UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Uma análise do middleware Ginga e da TV Digital no Brasil

Claudio Soares Lopes Júnior

Juiz de Fora
Dezembro de 2008

Uma análise do middleware Ginga e da TV Digital no Brasil

Claudio Soares Lopes Júnior

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Orientador: Marcelo Lobosco

Juiz de Fora
Dezembro de 2008

UMA ANÁLISE DO MIDDLEWARE GINGA E DA TV DIGITAL NO BRASIL

CLÁUDIO SOARES LOPES JÚNIOR

MONOGRAFIA SUBMETIDADA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

_____

Marcelo Lobosco, DSc
(Presidente)

_____

Raul Fonseca Neto, DSc

_____

Ciro de Barros Barbosa, DSc

Juiz de Fora
Dezembro de 2008

Conteúdo

1. Introdução 1
2. A TV Digital 3
2.1. Sistema de TV digital 5
2.1.1. Padrões internacionais 6
2.1.2. Codificação do Áudio 7
2.1.3. Codificação do Vídeo 8
2.1.4. Sistema de transporte 9
2.1.4.1. Carrossel de Dados 9
2.1.5. Modulação 10
2.2. Canal de Retorno 11
2.3. Set-top Box 11
2.4. IPTV 12
3. Contextualização no Brasil 15
3.1. SBTVD 16
3.2. Decreto n.º 4.901 16
3.2.1. Questão social 18
3.3. Decreto n.º 5.820 19
3.4. Início das transmissões 21
4. Middleware 23
4.1. Linguagem declarativa e procedural 24
4.2. Middlewares internacionais 25
4.2.1. Middleware europeu 25
4.2.2. Middleware americano 26
4.2.3. Middleware japonês 27

4.3. Middleware brasileiro 28
5. Middleware Ginga 29
5.1. História 30
5.2. Ginga-CC 31
5.3. Ginga-J 33
5.3.1. JavaTV 35
5.3.2. APIs de inovação do Ginga-J 37
5.3.3. Royalitites 39
5.4. Ginga-NCL 40
5.4.1. A linguagem NCL 41
5.4.2. NCLua 43
5.4.3. A ferramenta Composer 46
5.4.4. Exemplos 47
6. Construindo aplicações para a TV Digital 49
6.1. Descrição da aplicação Ginga-NCL 49
6.2. Código NCL 51
6.3. Dificuldades encontradas 63
7. Conclusões Finais 65
8. Bibliografia 67

Índice de Figuras

Figura 1 - Sistema de TV digital. Fonte: (Soares & Barbosa, 2008) 5
Figura 2 - Padrões de Referência. Fonte: (Soares & Barbosa, 2008) Adaptada. 7
Figura 3 - Receptor de TV Digital. Fonte: (Soares & Barbosa, 2008) 12
Figura 4 - Padrões utilizados no SBTVD. Fonte: (Soares & Barbosa, 2008) Adaptada. 20
Figura 5 - Padrã de referência do SBTV. Fonte: (Telemidia-PUC-Rio) Adaptada 29
Figura 6 - Arquitetura do Ginga-CC. Fonte: (Filho, Leite, & Batista, 2007) 31
Figura 7 - Contexto do Ginga-J. Fonte: (Filho, Leite, & Batista, 2007) Adaptada 34
Figura 8 - APIs do Ginga-J. Fonte: (Filho, Leite, & Batista, 2007) Adaptada 35
Figura 9 - Interface *Xlet* 36
Figura 10 - Ciclo de Vida de um Xlet.Fonte: (Morris) 36
Figura 11 - Estrutura básica de um documento NCL. Fonte: (Neto, Soares, Rodrigues, & Barbosa, 2007) Adaptada. 42
Figura 12 - Exemplo de código NCLua 46
Figura 13 - Ferramenta de autoria Composer. Fonte: (Neto, Soares, Rodrigues, & Barbosa, 2007) 47
Figura 14 - Programa Mosaico da Produtora de Multimeios da UFJF 50
Figura 15 - Início do programa com ícone de interação 61
Figura 16 - Painel interativo com foco no primeiro botão 62
Figura 17 - Painel interativo com foco no botão "receita" 62

Índice de Tabelas

Tabela 1 - Tipos de Configuração de Vídeo 9
Tabela 2 - Eletrodomésticos por lares brasileiros. Fonte: IBGE 2003. 18
Tabela 3 - Acesso à internet por classe social. Fonte: Anatel (Jul-03) 19
Tabela 4 - Codificação do áudio do sistema brasileiro de TV digital 20
Tabela 5 - Codificação do vídeo do sistema brasileiro de TV digital 21

Índice de Listagens

Listagem 1 - Base de regras 52
Listagem 2 - Base de regiões 52
Listagem 3 - Base de descritores 54
Listagem 4 - Base de conectores 55
Listagem 5 - Porta do contexto principal 56
Listagem 6 - Descrição das mídias 57
Listagem 7 - Contexto "painel" 58
Listagem 8 - Elo que inicia o ícone de interatividade 58
Listagem 9 - Elo que inicia o painel interativo 59
Listagem 10 - Elo que redimensiona o vídeo ao inciar o painel de interação 59
Listagem 11 - Elos referentes ao primeiro bloco do programa 60
Listagem 12 - Elos que definem o comportamento ao fechar o painel interativo 61

Resumo

Neste trabalho apresentamos o sistema brasileiro de TV digital (SBTVD), desenvolvido a partir do Decreto Presidencial 4.901, de 26 de novembro de 2003. Em especial, focamos na funcionalidade do seu *middleware*, denominado Ginga, e nas linguagens de programação utilizadas para o desenvolvimento de conteúdo interativo, Ginga-J e Ginga-NCL. Por fim, faremos uma análise dos principais problemas ainda existentes em Ginga, que foram encontrados durante o desenvolvimento de conteúdo interativo utilizando Ginga-NCL.

1. Introdução

A TV Digital ganhou projeção nacional nos últimos meses com o anúncio do padrão a ser adotado no Brasil. Sua adoção se dará gradativamente, porém, pode-se adiantar que seus benefícios serão grandes. Alguns autores comparam a mudança na transmissão da TV de analógico para digital, com a mudança da TV preta e branca para a TV colorida.

O uso da TV Digital cresce no mundo e traz diversas vantagens tanto para os provedores quanto para os telespectadores. No Brasil, em especial, ela também representa uma forma de promover a inclusão digital. A TV Digital oferece diversas vantagens. A primeira delas, melhor percebida pelo telespectador, é a qualidade da imagem. Dada a natureza discreta do conteúdo digital, é possível realizar uma recuperação muito precisa a partir do sinal recebido. Sendo assim, quando esse procedimento é feito corretamente, o conteúdo recebido é o mais próximo possível daquele que foi transmitido. De forma similar, a qualidade do áudio também sofre melhoras e mais canais de áudio também podem ser oferecidos. Isso possibilita o uso de tecnologias, como o *surround*, ou a escolha de línguas diferentes durante a exibição de um determinado programa. Outra vantagem da TV Digital é permitir que o conteúdo transmitido seja facilmente armazenado e com a mesma qualidade do sinal transmitido. Assim, o telespectador poderia, por exemplo, pausar ou solicitar um *replay*, entre outras funções de PVR (*Personal Video Recorder*), mesmo em transmissões feitas ao vivo. Essa nova forma de se prover TV é, além de tudo, uma maneira de oferecer conteúdo personalizado e interatividade para os telespectadores.

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

Ginga é o padrão brasileiro de *middleware* para TV Digital. Resultado de anos de pesquisas lideradas pela Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) e pela Universidade Federal da Paraíba (UFPB), Ginga reúne um conjunto de tecnologias e

inovações brasileiras que o tornam a especificação de middleware mais avançada e, ao mesmo tempo, mais adequada à realidade do país.

Este trabalho apresenta um apanhado geral sobre a TV digital e tecnologias envolvidas e está dividido com se segue: no capítulo 2 são tratados os conceitos e tecnologias envolvidas no ambiente de TV digital. O capítulo 3 descreve um histórico da TV digital no Brasil. O conceito de *middleware* e trabalhos relacionados são apresentados no capítulo 4. Com isso podemos demonstrar o Ginga no capítulo 5. No capítulo 6 são apresentadas as ferramentas para a construção de aplicações e as dificuldades encontradas na utilização destes. Por último o capítulo 7 apresenta as conclusões e trabalhos futuros.

2. A TV Digital

A TV digital nada mais é do que a digitalização da transmissão do sinal televisivo que estamos acostumados. Nas maiores operadoras brasileiras os programas de TV já são feitos de forma digital, apenas é transmitido de forma analógica. Nesse caso, todo processo é feito com equipamento digital: desde a filmagem, até a edição e pós-produção.

A TV Digital oferece inúmeras vantagens. Primeiramente, e melhor percebida pelo telespectador, é a qualidade da imagem. De forma semelhante, a qualidade do áudio também é melhor e também é possível disponibilizar mais canais de áudio ao telespectador. Isso possibilita o uso de tecnologias, como o *surround*, ou a escolha de línguas diferentes durante a exibição de um determinado programa.

No sistema de transmissão não guiado, que é o caso da transmissão de TV terrestre analógica e digital, o sinal original sofre diversas transferências e ruídos, limitando sua capacidade. O ruído está presente em todo o espectro de freqüência e não pode ser evitado. Na transmissão analógica provoca uma queda na qualidade do sinal e é percebido com o aparecimento de “chuviscos” na imagem. Na transmissão digital o ruído pode interferir em um nível digital de forma que este possa ser confundido com outro nível, causando a probabilidade de erro de bit. Códigos corretores de erros são utilizados com o intuído de se evitar tal problema. Caso a taxa de erros não seja muito grande, o código corretor é capaz de corrigir o problema, porém, se a taxa é alta, o código pode até introduzir novos erros. Assim, na TV digital, temos um sinal perfeito, com imagens sem “chuviscos”, ou não temos o sinal.

A melhor qualidade de som e imagem também é devido às técnicas de compressão utilizadas. A TV analógica possui uma qualidade de imagem próxima à da TV digital padrão (SDTV). Nesta, são gerados 29,97 quadros por segundo, cada quadro composto por 480 linhas ativas entrelaçadas e 704 pixels por linha, em uma TV com relação de aspecto igual a 4:3. Isto gera aproximadamente 162 Mbps. Essa taxa de bits deve ser transmitida em um canal de TV de 6MHz (no caso do Brasil), que suporta uma taxa de bits de 19,3 Mbps. As técnicas de compressão permitem isto e muito mais, podendo ter uma transmissão de alta qualidade (HDTV) com aproximadamente 30 quadros por segundo, 1920 pixels por linha e

1080 linhas. Também é possível com isto a transmissão do áudio no padrão 5.1 dentro desta mesma faixa de 6MHz (Soares & Barbosa, 2008).

As técnicas de compressão digital também permitem o envio de vários programas com menor qualidade (SDTV) em um mesmo canal no lugar de apenas um programa de alta qualidade (HDTV). Assim temos a chamada *multiprogramação*. Tais programas podem ser disponibilizados por uma mesma operadora e constituírem um só programa com informações adicionais. Um exemplo disto seria um programa com visões diferentes de uma mesma cena, por exemplo, uma partida de futebol com seleção de câmeras ou um filme com ângulos de visão baseado nos personagens. Ou podem ser programas independentes, sem nenhum relacionamento, o que nos leva a revisar o conceito de canal. Na TV analógica, o canal de 6 MHz (canal de freqüência) era confundido com o canal de programação (associado a uma radiodifusora). Agora, em um mesmo canal de freqüência é possível ter vários programas diferentes.

No uso do sinal analógico também há interferência no canal de freqüência adjacente, desta forma não é possível utilizá-los, sendo necessária a adoção de canais com um salto entre eles. Por este motivo nossas TVs são configuradas mostrando os canais 2, 5, 7, 10 como seqüência, por exemplo. Com o sinal digital é possível aproveitar todo o espectro. Juntando isso à multiprogramação, é possível divulgar uma gama muito maior de conteúdo de uma forma mais democrática.

Além de fornecer melhor qualidade de som e imagem e possibilidade de transmitir diversos programas em um mesmo canal, seus sinais poderão ser capturados por diversos tipos de equipamentos, como em um telefone celular. E, quem sabe, em um futuro próximo, poderemos assistir ao jogo de nosso time preferido na tela de nosso *PDA*, no relógio de pulso, em qualquer lugar que se esteja: em casa, dentro do carro, ônibus e assim por diante.

O impacto da TV digital é muito mais significativo, no entanto, do que a simples troca de um sistema de transmissão analógico para o digital, e muito mais do que uma melhora de imagem e de som. Além destas mudanças, adiciona-se a questão da interatividade, talvez a maior mudança no já costumeiro ambiente passivo no qual o telespectador assiste a seu programa de TV. Assim, uma das características mais importantes da TV digital é a integração de uma capacidade computacional significativa no dispositivo receptor,

permitindo o surgimento de uma vasta gama de novos serviços, como a oferta de guias eletrônicos de programas, o controle de acesso e a proteção de conteúdo, a distribuição de jogos eletrônicos, o acesso a serviços bancários (*T-banking*), serviços de saúde (*T-health*), serviços educacionais (*T-learning*), serviços de governo (*T-governmnet*) etc.

Além de tudo isso, a TV pode se tornar um elemento de extremo interesse para o comércio de grande porte, simplesmente pelo fato da maioria dos domicílios possuírem um televisor ou mais. Fica claro que este ramo não deve ser ignorado. Seu potencial é grande, e muito conteúdo digital, desde entretenimento até negócios vai ser demandado, cabendo a nós, desenvolvedores, suprir estas tendências de mercado.

2.1. Sistema de TV digital

Segundo (Soares & Barbosa, 2008), um sistema de TV digital é um sistema típico cliente/servidor. O servidor compõe o ambiente de uma radio difusora (parte esquerda da Figura 1) ou de um servidor de conteúdo, e o cliente o ambiente do usuário telespectador (parte direita da Figura 1).

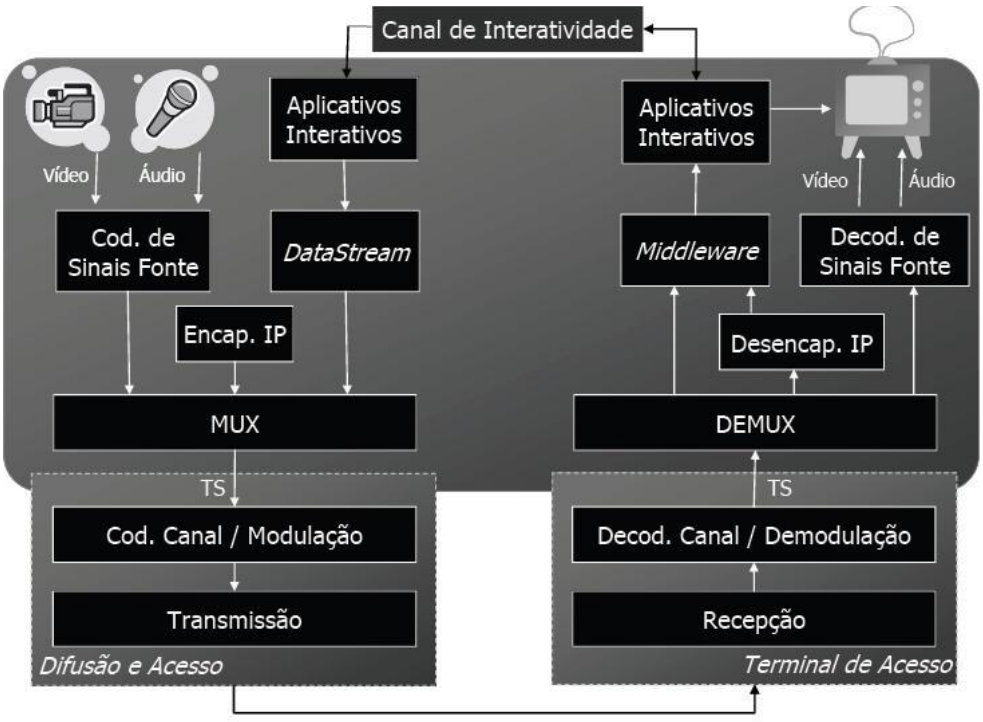

Figura 1 - Sistema de TV digital. Fonte: (Soares & Barbosa, 2008)

O vídeo e o áudio principal de um programa são capturados através de equipamentos de produção já utilizados para este fim pelas produtoras de conteúdo. Estes sinais, que podem ser capturados ao vivo ou provenientes de um repositório, são então codificados para a geração do fluxo de vídeo e áudio digitais comprimidos. Junto a este sinal são adicionados os aplicativos interativos e outros dados utilizados pelo programa. Todos estes dados são multiplexados em um fluxo de transporte (*Transport Stream*), modulados e transmitidos através de um canal de difusão.

Os dados chegam a um terminal de acesso através de um receptor e são demodulados e entregues a um demultiplexador. Este demultiplexador separa os sinais de áudio e vídeo principal dos dados e os envia diretamente à tela para exibição. Os dados adicionais são entregues ao *middleware* que é encarregado de interpretá-los. O aplicativo interativo é então iniciado e controlado pelo *middleware*, podendo enviar ou requisitar dados adicionais à emissora através de um canal de retorno.

2.1.1. Padrões internacionais

O Japão discute a digitalização da transmissão do sinal televisivo desde 1995, porém os primeiros testes de transmissão só ocorreram em 1999, com a criação do ISDB (*Integrated Services Digital Broadcasting*) um sistema totalmente digital, que entrou em operação na cidade de Tóquio, em regime experimental, no final do ano 2000. Além de Tóquio, as cidades de Osaka e Nogoya passaram em 2003 a ter os sinais de radiodifusão emitidos em formato digital. Em 2006 as imagens da TV digital terrestre passaram a ser captadas, também em regime experimental, por aparelhos celulares e receptores móveis, na cidade de Tóquio.

