Ferramentas de Acessibilidade para TV Digital Interativa com Java™

Marcos Vinícius Henke Arnoldo

Orientadora: Karen Selbach Borges

Universidade Luterana do Brasil (ULBRA) – Curso Tecnológico de Análise e Desenvolvimento de Sistemas

marcos.henke@gmail.com, www.b4it.com.br

Resumo. *O objetivo deste trabalho é disponibilizar protótipos de ferramentas que fazem uso da interatividade local da TV digital para a comunidade de desenvolvedores visando a disseminação da tecnologia no Brasil. Para tanto, foi estudada a linguagem de programação Java™ e as APIs que compõem o padrão de middleware GEM. A contribuição deste trabalho à sociedade é o desenvolvimento de ferramentas de acessibilidade para portadores de determinadas necessidades especiais. Para identificação das necessidades e ferramentas foram analisados o Decreto-lei 5296 (Lei de Acessibilidade), a norma ABNT NBR 15290 (Acessibilidade em Comunicação na Televisão) e outras ferramentas semelhantes já disponíveis para computadores pessoais.*

1 Introdução

A televisão é um meio de comunicação popular presente em 97,03% dos 53,1 milhões de domicílios brasileiros, sendo inclusive comum a existência de mais de um aparelho em diversas delas. Nas classes sociais mais baixas (D e E) esse índice chega a 93,31%, como podemos ver na Tabela 1, e muitas vezes é a única forma de acesso à informação destas famílias.

Tabela 1: Proporção de domicílios que possuem equipamentos de Tecnologia da Informação e Comunicação (NIC, 2007)

Percentual (%)		Televisão	Antena parabólica	TV a cabo	Rádio	Computador	Acesso à Internet
Total		**97,03**	**15,93**	**5,36**	**89,61**	**19,3**	**14,49**
REGIÕES DO PAÍS	SUDESTE	97,8	13,36	7,82	92,66	23,83	18,74
	NORDESTE	95,89	18,29	1,73	85,63	8,38	5,54
	SUL	96,86	16,53	4,6	94,73	24,24	16,9
	NORTE	95,7	18,84	2,74	75,66	9,97	6,15
	CENTRO-OESTE	96,71	20,46	3,22	83,8	18,35	13,05
RENDA FAMILIAR	ATÉ R$300	86,76	10,27	0	77,27	1,57	0,46
	R$301-R$500	95,93	11,45	0,9	84,85	2,36	1,22
	R$501-R$1000	97,86	14,44	2,71	90,49	13,73	8,9
	R$1001-R$1800	98,53	21,94	9,82	95,59	36,27	27,33
	R$1801 OU MAIS	99,82	25,26	20,57	96,34	59	50,53
CLASSE SOCIAL	A	100	21,51	45,73	99,58	82,79	81,49
	B	99,93	25,27	17	99,13	62,18	51,22
	C	99,74	16,62	4,51	95,14	18,55	12,1
	DE	93,31	11,76	0,79	80,62	2,76	1,61

A sociedade não é mais baseada na mão-de-obra e no capital, e sim na informação e no conhecimento. Quem não participa da "Sociedade da Informação" é considerado excluído social e o acesso à Internet ampliou ainda mais as diferenças,

devido ao elevado custo de aquisição e manutenção de um computador com acesso à rede. Desta forma, as classes mais altas adquirem maior conhecimento de forma mais rápida e conquistam as melhores oportunidades, enquanto as classes menos favorecidas permanecem desprovidas deste benefício (MONTEZ & BECKER, 2006).

Ciente dessa necessidade de informação, o governo brasileiro mantém diversos programas de inclusão social e está popularizando o computador e a Internet como meio de acesso à informação, principalmente em escolas públicas, mas não se pode ignorar a televisão e o rádio como mais um mecanismo de inclusão.

Dentro deste contexto, a TV digital, que significa transmissão de vídeo, áudio e dados em formato binário independente do meio, possui tecnologias que permitem interatividade e multiprogramação, por exemplo, e que podem auxiliar de forma expressiva a inclusão social através da inclusão digital. No lançamento oficial das transmissões digitais, ocorrido em 02 de dezembro de 2007 em São Paulo, o custo para possuir um receptor foi considerado impraticável pelos principais formadores de opinião do setor. Porém, estão sendo executadas medidas de popularização da TV digital através de campanhas educativas, incentivos a fabricantes de TVs e receptores (*set-top-boxes*) para redução do custo e linhas de financiamento, viabilizando assim o acesso das classes mais baixas a esse meio de acesso à informação.

Aprofundando-se mais na questão da inclusão social, de acordo com o Decreto-lei 5296, que define a Lei de acessibilidade, a TV digital deverá ser dotada de recursos para portadores de necessidades especiais visuais e auditivas, e as funcionalidades disponíveis no novo sistema facilitam esta tarefa. Outro ponto a considerar seria o de telespectadores com dificuldades de visão que, por exemplo, compartilham um mesmo aparelho com outras pessoas, sendo possível até mesmo que portadores de outras necessidades diferentes o utilizem.

Levando isto em consideração, este trabalho propõe como objetivo uma solução para esta questão através do desenvolvimento de uma ferramenta para criação de Perfis de Acessibilidade, além de divulgar a tecnologia empregada para a comunidade de desenvolvedores utilizando ferramentas gratuitas de desenvolvimento e testes disponíveis até o momento.

O capítulo 2 apresenta os principais conceitos e tecnologias que envolvem o Sistema Brasileiro de Televisão Digital (SBTVD) e algumas considerações sobre inclusão social e acessibilidade. No capítulo 3 serão expostas as ferramentas desenvolvidas utilizando a linguagem Java™ e as APIs (*Application Program Interfaces*, Interfaces do Programa de Aplicação) utilizadas. No capítulo 4 serão relatadas as conclusões derivadas da produção deste trabalho.

2 Referencial Teórico

Neste tópico será apresentada a pesquisa bibliográfica sobre os temas estudados para a construção deste trabalho, os principais conceitos sobre televisão digital interativa e acessibilidade, assim como a contextualização dos padrões e bibliotecas existentes para o desenvolvimento de aplicações interativas.

A TV Digital Interativa é viabilizada através de um conjunto de equipamentos e sistemas e envolve diversas organizações e interesses. Para o telespectador, que nesse ponto passa a ser também um usuário, a interação se dá através de um conjunto de

hardware e *software*, onde o *hardware* é o aparelho receptor (*set-top-box*) ou TV com este dispositivo embutido e o *software* engloba o *middleware* e aplicações associadas a serviços.

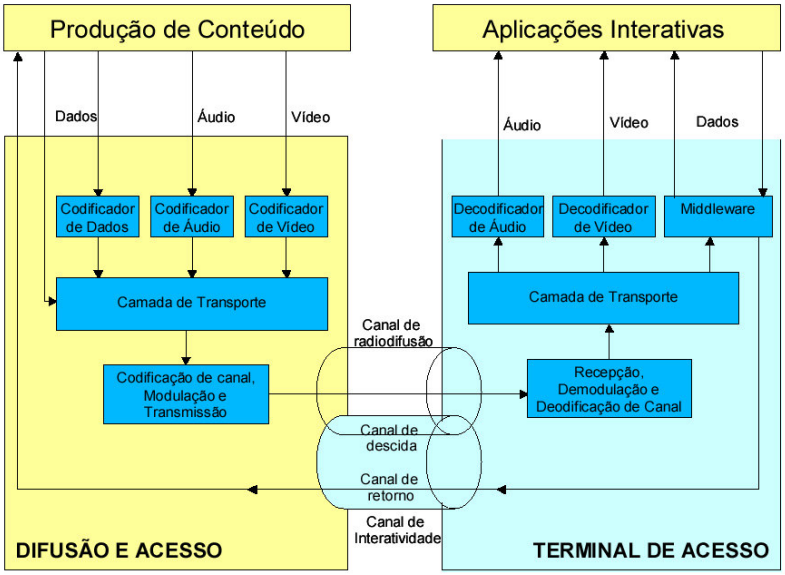

Figura 1: Diagrama de fluxo de informação (MC, 2007)

Pode-se observar na Figura 1 o fluxo de informação entre o produtor de conteúdo e as aplicações interativas que são executadas nos terminais de acesso. Os dados, o áudio e o vídeo são codificados (compactados), modulados e transmitidos pelas emissoras através do canal de radiodifusão e captados pelos receptores ou TVs com este recurso integrado. O aparelho receptor, ou terminal de acesso, efetua então a demodulação e decodificação do conteúdo e o exibe apropriadamente. Quando as aplicações interativas necessitarem enviar dados do terminal de acesso para as emissoras, a operação será efetuada através do Canal de Retorno e a resposta será recebida pelo chamado Canal de Descida. Estes dois canais formam então o Canal de Interatividade, elemento opcional nos *set-top-boxes* (MC, 2007).

2.1 Middleware

Middleware é o software que opera entre as aplicações e o sistema operacional do receptor, como podemos ver na Figura 2. Oferece um serviço padronizado para as aplicações ocultando particularidades das camadas inferiores (compressão, transporte, modulação) e permite a portabilidade para qualquer receptor digital que contenha o *middleware* instalado (MONTEZ & BECKER, 2006).

