O Brasil em 1974, foi o único país do mundo a adotar padrão de cores alemão PAL mantendo a compatibilidade com o padrão em preto e branco americano, gerando um híbrido PAL-M que é utilizado no país até os dias atuais [10],[12],[15]. A Tabela 1, encontrada em [18], resume as características dos três principais padrões de televisão aberta analógica que ainda são utilizados.

2.2 Origens dos sistemas de televisão digital

As primeiras iniciativas para promoção de uma evolução nos sinais de televisão foram realizadas pelos japoneses logo no início da década de 80. Eles experimentaram um sistema, ainda analógico, chamado de *Multiple Sub-Nyquist Sampling Encoding* (MUSE) que utilizava canais de 27 MHz, ou seja, quase 5 canais analógicos, para transmitir 1125 linhas por quadro, e com uma relação de aspecto de 2:1 e taxa de 30 quadros por segundo. Outro projeto que também seguia essa mesma idéia era europeu e tinha o nome de Eureka [14], mas foi rapidamente abandonado quando grandes empresas do setor eletroeletrônico passaram a dar maior atenção às

discussões americanas para desenvolver um sistema de televisão totalmente digital a partir de 1991.

O governo americano tinha como objetivo realizar a transmissão de sinais de imagem em alta definição, tipicamente EDTV e HDTV [13] e, para tanto, definiu que independentemente da tecnologia a ser adotada, teria que ser totalmente compatível com o padrão NTSC até então utilizado naquele país. Isso acabou com as expectativas japonesas de que o MUSE fosse considerado no processo. A união dos fabricantes em torno desse objetivo, ficou conhecida como a "Grande Aliança" [14] e resultou num padrão que foi então aceito pelo *Advanced Television Systems Committee* (ATSC), que deu o nome ao sistema americano.

Um novo padrão europeu, semelhante ao americano, surgiu no mesmo ano sendo denominado de *Digital Vídeo Broadcasting* (DVB) e, o padrão japonês inteiramente digital, veio apenas em 1999 sendo denominado de *Integrated Services of Digital Broadcasting Terrestrial* (ISDB) [11],[12] e [19]. Durante esse processo evolutivo foram criados novos conceitos, entre eles a denominação padrão de algumas resoluções de imagem, sugerida pela ITU-R BT.1125 conforme a Tabela 2, e baseada em [17] e [21].

Denominação	Relação Aspecto	Linhas/ Quadro	Pixels/ linha	Quadros/s	Som
High Definition Television (HDTV) (Alta definição de som e imagem)	16:9	1080	1920	30	Surround (5.1)
Enhanced Definition Television (EDTV) (Alta definição com menor n. linhas)	16:9	720	1280	30	Surround (5.1)
Standard Definition Television (SDTV) (Qualidade equivalente da TV analógica)	4:3	480	640	30	Stéreo
Low Definition Television (LDTV) (Para utilização em terminais celulares)	4:3	240	320	30	Mono

Tabela 2 – Resoluções padrão para sinais de vídeo

O Brasil começou seus estudos sobre TVD apenas em 1998, e após cerca de 3 anos de pesquisa chegou a esboçar um sistema genuinamente nacional visto que contava com um modulador com características inovadoras chamado de Sistema OFDM com Redução de Complexidade por Equalização Robusta (SORCER), que emprega técnicas de inteligência artificial no receptor visando melhoria de qualidade do sinal obtido. Além disso, instituições de pesquisa nacionais, desenvolveram uma importante parte do sistema de TVD chamado de GINGA [40], que será melhor abordado na seção 3.4 desse documento, mas cuja atribuição é gerar de aplicações interativas para apresentar aos usuários. Ao final de 2006 o governo brasileiro, após grandes disputas por parte de empresas e radio difusores, decidiu pela adoção do padrão japonês ISDB, que contava com o GINGA [40] e que, inicialmente foi chamado de *International System for Digital TV* (ISDTV) [40]. Posteriormente ficou acertado que o nome do sistema deve ser conhecido como ISDB Internacional.

As experiências japonesas com o MUSE foram importantes, pois despertaram o interesse do mundo pelo som e imagem de maior qualidade. Também naquele momento ficou claro que a utilização de canais de 27 MHz seria praticamente inviável [15]. A simples digitalização dos sinais analógicos, consistido apenas do processo de amostragem e quantização, também não bastaria pois resultaria em taxas da ordem de 360 Mbps para sinais de televisão com resolução

convencional e da ordem de 1.24 Gbps para sinais de HDTV [15]. Ambas as estimativas considerando a representação de 10 bits/pixel e demais características de acordo com a Tabela 2.

$$C = B \log2 (1 + S/N)$$

Equação 2 - Capacidade do Canal; B: Largura de Banda; S/N: Relação Sinal/Ruído

De acordo com a Equação 2, formulada pelo teorema de Shannon, a capacidade máxima de transmissão de um canal está intimamente ligada à relação sinal/ruído do canal de transmissão, portanto é bastante dependente da codificação e modulação utilizada. Entre as técnicas de modulação digital voltadas para transmissão de sinais de televisão temos: *Quadrature Amplitude Modulation* (QAM), *Quadrature Phase Shift Keying* (QPSK), *Coded Orthogonal Frequency Division Multiplexing* (COFDM) e *8 Vestigial Side Band* (8-VSB) entre outras. Cada um desses tipos de modulação é adequado para condições específicas, conforme item 2.4 desse mesmo texto. Mas em geral, a utilização de tais técnicas em canais de 6 MHz provê taxas de transmissão digital da ordem de 20 Mbps. Ou seja, a simples digitalização e a aplicação das melhores técnicas de modulação são insuficientes para viabilização da televisão digital.

A solução veio com o aumento da eficiência nas informações transmitidas proporcionada pela compressão dos sinais digitais. O princípio é simples, e baseia-se em retirar ao máximo a quantidade de informação redundante dos sinais de entrada, de modo que, a informação original possa ser recuperada de modo satisfatório no receptor [15], que nos sistemas de TVD é conhecido como *Set-top Box* (STB). Esse momento, de discussão dos primeiros sistemas de televisão digitais, coincidiu com ascensão da Internet em que já existia a mesma preocupação com a compressão de informações digitais. Em 1986, surgiu um grupo dentro do ISO/IEC chamado de *Joint Photograph Experts Group* (JPEG) que tinha como propósito estudar processos de compressão aplicados a imagens estáticas ou fotografias. Motivados pelos bons resultados, em 1988 o ISO/IEC destaca um novo grupo de estudos agora chamado de *Moving Pictures Experts Group* [20] (MPEG) que iniciou uma pesquisa mais generalizada para compressão de sinais multimídia.

O MPEG definiu uma família de padrões e, entre eles, existe o MPEG-2 [31], aprovado no final de 1995 e que trata especificamente de especificações e técnicas para uso em sistemas de televisão e DVD. O padrão MPEG-2 baseia-se na utilização de técnicas de compressão, descompressão, processamento e codificação de sinais de vídeo que levam em consideração as redundâncias espaciais e temporais dos sinais. O uso de técnicas definidas no MPEG-2 em sinais de vídeo HDTV, conforme definido na Tabela 2, resulta um sinal de saída, denominado *Transport Stream* (TS), com taxas que variam entre 16 e 20 Mbps, portanto dentro da capacidade do canal de 6 MHz para as modulações digitais disponíveis. A Tabela 3, baseada em [15] e [19], fornece uma noção do nível de compressão que pode ser obtido através do uso de técnicas definidas no MPEG-2.

	Sinal digital sem compressão Com 10 bits/pixel	Sinal digital comprimido Utilizando MPEG-2
HDTV	1,24 Gbps	16 ~ 20 Mbps
EDTV	1,1 Gbps	12 ~ 16 Mbps
SDTV	270 Mbps	8 ~ 12 Mbps
LDTV	46,08 Mbps	4 ~ 8 Mbps

Tabela 3 – Nível de compressão MPEG-2 para sinais ITU-R BT.1125

A compressão dos sinais, através de MPEG-2 [20], favorece a possibilidade de agregação de diferentes combinações de sinais de televisão dentro de um mesmo TS. Essa situação é ilustrada na Figura 2 [19].

Entretanto a forma de utilização do canal é definida pelos comitês gestores de cada padrão de TVD sendo regido por modelos de negócios regionais. Por exemplo, enquanto nos EUA o ATSC decidiu privilegiar apenas a transmissão contínua de sinais de alta definição, na Europa o DVB privilegia a diversidade de programação oferecendo normalmente 4 sinais SDTV no canal. Por outro lado, na Austrália [6], apesar de seu modelo ser baseado no DVB, a ocupação do canal é determinada pelo interesse dos geradores de sinal, de modo que há programas gerados em alta definição e outros que não são, oferecendo taxas que melhor convierem às diferentes ocasiões. O modelo adotado pelo Brasil [12], nesse aspecto, assemelha-se mais ao modelo Australiano.

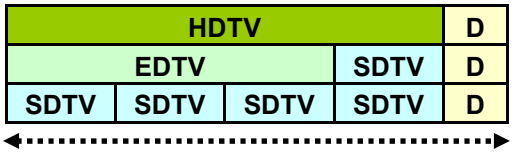

Figura 2 – Possibilidades de uso de canais com sinais ITU-R BT.1125

2.3 Conceitos utilizados no MPEG-2

Por ser utilizado como base para sistemas de TVD, apesar de não pertencer ao foco desse trabalho, é importante compreender pelo menos os conceitos básicos do MPEG-2 [22]. Ele é constituído por 9 especificações [31] e, dentre estas, 4 são especialmente importantes para o funcionamento de sistemas de TVD, e estão listadas na Tabela 4, baseada em [22] e [31]:

Especificação	Nº. Especificação	Atribuições
MPEG-2 Sistemas	ISO/IEC 13818-1	Multiplexação dos sinais de áudio, vídeo e dados.
MPEG-2 Vídeo	ISO/IEC 13818-2	Codificação de vídeo.
MPEG-2 Áudio	ISO/IEC 13818-3	Codificação de áudio.
DSM-CC	ISO/IEC 13818-6	A especificação do *Digital Storage Media-Command and Control* (DSM-CC) define entre outras coisas, como distribuir grandes quantidades de dados através do sistema de maneira cíclica.

Tabela 4 – Principais especificações do MPEG-2 para TVD

2.3.1 MPEG-2 Sistemas:

O principal papel do módulo MPEG-2 sistemas é realizar a multiplexação de diferentes sinais de entrada em um único sinal de saída. Dependendo do emprego do sistema o processo de multiplexação pode resultar em dois tipos diferentes de sinal de saída [12], [22] e [36]:

- *Transport Stream* (TS): Resultado de um processo de multiplexação mais adequado para realização do transporte do sinal. Nesse caso esses sinais estão sujeitos a apresentarem erros e por isso há a inserção de informações que ajudam na detecção de erros e na sincronização no receptor. Esse é o sinal utilizado para sistemas de TVD.
- *Program Stream* (PS): Resultado de um processo de multiplexação otimizado para baixa possibilidade de incidência de erros. Esse é o sinal gerado para uso em sistemas de DVD.

Os sinais de entrada do MPEG-2 Sistemas, áudio, vídeo e dados ao serem codificados formam fluxos contínuos de informações denominados de *Elementary Streams* (ES). Logo no início do processo os ES são segmentados em pequenos pacotes com alguns bytes de cabeçalho e dessa forma passam a ser chamados de *Packet Elementary Streams* (PES). Os PES carregam referencias de hora do pacote, ou *Time Stamp* [12].

Seguindo o processo, um multiplexador baseado em tempo coleta os vários PES e gera um fluxo contínuo chamado de *Transport Stream* (TS), conforme Figura 3 [12]. No TS os PES recebem mais bits extras de cabeçalho e passam a ser denominados *Transport Packet* (TP). Cada TP tem um tamanho fixo de 188 bytes a fim de facilitar o processo de sincronização, detecção de erros e correção no receptor [12] e [36].

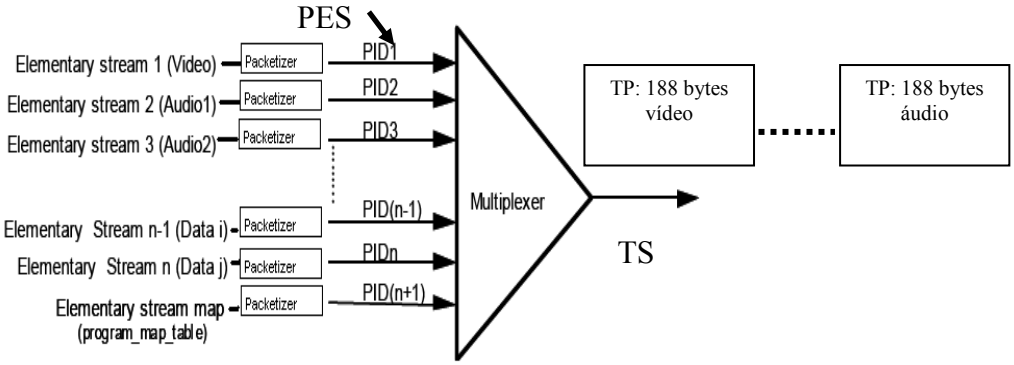

Figura 3 – Estrutura do sinal resultante do MPEG-2 Sistemas

Um TS, além dos PES, é também constituído por pacotes que prestam informações importantes sobre os PES que estão sendo transmitidos [24]. Essas informações são chamadas de *Program Specific Information* (PSI). Para TVD existem duas séries principais de bytes representando a PSI [24] e [36]:

- *Program Association Table* (PAT): Essa informação recebe sempre o PID "0". Nela está contida uma tabela com os PID de todos os PMT contidos no TS.
- *Program Map Table* (PMT): Há um PMT para cada programa sendo transmitido juntamente com o TS. Em uma PMT está contida a lista de PID dos PES que constituem um programa e entre outras coisas também permite identificar o tipo do PES para cada PID, ou seja, classifica se o PES trata-se de um pacote de áudio, vídeo ou dados.

2.3.2 MPEG-2 Vídeo:

O MPEG-2 vídeo é uma especificação cujo objetivo é realizar a compressão do sinal de entrada abordando três aspectos [32] e [33]:

- Redução das redundâncias espaciais (intra-quadro ou intra-frame): A análise sobre cada quadro do sinal de modo a identificar pixels vizinhos que sejam iguais ou muito semelhantes. Essa operação é realizada em algumas etapas, entretanto a técnica central utilizada é a Transformada Discreta Co-seno (DCT) nas imagens. A DCT é calculada para cada segmento da imagem tipicamente de 8 x 8 pixels. O resultado do cálculo em cada segmento são coeficientes que determinam o quanto os pixels de um determinado quadro são semelhantes entre si. Nesse processo, vários coeficientes obtidos têm valor zero e isso é explorado para diminuir a quantidade de informação que necessita ser transmitida. A ordem dos pixels que são levados em consideração no cálculo da DCT é bastante importante pois determina a eficiência do método em encontrar as redundâncias. Existem várias técnicas que são mais adequadas a diferentes tipos de imagem, entretanto a DCT no MPEG-2 vídeo, normalmente utiliza a ordem produzida através da varredura chamada "zig-zag em 45º" ilustrada na Figura 4 [33].

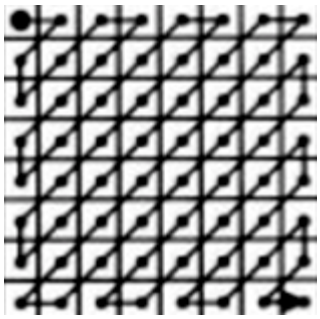

Figura 4 – Varredura de pixels em zig-zag de um quadro 8x8 pixels

- Redução das redundâncias temporais, inter-quadro ou inter-frame: Análise sobre as diferenças entre os quadros adjacentes que também mantém grandes semelhanças entre si. Mais uma vez a análise entre um quadro anterior e um posterior é realizada sobre blocos 8x8 correspondentes nas imagens consideradas que normalmente já passaram pela retirada das redundâncias temporais.

- Redução das redundâncias estatísticas: Essa análise no MPEG-2 vídeo é tipicamente chamada de estimação do movimento. Trata-se da realização de cálculos estatísticos sobre uma seqüência de imagens e com isso tentar predizer qual será a posição que determinados objetos assumirão na tela. Se os cálculos puderem ser validados na transmissão é possível transmitir apenas as informações sobre o deslocamento dos objetos que o receptor será capaz de reproduzir a imagem resultante.

É importante dizer que, nesses três processos, apresentados de maneira resumida, existem diversos parâmetros de ajuste que resultam em sinais de vídeo com maior ou menor grau de compressão e com resoluções diretamente proporcionais às taxas obtidas.

2.3.3 MPEG-2 Áudio:

O áudio é de fundamental importância pra que os usuários realmente tenham uma experiência diferente ao assistirem programas em TVD. Enquanto os sinais analógicos estão

providos apenas de um sinal estéreo mais um canal adicional chamado de *Second Audio Program* (SAP) na TVD, a utilização de técnicas de compressão definidas no MPEG-2 áudio não só, ocupam menos banda como também permitem transportar diversos canais independentes para o mesmo programa.

O princípio de funcionamento baseia-se, no conhecimento das características do ouvido humano e na retirada das possíveis redundâncias temporais presentes no sinal. Para tanto, existem diferentes tipos de codificadores que foram homologados pelo MPEG-2 áudio e, em todos eles, o sinal gerado é compatível com o MPEG-2 sistemas.

Por não ser enfoque desse trabalho não serão abordadas as técnicas utilizadas em cada codificador de áudio existente, mas é importante conhecer que existem essencialmente três codificadores compatíveis com MPEG-2 áudio e que são utilizados pelos sistemas de TVD na atualidade:

- *Audio Compression*-3 (AC-3), também conhecido por DOLBY Digital [37]: Formato de propriedade intelectual de empresas americanas. Esse formato não é padronizado pelo ISO/ITU mas é largamente utilizado em cinema e DVD sendo por essa razão bastante comum nos equipamentos de áudio e vídeo presentes no mercado. O sinal AC-3 é composto por 5 canais de áudio de alta qualidade mais 1 canal de baixa qualidade utilizado para aprimoramento de efeitos sonoros, chamado *Low Frequency Enhancement* (LFE). Essa formação é conhecida como sistema 5.1. Nessa configuração o sinal resultante total chega a taxas da ordem de 500 Kbps. O AC-3 é o codificador de áudio normalmente utilizado em sistemas de TVD ATSC. Em 2005 o AC-3 foi aprimorado para uso de técnicas semelhantes às adotadas no MPEG-2 AAC e foi rebatizado de *Enhanced* AC-3 (E AC-3) [37].

- MPEG-2 *Audio Backward Compatible* (MPEG-2 BC): Refere-se a codificadores que são compatíveis com as definições do MPEG-1 [20], entretanto incorporam o sistema 5.1. Dentre os principais codificadores desse tipo está o formato MP3, que já existia na especificação MPEG-1 e foi mantida como MPEG-2 BC [12]. Os sistemas DVB tipicamente utilizam esse tipo de codificação com exceção do DVB adotado na Austrália que adotou o AC-3.

- *Advanced Audio Coding* (AAC): Faz parte do grupo definido como MPEG-2 *Non Backward Compatible* (MPEG-2 NBC) [20], ou seja, que não é compatível com o MPEG-1. Esse codificador foi a grande novidade apresentada na definição do MPEG-2 áudio, pois emprega técnicas bem mais elaboradas de compressão que resultam em taxas de transmissão (*bit rates*) bem menores do que as taxas obtidas para a mesma qualidade com o MPEG-2 BC [20]. O AAC foi especificado para suportar até 48 canais de áudio, 16 canais LFE e até 16 canais adicionais para outras línguas, conhecidos como *overdub*. Apresenta ainda três abordagens diferentes, chamadas *profiles* para serem utilizadas dependendo da capacidade de processamento e memória do hardware utilizado no processo. Por essas razões [12] o AAC tem sido considerado o estado da arte em termos de codificação de áudio e, como é patenteado por um grupo de empresas envolvidas no desenvolvimento, seu uso requer o pagamento de taxas de propriedade intelectual motivo pelo qual, ainda não se popularizou completamente. Esse é o codificador adotado pelo ISDB japonês, também utilizado no Brasil.

2.3.4 DSM-CC:

Digital Storage Media Command and Control (DSM-CC) [22] refere-se a um conjunto de protocolos voltados para controle e gerenciamento de sinais compatíveis com a especificação MPEG-2 sistemas. A necessidade do DSM-CC deve-se ao fato de que a maior parte da banda de um canal TVD é normalmente preenchida com fluxos de áudio e vídeo [22]. A largura de banda reservada para transmissão de dados é bastante reduzida, mesmo porque, além de não conterem informações críticas, as informações do canal de dados ainda podem passar por um processo de armazenamento, ou *buffering,* no lado do receptor que não seriam desejadas na transmissão de áudio e vídeo. Dentre os protocolos definidos no DSM-CC apenas dois são amplamente utilizados em sistemas de TVD, o carrossel de dados e o carrossel de objetos [22], ambos considerados vitais para o funcionamento de aplicações de TVD Interativa.

A idéia básica do carrossel de dados, de acordo com [22] e [36], é realizar a segmentação de todas as informações em pacotes etiquetados, ou seja, marcados, com identificadores únicos para cada programa. O carrossel de dados deve garantir a inserção dos pacotes obtidos de forma cíclica para multiplexação com sinais de áudio e vídeo. Desse modo, no lado do receptor, para que o DSM-CC possa recuperar informações pertinentes a um determinado programa, basta aguardar a chegada dos pacotes com determinado identificador. Se algum erro for detectado em algum pacote basta descartá-lo e aguardar a repetição da transmissão daquele pacote. É essa característica cíclica que atribuiu o nome de "Carrossel" ao protocolo [12], [22] e [35]. É importante ter em vista que os identificadores utilizados para identificação dos pacotes são chamados de *association tags* e não têm os mesmos PID utilizados para identificação dos ES do MPEG–Sistemas apesar de estarem relacionados a eles. A razão disso é evitar que os identificadores dos pacotes do carrossel tenham que ser refeitos toda vez que haja modificação na estrutura do multiplexador do MPEG–Sistemas. A ligação entre os programas e os seus respectivos dados é feita através da inserção do *association tag* usado nos dados, dentro da PMT do programa.

Já o objeto carrossel de objetos é formado sobre a estrutura do carrossel de dados de modo que as informações sejam encapsuladas na forma de objetos. Esses objetos fornecem informações adicionais sobre a natureza dos pacotes do carrossel de dados, como a distinção de arquivos, diretórios e fluxos de dados. A representação dessa forma permite a abstração no lado do receptor de um sistema de arquivos completo facilitando não só o sincronismo como também a referência das aplicações de TVD interativa aos dados que manipula [22], [35] e [36].

2.4 Modulações utilizadas em televisão digital

Como são importantes qualificadores dos sistemas de televisão digital, é importante tomar conhecimento das características básicas dos principais métodos de modulação utilizados em sistemas de TVD.

2.4.1 Quadrature Amplitude Modulation (QAM)

MQAM É uma técnica de modulação digital em que, M possíveis símbolos, constituídos por $\log_2 M$ bits, são representados por um único par de valores para duas portadoras senoidais

com mesma freqüência em quadratura, ou seja, ortogonais (seno e co-seno) correspondendo amplitude e à fase do sinal [32].

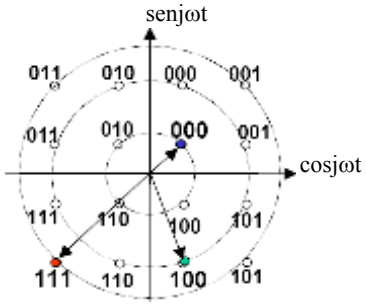

Figura 5 – Ilustração de uma constelação 16 QAM

Esse é considerado um esquema de modulação simples e eficiente quando utilizado por processadores tipo DSP adequados [26] e [32]. A eficiência melhora com a escolha da geometria dos símbolos levando em consideração a energia média entre símbolo. A grande vantagem do QAM é a possibilidade de alcançar altas taxas de transmissão, chegando a 40 Mbps [24]. A desvantagem do QAM é ser bastante susceptível a interferências quando transmitida no espaço livre, se os receptores são móveis. Por essas características o QAM é bastante adequado para transmissão em cabo [24] e [32].

2.4.2 Vestigial Side Band (VSB)

Criada e aprovada por integrantes da "Grande Aliança" a modulação VSB foi adotada apenas pelo padrão ATSC [14]. A modulação propriamente dita é bastante semelhante à AM-VSB que é empregada no sistema analógico do NTSC [19]. A diferença começa pelo sinal de entrada que é um TS em conformidade com o MPEG-2. Esse sinal é imediatamente embaralhado de maneira conveniente, por um módulo chamado *interleaver*, seguido por um módulo codificador, que pode ser [27] o codificador Trellis, ou o codificador Reed Solomon, que inserem informações adicionais utilizadas para detecção e recuperação de erros. Com isso, as características do sinal resultante têm um espectro praticamente plano em toda faixa dos 6 MHz disponibilizadas pelos canais NTSC [27] conferindo ao padrão uma boa eficiência de banda além de garantir uma potência média de transmissão praticamente constante e uma boa imunidade a ruídos de fase.

O sinal VSB possui uma única freqüência portadora piloto de baixa potência, no início da faixa do canal, que é utilizada para facilitar o sincronismo nos receptores além de evitar a interferência com co-canais NTSC [27] e [28]. A Figura 6 [27], ilustra o espectro de um sinal modulado em VSB. Dependendo do número de níveis discretos que são utilizados para representação de um símbolo e do uso ou não da codificação Trellis, o VSB pode ser usado em até 5 modos diferentes (2, 4, 8, 8-Trellis, and 16). Quanto maior o número de níveis, maiores as taxas de transmissão, entretanto, assim como no QAM, um maior número de níveis pode resultar em maiores taxas de erros devido aos efeitos dos ruídos externos ou desvanecimento do sinal. Por essa razão o ATSC definiu o 8-VSB, que usa a codificação Trellis com 8 níveis discretos, para transmissão no espaço livre e, definiu o uso do 16-VSB, que possui 16 níveis discretos, para transmissão via cabo. As capacidades de taxas de transmissão são respectivamente da ordem de 19,8 Mbps e de 38,6 Mbps [19],[27] e [28].

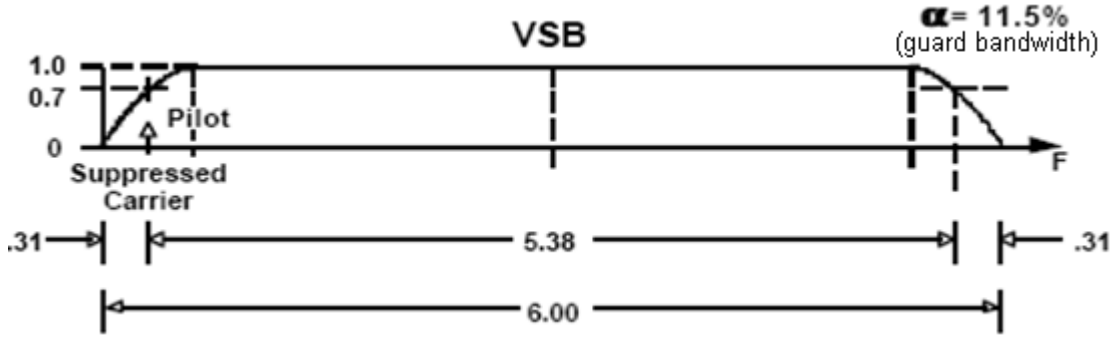

Figura 6 - Espectro de um sinal modulado com VSB

O VSB tem dois pontos negativos graves. O primeiro é que o sinal VSB preserva a má característica dos sistemas analógicos, que é a pouca robustez a situações de multipercurso comuns quando o sinal é transmitido em regiões densamente povoadas. Nesse caso, o resultado é a recepção de sinais defasados em fase que resultam no efeito conhecido como "fantasma na imagem". Receptores ATSC procuram utilizar técnicas de equalização adaptativa na tentativa de filtrar sinais defasados em fase e com amplitudes menores, entretanto o problema pode persistir caso sinais defasados tenham amplitudes próximas à dos sinais de interesse. O segundo problema do VSB é que os testes realizados, mostraram que o sistema tem uma péssima performance quanto à recepção desses sinais em terminais móveis [28]. Em 2004, o ATSC adotou o *Enhanced VSB* (E-VSB) um receptor 8-VSB com recursos adicionais para detecção e correção de erros [37] como método de modulação.

2.4.3 Coded Orthogonal Frequency Division Multiplexing (COFDM)

O COFDM é o esquema de modulação utilizado como base em transmissões terrestres tanto para o padrão europeu DVB, quanto pelo padrão japonês ISDB [12]. Uma sutil diferença no emprego dessa tecnologia de modulação no ISDB denomina essa modulação de *Band Segmented Transmission-OFDM* (BST-OFDM). O COFDM surgiu com a proposta resolver os problemas de multipercurso observados no VSB. O BST-OFDM surgiu como uma evolução do COFDM, permitindo que um mesmo sinal seja captado simultaneamente por receptores móveis e fixos. Baseado em [12],[29] e [30], alguns dos conceitos básicos do COFDM são:

- COFDM é o emprego da técnica OFDM, em que o sinal de entrada, que é um TS proveniente de um sistema MPEG-2, passa inicialmente por um processo de codificação com a inserção de bits para detecção e correção de erros, daí o nome *Coded* do COFDM.
- A técnica OFDM realiza a modulação de milhares de portadoras, dentro da faixa de 6,7 MHz ou 8 MHz, que transportam pequenos fluxos de bits do sinal de entrada.
- As portadoras são todas multiplexadas pela técnica FDM de modo que um símbolo é transmitido simultaneamente em várias portadoras. É possível visualizar que se trata de uma transmissão paralela em freqüências. Como os efeitos do multipercurso são sentidos diferentemente em cada freqüência, seus efeitos serão minimizados pois apenas partes do símbolo serão afetados.
- A Figura 7, baseada em [12] e [29], ilustra o espectro de um sinal OFDM. É possível visualizar que o alto número de portadoras torna o espectro praticamente retangular certificando que o sistema tem um bom aproveitamento da banda.

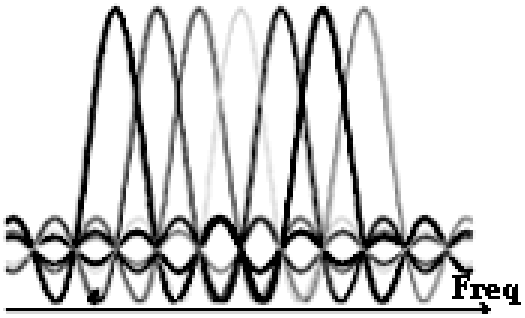

Figura 7 - Ilustração de um sinal modulado em COFDM

- As portadoras adjacentes são ortogonais, também conforme o nome da técnica, garantindo que os sinais não interfiram uns nos outros.
- Outro conceito importante é que, entre a transmissão de dois símbolos consecutivos deve existir um tempo de guarda maior do que o tempo necessário pelo circuito integrador utilizado no receptor para reconstituição do sinal original. Isso permite que o integrador não perceba os efeitos de alguns sinais defasados de algumas portadoras que por ventura tenham sofrido com o multipercurso.
- Existem diversos modos de funcionamento do COFDM dependentes do número de portadoras utilizadas. O DVB-T define o uso do modo 2K com 1705 portadoras de sinal sendo, 193 apenas para sinalização, sendo um modo indicado para transmissão móvel enquanto o modo 8K, indicado para transmissão sujeita a multipercurso, com 6817 portadoras sendo, 769 apenas para sinalização.
- O diferencial do BST-OFDM é partir da divisão do canal em 13 segmentos que recebem a modulação COFDM separadamente. Dependendo do tipo de receptor, levando em consideração aspectos como resolução de tela e mobilidade, poderá haver receptores com diferentes graus de resolução. Ou seja, se o gerador do sinal escolher preencher um canal com um sinal HDTV, esse sinal poderá ser recebido simultaneamente por receptores fixos ou móveis com diferentes resoluções.
- Segundo um relatório do FCC [28], o COFDM tem duas desvantagens principais em relação ao VSB. O COFDM tem uma capacidade de transmissão cerca de 5% menor do que o VSB, considerando um mesmo canal. A outra desvantagem é que o COFDM apresenta picos de potência maiores, de 2 a 3 dB maiores em relação ao VSB. Consequentemente os preços dos sistemas de transmissão COFDM são mais caros. Por outro lado o COFDM tem desempenho bastante superior em situações de multipercurso inclusive com uso de antenas internas, além de possibilitar a recepção móvel.

2.5 Canal de Retorno

O canal de dados [36], parte integrante do sistema de TVD, é um importante meio de transmissão para envio de informações que podem convertidas em conteúdos multimídia para serem apresentados ao usuário. Entretanto, o canal de dados é apenas unidirecional e distribuído simultaneamente para qualquer receptor do sinal e, nos dias atuais em que a Internet já difundiu a importância da troca de informações, fica claro que essa limitação do sistema seria um obstáculo que teria que ser vencido.

Para resolver essa questão foi inserido como parte integrante dos sistemas de TVD um canal de retorno que estabelece uma comunicação bidirecional entre o receptor e uma rede externa. Trata-se de um canal opcional e, apesar disso, é inegável a utilidade do canal de retorno no sistema já que permite que produtores de qualquer tipo de programa interativo obtenham importantes informações dos seus usuários e vice-versa.

O canal de retorno, nos sistemas conhecidos no mundo, sempre realiza a conexão entre o receptor de TVD e a Internet. Isso faz do canal de retorno um integrante ainda mais conveniente pois colabora na convergência de tecnologias, permitindo o aparelho de televisão receber conteúdos que normalmente exigiriam um computador pessoal. O que tem mudado nos últimos anos é a forma como a conexão à Internet tem sido realizada que varia de acordo com a capacidade tecnológica dos receptores. Os STBs disponíveis na atualidade, oferecem os seguintes métodos de conexão:

- Através de interfaces V.90 que estabeleciam a conexão TCP/IP discada através de linha telefônica convencional com velocidade máxima de 56 Kbps.
- Uma conexão TCP/IP permanente através de conectores RJ-45 adequados para conexão com modems ou *switches* externos.
- Uma conexão TCP/IP sem fio realizada através de interfaces BlueTooth ou WiFi.

Como em países como o Brasil, existem problemas de infra-estrutura que dificultam o acesso da população sobre conexões TCP/IP discadas ou a cabo [64], existem diversas propostas científicas que sugerem soluções para esta questão. Algumas bastante inovadoras como o uso de redes tipos *Power Line Communications* (PLC) [64] que utilizam a rede elétrica para acesso à rede TCP/IP.

2.6 Padrões de televisão digital

Sistemas de TVD são basicamente constituídos conforme a Figura 8, logo abaixo nesse texto e, baseada em [22] e [23], destaca os três subsistemas principais de um sistema de TVD:

- Subsistema para codificação e multiplexação de sinais de áudio, vídeo e dados (A/V&D) em TS no lado do transmissor e a recuperação, ou demultiplexação do sinal no lado do receptor. Para esse módulo todos os padrões de TVD da atualidade se baseiam no MPEG-2. Nesse caso, entre os padrões só há variação nos tipos de codificadores de áudio e vídeo que adotam.

- Subsistema para codificar e transmitir o sinal de TVD: Faz a codificação e modulação do TS em um sinal mais adequado para a transmissão no meio físico. Esse módulo é essencial, pois além de influenciar na questão da capacidade do canal, conforme a Equação 2, determina o quanto o sistema será robusto à degradação sofrida durante a transmissão. Esse módulo é considerado o verdadeiro qualificador dos padrões de TVD, uma vez que, cada um deles adota diferentes soluções que são determinantes para caracterização das suas vantagens e desvantagens.