O DVB, sistema Europeu de TV digital, tinha por objetivo desenvolver um sistema digital completo baseado num único padrão para vários países, que considerasse as características específicas de cada região. No final de 1995 ocorreram as primeiras transmissões digitais na Europa.

Nos EUA, as primeiras transmissões digitais iniciaram-se no final de 1998. No ano 2000, em face das dificuldades de recepção do sistema ATSC, a rede *Sinclair Broadcasting*, dos Estados Unidos, propôs a revisão do padrão e a adoção do DVB nos EUA. A FCC (*Federal*

Communications Commission) após realizar testes comparativos, optou por manter o padrão ATSC.

"Um sistema de TV digital é composto por um conjunto de padrões que regulam cada um dos procedimentos descritos na Figura 2, chamados padrões de referência" (Soares & Barbosa, 2008).

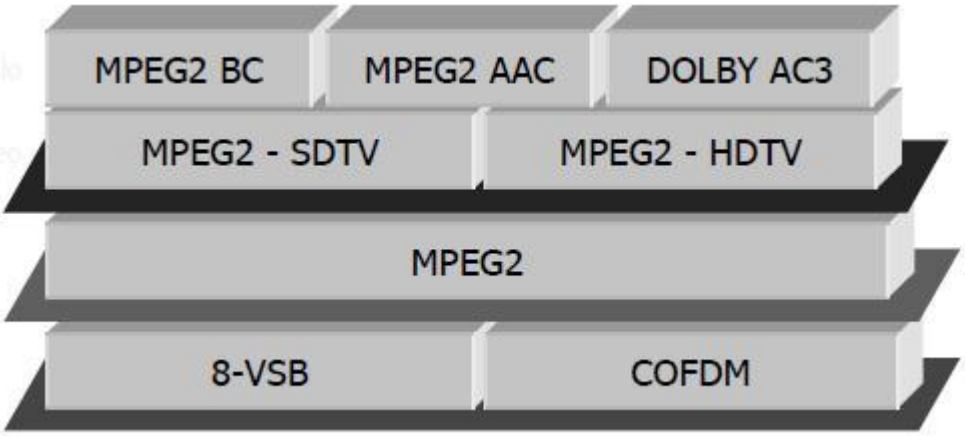

Figura 2 - Padrões de Referência. Fonte: (Soares & Barbosa, 2008) Adaptada.

2.1.2. Codificação do Áudio

A codificação de áudio de um sistema de TV digital tem como objetivo reduzir a taxa de bits de um sinal de áudio para sua transmissão, de modo que, quando decodificado e reproduzido seja praticamente indistinguível do original pelo ouvido humano.

Este subsistema compreende um modulo codificador e um decodificador. O codificador recebe como entrada o sinal de áudio original, realiza a compressão e gera um fluxo de áudio codificado para a camada de transporte do sistema. O decodificador recebe o fluxo de áudio codificado do demultiplexador, realiza sua decodificação e disponibiliza o áudio reconstruído.

Para o sistema americano de TV digital é utilizado o padrão de codificação AC3/ATSC. A codificação utiliza 5 canais de freqüência normal (20Hz a 20KHz) e um de baixa freqüência (20Hz a 120Hz). (Advanced Television Systems Committee, Inc., 2005)

O sistema europeu usa o padrão MP2 (MPEG-1 Layer 2), mas algumas implementações seguem o padrão MPEG-2 AAC, também adotado pelo padrão japonês.

2.1.3. Codificação do Vídeo

A vantagem mais perceptível na TV digital é a qualidade do vídeo apresentado. Isto se dá pela alta resolução atingida na transmissão dos sinais digitais. Para uma comparação, na TV analógica o número de linhas horizontais no canal de recepção é 360, em média, com 480 pontos distribuídos em cada uma dessas linhas. A menor resolução da TV digital para dispositivos fixos é de 480 linhas horizontais e a maior atinge impressionantes 1.080 linhas pro 1.920 pixels, ou seja, doze vezes mais nitidez do que a TV analógica. Outro fator de medição da qualidade é a taxa de quadros, que define a quantidade de quadros por segundo necessários para que uma imagem fique completa. Na TV convencional esta taxa é de 24 quadros por segundo entrelaçados (preenchem apenas metade das linhas, necessitando de duas passagens) enquanto que na TV digital esta taxa chega a 60 quadros por segundo progressivos (exibe a imagem inteira a cada quadro).

As resoluções encontradas na televisão digital são:

- SDTV (*Standard Definition Television*),
- HDTV (*High Definition Television*),
- LDTV (*Low Definition Television*).

SDTV é a qualidade mais parecida com a TV convencional. Possui 480 linhas, podendo ter o aspecto *widescreen* (16:9) ou normal. A HDTV é a melhor qualidade de imagem em uma TV digital. Pode ser HDTV com 720 linhas ou a chamada *full-HD*, que é a HDTV com a a resolução de 1920x1080. Para dispositivos portáteis existe a LDTV, que é uma qualidade inferior de definição. Devido à natureza destes dispositivos de possuírem telas muito menores este modo se torna ideal, pois é possível realizar a transmissão sem que toda a capacidade do canal de freqüência seja utilizada sendo, inclusive, transmitida em paralelo com outras modalidades de transmissão.

Tipo de Configuração	Número de linhas	Número de pixels por linha	Formato de tela
HDTV	1080	1920	16:9
	720	1280	16:9
SDTV	480	704	16:9
	480	640	4:3
LDTV	240	320	4:3

Tabela 1 - Tipos de Configuração de Vídeo

A codificação de vídeo busca reduzir a taxa de bits necessária para transmissão do sinal de vídeo. Isto é atingido através do processo de codificação do sinal que remove redundâncias (espacial, temporal e estatística) do sinal de vídeo. É uma tecnologia chave para a TV digital, já que a taxa de bits necessária para representar um sinal de vídeo digitalizado sem compressão pode chegar a 1,5 Gbps para sinais de HDTV, muito maior que os 19,2 Mbps suportados em um canal de freqüência de 6 MHz, por exemplo.

Seu funcionamento é semelhante à codificação de áudio. Os sistemas americano, europeu e japonês empregam o MPEG-2 video como padrão para codificação de vídeo.

2.1.4. Sistema de transporte

A camada de transporte situa-se entre o subsistema de codificação de sinais e o subsistema de codificação de canal e modulação.

Tanto os sistemas americano, europeu e japonês quanto o sistema brasileiro de TV digital padronizam a forma como informações audiovisuais, juntamente com os dados, devem ser multiplexadas em um único fluxo, adotando o mesmo padrão de multiplexação MPEG-2 System.

2.1.4.1. Carrossel de Dados

Quando um usuário acessa um canal televisivo este começa a receber instantaneamente as informações de áudio e vídeo deste canal. Mas como transmitir as aplicações, que possuem tamanhos variados? Se estas forem transmitidas no início do programa televisivo, um

usuário que começou a assistir tal programa após o seu início não terá acesso ao conteúdo interativo. Desta forma é utilizado o carrossel de dados para o envio dos dados necessários às aplicações interativas.

Os dados são transmitidos de maneira cíclica aos receptores pelo carrossel de dados através dos fluxos MPEG2. Assim, quando um usuário acessar o canal de transmissão, este começa a receber os dados imediatamente, da mesma forma que recebe o vídeo e o áudio. Quando o receptor obtiver os dados necessários o aplicativo é iniciado. Caso este aplicativo necessite de um dado específico ele deve aguardar que este dado seja retransmitido pelo carrossel.

É necessário que o funcionamento do carrossel seja otimizado de maneira a iniciar o aplicativo mais rapidamente possível, pois uma demora no início irá testar a paciência do telespectador e este pode desistir de utilizar o conteúdo interativo. Para isso arquivos maiores e usados com menos freqüência são transmitidos mais raramente do que os arquivos menores e os necessários para iniciar a aplicação.

2.1.5. Modulação

Processa o sinal codificado para que seja possível a sua transmissão com ocupação espectral limitada e com robustez às adversidades do canal de transmissão. Modulação é o processo no qual a informação a transmitir numa comunicação é adicionada a ondas eletromagnéticas. O transmissor adiciona a informação numa onda básica de tal forma que poderá ser recuperada na outra parte através de um processo reverso chamado demodulação.

O padrão americano ATSC de TV digital utiliza o sistema 8-VSB (*Vestigial Side Band*) para modulação do sinal. Já o sistema europeu (COFDM/DVB-T), japonês (BST-OFDM/ISDB-T) e brasileiro (BST-OFDM/SBTVD-T) utilizam um sistema baseado no OFDM (*Orthogonal frequency-division multiplexing*), que apresenta melhores resultados para transmissão terrestre, ou seja, via radiofreqüência.

2.2. Canal de Retorno

A comunicação realizada entre o receptor e as redes de telecomunicações é denominada canal de interação, de interatividade ou de retorno. Este canal é utilizado para permitir a comunicação em duas vias entre o usuário e a rede de televisão. Todos os dados que não são obtidos através do canal de descida e os dados transmitidos pelo usuário passam por este canal. Através dele é possível utilizar aplicações que permitem uma interação remota, como enquetes e emails, e o acesso a todos os serviços disponíveis na internet.

Um sistema de TV digital pode operar sem o canal de retorno. Nesse caso, as aplicações podem usar apenas dados transmitidos por difusão. Caso as aplicações permitam a interação com o usuário, o serviço é chamado de interatividade local.

Para a prática do canal de retorno pode ser usado qualquer tipo de rede de comunicação de dados, como linha telefônica, *Wi-Fi*, *WiMAX*, rede elétrica (PLC – *Power Line Communications*) etc.

2.3. Set-top Box

Existe um grande legado de TVs analógicas nas casas dos usuários. Estas TVs não podem sintonizar o sinal digital, funcionam apenas com o sinal analógico. Para atender este nicho são utilizados receptores com poder de processamento que são encarregados de tratar o sinal digital recebido e exibir as informações na tela da TV analógica. Estes receptores digitais são chamados de *set-top box*. A Figura 3 representa o diagrama em blocos de um receptor de TV digital, que pode estar embutido ou não em um aparelho de televisão. Note que o receptor implementa todos os sistemas tratados nas subseções anteriores.

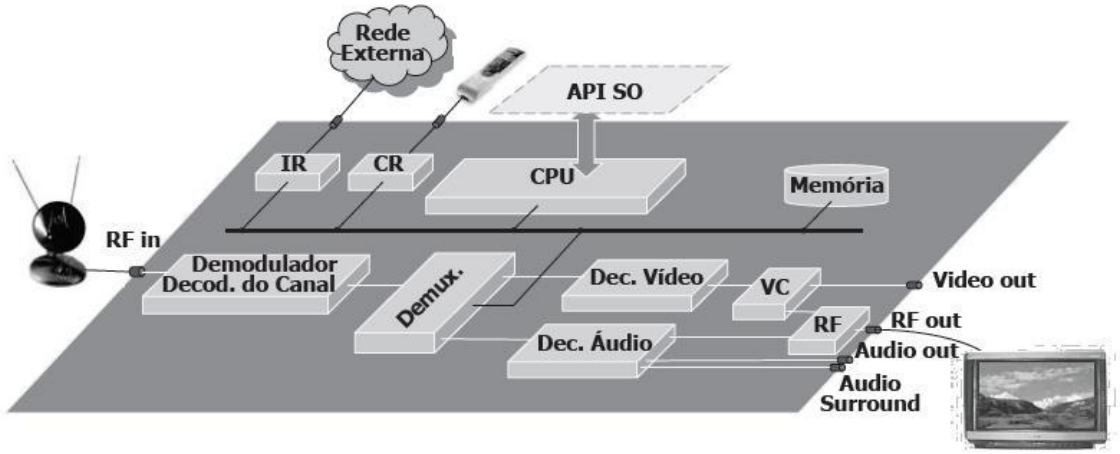

Figura 3 - Receptor de TV Digital. Fonte: (Soares & Barbosa, 2008)

Um terminal de acesso de TV digital, para oferecer um custo acessível, possui forte limitação na quantidade de recursos computacionais e hardware especializado para a exibição de conteúdo televisivo. Um terminal de acesso típico geralmente possui um processador de baixo custo para executar as tarefas das aplicações e do sistema operacional (possivelmente Linux ou Windows CE) e pouca memória para uso das aplicações. Obviamente, podem existir terminais mais sofisticados com capacidade de processamento mais alta para suportar aplicações mais complexas, disco rígido com capacidade de armazenamento para gravar os programas transmitidos, ou mesmo armazenar aplicações oferecidas pelos provedores de conteúdo. Entretanto, melhoras como essas aumentam sensivelmente o custo do terminal de acesso.

Ainda nos terminais de acesso, é comum existir uma camada de abstração, denominada *middleware*, responsável por esconder das aplicações a complexidade dos mecanismos definidos pelos padrões, protocolos de comunicação e até mesmo sistema operacional do equipamento[1].

2.4. IPTV

A IPTV é um exemplo do caso onde a tecnologia de TV é levada para o ambiente computacional. A sigla IPTV quer dizer *TV Over IP*, e significa que a TV é distribuída sobre o

[1] Atualmente, no Brasil, os set-top boxes ainda não possuem a camada de *middleware*. Desta forma não permitem interatividade, funcionando apenas como receptor dos canais (*zapper*).

protocolo da Internet, o IP. Isso não quer dizer que a TV é oferecida pela Internet, método que é chamado de Internet TV. Na verdade, os serviços de IPTV existentes são oferecidos em redes fechadas, parecido com o que é feito nas TVs por assinatura. Essas redes possuem tanto um canal de *downstream*, por onde as informações são recebidas, quanto um de *upstream*, por onde informações são enviadas para o provedor. A IPTV encontra-se, portanto, mais próximo do modelo da Internet, e isso permite que serviços, como o vídeo sob demanda, sejam mais facilmente oferecidos. Os serviços comumente encontrados para IPTV são:

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

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

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

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

3. Contextualização no Brasil

Os parágrafos seguintes são descritos de acordo com (Becker & Moraes, 2003).

A Sociedade Brasileira de Engenharia e Televisão – SET e a Associação Brasileira de Emissoras de Rádio e Televisão – ABERT estruturaram os primeiros estudos sobre a TV digital no Brasil. A partir desta data formou-se um grupo, fundado nestas duas associações, com o intuito de pesquisar a passagem do sistema analógico para o sistema digital.

Diante da abertura da Consulta Pública nº 65, de 27 de julho de 1998, a ANATEL (Agência Nacional de Telecomunicações) iniciou o procedimento para a escolha do modelo oficial de TV digital. Diversas emissoras de televisão manifestaram interesse em participar deste processo. A partir de então, começou a ação de contratação de especialistas do assunto com o intuito de dar consultorias aos pesquisadores.

Em 1999 foram importados e testados equipamentos dos três padrões internacionais mais estabilizados, o americano, o europeu e o japonês. Os testes finalizaram em 2000 e o governo da época tinha a clara intenção de escolher um padrão a ser estabelecido no país. O padrão japonês apresentou o melhor desempenho entre os padrões analisados por possuir um sistema de transmissão terrestre de alta qualidade e de freqüência idêntica à utilizada no Brasil, sendo necessárias apenas pequenas mudanças adaptativas para a implantação. Outro fator que pesou nesta decisão foi o modelo de transmissão para dispositivos portáteis que já se encontrava bastante avançado no Japão. Com o fim do mandato do governo vigente, a decisão sobre o sistema a ser adotado transferiu-se ao novo comando eleito.

O novo governo, com claras tendências sociais, dá um novo rumo às discussões com o intuito de esquadrinhar um novo padrão, com tecnologias brasileiras e soluções de cunho social.

Assim, é instituído o SBTVD. Ele se propõe ao desenvolvimento de um sistema que culmine num modelo brasileiro de televisão digital, com todos os serviços agregados que a nova tecnologia permite, mas com foco voltado para os interesses sociais.

3.1. SBTVD

A instituição do Sistema Brasileiro de Televisão Digital - SBTVD deu-se pelo Decreto Presidencial 4.901, de 26 de novembro de 2003. Ao convocar pesquisadores e técnicos brasileiros na área e representantes da sociedade civil vinculados ao setor, para participarem da construção deste importante projeto, o atual governo dava sinais de uma visão sobre o tema, completamente diversa daquela do governo liberal anterior, que tinha a intenção declarada de escolher um, dentre os três padrões internacionais: americano, europeu e japonês.

Por tratar-se de uma tecnologia digital, completamente diferente da esgotada tecnologia analógica dos televisores existentes no país, o SBTVD ambicionava um modelo capaz de priorizar a urgente questão da inclusão social, explicitada no decreto que o instituiu. Este requisito é inexistente na concepção dos padrões internacionais, com exceção de alguns poucos países que utilizam o padrão europeu. Esta foi, e continua sendo, a principal motivação do SBTVD.

Assim, enquanto os padrões internacionais privilegiavam aspectos associados à tecnologia empregada em seus modelos (alta definição, mobilidade etc.) o SBTVD perseguia o objetivo de priorizar a característica de interatividade, a fim de dotar a TV brasileira, presente em 90% dos lares, de serviços digitais (educação à distância, informações governamentais, acesso a serviços de saúde etc.) acalentando o sonho de permitir ao brasileiro comum o acesso à revolucionária Internet.

3.2. Decreto n.º 4.901

O Decreto 4.901 está baseado em um modelo social ímpar, no qual se destaca:

I. Promover a inclusão social, a diversidade cultural do País e a língua pátria por meio do acesso à tecnologia digital, visando à democratização da informação.

II. Propiciar a criação de rede universal de educação a distância.

III. Estimular a pesquisa e o desenvolvimento e propiciar a expansão de tecnologias brasileiras e da indústria nacional relacionadas à tecnologia de informação e comunicação.