2.2 Sistemas e Padrões

O governo brasileiro testou, através de entidades como ABERT/SET, os três principais padrões mundiais de TV digital existentes e os resultados se encontram disponíveis no site da SET (2000). Devido a interesses políticos, financeiros, tecnológicos e culturais, houve indefinição na decisão sobre a escolha de um dos padrões ou o desenvolvimento de um próprio. A burocracia demasiada para obtenção da documentação e recursos para os projetos de pesquisa resultou no cancelamento de diversas linhas de pesquisa em universidades.

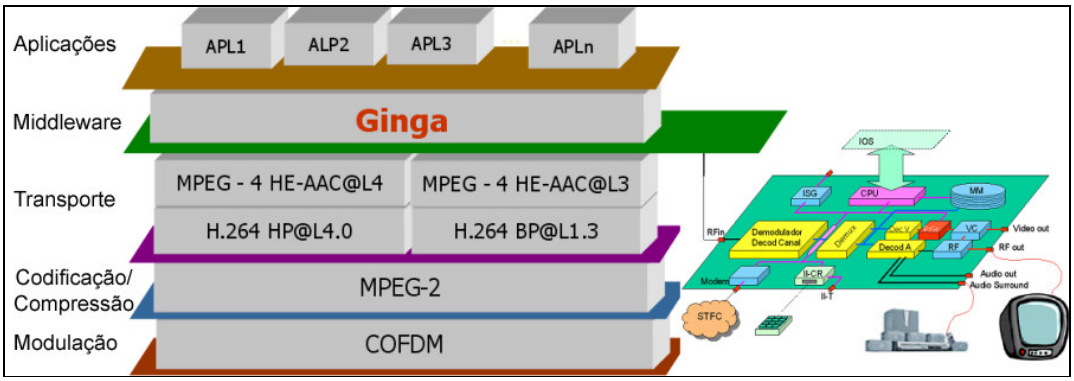

Figura 2: Camadas do aparelho receptor do SBTVD (GINGA, 2007)

A seguir, será feita uma apresentação sucinta dos principais padrões mundiais existentes e a decisão brasileira, que iniciou as transmissões em dezembro de 2007 (MC, 2007).

2.2.1 Padrão Europeu (DVB-MHP)

- Sistema de transmissão: DVB (*Digital Vídeo Broadcasting*)

  O enfoque inicial deste sistema foi a interatividade em detrimento da qualidade.

- Middleware: MHP (*Multimedia Home Platform*)

  Baseado no uso de uma JVM (*Java Virtual Machine*) e um conjunto de APIs possibilita que programas feitos em Java acessem recursos do receptor de forma padronizada. Uma aplicação DVB que utiliza Java é chamada DVB-J. Na especificação 1.1 do MHP foi introduzido o DVB-HTML. Possibilita, dentre diversos outros recursos, o *download* de aplicações que são armazenadas em memória persistente e acesso a *smart-cards*.

O padrão DVB-MHP é utilizado, além dos países europeus, na Austrália, Malásia, Índia, África do Sul e Honk-Kong na China (MONTEZ & BECKER, 2006).

2.2.2 Padrões Americanos (ATSC-DASE e OCAP)

- ATSC-DASE: Terrestre
  - Sistema de transmissão: ATSC (*Advanced Television Systems Committee*)

    O principal enfoque deste sistema é a transmissão em alta definição (HDTV, *High Definition Television*), em detrimento da interatividade e multiprogramação. Apresenta problemas na recepção através de antenas internas e não suporta recepção em dispositivos móveis, como por exemplo, em celulares.

  - Middleware: DASE (*Digital TV Application Software Environment*)

    Utiliza JVM (*Java Virtual Machine*) e permite o uso de linguagens declarativas, como o HTML. É utilizado nos Estados Unidos, Canadá, Coréia do Sul e Taiwan (MONTEZ & BECKER, 2006).

- CableLabs-OCAP (*OpenCable Applications Platform*): Baseado no MHP europeu é o *middleware* utilizado nas transmissões de TV a cabo dos Estados Unidos e possui grande influência na questão da padronização de *middlewares* (MORRIS & SMITH-CHAIGNEAU, 2005).

2.2.3 Padrão Japonês (ISDB-ARIB)

- Sistema de transmissão: ISDB (*Integrated Services Digital Broadcasting*)

  As vantagens deste sistema em relação aos demais são a flexibilidade de operação, boa recepção em antenas internas ou áreas encobertas e o bom suporte a aplicações móveis.

- Middleware: ARIB (*Association of Radio Industries and Businesses*)

  Definido pela organização de mesmo nome, esse *middleware* é formado por alguns padrões, como:

  - ARIB STD-B23 (*Application Execution Engine Platform for Digital Broadcasting*), baseada no *middleware* MHP, indica uma tendência de entrar em conformidade com o DVB-MHP, como veremos adiante.
  - ARIB STD-B24 (*Data Coding and Transmission Specification for Digital Broadcasting*), define uma linguagem declarativa denominada BML (*Broadcast Markup Language*), baseada em XML (*Extensible Markup Language*), é usada para especificação de serviços multimídia para TV digital (TONIETO, 2006).

O padrão ISDB-ARIB é utilizado somente no Japão, até então (MONTEZ & BECKER, 2006).

2.2.4 Sistema Brasileiro de Televisão Digital (SBTVD)

O governo brasileiro anterior a 2003 defendia a escolha e aplicação integral de um dos três padrões de transmissão terrestre citados nos tópicos anteriores. A necessidade de baixo custo dos receptores para tornar acessível às camadas mais pobres da população também influenciou na decisão de desenvolver um padrão nacional livre de licenças e *royalties*, a exemplo da decisão tomada pela China (MORENO, 2006).

Como houve atraso no cronograma devido a diversos fatores, em 2006 foi determinado como padrão de transmissão o japonês ISDB com algumas melhorias, chamado ISDB@Tb, e como *middleware* o brasileiro Ginga, apesar de seu nome não ser citado no decreto 5.820 de junho de 2006 (MONTEZ & BECKER, 2006).

2.2.4.1 Ginga

Ginga é o nome do *middleware* brasileiro desenvolvido para executar sobre o padrão ISDB@Tb. O padrão se subdivide em:

- Ginga-NCL: ambiente de apresentação. Assim como o BML japonês e o DVB-HTML, é uma linguagem de marcação como o XML, porém mais adequada para manipular hipermídia (MORENO, 2006).
- Ginga-J: ambiente de execução. Semelhante ao DVB-J, é uma linguagem procedural derivada do padrão japonês ARIB B23 e projetada para ser

compatível com o GEM (*Globally Executable MHP*, MHP Globalmente Executável) em primeira instância.

Este trabalho foi desenvolvido considerando que o *middleware* brasileiro suportaria as APIs contidas no GEM. Porém devido ao custo de licenciamento das bibliotecas proprietárias contidas no referido padrão, em março de 2008 o Fórum SBTVD e a Sun Microsystems (SUN), fabricante da linguagem de programação Java™ e da API Java TV™, assinaram um memorando para unir forças no desenvolvimento de uma plataforma *open-source* de desenvolvimento para TV digital. Com esta iniciativa a nova especificação, desenvolvida sobre a mesma base que outros padrões mundialmente adotados, oferecerá uma alternativa de baixo custo e será disponibilizada mundialmente através do Fórum SBTVD (SUN MICROSISTEMS).

Apesar da definição oficial que as APIs serão substituídas, durante a fase de construção deste protótipo a utilização do GEM era o que havia de concreto e disponível para a comunidade de desenvolvedores, por isso a decisão de manter a sua utilização. Acredita-se que será necessário um esforço futuro para conversão das aplicações para que elas se tornem compatíveis com o novo padrão e ainda não há estudo disponível sobre o impacto das alterações.

Considerando, então, ainda a utilização do GEM, a Figura 3 ilustra a compatibilidade do Ginga-J com outros sistemas, onde a API Vermelha indica recursos suportados somente no padrão brasileiro (identificado na imagem como ISDB@B e chamado atualmente de ISDB@Tb). A API Amarela representa recursos que podem ser adaptados a outros padrões e a API Verde simboliza recursos compatíveis com os padrões que suportam o GEM (GINGA, 2007). No próximo tópico serão citadas as bibliotecas utilizadas no protótipo.

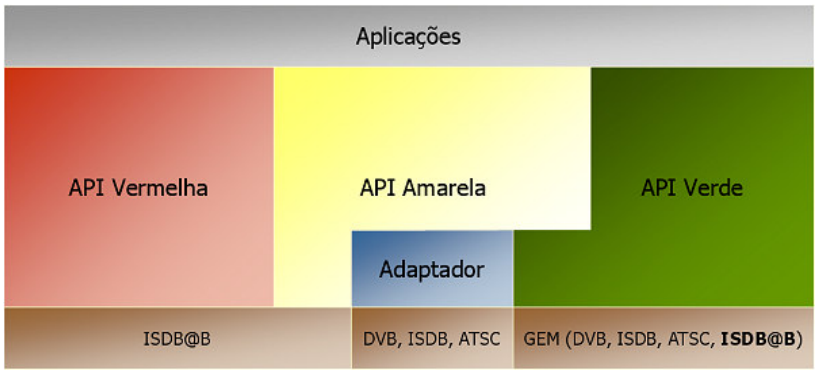

Figura 3: Compatibilidade do Ginga-J com outros sistemas (GINGA, 2007)

2.3 Bibliotecas de suporte a middlewares

Quando os *middlewares* surgiram já existiam várias bibliotecas e APIs para trabalhar com multimídia para a Internet e equipamentos de vídeo; estes recursos foram aproveitados e adaptados ao contexto da TV digital. No momento em que a Sun Microsystems desenvolveu a API Java TV™, esta foi prontamente adotada pelos *middlewares* (MONTEZ & BECKER, 2006).