- Subsistema decodificador do sinal de dados: Assim como os sinais de áudio e vídeo que são tratados no receptor para apresentação dos programas televisivos, os sinais de dados, que são transportados através de protocolos definidos no DSM-CC, contêm informações que são utilizadas no receptor para geração de conteúdos multimídia complementares, na maioria das vezes aplicações para interação com o usuário. Essa transformação dos dados em outros conteúdos é realizada por um *software* chamado de *middleware*.

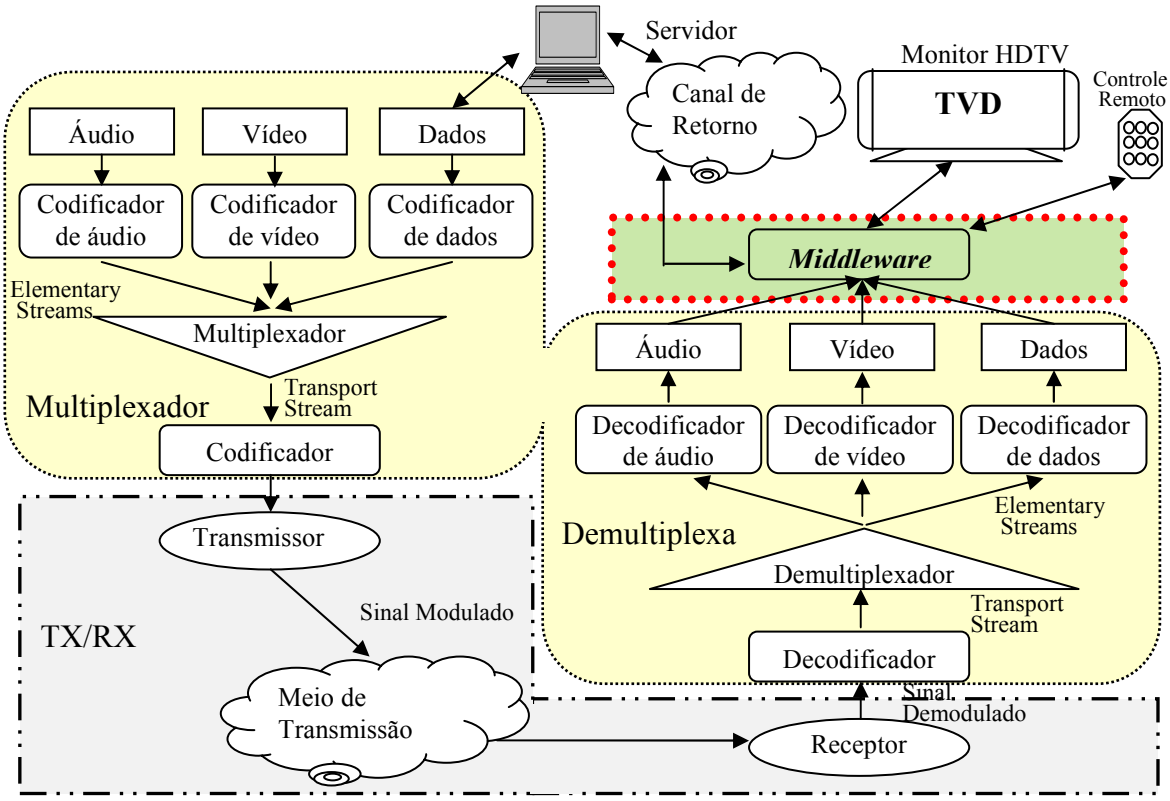

Figura 8 - Esquema de geração e transmissão de sinais de TVD

A Tabela 5, construída baseada em diversas referências [10], [11], [12], [13], [24], [27], e [32], sumariza as tecnologias empregadas nos principais padrões abertos de TVD existentes na atualidade, mais o padrão adotado no Brasil. Apesar de que as tecnologias envolvidas listadas na Tabela 5 já foram abordadas nesse capítulo, algumas observações ainda se fazem necessárias:

I – Todos os padrões podem ser adaptados para uso com canais de 6 MHz, 7 MHz ou 8 MHz. Nos EUA o ATSC continuou fazendo uso dos canais de 6 MHz adotados no seu sistema de TV analógico, assim como o ISDB-T no Japão e no Brasil. Os países da Europa também utilizaram seus antigos canais analógicos para transmissão de TVD, por isso podem ser encontradas as diversas configurações.

II – Todos os padrões estão em conformidade com o padrão MPEG-2, ou seja, poderiam optar por transmitir vídeos com diferentes resoluções e razões de aspecto, inclusive os padrões ITU-R BT.1125 descritos na Tabela 3. O modelo de negócios adotado no ATSC dos EUA é a de transmitir no canal de 6 MHz um sinal de HDTV (16:9). O modelo de negócios europeu, até o momento tem privilegiado a transmissão de até 6 sinais SDTV (4:3).

III – O BST-OFDM permite que os receptores recebam diferentes taxas de sinais dependendo da mobilidade e do formato da tela. Além disso, o modelo de negócios adotado no ISDB estabelece que cabe aos rádio-difusores definir a ocupação do canal da maneira que melhor lhes convier, alternando entre multiprogramação ou alta definição.

IV – Uma das contribuições do Brasil ao ISDB Internacional foi a adaptação do sistema para uso do codificador de vídeo definido na norma MPEG-4, também conhecido como H.264, que atinge taxas de compressão bem melhores do que obtidas utilizando MPEG-2 e descritas na Tabela 3. Trata-se de um codificador relativamente novo com custos de propriedade intelectual ainda bastante elevados. Por essa razão, até recentemente, o Japão adotou o uso de H.264 apenas para sinais LDTV com uso típico em receptores móveis, como aparelhos celulares.

V – Todos os padrões suportam múltiplos canais de áudio cuja codificação deve produzir conteúdos compatíveis com o MPEG-2 Sistemas. O ATSC nos EUA adotou o codificador americano proprietário DOLBY/AC3. Na Europa o DVB emprega padrões MPEG-2 *Backward Compatible* (MPEG-2 BC) que são compatíveis com o MPEG-1 áudio, como é o caso do MP3. O ISDB utiliza um codificador MPEG-2 AAC também de proprietário e classificado pelo MPEG-2 como MPEG-2 *Non Backward Compatible* (MPEG-2 NBC) que é considerado uma evolução em termos de desempenho e capacidade de compressão em relação ao MP3, fixando a mesma qualidade.

VI – Cada padrão define um componente do sistema que tem, entre outras atribuições, a capacidade de gerar aplicações de TVD. Essa questão será abordada no capítulo seguinte.

Características	**ATSC (Norte América)**	**DVB (Europa)**	**ISDB (Japão)**	**ISDB Internacional (Brasil)**
Taxa de Bits (Mbps)	19,8	De 4,98 até 31,97	De 4,98 até 31,97	De 4,98 até 31,97
Largura de faixa	6 MHz **(I)**	6, 7 ou 8 MHz	6 MHz **(I)**	6 MHz **(I)**
Codificador de vídeo	MPEG-2 vídeo **(II)** HDTV	MPEG-2 vídeo **(II)** (6 x SDTV)	MPEG-2 **(III;IV)** De LDTV à HDTV	MPEG-4 (H264) **(IV)** De LDTV à HDTV
Codificador de áudio	DOLBY/E-AC3**(V)**	MPEG-2 BC**(V)**	MPEG-2 AAC**(V)**	MPEG-2 AAC**(V)**
Multiplexação	MPEG-2 Sistemas	MPEG-2 Sistemas	MPEG-2 Sistemas	MPEG-2 Sistemas
Modulador para TX em Radiodifusão	8-VSB	COFDM	BST-OFDM	BST-OFDM
Suporte recepção móvel	Não	Sim, porém não simultaneamente no mesmo canal	Sim e, simultaneamente no mesmo canal	Sim e, simultaneamente no mesmo canal
Modulador TX em Cabo (C)	64-QAM	64-QAM	64-QAM	64-QAM
Modulador para TX via satélite (S)	QPSK	QPSK	QPSK	QPSK
Padrão de *Middleware*	DASE / OCAP / ACAP **(VI)**	MHP **(VI)**	ARIB **(VI)**	GINGA **(VI)**

Tabela 5 – Principais características dos padrões de TVD aberta

2.7 Considerações finais

O objetivo desse capítulo foi introduzir conceitos envolvidos no funcionamento dos sistemas de TVD aberta existentes na atualidade. Foram abordados aspectos básicos da visão que foram explorados na criação dos primeiros sistemas de televisão analógica através de um breve histórico. Partiu-se então para o entendimento dos desejos e desafios envolvidos na construção de sistemas de TVD e quais tecnologias foram empregredas. Tais conceitos são importantes na compreensão da Tabela 5 que resume o estado da arte da tecnologia de TVD.

Esse trabalho, entretanto, trata-se do desenvolvimento de uma plataforma para geração de aplicações interativas para TVD e para alcançar esse objetivo faz-se necessário obter um conhecimento bem mais aprofundado sobre o *middleware* que realiza a conversão das informações recebidas no STB através do canal de dados ou, do canal de retorno, em artefatos multimídia, inclusive aplicações interativas. Esse assunto foi pouco abordado no capítulo 2, pois será o assunto principal a ser tratado no capítulo 3.

3 Escolha do *Middleware* de TVD

Emmerich *et al* [34] define genericamente *middleware* como sendo uma camada de *software* existente entre as aplicações e o sistema operacional que proporciona aos programadores a abstração do funcionamento do *hardware* permitindo a realização de operações independente da solução do fabricante o que facilita consideravelmente o desenvolvimento de sistemas. O *middleware* pode ser empregado em diversos dispositivos de tecnologia inclusive como parte integrante de equipamentos STB quando sua denominação correta seria *middleware* para TVD. Como esse trabalho só aborda sistemas de TVD, o *middleware* para TVD será referenciado ao longo do texto simplesmente como *middleware*.

O *middleware* conforme utilizado em sistemas de TVD é um subsistema de *software* que pode ser acionado no STB para acessar as informações contidas nos pacotes de dados, coletados do canal de dados através do DSM-CC, e convertê-las em um conteúdo de multimídia, ou mais comumente em aplicações interativas [12]. Graças à atuação do *middleware*, desenvolvedores de *software* podem gerar aplicações abstraindo a disponibilidade de recursos de hardware de diferentes fabricantes de STB [17]. Tais aplicativos, gerados para os mais variados propósitos, podem ser interativos, ou seja, podem permitir que os expectadores deixem de ser meros expectadores e passem a utilizar o aparelho de televisão de forma mais ativa, solicitando serviços públicos e privados, realizando compras, escolhendo formas de visualização do programa, enfim influenciando o conteúdo ao qual gostam de assistir [6] e [32].

A importância dessa questão, fez com que cada um dos grandes consórcios de TV digital criasse seus próprios comitês para especificação do funcionamento de seu *middleware*. Dentre os padrões abertos de TVD, o primeiro *middleware* a ter sido definido foi o americano *DTV Application Software Environment* (DASE) [37], seguido pelo europeu *Multimedia Home Platform* (MHP) [35] e [39]. O *middleware* Japonês foi definido pela *Association of Radio Industries and Businesses* (ARIB) e inicialmente era constituído da parte conhecida por *Broadcast Markup Language* (BML) [12]. No modelo brasileiro será adotado um *middleware* desenvolvido por instituições de pesquisa nacionais batizado de GINGA [40].

Naturalmente que paralelamente, já existiam outros *middleware* comerciais em uso principalmente de sistemas de TV a cabo e via satélite. Um deles, o americano *Open Cable Platform* (OCAP) [37] que já tinha inclusive influenciado o desenvolvimento do MHP, por isso são similares, teve grande representatividade no mercado americano de STB's a ponto de provocar a divisão do mercado. Uma solução de consenso veio em 2005, quando o ATSC certificou o *middleware* chamado de *Advanced Common Application Platform* (ACAP) que mantém certas compatibilidades tanto com o DASE quanto com o OCAP [37].

3.1 Composição do *middleware*

O *middleware* de uma maneira geral é um subsistema de *software* que é executado diretamente pelo sistema operacional do STB. Ele é constituído de várias bibliotecas de funções, representadas na Figura 9. As bibliotecas por sua vez possuem interfaces, ou *Application Programming Interfaces* (APIs) bem definidas que, ao serem implementadas pelos fabricantes de hardware, estendem ao *middleware* suas funcionalidades.

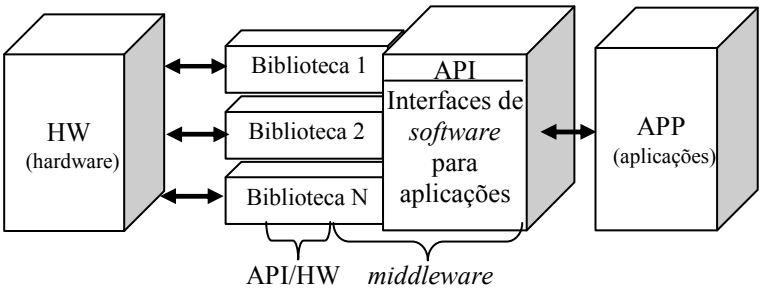

Figura 9 - Constituição genérica de um *middleware*

A partir da década de 1990 surgiram diversas bibliotecas de funções de *software*, entre proprietárias e de domínio público [12]. Tais bibliotecas surgiram com o propósito de proporcionar interoperabilidade entre as mais variadas aplicações de conteúdo áudio-visual e, pelo fato de terem sido bastante utilizadas, seus fundamentos acabaram sendo adotados, de forma reduzida ou mesmo estendida, pelos comitês de *middleware* como parte importante de suas especificações [12]. Abaixo estão relacionadas três das bibliotecas que mais influenciaram na especificação dos padrões abertos de *middleware* existentes na atualidade:

- *Digital Audio Visual Council* (DAVIC): Foi uma associação, formada por integrantes da indústria áudio-visual, criada com a finalidade de gerar uma especificação de interfaces para aplicações fim a fim. A API DAVIC aproveitou o uso das definições estabelecidas pela tecnologia *Multimedia and Hypermedia Information Coding Expert* (MHEG) para abstração das informações e incorporou o suporte a classes e interfaces Java. A DAVIC mantém sua importância por ter sido percussora para várias outras bibliotecas e também por ser constituinte do MHP [35].

- *Home Audio Video Interoperability* (HAVI): Essa biblioteca é resultado da iniciativa de vários fabricantes de equipamentos de áudio e vídeo mundiais [12]. Ela define uma série de interfaces para diferentes propósitos, mas com objetivo comum de garantir a interoperabilidade entre equipamentos de áudio e vídeo [46]. A especificação final data de 2001 e utiliza algumas das interfaces definidas no DAVIC além de outras próprias. A especificação HAVI, por si só, é independente de plataforma e de tecnologia que seja utilizada. Apesar do amplo escopo da HAVI, o MHP faz uso apenas da parte que gera a interface gráfica [45] com o usuário que inclui planos, textos, botões, listas, caixas de seleção. Esse uso confere à biblioteca uma enorme importância visto que estabelece o contato direto com usuário [35].

- JavaTV: Biblioteca desenvolvida pela SUN [38], naturalmente baseada na tecnologia Java e semelhante à idéia do J2ME (Java 2 *Micro Edition*) já adotada em várias soluções embarcadas como telefones celulares. A JavaTV é formada por uma série de API's que cobrem vários aspectos do desenvolvimento de *software* para TVD [35]. Ainda assim, apenas algumas API's da Java TV fazem parte da especificação MHP. As mais importantes APIs ou, pacotes de acordo com a nomeclatura Java, estão citadas na Tabela 6, baseada nas referências [35] e [38]. O funcionamento da biblioteca JavaTV, é dependente da instalação, na plataforma alvo, ou seja no STB, de um *software* interpretador de código Java, chamado de máquina virtual Java que normalmente é preparada para lidar com os escassos recursos do hardware do STB [38]. A aceitação

dessa tecnologia torna comum o fato dos fabricantes já disponibilizarem suas plataformas de hardware acompanhadas da máquina virtual Java apropriadamente compilada.

Nome do pacote	Principais atribuições
javax.tv.xlet	Contexto e gerenciador do ciclo de vida dos aplicativos Java para TVD interativa chamados de Xlets.
javax.tv.locator	Provê métodos e propriedades para acesso à informações provenientes do carrossel de dados e do carrossel de objetos além de outras informações contidas na DVB SI.
javax.tv.media	Fornece acesso aos componentes JMF para manipulação de conteúdo áudio-visual no monitor de TV.
javax.tv.net	Fornece acesso à datagramas IP fornecidos via canal de dados (Broadcast)
javax.tv.graphics	Trata-se de funções para construção de interfaces com componentes AWT.
javax.tv.util	Constituída por métodos para gerenciamento de eventos com temporização.
javax.tv.service	Acesso a outras informações disponíveis através da *Service Information* (SI). A SI contém além dos dados contidos na PSI (PMT+PAT) do TS do MPEG-2 mais tabelas específicas do DVB úteis para diversas aplicações de controle e gerenciamento da TVD.
javax.tv.service.guide	Fornece métodos específicos para acesso ao conteúdo atual e agendado do TS. Muito útil para aplicações tipo EPG.
javax.tv.service.navigation	Métodos para acesso mais detalhado à informações da SI e busca de serviços de acordo com critério de seleção definido pelo utilizador.
javax.tv.service.transport	Métodos baseados em consultas a SI relativo ao transporte do fluxo de TS.
javax.tv.service.selection	Métodos para seleção de conteúdos disponibilizados através do fluxo de TS.

Tabela 6 – Principais componentes da Java TV

Para se ter uma idéia da diversidade das soluções empregadas em middleware para TV Digital, o MHP define a utilização de várias API's incluindo as bibliotecas Java TV citadas na Tabela 6, algumas interfaces da biblioteca DAVIC, os componentes gráficos da biblioteca HAVI, além de outras interfaces complementares definidas pelo próprio MHP. Esse conjunto de bibliotecas constitui a especificação chamada *Globally Executable MHP* (GEM). O objetivo básico da especificação GEM é listar detalhadamente o comportamento de todas as bibliotecas e interfaces que constituem o MHP e permitir que diferentes implementações, mesmo que particulares, tenham funcionamento equivalente ao MHP [24] e [35].

3.2 Tipos de middleware:

Dependendo da forma como o *middleware* oferece suas funcionalidades para geração de aplicativos interativos ele pode ser classificado de duas maneiras [12] e [35]:

- Procedural: Quando as funcionalidades do *middleware* são oferecidas através da exposição das API's que o compõe. Nesse caso, para geração de um aplicativo, faz-se

necessário o desenvolvimento de um projeto de *software* no qual é fundamental a experiência dos profissionais envolvidos com o funcionamento dessas API's bem como o conhecimento de como realizar as chamadas de modo coerente a fim de se obter o programa funcional. A grande vantagem desse tipo de abordagem é a possibilidade de desenvolver programas interativos de grande complexidade conforme as possibilidades das bibliotecas e a criatividade dos desenvolvedores. A especificação GEM é baseada em utilização de *middleware* procedural.

- Declarativo: Quando as funcionalidades do *middleware* são oferecidas através da interpretação de informações contidas em arquivos, normalmente do tipo *Extensible Markup Language* (XML), seguindo uma formatação pré-definida. Desenvolver aplicações declarativas significa conhecer bem o formato e especificar valores para parâmetros, o que a princípio, é mais simples do que ter conhecimento em programação, mas por outro lado, bem menos flexível quanto às funcionalidades que se pode atingir com aplicativos. Entretanto, existem linguagens declarativas bastante complexas, que possuem diversas capacidades como, a execução de *scripts* e outros artefatos de *software* o que as aproxima nesse aspecto das linguagens procedurais. Por exemplo, o DVB-HTML e DASE, mencionados na Figura 10 baseada em [45], suportam *scripts* ECMA enquanto o GINGA-NCL [40] que é a parte declarativa do *middleware* brasileiro, dá suporte tanto a *scripts* ECMA quanto LUA [40].

A existência de desenvolvedores e geradores de conteúdos que defendem os dois tipos de abordagens tornou comum o fato dos padrões de *middleware* suportarem ambos. A Figura 10 mostra a composição dos *middleware* e ainda, a compatibilidade entre eles. *Middleware* procedurais estão representados em blocos preenchidos em cinza, enquanto *middleware* declarativos estão representados por blocos tracejados. As setas mostram a compatibilidade e, através disso é possível verificar que em todos os principais padrões, a parte procedural é compatível com o GEM, especificado através da norma ITU J.202. No caso do GINGA [40], além da porção procedural GINGA-J [41], compatível com o GEM, existe uma porção declarativa chamada de GINGA-*Nested Context Language* (GINGA-NCL) [40].

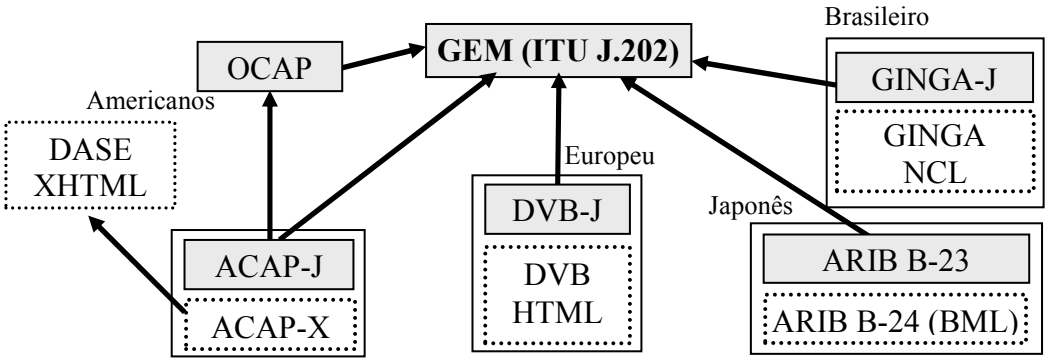

Figura 10 - Compatibilidade entre padrões de *middleware*

Normalmente *middleware* declarativos, definidos por um padrão, não são compatíveis uns com os outros, por exemplo, o BML japonês é uma versão ampliada do XHTML 1.1 [45] não compatível com DVB-HTML ou DASE. Da mesma forma o DASE não é compatível com DVB-HTML. O GINGA-NCL é uma exceção a essa regra [42], pois, tem o mérito de manter certas

compatibilidades com conteúdos baseados em HTML desenvolvidos para outras plataformas, como o DVB-HTML.

Quanto aos *middleware* procedurais, nos últimos anos tem havido uma tendência de convergência em direção a plataformas compatíveis com a especificação GEM que se destaca pela generalidade de suas soluções, e na aceitação acadêmica, a julgar pela grande quantidade de trabalhos científicos [35] sobre o assunto. Para constatar essa tendência basta verificar que além do americano ACAP, também o padrão japonês através do seu componente ARIB B-23 e o padrão brasileiro através do GINGA-J [40] e [41], são compatíveis com o GEM. Essa compatibilidade justifica o interesse sobre a especificação GEM e, sendo assim, será constantemente abordada nesse trabalho.

3.3 MHP/GEM

Já foi antecipado em vários pontos nesse texto, que o MHP é um padrão de *middleware* que reúne um conjunto de bibliotecas que foi definido pelo consórcio europeu de TVD DVB. Em termos de número de países, o MHP é de longe o que alcança maior abrangência no planeta sendo adotado por mais de 50 países [39].

A abrangência do padrão é muito maior considerando que o núcleo das especificações do MHP, sob o nome de GEM, é suportado de forma compatível por vários outros padrões de TVD, entre abertos (OCAP, ARIB-B23, GINGA-J) e proprietários (Cable Labs). Muito embora o conceito do GEM tenha surgido depois do MHP, é possível dizer que o MHP é uma possível implementação do GEM que é de fato o *middleware* definido pelo DVB e está devidamente registrado através da norma ITU-J202 [39] e [45]. A Figura 11, proveniente de [45], fornece a visão das bibliotecas que compõem a especificação GEM.

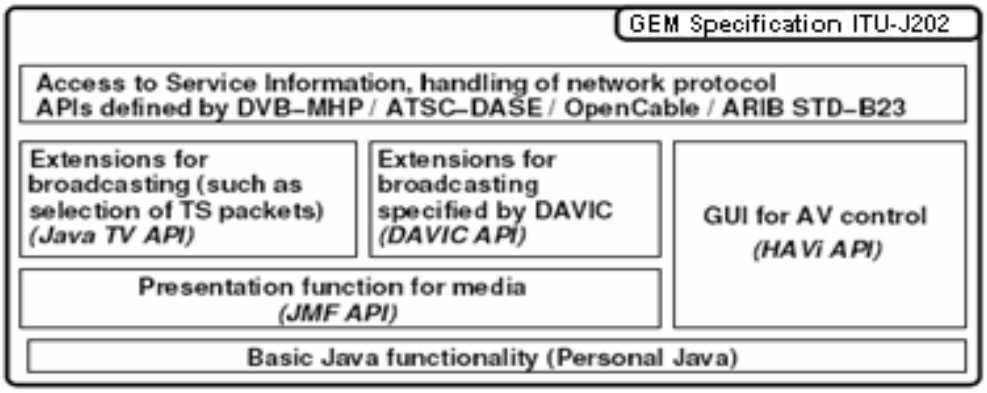

Figura 11 - Composição da especificação GEM ITU-J202

No que diz respeito ao MHP foram sendo definidos perfis de acordo com as funcionalidades que são disponibilizadas pelo *middleware* e isso é importante, pois impõe limite aos aplicativos que podem ser executados. Os perfis mais modernos sempre acumulavam as funcionalidades dos perfis anteriores. De acordo com as referências [24], [39] e [45], existem três perfis definidos para o MHP:

- *Enhanced Profile:* Foi o primeiro perfil definido no MHP. Não dá suporte a canal de retorno e, portanto para qualquer conexão IP, com capacidade para execução de

aplicações procedurais recebidas via canal de dados bem como as residentes no próprio sistema de arquivos do receptor.

- *Interactive Profile:* Dá suporte a canal de retorno, com conexão IP tanto com interfaces Ethernet quanto através de interfaces V90 (modem para conexão discada com limite de 56 Kbps). Com isso tornou-se possível enviar informações para servidores remotos, além de permitir que aplicações sejam carregadas também através do canal de retorno para execução no sistema.

- *Internet Access Profile:* O mais avançado dos perfis suporta apresentação de conteúdos da Internet e também aplicações declarativas DVB-HTML que inclui um programa na plataforma para apresentação de aplicações desse tipo e que inclui suporte à execução de scripts e apresentação de conteúdo multimídia.

O *Enhanced Profile* e o *Interactive Profile* foram definidos logo na versão 1.0 do MHP enquanto que o *Internet Access Profile* foi definido na versão 1.1, quando já existiam receptores com maiores recursos de processamento e memória tão necessários nesse segundo momento.

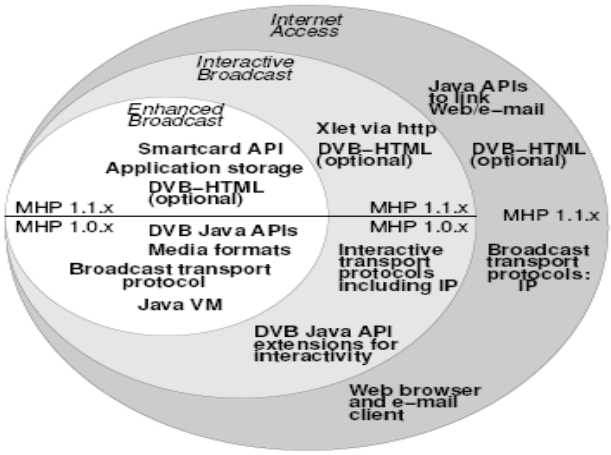

Figura 12 - Evolução do MHP

3.3.1 XLET

Apesar da diversidade de bibliotecas que compõem a especificação GEM a biblioteca JavaTV é sem dúvida a mais importante integrante do sistema. Além de consolidar o uso da tecnologia Java, vem da JavaTV a especificação da interface principal que todo aplicativo GEM obrigatoriamente deve possuir. Trata-se da interface *XLet*, termo esse que também é comumente utilizado para se referir à aplicações GEM.

Na tecnologia Java *XLet* refere-se à aplicações para TVD assim como applets referem-se à aplicações para execução em navegadores de Internet e *MIDLets* à aplicações para execução em aparelhos celulares. Essas analogias são bastante pertinentes até porque, vários dos conceitos utilizados em *Applets* e *MidLets,* são utilizados em *XLets*. Por exemplo, a interface *XLet* não possui um método "*Main*" e sim um método "*initXlet*" que é executado por um gerenciador de

aplicações, que é parte integrante do *middleware* presente no STB. Em contrapartida ao "*initXLet*" o *XLet* fornece o método "*destroyXlet*" além dos métodos "*startXlet*" e "*pauseXlet*". O gerenciador de recursos utiliza esses quatro métodos para controlar o ciclo de vida dos *XLets* caracterizando claramente quatro estados Carregado, Pausado, Ativo e Destruído que são atingidos dependendo da ordem de execução dos métodos. O mapa de transição dos possíveis estados está ilustrado na Figura 13 [24] sempre presente em qualquer referência sobre *XLets*.

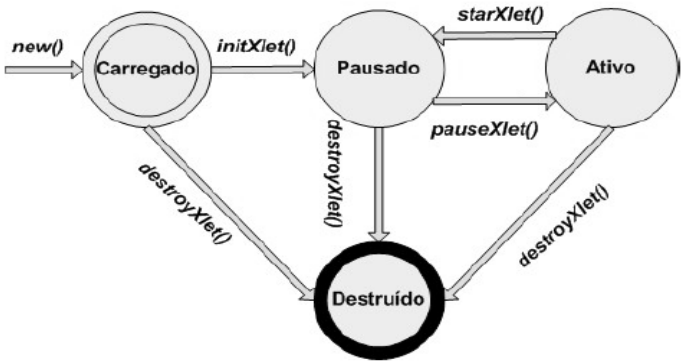

Figura 13 – Ciclo de vida de *XLets*

3.4 GINGA

O Ginga [42] é um *middleware* desenvolvido a partir da união de duas iniciativas bem sucedidas respectivamente da Universidade Federal da Paraíba, através do projeto conhecido como FlexTV [41], e da Pontifícia Universidade Católica do Rio de Janeiro, através do projeto chamado MAESTRO [42]. O FlexTV acabou sendo denominado de GINGA-J [41] pelo seu vínculo com a tecnologia Java e especialmente pela compatibilidade com o GEM. O MAESTRO passou a ser chamado de GINGA-NCL, pois se trata de um *middleware* declarativo cuja interpretação se baseia nas regras definidas numa linguagem especialmente desenvolvida para apresentação e sincronização de conteúdos interativos chamada de *Nested Context Language* (NCL) [40]. A Figura 10 ilustra o GINGA sendo constituído de uma parte declarativa, o GINGA-NCL e uma parte procedural GINGA-J independentes entre si, mas, que podem interagir de modo que, o GINGA-J possa iniciar um programa GINGA-NCL e vice-versa, fornecendo ao sistema uma flexibilidade incomum em relação a outras especificações de *middleware* [44].

No que diz respeito ao GINGA-J basta dizer que é a parte do *middleware* que dá suporte às funcionalidades definidas no GEM e mais tarde também através das normas do ITU-T J.200, J.201 e J.202 [44]. Portanto trata-se da consolidação das definições da parte procedural do MHP e isso garante que qualquer aplicação GEM compatível poderá ser executada pelo GINGA-J.

Já o GINGA-NCL é um *middleware* declarativo completamente inovador cujos princípios básicos são [42]:

- Criação de um poderoso modelo conceitual, chamado NCM *Nested Context Model,* voltado para representação e montagem de documentos multimídia relativos a aplicativos de TV digital interativa. A prinicpal característica do NCM é estabelecer relações temporais entre os diversos objetos multimídia declarados, realizando o que denominam de sincronismo entre mídias.

- Criação da linguagem NCL, que é uma linguagem declarativa, baseada na leitura de um arquivo XML e que fornece a concretização dos conceitos apresentados no modelo NCM.
- O GINGA-NCL possui um módulo central chamado de formatador, ou ainda "*formatter*", que faz a interpretação do arquivo escrito em NCL para geração das aplicações interativas.
- Apesar de ser auto-suficiente no que diz respeito à capacidade de gerar conteúdo multimídia, o NCL dá suporte à apresentação de vários outros tipos de documentos como HTML e também para utilização de scripts procedurais como, por exemplo, ECMA e LUA oferecendo uma grande flexibilidade para o sistema.
- Disponibilização de um aplicativo gráfico, chamado de "*Composer*", multiplataforma e que permite ao usuário inserir graficamente, uma área de desenho que representa a tela do televisor, elementos multimídia para geração de uma aplicação interativa. Ele permite exportar e importar arquivos NCL permitindo assim o desenvolvimento de aplicativos sem necessidade de grandes conhecimentos sobre detalhes da linguagem.

É importante, que nesse momento de definição de qual *middleware* será utilizado no desenvolvimento desse projeto, listar alguns aspectos que poderiam dificultar a utilização do GINGA-NCL:

- Um aplicativo puramente GINGA-NCL não será compatível com nenhum outro padrão de *middleware*.
- A linguagem NCL é uma linguagem nova, e só recentemente foram disponibilizados documentos que totalmente descrevem sua característica. Sendo assim, ainda são muito poucas as pessoas que dominam suas potencialidades. Os artefatos funcionais como o *software* "*Composer*" só foram disponibilizados após o licenciamento dessa tecnologia sob licença GPL o que aconteceu aproximadamente em Junho de 2007.
- A linguagem NCL é, sem dúvida nenhuma, bastante poderosa e impõe uma série de conceitos complexos para sua utilização e que exigem algum tempo de aprendizado, mesmo considerando a utilização do "*Composer*", que pode minimizar esse tempo.
- Ainda devido ao pioneirismo da tecnologia, apesar de que módulos do NCL tenham sido recentemente disponibilizados sob licença GPL ainda há poucos relatos concretos de seu funcionamento em plataformas reais. Por esse motivo, o sistema brasileiro de televisão digital foi lançado em dezembro de 2007 sem suporte a interatividade. Passado esse momento inicial, ainda é uma incógnita se a obrigatoriedade imposta pelo governo sobrevalecerá sobre os interesses econômicos dos grandes fabricantes que já dispõem de plataformas GEM compatíveis procedentes de implementações européias e americanas.
- Tecnicamente falando, é fácil especular que, sendo o GINGA um sistema poderoso, flexível e que apresenta tantas inovações em relação aos demais *middleware*, ele exige da plataforma do STB alta capacidade de processamento e memória para ser executado o que consequentemente resulta em aumento de custo do sistema para o usuário final. A constatação dessa afirmação só será possível com a chegada dos primeiros STB que disponibilizarem implementações comerciais do GINGA-NCL, o que ao que tudo indica, só deve acontecer ao final do primeiro semestre de 2008.

3.5 Considerações finais

Esse capítulo abordou o *middleware* que é o componente de maior relevância no sistema de TVD para esse trabalho. Também mostrou as bibliotecas com as quais, indepentendemente do padrão, os *middleware* são constituídos. Também abordou que os *middleware* podem ser classificados em procedurais ou declarativos.

Dentre os *middleware* procedurais o destaque fica para a especificação MHP/GEM que se tornou um padrão aceito mundialmente. Foi realizada uma breve abordagem sobre o Xlet que é uma interface integrante da especificação GEM e também o ponto de partida para o desenvolvimento de qualquer software para ser executado no sistema. Dentre os *middleware* declarativos, o GINGA-NCL, desenvolvido no Brasil, merece importante destaque pela sua flexibilidade além da inovadora abordagem de sincronização de mídias. Faz-se então uma exploração das características desse *middleware* de maneira conceitual, baseadas em artigos publicados pelo grupo desenvolvedor. As considerações finais sobre o GINGA-NCL abordam algumas dificuldades que poderiam ser encontradas para utilização desse *middleware* como base para o desenvolvimento da plataforma que se deseja obter.