Para alcançar os objetivos estabelecidos no Decreto Presidencial 4.901 de 26 de novembro de 2003, que instituiu o SBTVD, foi designado: o Comitê de Desenvolvimento, composto por nove Ministros de Estado e a Secretaria de Comunicação de Governo e Gestão Estratégica da Presidência da República; o Comitê Consultivo, composto por entidades representantes da sociedade civil; e o Grupo Gestor, composto por representantes de seis Ministérios e do Instituto Nacional de Tecnologia da Informação (ITI), da Secretaria de Comunicação de Governo e Gestão Estratégica da Presidência da República e da ANATEL. O Grupo Gestor, assessorado pelo Comitê Consultivo, tem a finalidade de definir as políticas da TV digital.

O Grupo gestor delimitou os objetivos estabelecidos pelo Decreto 4.901, de 2003, expostos no art. 1º em três macrofinalidades:

I. Inclusão social,

II. Flexibilidade dos modelos de exploração,

III. Desenvolvimento sustentável do serviço e da indústria correlata.

A análise conduzida no âmbito do Grupo Gestor acerca dos riscos e das oportunidades associados à implantação da TV digital terrestre, no Brasil, foi feita levando em consideração o impacto de cada evento (risco ou oportunidade) nessas macrofinalidades (CPqD, 2006).

O governo lançou 22 editais de pesquisa para o desenvolvimento de componentes que juntos formariam o SBTVD. Os trabalhos de pesquisa foram realizados por 22 consórcios compostos de 72 universidades, 34 instituições de pesquisa e empresas privadas, envolvendo quase mil e quinhentos pesquisadores distribuídos em 106 instituições, em onze Estados mais o Distrito Federal, sob a coordenação técnica da Fundação CPQD e gestão financeira do FINEP.

Foram feitas pesquisas em todo o Brasil nas diversas camadas de um sistema de TV digital, desde transmissão ao *middleware*. Mesmo com muitos desacreditando no potencial

brasileiro, devido ao pouco tempo disponível e por ser um país de terceiro mundo, ótimos resultados foram alcançados, porém nem tudo foi aproveitado.

3.2.1. Questão social

Sem dúvida a televisão é hoje, para nós brasileiros, a forma mais abrangente de distribuição de conteúdo. Como visto na Tabela 2, ela está presente em grande parte dos domicílios brasileiros, alcançando um número maior do que o numero de geladeiras e já ultrapassando o numero de rádios presentes. Também é importante ressaltar a pequena quantidade de microcomputadores existentes. Desta forma, a TV se consolida como principal meio de acesso à informação da população. A TV digital amplia mais ainda este espectro, uma vez que não apenas o áudio e vídeo principal de um programa televisivo poderão ser difundidos, como também outros dados. É assim imperativo que o Brasil tenha o domínio das tecnologias da informação nas inúmeras atividades que envolvem o acesso e a capacidade de gerar e apresentar informações televisivas, e que a absorção dessas tecnologias sirva para diminuir o desequilíbrio regional e social do país.

Tecnologia	Domicílios - Total: 47.558.659	Domicílios %
Fogão	46.481.918	97,7
Televisão	42.778.810	89,9
Rádio	41.775.232	87,9
Geladeira	41.215.385	86,7
Telefone	29.319.600	61,6
Microcomputador	6.743.522	14,2
Acesso à Internet	4.912.732	10,3

Tabela 2 - Eletrodomésticos por lares brasileiros. Fonte: IBGE 2003.

A TV digital possui várias características que possibilitam uma maior democratização na geração e distribuição de conteúdo. Começa pelo potencial aumento no número de canais em um mesmo espectro de freqüência, o que permite a presença de novos atores, tais como TVs universitárias, TVs comunitárias etc. Passa pela alternativa de conteúdo, que permite que um mesmo programa possa ter seu conteúdo modificado, dependendo do usuário, de seu aparelho receptor ou da região onde é exibido. Entre inúmeras outras possibilidades,

termina pelo oferecimento de “aplicações cidadãs”, tais como governo eletrônico e aplicações na área de educação e saúde.

Assim será possível disponibilizar aplicações governamentais. O usuário poderia, por exemplo, participar de alguma enquete sobre alguma lei em trâmite no congresso. Na área da saúde poderá marcar consultas em hospitais do SUS. Um novo sistema de educação a distância poderá ser instaurado também, com melhores resultados devido à maior interação disponível no novo meio.

Outro fator necessário para a inclusão digital seria um maior acesso à internet pela população. Na Tabela 3 podemos ver que a maioria esmagadora dos usuários da internet no Brasil situa-se nas classes A e B, que representam 24% do total da população. A outra parte, as classes C, D e E, correspondem a 9,3% dos usuários da internet. Pelo *set-top box* será possível também o acesso à grande rede através de um *browser* presente no aparelho, disponibilizando o acesso à informações e serviços já existentes para o resto da população que não tem acesso a um microcomputador em casa.

Classe de Renda	População (%)	Usuários da Internet (%)
A	5	42
B	19	48,7
C	31	9,3
D	32	
E	13	

Tabela 3 - Acesso à internet por classe social. Fonte: Anatel (Jul-03)

3.3. Decreto n.º 5.820

Após inúmeras pesquisas e tecnologias desenvolvidas, foi iniciado o processo de implantação do SBTVD com a publicação do Decreto 5.820, em 29 de junho de 2006. Com isso ficou-se decidido que o país utilizaria um sistema baseado no sistema japonês, porém com tecnologias tupiniquins incorporadas a este. Também ficou definido que seria designado às emissoras um novo canal de radiofreqüência a fim de permitir a transição para a tecnologia digital sem a interrupção da transmissão dos sinais analógicos.

Fica assim decretado que o SBTVD será baseado no ISDB-T, o sistema japonês de transmissão terrestre. Desta forma, o SBTVD é também conhecido como ISDB-TB. Como visto na Figura 4, foram utilizados os mesmos padrões para as camadas de modulação e transporte do sistema japonês.

Figura 4 - Padrões utilizados no SBTVD. Fonte: (Soares & Barbosa, 2008) Adaptada.

Para a codificação de áudio principal de um programa, o sistema brasileiro de TV digital adotou o padrão MPEG-4 (ABNT - Associação Brasileira de Normas Técnicas, 2008), com as características apresentadas na Tabela 4. Este sistema é considerado o estado da arte para áudio de alta qualidade, obtendo um desempenho muito melhor que os padrões utilizados nos sistemas internacionais. Ele é baseados em perfis, podendo ser utilizado tanto na transmissão para dispositivos portáteis, que não precisam de alta fidelidade sonora, quanto na transmissão para dispositivos fixos e móveis, com qualidade melhor dependendo do receptor.

	Receptores Fixos e Móveis	Receptores Portáteis
Padrão	ISSO/IEC 14496-3 (MPEG-4 AAC)	ISSO/IEC 14496-3 (MPEG-4 AAC)
Nível@Perfil	AAC@L4 (para multicanal 5.1) HE-AAC v1@L4 (para estério)	HE-AAC v2@L3 (dois canais)
Taxa de amostragem	48kHz	48kHz

Tabela 4 - Codificação do áudio do sistema brasileiro de TV digital

Para a codificação de vídeo, o sistema brasileiro emprega uma técnica mais recente que a dos outros padrões: o H.264, também conhecido como MPEG-4 Part 10 ou MPEG-4 AVC (*Advanced Video Coding*) (ABNT - Associação Brasileira de Normas Técnicas, 2008). O objetivo deste projeto foi criar um padrão capaz de prover um vídeo de boa qualidade a uma taxa substancialmente mais baixa do que os padrões anteriores (MPEG-2 entre eles), sem

aumentar muito sua complexidade, para facilitar uma implementação barata e eficiente. O H.264 é divido em perfis e níveis. No SBTVD é usado o perfil alto (HiP) para os receptores fixos e móveis e o perfil base (BP) para os receptores portáteis, conforme indica a Tabela 5.

	Receptores Fixos e Móveis	Receptores Portáteis
Padrão	ITU-T H.264 (MPEG-4 AVC)	ITU-T H.264 (MPEG-4 AVC)
Nível@Perfil	HP@L4.0	BP@L1.3
Número de linhas do nível	480 (4:3 e 16:9), 720 (16:9), 1080 (16:9)	SQVGA (160x120 ou 160x90), QVGA (320x240 ou 320x180) e CIF (352x288); todos em 4:3 e 16:9
Taxa de quadros	30 e 60 Hz	15 e 30 Hz

Tabela 5 - Codificação do vídeo do sistema brasileiro de TV digital

Também ficou definido que o SBTVD utilizaria o Ginga na camada de *middleware*. Este assunto será discutido em maiores detalhes nos capítulos posteriores.

3.4. Início das transmissões

No processo de transição no Brasil o canal analógico será mantido em funcionamento, enquanto o canal digital estiver sendo implantado.

O início das transmissões do SBTVD ocorreu no dia 02 de dezembro de 2007, em São Paulo. Depois foi a vez de Belo Horizonte, Rio de Janeiro, Goiânia, Curitiba e Porto Alegre. Porém ainda é realizada a transmissão sem interatividade, pois o *middleware* Ginga ainda não foi regulamentado para uso.

É provável que os outros países da América Latina adotem o SBTVD, garantindo a integração e a facilidade de suprimento de peças, equipamentos e soluções para o mercado de TV digital. Para alcançar esse objetivo o Fórum SBTVD e o governo brasileiro têm trabalhado em conjunto, realizando demonstrações do sistema em várias nações do continente.

Segundo (DTV), “É importante ressaltar que essa foi a primeira vez que se conseguiu realizar, no mundo, um trabalho de desenvolvimento e implementação de um sistema de TV

digital por meio de parceria entre emissoras de TV, fabricantes de transmissores, de receptores, a indústria de software, universidades, centros de pesquisa e o governo federal".

4. Middleware

Segundo (CPqD, 2006), “O *middleware* de um sistema de televisão digital corresponde a uma camada, em software, de abstração de hardware, sobre a qual são estruturadas as aplicações, independentemente das peculiaridades de hardware do terminal de acesso”.

Na televisão digital, o conteúdo pode ser multimídia e multiformato, diferente do que acontece com a televisão analógica, onde basta apenas apresentar o fluxo de áudio e vídeo captado pela antena. Desta forma, é necessário um receptor com capacidade de processamento que seja capaz de interpretar e gerar a exibição destas novas informações advindas deste novo cenário.

A padronização da interpretação destas informações adicionais e uma execução que seja independente do hardware utilizado é um grande problema deste novo paradigma de transmissão.

Uma solução para este problema foi a criação do *middleware*. Ele é o componente de software localizado entre a camada de hardware e as aplicações e define uma garantia semântica através de linguagens de programação padronizadas e suas APIs (*Application Program Interfaces*). Além disto, ele garante a abstração de hardware, podendo funcionar em equipamentos de diversos fabricantes diferentes.

As funcionalidades que um *middleware* deve prover podem ser resumidas em (CPqD, 2006):

- Carga, controle e reprodução adequada dos elementos de informação que compõem aplicativos associados a programas televisivos ou multimídia, considerando as características de não linearidade dos programas;
- Acesso às funcionalidades do terminal de acesso para as aplicações, através de uma máquina computacional virtual, provendo assim abstração do hardware;
- Acesso aos serviços (programa de televisão) e às suas informações;
- Segurança da informação, especialmente relacionados a: autenticidade e integridade das aplicações carregadas, envio e recepção de mensagens e armazenamento de informações;

- Suporte à interface com o usuário, com apresentação visual gráfica e tratamento de ações via controle remoto, teclado ou mouse;
- Armazenamento de informações, tais como: preferências do usuário e dados recebidos;
- Acesso ao canal de interatividade;
- Tratamento dos dados recebidos pelo canal de radiodifusão;
- Mecanismo para atualização do software residente no terminal de acesso.

4.1. Linguagem declarativa e procedural

O universo das aplicações para TV digital pode ser dividido em dois conjuntos: o das aplicações declarativas e o das aplicações procedurais.

Uma aplicação declarativa é aquela em que sua entidade "inicial" é do tipo "conteúdo declarativo". Analogamente, uma aplicação procedural é aquela em que sua entidade "inicial" é do tipo "conteúdo procedural".

Linguagens declarativas são linguagens em que o conteúdo da aplicação é baseado em uma descrição declarativa do problema. Este tipo de linguagem é mais utilizado por autores que não possuem conhecimento em linguagens de programação. Possuem um nível de abstração maior e são voltadas a um objetivo específico. Entre as linguagens declarativas mais comuns estão a NCL (*Nested Context Language*) (ABNT - Associação Brasileira de Normas Técnicas, 2008), SMIL (*Synchronized Multimedia Integration Language*) (W3C - World Wide Web Consortium, 2005) e XHTML (*eXtensible Hypertext Markup Language*) (W3C - World Wide Web Consortium, 2002).

Linguagens não declarativas são linguagens baseadas em uma implementação algorítmica do problema. Linguagens procedurais, a despeito do jargão utilizado no ambiente de TV digital, podem seguir diferentes paradigmas de programação, como orientação a objetos, programação baseada em componentes etc. Este tipo de linguagem requer maior especialização por parte do autor do que ao utilizar linguagens declarativas. A linguagem procedural mais comum em ambientes de TV digital é Java (Sun Microsystems).

4.2. Middlewares internacionais

No mundo existem diversos *middlewares* já em funcionamento. A seguir são apresentados os *middlewares* dos três principais padrões de TV digital existentes.

4.2.1. Middleware europeu

MHP (*Multimedia Home Platform*) (ETSI – European Telecommunications Standards Institute, 2005) é o *middleware* de TV Digital especificado pelo DVB (*Digital Video Broadcast*). Todas as especificações desenvolvidas pelo DVB são encaminhadas ao ETSI (*European Telecommunications Standards Institute* – que é o orgão ligado à União Européia responsável por padronizações na área de telecomunicações).

O MHP define o DVB-HTML e o DVB-J como seus ambientes declarativo e procedural, respectivamente. Porém uma aplicação MHP não precisa ser exclusivamente de um desses dois tipos, esta pode ser uma aplicação híbrida. Nestas aplicações é previsto elementos de ambas as linguagens, sendo prevista uma ponte que permite a comunicação entre esses dois paradigmas. É importante salientar, entretanto, que a implementação do DVB-HTML é opcional, já o DVB-J é um requisito obrigatório.

Aplicações DVB-HTML são baseadas em uma linguagem que estabelece uma extensão ao XHTML adotando, em conjunto com este, os padrões *Cascate Style Sheets* (CSS – responsável pelo layout e formatação das páginas HTML), *ECMAScripts* (responsável por habilitar a interatividade nos documentos XHTML) e *Document Object Model* (fornece a abstração que permite ao *ECMAScript* manipular estruturas e conteúdos de documentos XHTML).

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

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

Para estruturar uma padronização da plataforma e permitir que as aplicações criadas para o MHP possam ser rodadas em outros locais do mundo foi criado o GEM (*Globally Executable MHP*). O GEM consiste das APIs do MHP com exceção das que têm funcionalidades específicas do sistema europeu. Tais funcionalidades podem ser sobrescritas no sistema que é compatível com o GEM para que possa retratar as peculiaridades daquele padrão. Desta forma todo o trabalho criado na especificação do MHP está disponível para outros *middlewares* de todo o mundo.

4.2.2. Middleware americano

O *Advanced Television Systems Committee* (ATSC) é uma organização estadunidense responsavel por estabelecer os padrões para a TV. Inicialmente, foi desenvolvido pela ATSC um padrão denominado *DTV Application Software Environment* (DASE). O DASE adota modelos de aplicações baseados em linguagem procedural e declarativa. No ambiente declarativo é adotado XHTML e no ambiente procedural é adotado Java.

Paralelamente à evolução do DASE, desenvolveu-se o *OpenCable Applications Plataform* (OCAP) (OPENCABLE, 2005). Esse padrão foi desenvolvido pela CableLabs: um consórcio formado por membros da indústria da TV a cabo. Ele conta apenas com o ambiente procedural para execução de aplicações denominadas OCAP-J e desenvolvidas para a plataforma Java.

Dado o cenário mencionado, com dois tipos de *middlewares* despontando no território americano e a clara tendência à harmonização proposta pelo GEM, o ATSC introduziu a especificação do *Advanced Common Application Plataform* (ACAP) (ATSC - Advanced Television Systems Committee, 2005).

O padrão ACAP surgiu a partir de um trabalho em conjunto com participantes do DVB em setembro de 2004. Sua especificação é primariamente baseada no GEM e no DASE e inclui funcionalidades adicionais do OCAP. Foi criado com o intuito de substituir o DASE e permitir

uma compatibilidade com o GEM e com o OCAP. Dessa forma, além de permitir o desenvolvimento de aplicações portáteis entre o *middleware* americano e os compatíveis com o GEM, o padrão permite uma compatibilidade com as aplicações já existentes no sistema americano.

O conteúdo da aplicação pode ser de natureza declarativa ou procedural, sendo as que contem apenas conteúdo procedural chamadas de ACAP-J e as que possuem conteúdo declarativo chamadas de ACAP-X. O suporte ao ACAP-J é obrigatório e o suporte ao ACAP-X opcional, desta forma é reforçada a idéia de compatibilidade com o GEM. As aplicações ACAP-X são essencialmente documentos hipermídia consistidos de linguagem de marcação (XHTML), regras de estilos (CSS), scripts (ECMAScript), Xlets embutidos (aplicações ACAP-J) e elementos audiovisuais. Toda a definição desse ambiente é baseada na definição do DVB-HTML. O ambiente ACAP-J é uma combinação do GEM, OCAP e DASE.

4.2.3. Middleware japonês

O middleware padronizado pela ARIB ou *Association of Radio Industries and Business* (ARIB – Association of Radio Industries and Businesses, 2004) é utilizado no padrão Japonês ISDB. Assim com na maioria dos demais sistemas, a arquitetura ARIB é formada por dois subsistemas: um para execução de programas procedurais; e outro para a apresentação de programas declarativos. Entretanto, no padrão japonês não foram definidos elementos capazes de estabelecer uma ponte entre esses dois sistemas para a utilização de aplicações híbridas.