A API Java TV™, diferentemente de aplicações comuns Java, utiliza uma única JVM para controlar o ciclo de vida dos *Xlets*, a exemplo dos *Applets* utilizados em navegadores *Web*, com a diferença da inclusão do estado Pausado. Provê um meio da

aplicação se comunicar com o seu ambiente de execução e possui um gerente de aplicações que controla o estado do *Xlet*, que pode alternar entre cinco estados principais: Não-carregado, Carregado (*Loaded*), Pausado (*Paused*), Iniciado (*Started*) e Destruído (*Destroyed*), como podemos ver na Figura 4 (MORRIS & SMITH-CHAIGNEAU, 2005, p. 64).

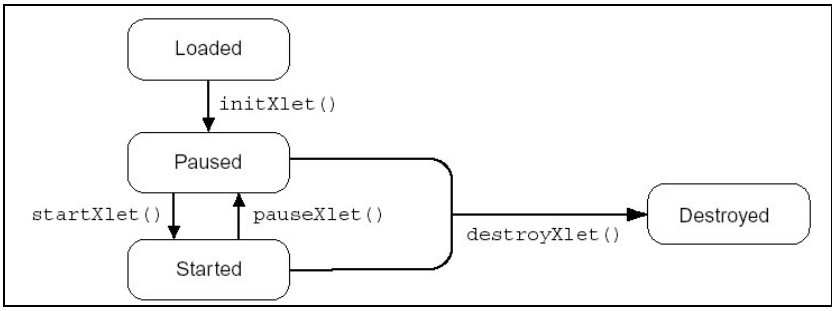

Figura 4: Ciclo de vida de um Xlet (MORRIS & SMITH-CHAIGNEAU, 2005)

Para este trabalho, que considerou o suporte ao GEM conforme explicitado no capítulo anterior, fez-se uso da biblioteca HAVi. Criada por um grupo de fabricantes incluindo Matsuchita/Panasonic, Sharp, Sony e Toshiba com o intuito de integrar TVs, DVDs e câmeras de vídeo, utiliza uma API Java para funções do controle remoto, displays e gráficos de TV e requer pagamento de licença para sua utilização e distribuição (MONTEZ & BECKER, 2006, p. 124).

Em fevereiro de 2009 foi disponibilizada ao público a especificação da nova API efetuada pela SUN, chamada Java DTV, e é aguardada a liberação de sua implementação pelo Fórum do SBTVD.

2.4 Tipos de Receptores Digitais

Os receptores possuem limitações de memória, resolução gráfica e capacidade de armazenamento quando comparados aos computadores modernos e exigem que os desenvolvedores considerem estas limitações de ambiente ao projetar suas aplicações. Devem considerar ainda os diferentes tipos de receptores e os recursos que cada categoria suporta.

No relatório dos consórcios do SBTVD os pesquisadores do *middleware* brasileiro Ginga propõem a divisão dos receptores de acordo com as suas funcionalidades, visando atender a diferentes segmentos da sociedade e suas necessidades (TONIETO, 2006). As diferenças técnicas consistem basicamente entre suportar interatividade, canal de retorno, gravação do conteúdo, integração com dispositivos móveis, comando de voz, captura de vídeo e suporte a funcionalidades do *middleware*.

2.5 Interatividade

Interatividade, para a TV digital, significa que o usuário pode influenciar na forma e conteúdo do que está assistindo, se comunicar e realizar diversos tipos de operações através dos serviços e aplicações disponíveis. Os níveis de interatividade, segundo Fernandes et al. (apud TONIETO, 2006), podem ser classificados em local e remoto, sendo este último subdividido entre intermitente e permanente.

2.6 Serviços e Aplicações

No ambiente da TV digital os serviços são executados nos servidores das emissoras e as aplicações no aparelho receptor. Conforme mencionado no tópico anterior, a interatividade pode ser local ou remota; então, para haver a interatividade remota, as aplicações necessitam se comunicar com serviços através do canal de interatividade.

Os tipos de serviços interativos podem ser classificados, segundo Freed (apud TONIETO, 2006), em: TV Avançada (*Enhanced TV*); TV Individualizada (*Individualized TV*); *Personal TV,* também chamado de PVR (*Personal Video Recorder*) ou DVR (*Digital Video Recorder*); EPGs, Guias Eletrônicos de Programação (*Eletronic Program Guides*); Internet TV; *On-Demand TV*; *Play TV* e *Banking & Retail,* aplicações de banco e comércio eletrônico.

Já as aplicações interativas podem ser classificadas em: *T-learning* ou *Educational TV,* aplicações de Ensino a Distância (EAD); *Community TV*, votações, veiculação de informações, suporte a comunidades virtuais, informações direcionadas a grupos específicos; *Global TV*, programação internacional com tradução automática de língua; *T-Commerce*, comércio eletrônico através da TV; T-Governo; *T-mail e* TV saúde. As aplicações governamentais possuem forte característica de inclusão social, que é também preocupação deste trabalho, como será descrito a seguir.

2.7 Inclusão Social e Acessibilidade

A inclusão social através da inclusão digital é uma maneira de disponibilizar informação às pessoas de baixo poder aquisitivo, portadores de deficiência física e idosos, entre outros, que não tem acesso a educação formal, Internet e outros meios. Sendo a TV digital um meio de se obter informação e, através da interatividade, participar da elaboração de conteúdo, é importante garantir que o maior número de pessoas possa usufruir desses benefícios.

O Decreto-lei 5296 de 2 de dezembro de 2004, a Lei de Acessibilidade, define o seguinte em relação à televisão:

> Art. 52. Caberá ao Poder Público incentivar a oferta de aparelhos de televisão equipados com recursos tecnológicos que permitam sua utilização de modo a garantir o direito de acesso à informação às pessoas portadoras de deficiência auditiva ou visual.
>
> Parágrafo único. Incluem-se entre os recursos referidos no caput:
>
> I - circuito de decodificação de legenda oculta;
>
> II - recurso para Programa Secundário de Áudio (SAP); e
>
> III - entradas para fones de ouvido com ou sem fio.
>
> § 2o A regulamentação de que trata o caput deverá prever a utilização, entre outros, dos seguintes sistemas de reprodução das mensagens veiculadas para as pessoas portadoras de deficiência auditiva e visual:
>
> I - a subtitulação por meio de legenda oculta;
>
> II - a janela com intérprete de LIBRAS; e
>
> III - a descrição e narração em voz de cenas e imagens.
>
> § 3o A Coordenadoria Nacional para Integração da Pessoa Portadora de Deficiência - CORDE da Secretaria Especial dos Direitos Humanos da

> Presidência da República assistirá a ANATEL no procedimento de que trata o § 1o.
>
> Art. 56. O projeto de desenvolvimento e implementação da televisão digital no País deverá contemplar obrigatoriamente os três tipos de sistema de acesso à informação de que trata o art. 52.(BRASIL, 2004)

A norma ABNT NBR 15290:2005 (NBR15290, 2005), Acessibilidade em Comunicação na Televisão, define os padrões de formatação e exibição de recursos como legendas ocultas (*Closed Captions*), SAP (*Secondary Audio Program*, Programa de Áudio Secundário), janela de exibição de LIBRAS (sistema lingüístico de comunidades surdas do Brasil) e cores de fundo, entre diversos outros elementos.

Este trabalho visa explorar os recursos avançados de interatividade local disponíveis nos receptores digitais de televisão compatíveis com Java TV e GEM, oferecendo ferramentas extras de acessibilidade não mencionadas no Decreto-lei 5296, mas já disponíveis em sistemas operacionais contemporâneos, como Lente de Aumento, Teclado e Controle Remoto virtuais, configuráveis através de perfis. Até o momento, não se tem conhecimento de trabalhos semelhantes relacionados à TV digital.

3 Apresentação e Descrição do Gerenciador de Perfis de Acessibilidade

Com o objetivo citado no capítulo 1 de tornar a experiência de uso da TV digital acessível a um maior número de pessoas e facilitar a entrada de dados em programas interativos foram desenvolvidas as ferramentas Teclado Virtual, Controle Remoto Virtual e Lente de Aumento, utilizando tecnologia Java através de APIs presentes no padrão de *middleware* GEM.

Considerando a questão de diferentes necessidades para diferentes pessoas que assistem a um mesmo aparelho, foi implementada a funcionalidade Perfis de Acessibilidade, onde podem ser configurados até três perfis diferentes, habilitando os recursos desejados para cada um. Estes perfis podem ser gerenciados e aplicados através de um Menu de Acessibilidade, ativado pelo controle remoto, e são persistidos em um arquivo XML armazenado no *set-top-box*. Este recurso permite ainda, por exemplo, que um perfil seja ativado em determinados programas e desativado em outros por um mesmo telespectador.