A conclusão final demonstrada por esse capítulo é que a especificação GEM oferece maior possibilidade compatibilização com outros sistemas, além de oferecer os recursos de *software* necessários para o desenvolvimento da plataforma que se deseja obter com esse trabalho. O próximo capítulo introduzirá alguns conceitos relativos a aplicações de TVD interativa e, em seguida fará a apresentação dos principais tipos de aplicações que se pode obter utilizando a tecnologia disponível para sistemas de TVD. O objetivo será identificar e caracterizar, dentre esses vários tipos, qual pode ser mais bem empregado em fins educativos.

4 Definição do Tipo de Aplicação

Esse capítulo caracterizará os tipos de aplicações que podem ser desenvolvidas em sistemas de TVD. A ênfase será dada ao tipo *T-Learning* que se refere ao emprego da TVD em fins educativos. Inicia-se pela apresentação do conceito de classificação de tipos de interatividade em TVD interativa, seguido para ilustração dos principais tipos de aplicações, ou categorias. Fica claro nesse ponto, a relevância do *T-Learning* para esse trabalho que parte então para a definição e caracterização de uma modalidade de aplicação. Essa caracterização será importante para definição da plataforma a ser desenvolvida.

4.1 Tipos de interatividade:

No que diz respeito à interatividade que uma aplicação de TVD pode proporcionar aos usuários, é importante caracterizar dois tipos básicos, apesar da possibilidade de subdivisão no conceito conforme apresentado em [45] e [46]:

- A interatividade local, ou básica, é aquela que não depende de envio ou recebimento de informações externas tendo tão somente o controle remoto do STB como forma de entrada. Esse tipo de interatividade não utiliza canal de retorno e é bem representado por alguns jogos e aplicações meramente informativas como um *Electronic Program Guide,* ou simplesmente EPG, apresentado logo abaixo no tópico 4.2.1. Segundo PENG [46], ainda se enquadram nesse grupo aplicações que fazem seleção de legendas ou seleção do ângulo da imagem.

- A interatividade remota é aquela em que se faz uso de um canal de retorno para envio de informações que são fornecidas pelo usuário. Essas informações são processadas por um servidor externo que poderá eventualmente modificar o conteúdo de algum arquivo de entrada do próprio aplicativo, configurando a interatividade remota em dois sentidos [46]. No Brasil ainda há expectativa de que, nos primeiros anos de implantação, sistemas de TVD não contem com a infra-estrutura necessária para utilização de canal de retorno, principalmente nas camadas mais pobres da população, embora conforme mencionado em 2.5, existem várias propostas acadêmicas que se proponham a resolver essa questão.

4.2 Categorias de aplicações de TVD interativa

Levando em consideração critérios quanto à usabilidade, é possível classificar as aplicações de TVD interativa em um número crescente de categorias. Há aplicações para realizar as mais diversas tarefas relacionadas a entretenimento, utilidade pública, serviços e educação. Dentre essas diversas categorias, algumas são básicas e mais populares. As principais categorias de aplicações de TVD são constantemente abordadas nas principais referencias, tais como [12], [24], [45], [46], [47], [53], e estão listadas a seguir.

4.2.1 Electronic Program Guide (EPG)

Esse é o tipo de aplicação mais comum, e com presença quase que obrigatória em qualquer STB, pois permite ao usuário visualizar e navegar, através de tabelas sobre a programação disponível entre todos os canais digitais recebidos. Isso é possível, pois esse tipo de aplicação acessa o conteúdo das tabelas PAT e PMT, provenientes do sinal MPEG-2 Sistemas, já abordado no tópico 2.3.1. Elas fornecem informações precisas sobre o conteúdo do TS em transmissão além de informações sobre programação futura.

Aplicações de melhor qualidade tendem a fornecer funções para busca seletiva dos programas desejados. A interatividade fica mais evidente, pois geralmente esse tipo de aplicação permite a programação no próprio STB para seleção de programas em horários determinados. Por isso, normalmente, esse tipo de aplicação já é parte do próprio sistema operacional do STB tendo o acesso necessário para armazenamento de informações no sistema de arquivos do aparelho. Não são raras as vezes em que, juntamente com o EPG, está disponível serviço para gravação de programas cujo termo em inglês é *Personal Video Recorder* (PVR) [24]. A Figura 14 ilustra a tela principal de duas aplicações de EPG, sendo a primeira proveniente do trabalho de WAISMAN [53] e, a segunda, a ilustração de uma aplicação comercial utilizada na Europa. Normalmente, aplicações EPG utilizam apenas interatividade local. É pouca a relação entre esse tipo de aplicação e processos educativos.

Figura 14 - Ilustrações de aplicações tipo EPG

4.2.2 T-Commerce

A tradução do termo *Televisual-Commerce* ou, simplesmente *T-Commerce* se refere a uma modalidade de aplicações interativas em que se explora a comercialização de produtos e serviços através da TV [47]. Na verdade, diferentemente da propaganda tradicional, aplicações desse tipo podem ser disponibilizadas durante a exibição de algum programa devido à sua relação direta com o produto induzindo a compra revelando uma fonte potencial de receita financeira para as geradoras de conteúdo. Por esse motivo, a despeito de qualquer outra inovação que a TVD venha a trazer para o usuário, fica fácil de perceber que esse tipo de aplicação é uma das principais justificativas para o interesse dos provedores de conteúdo pela implantação da tecnologia.

As aplicações com maior apelo comercial imprescindivelmente requerem a utilização de interatividade remota dada a necessidade de envio de informações do pedido do usuário. Mas, se

a propaganda for meramente informativa e, baseada em seleção da informação pelo usuário, pode também ser classificada como aplicação de interatividade local. Esse tipo de aplicação não apresenta utilidade para fins educativos.

4.2.3 T-Banking

Assim como a categoria *T-Commerce*, a categoria *T-Banking* também tem forte apelo comercial, sendo voltada para serviços bancários como acesso a saldos, extratos e realização de movimentações financeiras e que exigem uma série de funcionalidades ligadas à segurança cuja relevância e complexidade justificariam todo um estudo. Esse tipo de aplicação requer interatividade remota completa e não contribui de nenhuma maneira para educação via TVD.

4.2.4 T-Health

Trata-se de outra categoria de grande apelo social, pois se refere a aplicações voltadas para colaboração com área da saúde. Podem simplesmente oferecer informações sobre saúde quanto serem aplicações em que o médico interage diretamente com o paciente, coletando informações que levem a diagnósticos. Por essa razão, *T-Health* está relacionado a processos educativos. A Figura 15 [45], ilustra uma aplicação voltada para apresentação de instruções em primeiros socorros, configurando um exemplo de aplicação *T-Health*. Também existem aplicações *T-Health* tanto com interatividade local quanto remota.

Figura 15 - Ilustração de aplicação voltada para saúde

4.2.5 T-Government

O termo *T-Government*, é utilizado para aplicações com oferecimento de serviços governamentais [53]. Existem grandes perspectivas sobre essa modalidade que tem como objetivo tornar mais ágil, mais democrático e menos burocrático o acesso a informações públicas. São muitas as possibilidades e o Brasil que é um dos países com mais experiência em fornecimento de serviços públicos pela Internet certamente terá muito a contribuir nessa modalidade.

Essa categoria pode tanto apresentar aplicações que utilizam interatividade local, quando oferecem apenas informações de forma seletiva, quanto apresentar aplicações que exijam a comunicação com servidores externos. Levando em consideração que essa categoria relaciona-se ao oferecimento de serviços públicos, esse tipo de aplicação só faz sentido em processos educativos se fornecer algum tipo de serviço administrativo para instituições de ensino.

4.2.6 T-Learning

T-Learning é o termo utilizado para caracterizar qualquer tipo de aplicação para TVD voltada para fornecer serviços interativos educacionais [51]. DAMÁSIO *et al* [49] complementa afirmando que *T-Learning* é a convergência entre as tecnologias de TVD e *E-Learning* que, por sua vez, se refere ao uso da tecnologia computacional para prover treinamentos ou outras atividades educacionais.

É grande o interesse acadêmico a esse tipo de aplicação devido ao seu forte apelo social já que aplica o uso da tecnologia para nobres fins como educar, formar opinião e desse modo contribuir para uma sociedade mais justa em termos de oportunidades [48]. BATES [50] justifica que a importancia da televisão como instrumento educativo está na popularidade do aparelho, que está presente na maioria absoluta dos lares tanto de países desenvolvidos como em desenvolvimento como é o caso do Brasil [1] e [2]. Além disso, programas de TV bem elaborados possibilitam a demonstração de qualquer tipo de conceito uma vez que pode se valer de poderosos recursos gráficos que prendem a atenção das pessoas, facilitando o aprendizado. WALLDÉN *et al* [52] também aborda a questão e fornece uma classificação para processos educativos de acordo com o contexto e o propósito:

- Processos educativos formais: Aqueles fornecidos por instituições de ensino e treinamento formais e que fornecem diplomas e certificados reconhecidos pelos órgãos oficiais do país.

- Processos educativos não formais: Referem-se aos processos educativos fornecidos por instituições de ensino e treinamento, mas que não fornecem tipicamente diplomas ou certificados para o conhecimento difundido. Normalmente é obtido através de atividades em grupo de organizações da sociedade civil.

- Processos educativos informais: Referem-se aos processos educativos em que o indivíduo adquire valores, habilidades ou conhecimentos gerais a partir de experiências diárias em casa, no trabalho ou em qualquer ambiente propício.

- Processos educativos acidentais: São aqueles em que o indivíduo adquire conhecimentos involuntariamente ou sem a percepção do processo educativo em andamento. Isso pode acontecer durante a realização de processos recreativos como participação em jogos, assistir a certos filmes ou programas de competições. Este interessante caso é bastante difundido na literatura sob o termo "*Edutainment*" em referência à junção das palavras *Education* e *Entertainment*.

A classificação fornecida por WALLDÉN *et al* [52] é especialmente importante para esse trabalho pois fornece ao longo de seu texto algumas abordagens para fornecimento de aplicações de TVD interativa dependendo da faixa etária e escolaridade do aprendiz, ou seja do usuário de aplicações TVD com propósitos educativos. Entre os possíveis tipos de aplicações está a modalidade chamada de *Quiz* e que, pela sua relevância face aos objetivos desse trabalho será abordada no tópico seguinte.

4.3 Definições e justificativas acerca do *Quiz*

Quiz é um termo da língua inglesa que se refere a uma prova, exame ou teste em que o usuário responde de acordo com seus conhecimentos específicos ou opiniões pessoais às perguntas que sejam apresentadas. Trata-se de um termo bastante difundido através da Internet onde existem milhares de aplicações desse tipo em funcionamento o que facilita o entendimento de suas características.

Na prática, aplicações interativas do tipo *Quiz* são aquelas que apresentam ao usuário uma ou mais perguntas, acompanhadas de múltiplas alternativas e tendo uma ou várias respostas corretas que podem ser escolhidas pelo usuário. A resposta do usuário para cada pergunta é armazenada e, com a finalização do processo, devido à ocorrência de um evento disparado pelo usuário, o programa que controla o *Quiz* apresenta as respostas ao usuário. Havendo disponibilidade do canal de retorno as respostas são enviadas para algum outro servidor que cuida de processar e armazenar os resultados fornecidos.

Com esse princípio, de coletar opinião ou conhecimento do usuário, aplicações tipo *Quiz* podem fornecer importantes informações para as provedoras de conteúdo, sejam em programas de entretenimento e concursos, mas também para direcionamento de programação de acordo com o desejo do público.

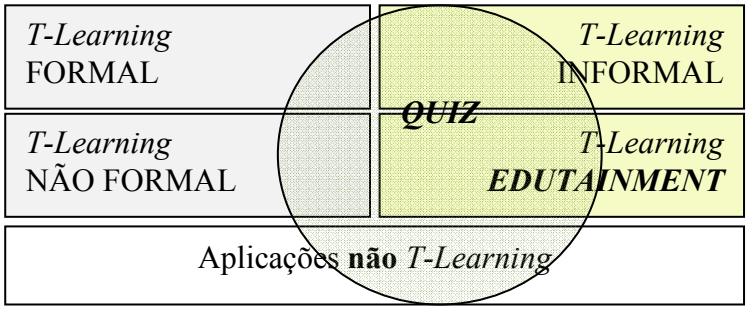

Figura 16 – Universo de aplicações tipo Quiz aplicadas em TVD

Sob o ponto de vista educacional, o tópico anterior já havia introduzido que, aplicações tipo *Quiz* constituem importantes ferramentas utilizadas em *T-Learning*. A Figura 16 é baseada na compreensão dos textos de WALLDÉN *et al* [52] e também de WAISMAN [53], e tem a intenção de ilustrar os seguintes aspectos:

- No universo de aplicações conhecidas do tipo *Quiz*, representado pelo círculo, existem várias aplicações voltadas a *T-Learning*.
- *Quizzes* também são utilizados para fins de simples entretenimento, ou seja, não *T-Learning*. Por exempo, a utilização do *Quiz* para que o usuário opine sobre qual sua opinião sobre o desfecho de um programa. O *Quiz* nesse caso obrigatoriamente requer a disponibilidade do canal de retorno configurando uma interatividade remota.
- O círculo tende para o lado das aplicações *T-Learning* informais e *edutainment*. Isso significa que, estatisticamente, tem-se conhecimento de um número maior aplicações de *Quiz* relacionadas a essas modalidades de *T-Learning*. Processos formais ou não formais de *T-Learning* tendem a usar a TVD interativa de outras formas, como por exemplo, para apresentação de conteúdos através de vídeos. Ainda assim, existem casos sobre aplicações tipo *Quiz* utilizadas também em processos formais e não formais.

Seguindo no intuito de justificar a importância de *Quizzes* em processos educativos existe a afirmação apresentada por O'KEEFFE *at al* [54], de que "*Quizzes* são naturalmente parceiras simbióticas da educação e, portanto, igualmente importantes para completar o processo educativo". Outra constatação vem de WAISMAN [53] que, em sua dissertação, descreve que *Quizzes* para TVD podem ser utilizados como ferramentas, a serviço dos professores provendo resultados diferentes dependendo do momento em que for empregado:

- Se o *Quiz* for utilizado antes da apresentação da teoria aos alunos (pré-didático): O resultado obtido servirá como diagnóstico da situação dos alunos.
- Se o *Quiz* for utilizado após a apresentação da teoria aos alunos (pós-didático) servirá como avaliação formal sobre os alunos.

Quanto ao tipo de interatividade o Quiz pode ser empregado nas seguintes situações:

- *Quiz* aplicado em *edutainment*: Pode ser utilizado tanto com interatividade local quanto remota.
- *Quiz* aplicado em T-*Learning* formal e informal: Requer interatividade remota.
- *Quiz* aplicado em programas não T-*Learning*: Requer interatividade remota.

A Figura 17, ilustra aplicações de TVD interativa tipo *Quiz* desenvolvidas pelo canal de televisão BBC.

Figura 17 - Ilustrações de aplicações tipo *Quiz*

4.4 Caracterização de aplicações tipo Quiz

A partir da constatação de que *Quizzes* atendem aos objetivos desse trabalho de desenvolver uma plataforma para geração de aplicações de TVD voltadas para educação torna-se necessário realizar o levantamento das principais características que esse tipo de aplicação deve suportar. Isso será imprescindível para a definição dos requisitos da plataforma no capítulo posterior. Para tanto, a primeira ação a ser tomada é realizar um levantamento sobre trabalhos científicos sobre o assunto. Foram consideradas informações retiradas das referências [53], [54], [55] e principalmente do sistema *Multi-user Object Oriented Distributed Learning Environment*, mais conhecido como Moodle, conforme referência indicada em [56].

Resumidamente, o Moodle trata-se de todo um sistema, composto de vários módulos para o gerenciamento de cursos de educação à distância através da Internet. Entre esses módulos, existe um que aborda especificamente *Quiz*, confirmando mais uma vez a importância desse tipo de aplicação como instrumento de avaliação educativa. Outro ponto comum desse módulo do Moodle com esse trabalho é que o seu princípio de funcionamento baseia-se também na construção da aplicação, em forma de páginas da WWW, a partir de um arquivo de configuração contribuindo para justificativa de que esse é um bom método para garantir a fácil reusabilidade e flexibilidade do sistema. O formato do arquivo de configuração do Moodle é chamado de *General Import Format Template* ou simplesmente GIFT. Esse formato contribuiu principalmente para o levantamento das características mais importantes de serem suportadas. Os tópicos que se seguem relacionam as características julgadas essenciais de acordo com as pesquisas realizadas.

4.4.1 Premissas de um *Quiz*

I. Um *Quiz* é constituído basicamente de uma ou mais perguntas. Cada pergunta oferece ao usuário uma ou mais alternativas, o que academicamente configura um questionário chamado de múltipla escolha. Normalmente não há limite quanto ao número de perguntas. Em se tratando de aplicações para TVD interativa as limitações serão impostas quanto ao consumo de memória consumida pela aplicação em funcionamento.

II. Cada pergunta pode conter um número independente de alternativas, ou seja, uma pergunta pode oferecer 3 alternativas e outra 5 alternativas. O limite no número de alternativas é principalmente visual, pois depende do tamanho da tela disponível, do posicionamento inicial da primeira alternativa, da fonte utilizada. Independentemente disso é suficientemente desafiador oferecer entre 3 e 5 alternativas para uma pergunta.

III. Aplicações *Quiz* para TVD devem oferecer mecanismos para que o usuário possa navegar entre as alternativas disponíveis.

IV. Cada pergunta pode ter uma única resposta correta ou várias respostas corretas, até o limite imposto pelo número de alternativas. Nesse último caso a pergunta passa a ser chamada de pergunta de múltiplas respostas.

V. Caso a pergunta tenha uma única resposta, ao escolher uma alternativa, a resposta marcada anterior, se existir, deve ser desmarcada. Caso a pergunta permita múltiplas respostas o sistema deve permitir que o usuário marque até todas as alternativas fornecidas. Ainda nesse último caso, o usuário desmarca uma alternativa previamente escolhida, a escolhendo novamente.

VI. Pode haver perguntas em que o usuário deve digitar a resposta solicitada, ao invés de escolher a alternativa. Esse tipo de questão comumente recebe o nome de questão aberta. Em se tratando de aplicações TVD há o limitador de entrada de dados via controle remoto, portanto questões desse tipo devem solicitar respostas de uma, no máximo duas palavras curtas.

VII. Embora em outras formas de exibição, como na Internet, várias perguntas possam ser apresentadas simultaneamente, em um *Quiz* para TVD é conveniente que cada pergunta seja apresentada individualmente na tela juntamente com as respectivas alternativas. Nesse caso cada questão acompanhada de suas alternativas constitui uma página.

VIII. Normalmente deve-se permitir ao usuário avançar ou retroceder páginas.

IX. Em *Quizzes* normalmente existe um mecanismo, geralmente um botão, cujo objetivo é permitir que o usuário indique para o programa que finalizou sua interação, ou seja, que terminou de responder às perguntas. Esse evento torna o usuário apto a conferir as respostas fornecidas com as respostas corretas e, sendo assim, as respostas não poderão mais ser modificadas.

X. Em se tratando de aplicação para TVD, ao finalizar a interação, caso o desenvolvedor tenha fornecido um endereço de servidor externo para envio das respostas e, se for possível estabelecer uma conexão, as respostas do usuário devem ser enviadas.

XI. *Quizzes* também podem utilizar informações do local físico em que estão sendo executados, por exemplo, data e hora do sistema, quantas perguntas estão disponíveis, qual o número correspondente à pergunta sendo visualizada, etc.

4.4.2 Requisitos da interface gráfica do Quiz para TVD

I. Um Quiz deve possuir uma tela de fundo, ou do inglês *background*, que se refere à área onde o aplicativo é projetado.
   a. O *background* pode ser representado por uma área delimitada por uma cor, chamada de cor de fundo.

   b. O *background* pode ser representado por uma imagem estática, que é chamada de imagem de fundo, de modo que se houver imagem de fundo a cor de fundo não será visível.

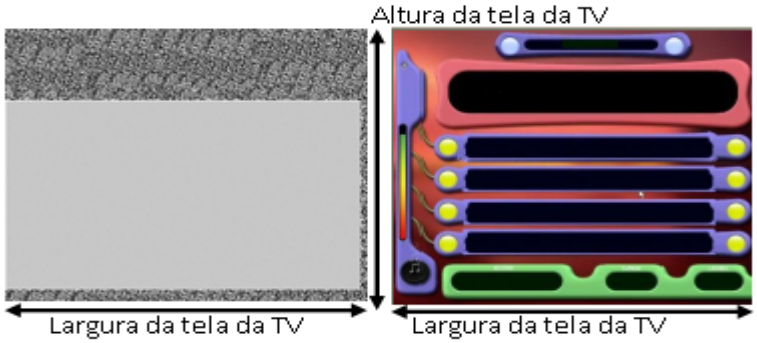

Figura 18 - Exemplos de telas de fundo

A Figura 18 mostra de um lado um exemplo de uma tela de fundo preenchida inteiramente de uma só cor, e do outro uma figura, obtida de autor anônimo a partir da Internet, como tela de fundo. Observe como uma figura pode oferecer ao usuário uma aparência diferenciada.

II. Um *Quiz* geralmente é constituído de um ou mais elementos de texto estáticos.

a. Na tela do aplicativo poderá haver um ou mais elementos de texto, conforme exemplo na Figura 19. Eles são utilizados para visualização da pergunta, das alternativas e de elementos que auxiliam na operação do programa pelo usuário.

b. Em *Quizzes* é importante que características do texto como, tipo de fonte, cor da fonte e tamanho possam ser manipuladas. No lado esquerdo da Figura 19, a cor normal do texto é preta, e vermelha para indicar a seleção atual da navegação, que na Figura 19 é a "Alternativa 3". No lado direito da mesma figura a cor normal do texto é branca e a cor verde é indicada para indicar a seleção como na "Alternativa 2".

c. Conforme exposto no escopo inicial, o uso de um número excessivo de alternativas, pode sobre carregar a tela de elementos gráficos, dificultando sua leitura.

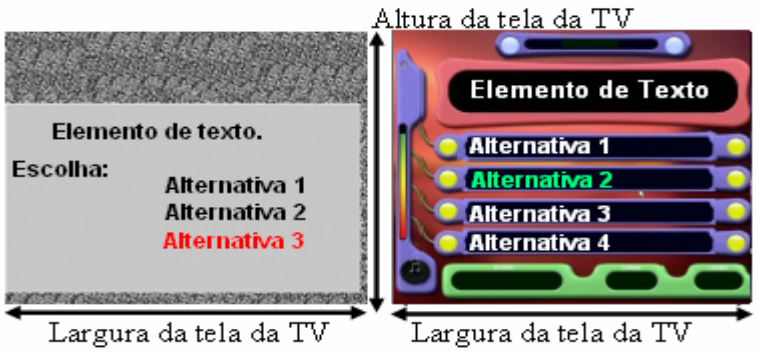

Figura 19 - Elementos de texto no *Quiz*

III. Um *Quiz* pode ser constituído de elementos gráficos (imagens), estáticos ou dinâmicos.

a. Na tela do aplicativo poderá haver um ou mais imagens estáticas ou dinâmicas (vídeo). A Figura 20 ilustra a inserção de figuras aleatórias na tela do *Quiz*. As imagens podem ser meramente decorativas, ou podem estar diretamente relacionadas à pergunta sendo veiculada.

Figura 20 - Elementos de imagem no *Quiz*

IV. Quanto à usabilidade a interface gráfica do *Quiz*, o sistema Moodle [56] indica que são interessantes as seguintes funcionalidades:

a. O *Quiz* deve permitir que o usuário possa navegar entre as alternativas propostas para a questão. A navegação entre as alternativas fica configurada através da possibilidade de distinção entre um elemento selecionado e os demais. Desse modo sempre só haverá uma alternativa selecionada. Uma maneira eficaz de indicar essa navegação é trocar a cor do elemento textual. Pelas características da tela, é conveniente que as alternativas sejam expostas igualmente espaçadas na vertical, por isso a navegação fica mais intuitiva fazendo uso das teclas *up* e *down* do controle remoto.

Figura 21 - Marcadores de escolha no *Quiz*

b. O *Quiz* deve permitir que o usuário possa marcar a alternativa selecionada como forma de indicar sua escolha perante a pergunta formulada. A tecla *enter* do controle remoto é apropriada para que o usuário sinalize a escolha da alternativa como resposta à questão. Para sinalizar a escolha, tendo em vista que a troca da cor do elemento textual já indica a navegação, para sinalizar a escolha do elemento, é conveniente empregar algum outro artifício. Um deles é a inserção de elementos gráficos na lateral esquerda, ou direita, da alternativa escolhida. A Figura 21 ilustra exemplos de elementos gráficos que podem ser utilizados.

c. O *Quiz* deve permitir que o usuário troque facilmente sua escolha. Caso a questão tenha uma única resposta, o evento de escolha sobre outra alternativa basta trocar o elemento gráfico de posição. Caso a questão seja de múltipla escolha, ao escolher um elemento alterna-se entre escolha e não escolha do elemento.

d. O *Quiz* deve ser constituído de uma ou mais questões, e cada uma delas deverá ser apresentada isoladamente na tela, como páginas constituídas por uma pergunta juntamente com suas respectivas alternativas.

e. O *Quiz* deve fornecer um procedimento para que o usuário indique o encerramento da sua disposição em responder às questões apresentadas, ou seja, que terminou de respondê-las. No caso de uma aplicação de TVD uma das maneiras possíveis para isso é a resposta do usuário a uma pergunta adicional específica para esse fim. A plataforma deve fornecer os meios de alterar a frase da pergunta e das respostas do usuário. A Figura 22 ilustra essa situação.

Figura 22 - Pergunta de finalização

f. Ao finalizar a aplicação, a aplicação bloqueia a alteração na escolha das alternativas, e caso tenha a informação das respostas através do arquivo de configuração, a interface deve sinalizar as respostas corretas. Sugere-se nesse momento, mudar a cor de fundo do texto das alternativas consideradas corretas o o que permitirá ao usuário verificar as respostas corretas. A Figura 23 ilustra a visualização das alternativas corretas juntamente com as respostas do usuário. No lado esquerdo, a "Alternativa 3" está indicada como correta, enquanto no lado direito a "Alternativa 1" e "Alternativa 4" estão sinalizadas como corretas.

Figura 23 - Destaque às alternativas corretas

g. Algumas perguntas eventualmente podem ser abertas. Para tanto é necessário que as interfaces tenham suporte a caixas de edição. Essa característica pode ser aproveitada também para solicitar informações importantes e que dizem respeito ao usuário. Por exemplo, nome e senha são informações muito importantes na identificação do usuário e que devem ser enviadas ao servidor. Há várias maneiras de realizar a captura de caracteres a partir do controle remoto. Uma delas, e talvez a mais simplificada é a mesma utilizada em telefones celulares, em que cada tecla representa um número pré-definido de caracteres e o número de toques dentro de um tempo também pré-determinado seleciona o caractere desejado.
   i. Texto de entrada é alfa-numérico.
   ii. Texto de entrada é apenas numérico e visível.
   iii. Texto de entrada é apenas numérico e os caracteres digitados devem ser ocultados.

A Figura 24 ilustra os três tipos de caixas de texto.

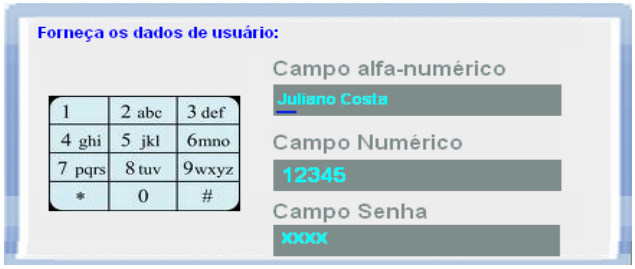

Figura 24 - Interface com caixas de texto

4.5 Considerações finais

Esse capítulo forneceu conceitos importantes sobre aplicações TVD como a classificação das aplicações quanto ao tipo de interatividade e quanto à usabilidade que oferecem. Para aplicações tipo T-Learning foram apresentadas classificações de acordo com o grau de formalidade educacional que empregam. Importantes referências forneceram justificativas suficientes para considerar que aplicações tipo Quiz são bastante utilizadas para contribuição a processos educativos, o que vai diretamente ao encontro dos objetivos estabelecidos nesse trabalho. Desse modo o capítulo finaliza enumerando as principais características desse tipo de aplicação que será utilizado para construção da plataforma proposta. O próximo capítulo procura reunir todos os conceitos e caracteristicas apresentados até aqui para definir as bases da plataforma que se deseja obter.

5 Caracterização conceitual da Plataforma EMTV

Até esse ponto foram estudados conceitos e apresentadas justificativas para provar a relevância de uma plataforma para geração de aplicações TVD com fins educativos. Esse capítulo se valerá desse estudo para caracterizar completamente a plataforma proposta. Inicia-se pelo escopo que agrega as principais questões que levaram às escolhas realizadas. Segue para a listagem das características iniciais básicas que conduzirão todo o desenvolvimento, entre elas a escolha de abordagem baseada em componentes. Finaliza com a identificação dos componentes essenciais para cumprimento dos objetivos previamente estabelecidos neste trabalho.

5.1 Escopo

O desenvolvimento de aplicações complexas, que precisam utilizar abordagem procedural, é uma tarefa difícil devido à diversidade de interfaces e padrões. São muitas as interfaces (bibliotecas) e muitas as preocupações que devem ser levadas em consideração durante a execução de aplicativos, que precisam ser executados nas mais diversas plataformas, de diferentes fabricantes. Um aplicativo para TV digital não pode causar uma exceção, ou seja, um erro de execução no aparelho televisor de um usuário, o que poderia levar à necessidade de reinicialização do aparelho. Dessa forma, muito mais do que conhecer profundamente todas as interfaces disponíveis e seus potenciais, é preciso saber como lidar com todas as possibilidades de falhas.

Já as aplicações declarativas, conforme já abordado, são mais simples, no sentido de que não necessitam se preocupar com tratamento de exceções já que normalmente são tratadas pelo próprio *middleware*. Apesar disso, houve na sua evolução a necessidade de que se tornassem genéricas o suficiente para realizarem as mais diversas tarefas, fazendo-as mais poderosas e conseqüentemente bem mais complexas do que nas suas origens. Por exemplo, para utilizar o GINGA-NCL brasileiro, é necessário tomar conhecimento do modelo NCM que deu origem ao NCL, bem como conhecimentos em scripts LUA ou ECMA conforme apresentado na seção 3.4. Não resta menor dúvida do enorme poder desse sistema na geração de aplicações interativas de alta complexidade, entretanto é inegável que seu uso exige um certo tempo de aprendizado e prévio conhecimento de algumas questões técnicas.

A proposta desse trabalho é definir todo um ambiente de trabalho, ou seja, uma plataforma para geração de uma família de aplicações que tenham aplicabilidade em processos educativos. A plataforma será aberta, sem custo e voltada para geradores de conteúdo de TVD que não disponham profissionais especializados em programação ou mesmo, de recursos para aquisição de caras ferramentas de desenvolvimento para TVD. Espera-se contribuir para a democratização da geração e uso de aplicativos interativos para TVD no Brasil.

Para tanto propõe e apresenta a implementação de uma camada de *software* adicional que fazendo uso do *middleware*, estende suas funcionalidades de modo que, tenha capacidade de gerar aplicações complexas e, ao mesmo tempo, seja fácil de ser manipulada por desenvolvedores, pois minimiza a necessidade de conhecimento sobre bibliotecas de *software* bem como de detalhes de arquitetura do sistema executor. Essa camada de *software* adicional, sob o ponto de vista das aplicações, assume o papel do *middleware* do sistema uma vez que continua realizando a conversão de informações de dados em aplicações interativas. Essa camada

adicional será referenciada ao longo desse texto como "Extensão de *Middleware* para TV Digital" ou, simplesmente pela sigla EMTV. A Figura 25 ilustra a proposta apresentada em forma de diagrama em blocos.

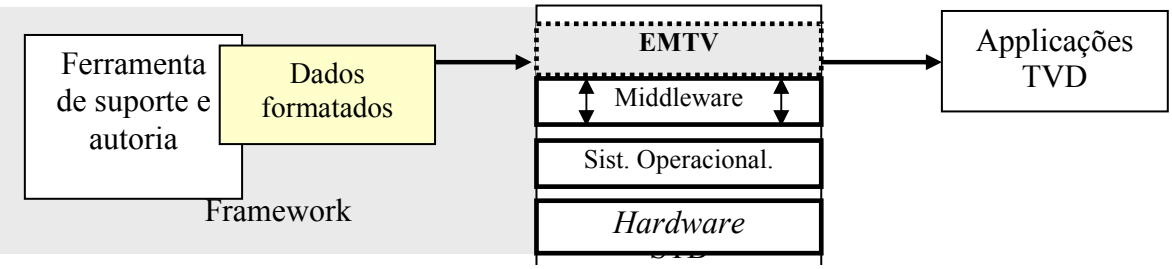

Figura 25 – Diagrama em blocos da plataforma proposta

5.2 Características desejáveis

Partindo dos objetivos definidos desde o início desse trabalho, dos conceitos fornecidos e do escopo fornecido é possível enumerar as principais caracteristicas que se espera alcançar com a plataforma EMTV.

I. Simplicidade e abordagem declarativa

Para cumprir o objetivo de que essa plataforma possa ser facilmente utilizada por programadores sem experiência é apropriado que o EMTV possa ser caracterizado como um *middleware* declarativo. Desse modo, conforme já abordado no capítulo 3, espera-se minimizar a necessidade de que os programadores tenham que ter qualquer conhecimento sobre projetos e estruturação de software típicos da abordagem procedural. Sendo declarativo, o programador terá apenas que fornecer ao EMTV um arquivo em formato que seja convenientemente definido nesse projeto para que seja simples, porém flexível o suficiente para atender as principais demandas dos programadores em termos de funcionalidades.

II. Compatibilidade com o GEM

Embora a plataforma proposta, sob o ponto de vista do programador deva ser declarativa, seu desenvolvimento poderia tanto ser realizado sobre a parte declarativa quanto sobre a parte procedural geralmente presentes nos *middleware* abertos disponíveis. Entretanto foi demonstrado no capítulo 3, que o uso de uma abordagem procedural, que obedeça a especificação GEM, é bastante conveniente. Isso porque, além de oferecer as bibliotecas necessárias e suficientes para o desenvolvimento de qualquer funcionalidade, o que em algum momento poderia ser frustrado com uma abordagem declarativa, essa escolha garantirá que plataforma possa funcionar como uma extensão de qualquer *middleware* compatível com a GEM, incluindo o próprio MHP, o GINGA-J e o ACAP.

A decisão de desenvolver o EMTV através dos recursos oferecidos pela especificação GEM define que a plataforma será um *XLet* Java significando que o seu carregamento poderá inicialmente ocorrer como qualquer outra aplicação procedural compatível com o GEM: Transportada via canal de dados, ou via o canal de retorno e, carregada pelo *middleware* que o executará, sob solicitação do usuário, como um *XLet* Java. A partir daí, o aplicativo assume seu

papel de extensor, dando à plataforma a capacidade de carregar não só o arquivo de configuração mas também arquivos de texto e imagem.

III. Plataforma configurável e de fácil reutilização

É fundamental, sob o ponto de vista desse trabalho, que a plataforma EMTV seja configurável, isto é, que suas características relativas a comportamento e aparência possam ser facilmente modificadas de acordo com as necessidades dos programadores. Além de valorizar a plataforma que poderá ser reaproveitada inúmeras vezes e em diferentes ocasiões converge com a escolha de oferecer abordagem declarativa. A Figura 26 ilustra que um arquivo de configuração contendo informações sobre conteúdo, posicionamento e comportamento de componentes são utilizados pelo EMTV para geração de aplicações.