Esse *middleware* é formado por alguns padrões como o ARIB STD-B24 (*Data Coding and Transmission Specification for Digital Broadcasting*) que define linguagem declarativa denominada BML (*Broadcast Markup Language*). Essa linguagem, baseada em XML (*Extensible Markup Language*) é usada para especificação de serviços multimídia para TV digital e fornece suporte à folha de estilos (CSS) e scripts (ECMAScript). Outra especificação do *middleware* é o ARIB STD-B23 (*Application Execution Engine Platform for Digital Broadcasting*). Essa especificação é baseada no GEM, e indica uma tendência do ARIB de tentar estabelecer uma conformidade com outros padrões de *middleware*. Entretanto, ao

contrário dos outros padrões, existe no mercado japonês uma maior penetração de aplicações declarativas e receptores que implementam suporte a tal paradigma. Isto se deve, possivelmente, ao fato da padronização tardia do ambiente procedural.

Outra facilidade proporcionada pelo ARIB é que ele permite adicionar EPG (*Electronic Program Guide*), índice e funções de gravação automática para melhorar a seleção da programação, facilitando assim a programação pessoal do usuário.

4.3. Middleware brasileiro

No Brasil diversas pesquisas forma feitos com o intuito de criar um padrão de *middleware* totalmente brasileiro, visto que esta é uma tecnologia chave para a implantação da interatividade no sistema de TV digital, um requisito para sua implantação no país. Assim foram criados dois *middlewares*: um procedural pelos pesquisadores da UFPB (Universidade Federal da Paraíba) e outro declarativo pelos pesquisadores da PUC-Rio (Pontifícia Universidade Católica do Rio de Janeiro). Depois de unidos se transformou no *middleware* brasileiro para a TV digital que recebeu o nome de Ginga. Este assunto será discutido com mais detalhes no capítulo seguinte.

5. Middleware Ginga

"Ginga é o nome do *middleware* aberto do Sistema Brasileiro de TV Digital (SBTVD)" (LAVID-UFPB e Telemidia-PUC-RJ). O *middleware* Ginga é resultado de anos de pesquisas lideradas pela Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) e pela Universidade Federal da Paraíba (UFPB). "Ginga reúne um conjunto de tecnologias e inovações brasileiras que o tornam a especificação de *middleware* mais avançada e, ao mesmo tempo, mais adequada à realidade do país" (LAVID-UFPB e Telemidia-PUC-RJ). A Figura 5 ilustra o padrão brasileiro de TV digital completo, incluindo o Ginga na camada do *middleware*.

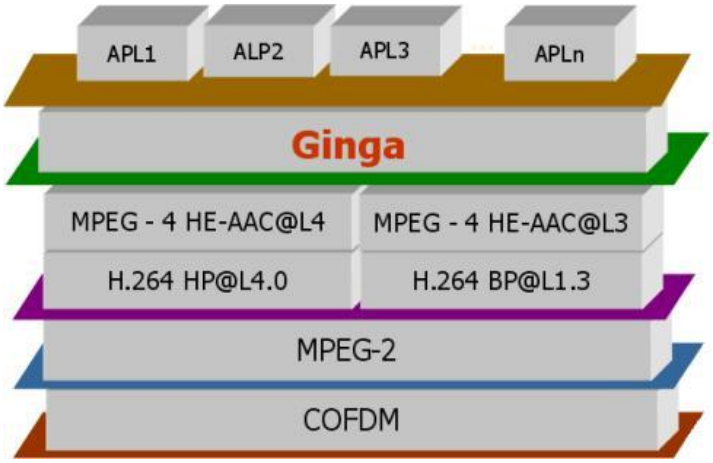

Figura 5 - Padrã de referência do SBTV. Fonte: (Telemidia-PUC-Rio) Adaptada

O *middleware* Ginga pode ser dividido em três subsistemas principais: Ginga-CC, Ginga-J e Ginga-NCL.

Ginga-CC (Ginga Common-Core) oferece o suporte básico para os ambientes declarativo (Ginga-NCL) e procedural (Ginga-J). Tem como funções principais a exibição dos vários objetos de mídia, o controle do plano gráfico, o tratamento de dados obtidos do carrossel de objetos DSM-CC, o tratamento do canal de retorno, entre outras.

Ginga-J foi desenvolvido pela UFPB para prover uma infra-estrutura de execução de aplicações baseadas em linguagem Java, com facilidades especificamente voltadas para o ambiente de TV digital.

Ginga-NCL “foi desenvolvido pela PUC-Rio para prover uma infra-estrutura de apresentação de aplicações baseadas em documentos hipermídia escritos em linguagem NCL” (Telemidia-PUC-Rio), “com facilidades para a especificação de aspectos de interatividade, sincronismo espaço-temporal de objetos de mídia, adaptabilidade e suporte a múltiplos dispositivos” (Telemidia-PUC-Rio). NCL possui Lua como sua linguagem de script.

As aplicações para o *middleware* Ginga não precisam ser exclusivamente procedurais ou declarativas. Existem elementos em ambos os sistemas que permitem a construção de aplicações híbridas: uma aplicação declarativa (NCL) pode possuir outra aplicação procedural embutida; ou, por outro lado, uma aplicação procedural pode referenciar o conteúdo declarativo, como, por exemplo, criando e iniciando apresentações.

5.1. História

Com o início das pesquisas sobre TV digital no Brasil, ficou claro que um dos pontos cruciais seria o desenvolvimento de um *middleware* nacional. Ao utilizar um dos *middlwares* já citados, o país enfrentaria um grande aumento de custos devido à propriedade industrial. O *middleware* é protegido pela legislação de patentes. Sendo assim, como o sistema é colocado em todos os *set-top boxes*, haveria um grande aumento de custos na produção dos aparelhos. Além deste custo na fabricação, haveria pagamentos de *royalties* pelos desenvolvedores e por cada programa desenvolvido. Sem este custo extra, o país possibilita um alto desenvolvimento da indústria de software.

O trabalho resultou no desenvolvimento do *middleware* declarativo Maestro. O *middleware* Maestro tem como núcleo o Formatador NCL (*Nested Context Language*). Seu objetivo principal é elaborar a proposta de um padrão de referência para sincronismo de mídias para o desenvolvimento de serviços e aplicações interativas para o Sistema Brasileiro de Televisão Digital. O projeto é coordenado pelo Laboratório TeleMídia da PUC-Rio.

Outro resultado importante foi o desenvolvimento do *middleware* FlexTV. O FlexTV foi desenvolvido seguindo o paradigma de desenvolvimento baseado em componentes para que novas funcionalidades fossem adicionadas com facilidades no decorrer do projeto. Outra preocupação foi em seguir a especificação do GEM de modo a tornar o *middleware*

compatível com os padrões internacionais, garantindo a interoperabilidade de aplicações entre estes. O projeto é coordenado pelo Laboratório de Aplicações de Vídeo Digital (LAVID) da UFPB.

A junção dos resultados de pesquisas, que originaram o *middleware* procedural FlexTV (passando a se chamar Ginga-J), com os resultados obtidos com o *middleware* declarativo Maestro (passando a se chamar Ginga-NCL), originaram o *middleware* Ginga.

5.2. Ginga-CC

Ginga-CC é módulo do *middleware* mais baixo nível e que está em contato direto com o hardware do sistema, oferecendo a abstração necessária aos outros dois módulos. A Figura 6 ilustra a arquitetura do Ginga Common Core e seus componentes são explicados, um a um, segundo (Junger, 2008).

Figura 6 - Arquitetura do Ginga-CC. Fonte: (Filho, Leite, & Batista, 2007)

- Sintonizador: É o módulo responsável por "sintonizar" o canal, escolhendo um canal físico e um dos fluxos de transporte que estão sendo enviados a este canal;
- Filtro de Seções: Uma vez sintonizado, o *middleware* deve ser capaz de acessar partes específicas do sistema de transporte. Para isso existe o Filtro de Seção, ele é capaz de buscar no fluxo a parte exata que as APIs necessitam para suas execuções. Funciona exatamente como um filtro, deixando passar apenas as informações requeridas pelas APIs;

- Processador de dados: É o elemento responsável por acessar, processar e repassar os dados recebidos pela camada física. Ele também fica responsável de notificar os outros componentes sobre qualquer evento que tenha sido recebido;
- Persistência: o Ginga é capaz de salvar arquivos, mesmo depois que o processo que o criou tenha sido finalizado, para que este possa ser aberto em outra oportunidade. Este é o módulo que dá suporte para que isto ocorra;
- Iniciador de aplicações: Esse módulo é o gerenciador de aplicações, ou seja, ele fica encarregado de carregar, configurar, inicializar e executar qualquer uma das aplicações dos ambientes declarativo e procedural. Ele também é responsável por controlar o ciclo de vida dessas aplicações, retirando-as quando necessário, além de controlar os recursos utilizados por suas APIs;
- Adaptador do A/V principal: com o adaptador de A/V principal, as aplicações conseguem enxergar os fluxos de áudio e vídeo principais. Isso se faz necessário quando a aplicação precisa controlar suas ações de acordo com o que esta sendo transmitido;
- Gerenciador Gráfico: como foi visto anteriormente os padrões de *middleware* definem como as imagens, vídeos, dados etc., são apresentados para o usuário, gerenciando as apresentações;
- Gerenciador de atualizações: componente que gerencia as atualizações feitas no sistema, verificando, baixando e atualizando o *middleware* sempre que necessário para correções de possíveis erros encontrados em versões anteriores ou adições de novas funcionalidades. Isto deve ser feito de forma transparente, sem incomodar o uso normal da TV pelo usuário;
- Exibidores de mídia: são as ferramentas necessárias para exibir os arquivos de mídia recebidos, como, por exemplo, os tipos MPEG, JPEG, TXT, MP3, GIF, HTML etc.;
- Interface com o usuário: este módulo é responsável por capturar e interpretar os eventos gerados pelo usuário, tais como comandos do controle remoto, e avisar aos outros módulos interessados;
- Gerenciamento de contexto: é o responsável por capturar as preferências do usuário, alertando os outros componentes interessados nestas preferências. Estas

informações podem ser os horários que o usuário costumar assistir à TV, bloqueio e desbloqueio de canais, entre outras;

- Canal de retorno: ele provê a interface das camadas superiores com o canal de interação (ou canal de retorno). Além disso, ele deve gerenciar o canal de retorno de forma que os dados sejam transmitidos assim que o canal esteja disponível, ou forçar a transmissão caso o usuário ou a aplicação tenha definido o horário exato para que isto ocorra;
- Acesso condicional: esse componente está encarregado de restringir conteúdos inapropriados recebidos pelos canais de programação, oferecendo assim segurança ao *middleware*.

5.3. Ginga-J

O Ginga-J foi desenvolvido com o intuito de criar o ambiente procadural do *middleware* brasileiro. A linguagem de programação utilizada é Java. Uma preocupação durante a construção do sistema foi a compatibilidade com o padrão internacional GEM. Desta forma é possível a interoperabilidade entre aplicativos do Ginga com outras plataformas presentes no mundo, tais como MHP, ACAP, OCAP e ARIB.

Para se adequar às necessidades impostas pelo padrão brasileiro foi necessário criar meios de acesso à dispositivos portáteis no Ginga-J. O telespectador pode interagir com o dispositivo Ginga através de dispositivos de interação que podem conter componentes de software Ginga, de forma que o dispositivo de interação possa enviar informações para o dispositivo Ginga utilizando as funcionalidades providas na especificação Ginga, ou seja, estes componentes de software que podem ser instalados nos dispositivos de interação permitem que as funcionalidades dos mesmos sejam exploradas, utilizando funcionalidades da API Ginga-J, por aplicações nos dispositivos Ginga (receptores de televisão digital). Para que um dispositivo de interação possa ser utilizado ele deve estar registrado com o dispositivo Ginga e durante esse processo o dispositivo de interação pode receber o componente de software necessário para viabilizar a comunicação com o dispositivo Ginga. Desta forma o dispositivo Ginga pode acessar mecanismo de entrada do dispositivo de interação, como o teclado, câmera e entrada de áudio, e mecanismo de saída, como tela,

saída de áudio, oferecendo uma experiência individualizada aos usuários. Este cenário pode ser melhor visualizado através da Figura 7.

Figura 7 - Contexto do Ginga-J. Fonte: (Filho, Leite, & Batista, 2007) Adaptada

Um *middleware* baseado na linguagem Java deve oferecer suporte à algumas APIs. Para isso o Ginga foi dividido em três conjuntos de APIs como mostrado na Figura 8. A API Verde é composta pelas API JavaTV da Sun, pelo pacote de classes de transmissão de dados *Digital Audio Video Interactive Consortium* (DAVIC) e pelo pacote de classes para interconexão de dispositivos *Home Audio/Video Interoperability* (HAVi), todos estes presentes na especificação do GEM. Aplicações desenvolvidas utilizando somente esta API podem rodar em qualquer middleware com compatível com o GEM. A API Amarela é composta pelo *Java Media Framework* (JMF) e pelas implementações específicas especificadas no GEM para o sistema brasileiro. Aplicações desenvolvidas utilizando esta API podem rodar nos outros *middlewares* apenas se forem empacotadas junto com adaptadores para os sistemas específicos onde rodarão. A API azul é a API de inovação do Ginga-J onde estão contidas as classes de acesso à dispositivos de interação. Aplicações desenvolvidas que utilizam esta API somente são compatíveis com o Ginga, não podendo ser executada em nenhum outro *middleware* existente.

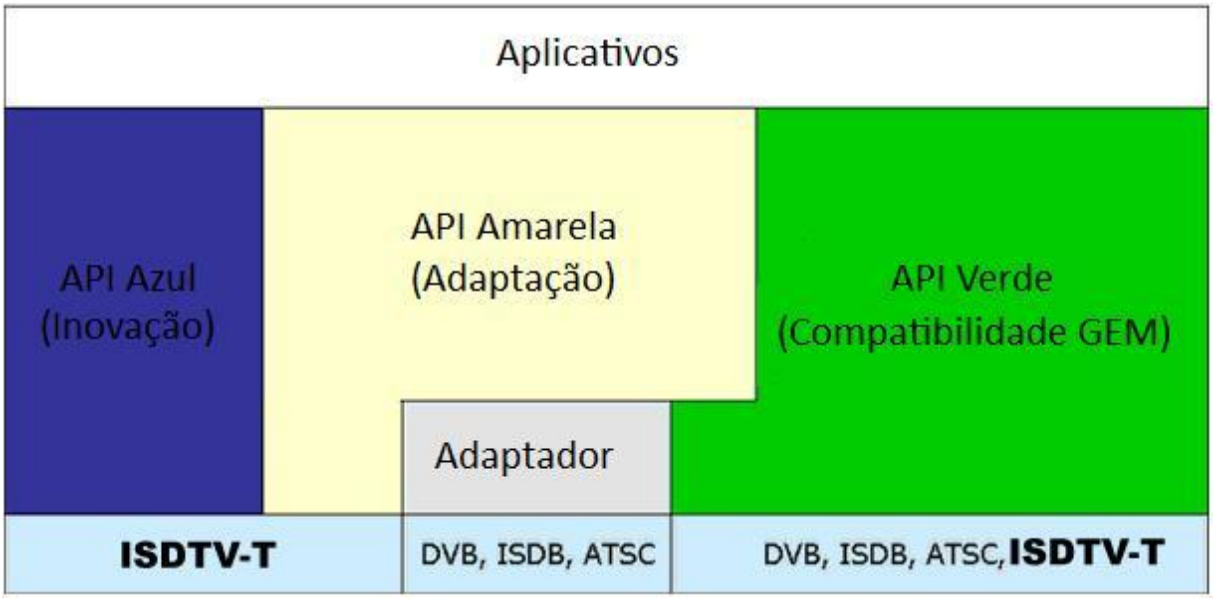

Figura 8 - APIs do Ginga-J. Fonte: (Filho, Leite, & Batista, 2007) Adaptada

Nas subseções seguintes são tratadas com mais detalhes as APIs JavaTV e a API Azul de inovação do Ginga. Também será discutido o problema com pagamentos de *royalties* na utilização do GEM.

5.3.1. JavaTV

Os pacotes da implementação JavaTV são usualmente encontrados nos *middlewares* dos padrões de TV digital e é o ponto de entrada de uma aplicação de TV digital. As classes definidas no pacote `javax.tv.xlet` merecem destaque por implementarem o ciclo de vida das aplicações. A classe principal das aplicações criadas para a TV implementa a interface `javax.tv.xlet.Xlet`, visualizada na Figura 9, e, por esse motivo, essas aplicações são genericamente denominadas *Xlets*. O conceito de *xlet* é parecido com o conceito dos *applets*. Através desta interface o gerenciador de aplicações pode iniciar, parar e controlar um *xlet*. As principais funções do gerenciador de aplicações são: sinalizar as mudanças de estados dos *Xlets* e controlar os recursos do *middleware* a partir da programação do *Xlet*.

```
 1.   public interface Xlet {
 2.
 3.     public void initXlet(XletContext ctx)
 4.       throws XletStateChangeException;
 5.
 6.     public void startXlet()
 7.       throws XletStateChangeException;
 8.
 9.     public void pauseXlet();
10.     public void destroyXlet(boolean unconditional)
11.       throws XletStateChangeException;
12.   }
```

Figura 9 - Interface *Xlet*

A Figura 10 apresenta os quatro estados de um *Xlet*: carregado (*loaded*), pausado (*paused*), iniciado (*started*) e destruído (*destroyed*). O gerenciador de aplicações é capaz de controlar múltiplas aplicações simultâneas, no entanto, somente uma aplicação é visível em um dado instante de tempo. No contexto dos estados, somente um *Xlet*, em um dado instante de tempo, pode se encontrar no estado iniciado.