Cada Perfil de Acessibilidade suporta a configuração dos recursos Lente de Aumento, Legendas (*Closed Captions*), Mudo, Volume alto e Controle Remoto Virtual. Os recursos Controle Remoto Virtual e Lente de Aumento podem ainda ser ativados e desativados rapidamente e de forma independente, sem a necessidade da configuração de um perfil para isso, e também são acessíveis através do menu. Estas ferramentas serão detalhas no próximo tópico.

3.1 Ferramentas Implementadas

Neste capítulo serão descritas as ferramentas de acessibilidade desenvolvidas e suas particularidades:

a) Teclado Virtual: consiste em um mecanismo de entrada de caracteres para campos de texto exibido na tela da TV, podendo ser utilizado em qualquer aplicação que necessite digitação de texto pelo usuário, como formulários de *login*, *T-commerce*, *T-mail* e muitos outros. A motivação para o desenvolvimento desta ferramenta é a rara disponibilidade de um dispositivo de

digitação conectado a um *set-top-box*, como um teclado alfanumérico, sendo que na quase totalidade dos casos o usuário dispõe apenas do controle remoto e a digitação se dá através do pressionamento consecutivo das teclas numéricas ou direcionais, a exemplo da digitação em aparelhos celulares. Este recurso é comumente encontrado em *videogames*, onde apenas se dispõe de um *joystick* para a operação.

Pode-se observar na Figura 5 um exemplo de aplicação utilizando o Teclado Virtual para digitação de texto. Ao focar o campo o teclado é exibido e ao escolher a tecla virtual "OK" ou "ENTER" o teclado é ocultado.

Figura 5: Teclado Virtual

Segundo Schwalb (2003), existem interfaces próprias para a implementação de dispositivos virtuais, como a *HKeyboardInputPreferred* da API HAVi, e classes e métodos para verificar as capacidades do sistema, ou seja, se o *set-top-box* possui um teclado ou *mouse* conectado. Para este protótipo não foi requerida sua utilização, mas para trabalhos futuros estuda-se utilizar a classe específica para a atividade da nova API Java DTV, chamada *com.sun.dtv.ui.Keyboard*.

b) Controle Remoto Virtual: devido à dificuldade de algumas pessoas para enxergar os pequenos botões de um controle remoto, principalmente idosos e portadores de hipermetropia, foi desenvolvido este protótipo que visa possibilitar maior interatividade e por conseqüência inclusão digital destas pessoas, sem a necessidade de aquisição de um controle especial. É exibida uma interface na tela da TV que reproduz os comandos de um controle remoto real, como ilustrado na Figura 6. A visualização das teclas é, então, facilitada para o telespectador, principalmente em telas com tamanho superior a 20" (vinte polegadas). Este precisará somente utilizar os botões direcionais e "OK" do controle real para selecionar e disparar os comandos desejados no controle virtual.

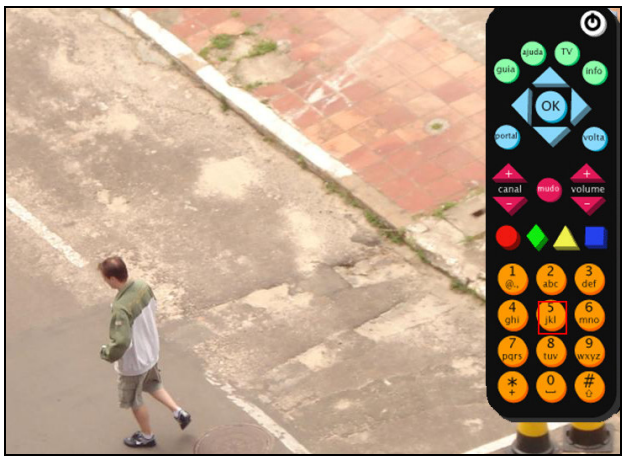

Figura 6: Controle Remoto Virtual

Quando o telespectador seleciona o botão desejado e o pressiona é gerado um evento contendo o código da tecla e redirecionado ao *middleware*, que se encarregará de processar o comando apropriadamente.

Como a proposta deste trabalho é apresentar apenas um protótipo, não foi realizado um estudo de ergonomia e *design* em relação às cores e tamanho de texto adequado para maximizar o benefício deste recurso, ficando como sugestão para implementações futuras. A imagem utilizada de fundo do Controle Remoto Virtual é distribuída juntamente com o *Ginga-NCL Player* (GINGA, 2007). A classe *HRcCapabilities* da API HAVi possui um método chamado *getRepresentation()*, que retorna o texto, a cor e o símbolo de cada tecla, sendo possível exibir corretamente um botão em uma determinada plataforma (SCHWALB, 2003). Nos testes efetuados no emulador este método retornou nulo. Conforme a documentação da API Java DTV será disponibilizada a classe *com.sun.dtv.ui.RemoteControl*, que representa o controle remoto, mas até o momento não há maiores informações de suas características.

c) Lente de Aumento: na concepção desta ferramenta foi definido que seriam efetuados o redimensionamento e reposicionamento do vídeo para proporcionar uma melhor visualização para idosos e portadores de miopia – dificuldade para ver ao longe. As APIs JavaTV e HAVi fornecem classes e métodos distintos, mas próprios para esta operação. Entretanto, o resultado esperado não pôde ser observado, devido a limitações do emulador e do *player* de vídeo, pois não foi possível centralizar a imagem ampliada, seja através do reposicionamento, seja pela seleção parcial do vídeo para ampliação.

Na Figura 7, em todas as imagens o ponto X,Y indica a posição inicial do vídeo, W = *Width* (Largura) e H = *Height* (Altura). Na imagem A, o retângulo vermelho simboliza a área visível do vídeo. As coordenadas seriam, por exemplo: X=0, Y=0, W=800, H=600. A imagem B seria o resultado do vídeo redimensionado e posicionado em um ponto X,Y com coordenadas negativas, ou seja, anteriores ao início da área visível da TV. As coordenadas seriam: X=-112, Y=-84, W=1024, H=768 (sendo que a área visível permanece sempre a mesma). Neste caso, era esperado que a parte principal do vídeo permanecesse visível,

mesmo aumentada, fornecendo assim uma melhor visualização por pessoas com miopia, por exemplo, e sem perder detalhes importantes do vídeo.

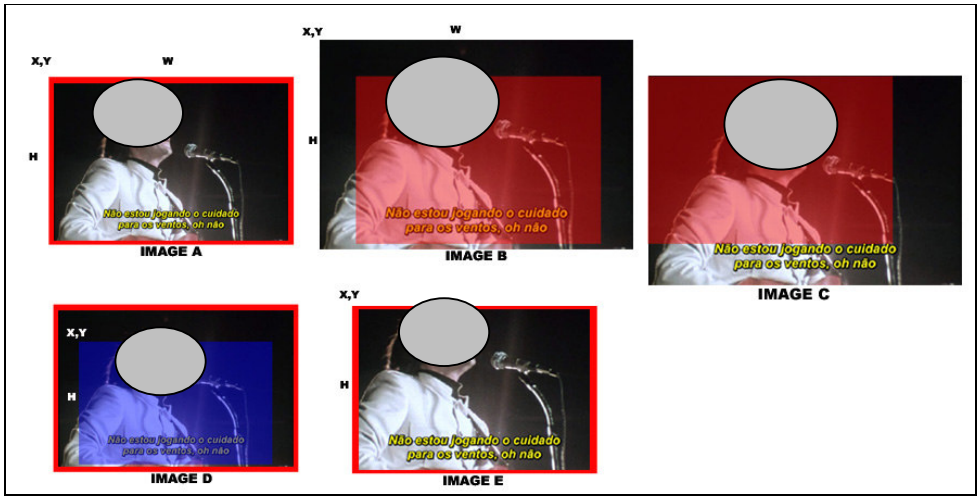

Figura 7: Limitação do ambiente impossibilitou o recurso Lente de Aumento

Mas, ao efetuar o comando de redimensionamento, o método desconsidera os pontos X,Y negativos e posiciona o vídeo na coordenada 0,0. Portanto, as áreas mais à direita e inferiores do vídeo acabam sendo perdidas pelo espectador, como podemos perceber na imagem C.

Foi ensaiada outra abordagem: na imagem D, o retângulo azul simboliza a área que se deseja exibir e ampliar. As coordenadas seriam: X=80, Y=60, W=640, H=480. A imagem E seria o resultado do redimensionamento para as coordenadas: X=0, Y=0, W=800, H=600. Como não há X,Y negativo, se esperava que o retângulo menor do vídeo ocupasse a área total visível, mas após efetuar a chamada do método de redimensionamento não se constatou nenhuma alteração.

Foram experimentados métodos das bibliotecas AWT e HAVi, e em nenhum caso foi obtido sucesso. Espera-se em trabalhos futuros testar a ferramenta em um ambiente real para comprovar se a mesma obterá êxito em sua proposta.

3.2 Implementações

A seguir serão apresentados os principais diagramas que descrevem a estrutura e processos do protótipo.