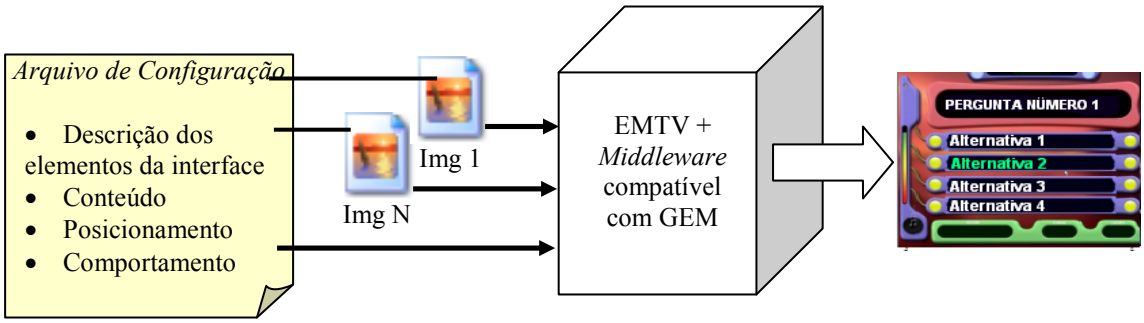

Figura 26 – Esquema de funcionamento da plataforma proposta

IV. Plataforma para geração de *Quiz*

A plataforma proposta nesse trabalho tem o intuito de, em um primeiro momento, ser especialmente útil em processos educativos colaborando não só para democratização da tecnologia de TVD como também para a melhoria das condições de vida das pessoas através da educação. O capítulo 4 desse trabalho se dedicou a demonstrar que existem diversos tipos de aplicações que podem ser distribuídas através de sistemas de TVD e, dentre esses tipos, também no capítulo 4 chegou-se à identificação da relação entre *Quiz* e os processos educativos, principalmente informais e acidentais ou ainda *edutainment*. Pela relação entre *Quiz* e educação, gerar aplicações desse tipo será um dos requisitos essenciais a ser cumprido pela plataforma proposta.

V. Plataforma baseada em componentes

Baseado na leitura do livro de HEINEMAN e COUNCIL (2001), indicado na referência [58], é possível definir que componentes de *software* são artefatos, constituídos de uma ou mais instâncias de classes, de tal sorte que sejam independentes o suficiente para prover alguma funcionalidade de interesse, seja visual, seja comportamental ou ambas. O comportamento e, as características de um componente de *software*, são definidas pelos seus atributos disponibilizados através de métodos públicos que constituem a interface do componente.

HEINEMAN e COUNCIL ainda listam ao longo do texto, vantagens em se desenvolver software baseado em componentes [58]. A Tabela 7 lista algumas dessas vantagens e as relaciona com os objetivos da plataforma EMTV.

Algumas características inerentes aos componentes de software	Relação com os objetivos da plataforma EMTV
O comportamento e, as características de um componente de *software*, são definidas pelos seus atributos disponibilizados através de métodos públicos que constituem a interface do componente.	O arquivo de configuração fornecido para a plataforma EMTV poderá fornecer todas as informações necessárias para manipulação dos atributos e métodos públicos dos componentes envolvidos.
Diferentes combinações de usos de componentes geram aplicações de conteúdo e/ou comportamentos diferentes.	Característica de grande importância dentro dos objetivos da plataforma.
Os componentes podem ser facilmente reutilizados.	Fazendo uso dessa característica a plataforma poderá gerar com facilidade aplicações com tantos elementos quantos forem desejados pelo programador.
Os componentes tornam fácil a manutenção e customização do *software*.	Esse é um requisito não funcional desejável em qualquer sistema de *software.*

Tabela 7 – Vantagens da abordagem baseada em componentes de software e a relação com a EMTV

Não é foco desse trabalho realizar toda a análise sobre a teoria modelagem de componentes conforme sugere HEINEMAN e COUNCIL entretanto, os aspectos apresentados são suficientemente interessantes para definir que a abordagem baseada em componentes é bastante conveniente para o desenvolvimento da plataforma EMTV.

VI. Ferramentas de manipulação e validação do arquivo de configuração

Conforme já exposto o arquivo de configuração a ser utilizado deverá ser simples o suficiente para que possa ser manipulado por programadores sem conhecimentos avançados em programação. Ainda assim, para que a plataforma fique totalmente caracterizada é necessário que sejam oferecidos meios para validação e edição desse arquivo.

A ferramenta de edição é importante pois oferece ao programador uma facilidade a mais no momento de descrever o conteúdo e o comportamento das aplicações. A validação é importante pois apesar da simplicidade na formatação, quanto maior o número de componentes que o programador deseje ter na aplicação maior o tamanho físico do texto do arquivo de configuração necessário para descrevê-lo.

5.3 Representação em blocos da plataforma EMTV

A Figura 27 representa conceitualmente a estrutura em blocos da plataforma EMTV. Conforme apresentado na caracterização da EMTV faz-se necessário um bloco que fará a interpretação do conteúdo do arquivo de configuração que será fornecido pelo programador. Esse arquivo contém informações sobre todos os componentes que serão criados e manipulados por um bloco gerenciador de componentes. Os componentes têm seu comportamento dependente não só das informações contidas no arquivo de configuração como também de eventos provenientes

do sistema de TVD que incluem eventos de sinalização do sinal e também eventos provenientes do controle remoto do usuário da aplicação. A realização de todo sistema só é possível graças à atuação do *middleware* compatível com o GEM que será utilizado através da implementação de interfaces adequadas como parte do EMTV.

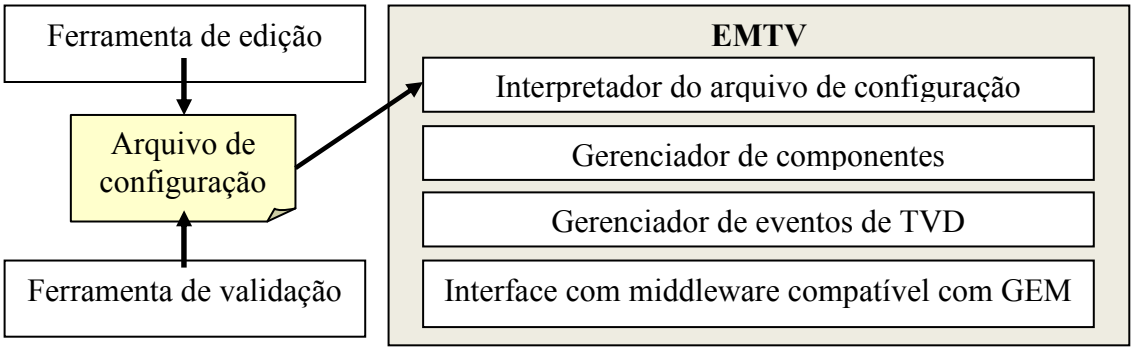

Figura 27 – Estrutura em blocos da plataforma EMTV

5.4 Definição dos componentes mínimos da EMTV

Outra vantagem de desenvolver a EMTV baseada em componentes é que esses artefatos possuem além do comportamento, também implementações independentes e que, a princípio, não deveriam interferir uns nos outros. Essa característica permitirá o desenvolvimento contínuo e em etapas da plataforma, isto é, a plataforma poderá progredir continuamente na medida em que novos componentes sejam implementados. Neste trabalho pretende-se apenas implementar os componentes básicos necessários para o cumprimento do objetivo de gerar aplicações tipo *Quiz* de TVD com a possibilidade de uso, ou não, do canal de retorno.

Com o auxílio da Figura 28, que resume as considerações apresentadas nos tópicos 4.4.1 e 4.4.2 quanto às características necessárias em aplicações tipo *Quiz*, é possível identificar os componentes mínimos que devem compor a plataforma EMTV.

Figura 28 – Identificação dos componentes básicos

Fica então determinado que os componentes mínimos necessários para essa primeira versão do EMTV serão:

I. Componente *Connection* (CONN): Para realizar a conexão com um servidor remoto através do canal de retorno.
    a. Deve tentar realizar uma conexão com um servidor remoto através de rede TCP/IP preferencialmente por conexão física permanente, se disponível ou tentar realizar a discagem utilizando linha telefônica, se disponível fisicamente.
    b. Deve permitir que o programador defina o endereço IP de destino e a porta.
    c. Deve fornecer regras para montagem da string de envio com o preenchimento de informações coletadas na própria aplicação em funcionamento.

II. Componente *Screen* (SCR): Para controlar as telas, isto é, do plano de fundo da aplicação
    a. Deve permitir o controle das dimensões e posicionamento da janela no monitor de TV.
    b. Deve permitir definir cores para a janela.
    c. Deve permitir definir uma figura de fundo.

III. Componente *Image* (IMG): Para inserir imagens na tela
    a. As imagens serão carregadas a partir do carrossel de dados ou através do canal de retorno.
    b. Deve permitir o posicionamento da imagem na tela.
    c. Deve permitir instanciar quantas imagens quanto o programador desejar.
    d. Deve prover maneira de controlar a visibilidade da imagem.

IV. Componente *Image Button* (IMGBTN): Extensão de IMG, para exibir uma imagem conforme um evento do controle remoto.
    a. Apresenta uma imagem numa condição normal e apresenta outra imagem durante alguns milésimos de segundo após um evento definido pelo programador.

V. Componente *Text* (TXT): Para inserir textos de uma linha sobre a tela
    a. Deve permitir o posicionamento, a dimensão e a fonte do texto na tela.
    b. Deve permitir controlar cor da fonte e do fundo do texto.
    c. Deve permitir mostrar informações coletadas do aplicativo.
    d. Deve prover maneira de controlar a visibilidade do texto.

VI. Componente *Question Group* (QUEST): Para controlar todas as funcionalidades exigidas por uma aplicação tipo Quiz
    a. Deve permitir o programador inserir as perguntas e suas respectivas alternativas.
    b. Deve permitir que o número de alternativas de uma pergunta é independente do número de alternativas de outra.
    c. Deve permitir o posicionamento, a cor da fonte e a cor do fundo utilizado nas perguntas.
    d. Deve permitir o posicionamento inicial, o incremento na coordenada X e na coordenada Y, a cor da fonte e a cor do fundo das alternativas.
    e. Deve controlar a navegação entre as alternativas.
    f. Deve permitir a escolha da cor da fonte a ser utilizado quando uma alternativa está em foco.
    g. Deve permitir a definição e o posicionamento de uma imagem que servirá para indicar a escolha de uma alternativa como resposta. O posicionamento deve ser

definido a partir de um valor de incremento na coordenada X e na coordenada Y em relação à posição da alternativa selecionada.

h. Deve permitir a definição se a pergunta possui uma única resposta ou múltiplas respostas.
i. Deve oferecer possibilidade de converter as alternativas em caixas de texto alfa-numéricas, numéricas ou senhas numéricas.
j. Deve permitir a definição da pergunta de finalização de interação.
k. Deve permitir a escolha da cor de fundo para alternativas tidas como corretas. Essa cor só será utilizada após a finalização da interação.

5.5 Considerações finais

Esse capítulo apresentou conceitualmente a plataforma EMTV. Definiu as principais características que deverão reger o funcionamento da plataforma como o fato de utilizar *middleware* compatível com o GEM apesar de que, sob o ponto de vista do programador, a EMTV se apresentará como uma plataforma declarativa visando a simplicidade de utilização. A plataforma em um primeiro momento estará voltada para aplicações tipo *Quiz* que têm comprovadamente utilidade em processos educativos. Outra importante definição é a escolha de desenvolvimento de uma plataforma baseada em componentes, que contribuirá de maneira muito positiva para vários dos objetivos estabelecidos. O capítulo se encerra com o levantamento dos componentes que serão fundamentais nessa primeira versão da plataforma. O próximo capítulo será dedicado ao esclarecimento e registro da implementação da plataforma EMTV.

6 Arquitetura e Implementação da Plataforma EMTV

A Figura 29 demonstra a metodologia que foi adotada para implementação da plataforma EMTV. A primeira parte dessa metodologia foi concluída no capítulo anterior e consistia da definição conceitual da plataforma que se deseja obter. Desse ponto em diante serão descritas as ações que foram tomadas para obtenção da plataforma proposta. As ações são divididas entre uma etapa de levantamento da arquitetura, que converte os conceitos elaborados no capítulo 5 em soluções técnicas, e uma etapa de implementação propriamente dita que registra aspectos técnicos de programação.

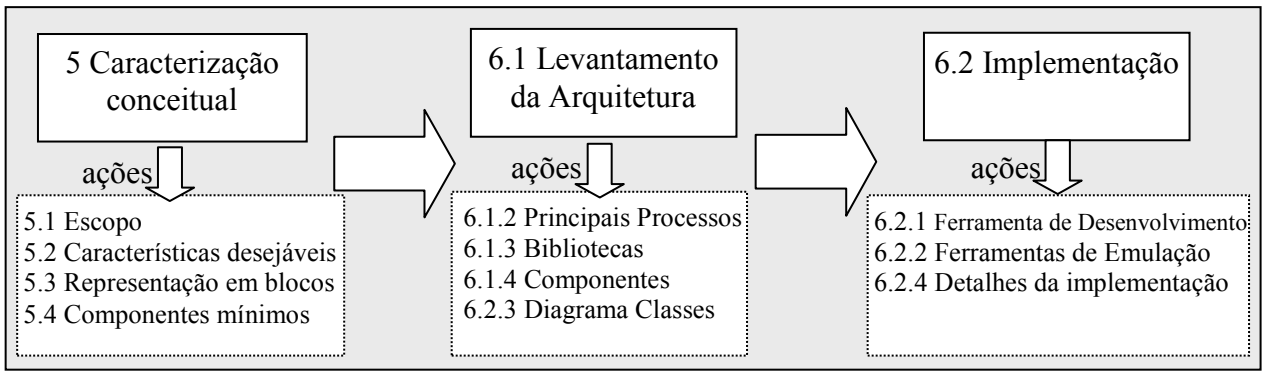

Figura 29 - Metodologia para construção da plataforma

6.1 Levantamento de arquitetura

O levantamento dos principais processos será importante para guiar o processo de desenvolvimento do sistema. Os processos descritos a seguir representam apenas uma simplificação esquemática do sistema. Cada processo poderá compreender uma ou mais classes de controle que dependem da análise do projeto.

6.1.1 Requisitos não funcionais importantes

As definições fornecidas no capítulo 5 fornecem as principais características funcionais para a construção da plataforma. Mas existem também alguns aspectos não funcionais que são igualmente importantes, não só para o bom funcionamento da plataforma, como também para o cumprimento dos objetivos estabelecidos no projeto:

I. Prover tratamento adequado de erros

É essencial que em todo o código da plataforma estejam previstas o tratamento adequado e amigável de todo e qualquer erro que possa vir a acontecer durante a execução do aplicativo.

II. Minimizar uso de espaço físico, de memória e de processamento

É imprescindível levar em consideração a escassez de recursos de processamento e memória do STB. Por essa razão o código fonte da plataforma deve procurar ser o menor e mais

eficiente possível. Nesse contexto, menor se refere ao número de *bytes* e influencia no tempo de carregamento e no número de rotinas a serem executadas. O parâmetro eficiência, apesar de ser subjetivo, pode ser validado se a velocidade de resposta do programa devido a uma ação do usuário, ou do próprio sistema, ficar em torno de 1 segundo.

III. Formato do arquivo de configuração legível e fácil de ser entendido

Um dos objetivos mais importantes da plataforma é simplificar a ação dos desenvolvedores, por isso, o formato do arquivo de configuração deverá apelar tanto quanto possível para simplicidade, evitando recair sobre o mesmo problema de outras linguagens declarativas que, conforme já mencionado no capítulo 3 desse texto, tornaram-se demasiadamente complexas. Isso também colabora para manter o interpretador compacto e a execução rápida.

6.1.2 Processos

Pelo que foi exposto anteriormente já é possível identificar alguns processos de software, ou seja, módulos funcionais de software que deverão estar presentes no EMTV ilustrados na Figura 30.

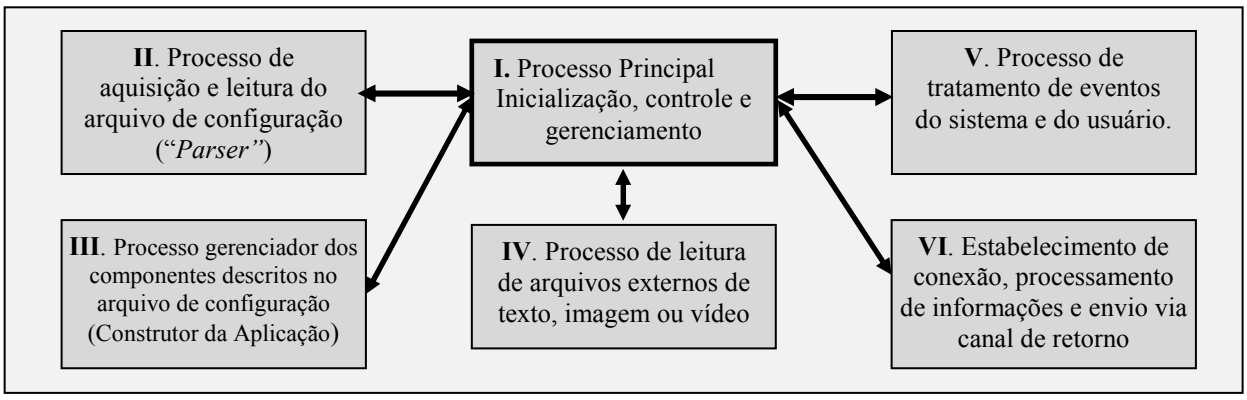

Figura 30 - Principais processos do EMTV

I. Processo principal

Será o processo coordenador do sistema. Responsável pela inicialização e coordenação dos demais processos. Da forma como está idealizado deve, repassar os eventos recebidos pelo STB, seja proveniente do canal de dados, seja do controle remoto para tratamento pelos processos. O processo principal também é o responsável pela finalização adequada dos recursos de memória utilizados pela plataforma.

II. Processo de aquisição e leitura do arquivo de configuração

Mais do que realizar a leitura do arquivo a partir do stream de dados do sinal digital, ou do canal de retorno, esse processo poderá ser capaz de detectar alguma alteração no arquivo na sua origem. Isso será muito útil quando o gerador de programação desejar alterar

sincronizadamente o comportamento das aplicações já iniciadas. Esse processo essencialmente não realiza nenhuma operação com as informações adquiridas.

III. Processo gerenciador de componentes

Construtor da aplicação, ou builder: Realiza o processamento das informações extraídas pelo processo de leitura do arquivo de configuração. O objetivo desse processo é identificar os componentes suportados pela plataforma, criá-los de acordo com as características apontadas e, finalmente armazená-los em estruturas adequadas para o gerenciamento por outros processos. Esse processo possui fundamental importância na plataforma pois de fato cria e exibe a interface com o usuário.

IV. Processo de carregamento de arquivos externos

De maneira similar ao processo de carregamento do arquivo de configuração, esse processo idealmente realiza o carregamento síncrono ou assíncrono de arquivos indicados no arquivo de configuração como imagens estáticas, imagens de vídeo, arquivos de texto e eventualmente de sons.

V. Processo de tratamento de eventos

É o processo responsável pela operação das funcionalidades do sistema. Uma vez carregados todos os componentes gráficos e não gráficos esse processo realiza o tratamento de todos os eventos de entrada e saída do sistema que são repassados pelo processo principal. Sendo assim responde a eventos do controle remoto e cuida da modificação das características dos componentes da aplicação.

VI. Processo de para envio de informações pelo canal de retorno

Esse processo é idealizado para realizar o estabelecimento e a monitoração de conexões para envio de informações via canal de retorno. As conexões podem ser discadas ou fixas e estabelecidas de acordo com informações fornecidas descrito no arquivo de configuração. É também responsável pela montagem das informações para serem enviadas, provenientes das respostas do usuário às perguntas do *Quiz*.

6.1.3 Bibliotecas utilizadas nos processos

Um novo passo para o desenvolvimento da plataforma é esboçar uma relação das bibliotecas de funções que devem ser utilizadas para o cumprimento dos objetivos e das características. O ponto de partida é o conhecimento das bibliotecas utilizadas pela especificação GEM. Parte-se então para a análise dos principais processos da aplicação, listados no tópico 6.1.2, para predição do que será necessário para o desenvolvimento das funcionalidades. As Tabela 8, Tabela 9, Tabela 10, Tabela 11 e, Tabela 12 relacionam as bibliotecas que serão utilizadas nos processos identificados às principais ações a serem executadas.

Processo Principal	
Bibliotecas	Ações
JavaTV	Implementação da interface *XLet* em conformidade com a especificação GEM.
HAVI	Para configuração e acesso às funções gráficas para apresentação da biblioteca. Isso inclui acesso às classes: • HScreen: Refere-se à representação da tela onde o sistema será projetado. • HScene: Classe que implementa um contêiner para recebimento de componentes gráficos da HAVI e AWT fazendo uso de configurações presentes HScreen. Permite o controle de tamanho e posicionamento da área visível de componentes gráficos. • HSceneFactory: Para conseguir a partir do método getBestScene uma instancia de HScene.
DAVIC	Para controle de alocação e liberação de recursos para o *XLet* através da implementação de funções da interface ResourceClient e de uso da classe ResourceProxy.
Java	Algumas classes fornecidas pelo Java já são visivelmente necessárias como: • Runnable: Para utilização de Threads • KeyListener: Para aquisição de eventos de teclas do controle remoto.

Tabela 8 – Bibliotecas a serem utilizadas no processo principal

Processo de aquisição e leitura do arquivo de configuração	
Java ou DVB	O foco desse processo será lidar com solicitações e manipulação de arquivos. A especificação GEM abre duas possibilidades para realizar essa função: • DSMCCObject: Trata-se de uma classe fornecida pela biblioteca DVB e dá acesso à funcionalidades de carregamento de arquivos utilizando o carrossel de dados. Essa abordagem é vantajosa por fornecer a possibilidade de carregamento assíncrono de arquivos além de notificar caso haja alteração no arquivo na origem. • FILE: Essa classe fornecida pelo Java também dá acesso ao carregamento de arquivos externos, porém de maneira síncrona. O uso dessa classe é a princípio mais simples que a utilização do DSMCCObject. Durante a fase de desenvolvimento será utilizada a classe FILE tendo em vista o desejo de utilizar a classe DSMCCObject.

Tabela 9 – Bibliotecas a serem utilizadas no processo de aquisição e leitura do arquivo de configuração

Processo gerenciador de componentes	
nanoXML	A especificação GEM não especifica diretamente nenhuma biblioteca ou mesmo classe para manipulação de arquivos em formato XML. Sendo assim, como o sistema requer tal funcionalidade, é necessário provê-lo de tal funcionalidade. Na página de internet Interactive TV Web [35], existem recomendações de algumas bibliotecas externas gratuitas que poderiam exercer essa função. Entre as recomendadas encontra-se a biblioteca nanoXML [57], que possui uma versão chamada de nanoXML lite que oferece todas as funcionalidades básicas necessárias e ainda destaca-se pela eficiência e pequeno tamanho em bytes (cerca de 6 KB).
Java	Ao interpretar o conteúdo do arquivo de configuração, o sistema gera objetos que representam os componentes descritos. Esses novos objetos serão inseridos em listas cujas funcionalidades são fornecidas pela biblioteca Java, como por exemplo, a *ArrayList*.
*	Além de acessar as bibliotecas já mencionadas no processo principal

Tabela 10 – Bibliotecas a serem utilizadas no processo gerenciador de componentes

Processo de tratamento de eventos	
Bibliotecas	Principais classes acessadas
Java	Java.AWT.Image: Para exibição de imagens estáticas. Java.AWT.JMF.MediaTracker: Para controle de exibição de imagens. Threads: Algumas Threads serão necessárias para realização de processamento de eventos evitando travamentos na interface. KeyListener: Para tratamento de teclas fornecidas pelo processo principal.
HAVI	HText: Para exibição de textos em geral. HSinglelineEntry: Exibição de textos editáveis de uma linha (para entrada).
DVB	DVBColor: Para controle de cores dos objetos.
*	Além de acessar as bibliotecas já mencionadas no processo principal.

Tabela 11 – Bibliotecas a serem utilizadas no processo de tratamento de eventos

Processo para envio de informações pelo canal de retorno	
Bibliotecas	Principais classes acessadas
Java	Threads: Para realizar o processamento de eventos ligados a conexão e transmissão de dados sem comprometer o comportamento da interface gráfica.
DVB	RCInterfaceManager: Para gerenciamento de conexões estabelecidas. ConnectionRCInterface: Para aquisição de informações de interfaces de conexão TCP/IP.
*	Além de acessar as bibliotecas já mencionadas no processo principal.

Tabela 12 – Bibliotecas a serem utilizadas no processo de controle do canal de retorno

Já é possível visualizar que a plataforma utilizará apenas uma biblioteca adicional em relação ao que já é oferecido pela especificação GEM. Trata-se da biblioteca NanoXML, utilizada no processo de interpretação do arquivo de configuração e, que é considerada uma biblioteca pequena (~ 6K bytes) e eficiente. Se, utilizando tão poucos recursos, perto de tantos outros que poderiam ser explorados, for suficiente para a implementação dos requisitos levantados para a EMTV, esse projeto terá cumprido seu papel em elaborar uma plataforma simples e eficaz na solução de um problema.

6.1.4 Identificação, criação e gerenciamento dos componentes

Conforme abordado nas seções 5.3 e 5.4, componentes de software são elementos que podem ser manipulados através de atributos e métodos que são disponibilizados na sua interface pública. O processo de gerenciamento de componentes construirá os componentes baseado nas informações fornecidas no arquivo de configuração que desse modo deve ser organizado de maneira a descrever inequivocamente os componentes a serem criados e utilizados pela EMTV.

O formato XML, que é bastante comum principalmente na Internet, é bastante conveniente para esse contexto, pois resulta em um arquivo legível, fácil de ser utilizado tanto por humanos quanto pela plataforma que poderá facilmente identificar elementos e atributos que se referem a componentes a serem criados na plataforma. Outra vantagem do XML é que esse formato define um método de validação, realizado através de um arquivo adicional, chamado de *Document Type Definition* (DTD). Um DTD descreve as regras de formação de arquivos XML. Existem diversos programas, muitos dos quais gratuitos que, além de servirem para edição, fazem a leitura de arquivos de DTD e são capazes de apontar possíveis falhas de sintaxe do documento. A validação é extremamente importante para evitar erros na geração dos aplicativos que serão distribuídos por sistemas de TVD e, sob o ponto de vista desse trabalho, fornece um meio de validação que faz parte dos requisitos da plataforma estabelecidos na seção 5.2.

O processo de gerenciamento realiza a leitura e identificação dos componentes no arquivo de configuração de maneira seqüencial, do começo até o final do arquivo. Isso é importante, pois a ordem da descrição dos componentes tem duas conseqüências imediatas:

- Componentes gráficos descritos posteriormente estarão à frente de componentes descritos anteriormente se houverem pontos de interseção. Essa situação está ilustrada na Figura 31. Certos efeitos visuais só poderão ser obtidos observando-se essa regra.

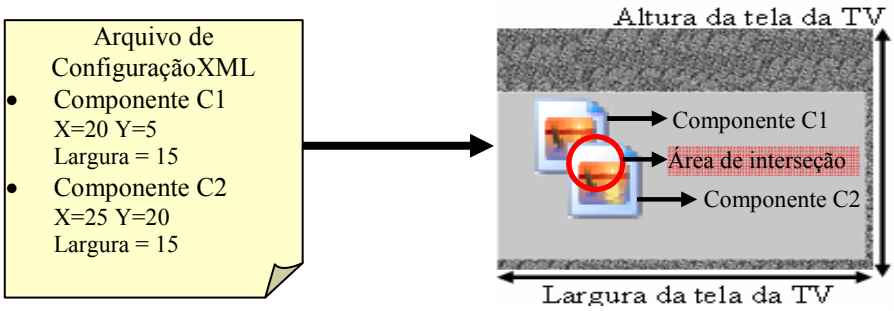

Figura 31 – Ordem de inserção dos componentes gráficos na tela

- Os componentes são inseridos não só na tela, mas também na lista de objetos que têm eventos monitorados. Essa lista é também utilizada para realização do refrescamento da

tela, portanto elementos inseridos antecipadamente têm reação a eventos e atualização mais rápida. Naturalmente que devido à alta velocidade de processamento do STB esse efeito torna-se imperceptível.

Cada componente suportado pelo EMTV possui rotinas próprias de interpretação dos seus atributos de modo que, a criação dos componentes, segue a seqüência apresentada na Figura 32:

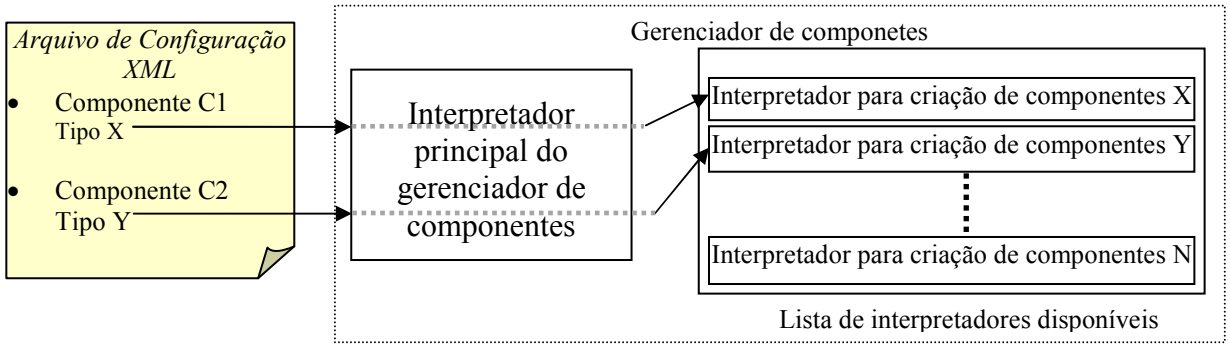

Figura 32 - Processo de interpretação dos componentes

Para essa versão inicial do sistema foram desenvolvidos componentes já apresentados no tópico 5.4 desse texto. Antes de realizar a apresentação de detalhes da especificação dos componentes faz-se necessário definir algumas convenções utilizadas nos campos dos componentes:

I. Quanto ao nome dados aos campos
   a) Todos os nomes dos campos começam em "i", indicando se tratar de um campo tipo inteiro, ou começam em "s", indicando se tratar de um campo textual ou *string*. O objetivo com isso foi o de facilitar os algoritmos de interpretação dos campos, tornando-os mais eficientes ou mais fáceis de serem mantidos.
   b) Quando há agrupamento de campos, utiliza-se uma sigla derivada do inglês como nome do grupo.

II. Campos representando cores
   a) Um campo que indica uma cor é representado através de um texto com a descrição de 4 valores inteiros separados por ";" que variam de 0 a 255, representando respectivamente o nível de vermelho, verde, azul e o nível de transparência (RGB + *alpha-blending*).

III. Conteúdo
   a) Os campos numéricos são todos inteiros positivos, salvo exceções em que for explicitamente colocado diferentemente.
   b) Praticamente todos os campos indicados para um determinado componente, nessa versão inicial do sistema, têm que estar presentes no arquivo de configuração, sob pena do sistema recusar o arquivo de configuração. Há raros casos, que serão mencionados em que o campo não tem presença obrigatória junto aos demais campos do componente no arquivo de configuração.

6.1.5 Caracterização e sintaxe dos componentes no arquivo de configuração

I. Componente "*Connection*" (CONN)

a) Descrição

Trata-se do único componente não visual necessário para o atendimento dos requisitos básicos da plataforma. Na versão inicial do sistema esse componente só é acionado pela indicação do fim da interação, conforme apresentado na Figura 22, e entre suas ações mais relevantes estão:

i. Responsável pelas tentativas de estabelecimento de conexão a uma rede TCP/IP através de uma interface Ethernet ou através de linha discada.
ii. Montagem da mensagem contendo informações relativas às respostas das questões respondidas pelo usuário do sistema.
iii. Envio da mensagem e geração de evento utilizado pelo sistema para indicação do sucesso ou falha do procedimento.

Os campos desse componente estão distribuídos em dois grupos:

i. "*Dial"*: Campos relativos ao estabelecimento de conexão discada. Só serão utilizados se, não há conexão TCP/IP fixa disponível através de um cabo Ethernet e há uma interface tipo V.90 (modem para conexão discada usando linha telefônica com limite de 56 Kbps).
ii. "SRV": Campos relativos à informação sobre o servidor para o qual deverão ser enviadas as informações coletadas na aplicação. Essas informações só são utilizadas caso uma conexão TCP/IP seja estabelecida.

b) Restrições de uso:

Deve haver um único componente tipo Connection no arquivo de configuração. Da forma como está projetado inicialmente, declarações subseqüentes sobre escreverão as anteriores.

c) Ilustração:

Não é um componente visual.

A Tabela 13, logo abaixo, fornece a relação dos campos desse componente.

Identificador do campo	Tipo do campo	Descrição do campo
Campos do grupo "Dial"		
sPhone	Texto	O número para ser discado para estabelecimento da conexão *dial up*.
sUsr	Texto	O usuário a ser utilizado para autenticação do estabelecimento da conexão *dial up*.

Identificador do campo	Tipo do campo	Descrição do campo
Campos do grupo "SRV"		
sAddr	Texto	O endereço IP do servidor que receberá as informações.
sPort	Texto	A porta para tentativa de conexão ao servidor que receberá as informações.
sInfo	Texto	Um texto, contendo curingas sinalizados por começarem com o caractere '$', e que será utilizado para montagem da informação a ser enviada. O item 6.1.6 esclarece a atuação do componente quanto ao campo "*sInfo*".

Tabela 13 – Atributos do componente "*Connection*"

d) Descrição de um componente tipo "*Connection*" no arquivo XML:

```
<CONN>
        <DIAL>
            <sPhone> 36146500 </sPhone>
            <sUsr> itv </sUsr>
            <sPass> password </sPass>
        </DIAL>
        <SRV>
            <sAddr> www.terra.com.br </sAddr>
            <sPort> 80 </sPort>
            <sInfo>/index.php?alpha=$0.1&num=$0.3&pass=$0.5&$D-$M-$A-$H:$m&resp=$resp </sInfo>
        </SRV>
</CONN>
```

II. Componente "*Screen*" (SCR)

a) Descrição:

Trata-se de um componente gráfico que representa a tela sobre a qual todos os demais componentes gráficos são projetados. Os atributos desse componente oferecem permitem a customização da sua aparência.

b) Restrições de uso:

Deve haver um único componente tipo "*Screen*" no arquivo de configuração. Da forma como está projetado inicialmente, declarações subseqüentes sobre escreverão as anteriores.

c) Ilustração:

A ilustração desse componente é bem retratada pela Figura 18.

A Tabela 14 fornece a relação dos campos desse componente.

Identificador do campo	Tipo do campo	Descrição do campo
iid	Inteiro	Um identificador inteiro para a tela
ix	Inteiro	Um valor para coordenada cartesiana x onde começa a tela. 0 é o topo esquerdo do monitor TV e cresce para baixo.
iy	Inteiro	Um valor para coordenada cartesiana y onde começa a tela. 0 é o topo esquerdo do monitor da TV e cresce da esquerda para direita.
ih	Inteiro	Um valor que representa a altura da tela a partir do ponto iy em direção à parte inferior do monitor da TV.
iw	Inteiro	Um valor inteiro que representa a largura da tela a partir do ponto ix em direção à parte inferior do monitor da TV
sBgcolor	Texto	A cor de fundo para o componente tipo "*Screen*" conforme a Figura 18 do lado esquerdo. Lembrando que se uma imagem for indicada no campo "*sBgImg*", essa cor de fundo não será visível onde for sobre escrita pela figura.
sBgImg	Texto	Indica o caminho completo de um arquivo contendo uma figura para ser buscado via canal de dados, através do *carousel* do sistema de TV digital, ou para ser buscado via canal de retorno. O formato do arquivo de imagem preferencialmente é o JPG, GIF ou PNG. O suporte à transparência de formatos como GIF e PNG depende do STB, portanto num primeiro momento deve ser evitada.