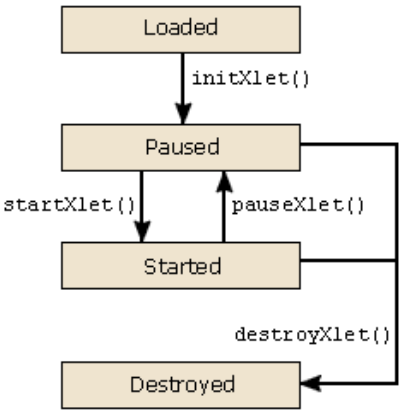

Figura 10 - Ciclo de Vida de um Xlet.Fonte: (Morris)

Quando o gerenciador de aplicações recebe um novo *Xlet*, o construtor desse *Xlet* é executado. Durante a execução do construtor, o *Xlet* encontra-se no estado carregado. Posteriormente, quando uma das condições de iniciação do *Xlet* ocorre, o método `initXlet()` da aplicação é executado, tendo como parâmetro uma instância de `javax.tv.xlet.XletContext`, que é o contexto do *Xlet*. Este contexto é um meio da aplicação obter mais detalhes sobre o ambiente no qual é executada e notificar mudanças de estados para o mesmo. Nesse momento, o *Xlet* passa para o estado pausado. A partir desse ponto, quando o método `startXlet()` é executado, o *Xlet* passa para o estado iniciado. A qualquer momento, durante o estado iniciado de um *Xlet*, o gerenciador de

aplicações pode solicitar a execução do método `pauseXlet()` e retornar a aplicação para o estado pausado. Quando um *Xlet* termina sua execução, o seu método `destroyXlet()` é executado e os recursos alocados são liberados. As condições de término de um *Xlet* são idênticas às condições de iniciação.

5.3.2. APIs de inovação do Ginga-J

As funcionalidades inovadoras do Ginga-J, fornecidas pela sua API Azul, permite o desenvolvimento de aplicações avançadas, explorando a integração com outros dispositivos como telefones celulares, PDAs, computadores portáteis e virtualmente qualquer outro dispositivo móvel com capacidade de processamento e comunicação (Filho, Leite, & Batista, 2007).

Um telefone celular pode ser utilizado como um canal de retorno para o ambiente de TV, usado como um controle remoto, como um dispositivo de interação (para responder a enquetes de maneira individual, por exemplo) etc.

A API de Integração de Dispositivos agrega ao Ginga funcionalidades relacionadas à maneira como é feita a interação dos usuários com o receptor de TV Digital. A API integra as especificações do Ginga-J e foi completamente idealizada e especificada pelo grupo de trabalho responsável pelos estudos do *middleware* para o Sistema Brasileiro de TV Digital (Silva, Batista, Leite, & Filho, 2007).

Para que os dispositivos possam ser efetivamente utilizados pelo Ginga eles precisam estar registrados junto ao *middleware*. O registro deve ser feito pela própria interface do sistema, que deve permitir a conexão de dispositivos através de diferentes redes: *Bluetooth*, *wi-fi*, infra-vermelho, USB entre outras, ficando a critério do desenvolvedor do receptor decidir que meios de conexão estarão disponíveis. O registro junto ao *middleware* só é possível para dispositivos que já possuem o módulo Ginga instalado, porém o processo de instalação também pode ser realizado pela interface do sistema.

Uma vez que um dispositivo esteja registrado junto ao Ginga (conectado) sua interação com o mesmo pode ser feita de forma automática: os recursos presentes no dispositivo, como teclado, tela, microfone, câmera, alto-falantes e outros, estarão disponíveis para as

aplicações através da API de Integração de dispositivos do Ginga. Como exemplo de uso simples, podemos conectar um celular ao Ginga e passar a usar o seu teclado como controle remoto e dispor de suas funcionalidades básicas – troca de canal, controle do volume do som etc.; em outro caso, uma aplicação interativa em execução poderá dispor de uma segunda tela de exibição (a tela do celular), que pode ser utilizada para apresentar ao usuário informações adicionais às exibidas na tela da televisão.

Os recursos dos dispositivos disponíveis para as aplicações dependem dos dispositivos conectados com o receptor – caso um celular possua câmera, será possível, por exemplo, que uma aplicação requisite a captura de uma imagem de tal dispositivo. A API dispõe de métodos para consulta de quais recursos estão disponíveis em cada dispositivo conectado.

A API de Integração de Dispositivos oferece a funcionalidade de identificar a origem de cada interação, relacionando-a a cada dispositivo individualmente. Essa característica viabiliza aplicações multiusuário.

A API especifica apenas um pacote, o pacote `br.org.sbtvd.interactiondevices`, cujas classes componentes serão detalhadas a seguir (Silva, Batista, Leite, & Filho, 2007):

- Classe `GRemoteDeviceManager`: a classe `GRemoteDeviceManager` define um objeto que deve administrar todas as conexões com os dispositivos de interatividade registrados com o Ginga;
- Classe `GRemoteDevice`: a classe `GRemoteDevice` define um objeto que deve ser uma representação abstrata de um dispositivo de interação. Essa classe oferece métodos que possibilitam a recuperação de informações acerca dos dispositivos registrados (tipo do dispositivo, funcionalidades disponíveis etc.), bem como explorar as funcionalidades disponíveis do mesmo (utilizar recursos de gravação de áudio, vídeo, captura de imagens etc.);
- Interface `GRemoteDeviceActionListener`: a interface `GRemoteDeviceActionListener` deve conter métodos que devem ser implementados por qualquer objeto que deva ser notificado sobre eventos relacionados às atividades dos dispositivos de interação registrados com o Ginga;

- Classe `GRemoteEvent`: a classe `GRemoteEvent` descreve um objeto que representa um evento relacionado a um dispositivo de interação registrado com o Ginga. Este objeto encapsula dados relacionados ao evento;
- Classe `GRemoteKeyEvent`: a classe `GRemoteKeyEvent` estende `java.awt.event.KeyEvent`, e é relacionada a eventos de teclas originados em dispositivos de interatividade. Sua utilização é feita da mesma maneira como nos eventos `awt` normais, bastando fazer um *downcasting* quando necessário.
- Classe `GRemoteUserEvent`: A classe `GRemoteUserEvent` estende `org.dvb.event.UserEvent`, e é relacionada a eventos do usuário originados em dispositivos de interatividade.

5.3.3. Royalitites

Na especificação inicial do Ginga-J estava previsto o uso de todas as classes definidas no padrão GEM. Porém, descobriu-se que a utilização destas classes, dos pacotes DAVIC, HAVi e DVB, poderiam acarretar a obrigatoriedade de pagamento de *royalties* pelos desenvolvedores de geradores de conteúdos. Isto inviabilizaria o esforço feito na construção do *middleware*, visto que o não pagamento destes *royalties* foi uma das questões levantadas para o início das pesquisas.

Devido a este fato, o fórum do Sistema Brasileiro de TV Digital se uniu à Sun Microsystems com o intuito em desenvolver uma solução em código aberto com as funcionalidades equivalentes (Sun Microsystems, Inc., 2008). Em novembro de 2008 a Sun cumpriu o prometido de desenvolver a solução alternativa ao GEM em código aberto e entregou ao fórum SBTVD. (Converge Comunicações, 2008). Resta agora aos desenvolvedores do Ginga-J realizarem as adaptações necessárias para a disponibilização do *middleware*.

5.4. Ginga-NCL

O Ginga-NCL é a parte declarativa do *middleware* brasileiro, ou seja, é a responsável por processar e apresentar as aplicações que são baseadas em uma linguagem desenvolvida pela PUC-Rio e batizada de NCL (*Neted Context Language*). Essa linguagem não mistura a definição do conteúdo de um documento com sua estruturação, não definindo assim nenhum objeto de mídia. O que o NCL faz é definir um descritor, que funciona como uma espécie de cola e mantém todos estes objetos juntos durante uma apresentação. Com isso, faz-se necessário adicionar ao Ginga ferramentas que tratem estes objetos de mídia.

Outros objetos que devem ser considerados são os baseados em XHTML, que são tratados como um caso particular de objetos de mídia. Dependendo do *browser* escolhido como exibidor de mídia XHTML, pode-se haver compatibilidade com outros *middlewares* existentes no mercado. É importante observar que o *browser* que dá suporte aos objetos baseados em XHTML pode também dar suporte à linguagem imperativa *ECMAScript*, sendo, no entanto, desencorajado o uso de XHTML para a definição de relacionamentos temporais, já que com NCL é possível definir a sincronização das mídias com maior facilidade. Além do *ECMAScript*, outros objetos imperativos podem ser tratados pelo NCL como objetos de mídia. É o caso do *Xlet*, que inclusive faz parte da ponte entre o ambiente declarativo e procedural.

Outro objeto que faz parte da ponte entre os dois ambientes, tendo assim suporte em Ginga, são os objetos Lua. A linguagem Lua tem diversas características que fazem dela ideal para a configuração, automação e prototipagem rápida. Sendo assim, sua máquina virtual foi acoplada ao formatador NCL, permitindo às aplicações NCL trocar informações com programas Lua.

Por fim temos o mais importante dos módulos do Ginga-NCL. Já citado anteriormente, o Formatador NCL é responsável por controlar as apresentações NCL, mantendo a relação de sincronismo entre os objetos de mídia. Enquanto um objeto está sendo exibido, um evento pode ser gerado. Por exemplo, durante a exibição de um vídeo, pode haver a seleção de um botão que dispara uma imagem e um texto. Como esses objetos são tratados por ferramentas específicas de cada um, é papel do Formatador NCL reconhecer esse novo

evento e avisar às outras ferramentas que devem exibir seus arquivos. Para que isto ocorra, uma API padrão é definida, fazendo que todas as ferramentas de exibição padrão obedeçam ao Formatador NCL. De forma similar, sempre que um *browser* é acoplado ao equipamento, este deve obedecer às especificações do Formatador ficando assim compatível com o Ginga.

Tudo isto permite que a linguagem NCL seja extremamente flexível e adaptável, atendendo assim, a nova demanda que surgiu nos últimos anos em que os padrões buscam cada vez mais compatibilidades entre eles.

5.4.1. A linguagem NCL

A linguagem declarativa NCL tem por base o modelo NCM (*Nested Context Model*) (Soares & Rodrigues, 2005) que é um modelo conceitual centrado na representação e tratamento de documentos hipermídia também desenvolvido na PUC-Rio. Este modelo define dois tipos de nós, que no NCL são os nós de mídia e os nós de contexto, que são compostos pelos nós de mídias e suas relações.

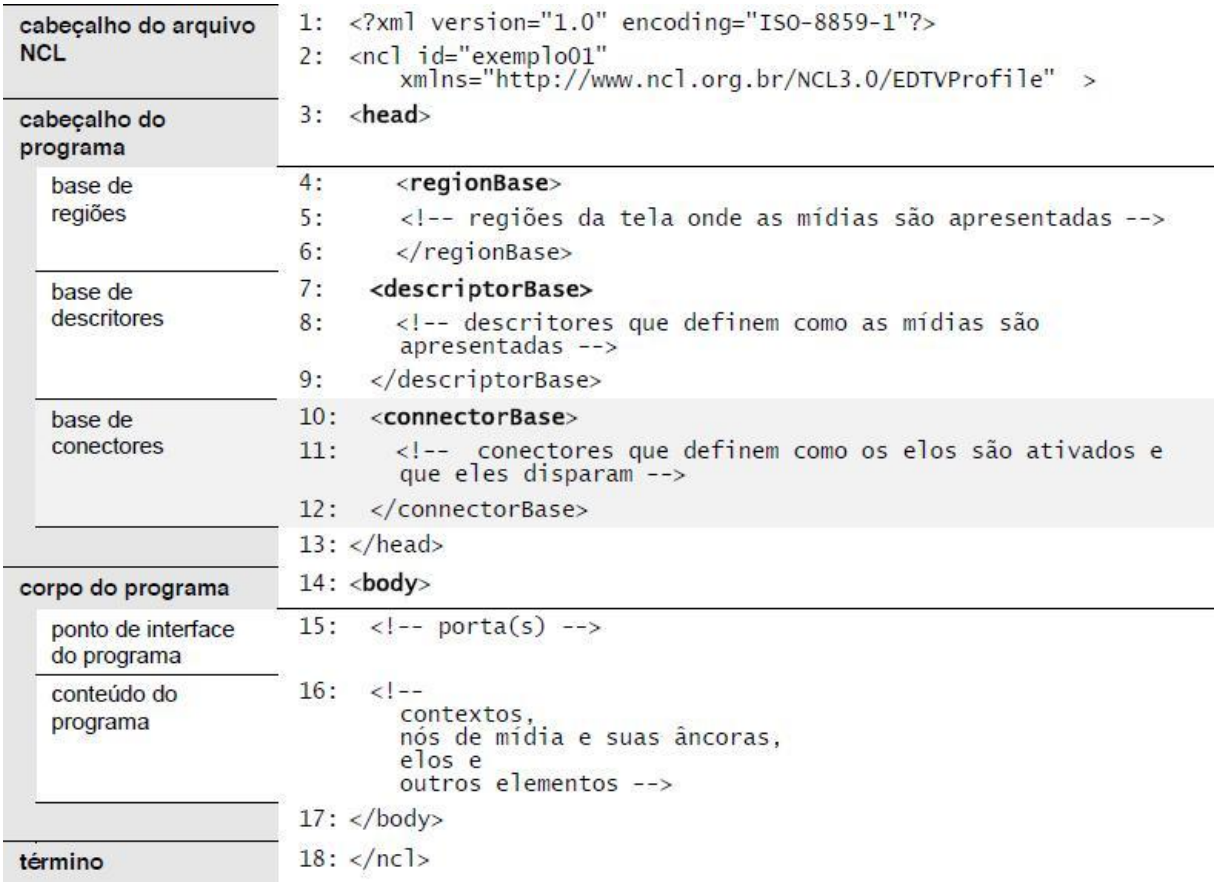

Figura 11 - Estrutura básica de um documento NCL. Fonte: (Neto, Soares, Rodrigues, & Barbosa, 2007) Adaptada.

Na Figura 11 podemos ver a estrutura básica de um documento NCL. As linhas 1 e 2 são o cabeçalho do documento NCL, como todo documento baseado em XML deve possuir. A tag raiz `<ncl>` contém o identificador do documento e seu *namespace*.

Da linha 3 até a linha 13 temos o cabeçalho do programa. Em `<regionBase>` definimos as regiões, que são áreas onde cada mídia será apresentada na tela. Uma região indica a posição e as dimensões da área onde uma mídia será apresentada na tela através da tag `<region>`. Em outras palavras, uma região serve para inicializar a posição dos nós de mídia em um local específico. Dentro de `<descriptorBase>` são definidos os descritores, que definem como um nó de mídia será apresentado, incluindo uma associação com a região onde será exibido.

Os elos (`<link>`) associam os nós através dos conectores, que definem a semântica da associação entre os nós. Na NCL o sincronismo não é feito através de marcas de tempos e sim através dos conectores. Inseridos no documento através de `<conector>`, eles definem

os papéis (`role`) que os nós de origem e destino exercem nos elos que utilizam o conector. Os conectores são inseridos no documento na região `<connectorBase>`.

A partir da linha 14 temos o corpo do programa. O elemento `<body>` é o contexto que contém todos os nós do documento, sejam eles nós de mídia ou nós de contexto. Contextos são utilizados para estruturar o documento. Eles podem ser aninhados para refletir a estrutura do documento e ajudar o autor a organizar os segmentos do programa audiovisual interativo.

Um documento NCL deve declarar obrigatoriamente pelo menos uma porta, a porta de entrada do programa indicando qual é o nó de mídia ou contexto inicial. Uma porta é um ponto de acesso externo ao conteúdo de um contexto. Para o elo apontar a um nó interno de um contexto, este contexto deve possuir uma porta que leve a este nó.

Um nó de mídia `<media>` define o objeto de mídia propriamente dito: vídeo, áudio, texto etc. Ele deve apresentar o arquivo de origem da mídia através do atributo `src` e o descritor que regulará sua apresentação. As âncoras são utilizadas como ponto de entrada para os nós de mídia. Uma âncora pode ser do tipo âncora de conteúdo (`<area>`), que define um segmento da mídia (intervalo de tempo ou região no espaço), ou do tipo âncora de propriedade, que se refere a uma propriedade de um nó de origem ou destino (volume do som em um nó de áudio ou vídeo, coordenadas e dimensões de um nó de mídia visual etc.).

Por fim, na linha 18 temos a conclusão do documento com o fechamento da tag raiz.

Maiores detalhes sobre a linguagem NCL podem ser encontrados em sua especificação (ABNT - Associação Brasileira de Normas Técnicas, 2008).

5.4.2. NCLua

Lua é uma linguagem de programação poderosa, rápida e leve, projetada para estender aplicações. Lua combina sintaxe simples para programação procedural com poderosas construções para descrição de dados baseadas em tabelas associativas e semântica extensível. Lua é tipada dinamicamente, é interpretada a partir de *bytecodes* para uma

máquina virtual baseada em registradores, e tem gerenciamento automático de memória com coleta de lixo incremental. Essas características fazem de Lua uma linguagem ideal para configuração, automação (*scripting*) e prototipagem rápida.

Para a integração de Lua com o NCL, foi criada uma nova classe de objetos de mídia. Estes objetos de mídia recebem como fonte um arquivo com extensão “lua”, ou seja, um script Lua. Estes objetos de mídia são chamados neste contexto de NCLua.