3.2.1 Diagrama de Casos de Uso

A Figura 8 representa através dos Casos de Uso as funcionalidades do protótipo disponíveis ao telespectador.

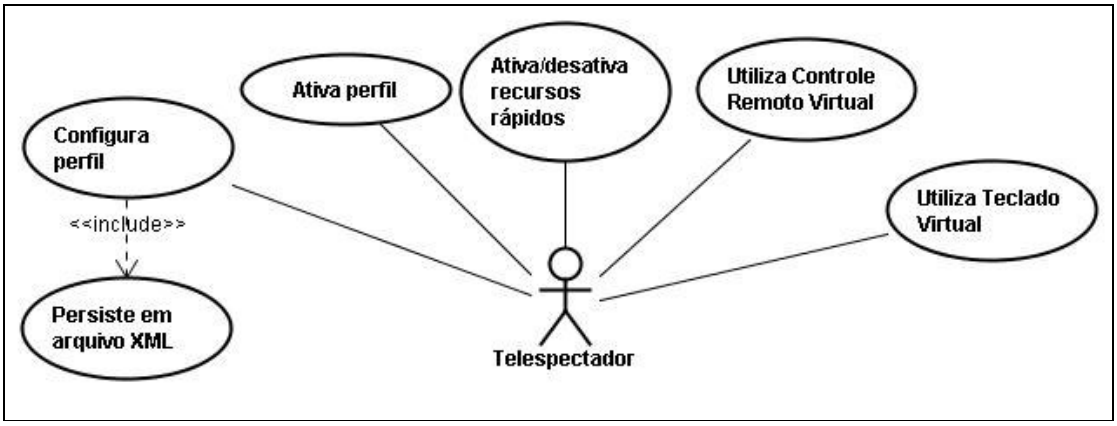

Figura 8: Diagrama de Casos de Uso

a. Configura perfil: esta funcionalidade permite ao telespectador (ou usuário) habilitar as ferramentas de acessibilidade desejadas. Podem ser configurados até três perfis diferentes.

b. Persiste em arquivo XML: ao escolher a opção "Gravar", a configuração do perfil é armazenada em formato XML em um arquivo no *set-top-box*, podendo então ser recuperada a qualquer momento, seja para alteração ou para utilização. O arquivo permanece armazenado mesmo que o aparelho seja desligado.

c. Ativa perfil: esta funcionalidade lê a configuração previamente armazenada e ativa as ferramentas configuradas para o perfil selecionado.

d. Ativa/desativa recursos rápidos: ativa ou desativa a ferramenta de acessibilidade selecionada instantaneamente, sem a necessidade de vínculo a um determinado perfil.

e. Utiliza Controle Remoto Virtual: quando esta ferramenta estiver habilitada qualquer botão pressionado pelo telespectador no controle remoto real exibirá na tela da TV o Controle Remoto Virtual. O usuário utiliza, então, os botões direcionais e o botão "OK" do controle real para selecionar e pressionar os botões do controle virtual.

f. Utiliza Teclado Virtual: esta funcionalidade visa facilitar o preenchimento de campos de formulários. Consiste na exibição de caracteres alfanuméricos, permitindo que o usuário os selecione e insira no campo focado através dos botões direcionais e "OK" do controle remoto real.

3.2.2 Diagrama de Classes

A estrutura de classes e métodos do sistema são apresentadas nas Figuras 9 e 10. A principal classe exibida é a *AccessibilityManager*, pois gerencia a aplicação dos perfis e as ferramentas de ativação rápida. A classe *TheXlet*, cuja declaração é exibida na Listagem 1, implementa a interface *Xlet* da API Java TV e, portanto, gerencia o ciclo de vida da aplicação.

Listagem 1: Declaração da classe TheXlet

```
public class TheXlet extends HContainer
        implements Xlet, Runnable, KeyListener, ResourceClient
```

A classe *B4Player* contém métodos para manipular o *player* de vídeo. A classe *Zoom* controla o recurso de redimensionamento do vídeo para a Lente de Aumento. Já as classes *VRemoteControl* e *VKeyboard*, que respectivamente implementam o Controle Remoto Virtual e o Teclado Virtual, estendem a classe abstrata *VirtualInput*, que define atributos e métodos comuns para estas formas de entrada.

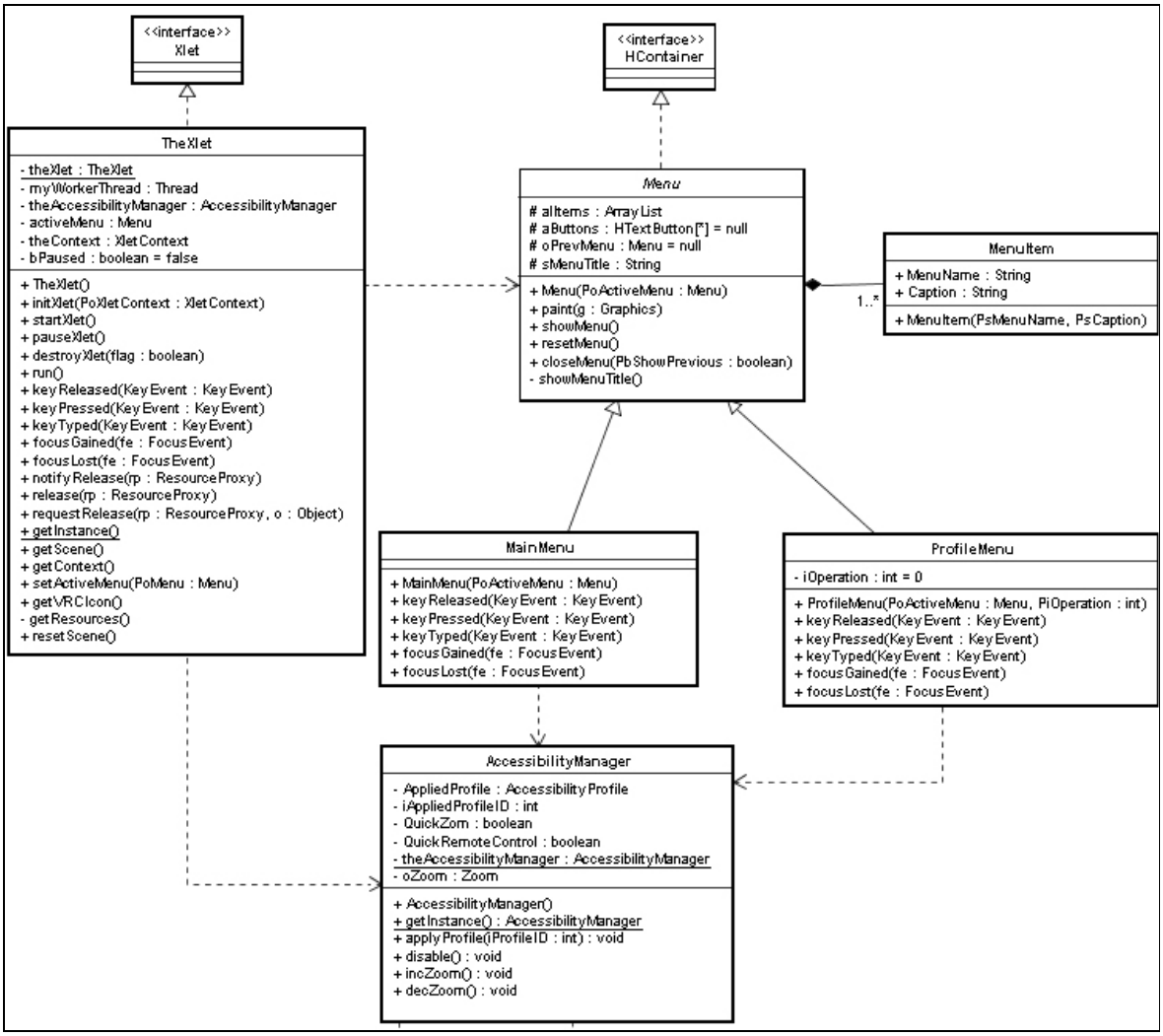

Figura 9: Diagrama de Classes (Parte 1)

Os principais métodos da classe *VKeyboard* estão reproduzidos na Listagem 2: o construtor *VKeyboard(HTextValue PoFocusedControl)* recebe como parâmetro o controle que receberá os caracteres selecionados. O método *keyPressed* é disparado a cada pressionamento de tecla do controle remoto e tem como objetivo determinar qual o caractere associado ao botão selecionado pelo usuário, conforme exibido na Figura 5 do capítulo 3.1, e reproduzí-lo no campo em foco. Os métodos *paint* e *showVInput* estão relacionados à exibição da interface gráfica do Teclado Virtual.