Tabela 14 – Atributos do componente "*Screen*"

d) Observações:

i. Para o funcionamento do sistema a presença desse componente no arquivo de configuração é obrigatória.

e) Descrição de um componente tipo "*Screen*" no arquivo XML:

<**SCR**>
<**iid**> 0 </**iid**>
<**ix**> 10 </**ix**>
<**iy**> 15 </**iy**>
<**ih**> 570 </**ih**>
<**iw**> 550 </**iw**>

<**sBgcolor**> 200;150;45;220;</**sBgcolor**>
<**sbgImg**> bg2.png </**sbgImg**>
</**SCR**>

III. Componente "*Text*" (TXT)

a) Descrição:

Representa um elemento de texto, estático ou não que será colocado sobre a tela da aplicação sendo formatada.

b) Restrições de uso:

Não há restrições quanto ao número desses componentes na aplicação.

c) Ilustração:

A ilustração desse componente é bem retratada pela Figura 19.

A Tabela 15 fornece a relação dos campos desse componente.

Identificador do campo	Tipo do campo	Descrição do campo
iid	Inteiro	Um identificador inteiro para a tela
ix	Inteiro	Um valor para coordenada cartesiana x para o componente visual. 0 é o topo esquerdo do monitor TV e cresce para baixo.
iy	Inteiro	Um valor para coordenada cartesiana y para o componente visual. 0 é o topo esquerdo do monitor da TV e cresce da esquerda para direita.
ih	Inteiro	Um valor que representa a altura da tela a partir do ponto iy em direção à parte inferior do monitor da TV.
iw	Inteiro	Um valor inteiro que representa a largura da tela a partir do ponto ix em direção à parte inferior do monitor da TV
sValue	Texto	O texto para ser inserido propriamente dito. Esse campo suporta algumas palavras chave que representam variáveis com informações sobre o funcionamento do sistema conforme o item 0.
sBgcolor	Texto	A representação de uma cor para a caixa de texto compreendida dentro da área estabelecida pelos parâmetros ix,iw,iy,ih.

sForeColor	Texto	A representação de uma cor para a face do texto.
sFontName	Texto	O nome da fonte que será utilizada para apresentação do texto. Se omitida deve ser utilizada a fonte Arial, geralmente presente no sistema. A fonte deve pertencer ao grupo de fontes suportadas pela máquina virtual Java do STB. Outra fonte bastante mencionada nas referências é a Tiresias.
iFontSize	Inteiro	Representa o tamanho da fonte para ser utilizada na apresentação do texto. Para a fonte ARIAL o tamanho 18 é uma referencia para boa visualização.
sCondition	Texto	Um texto que representa uma expressão booleana que valida a apresentação ou não do texto na tela de acordo com informações disponíveis da própria aplicação. Essa funcionalidade, compartilhada por outros componentes está descrita em 6.1.8.
sEffect	Texto	Um texto que pode assumir os valores "*scroll*", "*blink*" ou ainda ser deixado vazio, para representação de um efeito no texto. Se o valor é "*scroll"* apresenta-se um efeito de deslocamento na tela de um caractere do texto a cada 800ms (milissegundos) da esquerda para direita. Se o valor é "*blink*" o efeito será de fazer com que o texto pisque no período de 800ms (milissegundos). Se o parâmetro for deixado vazio não haverá efeito no texto.

Tabela 15 – Atributos do componente "Text"

d) Observações:

i. Se os valores para ix e iy corresponderem à uma posição fora da área delimitada pelo componente "Screen" o componente não será visível.

e) Descrição de um componente "*Text*" no arquivo XML:

<**TXT**>
<**iid**> 1 </**iid**>
<**ix**> 68 </**ix**>
<**iy**> 500 </**iy**>
<**iw**> 440 </**iw**>
<**ih**> 30 </**ih**>
<**sValue**> UFAM – Universidade Federal do Amazonas. </**sValue**>
<**sBgColor**> 236; 236; 236; 80; </**sBgColor**>
<**sForeColor**> 0; 20; 225; 220; </**sForeColor**>
<**sFontName**> ARIAL </**sFontName**>
<**iFontSize**> 18 </**iFontSize**>
<**sCondition**> NOT $IE </**sCondition**>

<**sEffect**> SCROLL </**sEfect**>
</**TXT**>

IV. Componente "*Image*" (IMG)

a) Descrição:

Representa uma imagem sobre a tela será colocada sobre a tela da aplicação sendo formatada.

b) Restrições de uso:

Não há restrições quanto ao número desses componentes na aplicação.

c) Ilustração:

A ilustração desse componente é bem retratada na Figura 20.

d) Observações:

i. A largura e altura da imagem são dadas pela dimensão física da figura indicada no arquivo.
ii. Se os valores para ix e iy corresponderem a uma posição fora da área delimitada pelo componente "*Screen*" o componente não será visível.

e) Descrição de um componente "*Image*" no arquivo XML:
<**IMG**>
<**iid**> 9 </**iid**>
<**ix**> 160 </**ix**>
<**iy**> 320 </**iy**>
<**sFile**> theend.png </**sFile**>
<**sCondition**> $IE </**sCondition**>
</**IMG**>

A Tabela 16 fornece a relação dos campos desse componente.

Identificador do campo	Tipo do campo	Descrição do campo
iid	Inteiro	Um identificador inteiro para a tela.
ix	Inteiro	Um valor para coordenada cartesiana x para o componente visual. 0 é o topo esquerdo do monitor TV.
iy	Inteiro	Um valor para coordenada cartesiana y para o componente visual. 0 é o topo esquerdo do monitor da TV.

Identificador do campo	Tipo do campo	Descrição do campo
sFile	String	Indica o caminho completo de um arquivo contendo uma figura para ser buscado via canal de dados, através do *carousel* do sistema de TV digital , ou para ser buscado via canal de retorno. O formato do arquivo de imagem preferencialmente é o JPG, GIF ou PNG. O suporte à transparência de formatos como GIF e PNG depende do STB portanto num primeiro momento deve ser evitada.
sCondition	String	Um texto que representa uma expressão booleana que valida a apresentação ou não do texto na tela de acordo com informações disponíveis da própria aplicação. Essa funcionalidade, compartilhada por outros componentes está descrita em 6.1.8.

Tabela 16 – Atributos do componente "*Image*"

V. Componente "*Image Button*" (IMGBTN)

a) Descrição:

Representa uma área que apresenta duas diferentes imagens dependendo da captura de eventos do controle remoto. Trata-se de um componente cujo principal propósito é simular o efeito visual do pressionamento de um botão. A imagem sendo mostrada é trocada dependendo da detecção ou não do pressionamento de uma tecla cujo código seja indicado por configuração.

b) Restrições de uso:

Não há restrições quanto ao número desses componentes na aplicação.

c) Ilustração:

A Figura 33 mostra o exemplo de duas imagens que podem ser utilizadas em um componente como esse. A troca de imagens dá a sensação visual de pressionamento de um botão.

Figura 33 – Imagens com efeito de pressionamento no botão

A Tabela 17 fornece a relação dos principais campos desse componente.

Identificador do campo	Tipo do campo	Descrição do campo
iid	Inteiro	Um identificador inteiro para a tela

Identificador do campo	Tipo do campo	Descrição do campo
ix	Inteiro	Um valor para coordenada cartesiana x para o componente visual. 0 é o topo esquerdo do monitor TV.
iy	Inteiro	Um valor para coordenada cartesiana y para o componente visual. 0 é o topo esquerdo do monitor da TV.
sFile	String	Indica o caminho completo de um arquivo contendo uma figura para ser buscado via canal de dados, através do *carousel* do sistema de TV digital, ou para ser buscado via canal de retorno. O formato do arquivo de imagem preferencialmente é o JPG, GIF ou PNG. O suporte à transparência de formatos como GIF e PNG depende do STB, portanto num primeiro momento deve ser evitada.
sCondition	String	Um texto que representa uma expressão booleana que valida a apresentação ou não do texto na tela de acordo com informações disponíveis da própria aplicação. Essa funcionalidade, compartilhada por outros componentes está descrita em 6.1.8.
sOnKDown	String	Da mesma forma que "sFile", trata-se de um atributo com indicação de um arquivo contendo uma figura. Essa imagem será apenas visível na tela enquanto a tecla cujo código for indicado no atributo "*iKey*" estiver sendo pressionada no controle remoto pelo usuário do sistema.
iKey	String	Um valor inteiro que representa o código da tecla do botão do controle remoto. O **Anexo A** apresenta uma tabela contendo os códigos que podem ser utilizados nesse campo.

Tabela 17 – Atributos do componente "*Image Button*"

d) Observações:

i. A largura e altura da imagem são dadas pela dimensão física da figura indicada no arquivo.
ii. Se os valores para ix e iy corresponderem a uma posição fora da área delimitada pelo componente "*Screen*" o componente não será visível

e) Descrição de um componente "*Image Button*" no arquivo XML:

<**IMGBTN**>
<**iid**>10 </**iid**>
<**ix**> 90 </**ix**>
<**iy**> 425 </**iy**>
<**sFile**> </**sFile**>

<**sCondition**></**sCondition**>
<**sOnKDown**> leftdown.PNG </**sOnKDown**>
<**iKey**> 37 </**iKey**>
</**IMGBTN**>

No exemplo acima, o fato do atributo "sFile" estar vazio, indica nenhuma figura é mostrada enquanto a tecla não estiver pressionada. A figura do arquivo indicado em "*sOnKDown*" é mostrada enquanto a tecla, representada pelo valor 37, no exemplo, estiver pressionada.

VI. Componente "*Questions Group*" (QUEST)

a) Descrição:

O componente "*Questions Group*", também mencionado como "Grupo de Questões", é um dos mais importantes componentes especificados para esse sistema, pois é responsável pela montagem e manipulação das perguntas que compõem o *Quiz*. Entre as suas funções e capacidades:

i. Armazena cada pergunta com suas respectivas alternativas.
ii. Faz o posicionamento da pergunta e das alternativas na tela.
iii. Manipula eventos de navegação entre as alternativas.
iv. Manipula eventos para escolha de uma alternativa.
v. Manipula eventos para troca de perguntas, ou seja, páginas de perguntas.
vi. Insere uma pergunta adicional, após a apresentação da última pergunta, para questionamento do desejo de finalização da interação de escolhas.
vii. Faz a marcação dos elementos considerados corretos.
viii. Manipulação de eventos do usuário para alternativas com texto editável.

Os campos desse componente estão distribuídos em três grupos aninhados, ou seja, um grupo contido dentro do outro:

i. O primeiro grupo diz respeito aos campos que controlam informações genéricas do componente, como por exemplo, posicionamento inicial do texto das perguntas, posicionamento inicial do texto das alternativas, cores das fontes dos textos entre outros que serão listados. Esses campos só podem aparecer uma única vez na descrição do componente.

ii. O segundo grupo é chamado de "*Question*", ou do português "Questão", serve para apresentação de uma pergunta dentro do grupo de questões. Deve haver um grupo "*Question*" para cada questão do *Quiz*. Dentro desse grupo estão inseridos um ou mais campos chamados de "*Alternative*", que representam cada alternativa da questão.

iii. O componente ainda suporta um terceiro grupo, inserido dentro do grupo "*Question*", opcional, e que se for utilizado deve aparecer ao final da relação de alternativas. Trata-se de uma imagem, que exerce função semelhante ao componente já apresentado, Componente "*Image*" , aqui chamado de "*QImage*" e, que representa uma imagem

estática atrelada à questão e, sendo assim, somente visível quando a questão estiver visível.

A Figura 34 ilustra a distribuição dos elementos do componente "*Questions Group*". Cabe ressaltar que o posicionamento dos elementos é definido em configuração e que, da forma como foi planejado, cada conjunto "Pergunta / Alternativas" é apresentado individualmente na tela.

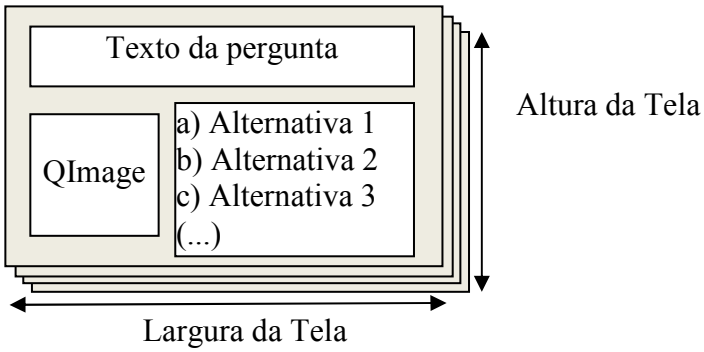

Figura 34 - Distribuição de elementos no componente "*Quest*"

b) Restrições de uso:

Deve haver um único componente tipo "*Questions Group*" no arquivo de configuração. Não há, a princípio, limitações internas quanto ao número de questões e quanto à presença ou não da "*Question Image*".

c) Ilustração:

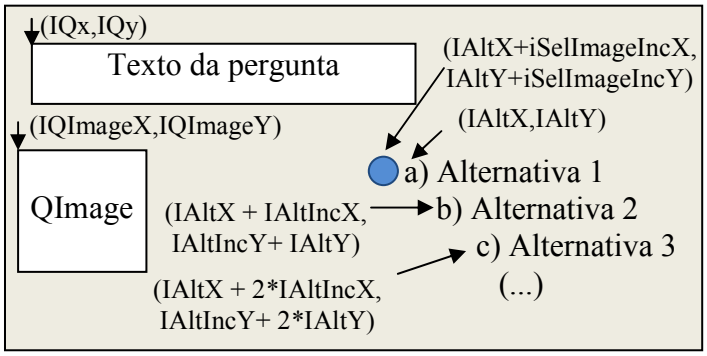

Figura 35 – Ilustração do uso dos campos para posicionamento dos elementos

Ilustrações desse componente estão retratadas nas Figura 21, Figura 22, Figura 23 e Figura 24. A Figura 35 colabora no entendimento do posicionamento (x,y) dos elementos dentro do componente.

A Tabela 18 fornece a relação dos campos desse componente.

Identificador do campo	Tipo do campo	Descrição do campo
Campos genéricos do "*Question Group*" (1)		
iid	Texto	Um identificador numérico para o componente.
iQx	Inteiro	Um valor inteiro para coordenada cartesiana x para inserção da pergunta.
iQy	Inteiro	Um valor inteiro para coordenada cartesiana y para inserção da pergunta.
iQw	Inteiro	Um valor inteiro que representa a largura do texto que será ocupada pela pergunta a partir do ponto (iQx,iQy) em direção ao lado direito da tela.
iQh	Inteiro	Um valor inteiro que representa a altura do texto que será ocupada pela pergunta a partir do ponto (iQx, iQy) em direção à parte inferior da tela.
iAltx	Inteiro	Um valor inteiro para coordenada cartesiana x para inserção da primeira alternativa da questão.
iAlty	Inteiro	Um valor inteiro para coordenada cartesiana y para inserção da primeira alternativa da questão.
iAltw	Inteiro	Um valor inteiro que representa a largura do texto que será ocupada pela primeira alternativa da questão a partir do ponto (iAltx , iAlty) em direção ao lado direito da tela.
iAlth	Inteiro	Um valor inteiro que representa a altura do texto que será ocupada pela pergunta a partir do ponto (iAltx , iAlty) em direção à parte inferior da tela.
iAltIncX	Inteiro	Um valor inteiro para representar qual incremento na direção x deve ser utilizado entre duas alternativas sucessivas da mesma pergunta a serem apresentadas.
iAltIncY	Texto	Um valor inteiro para representar qual incremento na direção x deve ser utilizado entre duas alternativas sucessivas da mesma pergunta a serem apresentadas.

sSelImageIncX	Inteiro	Um valor inteiro positivo ou negativo para indicar o incremento em relação à posição inicial X da alternativa em que a imagem, indicada pelo campo "*sSelIndImage*", será inserida quando o usuário pressionar a tecla "Enter" do controle remoto tendo a intenção de marcar a alternativa como sua resposta à pergunta formulada.
sSelImageIncY	Inteiro	Um valor inteiro positivo ou negativo para indicar o incremento em relação à posição inicial Y da alternativa em que a imagem, indicada pelo campo "*sSelIndImage*", será inserida quando o usuário pressionar a tecla "Enter" do controle remoto tendo a intenção de marcar a alternativa como sua resposta à pergunta formulada.
sQBgcolor	Texto	A representação de uma cor para a caixa de texto da pergunta compreendida dentro da área estabelecida pelos parâmetros iQx,iQw,iQy,iQh.
sQForecolor	Texto	A representação de uma cor para o texto da pergunta.
sAltBgcolor	Texto	A representação de uma cor para a caixa de texto de todas as alternativas da pergunta.
sAltForeColor	Texto	A representação de uma cor para o texto de todas as alternativas que não estiverem selecionadas pela navegação do usuário.
sAltValBgcolor	Texto	A representação de uma cor para ser utilizada na caixa de texto de todas as alternativas que forem indicadas no arquivo de configuração como resposta para a pergunta. Essa cor só será utilizada após a finalização da interação do usuário que ocorre conforme descrito em RF-4.5.
sAltSelColor	Texto	A representação da cor de texto que será utilizada na alternativa para indicar o posicionamento da navegação do usuário conforme descrito em RF-4.1.
sSelIndImage	Texto	Indica o arquivo de uma figura para ser buscado via canal de dados e através do *carousel* do sistema de TV digital para indicar que o usuário fez a escolha por uma determinada alternativa. Esse arquivo deve estar no formato JPG, GIF ou PNG.

sConfEndQuest	Texto	Trata-se da pergunta adicional que será apresentada ao final da exibição de todas as perguntas inseridas no grupo. A pergunta deve questionar se o usuário deseja finalizar o teste. Esse evento serve para indicar o início dos eventos de finalização.
sConfEndYes	Texto	Frase ou palavra da alternativa que confirma a finalização do teste.
sConfEndNo	Texto	Frase ou palavra da alternativa que não confirma a finalização do teste.
Campos do grupo "*Question*" (1+) Parâmetro adicional do indicador do componente: • *multiAnswer="true"* -> Opcional para indicar que a questão possui múltiplas alternativas consideradas corretas o que altera o comportamento da aplicação para essa questão conforme descrito em RF-4.3. Se omitido equivale a *multiAnswer="false"*.		
sText	Texto	Campo obrigatório que se refere ao texto da pergunta ser formulada.
sAlt	Texto	Esse campo deve aparecer 1 ou mais vezes no grupo "Question" representando o texto da alternativa a ser apresentada para a questão. Parâmetros opcionais: • *isAnswer="true"* => Opcional para indicar que a alternativa trata-se de uma resposta para a questão. Se omitido equivale a *isAnswer="false"*. Todas as alternativa marcadas com *isAnswer="true"* serão sinalizadas como corretas ao final da iteração. Cabe ao desenvolvedor do arquivo de configuração estar coerente com o parâmetro da "Question" "multiAnswer". • *select="false"* => Opcional para indicar se a alternativa pode ou não ser selecionada durante a navegação através do controle remoto. Quando a alternativa não é selecionável, ela se torna um texto simples o que é apropriado para indicar o título de uma caixa de texto editável conforme mostrado na Figura 24. Se esse parâmetro for omitido equivale a *select="true"*.

sEdt	Texto	Esse campo deve aparecer 1 ou mais vezes no grupo "*Question*" representando uma caixa de texto editável conforme mostrado na Figura 24. Parâmetro opcional: • *type="numeric"* / *type="alpha"* / *type="pass"* => Se omitido equivale a type =" numeric". Para indicação do tipo de dados de entrada da caixa de texto, respectivamente "numérico", "alpha-numérico", ou tipo "senha". O tipo "senha" equivale ao tipo "numérico", entretanto, os caracteres digitados são substituídos na tela por caracteres curinga como "*".
Campos do grupo "*QImage*" (1)		
sFile	Texto	Indica o caminho completo de um arquivo contendo uma figura para ser buscado via canal de dados, através do *carousel* do sistema de TV digital, ou para ser buscado via canal de retorno. O formato do arquivo de imagem preferencialmente é o JPG, GIF ou PNG. O suporte à transparência de formatos como GIF e PNG depende do STB, portanto num primeiro momento deve ser evitada.
iQImageX	Inteiro	Um valor para coordenada cartesiana x para o componente visual. 0 é o topo esquerdo do monitor TV.
iQImageY	Inteiro	Um valor para coordenada cartesiana y para o componente visual. 0 é o topo esquerdo do monitor da TV.

Tabela 18 – Atributos do componente "*Question Group*"

d) Observações:

i. Da forma como demonstrado na Tabela 18, os campos estão agrupados conforme aparecem no arquivo de configuração, entre valores inteiros e valores textuais a fim de facilitar o algoritmo processo de interpretação do componente.
ii. Todos os elementos gráficos do "*Question Group*" tendem a sobrepor os elementos inseridos via arquivo de configuração como "*Text*", "*Image*" e "*ImageBtn*".
iii. A largura e altura da imagem "*QImage*" são dadas pela dimensão física da figura indicada no arquivo.
iv. Se os valores para iQx e iQy, bem como iAltx e iAltY corresponderem a uma posição fora da área delimitada pelo componente "Screen" o componente não será visível

e) Exemplo de descrição de um arquivo de configuração para um componente "*Question Group*" com 3 perguntas sendo a primeira utilizando caixas de texto para coleta de informações pessoais e as duas subseqüentes com 3 e 4 alternativas respectivamente:

```
<QUEST>
  <iid> 11 </iid>
  <iQx> 120 </iQx>
  <iQy> 23 </iQy>
  <iQw> 380 </iQw>
  <iQh> 18 </iQh>
  <iAltx> 210 </iAltx>
  <iAlty> 160 </iAlty>
  <iAltw> 200 </iAltw>
  <iAlth> 25 </iAlth>
  <iAltIncX> 0 </iAltIncX>
  <iAltIncY> 30 </iAltIncY>
  <iSelImageIncX> -23 </iSelImageIncX>
  <iSelImageIncY> 2 </iSelImageIncY>
  <sQBgcolor>236;236;236;80;</sQBgcolor>
  <sQForecolor>0;20;225;220;</sQForecolor>
  <sAltBgcolor>236;236;236;80;</sAltBgcolor>
  <sAltValBgcolor>0;225;50;100;</sAltValBgcolor>
  <sAltForeColor>0;30;25;120;</sAltForeColor>
  <sAltSelColor>0;20;225;220;</sAltSelColor>
  <sSelIndImage>selected.png</sSelIndImage>
  <sConfEndQuest>Fim das perguntas. O que você deseja fazer?</sConfEndQuest>
  <sConfEndYes>Finalizar</sConfEndYes>
  <sConfEndNo>Revisar respostas</sConfEndNo>
  <Question>
        <sText>Texto da pergunta 1 </sText>
        <sAlt select="false"> Caixa de Texto 1 </sAlt>
        <sEdt type="alpha"> </sEdt>
        <sAlt select="false"> Caixa de Texto 2</sAlt>
        <sEdt type="pass"> </sEdt>
        <QImage>
            <sFile> keypad.png </sFile>
            <iQImageX> 100 </iQImageX>
            <iQImageY> 195 </iQImageY>
        </QImage>
  </Question>
  <Question>
        <sText>Texto da pergunta 2 ?</sText>
        <sAlt> a)  Alternativa 1</sAlt>
        <sAlt> b) Alternativa 2</sAlt>
        <sAlt isAnswer="true"> Alternativa 3</sAlt>
        <QImage>
            <sFile> history.png </sFile>
            <iQImageX> 100 </iQImageX>
            <iQImageY> 175 </iQImageY>
        </QImage>
  </Question>
```

```
<Question>
        <sText> Texto da pergunta 3</sText>
        <sAlt>a) Alternativa 1 </sAlt>
        <sAlt> a) Alternativa 2</sAlt>
        <sAlt> a) Alternativa 3 </sAlt>
        <sAlt isAnswer="true"> a) Alternativa 4</sAlt>
</Question>
</QUEST>
```

6.1.6 O campo "*sInfo*" do componente "*Connection*"

O objetivo desse campo no componente "*Connection*" é gerar uma URL contendo informações para que possam ser interpretadas e armazenadas por um servidor de HTTP externo. Um algoritmo formatador presente no componente verifica o texto fornecido nesse campo e, ao identificar algumas palavras chave, as substitui pelos valores correspondentes.

A versão inicial desse sistema suporta apenas palavras chave indicadas na Tabela 19:

Palavra reservada	Descrição do campo
$d;** **$M; **$A;** **$H;** **$m**	Representam respectivamente o dia, o mês, o ano, a hora e minuto, todos com dois algarismos, configurado no STB no momento do envio da informação para o servidor externo.
$X.Y	X corresponde a um número inteiro indicando o número de uma questão, começando por zero, contida no "*Question Group*" e Y corresponde a um número inteiro indicando o número da alternativa da referida questão X. Assim, $0.3 é substituído pelo texto da alternativa 4 da pergunta número 1 do "*Question Group*". O verdadeiro sentido dessa funcionalidade é buscar a informação fornecida por caixas de texto, cujo posicionamento no componente "Question Group" deve ser conhecido pelo escritor do arquivo de configuração do sistema.
$resp	Essa palavra chave no campo "*sInfo*" é substituída pelas respostas fornecidas pelo usuário do sistema. Tais repostas obedecem a seguinte regra: • As respostas são fornecidas em um único texto seqüencialmente. v. Quando a pergunta está marcada pelo arquivo de configuração como tendo múltiplas respostas, cada resposta fornecida pelo usuário virá entre os caracteres "[" e "]". Verifique o exemplo logo abaixo no texto.

Tabela 19 – Palavras reservadas suportadas no campo "*sInfo*" do componente "*Connection*"

Segue-se um exemplo do uso da reservada $resp. Suponha um sistema com as seguintes características:

- Pergunta 0 é a ilustrada na Figura 24, ou seja, tem 3 caixas de texto, precedidas de alternativas não selecionáveis que servem como título.
- Pergunta 1 tem 1 única resposta e usuário marcou a alternativa "2"
- Pergunta 2 tem múltiplas respostas e usuário marcou as alternativas "1", "3" e "4"
- Pergunta 3 tem 1 única resposta e usuário marcou a alternativa "1"
- O campo "*sInfo*" do componente "*Connection*":

/index.php?alpha=$0.1&num=$0.3&pass=$0.5&$D-$M-$A-$H:$m&resp=$resp

A informação enviada para o servidor externo seria:
/index.php?usr=**Juliano Costa**&num=**12345**&pass=**1234**&**15-09-07-21:20**&resp=**-1[023]0**

Campo $0.1 (Figura 24)	Campo $0.3 (Figura 24)	Campo $0.5 (Figura 24)	D-M-A-H-m Data/hora	Respostas fornecidas.

6.1.7 Campo "*sValue*" do componente "*Text*"

Antes de ser exibido, o campo "*sValue*" do componente "*Text*" passa por uma verificação para substituição de palavras reservadas que representam variáveis do sistema em funcionamento. Nessa versão inicial do sistema, são suportadas as palavras reservadas indicadas na Tabela 20:

Palavra Reservada	Descrição do campo
$QP	*Question Position*: Indica o número da questão sendo exibida na tela partindo do valor zero.
$AP	*Alternative Position:* Indica o número da alternativa atual selecionada pelo usuário começando em zero.
$AC	*Alternative Count:* Indica o número total de alternativas da pergunta corrente.
$QC	*Questions Count:* Indica o número total de questões do grupo
$d;** **$M; **$A;** **$H;** **$m**	Representam respectivamente o dia, o mês, o ano, a hora e minuto, todos com dois algarismos, configurado no STB no momento do envio da informação para o servidor externo.

Tabela 20 – Palavras reservadas suportadas no campo "*sValue*" do componente "*Text*"

6.1.8 O campo "s*Condition*" dos componentes "*Text*", "*Image*" e "*Image Button*"

Conforme já estabelecido, os componentes "*Text*", "*Image*" e "*Image Button*" possuem o campo textual chamado *condition* cujo objetivo é estabelecer uma expressão booleana simples cujo resultado determina se os seus respectivos componentes gráficos estarão visíveis ou não.

As variáveis disponíveis para a operação estão descritas na Tabela 21:

Palavra Reservada	Descrição do campo
$QP	*Question Position*: Indica o número da questão sendo exibida na tela partindo do valor zero.
$AP	*Alternative Position:* Indica o número da alternativa atual selecionada pelo usuário começando em zero.
$AC	*Alternative Count:* Indica o número total de alternativas da pergunta corrente.
$QC	*Questions Count:* Indica o número total de questões do grupo
$IE	*Interaction End: Trata-se de u*ma informação booleana indicando se o usuário já finalizou ou não a interação com o sistema.
$IS	*Information Sent:* Trata-se de uma informação booleana indicando se o sistema já enviou ou não as informações fornecidas pelo usuário para um servidor remoto.
$TS	*Trying Send Info:* Informação booleana que indica que o sistema está tentando realizar uma conexão para enviar informações para um servidor remoto.
$SF	*Sending Failed:* Informação booleana que só é atualizada após tentativa de envio e indica que não foi possível enviar as informações do sistema para o servidor remoto.
$R0	*Response 0:* Trata-se de uma informação booleana indicando que a pergunta corrente não possui alternativas selecionáveis, nesse caso apenas caixa de texto editável.
$R1	*Response 1:* Trata-se de uma informação booleana indicando se a pergunta corrente possui uma única resposta.
$RM	*Response Multiple: Trata-se de u*ma informação booleana indicando se a pergunta corrente possui várias respostas. Naturalmente $RM é igual à NOT $R1.

Tabela 21 – Palavras reservadas suportadas no campo "s*Condition*" de vários componentes

Tendo em vista o compromisso de manter a simplicidade e o baixo consumo de memória e processamento, o número de variáveis, operadores e de operações é limitado. As operações suportadas são *AND* (Operação "E"), *OR* (Operação "Ou"), N*OT* (Operação "Não"), > (Maior), < (Menor), >= (Maior ou igual) e <= (Menor ou igual)

As expressões suportadas podem ter um, dois ou três operandos. Exemplos:

$IE; **$QP=3; $R1** *AND NOT* **$IE; $IE** *AND* **$I*S*** *AND* $R1

Com esse recurso uma imagem pode assumir o mesmo comportamento do grupo "*QImage*" dentro do componente "*Question Group*". Por exemplo, se no campo "*sCondition*" houver "$QP = 2", o componente será visível se a questão corrente for questão 3, tendo em vista que a questão 1 equivale ao número zero.

Observações:

i. Se houver um erro no formato da expressão no campo "*sCondition*" o componente será constantemente visível.
ii. Não há verificação de consistência lógica na expressão.

6.2 Detalhamento da implementação

6.2.1 Ferramenta de desenvolvimento

A tecnologia Java surgiu por volta de 1996 e desde então se firmou como uma das mais utilizadas na atualidade para desenvolvimento dos mais diversos tipos de *software*. Além da SUN, que é a fundadora e mantenedora, várias outras grandes empresas investiram no desenvolvimento de excelentes plataformas de desenvolvimento, chamadas de IDE, baseadas em Java. Apesar disso, a enorme popularização da tecnologia também motivou o aparecimento de IDEs gratuitas e de código livre. Entre estas, a IDE Eclipse [65] destaca-se como uma das ferramentas mais utilizadas devido às facilidades que oferece em relação à flexibilidade e possibilidade de uso de funcionalidades extras gratuitas disponibilizadas através de artefatos de software chamados *plugins*.

O fato de o sistema EMTV utilizar tecnologia Java, principal tecnologia suportada pelo Eclipse, a gratuidade assim como as facilidades que oferece para o desenvolvimento do *software*, tornam a escolha dessa IDE mais do que conveniente para produção do EMTV. Foi utilizada a versão 3.2 Eclipse disponível na época de realização do trabalho.

Durante esse processo foram ainda utilizadas algumas extensões de *software*, chamadas de "plug-ins", fornecidas por terceiros igualmente gratuitamente, que colaboraram positivamente no processo. Entre os "plug-ins" utilizados:

i. Omondo, disponível em http://www.omondo.com: Para geração de diagrama de classes embora o diagrama gerado pelo "*plug-in*" tenha sido manualmente modificado para obtenção da Figura 37.
ii. ArgoUML, disponível em http://argouml.tigris.org/: Para geração de diagramas de classes e de seqüência.

6.2.2 Ferramentas de emulação

Ter a disponibilidade de um sistema completo de transmissão e recepção de TVD durante o processo de desenvolvimento e depuração principalmente no Brasil é algo extremamente raro. Ainda hoje, às vésperas do lançamento oficial do sistema no país, há muitos poucos laboratórios dotados da infra-estrutura adequada. Num momento inicial do desenvolvimento desse projeto, também a UFAM não dispunha dos caros equipamentos necessários e dessa maneira foi necessário recorrer ao uso de emuladores para implementação do EMTV.

Existem muitas ferramentas emuladoras no mercado e, infelizmente as melhores, que são aquelas que simulam com perfeição todo o ambiente de um sistema de TVD, são ferramentas pagas e proprietárias. Abaixo estão relacionadas algumas ferramentas pagas que foram verificadas, entre diversas existentes:

- JameAuthor[59]: Ferramenta para sistemas operacionais Windows, disponível em http://jame.tv/ que permite o desenvolvimento de aplicações de TVD de uma maneira bastante semelhante a esse trabalho. Ela fornece classes de componentes e disponibiliza um aplicativo gráfico que permite a inserção dos componentes na aplicação. A empresa oferece uma versão de testes livres que funciona por até 30 dias após a primeira instalação.

- Osmosys[60]: Trata-se de uma empresa privada, com o endereço eletrônico http://www.osmosys.tv, que oferece toda uma linha de produtos e serviços na área de TVD. Oferecem desde soluções de hardware, pacotes prontos de *middleware*, incluindo diferentes versões de *middleware* GEM compatíveis, ferramentas de depuração e emulação. Os testes realizados com o emulador GEM para sistemas operacionais Windows dessa empresa foram bastante satisfatórios. Baseado em testes realizados pela equipe de pesquisa em TVD do Ceteli, foi verificado que essa ferramenta permite desenvolver qualquer sistema para TVD com a certeza de que o funcionamento será equivalente em um sistema real compatível com GEM.

- Cardinal Systems: Trata-se de uma empresa privada, com o endereço eletrônico http://www.cardinal.fi/ que também fornece um conjunto completo de soluções para sistemas de TVD incluindo *middleware* MHP, ferramentas de autoria com suporte a um pacote extra de componentes para serem utilizados em aplicações de TVD além de um emulador bastante confiável. Além de oferecer uma interface para interpretação de linguagem declarativa proprietária de modo a minimizar a necessidade de programação procedural, o sistema cardinal oferece um método para que programadores experientes possam criar componentes próprios para serem posteriormente utilizados na parte declarativa.

Projetos acadêmicos normalmente fazem uso de ferramentas de emulação gratuitas. Entre essas há duas especialmente interessantes:

- XLetView [61]: Essa é a ferramenta Windows de emulação mais mencionada na maioria dos trabalhos acadêmicos relacionados à sistemas de TVD. Ela está disponível em http://XLetView.sourceforge.net/ . Trata-se de uma boa ferramenta mas que entretanto

não oferece a emulação de todas as classes do GEM e ainda assim algumas implementações não são equivalentes a plataformas reais. A última versão do XLetView é a 0.3.6 que foi lançada em 2004 e desde então não está sendo continuada. Há na referência [35] desse trabalho, a página http://www.interactivetvweb.org/resources/code/index.shtml, que oferece implementações para emulação das classes MHP utilizando o XletView, embora ofereça pouca melhora em relação à equivalência com um sistema real. Apesar disso tudo, essa ferramenta se destaca pela simplicidade de uso, de configuração e de estabilidade além de permitir o desenvolvimento de sistemas simples para TVD e até mesmo de sistemas mais complexos, desde que se tenha ciência de que serão necessários ajustes para o funcionamento em sistemas reais.