Para manter a independência da linguagem NCL, os scripts NCLua são tratados como nós de mídias assim como todos os outros. Desta forma é utilizado a criação de elos para o controle dos elementos NCLua a serem executados, usando todos os eventos disponíveis para tal fim, e também sua relação temporal com outras mídias. Isto determina o início e o fim de um script no programa apresentado. As âncoras de conteúdo (áreas) e de propriedade são passadas como parâmetros para a aplicação NCLua que será executada, sendo que no caso das propriedades, é passado também o seu valor como em qualquer outra relação entre mídias. Fica a critério do autor do script decidir como serão utilizado estes parâmetros no interior da aplicação. Como exemplo, âncoras do tipo propriedades podem ser utilizadas como varáveis de mesmo nome na aplicação. Desta forma as aplicações NCLua se tornam nada mais do que um tratador de eventos, seja esses eventos oriundos do documento NCL ou de outra fonte externa como o controle remoto do usuário.

Os scripts NCLua são essencialmente scripts Lua com algumas pequenas adaptações. Algumas bibliotecas, como as que interagem com o sistema em que está sendo executado o script, foram removidas por não terem utilidade no contexto em que está empregado. Outra mudança são os novos módulos disponíveis:

- Módulo `event`: permite que a aplicação NCLua se comunique através de eventos;
- Módulo `canvas`: oferece uma API para desenhar primitivas gráficas e imagens;
- Módulo `settigns`: exporta uma tabela com variáveis definidas pelo autor do documento NCL e variáveis reservadas;
- Módulo `persistent`: exporta uma tabela com variáveis persistentes, que estão disponíveis para manipulação apenas por objetos imperativos.

Um fato comum entre estes novos módulos é o de serem extremamente simples oferecendo, de maneira geral, apenas o acesso a primitivas disponíveis em qualquer sistema computacional. Apesar de trazer algumas vantagens como um padrão que permite acompanhar uma evolução no mercado e possuir uma definição pequena e de rápido entendimento, isto torna o desenvolvimento de aplicações complexas extremamente trabalhoso.

Dentre estes módulos apresentados, o mais importante é o módulo `event`, uma vez que este está intimamente ligado com a ponte entre o documento NCL e o script NCLua. Para que a aplicação receba eventos do formatador é necessário que ele registre uma função que trate o evento recebido através de uma chamada à `event.register` como mostrado na linha 10 da Figura 12. A função registrada (linha 2) deve possuir um único parâmetro: o evento recebido. Para a comunicação no sentido contrário, o NCLua deve fazer uma chamada à função `event.post` (linha 7) passando como parâmetro o evento enviado ao documento NCL. Esses eventos são tabelas Lua simples cujo campo `class` identifica a classe do evento. As classes podem ser `ncl`, que é a classe de eventos relacionado à ponte entre o NCL e o NCLua, `key`, que é a classe relacionada à eventos do controle remoto, `tcp`, classe relacionada com conexão à internet (canal de retorno), `sms`, relacionada à mensagens de texto e `user`, que é um tipo de classe especial que contém inicialmente apenas o atributo `class` e é utilizada para eventos customizados.

A classe ncl possui os seguintes atributos:

- `type`: recebe o valor `presentation` para âncoras de conteúdo e `attribution` para âncoras de propriedades;
- `action`: assume um dos valores das transições: `start`, `stop`, `pause`, `resume`, `abort`;
- `area` ou `property`: o nome da área, caso `type` seja `presentation`, ou o nome da propriedade, caso o `type` seja `attribution`;
- `value`: caso o `type` seja `attribution` recebe o valor da propriedade.

Na Figura 12 podemos ver um exemplo de código. Na primeira linha é criada uma variável com valor zero. A função `handler` é criada na linha 2, esta é a função que irá tratar o evento. As linhas 3 e 4 fazem a verificação do evento. Caso ele o nome da propriedade do evento seja ‘propriedade’ e a ação do tipo `start`, a variável recebe o valor da propriedade (linha 5),

a ação do evento é atualizada para `stop` (linha 6) e o evento é enviado para o formatador NCL (linha 7). Por fim, devemos registrar a função `handler`, como feito na linha 10.

```
 1.   local valor = 0
 2.   function handler(evt)
 3.     if(evt.property == 'propriedade') and
 4.       (evt.action == 'start') then
 5.        valor = evt.value        -- atribui o valor
 6.        evt.action = 'stop'      -- atualiza o evento
 7.        event.post(evt)          -- sinaliza o fim
 8.     end
 9.   end
10.   event.register(handler)
```

Figura 12 - Exemplo de código NCLua

Maiores detalhes sobre o NCLua pode ser encontrados em (ABNT - Associação Brasileira de Normas Técnicas, 2008).

5.4.3. A ferramenta Composer

Composer é uma ferramenta para edição de documentos NCL de modo visual desenvolvida pela PUC-Rio. Seu objetivo é facilitar o desenvolvimento por pessoas não familiarizadas com programação em linhas de código. Nesta ferramenta, as abstrações são definidas em diversos tipos de visões que permitem simular um tipo específico de edição (estrutural, temporal, leiaute e textual). Essas visões funcionam de maneira sincronizada, a fim de oferecer um ambiente integrado de autoria e podem ser visualizadas na Figura 13.

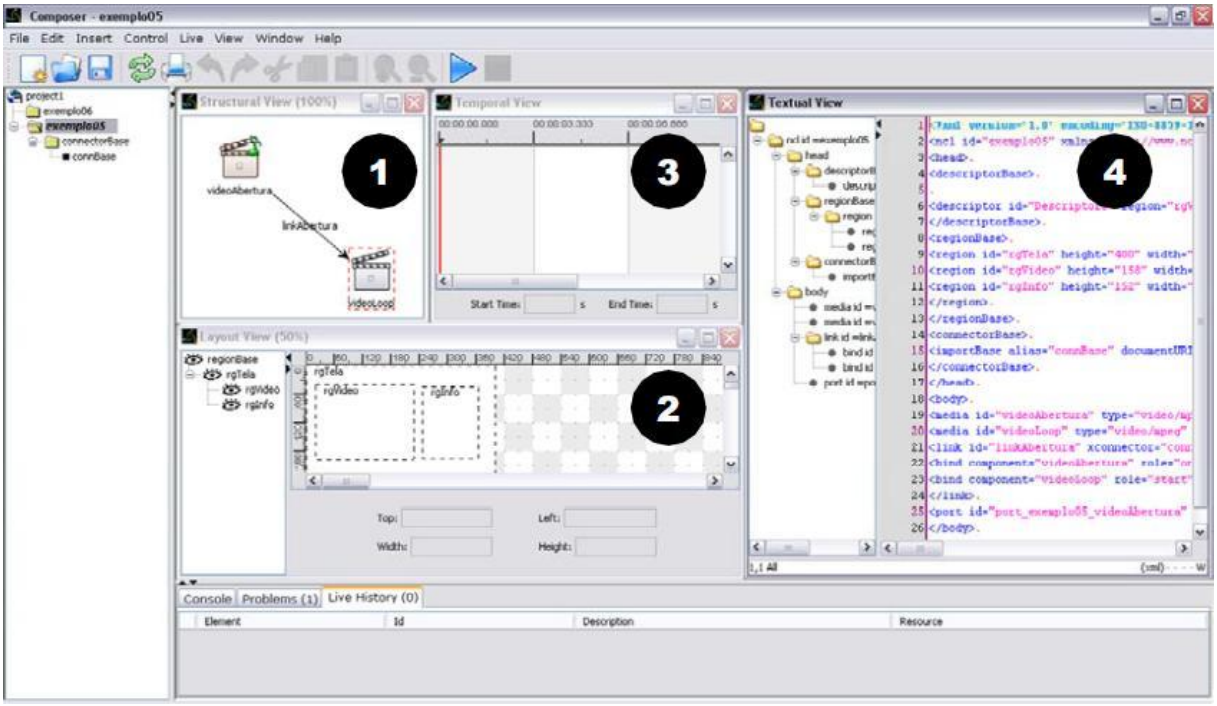

Figura 13 - Ferramenta de autoria Composer. Fonte: (Neto, Soares, Rodrigues, & Barbosa, 2007)

A visão estrutural apresenta os nós e os elos entre os nós e está indicada na Figura 13 com o número 1. Nesta visão é possível criar os nós de mídia, contextos e elos e também definir suas propriedades. A visão de leiaute, representada pelo número 2, permite a criação e edição das regiões da tela onde o documento será apresentado. A visão temporal ilustra o sincronismo temporal entre os nós de mídia e está representada pelo número 3. A última visão, representada pelo número 4, é a visão textual. Esta visão mostra o código NCL criado e pode ser editada como num editor de texto comum, porém com a facilidade da coloração da sintaxe para uma melhor visualização.

A mudança em qualquer uma das visões é refletida nas outras. A ferramenta também conta um *player* integrado que permite a execução dos documentos de maneira emulada.

5.4.4. Exemplos

Para um melhor entendimento sobre aplicações NCL é recomendado um tutorial onde são apresentados diversos exemplos com níveis de complexidade diferentes disponível em (Neto, Soares, Rodrigues, & Barbosa, 2007). Apesar de não disponível a versão final do

Ginga-NCL, os exemplos e aplicativos desenvolvidos podem ser executados de forma satisfatória através da máquina virtual disponível em (Comunidade Ginga: Ginga-NCL).

Exemplos de aplicativos para Ginga-J ainda não são encontrados pois o ambiente encontra-se em desenvolvimento.

6. Construindo aplicações para a TV Digital

Como o Ginga-J ainda não se encontra disponível, aplicações para testes podem ser desenvolvidas utilizando o *XleTView* (XleTView). Note que estas aplicações devem ser construídas baseadas no padrão GEM, devendo sofrer as devidas alterações para a posterior execução no Ginga. A dificuldade encontrada é o conhecimento das diversas APIs disponíveis no *framework* GEM. Isto torna um pouco desencorajador o trabalho, pois não se sabe se estas APIs serão suportadas pelo Ginga-J devido aos problemas de *royalties*. Provavelmente todas as aplicações devem sofrer adaptações, já que foi noticiado que a Sun desenvolveu uma nova solução livre à estas APIs para o Ginga. Este trabalho não visa apresentar uma aplicação desenvolvida para o padrão GEM.

Para o desenvolvimento de aplicações em Ginga-NCL foram testadas as ferramentas Composer, bem como o plugin disponível para a IDE Eclipse, NCL Eclipse. Para a execução das aplicações foram utilizados o Ginga-NCL Emulator for Windows e a máquina virtual Ginga-NCL Virtual Set-top Box, que executa em uma maquina virtual que emula o funcionamento do Set-Box. O software VMWare Player eh utilizado para emular o set-top box.

6.1. Descrição da aplicação Ginga-NCL

Para a avaliação das linguagens disponiveis para o desenvolvimento de aplicacoes interativas no middleware Ginga, foi desenvolvida uma aplicação em Ginga-NCL. Assim foi necessário a escolha de um vídeo que pudesse ser iniciado junto com a aplicação. Decidiu-se também que a aplicação seria totalmente relacionada ao vídeo exibido, de forma a avaliar as vantagens encontradas na utilização da linguagem declarativa NCL parar estes tipos de aplicações.

O primeiro passo foi escolher o vídeo a ser apresentado. Este vídeo deveria ser, de preferência, um programa televisivo com características comumente encontradas em programas veiculados nas emissoras brasileiras. Porém não é possível apenas copiar um programa transmitido na TV, pois estes possuem direitos autorais que devem ser respeitados. A solução encontrada foi utilizar um programa produzido pela Produtora de

Multimeios da Universidade Federal de Juiz de Fora. O programa Mosaico foi o escolhido para os experimentos.

Segundo o site da produtora o “Mosaico é um programa de cunho cultural, que apresenta a cada edição as particularidades e as curiosidades dos bairros de Juiz de Fora. Através de bate-papos e da apresentação dos lugares visitados, é possível montar, de forma descontraída, um mosaico da identidade da cidade”. Este programa é veiculado em canal aberto local pela TVE, canal 12 de Juiz de Fora. Para a aplicação foi escolhido o capítulo sobre o bairro Borboleta. Este capítulo foi escolhido por apresentar um relacionamento de tempo para os três blocos disponíveis e para a receita apresentada no programa.

Figura 14 - Programa Mosaico da Produtora de Multimeios da UFJF

O segundo passo foi definir qual o tipo de interação seria disponibilizada ao telespectador do programa. Decidiu-se criar um painel ao redor do vídeo do programa que possibilite a interação com o usuário através de seis botões:

- Bloco 1: exibe na barra inferior do painel informações sobre os entrevistados do bloco 1 e conteúdo relacionado;
- Bloco 2: exibe na barra inferior do painel informações sobre os entrevistados do bloco 2 e conteúdo relacionado;
- Bloco 3: exibe na barra inferior do painel informações sobre os entrevistados do bloco 3 e conteúdo relacionado;
- Receita: exibe na barra inferior do painel os ingredientes da receita exibida pelo programa;

- Mapa: exibe na barra inferior do painel uma imagem com o mapa do bairro Borboleta;
- Contato: exibe na barra inferior do painel informações de contato da Produtora de Multimeios da UFJF.

Não é conveniente exibir a aplicação diretamente a todos os usuários, já que nem todos podem estar interessados em interagir com o programa, desejando apenas assisti-lo em seu televisor no formato tela cheia. Desta forma, foi criado um ícone de interatividade que se inicia junto ao programa na parte superior direita do vídeo. O usuário pode então iniciar a aplicação através da tecla “MENU” do controle remoto. A qualquer momento o usuário pode também fechar o painel e voltar a assistir o programa em tela cheia apertando novamente a tecla “MENU”.

Outra característica importante da aplicação seria demonstrar uma relação temporal sobre o vídeo apresentado. De forma a suprir esta necessidade foi disponibilizado a utilização de botões do controle remoto para a escolha do bloco a ser exibido. Além disto, é também possível ao usuário escolher o trecho que exibe a receita. Informações sobre quais botões devem ser acionados foram adicionadas a cada painel de informação pertinente. Para visualizar o bloco 1 do programa, basta ao usuário pressionar a tecla vermelha (RED em NCL) do controle remoto enquanto o painel interativo estiver em execução e este bloco será imediatamente exibido. De modo análogo se o usuário desejar exibir o bloco 2, basta pressionar a tecla verde (GREEN), se desejar exibir o bloco 3, basta pressionar a tecla amarela (YELLOW) e finalmente, se desejar visualizar o trecho onde é exibida a receita, basta pressionar a tecla azul (BLUE) do controle remoto enquanto o painel estiver em execução.

6.2. Código NCL

O código inicia-se com a criação da estrutura NCL já apresentada neste trabalho. Como podemos observar na Listagem 1, no cabeçalho do programa, seção `<head>`, as primeiras linhas de código definem a base de regras do documento. Estas regras são utilizadas em conjunto com um nó de mídia do tipo `application/x-ginga-settings` para o controle da

mídia a ser exibida no painel de informações. Todos os atributos de comparação são do tipo `eq` e comparados a números inteiros referentes à opção escolhida pelo usuário.

```
15        <ruleBase>
16            <rule comparator="eq" id="opcao1" value="1" var="opcao"/>
17            <rule comparator="eq" id="opcao2" value="2" var="opcao"/>
18            <rule comparator="eq" id="opcao3" value="3" var="opcao"/>
19            <rule comparator="eq" id="opcao4" value="4" var="opcao"/>
20            <rule comparator="eq" id="opcao5" value="5" var="opcao"/>
21            <rule comparator="eq" id="opcao6" value="6" var="opcao"/>
22        </ruleBase>
```

Listagem 1 - Base de regras

O segundo trecho de código é referente à base de regiões e pode ser visualizado na Listagem 2. Nestas linhas são declaradas todas as regiões utilizadas pela aplicação interativa. Os valores de posição e tamanho foram todos utilizados de forma percentual. Esta decisão foi tomada de forma a permitir a exibição completa do programa em telas de tamanhos variados.

```
27        <regionBase>
28            <region height="100%" id="rTV" left="0" top="0" width="100%">
29                <region height="100%" id="rgVideo" left="0" top="0" width="100%" zIndex="1"/>
30                <region height="100%" id="rgFundo" left="0" top="0" width="100%" zIndex="0"/>
31                <region height="65%" id="rgVideoPequeno" top="2%" left="19%" width="65%"/>
32                <region height="5%" id="rgIcone" right="5%" top="5%" width="4%" zIndex="20"/>
33                <region height="70%" id="rgMenuEsquerda" left="5%" width="11%">
34                    <region height="13%" id="rgBloco1" top="33%" width="100%" zIndex="1"/>
35                    <region height="13%" id="rgBloco2" top="57%" width="100%" zIndex="1"/>
36                    <region height="13%" id="rgBloco3" top="80%" width="100%" zIndex="1"/>
37                </region>
38                <region height="70%" id="rgMenuDireita" right="5%" width="7%">
39                    <region height="12.5%" id="rgReceita" top="33%" width="100%" zIndex="1"/>
40                    <region height="12.5%" id="rgMapa" top="56%" width="100%" zIndex="1"/>
41                    <region height="12.5%" id="rgContato" top="78%" width="100%" zIndex="1"/>
42                </region>
43                <region bottom="0" height="28%" id="rgInformacao"/>
44            </region>
45        </regionBase>
```

Listagem 2 - Base de regiões

Abaixo segue uma descrição sobre as regiões criadas:

- rgTV: região que contempla toda a extensão da tela da TV. Todas as outras regiões são sub-regiões desta e seus atributos de posição e tamanho são relativos à mesma;
- rgVideo: região onde é apresentado o vídeo com o programa Mosaico em tela inteira;
- rgFundo: região onde é apresentada a imagem de fundo da aplicação;

- rgVideoPequeno: região onde é apresentado o programa Mosaico em tamanho menor, encaixando-se no centro do painel da aplicação;
- rgIcone: região onde é apresentado o ícone de interatividade;
- rgMenuEsquerda: região onde é apresentado o menu esquerdo da aplicação. As três regiões seguintes são sub-regiões desta:
    - rgBloco1: sub-região onde é apresentado o botão referente ao primeiro bloco;
    - rgBloco2: sub-região onde é apresentado o botão referente ao segundo bloco;
    - rgBloco3: sub-região onde é apresentado o botão referente ao terceiro bloco;
- rgMenuDireita: região onde é apresentado o menu direito da aplicação. As três regiões seguintes são sub-regiões desta:
    - rgReceita: sub-região onde é apresentado o botão referente à receita;
    - rgMapa: sub-região onde é apresentado o botão referente ao mapa;
    - rgContato: sub-região onde é apresentado o botão referente à informações de contato da produtora;
- rgInformacao: região onde é apresentada as informações de acordo com a opção escolhida pelo usuário.