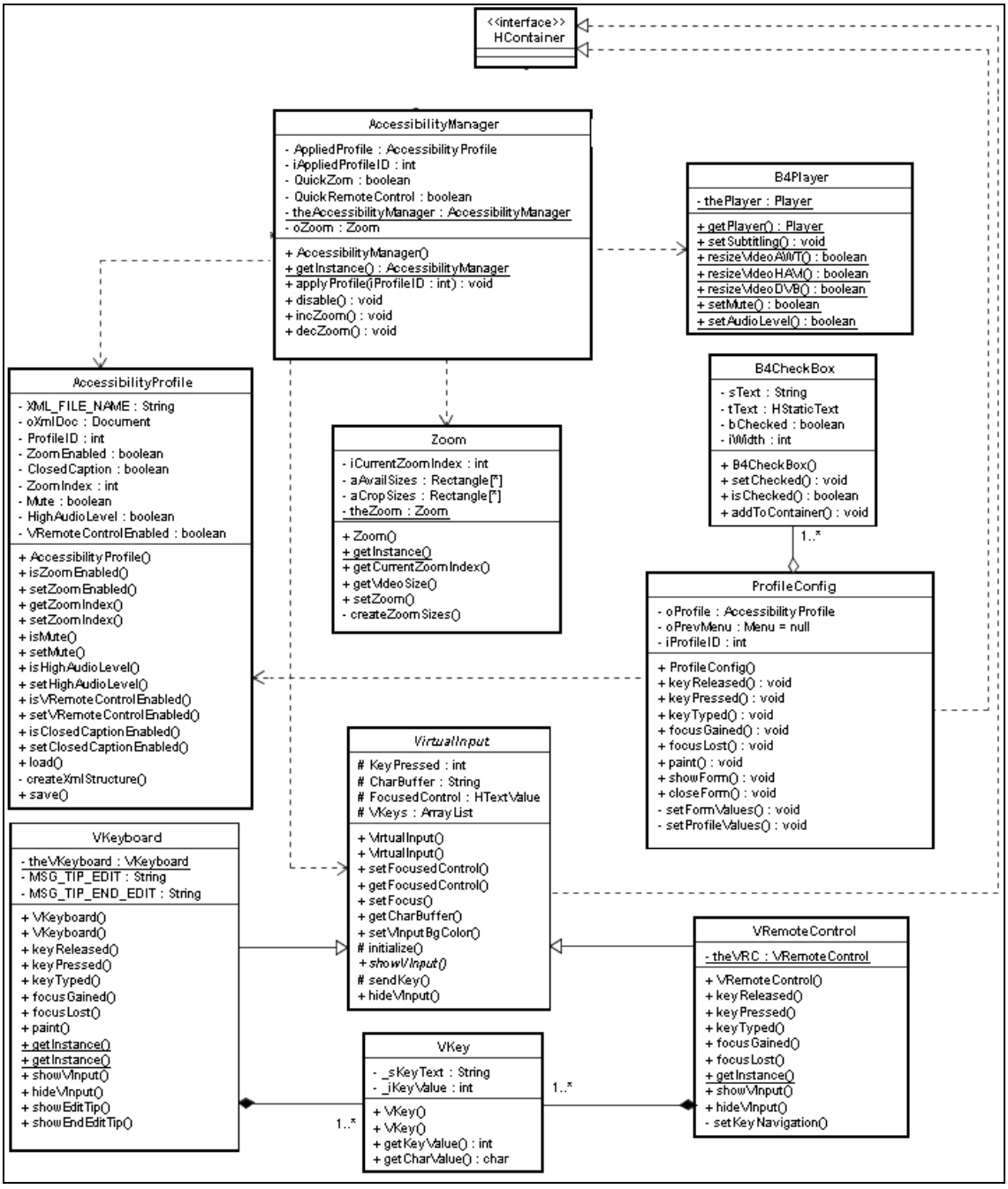

Figura 10: Diagrama de Classes (Parte 2)

Listagem 2: Principais métodos da classe VKeyboard

```
/***  Implementa um Teclado Virtual.*/

public class VKeyboard extends VirtualInput {

  /***  Construtor.  Recebe  o  objeto  que  está  focado  e  receberá  os  caracteres
selecionados.

   *  @param  PoFocusedControl  Objeto  focado  que  deverá  receber  os  caracteres
selecionados.*/

  public VKeyboard(HTextValue PoFocusedControl)  {
```

```
  super(PoFocusedControl);
  theVKeyboard = this;
}

/*** Interface: KeyListener. Verifica a tecla pressionada no controle remoto.*/
public void keyPressed(java.awt.event.KeyEvent KeyEvent)  {
  String sClassName = KeyEvent.getSource().getClass().getName();
  int key = KeyEvent.getKeyCode();
  if (key == HRcEvent.VK_ESCAPE)      // Exit
    hideVInput();
  else if ((key >= 48) &&
           (key <= 57)) {
    ((HSinglelineEntry)FocusedControl).insertChar((char)key);
    ((HSinglelineEntry)FocusedControl).caretNextCharacter();
  }
  else if (key == HRcEvent.VK_ENTER) {
    if (sClassName.contains("HSinglelineEntry")) {
      HSinglelineEntry oInput = (HSinglelineEntry)KeyEvent.getSource();
      theVKeyboard = VKeyboard.getInstance();
      theVKeyboard.setFocusedControl(oInput);
      theVKeyboard.showVInput();
      theVKeyboard.transferFocus();
      theScene.repaint();
    } else {
      VKey oButton = (VKey)KeyEvent.getSource();
      String sName = oButton.getName();
      // Armazena em buffer o caractere
      CharBuffer += oButton.getKeyValue();
      if (FocusedControl != null) {
        if (oButton.getKeyValue() == HRcEvent.VK_LEFT) {
          ((HSinglelineEntry)FocusedControl).caretPreviousCharacter();
        } else if (oButton.getKeyValue() == HRcEvent.VK_RIGHT) {
          ((HSinglelineEntry)FocusedControl).caretNextCharacter();
        } else if (oButton.getKeyValue() == HRcEvent.VK_ENTER) {
          hideVInput();
        } else if (oButton.getKeyValue() == HRcEvent.VK_DELETE) {
          ((HSinglelineEntry)FocusedControl).deleteNextChar();
        } else if (oButton.getKeyValue() == HRcEvent.VK_BACK_SPACE) {
          ((HSinglelineEntry)FocusedControl).deletePreviousChar();
        } else  {
          // Caracteres alfanuméricos e símbolos
          ((HSinglelineEntry)FocusedControl).insertChar(oButton.getCharValue());
          ((HSinglelineEntry)FocusedControl).caretNextCharacter();
        }
```

```
      /*(...)*/
  }

  /*** Cria um background translúcido para o teclado.*/
  public void paint(Graphics g)  {
    g.setColor(VInputBgColor!=null?VInputBgColor:new DVBColor(255, 255, 253, 70));
    g.fillRect(0, 0, getWidth(), getHeight());
    super.paint(g);
  }

  /*** Exibe o teclado, criando as teclas e associando os seus valores.*/
  public void showVInput()  {
    showEndEditTip();
    HSinglelineEntry oInput = (HSinglelineEntry)FocusedControl;
    oInput.setEditMode(true);
    CharBuffer = "";  // Limpa o buffer de caracteres
    // Cria todas as teclas do intervalo de chr 40 a 93
    if (VKeys == null)
      VKeys = new ArrayList();
    int iTop  = 0;
    int iLeft = 20;
    VKey mKeyRows[][] = new VKey[4][18]; // Cria uma matriz de 4 linhas x 18 colunas.
    int i = 40;
    for (int r=0; r<3; r++)       // Linhas
      for (int c=0; c<18; c++)    // Colunas
        if (i<94)                 // KeyCodes 40 -> 93
        {
          iTop  = r * 42;         // 42 = espaço total entre um botão e outro
          iLeft = (c * 42) + 20;
          mKeyRows[r][c] = new VKey(Character.toString((char)i), i, this, iLeft, iTop,
40); // Adiciona à matriz para controle de foco
          this.VKeys.add(mKeyRows[r][c]);   // Adiciona à lista do teclado
          i++;
        }  else  break; // Aborta o loop
    // Cria teclas especiais na última linha
    iLeft = 20; iTop  += 42;
    mKeyRows[3][0] = new VKey("     ", HRcEvent.VK_SPACE, this, iLeft, iTop, 82);
    this.VKeys.add(mKeyRows[3][0]);
    iLeft += 84;
    mKeyRows[3][1] = new VKey("<=", HRcEvent.VK_LEFT, this, iLeft, iTop, 82);
    this.VKeys.add(mKeyRows[3][1]);
    iLeft += 84;
    mKeyRows[3][2] = new VKey("=>", HRcEvent.VK_RIGHT, this, iLeft, iTop, 82);
    this.VKeys.add(mKeyRows[3][2]);
```

```
iLeft += 84;
mKeyRows[3][3] = new VKey("ENTER", HRcEvent.VK_ENTER, this, iLeft, iTop, 124);
this.VKeys.add(mKeyRows[3][3]);
iLeft += 126;
mKeyRows[3][4] = new VKey("BACK", HRcEvent.VK_BACK_SPACE, this, iLeft, iTop, 124);
this.VKeys.add(mKeyRows[3][4]);
iLeft += 126;
mKeyRows[3][5] = new VKey("DEL", HRcEvent.VK_DELETE, this, iLeft, iTop, 82);
this.VKeys.add(mKeyRows[3][5]);
iLeft += 84;
theScene.add(this);
this.setVisible(true);
// Controle de foco
this.setFocus();
for (int r=0; r<4; r++)       // Linhas
  for (int c=0; c<18; c++)    // Colunas
    if (mKeyRows[r][c] != null)    {
      // Calcula e efetua ajustes nos botões de navegação
      int iRup   = r==0?3:r-1;
      int iCup   = r==0?0:c;
      int iRdown  = r==3?0:r+1;
      int iCdown  = (r==2 && c>5)?0:c;
      int iRleft  = r;
      int iCleft  = c==0?(r==3?5:17):c-1;
      int iRright = r;
      int iCright = c==17?0:(r==3 && c==5?0:c+1);
      // Determina os controles de acordo com as coordenadas da matriz
      VKey kUp    = mKeyRows[iRup][iCup];
      VKey kDown  = mKeyRows[iRdown][iCdown];
      VKey kLeft  = mKeyRows[iRleft][iCleft];
      VKey kRight = mKeyRows[iRright][iCright];
      // Atribui a navegação
      mKeyRows[r][c].setFocusTraversal( kUp==null?null:kUp,
                                        kDown==null?null:kDown,
                                        kLeft==null?null:kLeft,
                                        kRight==null?null:kRight);
      mKeyRows[r][c].setMove(HRcEvent.VK_RIGHT, kRight);
        System.out.println("setFocusTraversal: " + r + "x" + c + " " +
                iRup + "x" +
                iCup + kUp.getTextContent(HState.NORMAL_STATE) + " " +
                iRdown + "x" +
                iCdown + kDown.getTextContent(HState.NORMAL_STATE) + " " +
                iRleft + "x" +
                iCleft + kLeft.getTextContent(HState.NORMAL_STATE) + " " +
```

```
                    iRright + "x" +
                    iCright + kRight.getTextContent(HState.NORMAL_STATE));
          }
        } // for
  }
```