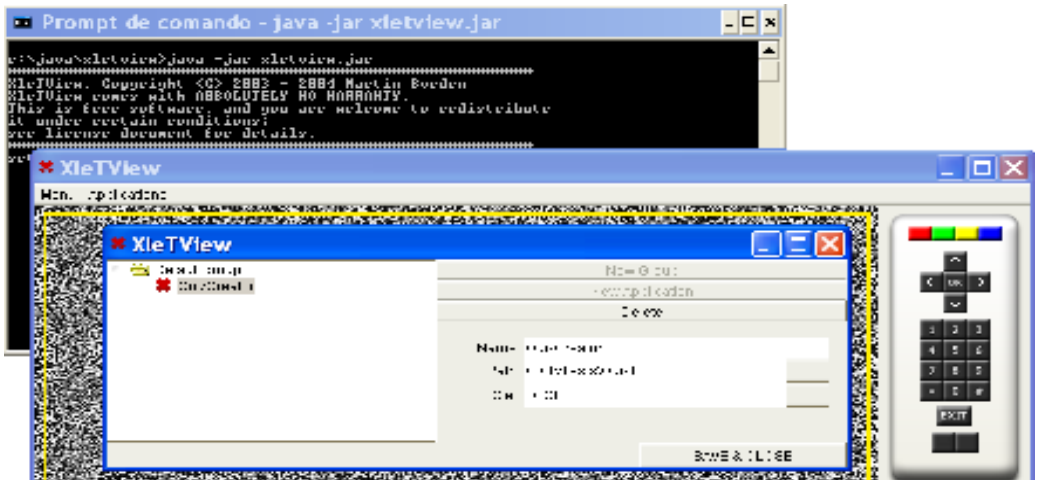

Figura 36 – Ilustração da tela de configuração do XLetView

- OpenMHP: Trata-se de outra ferramenta bastante mencionada em projetos acadêmicos, disponível em http://www.openmhp.org/. Essa plataforma oferece classes compatíveis com a especificação GEM além de uma ferramenta de emulação mantidas por um grupo de empresas e universidades finlandesas. Para esse trabalho tentou-se utilizar essa ferramenta, mas apesar dos esforços seguindo as instruções fornecidas pela documentação fornecida não foi obtido o resultado esperado, certamente devido a alguma falha de configuração ou devido à falta de algum componente não identificado.

Para desenvolvimento desse sistema foi utilizado o emulador XLetView [61], ilustrado na Figura 36, visto a indisponibilidade inicial dos equipamentos necessários e da impossibilidade de adquirir caros emuladores proprietários. A ferramenta OpenMHP não foi utilizada devido ao insucesso nas tentativas de configurá-la para uso.

6.2.3 Diagrama de classes da plataforma proposta

A etapa de levantamento da arquitetura da plataforma EMTV realizado durante esse estudo se encerra com a apresentação de um diagrama, Figura 37, com destaque ao relacionamento entre as principais classes da EMTV.

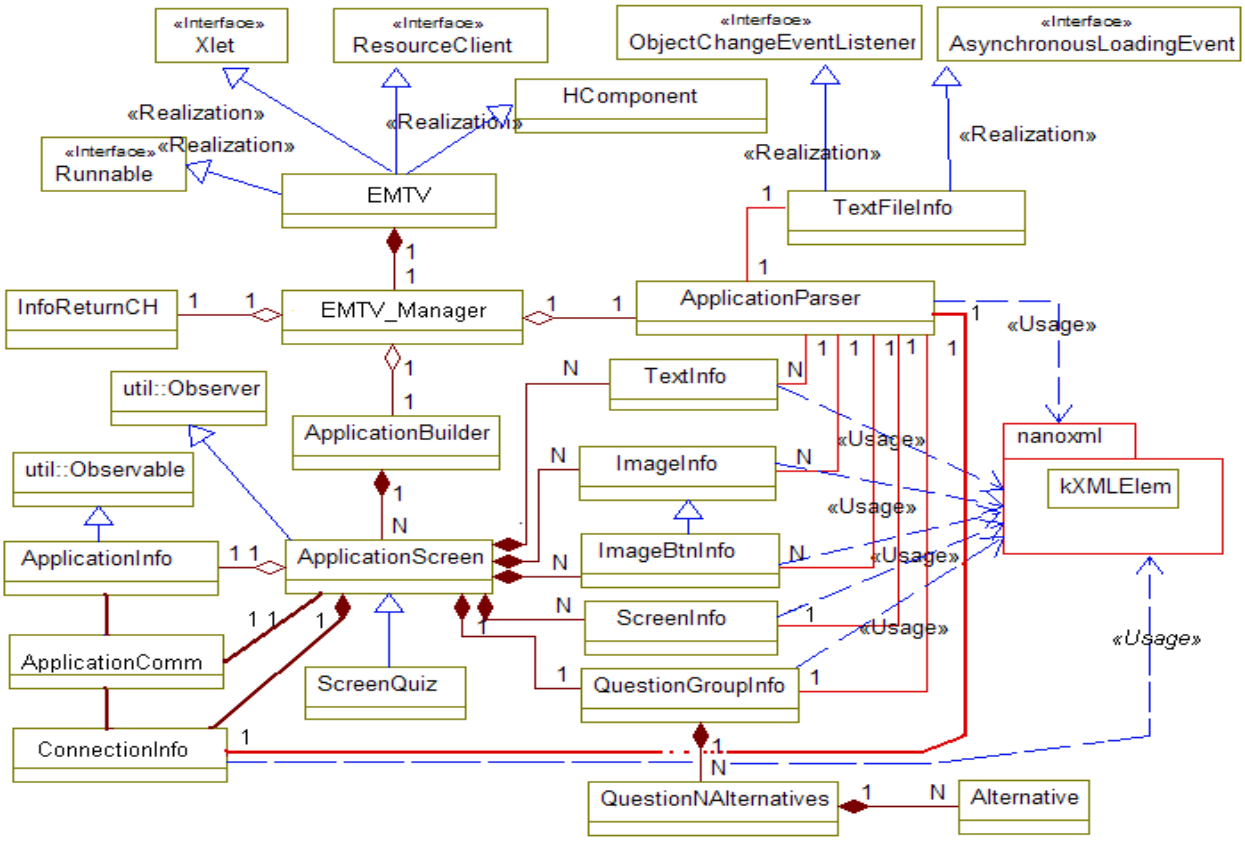

Figura 37 – Diagrama das principais classes do EMTV

O relacionamento entre as classes do diagrama está descrito no tópico 6.2.4 que trata da implementação desse sistema.

6.2.4 Implementação das classes

Segue a descrição da implementação e relacionamento das principais classes do diagrama apresentado na Figura 37:

a) "*EMTV*":

Essa é a classe principal do sistema, e inicialmente implementa métodos da interface *XLet* da JavaTV, como *initXlet*, *startXlet*, *pauseXlet* e *destroyXlet*, que concedem ao STB e indiretamente ao sistema de TVD, o controle do funcionamento da aplicação. A herança direta da classe "*HComponent*" torna essa classe um container de objetos, que serão criados em outras partes do sistema e que poderão ser inseridos na tela e receber os eventos do usuário para serem tratados pelo sistema. A herança da interface da classe "*Runnable*" lhe permite instanciar novos processos, ou "*Threads*" com o propósito separar o tratamento de eventos da interface do processamento do sistema. A herança da interface "Resource Client" proveniente da especificação MHP é necessária para envio de sinalização ao STB de reserva e liberação de recursos utilizados para execução do aplicativo.

b) "*EMTV_Manager*":

Trata-se de uma classe instanciada uma única vez pela aplicação e mantida durante todo o seu ciclo de vida para gerenciamento do processo de construção e manutenção da aplicação. As principais ações realizadas são:

i. Durante a construção da aplicação: Cria um objeto do tipo "ApplicationParser" que fornece uma lista dos objetos contendo informações fornecidas e formatadas no arquivo de configuração. Repassa a lista fornecida como parâmetro para o objeto do tipo "*ApplicationBuilder*" gere a aplicação. Repassa a instancia da lista de objetos visuais para serem inseridos no container "EMTV".
ii. Após a construção da aplicação: Repassa as informações de eventos e requisições do sistema de TVD para tratamento pela aplicação em execução. Inicia o processo de refrescamento dos componentes visuais na tela em resposta à solicitação da "EMTV".

c) "*ApplicationParser"*:

O construtor dessa classe recebe como parâmetro o nome do arquivo de configuração que será buscado via *carousel* ou sistema de arquivos do sistema de TVD. Normalmente a classe "*EMTV_Manager*" se refere ao arquivo de nome "appConfig.xml" que dessa forma como é referenciada deverá estar presente no mesmo local físico em que o sistema foi inicialmente carregado. O processo de carregamento é realizado por uma instancia de "*TextFileInfo*".

O método dessa classe "*ParseFile*" varre o conteúdo do arquivo de configuração e graças à manipulação direta da biblioteca de interpretação XML, nanoXML, identifica os componentes e instancia as classes especializadas na sua interpretação e armazenamento tal como ilustrado na Figura 32. A estratégia dessas classes é a mesma da própria "*ApplicationParser*", ou seja varre o arquivo seqüencialmente usando a biblioteca nanoXML em busca dos parâmetros do componente.

Como resultado o objeto do tipo "*ApplicationParser*" fornece uma lista contendo uma ou mais instancias de classes como "*ScreenInfo*", "*TextInfo*", "*ImageInfo*", "*ImageBtnInfo*" e "*QuestionGroupInfo*".

d) "TextFileInfo*"*:

Essa classe faz o carregamento de arquivos de texto como é o caso do arquivo de configuração. Há duas implementações possíveis:

i. Através da manipulação da classe "DSMCCObject", conforme trecho de código na Figura 38, que faz o carregamento arquivos a partir de requisição direta ao *carousel* de dados do sistema de TVD. Além de permitir o carregamento assíncrono de arquivos, o uso dessa classe da especificação MHP notifica o sistema em caso de alteração do arquivo na origem. Isso é bastante interessante, pois permitiria que mesmo em funcionamento o sistema carregasse novas informações atualizadas. Se a aplicação se tratar de uma votação, o resultado parcial poderia ser constantemente verificado. A desvantagem de se utilizar essa implementação é que a verificação do seu funcionamento fica dependente de um sistema real ou de um emulador, tal qual o Osmosys [60], capaz de reproduzir o comportamento do DSMCC.

```
(…)
DSMCC_File= new DSMCCObject("/appConfig.xml");

DSMCC_File.addObjectChangeEventListener(this);

DSMCC_File.setRetrievalMode(DSMCCObject.FROM_STREAM_ONLY);

if(this.async)
        DSMCC_File.asynchronousLoad(this);
else
        DSMCC_File.synchronousLoad();
(…)
```

Figura 38 – Trecho de código para carregamento de arquivo a partir da classe DSMCCObject

ii. Através do carregamento direto do arquivo através de classes da API Java como "*FileInputStream*" conforme trecho de código na Figura 39. Apesar de não realizar o carregamento assíncrono e de não receber notificações sobre a alteração do arquivo, esse mecanismo é bastante simples de ser implementado e seu funcionamento é o mesmo tanto em ambiente emulado quanto em ambiente real se o carregamento do arquivo se faz através de canal de retorno. Se o arquivo de configuração se encontra no DSMCC esse mecanismo não funcionará.

```
(…)
fileStream = new FileInputStream(DSMCC_File);
(…)
```

Figura 39 – Trecho de código para carregamento de arquivo a partir da classe FileInputStream

e) "ApplicationBuilder*"*:

Trata-se de uma classe de controle gerada a partir da instancia de "*JCQ_Manager*" após a realização da interpretação do arquivo de configuração. De uma maneira mais abrangente essa classe deveria, baseado no conjunto de objetos fornecidos por "*ApplicationParser*", criar o construtor de aplicações apropriado. No caso da versão inicial do sistema em que o objetivo é gerar aplicações tipo *Quiz* só há um construtor disponível, a classe "*ScreenQuiz*" que se trata da especialização da classe "*ApplicationScreen"*.

Versões mais elaboradas do sistema poderiam implementar diferentes tipos de construtores, que poderiam lidar diferentemente com as informações dos componentes fornecidos na lista.

f) "*ApplicationScreen"*:

Trata-se de uma classe abstrata com os métodos básicos necessários para geração dos componentes baseados nas informações contidas na lista fornecida por "*ApplicationParser*". Ela identifica grupos de componentes, separados em grupos de textos e imagens possibilitando o refrescamento controlado dos elementos da tela.

Essa classe é definida como abstrata pois apesar de conter métodos básicos para a criação e manipulação dos objetos gráficos, cabe a uma classe especializada no tipo de aplicação fornecer o comportamento desejado ao conjunto de objetos. Entre os principais métodos fornecidos estão:

i. Método *"SetupScreen"*:
   - Recebe como parâmetro um objeto do tipo "*ScreenInfo*" para geração da tela da aplicação.
   - Delimita o tamanho da tela visível através da manipulação do objeto "*HScene*", descrito em 6.1.3 e gerado na inicialização de "*EMTV*". O trecho de código relevante está ilustrado na Figura 40.

```
(...)
    m_scene.setBounds(scrInfo.getX(),scrInfo.getY(),scrInfo.getWidth(),scrInfo.getHeight());
(...)
```

Figura 40 – Trecho de código para configuração das dimensões da tela

   - Se houver informação sobre uma figura de fundo, será criado um componente do tipo "*ImageInfo*" de tal sorte que a imagem de fundo será criada e manipulada como outra imagem qualquer do sistema utilizando o método "*AddImage"*.

ii. Método "*AddImage*":
   - Possui várias possíveis assinaturas para construção dos componentes "*Image*" e "*ImageBtn*" e os insere na aplicação.
   - Baseia-se no uso das informações contidas em objetos tipo "*ImageInfo*".
   - Se o objeto "*ImageInfo*" passado for do tipo "*ImageInfoBtn"* serão carregadas duas imagens em conformidade com a descrição fornecida no tópico 5.4.
   - Utiliza biblioteca JMF, mais especificamente das classes "*MediaTracker*" e "*ToolKit*" descritas em 6.1.3, para carregamento de imagens conforme trecho da Figura 41.
   - A figura carregada é inserida na tela, e ficará constantemente associada ao elemento "*ImageInfo*" que controla seu comportamento.

```
(…)
MediaTracker mt = new MediaTracker(this);
img = Toolkit.getDefaultToolkit().getImage(fileName);
mt.addImage(img,0);
try
{
  mt.waitForAll();
}
(…)
```

Figura 41 – Trecho de código para carregamento de imagem

iii. Método "*AddText*":
   - Constrói o componente tipo "*Text*", editável ou não e o insere na aplicação.

- Gera a partir de "*TextInfo*" um objeto do tipo "*ApplicationText*" que cuida do comportamento do componente, inclusive dos efeitos definidos no campo "*sEffect*" descrito na Tabela 15, e o campo "*sValue*" mencionado em 0.
- O método "*Build*" da classe "*ApplicationText*" recebe como parâmetro um objeto do tipo "*ApplicationInfo*" gerado e constantemente atualizado pela instancia de "*ApplicationScreen*" e que notifica "*ApplicationText*" sobre as mudanças no estado da aplicação que poderão influenciar no conteúdo do texto e sua visibilidade na tela, graças respectivamente às ações dos métodos "*FormatText*" e "*GetCheckCondition"*.
- O objeto gráfico central de "*ApplicationText*" é um objeto do "*HText*" gerado a partir de especificações em "*TextInfo*" conforme trecho de código indicado na Figura 42.

iv. Método "*SetupConnectionInfo*":
- Associa à aplicação as informações de conexão através do objeto tipo "*ConnectionInfo*".

```
(…)
this.textVisible = this.txtInfo.GetCheckCondition(appInfo);
this.text =  new HText(  this.FormatText(this.txtInfo.GetText()),
                         tInfo.GetX(), tInfo.GetY(), tInfo.GetWidth(), tInfo.GetHeight(),//h
                         txtFont, tInfo.GetForeColor(),tInfo.GetBackColor(),layout);

this.text.setVisible(textVisible);
(…)
```

Figura 42 – Trecho de código que gera e formata o objeto "*HText*" para o componente "*Text*"

g) "*ScreenQuiz"*:

Trata-se de uma classe filha da classe "*ApplicationScreen"* e faz a implementação do comportamento do sistema para configuração de um *Quiz* conforme já estabelecido em itens anteriores. Realiza as seguintes ações:

i. Implementa métodos específicos do comportamento de aplicações tipo *Quiz* como é o caso do componente "*QuestionGroup*".
ii. Identifica os componentes presentes na lista gerada pelo "*ApplicationParser*" e faz as chamadas adequadas para inserção do componente na aplicação.
iii. Faz o tratamento adequado para o pressionamento de teclas do controle remoto:
- Teclas seta esquerda (←) e seta direita (→) são utilizadas para troca de perguntas do grupo de questões.
- Teclas seta acima (↑) e seta abaixo (↓), quando a interatividade não foi finalizada, são utilizadas para troca de alternativa selecionada.
- Teclas "0" até "9" são repassadas para uma caixa de edição se estiver selecionada. Ainda nesse caso, a tecla de função "Vermelha", comumente presente em controles remotos de televisores de TVD, é utilizada para apagar caracteres.
- Tecla "Exit" para fechamento imediato do programa.

h) "*ApplicationComm*"

A classe "*Application Communication*s" é a responsável pelo comportamento do componente "*Connection*" apresentado em 6.1.5. Logo no seu construtor, recebe uma referência do objeto do tipo "*ApplicationInfo*". Durante a atuação dessa classe há a atualização constante das variáveis $TS, $IS e $SF de "*ApplicationInfo*" descritas na Tabela 21. Os principais métodos dessa classe são:

i. Método "*StartSendInfo*":
   - Esse método é invocado diretamente pela classe "*ScreenQuiz*" que passa como parâmetros a referência ao objeto do tipo "*ConnectionInfo*", além da referência para a lista de questões, que é uma instancia de "*ApplicationGroupQuestions"*. A primeira referência serve para o estabelecimento da conexão com um servidor externo. A segunda referência serve para, juntamente com a referência de "*ApplicationInfo*", fornecida no construtor, fornecer as informações necessárias para montagem da mensagem a ser enviada, seguindo a especificação definida em 6.1.6.
   - Esse método inicia um novo processo ou "*Thread*" para evitar qualquer bloqueio da interface gráfica durante o envio das informações.

ii. Método "*InitInterface*":
   - Esse método, acessa a classe "*RCInterface*" definida na biblioteca DVB e que fornece uma lista das interfaces físicas disponíveis no STB que permitem a transmissão pacotes IP. Nos STB da atualidade, conforme já adiantado em 2.5, podem aparecer dois tipos de interface:
     - Interface V.90, que através de um conector do tipo RJ-11 pode ser conectada à uma linha telefônica convencional sendo assim capaz de estabelecer uma conexão TCP/IP discada.
     - Interface Ethernet, que através de um cabo tipo RJ-45 disponibiliza um link direto com uma rede TCP/IP externa, normalmente fornecida por conexão a cabo ou por rádio.

```
/**< Função que verifica a disponibilidade de uma interface TCP/IP permanente disponível no STB*/
public void initInterface(){
    if(rcm!=null)                    //rcm é um objeto estático fornecido pela biblioteca DVB
    {
        RCInterface[] interfaces = rcm.getInterfaces();    //Fornece todas as interfaces de conexão disponíveis no STB

        if((interfaces!=null))
        {
            if(interfaces.length>0)
            {
                this.rcInterface = (ConnectionRCInterface)interfaces[0];
            }
        }
    }
}
```

Figura 43 – Função que verifica a disponibilidade de interface TCP/IP permanente no STB

- No Brasil, já há sugestões de empresas de telecomunicações para oferecer conexões sem fio, também conhecidas como Wireless. Ainda nesse caso, o comportamento com o sistema seria o mesmo de uma interface Ethernet.
- O papel desse método, mostrado na Figura 43, é simplesmente fornecer para a classe "*ApplicationComm"* a primeira interface disponível, que se presente, por convenção deve ser do tipo Ethernet.

iii. Método "*GetIsConnected*":

- Retorna uma variável booleana para indicar se, a "RCInterface", associada a "*ApplicationComm*", já possui ou não link de uma conexão TCP/IP ativa ou não.

iv. Método "*EstabilishDialConnection*":

- Se após a execução de "*InitInterface*", o método "*GetIsConnected*" retornar "*false*", então o método "EstabilishDialConnection" será utilizado na tentativa de estabelecer uma conexão TCP/IP utilizando uma possível interface V-92 se estiver disponível no STB.
- Se a interface estiver presente, serão utilizadas as informações do subgrupo "*Dial*", fornecido no arquivo de configuração conforme Tabela 13, para solicitar a discagem para um servidor de conexões TCP/IP via linha discada.
- Se a discagem for bem sucedida, fornece as informações de usuário e senha para autenticação no servidor de conexões.
- Ao executar com sucesso as instruções acima, a conexão TCP/IP está disponível e o método "*GetIsConnected*" retornará verdadeiro.
- Todo o processo de sinalização, e discagem e sincronização é realizado pelo próprio sistema operacional do STB sob intermédio do *middleware*.
- A Figura 44 apresenta as principais chamadas desse método.

```
public boolean EstablishDialConnection(ConnectionInfo cInfo) {
(...)
if (rcInterface instanceof ConnectionRCInterface)
{
    try {  rcInterface.reserve(this,null); } catch (PermissionDeniedException e) { return false; }

    if (!rcInterface.isConnected()){
            try
            {
                try { rcInterface.setTarget(new ConnectionParameters(cInfo.dialPhone,cInfo.dialUser,cInfo.dialPass)); }
                catch (IncompleteTargetException e) {return false; }

                try { rcInterface.connect(); } catch (java.io.IOException e) {return false; }

                try { synchronized (connectionWait) {connectionWait.wait();}} catch (Throwable ex) {return false; }
             }
            catch (PermissionDeniedException e) {return false; }
}
(...)
```

Figura 44 – Trecho de função que tenta estabelecer uma conexão "*dial-up*"

i) "*ApplicationGroupQuestions*"

É uma classe que faz a implementação do componente "*QuestionGroup*" descrito em 6.1.5. No construtor também recebe a referência do objeto "*ApplicationInfo*" da aplicação.

i. Método "*Build*":
   a) Recebe como parâmetro um objeto tipo "*QuestionGroupInfo*" contendo as informações mencionadas na Tabela 18 e extraídas do arquivo de configuração.
   b) A Figura 45 traz um trecho do código que cria o texto da questão que utiliza a classe "HStaticText", proveniente da biblioteca HAVI.

```
(…)
if(this.qgInfo.questionsList.size()>0)
{
        QuestionNAlternatives questionAlts =((QuestionNAlternatives)gInfo.questionsList.get(this.page)). GetQuestion();
        //Texto da pergunta
        question =  new HStaticText(questionAlts, gInfo.GetQX(), gInfo.GetQY(), gInfo.GetQWidth(), gInfo.GetQHeight()
                                    font,  gInfo.GetQForeColor(), gInfo.GetQBgColor(), layout);
        this.question.setVisible(true);
        this.question.setHorizontalAlignment(HText.HALIGN_LEFT);
        this.question.setVerticalAlignment(HText.VALIGN_CENTER);
        this.hcontainer.add(this.question);

        for(int i=0;i< questionAlts.GetAlternativeCount();  i++)
        {
            AddNewAltText(i); //faz a inserção das alternativas.
        }
        this.SetSelectUpDownAlternative(false); //Faz a escolha da primeira alternativa selecionável.
}
(…)
```

Figura 45 – Trecho de código que gera o elemento de texto para as questões

ii. Método "*AddNewText*":
   - Esse método é invocado pelo método "*Build*", tal como aparece na Figura 45, e tem o papel de criar o texto representa a alternativa da questão.
   - Ainda na Tabela 18, há o esclarecimento que a alternativa ser sinalizada com algumas propriedades que mudam seu comportamento na tela:
     - Em se tratando de uma alternativa comum, será criado um texto não editável e baseado na classe "*HText*" fornecida pela biblioteca HAVI.

```
(…)
QuestionNAlternatives  questionAlts = ((QuestionNAlternatives)this.qgInfo.questionsList.get(this.page));
if(qa.GetIsEditable(pos)) // TESTA SE trata-se de campo editável
{    alts = (HVisible) new HSinglelineEntry(qa.GetAlternative(pos),  //Trata-se de campo editável
                                            this.qgInfo.GetAltX()+pos*this.qgInfo.GetAltIncX(),
                                            this.qgInfo.GetAltY()+pos*this.qgInfo.GetAltIncY(),
                                            this.qgInfo.GetAltWidth(), this.qgInfo.GetAltHeight(),
                                            questionAlts.GetMaxSize(pos),  new Font("Tiresias",Font.BOLD,20),
                                            this.qgInfo.GetAltForeColor());
        alts.setBackground(this.qgInfo.GetAltBgColor());
         ((HSinglelineEntry)alts).setEnabled(true);
         ((HSinglelineEntry)alts).setBordersEnabled(true);
         ((HSinglelineEntry)alts).setMaxChars(qa.GetMaxSize(pos));
         if(qa.GetAltEdtType(pos) == EditType.PASSWORD)
           ((HSinglelineEntry)alts).setEchoChar('X'); //Se é um campo editável para password, caracteres serão mascarados
}else{ //Trata-se de um texto regular
        alts = new HText(qa.GetAlternative(pos),
                         this.qgInfo.GetAltX()+pos*this.qgInfo.GetAltIncX(),
                         this.qgInfo.GetAltY()+pos*this.qgInfo.GetAltIncY(),
                         this.qgInfo.GetAltWidth(), this.qgInfo.GetAltHeight(),//h
                         new Font("Tiresias",Font.BOLD,20), this.qgInfo.GetAltForeColor(),
                         this.qgInfo.GetAltBgColor(),
                         this.layoutAlt);
}(…)
```

Figura 46 – Trecho de código que gera a pergunta e as alternativas na tela

- Em se tratando de uma alternativa editável, será criada uma caixa de texto, baseada na classe "HSinglelineEntry" também proveniente da HAVI. Nesse caso, ainda são verificadas propriedades para checar o tipo de entrada da caixa. Se por exemplo, se tratar de uma entrada para senha, as teclas digitadas serão convertidas para o caractere "X". Um trecho desse método é mostrado na Figura 46.

A classe "*ApplicationGroupQuestions*" oferece ainda os seguintes métodos:

iii. Método "*GetAlternativeCount*":
- Fornece a quantidade de alternativas da questão.

iv. Método "*GetAlternativeText*":
- Fornece o texto, em forma de "*string*", de uma determinada posição de alternativa.

v. Método "*GetHasEndQuestion*":
- Retorna um valor booleano que indica se o arquivo de configuração forneceu a pergunta de finalização de interação com o usuário.

vi. Método "*GetPageCount*":
- Retorna, através de um número inteiro, o número de questões, ou seja páginas de questões, fornecidas através do arquivo de configuração.

vii. Método "*GetCurrentPageNumber*":
- Retorna, através de um número inteiro, a posição corrente da pergunta sendo visualizada em relação ao restante de perguntas e desconsiderando a pergunta de finalização se houver.

viii. Método "*FinishCurrentQuiz*":
- Método sem retorno que é utilizado para finalizar a interação do *Quiz* com o usuário.
- Esse método é invocado a partir da classe "*ScreenQuiz*" em resposta à resposta seleção da alternativa que representa sim à pergunta de finalização de interação.

ix. Método "*ChangePage*":
- Método utilizado para alterar a pergunta, ou página corrente que está sendo apresentada na tela.
- Faz a verificação se o número de alternativas da próxima questão é menor ou maior do que a anterior para verificar a necessidade de criar novos textos de alternativas ou esconder outras. Isso evita a destruição desnecessária de textos que poderão ser reaproveitados numa outra página.
- Se a nova página será a posterior ou anterior em relação à atual, dependerá da tecla pressionada pelo usuário e repassada pela classe "*ScreenQuiz"*.
- As escolhas realizadas pelo usuário em interações anteriores são mantidas nas trocas de páginas.

x. Método "*GetQuestionIsMultiSelect*":
   - Retorna variável booleana para indicar se a página sendo mostrada no momento corresponde a uma pergunta de múltipla escolha.

xi. Método "*GetCurrentQuestionChosenIndexs*":
   - Fornece um vetor com os valores inteiros correspondentes aos números das alternativas que foram escolhidas pelo usuário.
   - A primeira alternativa corresponde ao valor "0".
   - Naturalmente se a questão não é de múltipla escolha, haverá na lista apenas um elemento.

xii. Método "*SetSelectAlternative*":
   - Trata-se da ação de navegação do usuário a partir do controle remoto.
   - Incrementa ou decrementa o número da alternativa selecionada.
   - Substitui a cor da nova e da antiga alternativa selecionada conforme definido em 4.4.2.
   - Durante a seleção de alternativas, descarta aquelas marcadas como não selecionáveis a partir do arquivo de configuração.

xiii. Método "*SetChoosenCurrentSelectedAlternative*":
   - Marca a seleção da alternativa corrente como resposta do usuário à pergunta formulada seguindo especificação em 4.4.2.

xiv. Método "*GetDemandingKeys":*
   - Fornece indicação se a alternativa selecionada no momento trata-se de uma caixa de texto editável.

xv. Método "*InformKeyInput":*
   - Fornece tecla passada pela classe "*ScreenQuiz*" para processamento de uma caixa de texto editável.
   - Faz a análise do tipo de dado da caixa de texto

i) "*QuestionNAlternatives*"

Essa classe agrega o controle sobre o conjunto formado por 1 pergunta mais suas respectivas alternativas. A instancia da classe "*ApplicationGroupQuestions"* possui uma lista com um conjunto de objetos "*QuestionNAlternatives".* Por sua vez a classe "*QuestionNAlternatives"* é constituída por uma lista de instancias de "*Alternative*" que se trata de uma classe privada de "*QuestionNAlternatives"* e representa naturalmente uma única alternativa. O relacionamento de agregação entre as classes é mostrado na Figura 47.

Desse modo essa classe agrega informações que dizem respeito apenas ao conjunto, como por exemplo, a alternativa que está selecionada, as alternativas que foram escolhidas entre outras. Por essa razão boa parte das responsabilidades da classe "*ApplicationGroupQuestions"* são resultado de manipulação direta de propriedades da "*QuestionNAlternatives"*.

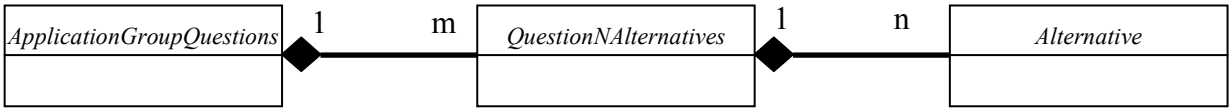

Figura 47 – Relacionamento da classe "*QuestionNAlternatives*"

j) "*EditSingleLineText*":

Essa classe contra o comportamento da alternativa editável, isto é, a alternativa que no arquivo de configuração foi indicada com o marcador "<*sEdt*>". Ela instancia um novo processo que cuida do posicionamento do cursor.

i. Método "*SelectKey":*
   - Recebe as teclas pressionadas pelo usuário e repassadas por "*ScreenQuiz*" quando o método "*GetEditable*" da classe "*Alternative"* retorna verdadeiro.
   - Faz uma verificação do tipo de campo da caixa de texto:
     - Se o campo é numérico ou do tipo senha cada tecla do controle remoto que possui apenas números é repassada para a caixa de texto.
     - Se o campo é do tipo senha cada caractere digitado é substituído na tela pelo caractere 'X', entretanto o seu conteúdo é preservado.
     - Se o campo é do tipo alfa-numérico, cada tecla de 2 a 9 possui de 3 a 4 caracteres associados, seguindo um padrão internacional utilizado em qualquer teclado numérico desde telefones a controles remotos. O recebimento de um mesmo caractere num intervalo de tempo menor do que 1,8 segundos, resulta na alternância do caractere visualizado. Se duas teclas diferentes forem pressionadas seguidamente, mesmo que num intervalo menor do que 1,8 segundos, o primeiro caractere de cada tecla será mostrado. A Figura 24 tem uma ilustração de um teclado numérico em que cada tecla possui grafados os caracteres que representa além do número propriamente dito.

6.3 Considerações finais

Esse capítulo focou os aspectos técnicos envolvidos na definição da arquitetura e implementação da plataforma EMTV. Começando pela arquitetura em que foram definidas as soluções tecnológicas para as caracteristicas desejadas apresentadas no capítulo 5, por exemplo, o levantamento dos principais processos internos da plataforma e as bibliotecas de funções utilizadas. Chega-se à importante decisão de utilizar o formato XML no arquivo de configuração que muito contribui para a identificação dos componentes e para o processo de validação. Foram apresentados os campos utilizados para descrição dos componentes no arquivo de configuração. Finalmente foram apresentados aspectos de implementação importantes como referência da solução. O próximo capítulo abordará as experiências realizadas para teste e validação da plataforma.

7 Execução e Experiências

Esse capítulo aborda questões relacionadas ao uso do sistema desenvolvido bem como as formas de validação do seu funcionamento.

7.1 Validação do arquivo de configuração

Conforme abordado no tópico 6.1.4, o DTD é um documento auxiliar ao XML propriamente dito, e que se fornecido para um interpretador de XML o habilita a realizar a validação de que o arquivo XML está escrito seguindo as regras pré-definidas.

No caso da biblioteca interpretadora de XML utilizada nesse projeto, o nanoXML, a funcionalidade de validação do arquivo de configuração diante de um DTD não é realizada para minimizar a quantidade de processamento e de código fonte que são requisitos prioritários. Ainda assim é importante no escopo desse projeto desenvolver um arquivo de DTD para o arquivo de configuração permitindo que o arquivo seja validado antes mesmo de ser inserido no sistema de TVD.

O arquivo DTD desenvolvido está apresentado no **Anexo B**.

7.2 Escrevendo arquivos de configuração para o sistema

Arquivos XML, tais como o arquivo de configuração desse projeto, têm a grande vantagem de serem facilmente legíveis por seres humanos. Como se trata de um formato do tipo texto, permite ser confeccionado por qualquer editor de texto desde que seja salvo sem nenhuma formatação, isto é, seja salvo no formato somente texto.

Existem dezenas de editores de texto entre pagos e gratuitos. Alguns desses são bastante indicados para programação, entre eles, o JFE [63] que foi utilizado nesse trabalho para edição do arquivo de configuração por dois motivos:

ii. Possibilidade de configurar palavras chave de modo que possam ter suas cores alteradas, ou seja "*syntax highlighting*". Isso permite que diferentes palavras chave para parâmetros e definição de componentes podem assumir diferentes cores e tipos de fonte, facilitando bastante a leitura e entendimento do arquivo pelos desenvolvedores.

iii. Possibilidade de registrar botões adicionais para funções de macro. Foram desenvolvidas para esse projeto macros para cada componente suportada pelo sistema de modo que ao serem pressionadas, colam no texto todos os elementos constituintes de um componente, evitando a necessidade de consulta a esse texto ou a modelos. Foram geradas macros para o JFE para todos os componentes suportados no sistema.

O **Anexo D** fornece o trecho do arquivo de configuração do JFE que define o destaque das palavras chave e também o conteúdo das macros para os componentes. A Figura 48 mostra a interface principal do Jen's editor, como o "*highlighting*" contribui para legibilidade do texto e os botões de macro inseridos.