A seguir foi definida a base dos descritores. O código pode ser visualizado na Listagem 3. Foram criados os descritores para as regiões do vídeo, da imagem de fundo, do ícone de interação, do menu da esquerda, do menu da direita, do painel de informações e de cada um dos botões dos menus. Note que os descritores de botões dos menus contêm informações de navegabilidade e de foco através dos atributos `focusBorderColor`, `focusBorderWidth`, `focusIndex`, `moveDown`, `moveLeft`, `moveRight` e `moveUp`.

```
50          <descriptorBase>
51              <descriptor id="dVideo" region="rgVideo"/>
52              <descriptor id="dFundo" region="rgFundo"/>
53              <descriptor id="dIcone" region="rgIcone"/>
54              <descriptor id="dMenuEsquerda" region="rgMenuEsquerda"/>
55              <descriptor id="dMenuDireita" region="rgMenuDireita"/>
56              <descriptor id="dInformacao" region="rgInformacao"/>
57              <!-- descritores do menu -->
58              <descriptor focusBorderColor="blue" focusBorderWidth="-2"
59                  focusIndex="1" id="dBloco1" moveDown="2" moveLeft="4"
60                  moveRight="4" moveUp="3" region="rgBloco1"/>
61              <descriptor focusBorderColor="blue" focusBorderWidth="-2"
62                  focusIndex="2" id="dBloco2" moveDown="3" moveLeft="5"
63                  moveRight="5" moveUp="1" region="rgBloco2"/>
64              <descriptor focusBorderColor="blue" focusBorderWidth="-2"
65                  focusIndex="3" id="dBloco3" moveDown="1" moveLeft="6"
66                  moveRight="6" moveUp="2" region="rgBloco3"/>
67              <descriptor focusBorderColor="blue" focusBorderWidth="-2"
68                  focusIndex="4" id="dReceita" moveDown="5" moveLeft="1"
69                  moveRight="1" moveUp="6" region="rgReceita"/>
70              <descriptor focusBorderColor="blue" focusBorderWidth="-2"
71                  focusIndex="5" id="dMapa" moveDown="6" moveLeft="2"
72                  moveRight="2" moveUp="4" region="rgMapa"/>
73              <descriptor focusBorderColor="blue" focusBorderWidth="-2"
74                  focusIndex="6" id="dContato" moveDown="4" moveLeft="3"
75                  moveRight="3" moveUp="5" region="rgContato"/>
76          </descriptorBase>
```

Listagem 3 - Base de descritores

Para finalizar a seção do cabeçalho do documento é descrito a base de conectores. A base de conectores poderia ser importada de um arquivo externo já existente na ferramenta Composer que contém um conjunto com os conectores mais utilizados na linguagem. Porém optou-se na criação dos conectores necessários um a um como forma de demonstração. O código pode ser visualizado na Listagem 4. O primeiro conector criado foi o conector “onBeginStartN”. Este é um conector simples que inicia um número n de nós de mídias no momento em que outro nó é iniciado. É utilizado para iniciar o ícone de interação no mesmo momento que o vídeo principal inicia. O conector “onKeySelectionSetStartStop” é um conector mais avançado que utiliza ações múltiplas e define dois parâmetros: “tecla” que recebe a tecla acionada no controle remoto e “valor” que recebe o valor da propriedade do nó a ser modificada. Note que poderia inserir diretamente a tecla e o valor no conector, porém é uma boa prática atribuir parâmetros de modo a garantir maior reutilização do conector. Este conector é utilizado para acionar o painel de interação no momento em que o ícone de interação é exibido na tela. O atributo valor faz-se necessário para definir a opção inicial do menu de informações.

```
81          <connectorBase>
82              <causalConnector id="onBeginStartN">
83                  <simpleCondition role="onBegin"/>
84                  <simpleAction max="unbounded" qualifier="par" role="start"/>
85              </causalConnector>
86
87              <causalConnector id="onKeySelectionSetStartStop">
88                  <connectorParam name="tecla"/>
89                  <connectorParam name="valor"/>
90                  <simpleCondition key="$tecla" role="onSelection"/>
91                  <compoundAction operator="par">
92                      <simpleAction role="set" value="$valor"/>
93                      <simpleAction role="start"/>
94                      <simpleAction role="stop"/>
95                  </compoundAction>
96              </causalConnector>
97
98              <causalConnector id="onKeySelectionStopSetStart">
99                  <connectorParam name="tecla"/>
100                 <connectorParam name="valor"/>
101                 <simpleCondition role="onSelection" key="$tecla"/>
102                 <compoundAction operator="seq">
103                     <simpleAction role="stop" delay="0.0s"/>
104                     <simpleAction role="set" value="$valor" delay="0.0s"/>
105                     <simpleAction role="start" delay="0.0s" />
106                 </compoundAction>
107             </causalConnector>
108
109             <causalConnector id="onBeginSet">
110                 <connectorParam name="valor"/>
111                 <simpleCondition role="onBegin"/>
112                 <simpleAction role="set" value="$valor"/>
113             </causalConnector>
114
115             <causalConnector id="onKeySelectionStopStart">
116                 <connectorParam name="tecla"/>
117                 <simpleCondition role="onSelection" key="$tecla"/>
118                 <compoundAction operator="seq">
119                     <simpleAction role="stop"/>
120                     <simpleAction role="start" delay="0.2s"/>
121                 </compoundAction>
122             </causalConnector>
123
124         </connectorBase>
```

Listagem 4 - Base de conectores

Em seguida é criado o conector “onKeySelectionStopSetStart”. Este conector é muito parecido com o anterior, porém suas ações são executadas em seqüência diferente. Este conector é utilizado no acionamento de todos os botões dos menus da aplicação. O conector “onBeginSet” é um conector simples com um parâmetro “valor” e é utilizado para redimensionar o vídeo no momento que o painel é iniciado e no momento em que é fechado e o ícone de interação é mostrado novamente na tela. Para finalizar é criado o conector

“onKeySelectionStopStart”, utilizado para viabilzar a seleção do conteúdo através das teclas coloridas. Assim o cabeçalho do documento é finalizado.

Após a finalização do cabeçalho é iniciado a criação do corpo do documento onde são definidos os contextos, mídias e estrutura do programa. A primeira tag definida é o ponto de entrada do documento como visto na Listagem 5. O vídeo principal do programa Mosaico é iniciado quando o documento é interpretado. Esta porta é um elemento obrigatório em um documento NCL, sem ela não é possível definir qual mídia será executada ao iniciar o aplicativo.

```
131         <!--+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
132         !  PONTO DE ENTRADA:
133         !  indica o componente onde o programa inicia
134         !+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++-->
135         <port component="video" id="pInicio"/>
```

Listagem 5 - Porta do contexto principal

Logo após são definidas as mídias do documento na Listagem 6. A primeira mídia definida é “settings”. Esta é uma mídia do tipo `application/x-ginga-settings` e, como já mencionado, é utilizada para armazenar a opção de informação escolhida pelo usuário. Ela define apenas uma propriedade, “opcao”, que é a propriedade encarregada de armazenar o valor inteiro referente à opção escolhida. A segunda mídia descrita é a mídia principal do documento, o vídeo do programa exibido. Note que o atributo `src` recebe como valor o caminho local do vídeo a ser exibido. Este recurso foi necessário por não ser possível emular um gerador de fluxo MPEG2. A mídia video também define algumas âncoras. São quatro âncoras de conteúdo (`<area>`) e apenas uma de propriedade. As âncoras de conteúdo definidas são referentes ao espaço de tempo do vídeo onde estão localizados os blocos do programa e a receita exibida. A propriedade definida `bounds` se refere ao tamanho e local de exibição do vídeo na tela. Esta propriedade recebe quatro valores separados por espaço, poupando-nos de declarar cada propriedade separadamente, que indicam em seqüência: a posição em relação à margem esquerda da região (referente à propriedade `left`), a posição em relação ao topo da região (referente à propriedade `top`), o valor da largura da mídia (referente à propriedade `width`) e o valor da altura da mídia (referente à propriedade `height`). A última mídia descrita na listagem é a mídia do ícone de interação, que nada mais é do que uma imagem do tipo png.

```
141         <media id="settings" type="application/x-ginga-settings">
142             <property name="opcao"/>
143         </media>
144
145         <media descriptor="dVideo" id="video" src="media/video.mpg" type="video/mpeg">
146             <area begin="6s" end="100s" id="bloco1"/>
147             <area begin="338s" end="400s" id="bloco2"/>
148             <area begin="800s" end="900s" id="bloco3"/>
149             <area begin="241s" end="299s" id="receita"/>
150             <property name="bounds"/>
151         </media>
152
153         <media descriptor="dIcone" id="icone" src="media/icone.png"/>
```

Listagem 6 - Descrição das mídias

Em seguida é definido na Listagem 7 o contexto “painel”. Decidiu-se aqui por utilizar um contexto de forma a garantir uma melhor estruturação do documento. Este contexto possui todas as mídias que são referentes ao painel de interação. É iniciado com a definição das portas de entrada do contexto, ou seja, as mídias que poderão ser acessadas de fora deste contexto. São definidas também todas as mídias que fazem parte do painel interativo:

- fundo: imagem no formato png do plano de fundo do painel interativo;
- botaoBloco1: imagem no formato png do botão referente ao primeiro bloco;
- botaoBloco2: imagem no formato png do botão referente ao segundo bloco;
- botaoBloco3: imagem no formato png do botão referente ao terceiro bloco;
- botaoReceita: imagem no formato png do botão referente à receita;
- botaoMapa: imagem no formato png do botão referente ao mapa;
- botaoContato: imagem no formato png do botão referente a informações de contato da produtora de multimeios.

Além destas, também foi definido um componente chave para o funcionamento da aplicação, que é um componente do tipo `<switch>`. Este componente se baseia em regras para definir qual mídia será acessada. Neste caso específico, ele utiliza as regras definidas no cabeçalho do documento e estas utilizam a propriedade definida no nó de configuração Ginga “settings”, descrito anteriormente, para decidir qual mídia será exibida no painel de informações. As mídias são páginas em formato HTML, que possuem as informações sobre cada bloco, e uma imagem do mapa do bairro em formato png. Com este bloco é encerrada a descrição deste contexto e também das mídias utilizadas no documento. O próximo passo é definir os elos que regem o sincronismo entre estas mídias.

```
155        <context id="painel">
156            <port id="pFundo" component="fundo"/>
157            <port component="informacao" id="pInformacao"/>
158            <port component="botaoBloco1" id="pBotaoBloco1"/>
159            <port component="botaoBloco2" id="pBotaoBloco2"/>
160            <port component="botaoBloco3" id="pBotaoBloco3"/>
161            <port component="botaoReceita" id="pBotaoReceita"/>
162            <port component="botaoMapa" id="pBotaoMapa"/>
163            <port component="botaoContato" id="pBotaoContato"/>
164
165            <media descriptor="dFundo" id="fundo" src="media/fundo.png"/>
166            <media descriptor="dBloco1" id="botaoBloco1" src="media/opcao1.png"/>
167            <media descriptor="dBloco2" id="botaoBloco2" src="media/opcao2.png"/>
168            <media descriptor="dBloco3" id="botaoBloco3" src="media/opcao3.png"/>
169            <media descriptor="dReceita" id="botaoReceita" src="media/opcao4.png"/>
170            <media descriptor="dMapa" id="botaoMapa" src="media/opcao5.png"/>
171            <media descriptor="dContato" id="botaoContato" src="media/opcao6.png"/>
172
173            <switch id="informacao">
174                <bindRule constituent="textoBloco1" rule="opcao1"/>
175                <bindRule constituent="textoBloco2" rule="opcao2"/>
176                <bindRule constituent="textoBloco3" rule="opcao3"/>
177                <bindRule constituent="textoReceita" rule="opcao4"/>
178                <bindRule constituent="textoMapa" rule="opcao5"/>
179                <bindRule constituent="textoContato" rule="opcao6"/>
180
181                <media descriptor="dInformacao" id="textoBloco1" src="media/bloco1.html"/>
182                <media descriptor="dInformacao" id="textoBloco2" src="media/bloco2.html"/>
183                <media descriptor="dInformacao" id="textoBloco3" src="media/bloco3.html"/>
184                <media descriptor="dInformacao" id="textoMapa" src="media/mapa.png"/>
185                <media descriptor="dInformacao" id="textoReceita" src="media/receita.html"/>
186                <media descriptor="dInformacao" id="textoContato" src="media/contato.html"/>
187            </switch>
188
189        </context>
```

Listagem 7 - Contexto "painel"

A descrição dos elos é talvez a parte mais importante de um documento NCL, pois é através deles que se controla toda a execução da aplicação e a relação entre as mídias. O elo que inicia a apresentação do ícone de interatividade pode ser visualizado na Listagem 8. Este é um elo simples que utiliza o conector “onBeginStartN” definido anteriormente no cabeçalho do documento. As tags `<bind>` relacionam as mídias com os papéis definidos no conector. Neste caso a mídia “icone” assume o papel `start` e o papel `onBegin` é assumido pela mídia “video”, ou seja, a figura do ícone de interação deve ser iniciada assim que o vídeo do programa começar a ser exibido.

```
196        <!-- Inicia o ícone de interatividade junto com o vídeo -->
197        <link xconnector="onBeginStartN">
198            <bind component="video" role="onBegin"/>
199            <bind component="icone" role="start"/>
200        </link>
```

Listagem 8 - Elo que inicia o ícone de interatividade

O segundo elo definido é o elo que define a apresentação do painel de interatividade como descrito na Listagem 9. Este elo utiliza o conector “onKeySelectionSetStartStop” e as mídias “icone”, “settings” e “painel”. Seu funcionamento é com se segue: enquanto o ícone de interação estiver ativado e a tecla “MENU” é pressionada é definido o valor inteiro 1 para a propriedade (definida aqui no parâmetro `interface`) “opcao”, o contexto “painel” é iniciado e a exibição do ícone de interação é interrompida. Quando um contexto é iniciado, o formatador NCL procura todas as portas descritas neste contexto e inicia todas as mídias referenciadas por estas. Isto demonstra que definir contexto, além de garantir uma melhor estruturação do documento, também facilita no trabalho de iniciar várias mídias ao mesmo tempo.

```
202         <!-- Aciona o painel de interatividade -->
203         <link xconnector="onKeySelectionSetStartStop">
204             <bind component="icone" role="onSelection">
205                 <bindParam name="tecla" value="MENU"/>
206             </bind>
207             <bind component="settings" role="set" interface="opcao">
208                 <bindParam name="valor" value="1"/>
209             </bind>
210             <bind component="painel" role="start"/>
211             <bind component="icone" role="stop"/>
212         </link>
```

Listagem 9 - Elo que inicia o painel interativo

O elo seguinte define o comportamento de redimensionamento do vídeo principal assim que o painel de interação é iniciado (Listagem 10). Utiliza o conector “onBeginSet”. Os valores de redimensionamento são passados para o parâmetro “valor” que define o valor da propriedade `bounds` da mídia “video”. Assim como nas regiões, aqui são usados valores percentuais de forma a garantir uma experiência de visualização idêntica em diferentes tamanhos de telas.

```
214         <!-- Redimensiona o video ao iniciar o painel -->
215         <link xconnector="onBeginSet">
216             <bind component="painel" role="onBegin"/>
217             <bind component="video" interface="bounds" role="set">
218                 <bindParam name="valor" value="19%,2%,65%,65%"/>
219             </bind>
220         </link>
```

Listagem 10 - Elo que redimensiona o vídeo ao inciar o painel de interação

Os elos definidos na Listagem 11 definem o comportamento ao selecionar o botão referente ao primeiro bloco no menu e ao acionar o botão vermelho no controle remoto. Quando o usuário estiver com o foco no botão do primeiro bloco e acionar a tecla “ENTER” (em ncl `VK_ENTER`) o painel de informações deve ser interrompido, o valor da propriedade “opcao” da mídia de configuração recebe o valor inteiro 1 e o painel é iniciado novamente com a nova configuração. O conector “onKeySelectionStopSetStart” é usado para descrever este comportamento. Os elos dos demais botões serão omitidos neste documento por apresentarem comportamento análogo a este descrito, com apenas pequenas modificações nos valores atribuídos à opção de configuração.