Outra classe importante é a *AccessibilityProfile*, responsável pela persistência dos perfis em um arquivo XML. As classes *MainMenu* e *ProfileMenu* estendem a classe abstrata *Menu* e exibem os menus do sistema. Pode-se acompanhar na Figura 11 o Diagrama de Seqüencia para exibição do menu principal, assim como nas listagens 3 e 4 o detalhamento dos métodos ilustrados no diagrama.

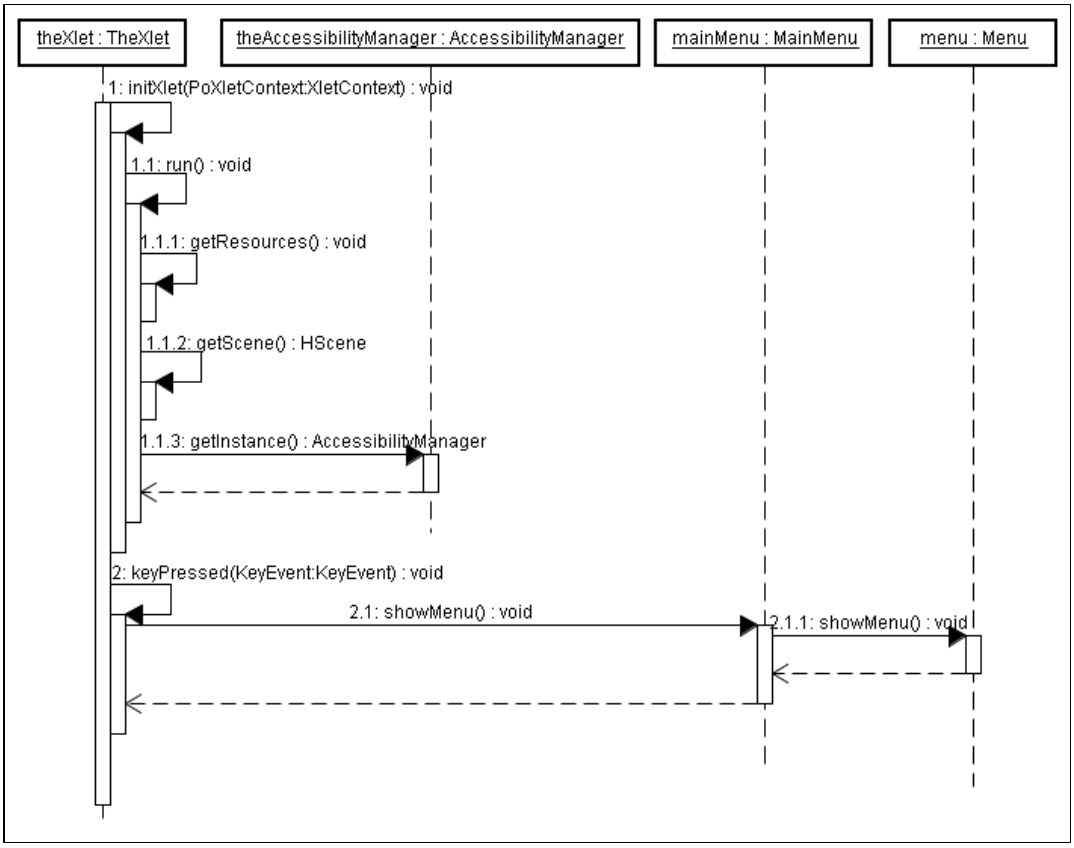

Figura 11: Diagrama de Seqüência - Exibição do menu principal MainMenu.showMenu()

Listagem 3: Classe TheXlet

```
  /*** Carrega os recursos da aplicação. */
  private void getResources()  {
    HSceneFactory oFactory = HSceneFactory.getInstance();
    // Cria um HSceneTemplate em tela cheia
    HSceneTemplate hst = new HSceneTemplate();
    hst.setPreference(
            HSceneTemplate.SCENE_SCREEN_DIMENSION,
```

```
            new org.havi.ui.HScreenDimension(1,1),        // Tamanho
            HSceneTemplate.REQUIRED);
    hst.setPreference(
            HSceneTemplate.SCENE_SCREEN_LOCATION,
            new org.havi.ui.HScreenPoint(0,0),            // Posição
            HSceneTemplate.REQUIRED);
    theScene = oFactory.getBestScene(hst);
    // Carrega imagens
    MediaTracker tracker = new MediaTracker(this);
    // Carrega imagem do icone de interatividade
    imgITV = Toolkit.getDefaultToolkit().getImage("itv.gif");
    // Carrega a imagem de fundo do Controle Remoto Virtual
    imgVRC = Toolkit.getDefaultToolkit().getImage("remote-control.gif");
    icoITV = new HStaticIcon(imgITV, theScene.getWidth() - 100, 20, 48, 48);
    icoVRC = new HStaticIcon(imgVRC, 0, 0, 162, 500);
    tracker.waitForAll();
    // Define o tamanho do Xlet para ser igual ao tamanho da cena
    Rectangle rect = theScene.getBounds();
    setBounds(rect);
    theScene.addKeyListener(this); // O próprio Xlet é o listener
  }

  /*** Remove todos os elementos da cena e exibe o icone de interatividade.*/
  public void resetScene()  {
    theScene.setVisible(false); // Oculta
    theScene.removeAll();       // Remove todos os objetos da cena
    activeMenu = null;
    // Adiciona o icone de interatividade (criado previamente, e nunca destruído)
    theScene.add(icoITV);
    theScene.setVisible(true);  // Exibe a cena
    theScene.requestFocus();    // Foca a cena
  }

  /*** Interface: KeyListener. Verifica a tecla pressionada no controle remoto.
   * Exibe o menu principal se alguma tecla de interatividade ou Enter/OK for
pressionada.*/
  public void keyPressed(java.awt.event.KeyEvent KeyEvent)  {
    int key = KeyEvent.getKeyCode();
    if ((key == HRcEvent.VK_ENTER) ||
        ((key >= HRcEvent.VK_COLORED_KEY_0) &&
         (key <= HRcEvent.VK_COLORED_KEY_4)))    {
      MainMenu mainMenu = new MainMenu(null);
      mainMenu.showMenu();
    }
```

```
    else if (key == HRcEvent.VK_ESCAPE)
      resetScene();     // Limpa a cena
  }
```

Listagem 4: Classe Menu

```
  /***  Exibe os botões do menu, baseado no conteúdo do ArrayList alItems.*/
  public void showMenu()  {
    theScene.setVisible(false);
    theXlet.setActiveMenu(this);          // Define este menu como ativo
    int i = 0;
    showMenuTitle();
    int x = 50; // Posição do primeiro botão
    // Cria o vetor de botões do tamanho dos itens previamente especificados
    aButtons = new HTextButton[alItems.size()];
    Iterator oIt = alItems.iterator();
    MenuItem oItem = (MenuItem)oIt.next();
    while (oItem != null)    {
      aButtons[i] = new HTextButton(oItem.Caption, 10, x, 400, 35,
              new Font("Tiresias", 1, 36),
              Color.white,
              oItem.MenuName  ==  "itmVoltar"?Color.red:Color.blue,  //  Botão  voltar
sempre vermelho
              new HDefaultTextLayoutManager()); // Cria o botão
      aButtons[i].addKeyListener(this);
      aButtons[i].addFocusListener(this);
      aButtons[i].setName(oItem.MenuName);      // Importante: nome do botão!
      this.add(aButtons[i]);
      if (oIt.hasNext())
        oItem = (MenuItem)oIt.next();
      else
        oItem = null;
      x +=40; // Espaço para o início do próximo botão
      i++;
    }
    theScene.add(this);           // Adiciona o menu à cena
    this.setVisible(true);
    this.repaint();
    theScene.setVisible(true);
    // Atribui os controles de foco (semelhante a TabOrder)
    if (aButtons[0] != null)      {
      aButtons[0].requestFocus();
      for (i=0; i<aButtons.length; i++)      {
        int iPrev = i==0?aButtons.length-1:i-1;
        int iNext = i==aButtons.length-1?0:i+1;
```

```
        aButtons[i].setFocusTraversal(aButtons[iPrev], aButtons[iNext], null, null);
      }
    }
  }
```

Por sua vez, a classe *ProfileConfig* implementa um formulário para configuração dos Perfis de Acessibilidade utilizando a classe *B4CheckBox*. Por fim, as classes *MenuItem* e *VKey* contêm os atributos necessários para os itens de menu e as teclas virtuais, respectivamente.