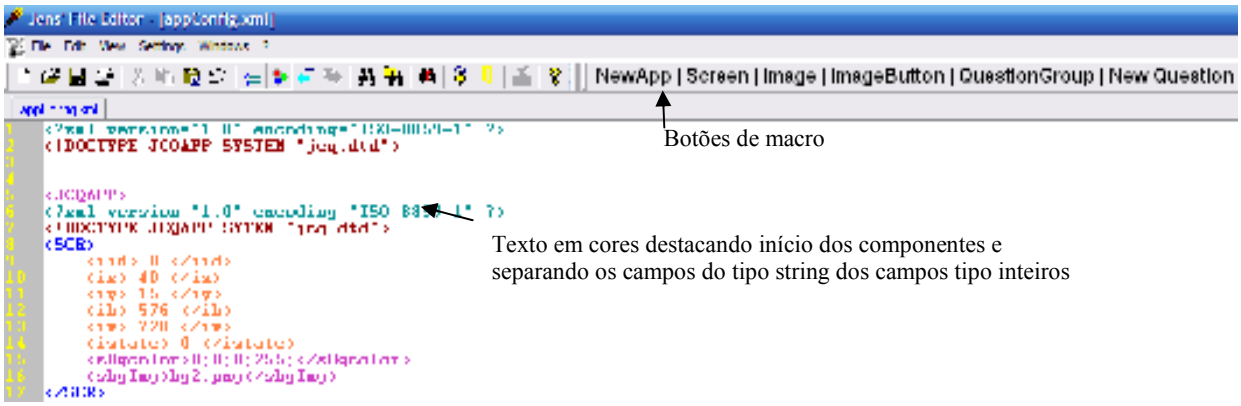

Figura 48 – Interface do editor JFE utilizado para edição do arquivo de configuração

7.3 A estrutura do arquivo de configuração

Conforme foi esclarecido em vários pontos do texto, a plataforma é baseada em componentes. Alguns deles são obrigatórios e só podem aparecer uma vez no arquivo de configuração, outros podem aparecer nenhuma ou várias vezes. A Figura 49 ilustra a estrutura possível para um arquivo de configuração. O Anexo C apresenta um exemplo de um arquivo de configuração de um *Quiz* utilizado como demonstração.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>          <!—Cabeçalho versão do XML -->
<!DOCTYPE JCQAPP SYSTEM "jcq.dtd">                    <!—Cabeçalho definição do DTD -->

<JCQAPP>                                              <!—TAG de abertura da configuração -->
  <CONN> </CONN>          <!— Componente “Connection” 0 ou 1 vezes -->
  <SCR> </SCR>            <!— Componente “Screen” 1 vez obrigatório -->
  <TEXT> </TEXT>          <!— Componente “Text” 0 ou M vezes-->
  <IMG> </IMG>            <!— Componente “Image” 0 ou N vezes-->
  <IMGBTN> </IMGBTN>      <!— Componente “Image Button” 0 ou P vezes-->
  <QUEST>                 <!— Componente “Question Group” 1 vez obrigatório -->
      <Question>…</Question>      <!— Componente “Question” 1 até R vezes -->
  </QUEST>

</JCQAPP>                 <!—TAG de fechamento da configuração -->
```

Figura 49 – Estrutura do arquivo de configuração

7.4 Resultado obtido com o XLetView:

Conforme apresentado na seção 6.2.2, o XLetView [61] é um emulador MHP gratuito que foi utilizada durante a fase de desenvolvimento do sistem. Foram realizadas as configurações necessárias, conforme já ilustrado na Figura 36 para que a EMTV pudesse ser executada através do emulador permitindo testar e validar as funcionalidades conforme descritas na seção 5.2. A

seguir, são apresentadas algumas ilustrações da plataforma EMTV sendo validada com o arquivo de configuração registrado no anexo C:

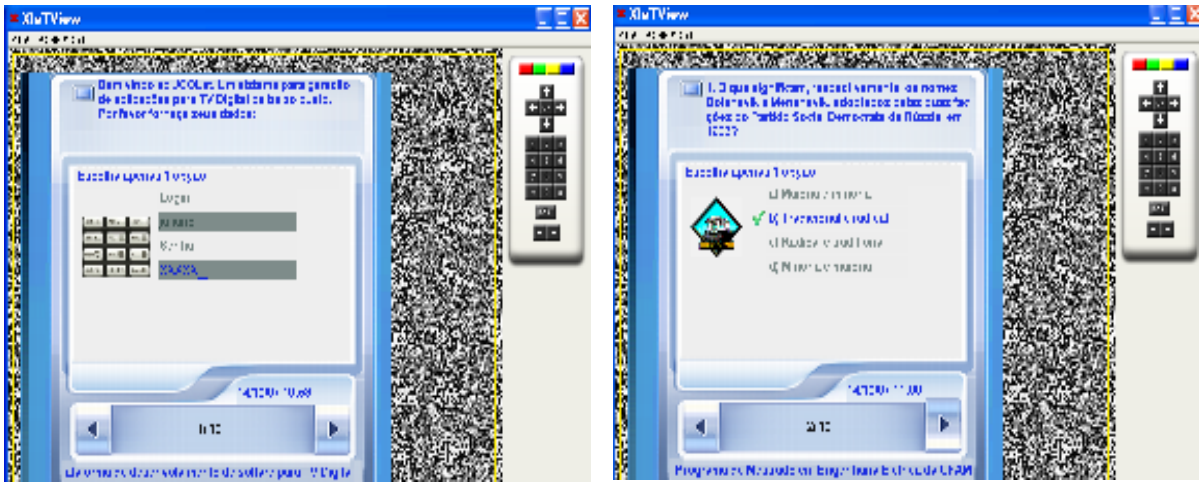

Figura 50 – Experiência no XLetView**. A) Tela com campos edição B) Pergunta de 1 única resposta**

À esquerda da Figura 50 está ilustrada a tela gerada pela EMTV no XLetView de uma tela com campos editáveis para aquisição de informações de usuário e senha. Nesse caso a pergunta foi convertida em uma mensagem de saudação e, a primeira e terceira alternativas foram convertidas em títulos respectivamente para a segunda e quarta alternativas editáveis. À direita da Figura 50 está ilustrada uma pergunta de uma única resposta correta em que o usuário selecionou e escolheu a alternativa "b".

Figura 51 – Experiência no XLetView**. A) Pergunta de múltiplas respostas B) Pergunta de finalização**

À direita da Figura 51está ilustrada uma pergunta que foi marcada no arquivo de configuração como pergunta de múltiplas respostas. Nessa figura o usuário escolheu como resposta as alternativas "a", "b" e "d", sendo que esta última alternativa encontrava-se selecionada. À esquerda da Figura 51está ilustrada a pergunta de finalização de interação do componente "*Quiz Group*".

Figura 52 – Experiência no XLetView**. A) Destaque à alternativa correta B) Destaque às múltiplas alternativas corretas**

A Figura 52 demonstra a aplicação destacando as alternativas marcadas no arquivo de configuração como corretas, após a finalização de interação do componente "*Quiz Group*".

7.5 Resultado obtido no laboratório da UFAM:

A segunda etapa de validação de funcionamento da plataforma foi sua execução em um sistema real de TVD disponível no laboratório de TVD do Centro de Pesquisa e Desenvolvimento em Tecnologia Eletrônica e da Informação, ou CETELI que está vinculado à UFAM. Nesse laboratório estão disponíveis modernos equipamentos que podem ser utilizados para realização de diversos testes, inclusive a transmissão de um sinal de TVD em DVB ou ATSC via cabo ou via espaço livre, além de um equipamento para geração do sinal cíclico proveniente de carrossel de dados que pode ser inserido no canal de dados.

Entretanto nesse trabalho foram utilizados apenas um monitor de TVD de alta resolução e um STB provido do *middleware* MHP de nível interativo, ou interactive profile conforme abordado no tópico 3.3 e ilustrado na Figura 12. Através desse STB é possível realizar o carregamento de XLets diretamente através do canal de retorno devidamente conectado a uma rede TCP/IP. A conexão entre os equipamentos utilizados para teste é ilustrada na Figura 53 que representa:

- O STB do fabricante ADB provido do middleware MHP e *Interactive Profile* possui um conector RJ-45 que é utilizado para conexão com uma rede TCP/IP permanente e que corresponde ao canal de retorno.
- Na mesma rede TCP/IP está conectado um computador onde está instalado um servidor de aplicações. No caso desse teste foi utilizado um servidor Apache versão 2.0.59 que pode ser encontrado no site www.apache.org além de um interpretador de scripts PHP versão 4.0.6 disponível em www.php.net.
- Um usuário através de um programa embutido no próprio STB pode requisitar ao servidor Apache o envio do XLet EMTV. De acordo com os procedimentos do fabricante do STB, isso é realizado através da solicitação de um arquivo texto e que contém todas as informações necessárias para que em seguida o Apache

forneça o EMTV.jar, que é o XLet empacotado para execução no STB e visualização no monitor HDTV.

- O interpretador PHP é apenas utilizado no mesmo servidor para execução de script que registra as eventuais respostas coletadas no *Quiz*.
- O monitor utilizado é de fabricação da SAMSUNG.

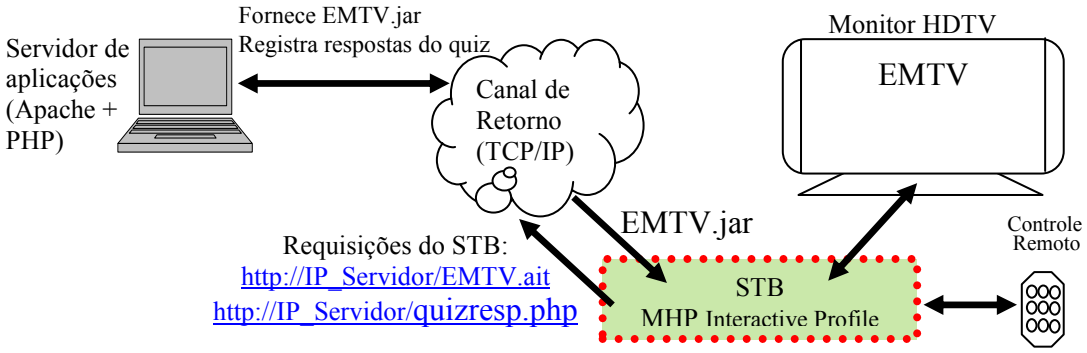

Figura 53 – Experiência utilizando um STB real e um servidor de aplicações

O mesmo resultado seria obtido se o carregamento do EMTV ocorresse diretamente através do canal de dados do sistema de TVD. Esse teste só não foi realizado, pois para tanto seria necessário que o laboratório possuísse um software adequado para geração de tabelas necessárias para o funcionamento do carroussel.

A seguir são mostradas algumas fotografias tiradas no laboratório da CETELI durante a execução dos testes com o EMTV.

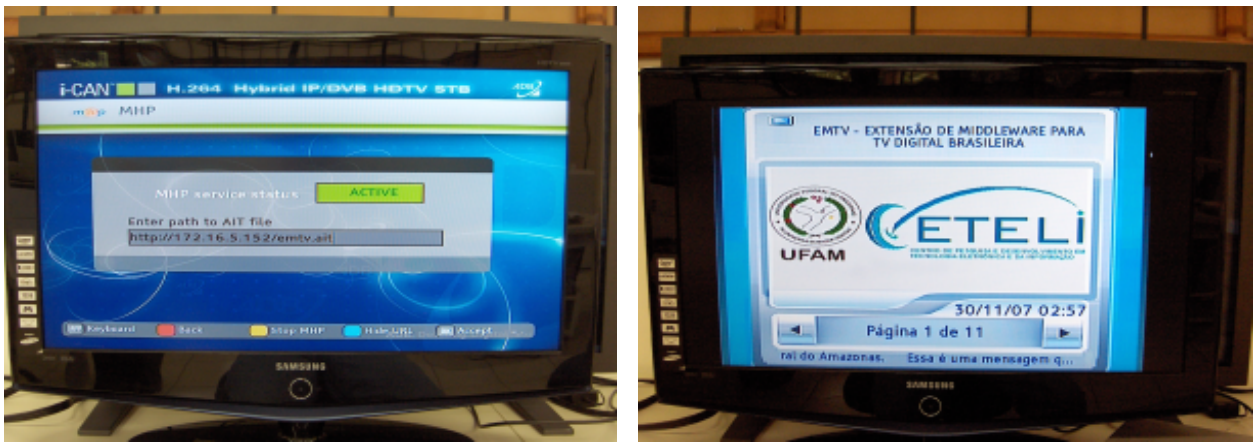

Figura 54 – A) A seleção do XLet através do STB ADB B) Aplicação de teste utilizando EMTV

Na parte esquerda da Figura 54 mostra a tela de configuração do STB da marca ADB para carregamento da EMTV a partir de um servidor Apache externo conectado na mesma rede do STB. À direita da Figura 54 está a tela inicial do aplicativo gerado com a EMTV a partir do arquivo de configuração do Anexo C. Essas ilustrações mostram que a plataforma foi carregada corretamente e que foi capaz de gerar a aplicação esperada.

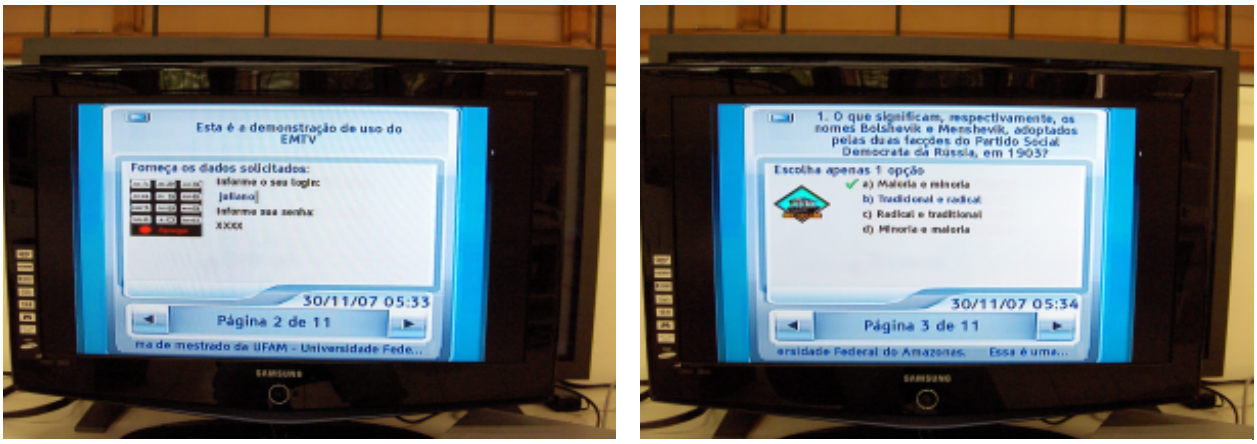

Figura 55 – A) Tela de aplicação com caixa de entrada B) Tela de pergunta selecionada e 1 resposta

A Figura 55 ilustra os testes com uma tela com caixas de texto editáveis à esquerda e a navegação e escolha de uma questão com uma única resposta na parte direita da figura.

Figura 56 – A) Tela de pergunta de múltiplas respostas B) Tela após finalização de interatividade enviando dados pelo canal de retorno

A Figura 56 mostra à esquerda testes com uma questão com múltiplas respostas e à direita a tela apresentada durante o processo de envio das informações coletadas através do canal de retorno. Essa tela à direita aparece imediatamente após a finalização de interação.

7.6 Comparação com trabalhos semelhantes

Realizar a comparação desse trabalho com outros é uma tarefa bastante complicada. O principal motivo é que apesar de existirem diversos sistemas proprietários voltados para o desenvolvimento de aplicações para TV, os consumidores desse tipo de sistemas são normalmente constituídos de um público bastante restrito: Produtores de conteúdo de televisão. Isso faz com que além do preço do sistema ser caro, normalmente ainda faz-se necessário adquirir equipamentos especiais elevando ainda mais o custo e tornando impossível o acesso acadêmico a esses recursos. Durante a realização desse trabalho houve a oportunidade de

experimentar o sistema Jame ITV [59], já abordado na seção 6.2.2, pois oferece uma versão de avaliação de alguns dias contribuindo para formação de opinião.

Outra abordagem possível foi realizar uma criteriosa pesquisa para encontrar trabalhos que de alguma forma relacionam-se com o tema abordado, principalmente entre trabalhos acadêmico-científicos. Nesse campo destaca-se inicialmente o GINGA-NCL que é a parte declarativa do middleware criado no Brasil e abordado na seção 3.4. Recentemente o grupo desenvolvedor do middleware GINGA disponibilizou uma ferramenta de autoria, chamada "*Composer*" e que possibilitou alguma comparação.

A convenção utilizada na comparação foi:

- **0**: O sistema apresenta a funcionalidade, porém sem aspecto diferencial, positivo ou negativo, de destaque em relação ao sistema comparado.
- +: O sistema apresenta um diferencial positivo em relação ao sistema comparado.
- -: O sistema apresenta um diferencial negativo em relação ao sistema comparado.

7.6.1 Comparação com o sistema Jame

A Tabela 22 abaixo compara o EMTV nesse trabalho com uma versão de avaliação, ou *trial*, do sistema Jame de agosto de 2007.

Característica	EMTV	Sist. Jame	Observações
Preço	+	-	• Jame é um sistema comercial proprietário estrangeiro.
GEM	**0**	**0**	• Ambos os sistemas são baseados na especificação GEM. • O sistema Jame é disponibilizado em uma versão MHP e uma versão OCAP. • O EMTV só foi testado em um sistema MHP.
Baseado em Componentes	**0**	**0**	• Ambos os sistemas são baseados em componentes.
Número de componentes	**0**	+	• O sistema Jame oferece 9 componentes gráficos: *BackGroundImage*, *Image*, *Single Line Text*, *Date And Time*, *IFrame*, *MultiLine Text*, *Scaled Video*, *PFrame* e *Single Colored Rectangle*; • O EMTV tem 5 componentes gráficos: *Screen* (equivalente ao BackGroundImage), *Text* (Single Line Text do Jame), *Image* (Image do Jame), *ImageButton* e *Question Group* (específico para Quiz).
Qualidade Gráfica	**0**	**0**	• Os componentes gráficos oferecidos no sistema Jame baseiam-se no carremento de arquivos gráficos externos de maneira bastante semelhante ao oferecido no EMTV. Os resultados gráficos obtidos são, nesse caso, equiparados.
Quiz	+	**0**	• Embora o sistema Jame possibilite a geração de programas do tipo *Quiz* o EMTV oferece um componente específico para esse fim o que facilita o desenvolvimento.
Componente para envio de informações a	+	-	• O EMTV oferece o componente "*Connection*" que estabelece uma conexão caso haja rede física disponível, coleta e envia informações fornecidas pelo usuário através

servidores externos			do sistema. • A versão avaliada do Jame não continha evidencias de tal funcionalidade.
Baseado em Leitura de arquivo XML	**0**	**0**	• Ambos os sistemas baseiam-se na leitura de um arquivo XML para formação da aplicação. • O sistema Jame tem a preocupação de converter o XML para um formato binário minimizando o tempo de transmissão do sistema pela rede de TVD o que lhe oferece uma tendência de vantagem.
Flexibilidade na geração de aplicações	**0**	**0**	• Cada tela de uma aplicação no sistema Jame possui um arquivo de configuração distinto enquanto no EMTV há apenas um arquivo de configuração e diferentes telas são manipuladas através da própria aplicação. • O sistema Jame permite programar através do arquivo de configuração um número maior de eventos como *Focused*, *Pressed*, *Disabled*. • A versão atual do EMTV embora possa ser estendida para suporte a outros tipos de aplicação, atualmente se dedica a solução de aplicações tipo *Quiz*.
Facilidade na geração de aplicações	**0**	+	• Ambas as plataformas são baseadas em XML e, ambas possuem sintaxes próprias e relativamente simples para geração de aplicações. • O sistema Jame atualmente é mais fácil de ser utilizado, pois oferece uma ferramenta de autoria gráfica chamada Jame Author.
Carregamento do sistema	**0**	**0**	• Ambos os sistemas podem ser carregados como uma aplicação *XLet* através do sistema de TVD ou serem inseridos em STBs como parte integrante do equipamento.
Atualização dinâmica	**-**	+	• O sistema Jame, se estiver diretamente instalado no STB, oferece método para atualização através do sistema de TVD e o EMTV não abordou essa questão.

Tabela 22 – Algumas comparações desse trabalho com o sistema Jame

7.6.2 Comparação com o sistema GINGA-NCL

Característica	EMTV	GINGA-NCL	Observações
Preço	+	**-**	• GINGA NCL [40] foi disponibilizado sob licença GPL v2 livre de royalties. • O EMTV pode ser utilizado sem qualquer restrição de uso e distribuição.
Complexidade	**0**	+	O GINGA-NCL é um *middleware* que vem sendo desenvolvido há vários anos tendo recebido contribuições de várias pessoas e trabalhos científicos. É natural que seja mais poderoso e consequentemente mais complexo do que

			o EMTV.
Forma de carregamento	+	**0**	• O GINGA-NCL é parte importante do middleware GINGA e para ser utilizado deverá estar previamente instalado fisicamente no STB. Isso não deve ser um problema, pois o sistema brasileiro de TVD exige a presença da tecnologia no equipamento. • O EMTV pode ser carregado através do sistema de TVD como qualquer outra aplicação Xlet. O EMTV pode ainda ser parte integrante do próprio STB.
Baseado em Componentes	**0**	**0**	• O GINGA NCL define cada elemento que pode ser manipulado em uma aplicação como elemento de mídia dada a abrangência das possibilidades oferecidas. Esses elementos de mídia não deixam de serem componentes. Há o suporte a manipulação de diversas propriedades de tais elementos. • O EMTV é baseado em componentes.
Número de componentes	**0**	+	• O GINGA NCL oferece suporte a uma grande quantidade de elementos de mídia podendo ser imagens, textos, scripts, vídeos e outros programas NCL, programas procedurais. • A atual versão do EMTV oferece um número limitado de componentes: Apenas 5 componentes gráficos e 1 componente não gráfico.
Sincronização entre componentes	**0**	+	• O GINGA NCL possui como premissa a possibilidade de sincronizar a apresentação dos componentes na tela baseado em temporização. • O EMTV é oferece apenas suporte à sincronização baseada em um número limitado de eventos.
Quiz	+	**0**	• Embora o GINGA NCL possibilite a geração inclusive de complexos programas do tipo *Quiz* o EMTV oferece um componente específico para esse fim o que facilita o desenvolvimento.
Baseado em Leitura de arquivo XML	**0**	**0**	• Ambos os sistemas baseiam-se na leitura de um arquivo XML para formação da aplicação. • O arquivo NCL é baseado no modelo NCM.
Ferramenta de autoria	**-**	+	• O GINGA-NCL oferece gratuitamente a ferramenta de autoria chamada *Composer*. Ela oferece funcionalidades que permitem criar documentos NCL que geram as aplicações de TVD. Apesar de se encontrar em aprimoramento essa ferramenta abstrái parte dos conceitos do NCM e que tornariam o desenvolvimento mais difícil.

Tabela 23 – Algumas comparações desse trabalho com o GINGA NCL

7.6.3 Comparação com o sistema AppTV

Característica	EMTV	AppTv	Observações
Preço	**0**	**0**	• O AppTV [25] é uma plataforma desenvolvida em ambiente acadêmico assim como o EMTV. A princípio não há nenhum custo envolvido em seu uso.
Forma de carregamento	**0**	**0**	• Ambos os sistemas podem ser carregados pelo sistema de TVD através do gerenciador de *XLet* presente em STBs compatíveis com GEM.
Baseado em Leitura de arquivo XML	**0**	**0**	• Ambos os sistemas são baseados na leitura de arquivos XML que são utilizados para a montagem da aplicação. • Ambos os sistemas partiram de formato proprietário para o XML.
Tipos de aplicações	**0**	+	• O AppTV foi apresentado já com suporte a dois tipos de aplicações: *Quiz* e "Informações", para aplicações apresentativas, enquanto o EMTV suporta inicialmente apenas aplicações tipo *Quiz*. • O EMTV apenas está estruturado para implementação de novos tipos de aplicações.
Baseado em componentes	**0**	**0**	• O AppTv é também de certa forma baseado em componentes na medida em que realiza a leitura de um arquivo de configuração externo que descreve o comportamento da interface.
Número de componentes	+	-	• Equivalentemente o AppTV oferece apenas 3 tipos de componetes: 1 para configuração do posicionamento e cor de fundo da aplicação, 1 para montagem do *Quiz* e 1 para montagem de aplicações informativas. • A atual versão do EMTV oferece 5 componentes gráficos e 1 componente não gráfico que podem ser utilizados para compor diferentes formatos de aplicativos.
Qualidade Gráfica	+	-	• O EMTV oferece 5 componentes, cada um deles com diversas propriedades que podem resultar numa infinidade possibilidades de layout das aplicações. Suporta imagens que podem ser visíveis ou não a partir de eventos na aplicação, suporta textos com possibilidade de efeitos tais como *blinking* e *scrolling* além de conteúdo dinâmico definido no arquivo de configuração. • Os componetes oferecidos no AppTV oferecem controle limitado sobre propriedades como a troca de cor de fonte e de posicionamento dos textos dentro de botões. O resultado é que esse sistema é capaz de gerar aplicações com aparência pouco atrativas, ou seja pobre em recursos visuais.
Funcionalidades no Quiz	+	-	• O AppTV oferece poucas das funcionalidades levantadas nesse trabalho necessárias em um sistema de Quiz voltado

			para educação. Não oferece, por exemplo, capacidade de múltipla escolha, não oferece campos editáveis, não finaliza a aplicação de modo a mostrar quais são as respostas fornecidas corretas, não envia o conteúdo coletado através de canal de retorno, não permite que diferentes questões tenham diferentes números de alternativas.
Componente de comunic. externa	+	-	• A versão avaliada do AppTV não provê comunicação com servidor externo. • O EMTV oferece o componente Connection que estabelece uma conexão caso haja rede física disponível, coleta e envia informações fornecidas pelo usuário através do sistema.
Suporte a scripts	+	-	• O EMTV oferece suporte a scripts simples que se limita a expressões boleanas, mas que dão ao sistema a capacidade de alterar seu conteúdo de acordo com eventos definidos no arquivo de configuração. • Não há menções no AppTV de nenhum suporte desse tipo.
Ferramenta de autoria	-	+	• O EMTV não oferece ferramenta de autoria gráfica enquanto o AppTV já conta com uma ferramenta de autoria chamada de InteracTV [62].

Tabela 24 – Algumas comparações desse trabalho com o AppTV

7.7 Considerações finais

Esse capítulo inicialmente apresentou as experiências que foram realizadas para testes, validação e verificação das funcionalidades implementas no desenvolvimento da plataforma EMTV. Foram basicamente utilizadas duas abordagens em que, na primeira, foi utilizado o emulador XletView e, na segunda, foi utilizado um equipamento real de TVD. A conclusão a que se chega é que a plataforma foi desenvolvida e que funciona de acordo com os requisitos levantados ao longo desse trabalho. Finalmente foi realizado um estudo comparativo entre a plataforma obtida e outros projetos que tivessem alguma relação com a EMTV. Nesse aspecto, demonstrou-se que a plataforma ainda é bastante simples se for comparada com sistemas comerciais, como o sistema Jame [59], ou sistemas que vêm sendo desenvolvidos há bastante tempo como o caso do GINGA-NCL [43], porém apresenta diferenciais positivos quando leva-se em consideração o componente oferecido para gerar aplicações tipo Quiz, voltadas para propósitos educativos. O *framework* AppTV [25], também vem sendo recentemente desenvolvido por outra instituição de pesquisa acadêmica, contém as mesmas prerrogativas utilizadas no EMTV e, oferece como diferencial um componente voltado para aplicações apresentativas. Por outro lado, o AppTV não menciona testes com um sitema real de TVD, não oferece controle de visibilidade dos componentes baseado em eventos, não oferece vários componentes visuais além do componente de envio pelo canal de retorno que estão já implementados no EMTV.

8 Conclusões Finais

O objetivo desse trabalho era desenvolver uma plataforma, que foi denominada EMTV, sem custo e voltada para desenvolvedores com pouca experiência em programação de modo que facilitasse a geração e distribuição de aplicações para o sistema de televisão digital empregado no Brasil. Para que a plataforma pudesse contribuir com o desenvolvimento social do país as aplicações geradas deveriam possuir utilidade se empregadas em processos educativos.

Para possibilitar o desenvolvimento da EMTV partiu-se de um estudo geral sobre todas as tecnologias envolvidas no funcionamento dos principais sistemas de TVD existentes no mundo. Foi abordado desde o princípio de funcionamento dos primeiros sistemas de televisão até a evolução aos sistemas digitais. Foram realizadas constantes análises sobre as escolhas tecnológicas realizadas pelo governo brasileiro visando aproximar esse trabalho da realidade prática do país. Os conceitos apresentados foram importantes, pois além de fornecerem conhecimentos sobre o estado da arte da tecnologia, contribuiram para a posterior definição da arquitetura de software da plataforma tendo em vista a conexão entre as várias partes integrantes dos sistemas de TVD.

Até então, a tecnologia de TVD foi abordada de maneira geral, levando em consideração inclusive aspectos de *hardware.* O estudo prossegue restringindo seu foco ao *software* que constitui um dos principais componentes da TVD, que é o *middleware*. Foi apresentado o conceito de *middleware* aplicado em TVD e em seguida uma contextualização dos padrões de *middleware* que foram criados para os sistemas de TVD abertos, inclusive para o brasileiro. Essa parte do estudo foi especialmente importante para tomada da decisão que, dentre os padrões de *middleware* existentes, o padrão definido pela especificação GEM seria mais interessante para o desenvolvimento da plataforma em virtude das possibilidades que oferece e da convergência dos padrões de *middleware* nessa direção.

Esse trabalho prosegue com a realização de um levantamento dos tipos de aplicações para TVD que podem ser encontrados em países onde os sistemas digitais já se encontram em funcionamento. A diversidade de aplicações tornou relevante apresentar alguns conceitos importantes para classificação dessas aplicações por tipo de interatividade. Também foram apresentados alguns dos principais termos utilizados para categorização de aplicações. Entre essas categorias está a *T-Learning* que possui especial importancia no contexto desse trabalho. Um estudo um pouco mais aprofundado sobre essa categoria de aplicações permitiu identificar que aplicações comumente chamadas de *Quiz* fazem parte dessa categoria, pois contribuem em processos educativos. Foram apresentados vários conceitos que não só justificam essa afirmação como também a escolha desse tipo de aplicação para cumprimento de um dos objetivos básicos desse trabalho.

Tendo definido que a plataforma gera aplicações tipo *Quiz,* foi apresentada uma caracterização desse tipo de aplicação, partindo da definição do termo até a apresentação dos requisitos essenciais, levantados a partir de sistemas com objetivos semelhantes tais como o Moodle que possui um módulo específico para geração de *Quiz* para Internet.

A integração entre todos os conceitos descritos até então foram utilizados para definir conceitualmente as principais características da plataforma EMTV:

- Extensão de um *middleware* compatível com a GEM: Como a plataforma age assim como o *middleware* convertendo dados em aplicações, abstraindo o funcionamento do *hardware* pode-se dizer que a plataforma atua como uma extensão do *middleware*.

- Simplicidade e abordagem declarativa: Conclusões ao logo do texto mostraram que a abordagem declarativa pode simplificar o uso da plataforma. Embora para ser desenvolvida, a plataforma utilizou um *middleware* procedural, sob o ponto de vista do programador, a plataforma se comporta como um *middleware* declarativo.
- Configurável e de fácil reuso: Importante para garantir a utilidade da plataforma em diferentes ocasiões. A plataforma recebe um arquivo de configuração externo a partir do qual gera as aplicações. Arquivos de configuração diferentes geram aplicações com aparência e/ou comportamentos diferentes. Essa preocupação torna a solução mais complexa pela flexibilidade que oferece e caracteriza uma plataforma para geração de toda uma família de aplicações e não tão simplesmente de uma única aplicação.
- Capacidade de gerar aplicações tipo *Quiz* que, conforme apresentado, é bastante utilizada para fins educativos.
- Baseada em componentes: Essa foi uma importante necessidade levantada ao longo desse estudo. Nessa ocasião foram apresentadas definições e vantagens no emprego dessa abordagem no desenvolvimento da plataforma.
- Oferecimento de ferramenta de validação e edição: Afim de que a plataforma possa ser completamente caracterizada foram definidos os procedimentos para validação e edição do arquivo de configuração.

A definição das características e o conhecimento de seus requisitos permitiram a realização de um estudo para o levantamento de aspectos técnicos da plataforma:

- Foram identificados e descritos os componentes básicos necessários para cumprimento dos objetivos funcionais da plataforma. Foram definidos 5 componentes gráficos, denominados respectivamente "*Application Screen*", "*Application Text*", "*Application Image*", "*Image Button*", e "*Application Question Group*". Foi também definido 1 componente não gráfico cujo propósito é realizar a comunicação da plataforma com um servidor externo através do canal de retorno se estiver disponível. Esse componente não visual foi denominado "*Application Communication*".
- Foi escolhido o formato XML para formação do arquivo de configuração. Isso contribuiu não só para garantia de legibilidade e simplicidade na manipulação do arquivo de configuração como também para o cumprimento do objetivo de apresentar um método de validação do mesmo, tendo em vista que existem diversas ferramentas, inclusive gratuitas que, fazendo uso de um DTD, validam a sintaxe do arquivo.
- Foram identificados os principais processos internos isto é, módulos de funcionalidades, do software bem como identificadas as bibliotecas necessárias para implementação técnica de cada um deles. Com isso foi possível identificar que a única biblioteca de terceiros, isto é, que não faz parte da especificação GEM nem das classes definidas para a plataforma, é a biblioteca NanoXML [57] que, além de fisicamente pequena em termos de *bytes*, é recomendada por referências para a interpretação de arquivos XML.
- Tendo a definição de uso do XML e a definição dos componentes e de suas características, partiu-se para o detalhamento dos campos de cada componente e das regras de formação do arquivo de configuração.

Esse trabalho registra então, como foi realizado o processo de implementação da plataforma. Parte um breve estudo sobre as ferramentas de desenvolvimento disponíveis no mercado e demonstra as vantagens na utilização da ferramenta Eclipse. Foi apresentado que na

primeira etapa do desenvolvimento da plataforma foi necessário o uso de uma ferramenta de emulação. Sendo assim, é apresentada nesse trabalho, uma lista dos principais emuladores de sistemas de TVD disponíveis no mercado que é seguida pela conclusão que, apesar de algumas deficiências, a ferramenta XletView [61] permitiria o desenvolvimento de grande parte das funcionalidades da plataforma. Antecedendo a parte de implementação do software foi apresentado um diagrama de classes de modo a obter a plataforma definida. O que se segue a partir desse ponto do trabalho é o registro de detalhes de implementação técnica das principais classes bem como seu comportamento na plataforma.

A parte final desse trabalho se dedicou a apresentar a utilização da plataforma, as experiências realizadas e a comparação entre essa plataforma e sistemas com funcionalidades semelhantes. Quanto à utilização, foi mostrado o uso de um editor de textos comum, programado com alguns botões de macros que auxiliam o programador na montagem do arquivo de configuração, já que a plataforma ainda não conta com ferramenta gráfica de autoria. Também demonstrou que a validação pode ser realizada por programas, como por exemplo, navegadores de Internet, através do uso do DTD, fornecido no Anexo B desse trabalho.

No que diz respeito às experiências ficou registrado que foram realizadas duas etapas de experiências:

- Uma primeira etapa em que foram realizadas experiências utilizando emulador XletView[61]: Pela indisponibilidade de equipamentos, foi utilizado um emulador de um STB com middleware GEM e, conforme justificado ao longo do texto, dentre os emuladores existentes, foi utilizado o XletView que, apesar das limitações, permitiu desenvolver boa parte da arquitetura da plataforma proposta.
- Uma segunda etapa em que foram realizadas experiências utilizando um STB e um monitor de HDTV reais: A plataforma obtida após a realização da primeira etapa de experiências foi ajustada para funcionar em sistemas reais de TVD. Foi utilizado um STB equipado com um *middleware* MHP *profile* 1.0.3 que provê uma interface ethernet através da qual é possível carregar a plataforma instalada em um servidor de aplicações Apache. A experiência também realizou a transmissão de dados coletados no *Quiz* gerado pela EMTV para o mesmo servidor Apache, convenientemente utilizado para coleta desses dados. Nesse trabalho foi apresentada uma ilustração da conexão entre os equipamentos envolvidos além de fotografias tiradas da plataforma em funcionamento.

Esse trabalho finalmente apresenta um levantamento de sistemas cuja finalidade seja semelhante à finalidade da plataforma proposta. Nesse levantamento é considerado um sistema comercial, embora outros também pudessem ser utilizados, 1 *middleware* declativo aberto e 1 trabalho acadêmico nas comparações e conclui que nas três comparações, a EMTV sempre apresenta algum diferencial positivo quando levado em consideração o componente direcionado para geração de *Quiz*.

A conclusão final é que a plataforma EMTV foi proposta segundo motivações legítimas detalhadamente analisadas ao longo desse texto. O resultado foi a obtenção de uma plataforma em funcionamento em um sistema real de TVD e que, apesar de limitações que podem ser superadas em trabalhos futuros, facilita do desenvolvimento de aplicações tipo *Quiz*. A EMTV está disponibilizada gratuitamente pra uso de toda a comunidade acadêmico-científica.

8.1 Pontos de melhoria e sugestões para trabalhos futuros

Esse trabalho resultou na primeira versão da EMTV para cumprimento dos objetivos expostos ao longo desse texto. Naturalmente que há bastante espaço para melhoria de algoritmos, para melhoramento de performance e de recursos gráficos. A seguir estão relacionados alguns itens que poderiam contribuir no processo de evolução da EMTV:

i. Estudo para desenvolvimento de novos componentes gráficos, essenciais para a boa experiência do usuário do aplicativo. Algumas sugestões:
   - Estudo de adaptações no componente "*Image*" para exibição de imagens animadas e vídeos.
   - Desenvolvimento de componente para execução de arquivos contendo áudio.

ii. Estudo relativo à questão de segurança na transmissão dos dados através do canal de retorno.
   - Já existem alguns trabalhos que desenvolveram bibliotecas apropriadas para criptografia e certificação dos dados, tanto no lado do STB quanto no lado do servidor remoto. Na medida em que se considerar esse sistema para geração de aplicações voltadas para T-Banking isso será fundamental.

iii. Estudo para implementação de suporte à execução de scripts fornecendo ao sistema capacidade para realizar tarefas mais complexas e customizadas.

iv. Criação de uma ferramenta gráfica para geração do arquivo de configuração do sistema. Esse tipo de ferramenta é chamado de ferramenta de autoria e facilita bastante o trabalho de desenvolvimento de aplicações, pois evita totalmente que o programador precise conhecer detalhes de sintaxe do arquivo de configuração.

v. Estudo sobre como realizar a identificação inequívoca do STB. A especificação GEM não fornece nenhum método essencial para identificação do STB por um servidor remoto. Isso foi contornado na plataforma EMTV inserindo suporte a caixas de edição de texto através das quais o usuário da aplicação fornece seus dados pessoais que são enviados ao servidor. Buscar maneiras alternativas de fazer essa identificação é sem dúvida uma contribuição interessante para esse trabalho.

vi. Estudo sobre novos tipos de aplicação que podem ser geradas pela EMTV.

Bibliografia

[1] NIELSEN MEDIA RESEARCH, 2006.

[2] Preto, Nelson de Lucca; FERREIRA, Simone de Lucena: "Possibilidades Interativas do Sistema Brasileiro de Televisão Digital Terrestre", 2006.

[3] COMITÊ GESTOR DA INTERNET NO BRASIL: "Pesquisa sobre o uso das tecnologias da informação e comunicação", 2005. Disponível em http://www.cgi.br/publicacoes/artigos/artigo33.htm acessado em Junho de 2007.

[4] UNCTAD: "Millennium Development Goals Indicators", 2006. Disponível em http://unstats.un.org/unsd/mdg/Host.aspx?Content=Data/Trends.htm, em Junho de 2007.

[5] HOFFMAN, Donna L.; NOVAK, Thomas P.; VENKATESH, Alladi: "Has the Internet become indispensable?", 2004. Communications of the ACM, Volume 47 Issue 7.

[6] ROSA, Almir: "TV Digital. Entrando no ar! Agora, no Brasil", 2003. XXVI Congresso Anual em Ciência da Comunicação, Belo Horizonte/MG.

[7] SANTOS, Cristina Omena: "Reflexões sobre a convergência tecnológica: A TV Digital Interativa no Brasil", 2001. Disponível em http://bocc.ubi.pt/pag/santos-adriana-tv-digital-interactiva-no-brasil.pdf, em Junho de 2007.

[8] MINISTÉRIO DA CULTURA DO BRASIL: "Questões Centrais para uma tomada de decisão – Sugestões do Ministério da Cultura ao Comitê de Desenvolvimento do SBTVD", 2006.

[9] SACRINI, Marcelo: "O uso da televisão digital no contexto educativo".2005. ETD Educação Temática Digital,Campinas, SP, v.7, n.1, p.31-44, dez. 2005.

[10] MACHADO, Arlindo: Livro "A televisão levada a sério". Editora Senac, 4ª Edição, 2000.

[11] Bolaño, César; VIEIRA, Vinícius Rodrigues: "TV digital no Brasil e no mundo: estado da arte", 2004. Revista de Economía Política de las Tecnologías de la Información y Comunicación www.eptic.com.br, Vol. VI, n. 2, Maio – Ago. 2004.

[12] FERNANDES, Jorge; LEMOS, Guido; SILVEIRA, Gledson: "Introdução à Televisão Digital Interativa: Arquitetura, Protocolos, Padrões e Práticas", 2004. Jornada de Atualização em Informática do Congresso da Sociedade Brasileira de Computação, JAI-SBC, em Salvador – BA Agosto de 2004. Anais do JAI—SBC, 2004.

[13] RÉGIS, Marcus Vinícius de O.; FECHINE, Joseana Macedo: "Introdução ao sistema de TV Digital", 2005.

[14] WU, Yiyan; HIRAKAWA, Shuji; ULRICH, H. Reimers; WHITAKER, Jerry: "Overview of Digital Television Development Worldwide", 2006. Proceedings of the IEEE, vol. 94, n. 1, Jan 2006.

[15] YAMADA, Fujio; SUKYS, Francisco; BEDICKS, Gunnar Jr; AKAMINE, Cristiano; RAUNHEITTE, Luís T. M.; DANTA, Carlos E.: "Sistema de TV Digital", 2004. Revista de Engenharia e Computação da Universidade Presbiteriana Mackenzie- Ano 5, número 5 – 2004.

[16] SCURI, Antonio Escaño: "Fundamentos da Imagem Digital", 1999. Tecgraf/PUC-Rio,1999. Fonte: http://www.tecgraf.pucrio.br/~rtoledo/cg1/apostila%20imagem%20digital.pdf, em julho de 2007.

[17] OLIVEIRA, Etienne C. R. de;ALBUQUERQUE, Célio Vinícius N. de: "TV Digital Interativa: Padrões para uma nova era", 2005.

[18] CCIR: "Characteristics of television systems. Report 624-4", Reports of the CCIR, 1990, Annex to Volume XI-Part 1, page 9, 1990.

[19] MENDES, Luciano L.; FASOLO, Sandro A.: "Introdução a Televisão Digital",2002. Disponível em: http://cict.inatel.br/nova2/docentes/lucianol/Artigos/SIT2002/TV_Digital_SIT2002.pdf, em Julho de 2007.

[20] MPEG: "Moving Pictures Experts Group". Disponível em http://www.chiariglione.org/mpeg/ em Julho de 2007.

[21] ITU-R: International Telecommunication Union - Radiocommunication. Disponível em: http://www.itu.int/net/home/index.aspx

[22] PICCIONI, Carlos Alexandre. Modelo e implementação de um serviço de datacasting para televisão digital, 2005. 116f. Dissertação para o programa em pós-graduação em engenharia elétrica da Universidade Federal de Santa Catarina.

[23] JONES, John: "DVB-MHP/Java TV™ Data Transport Mechanisms", 2002

[24] ANDREATA, Jomar Alberto: "InteraTV: Um Portal para Aplicações Colaborativas em TV Digital Interativa Utilizando a Plataforma MHP, 2006. 110f. Dissertação para o programa em pós-graduação em engenharia elétrica da Universidade Federal de Santa Catarina.

[25] HATTORI, Lile; SILVA, Sylvio Siqueira;TAVARES, Tatiana A.; NETO, Manoel C. M.; SAIBEL, Celso – "Utilizando o framework AppTV no desenvolvimento de aplicações para TV Digital Interativa", 2005.

[26] GOMES, Geraldo Gil R; MENDES, Luciano L.:"Estudo da constelação 16QAM". Disponível em http://cict.inatel.br/nova2/docentes/lucianol/Artigos/Incitel/constelacao_16QAM.pdf. Disponível em Julho de 2007.

[27] CITTA, Richard; SGRIGNOLI,Gary: "ATSC Transmission system: VSB Tutorial", 1997.

[28] FEDERAL COMMUNICATIONS COMISSION: "DTV Report on COFDM and 8-VSB Performance", 1999.

[29] STOTT, J. H.: "Paper: Explaining some of the magic of COFDM", 1997. Proceedings of 20th Internation Television Symposium, June of 1997. Disponível em http://www.bbc.co.uk/rd/pubs/papers/paper_15/paper_15.shtml em Julho de 2007.

[30] RODRIGUES, Ana Luiza; GOMES, Regina M.: "Modulação COFDM – Uma proposta atrativa para os padrões de TV digital". Revista Digital Online, Vol 3 Ago 2004. Disponível em http://www.revdigonline.com/artigos_download/art_16.pdf em Julho de 2007.

[31] MODY, Mihir: "Guide to MPEG-2 System Standard", 2005. Disponível em http://pdf.chinaecnet.com/pdf1/pdf.newpro2005/Other/ti_mpeg2(11).pdf em Julho de 2007.

[32] BASTOS, Arílson; FERNANDES, Sérgio: "Livro: Televisão Digital". 1ª edição, Rio de Janeiro – RJ, 2004. Editora Forense. ISBN 85-902135-5-2.

[33] NIGRI, H. S.;ALBUQUERQUE, Marcelo P. e Márcio P.: "O padrão MPEG para compressão de vídeo", 2000. Disponível em ftp://ftp2.biblioteca.cbpf.br/pub/apub/2000/nt/nt_zip/nt00200.pdf em Julho de 2007.

[34] EMMERICH, Wolfgang; AOYAMA, Mikio; SEVENTEK, Joe: "The Impact of Research on Middleware Technlogy", ACM SIGSOFT Software Engineering Notes, page 21, volume 32 number 1, 2007.

[35] Interactive TV Web – Digital TV, DVB and ATSC tutorials. Disponível em http://www.interactivetvweb.org em julho de 2007.

[36] INFORMATION SOCIETY TECHNOLOGIES – MHP Knowledge Database MHP-KDB, 2006 disponível em http://www.mhp-knowledgebase.org/publ/mhp-guide.pdf

[37] ATSC Home. Disponível em http://www.atsc.org em julho de 2007.

[38] JavaTV API 1.1 – Disponível em http://java.sun.com/javame/reference/apis/jsr927/ em Julho 2007.

[39] DVB-MHP Digital Vídeo Broadcasting Multimedia Home Platform. Disponível em http://www.mhp.org em julho de 2007.

[40] *Middleware* GINGA. Disponível em http://www.ginga.org.br/ em agosto de 2007.

[41] Site do laboratório de aplicações de vídeo digital. Disponível em http://jdvod.lavid.ufpb.br/ em agosto de 2007.

[42] *Middleware* GINGA – TV interativa se faz com GINGA. Disponível em http://www.ncl.org.br em agosto de 2007.

[43] LABORATÓRIO TELEMIDIA – Ambiente para Desenvolvimento de Aplicações Declarativas para a TV Digital Brasileira, 2007.

[44] LEMOS, Guido; LEITE,Luiz Eduardo; FREIRE, Carlos Eduardo C. – GINGA-J The Procedural *Middleware* for the Brazilian Digital TV System, 2006.

[45] CESAR, Pablo: "A Graphics Software Architecture for High-End Interactive TV Terminals", Dissertation for the degree of Doctor of Science in Technology presented on Helsinki University of Technology, 2005. ISBN 951-22-7888-X.

[46] PENG, Chengyuan: "Digital Television Applications", Dissertation for the degree of Doctor of Science in Technology presented on Helsinki University of Technology, 2002. ISBN 951-22-6171-5

[47] JOLY, Ana Vitoria: "Programação Educativa Destinada à Televisão Interativa", 2003. Disponível em: http://www.bocc.ubi.pt/pag/joly-vitoria-programacao-educativa-televisao-interactiva.pdf em Julho de 2007.

[48] AMARAL, Sergio Ferreira; PACATA, Daniel Moutinho: "TV Digital Interativa no Espaço Educacional". Artigo publicado no Jornal da Unicamp em 2003. Disponível em http://www.unicamp.br/unicamp/unicamp_hoje/ju/setembro2003/ju229pg2b.html em agosto de 2007.

[49] DAMÁSIO, Manoel; QUICO, Célia: "T-Learning and Interactive Television Edutainment: the Portuguese Case Study", 2003.

[50] BATES, Peter; ATWERE, Daniel: “Interactive TV: A learning platform with potential, Learning and Skills Development Agency”, 2003. Disponível em: http://www.lsda.org.uk/files/pdf/1443.pdf. ISBN 1853388351.

[51] BERTOTI, Giuliano A.; ALMEIDA, Felipe A. D.; BACCAN, Davi D. : “Educação à Distância Mediada pela Televisão Interativa: Panorama Atual e Perspectivas para o Brasil”, 2004. Disponível em http://sbie2004.ufam.edu.br/anais_cd/extras/anaisvolI/vDigital/artigos/5544.pdf

[52] WALLDÉN, Sari; SORONEN, Ane: “Edutainment. From Television and Computers to Digital TV”, 2004. Disponível em www.uta.fi/hyper/julkaisut/b/fitv03b.pdf em 2007.

[53] WAISMAN, Thais: “Usabilidade em Serviços Educacionais em Ambientes de TV Digital”. Dissertação apresentada à Escola de Comunicação e Artes da Universidade de São Paulo para obtenção de título de doutora em ciências da comunicação. 2006.

[54] JOHNSON, Kevin; HALL, Timothy; O’KEEFFE, Derek: “Generation of Quiz Objects (QO) with Quiz Engine Developer (QED)”, 2005. ISBN: 0-7695-2385-4.

[55] LTSC Home - Learning Technology Standards Committee. Disponível em http://ieeeltsc.org/ em 2007.

[56] Moodle Course Management System Home Page. Disponível em http://moodle.org/ em 2007.

[57] NanoXML Home – Disponível em http://nanoxml.cyberelf.be/ em 2007.

[58] HEINEMAN, George T.; COUNCILL, William T. Livro: “Component-Based Software Engineering”. 1st edition, 2001. Addison-Wesley. ISBN 0-201-70485-4.

[59] JAME – The Itv Production System – Disponível em http://jame.tv/jame/ em 2007.

[60] Osmosys – Digital Television Sollutions – Disponível em http://www.osmosys.tv/ em 2007.

[61] XLetView – Free MHP Emulator – Disponível em http://XLetView.sourceforge.net/ em 2007.

[62] ALMEIDA, Fabiano B. M. de; SOUZA, Ricardo B.; NETO, Manoel C. M.: “Ferramenta de Autoria Gráfica para Construção de Aplicações Utilizando o Framework AppTV”, 2007.

[63] JFE Jens’ File Editor Disponível em http://home.arcor.de/jensaltmann/JFE/jfe_eng.htm

[64] CARVALHO, Fabrício B. S. de;ALENCAR, Marcelo Sampaio: “Análise da transmissão do canal de retorno do Sistema Brasileiro de TV Digital via Power Line Communications”, 2005.

[65] Eclipse.org Home – Disponível em http://www.eclipse.org em 2007.

Anexos

A. Tabela dos principais códigos para o campo “iKeys” do componente “ImageButton”:

Teclas	Código
‘0’..‘9’	48 .. 57
VK_ENTER	10
VK_LEFT	37
VK_UP	38
VK_RIGHT	39
VK_DOWN	40
RED BUTTON	403
GREEN BUTTON	404
YELLOW BUTTON	405
BLUE BUTTON	406

B. DTD para o arquivo de configuração