Para o início do trecho referente a cada bloco e à exibição da receita são utilizadas as teclas coloridas do controle remoto. O elo que define o comportamento referente ao primeiro bloco utiliza o conector “onKeySelectionStopStart”. Enquanto o painel estiver ativado e a tecla vermelha é acionada pelo usuário o vídeo é abortado e é iniciado novamente no momento definido pela âncora de conteúdo “bloco1”, que é exatamente o momento em que se inicia o primeiro bloco do programa televisivo. Os elos que definem o acionamento das demais teclas coloridas serão aqui omitidos por possuírem comportamento equivalente a este apresentado.

```
222         <!-- Botão do Bloco 1 -->
223         <link xconnector="onKeySelectionStopSetStart">
224             <bind component="painel" interface="pBotaoBloco1" role="onSelection">
225                 <bindParam name="tecla" value="VK_ENTER"/>
226             </bind>
227             <bind component="painel" interface="pInformacao" role="stop"/>
228             <bind component="settings" interface="opcao" role="set">
229                 <bindParam name="valor" value="1"/>
230             </bind>
231             <bind component="painel" interface="pInformacao" role="start"/>
232         </link>
233         <link xconnector="onKeySelectionStopStart">
234             <bind component="painel" interface="pBotaoBloco1" role="onSelection">
235                 <bindParam name="tecla" value="RED"/>
236             </bind>
237             <bind component="video" role="stop"/>
238             <bind component="video" interface="bloco1" role="start"/>
239         </link>
```

Listagem 11 - Elos referentes ao primeiro bloco do programa

Pela Listagem 12 podemos acompanhar os dois últimos elos definidos no documento. O primeiro define que enquanto o painel estiver acionado e o usuário acionar a tecla MENU do controle remoto, este é interrompido e o ícone de interação é novamente exibido na tela. O

segundo define que quando o ícone é acionado o vídeo é redimensionado para o tamanho inteiro da tela. Resumindo, estes dois elos definem o comportamento do momento em que o painel de interação é fechado.

```
322         <!-- Retornar -->
323         <link xconnector="onKeySelectionStopStart">
324             <bind component="painel" interface="pInformacao" role="onSelection">
325                 <bindParam name="tecla" value="MENU"/>
326             </bind>
327             <bind component="painel" role="stop"/>
328             <bind component="icone" role="start"/>
329         </link>
330         <link xconnector="onBeginSet">
331             <bind component="icone" role="onBegin"/>
332             <bind component="video" interface="bounds" role="set">
333                 <bindParam name="valor" value="0,0,100%,100%"/>
334             </bind>
335         </link>
```

Listagem 12 - Elos que definem o comportamento ao fechar o painel interativo

As seguintes imagens demonstram o programa em execução na máquina virtual VMware.

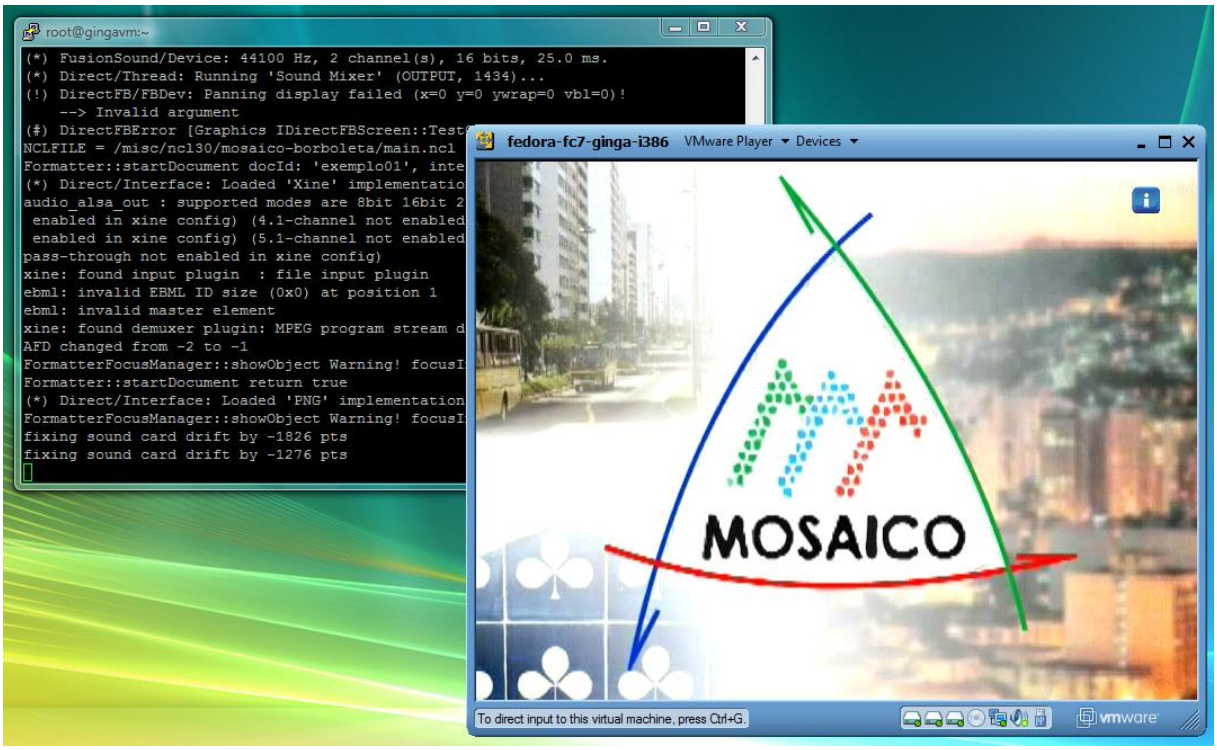

Figura 15 - Início do programa com ícone de interação

Figura 16 - Painel interativo com foco no primeiro botão

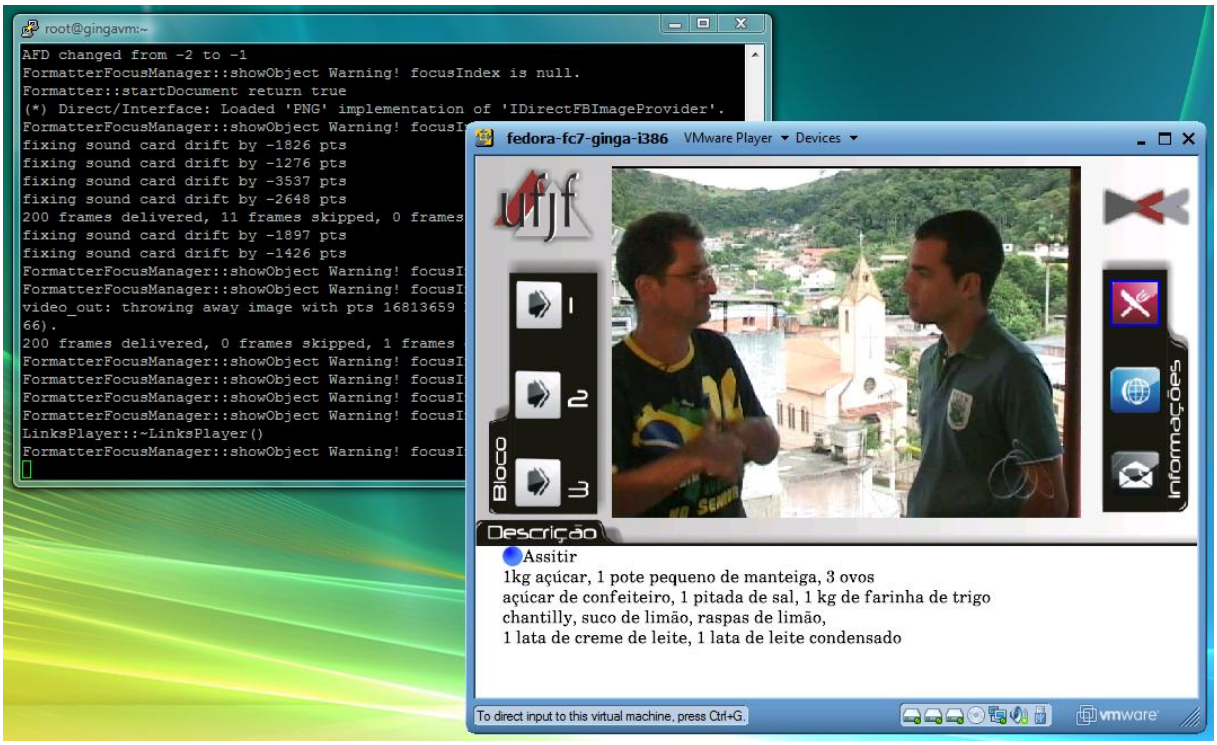

Figura 17 - Painel interativo com foco no botão "receita"

6.3. Dificuldades encontradas

Devido à ausência de ferramentas para geração do fluxo MPEG-2 e do carrossel de dados DSM-CC, as aplicações executam apenas com arquivos de mídias locais.

O desenvolvimento da aplicação apresentada foi iniciado através da ferramenta Composer. Porém esta apresentou um comportamento muito instável, causando diversos congelamentos e distorções entre as visões. Algumas vezes também não era possível salvar o trabalho ou abrir outro já salvo sem que se tivesse que finalizá-la e abri-la novamente. Relacionamentos avançados também não são possíveis de serem criados através das ferramentas visuais, sendo necessária a utilização da visão textual para tal fim. A despeito disto a ferramenta pode ser de grande utilidade se corrigidos os problemas, pois tem um ambiente agradável e de fácil visualização do modelo da aplicação. Como grande parte da aplicação deviria ser desenvolvida na visão textual e esta apresentava um desempenho insatisfatório, foi decidido utilizar o ambiente Eclipse com o plugin NCL Eclipse instalado.

Apesar de somente disponibilizar edição manual do texto, a ferramenta NCL Eclipse apresentou desempenho satisfatório, sendo indicada para usuários acostumados à edição de código baseado em XML através de editores de texto. Algumas mensagens de erro inconclusivas foram algumas vezes apresentadas ao tentar utilizar o recurso de assistência de código.

Para os testes de execução do documento foi utilizado inicialmente o emulador do Ginga-NCL. Este apresentou diversos problemas, como a impossibilidade de exibição de alguns elementos na tela, falta de suporte à transparência e problemas de interação com o controle remoto. Além disto, por diversas vezes a aplicação não respondia ao comando de execução e congelava sem motivo aparente e sem exibição de mensagens de erros. Depois de definido o elo que iniciava o painel de interação através da tag `<switch>`, o emulador não realizava a execução do código e apresentava código de erro inconclusivo.

Devido a estes problemas decidiu-se utilizar o set-top box virtual. O único problema desta abordagem é o maior tempo gasto para testar a aplicação, pois para isto é necessário realizar uma conexão sftp com a máquina virtual para enviar os arquivos necessários e uma conexão ssh para a execução do programa. Outro problema encontrado é que o *browser*

implementado não suporta folha de estilos, assim a formatação dos documentos HTML com informações sobre o programa foi perdida na execução.

7. Conclusões Finais

Neste trabalho foi feito uma pesquisa sobre a o sistema de TV digital de modo a facilitar o entendimento para futuros trabalhos. Foram descritos os sistemas internacionais de modo a criar uma interligação entre estes e o sistema brasileiro. Outro fator importante foi a discussão sobre o decreto que implantou a TV digital no país e com este está relacionado com as pesquisas e padrões utilizados.

Um ponto importante do trabalho foi a apresentação do *middleare* Ginga e suas linguagens de programação. Fez-se necessário a comparação com outros padrões de *middlewares* para demonstrar as escolhas utilizadas na construção do Ginga. Além do que foi visto neste trabalho são apresentadas referências para iniciar no desenvolvimento de aplicações para este sistema.

O advento da TV digital traz consigo uma gama enorme de possibilidades de pesquisas nas suas diversas camadas. Também é presenciado o nascimento de um novo ramo de comércio no desenvolvimento de softwares no Brasil. Isto carrega também novas fontes de pesquisas em engenharia de software para a criação de programas interativos de qualidade e com reuso das tecnologias.

Algumas sugestões para trabalhos futuros podem ser:

- Estudo sobre o suporte à múltiplos dispositivos de exibição e interação com a TV digital. Com o amadurecimento e disponibilização do *middleware* Ginga torna-se possível este estudo, podendo realizar experiências com celulares, *PDAs* e dispositivos como webcams, leitores de impressão digital etc.;
- *Social* TV, a TV com conteúdo compartilhado. Num conceito semelhante ao da web 2.0, usuários da TV digital poderiam participar de programas de TV ao vivo ou criar conteúdos para canais a partir de casa e revolucionar o universo televisivo;
- Um estudo sobre o papel da TV na sociedade e como as aplicações da TV digital podem ajudar no aspecto social do país;

- Estudo de interfaces visuais nas aplicações para a TV. Estas aplicações necessitam de uma interface mais amigável e de fácil visualização, pois interagem com pessoas de todos os tipos e de todas as idades;
- Otimização do carrossel de dados. Um ponto importante na distribuição do conteúdo interativo junto com a programação normal é o tempo de carregamento de uma aplicação. Se este tempo for muito grande o usuário tende a desistir do conteúdo. Desta forma um estudo na otimização do carrossel de dados faz-se necessário para diminuir este tempo de carregamento.

8. Bibliografia

ABNT - Associação Brasileira de Normas Técnicas. (2008). *Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital Parte 2: Ginga-NCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações.* NBR 15606-2.

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

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

Advanced Television Systems Committee, Inc. (2005). *Digital Audio Compression Standard (AC-3, E-AC-3) Revision B.* Washington, D.C.

ARIB – Association of Radio Industries and Businesses. (2004). *Data Coding and Transmission Specifications for Digital Broadcasting Volume 2: XML-Based Multimedia Coding Schema.* STD-B24 Versão 4.

ATSC - Advanced Television Systems Committee. (2005, Agosto). ATSC Standard: Advanced Common Application Platform (ACAP). *Padrão A/101* . Washington, EUA.

Becker, V., & Moraes, Á. (2003). *A necessidade da inovação no conteúdo televisivo digital: uma proposta de comercial para TV interativa.*

Converge Comunicações. (2008, Outubro 29). *TI INSIDE Online: Sun vai liberar Ginga Java para desenvolvedores até semana que vem.* Retrieved Novembro 10, 2008, from TI INSIDE Online: http://www.tiinside.com.br/News.aspx?ID=99997&C=265

CPqD. (2006). *Arquitetura de referência.* Campinas, SP.

CPqD. (2006). *Especificação técnica de referência.* Campinas, SP.

CPqD. (2006). *Modelo de referência.* Campinas, SP.

DTV. (n.d.). *Entenda a TV digital*. Retrieved dezembro 07, 2008, from DTV - Site oficial da TV digital brasileira: http://www.dtv.org.br/materias.asp?menuid=3&id=5

ETSI – European Telecommunications Standards Institute. (2005, Maio). Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.1.1. *ETSI TS 102 B12* .

Filho, G. L., Leite, L. E., & Batista, C. E. (2007). Ginga-J: The Procedural Middleware for the Brazilian Digital TV System. *Journal of the Brazilian Computer Society , 13* (4), pp. 47-56.

Junger, H. S. (2008). *Um Framework para o Módulo de Sintonização do Middleware Ginga.* PUC-Rio.

LAVID-UFPB e Telemidia-PUC-RJ. (n.d.). *Ginga Digital TV Middleware Specification*. Retrieved deembro 07, 2008, from TV interativa se faz com Ginga: http://www.ginga.org.br/

Morris, S. (n.d.). *An Introduction To Xlets*. Retrieved dezembro 07, 2008, from TV Without Borders: http://www.interactivetvweb.org/tutorials/javatv/xlet_intro

Neto, C. d., Soares, L. F., Rodrigues, R. F., & Barbosa, S. D. (2007). *Construindo Programas Audiovisuais Interativos Utilizando a NCL 3.0 e a Ferramenta Composer.* PUC-Rio, Rio de Janeiro, RJ.

OPENCABLE. (2005). *Opencable Application Plataform Specification.* OC-SP-OCAP1.0-I16-050803.

República Federativa do Brasil. (n.d.). *Comunidade Ginga: Ginga-NCL*. Retrieved Novembro 20, 2008, from Portal do Software Público Brasileiro: http://www.softwarepublico.gov.br/dotlrn/clubs/ginga/gingancl/one-community?page_num=0

Rodrigues, R. F., & Soares, L. F. (2006). *Produção de Conteúdo Declarativo para TV Digital.* Rio de Janeiro, RJ.

Silva, L. D., Batista, C. E., Leite, L. E., & Filho, G. L. (2007, Outubro). Suporte para desenvolvimento de aplicações multiusuário e multidispositivo para TV Digital com Ginga. *T&C Amazônia* , 75-84.

Soares, L. F., & Barbosa, S. D. (2008). TV digital interativa no Brasil se faz com Ginga: Fundamentos, Padrões, Autoria Declarativa e Usabilidade. In T. Kowaltowski, & K. Breitman, *Atualizações em Informática* (pp. 105-174). Rio de Janeiro, RJ: Editora PUC-Rio.

Soares, L. F., & Rodrigues, R. F. (2005). *Nested Context Model 3.0 Part 1 - NCM Core.* Monografias em Ciência da Computação, PUC-Rio, Departamento de Informática, Rio de Janeiro, RJ.

Sun Microsystems. (n.d.). Retrieved novembro 10, 2008, from Java Media Framework API (JMF): http://java.sun.com/javase/technologies/desktop/media/jmf/

Sun Microsystems. (n.d.). *Developer Resources for Java Technology*. Retrieved Novembro 2008, from Sun Developer Network: http://java.sun.com/

Sun Microsystems, Inc. (2008, Março 4). *Sun Microsystems And SBTVD Forum To Develop Open-Source Java Solution For Brazil's Digital TV System*. Retrieved Setembro 14, 2008, from Sun Microsystems: http://www.sun.com/aboutsun/pr/2008-03/sunflash.20080304.2.xml

Telemidia-PUC-Rio. (n.d.). *Ginga-NCL - Declarative DTV Middleware*. Retrieved dezembro 07, 2008, from Ginga-NCL: http://www.gingancl.org.br/

Tonieto, M. T. (2006). *Sistema brasileiro de TV digital - uma análise política e tecnológica na inclusão social.* Dissertação de mestrado, Universidade Estadual do Ceará, Centro de Ciências e Tecnologia, Fortaleza-CE.

W3C - World Wide Web Consortium. (2005). *Synchronized Multimedia Integration Language (SMIL 2.1).* W3C Recommendation.

W3C - World Wide Web Consortium. (2002). *XHTML™ 1.0 - The Extensible HyperText Markup Language (Second Edition).* W3C Recommendation.

XleTView. (n.d.). Retrieved Novembro 20, 2008, from XleTView: http://www.xletview.org/