A listagem 5 provê um exemplo de utilização prática do Teclado Virtual, onde é criado um campo de entrada de texto chamado *oInput01* e associado ao seu evento *keyListener* uma instância da classe *VKeyboard*. Essa associação é responsável por tratar a seleção dos caracteres e por sua vez a exibição no campo de texto.

Listagem 5: Exemplo de utilização do Teclado Virtual em um formulário

```
  /*** Cria os objetos necessários e limpa a cena. */
  public void showForm()  {
    this.theXlet = TheXlet.getInstance();
    this.theScene = theXlet.getScene();
    theScene.setVisible(false);
    theScene.removeAll(); // Garante que não ficou nenhum "residuo" na cena
    Zoom.getInstance().setZoom(1); // Zoom 50%
    /* (...) */
    // Prepara o teclado virtual
    oVKeyboard = VKeyboard.getInstance();
    oVKeyboard.setVInputBgColor(Color.orange);
    oVKeyboard.showEditTip();
    // Cria os controles
    HStaticText lbl01 = new HStaticText("Nome:",
                              420, 70, 300, 25,
                              new Font("Tiresias", 1, 20),
                              Color.black, Color.orange,
                              new HDefaultTextLayoutManager());
    lbl01.setHorizontalAlignment(HVisible.HALIGN_LEFT);
    oInput01 = new HSinglelineEntry("",
                            420, 100, 300, 25,
                            20,
                            new Font("Tiresias", 1, 20),
                            Color.black);
    oInput01.setBackgroundMode(HVisible.BACKGROUND_FILL);
    oInput01.setBackground(Color.white);
    oInput01.setHorizontalAlignment(HVisible.HALIGN_LEFT);
    oInput01.setBordersEnabled(true);
    oInput01.addKeyListener(oVKeyboard); // Importante: é o que determina se o teclado
virtual será exibido
```

```
    oInput01.addKeyListener(this);
    oInput01.addFocusListener(this);
    /* (...) */
    // Adiciona ao container
    this.add(lbl01);
    this.add(oInput01);
    theScene.add(this);           // Adiciona o form à cena
    this.setVisible(true);
    theScene.setVisible(true);
    oInput01.requestFocus();      // Dispara o evento de foco e exibe o teclado virtual
por consequencia
    /* (...) */
  }
```

4 Conclusões

Podemos concluir que os objetivos deste trabalho, que eram difundir a utilização da linguagem Java™ no ambiente da TV digital para a comunidade de desenvolvedores e disponibilizar uma base de ferramentas para maior inclusão social de portadores de necessidades especiais, foram alcançados. Os principais resultados foram protótipos de ferramentas de acessibilidade configuráveis através de perfis, que podem ser utilizadas para beneficiar uma parcela da população que estaria excluída da TV digital sem estes auxílios ou de televisores e controles remotos especiais.

As experiências realizadas durante este trabalho relataram que o emulador *XletView* fornece funcionalidades básicas para o teste de *Xlets*, mas para um completo suporte aos recursos requeridos pelas ferramentas propostas seria necessário um *set-top-box* ou um emulador com suporte a esses recursos. Foi utilizado este emulador por ser o mais popular e recomendado pelos desenvolvedores de *Xlets*, além de ser *open source* e gratuito. Como não foram realizados testes em ambientes reais até o momento de produção deste trabalho, não se sabe se as limitações relatadas aqui se aplicam a um *set-top-box* preparado para o SBTVD. Espera-se em trabalhos futuros obter a resolução dos problemas encontrados através da implementação e teste das ferramentas em ambientes reais, para que em decorrência seja possível apresentá-las a portadores de necessidades especiais, coletar os resultados dos testes e aperfeiçoá-las, visando a sua efetiva adoção pelos desenvolvedores de aplicações.

A comunidade interessada no desenvolvimento de aplicações para TV digital aguarda o lançamento do Ginga-J e ferramentas para testar as aplicações desenvolvidas, possibilitando assim o desenvolvimento de aplicações utilizando a linguagem Java™, amplamente utilizada no mercado. Há muita expectativa sobre o resultado do trabalho resultante do memorando assinado entre o Fórum SBTVD e a SUN, que permitirá o desenvolvimento e distribuição do *middleware* e aplicações sem a necessidade de pagamento de licenças.

Como extensões deste trabalho poderiam ser desenvolvidas ferramentas de reconhecimento de comandos de voz e leitura de menus e demais textos exibidos na tela, com o objetivo de possibilitar aos portadores de necessidades especiais o acesso aos recursos da TV digital. Também pode ser implementado um algoritmo de predição de palavras baseado em um dicionário para o Teclado Virtual.

Foi constatado durante o período de produção deste trabalho que há um grande interesse da população em geral nos recursos disponibilizados pela TV digital, mas o objetivo da inclusão social proposto inicialmente no projeto do SBTVD está distante de ser alcançado. Isto se dá em parte devido ao alto custo dos aparelhos e a falta de interatividade, pois os receptores disponibilizados até o momento são terminais *Zapper* (TONIETO, 2006), e, portanto, não possuem o *middleware* Ginga. Acredita-se que iniciativas futuras viabilizem o projeto, como a adoção de bibliotecas de desenvolvimento livres de licenças, incentivos fiscais aos fabricantes e a disponibilização de atrativos como interatividade e mobilidade, o que resultará na aceitação pela população e produção em larga escala dos aparelhos receptores.

5 Referências Bibliográficas

BRASIL. Lei nº. 5296, de 2 de dezembro de 2004. Regula prioridade de atendimento e acessibilidade. In: SENADO FEDERAL. **Legislação Republicana Brasileira**. Brasília, 2004. Disponível em: <http://www6.senado.gov.br/legislacao/ListaTextoIntegral.action?id=227619>. Acesso em: out. 2007.

FERNANDES, Jorge; LEMOS, Guido; SILVEIRA, Gledson. **Introdução à televisão digital interativa: arquitetura, protocolos, padrões e práticas**. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 24., 2004, Salvador. Disponível em: <http://www.cic.unb.br/docentes/jhcf/MyBooks/itvdi/texto/itvdi.pdf>. Acesso em: mai. 2006.

FREED, Ken. **Desenvolvimento de aplicações para TV Digital Interativa**. Disponível em: <http://www.media-visions.com/itv-newbies.html>. Acesso em: ago. 2006.

GINGA. **Portal do Software Público – Ginga**. Disponível em <http://www.softwarepublico.gov.br/dotlrn/clubs/ginga/>. Acesso em: nov. 2007.

MC, Ministério das Comunicações. **Sistema Brasileiro de TV Digital**. Disponível em: <http://sbtvd.cpqd.com.br>. Acesso em: ago. 2007.

MONTEZ, Carlos; BECKER, Valdecir. **TV Digital Interativa: Conceitos, Desafios e Perspectivas para o Brasil**. Florianópolis: Ed. UFSC, 2006, 2ª ed. 200 p.

MORENO, Marcio F. **Um Middleware Declarativo para Sistemas de TV Digital Interativa**. Rio de Janeiro, 2006. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

MORRIS, Steven; SMITH-CHAIGNEAU, Anthony**. Interactive TV Standards**. Burlington: Ed. Focal Press, 2005. 585 p.

NBR15290. **Acessibilidade em comunicação na televisão**. Disponível em: <http://www.mj.gov.br/sedh/ct/corde/dpdh/corde/ABNT/NBR15290.pdf>. Acesso em: out. 2007.

NIC. **Comitê Gestor da Internet – Núcleo de Informação e Coordenação**. Disponível em <http://www.nic.br>. Acesso em nov. 2007.

SCHWALB, Edward M.. **iTV Handbook: Technologies and Standards**. New Jersey:

Ed. Prentice Hall, 2003. 752 p.

SET. **Sociedade Brasileira de Engenharia de Televisão e Telecomunicações**. Disponível em: <http://www.set.com.br/tecnologia.htm>. Acesso em: ago. 2007.

SUN MICROSISTEMS. **Especificação da API Java TV**. Disponível em: <http://java.sun.com/products/javatv>. Acesso em: ago. 2007.

TONIETO, Márcia. **Sistema Brasileiro de TV Digital - SBTVD - Uma análise política e tecnológica na inclusão social. Fortaleza, 2006**. Dissertação de Mestrado – Centro de Ciências e Tecnologia, Universidade Estadual do Ceará.