```
<!ELEMENT JCQAPP (SCR|TXT*|IMG*|IMGBTN*|QUEST?)* >
<!ELEMENT SCR (iid|ix|iy|ih|iw|istate|sBgcolor?|sbgImg?)>
<!ELEMENT iid (CDATA)>
<!ELEMENT ix (CDATA)>
<!ELEMENT iy (CDATA)>
<!ELEMENT ih (CDATA)>
<!ELEMENT iw (CDATA)>
<!ELEMENT istate (CDATA)>
<!ELEMENT sbgcolor (CDATA)>
<!ELEMENT sBgImg (CDATA)>
<!ELEMENT TXT (iid|ix|iy|ih|iw|sValue|sBgColor?|sForeColor?|sFontName?|iFontSize?|sCondition?|sEfect?)>
<!ELEMENT iid (CDATA)>
<!ELEMENT ix (CDATA)>
<!ELEMENT iy (CDATA)>
<!ELEMENT ih (CDATA)>
<!ELEMENT iw (CDATA)>
<!ELEMENT sValue (CDATA)>
<!ELEMENT sForeColor (CDATA)>
<!ELEMENT sFontName (CDATA)>
<!ELEMENT iFontSize (CDATA)>
<!ELEMENT sCondition (CDATA)>
<!ELEMENT sEfect (CDATA)>
<!ELEMENT IMG (iid|ix|iy|sFile|sCondition?)>
<!ELEMENT sFile (CDATA)>
<!ELEMENT IMGBTN (iid|ix|iy|sFile|sCondition?|sOnKDown|iKey)>
<!ELEMENT sOnKDown (CDATA)>
<!ELEMENT iKey (CDATA)>
<!ELEMENT QUEST (iid|iQx|iQy|iQw|iQh|iAltx|iAlty|iAltw|iAlth|
                 iAltIncX?|iAltIncY?|iSelImageIncX?|iSelImageIncY?|sQBgcolor?|
                 sQForecolor?|sAltBgcolor?|sAltValBgcolor?|sAltForeColor?|
                 sAltSelColor?|sSelIndImage?|sConfEndQuest?|sConfEndYes?|
                 sConfEndNo?|Question*)>
<!ELEMENT iQx (CDATA)>
<!ELEMENT iQy (CDATA)>
<!ELEMENT iQw (CDATA)>
<!ELEMENT iQh (CDATA)>
<!ELEMENT iAltx (CDATA)>
<!ELEMENT iAlty (CDATA)>
<!ELEMENT iAltw (CDATA)>
<!ELEMENT iAlth (CDATA)>
<!ELEMENT iAltIncX (CDATA)>
<!ELEMENT iAltIncY (CDATA)>
<!ELEMENT iSelImageIncX (CDATA)>
<!ELEMENT iSelImageIncY (CDATA)>
```

```
<!ELEMENT sQBgcolor (CDATA)>
<!ELEMENT sQForecolor (CDATA)>
<!ELEMENT sAltBgcolor (CDATA)>
<!ELEMENT sAltValBgcolor (CDATA)>
<!ELEMENT sAltForeColor (CDATA)>
<!ELEMENT sAltSelColor (CDATA)>
<!ELEMENT sSelIndImage (CDATA)>
<!ELEMENT sConfEndQuest (CDATA)>
<!ELEMENT sConfEndYes (CDATA)>
<!ELEMENT sConfEndNo (CDATA)>

<!ELEMENT Question (sText,sAlt+)>
<!ATTLIST Question multiAnswer CDATA #IMPLIED>

<!ELEMENT sText (CDATA)>

<!ELEMENT sAlt (CDATA)>
<!ATTLIST sAlt isAnswer CDATA #IMPLIED>
<!ELEMENT isAnswer (CDATA)>
```

C. Arquivo de configuração de uma aplicação “QUIZ” utilizando o sistema desenvolvido

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE JCQAPP SYSTEM "jcq.dtd">

<JCQAPP>
<CONN>
   <DIAL>
      <sPhone> 36146500 </sPhone>
      <sUsr> itv </sUsr>
      <sPass> password </sPass>
   </DIAL>
   <SRV>
      <sAddr> www.terra.com.br </sAddr>
      <sPort> 80 </sPort>
      <sInfo>/index.php?alpha=$0.1?num=$0.3?pass=$0.5?$D-$M-$A-$H:$m?resp=$resp </sInfo>
</SRV>
</CONN>
<SCR>
   <iid> 0 </iid>
   <ix> 40 </ix>
   <iy> 15 </iy>
   <ih> 576 </ih>
   <iw> 720 </iw>
   <istate> 0 </istate>
   <sBgcolor>0;0;0;255;</sBgcolor>
   <sbgImg>bg2.png</sbgImg>
</SCR>
<TXT>
   <iid> 1 </iid>
   <ix> 120 </ix>
   <iy> 500 </iy>
   <iw> 570 </iw>
   <ih> 30 </ih>
   <sValue> Plataforma de desenvolvimento de softare para TV Digital. Programa de Mestrado em Engenharia Elétrica
da UFAM </sValue>
   <sBgColor> 236;236;236;255; </sBgColor>
   <sForeColor> 0;20;225;220; </sForeColor>
   <sFontName> ARIAL </sFontName>
   <iFontSize> 24 </iFontSize>
   <sCondition> NOT $IE </sCondition>
   <sEfect> SCROLL </sEfect>
</TXT>
<TXT>
   <iid> 4 </iid>
   <ix> 120 </ix>
   <iy> 130 </iy>
   <iw> 380 </iw>
```

```
    <ih> 30 </ih>
    <sValue> Escolha apenas 1 opção </sValue>
    <sBgColor> 236;236;236;255; </sBgColor>
    <sForeColor> 0;0;225;220; </sForeColor>
    <sFontName> TIRESIAS </sFontName>
    <iFontSize> 24 </iFontSize>
    <sCondition>NOT $IE AND $R1</sCondition>
    <sEfect>  </sEfect>
</TXT>
<TXT>
    <iid> 3 </iid>
    <ix> 120 </ix>
    <iy> 500 </iy>
    <iw> 440 </iw>
    <ih> 30 </ih>
    <sValue> Falhou ao enviar informações pelo canal de retorno </sValue>
    <sBgColor> 236;236;236;255; </sBgColor>
    <sForeColor> 221;0;0;220; </sForeColor>
    <sFontName> ARIAL </sFontName>
    <iFontSize> 24 </iFontSize>
    <sCondition> $IE AND NOT $IS </sCondition>
    <sEfect> BLINK </sEfect>
</TXT>
<TXT>
    <iid> 2 </iid>
    <ix> 120 </ix>
    <iy> 500 </iy>
    <iw> 440 </iw>
    <ih> 30 </ih>
    <sValue> Suas informações foram enviadas com sucesso pelo canal de retorno. Obrigado. </sValue>
    <sBgColor> 236;236;236;255; </sBgColor>
    <sForeColor> 0;2221;0;220; </sForeColor>
    <sFontName> ARIAL </sFontName>
    <iFontSize> 24 </iFontSize>
    <sCondition> $IE AND $IS </sCondition>
    <sEfect> BLINK </sEfect>
</TXT>
<TXT>
    <iid> 6 </iid>
    <ix> 120 </ix>
    <iy> 130 </iy>
    <iw> 380 </iw>
    <ih> 30 </ih>
    <sValue> Agora confira suas respostas </sValue>
    <sBgColor> 236;236;236;255; </sBgColor>
    <sForeColor> 0;20;225;220; </sForeColor>
    <sFontName> TIRESIAS </sFontName>
    <iFontSize> 24 </iFontSize>
    <sCondition> $IE </sCondition>
    <sEfect> blink </sEfect>
</TXT>
<TXT>
    <iid> 8 </iid>
    <ix> 280 </ix>
    <iy> 445 </iy>
    <iw> 80 </iw>
    <ih> 30 </ih>
    <sValue>$QP / $QC</sValue>
    <sBgColor> 192;211;235;255; </sBgColor>
    <sForeColor> 0;20;225;220; </sForeColor>
    <sFontName> TIRESIAS </sFontName>
    <iFontSize> 30 </iFontSize>
    <sCondition> NOT $IE </sCondition>
    <sEfect> </sEfect>
</TXT>
<TXT>
    <iid> 7 </iid>
    <ix> 350 </ix>
    <iy> 400 </iy>
    <iw> 200 </iw>
```

```
    <ih> 30 </ih>
    <sValue> $D/$M/$Y $h:$m</sValue>
    <sBgColor> 192;211;235;255; </sBgColor>
    <sForeColor> 0;20;225;220; </sForeColor>
    <sFontName> TIRESIAS </sFontName>
    <iFontSize> 30 </iFontSize>
    <sCondition></sCondition>
    <sEfect> </sEfect>
</TXT>
<IMG>
    <iid> 9 </iid>
    <ix> 190 </ix>
    <iy> 320 </iy>
    <sFile>theend.PNG</sFile>
    <sCondition> $IE </sCondition>
</IMG>
<IMGBTN>
    <iid>10 </iid>
    <ix> 120 </ix>
    <iy> 430 </iy>
    <sFile> </sFile>
    <sCondition></sCondition>
    <sOnKDown> leftdown.PNG </sOnKDown>
    <iKey> 37 </iKey>
</IMGBTN>
<IMGBTN>
    <iid>2 </iid>
    <ix> 470 </ix>
    <iy> 430 </iy>
    <sFile> </sFile>
    <sCondition></sCondition>
    <sOnKDown> rightdown.PNG </sOnKDown>
    <iKey> 39 </iKey>
</IMGBTN>
<QUEST>
    <iid> 11 </iid>
    <iQx> 150 </iQx>
    <iQy> 20 </iQy>
    <iQw> 390 </iQw>
    <iQh> 100 </iQh>
    <iAltx> 235 </iAltx>
    <iAlty> 160 </iAlty>
    <iAltw> 200 </iAltw>
    <iAlth> 25 </iAlth>
    <iAltIncX> 0 </iAltIncX>
    <iAltIncY> 30 </iAltIncY>
    <iSelImageIncX> -23 </iSelImageIncX>
    <iSelImageIncY> 2 </iSelImageIncY>
    <sQBgcolor>236;236;236;255;</sQBgcolor>
    <sQForecolor>0;20;225;220;</sQForecolor>
    <sAltBgcolor>236;236;236;255;</sAltBgcolor>
    <sAltValBgcolor>0;225;50;255;</sAltValBgcolor>
    <sAltForeColor>0;30;25;120;</sAltForeColor>
    <sAltSelColor>0;20;225;220;</sAltSelColor>
    <sSelIndImage>selected.png</sSelIndImage>
    <sConfEndQuest>Fim das perguntas. O que você deseja fazer?</sConfEndQuest>
    <sConfEndYes>Finalizar</sConfEndYes>
    <sConfEndNo>Revisar respostas</sConfEndNo>
    <Question>
      <sText>Bem vindo ao EMTV. Um sistema para geracão de aplicações para TV Digital com baixo custo. Por favor
forneça seus dados:</sText>
      <sAlt select="false"> Informe o seu login: </sAlt>
      <sEdt type="alpha"> </sEdt>
      <sAlt select="false"> Informe sua senha: </sAlt>
      <sEdt type="pass"> </sEdt>
      <QImage>
        <sFile> keypad.png </sFile>
        <iQImageX> 120 </iQImageX>
        <iQImageY> 170 </iQImageY>
      </QImage>
```

```
        </Question>
        <Question>
            <sText>1. O que significam, respectivamente, os nomes Bolshevik e Menshevik, adoptados pelas duas facções
do Partido Social Democrata da Rússia, em 1903?</sText>
            <sAlt> a) Maioria e minoria</sAlt>
            <sAlt> b) Tradicional e radical</sAlt>
            <sAlt isAnswer="true"> c) Radical e traditional </sAlt>
            <sAlt> d) Minoria e maioria</sAlt>
            <QImage>
               <sFile> history.png </sFile>
               <iQImageX> 120 </iQImageX>
               <iQImageY> 175 </iQImageY>
            </QImage>
        </Question>
        <Question multiAnswer="true">
            <sText>8. Exemplo de uma questão múlti-resposta</sText>
            <sAlt isAnswer="true">a) Verdadeira1 </sAlt>
            <sAlt isAnswer="true">b) Verdadeira2 </sAlt>
            <sAlt> c) Falsa </sAlt>
            <sAlt isAnswer="true"> d) Verdadeira3 </sAlt>
        </Question>
    </QUEST>
</JCQAPP>
```

D. Configuração do Editor JFE

```
 [EMTV]
FileAssoc=*.xml
number=9
syntax2=<SCR>,1
syntax1=<JCQAPP>,2
syntax3=</SCR>,33554433
syntax4=<?,167772165
syntax5=<!,167772166
syntax7=<s,33554434
syntax6=<i,167772160
syntax8=<,134217730
syntax9=$,3
 [ExecApplication]
CaptureOut=0
RedirToFile=0
AppParms=
AppDir=
Compiler=
AppExe=C:\iTvTests\Quiz1\newApp.mcr
SaveBeforeRun=1
ShortPathname=0
ChangeDir=1
Flags=0
Toolname=newApp
[ExecApplication1]
CaptureOut=0
RedirToFile=0
AppParms=
AppDir=
Compiler=
AppExe=C:\iTvTests\Quiz1\screen.mcr
SaveBeforeRun=1
ShortPathname=0
ChangeDir=1
Flags=0
Toolname=screen
[ExecApplication2]
CaptureOut=0
RedirToFile=0
AppParms=
AppDir=
Compiler=
AppExe=C:\iTvTests\Quiz1\Text.mcr
SaveBeforeRun=1
ShortPathname=0
ChangeDir=1
Flags=0
Toolname=text
[ExecApplication3]
CaptureOut=0
RedirToFile=0
AppParms=
AppDir=
Compiler=
AppExe=C:\iTvTests\Quiz1\Img.mcr
SaveBeforeRun=1
ShortPathname=0
ChangeDir=1
Flags=0
Toolname=img
[ExecApplication4]
CaptureOut=0
RedirToFile=0
AppParms=
AppDir=
Compiler=
AppExe=C:\iTvTests\Quiz1\ImgBtn.mcr
SaveBeforeRun=1
ShortPathname=0
ChangeDir=1
Flags=0
Toolname=ImgBtn
[ExecApplication5]
CaptureOut=0
RedirToFile=0
AppParms=
AppDir=
Compiler=
AppExe=C:\iTvTests\Quiz1\Quest_Init.mcr
SaveBeforeRun=1
ShortPathname=0
ChangeDir=1
Flags=0
Toolname=Quest_Init
[ExecApplication6]
CaptureOut=0
```

```
RedirToFile=0                                   SaveBeforeRun=1
AppParms=                                       ShortPathname=0
AppDir=                                         ChangeDir=1
Compiler=                                       Flags=0
AppExe=C:\iTvTests\Quiz1\Quest_AddQuest.mcr
```

As macros “newApp.mcr”, “screen.mcr”, “Img.mcr”, “ImgBtn.mcr”, “Quest_AddQuest.mcr”, e “Quest_Init.mcr” são arquivos binários gerados pelo próprio JFE. Tais arquivos fornecem os campos de cada componente ao qual se referem seguindo as especificações definidas em **Erro! Fonte de referência não encontrada.**.

E. Configuração do servidor de coleta de resultados dos usuários

Para coleta das informações fornecidas pelos usuários do sistema foram utilizados:

- Um computador pessoal do tipo PC, coincidentemente o mesmo utilizado para fornecer os binários da plataforma estendida para o “caroussel de dados” do sistema de TVD.
- O programa servidor de páginas HTML Apache.
- O interpretador de scripts PHP para execução de 2 scripts principais:
    - Script para recepção e armazenamento dos resultados.
    - Script para visualização dos resultados armazenados.

A seguir o código fonte dos arquivo “quizresp.php” e “Quiz.php” que realizam a recepção e armazenamento dos resultados fornecidos pelos usuários.

```
<?php
    require_once("Quiz.php");

        $user   = $_GET["user"];
        $pass   = $_GET["pass"];
        $quizId = $_GET["quizId"];
        $totalQ = $_GET["totalQ"];
        $datet  = $_GET["datet"];
        $resp   = $_GET["resp"];

        $quiz = new quiz();

        if($quiz->init("responses"))
        {
                echo $quiz->writeResponse($user,$pass,$quizId,$totalQ,$datet,$resp);
                exit();
        }
        echo "ERROR\n";
?>
 (…)
```

Arquivo Quiz.rpt:

```
<?php
class Quiz
{
   protected $local, $config_dir, $cur_file, $cur_var, $mcontent, $file;
   function __construct(){ $this->config_dir  = "/";}
   function __destruct(){}

   public function init($database_file)
   {
      $this->setBaseFile($database_file);
      $this->mcontent = $this->readFile();
       return true;
   }

   public function setBaseFile($database_file) {if($database_file) { $this->cur_file = $database_file; }}

   public function getCurFile() { return $this->cur_file; }

   public static function getFile($file, &$content)
   {
      if(file_exists($file)){
         $content = file_get_contents($file);
         return true;
      }else{ Extras::box_info("Arquivo inexistente: ".$file,E_USER_ERROR); }
      return false;
   }

   protected function readFile(){
      $this->file = $this->cur_file.".txt";
      if(file_exists($this->file)) { $content = file_get_contents($this->file); }
      return $content;
   }

   public function writeResponse($user,$pass,$quizId,$totalQuest,$datetime,$resp)
   {
            if($this->mcontent!="") {
                     $lines  = explode("\n", $this->mcontent);
                     $len    = count($lines);
                     for($i=0;$i<$len;$i++){
                              $fields = explode("|",$lines[$i]);
                              if(count($fields)<4)
                                 continue;

                           if((strcasecmp ($fields[0], $user )==0 )&&
                              (strcasecmp ($fields[1], $pass )==0 )){ return "Suas respostas já foram registradas em " . $fields[4]; }
                        }
                     }
                     if(($fh = fopen($this->file,'a+'))) {
                              fwrite($fh,"$user|$pass|$quizId|$totalQuest|$datetime|$resp\n");
                              fclose($fh);
                     }
                     else{  return "Falhou ao registrar no servidor!"; }
                     return "Obrigado $user. Respostas computadas!";
            }
} ?>
```

A seguir o código fonte do arquivo “quizrpt.php” que fornece uma visualização simples dos resultados coletados.

```
<?php
   print ssection("Respostas coletadas no QUIZ");
   $filepath = "C:/ceteli/bugzilla/odbc/responses.txt";
   if(!$file=fopen($filepath, 'r')) {
      table_msg_echo("N&atilde;o foi possível abrir o arquivo $swapFile" , "center", "100%");
   }else{
     print title("Lista de resultados distintos registrados");
     $result_found = 0;
     if(filesize($filepath)>0)
     {
         print   "<table class=\"full\" width=\"100%\" border=0 cellspacing=0 cellpadding=0>".
                 '<th align="left" width="10%">Usuário</th>'. '<th align="center" width="10%">Pass</th>'.
                 '<th align="center" width="3%">QuizID</th>'. '<th align="center" width="8%"># Questões</th>'.
                 '<th align="center" width="12%"># Data-Hora</th>'. '<th align="left"># Respostas</th>'."</td>";
        while($line = fgets($file, BUFFER_LENGTH))
        {
           $color = ($color == COLOR_LIST_2) ? COLOR_LIST_1 : COLOR_LIST_2;
           $quizResponses = explode("|", $line);
           $quizResponses[0] = trim($quizResponses[0]);
           $result_found++;
           if($quizResponses[0] != ""){
              print   tbr($color, "", "top").
                    "<td align=\"left\"><font size=\"3\">".$quizResponses[0]. "</td>".
                    "<td align=\"center\"><font size=\"3\">".$quizResponses[1]. "</td>".
                    "<td align=\"center\"><font size=\"3\">".$quizResponses[2]. "</td>".
                    "<td align=\"center\"><font size=\"3\">".$quizResponses[3]. "</td>".
                    "<td align=\"center\"><font size=\"3\">".$quizResponses[4]. "</td>"..
                    "<td align=\"left\"><font size=\"4\">".$quizResponses[5]. "</td>".;
           }
        }
        if($result_found > 0) {print tbend();}
     }
     else{print box_info("N&atilde;o foram retornados resultados para sua procura."); }
     $canDelete = fclose($file);
   } ?>
```

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo