

Sistema de Ensino-Aprendizagem da Língua de Sinais para TV Digital

por
Bianca Martins

Trabalho Individual II
TI-2007/1

Orientador: **Prof. Dr. Paulo Roberto Gomes Luzzardi**
Co-Orientadora: **Prof. Dr. Tatiana Aires Tavares**

Pelotas, março de 2008

AGRADECIMENTOS

Agradeço a Deus, por me dar força, dia após dia ao longo desta caminhada.

A minha família, em especial aos meus pais, pela força e colaboração nesta jornada.

Ao meu esposo Gilberto, pela compreensão e carinho nos momentos de stress.

Aos meus amigos e colegas do Senac, pelo apoio na realização deste trabalho, em especial ao meu diretor Eduardo Cassal, as pedagogas Maristela Kellermann e Giovana da Silva Simões.

A todos os meus colegas de mestrado pelo apoio nos momentos difíceis.

Aos Professores Dr. Paulo Roberto Gomes Luzzardi e Dr. Tatiana Aires Tavares e ao bolsista Cauane Blumenberg Silva pela amizade, compreensão, competência e empenho na orientação desta dissertação e pela confiança depositada em mim.

E por fim, a todas as pessoas de forma direta ou indireta, que contribuíram para a realização deste trabalho.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS	5
Resumo	5
Abstract	6
Introdução	7
Um Breve Histórico	10
Ginga-NCL	13
GINGA-J	16
API Java TV	17
Considerações Finais	19
Referências Bibliográficas	21

LISTA DE ABREVIATURAS

SBTVD	Sistema Brasileiro de TV Digital
NCL	Nested Context Language)
SMIL	Synchronized Multimedia Integration Language
XHTML	eXtensible Hypertext Markup Language
ISO-MHEG	<i>Multimedia and Hypermedia Experts Group</i>
STB	<i>Set-top-box</i>
ACAP-X	<i>Advanced Common Application Platform</i>
DVB	<i>Digital Video Broadcasting</i>
ATSC	<i>Advanced Television System Comitee</i>
ISDB	<i>Integrated Services Digital Broadcasting</i>
SDTV	<i>Standard Definition Television</i>
HDTV	<i>High- Definition Television</i>
API	<i>Application Programming Interface</i>
VM	<i>Virtual Machine</i>
RTOS	<i>Real Time Operating System</i>
HAVI	<i>Home Audio-Video interoperability</i>
DAVIC	<i>Digital Audio-Video Council</i>
INES	<i>Instituto Nacional de Educação de Surdos</i>
ASL	<i>Língua Americana de Sinais</i>
FNDE	<i>Fundo Nacional de Desenvolvimento da Educação</i>
TV	<i>Televisão</i>
TVDI	<i>TV Digital e Interativa</i>
DAVIC	<i>Digital Audio Visual Council</i>
APIs	<i>Application Programming Interface)</i>
MHP	<i>Multimedia Home Platform</i>

Resumo

O Brasil já está se preparando para a implantação definitiva da TV digital. O potencial de disseminação desta nova tecnologia e a disponibilidade de conteúdos multimídia disponíveis oferecem uma abordagem promissora para o ensino-aprendizagem. Os ambientes educacionais, utilizados para gerenciar conteúdos e para hospedar serviços colaborativos na *Web*, estão sendo adaptados para oferecerem suporte ao ensino-aprendizagem na TV digital.

Este trabalho visa a investigação dos elementos necessários para o desenvolvimento de um sistema Ensino-Aprendizagem da Língua de Sinais para TV Digital, através da construção de um protótipo que utilize as tecnologias disponíveis até o momento: **Java TV** ou **GINGA NCL**.

A seguir será apresentado um estudo preliminar destas tecnologias destinadas ao desenvolvimento de aplicações interativas, bem como uma análise dos elementos necessários para o bom uso da TV Digital e a importância dessa nova mídia na educação.

Para validar estes conceitos, futuramente será apresentada uma aplicação e toda a infra-estrutura necessária para o desenvolvimento de interfaces voltadas a TV Digital, como linguagens, os programas que serão utilizados para o desenvolvimento e o ambiente de simulação de TV Digital para fins de emulação e execução de um **Sistema de Ensino-Aprendizagem da Língua de Sinais para TV Digital**.

Abstract

Brazil is already preparing for the final deployment of digital TV. The potential for spread of this new technology and the availability of multimedia content available offer a promising approach to the teaching-learning. The educational environments, used to manage content and to host collaborative services on the Web, are being adapted to offer support to the teaching-learning in digital TV.

This work aims to research the elements necessary for the development of a system Teaching-Learning the language of signs for Digital TV, through the construction of a prototype using the technology available so far: **Java TV** or **GINGA NCL**.

The following will be presented a preliminary study of these technologies for the development of interactive applications, as well as an analysis of the elements necessary for the proper use of Digital TV and the importance of this new media in education.

To validate these concepts in future will be presented to an application, and all the infrastructure necessary for the development of interfaces targeted to Digital TV, like languages, programmes that will be used for the development and simulation environment for digital TV for emulation and implementation of a **System Tutor Interactive Education**.

Introdução

Middleware é a camada de *software* localizada entre as aplicações e o sistema operacional. Tem por objetivo oferecer às aplicações o suporte necessário para seu rápido e fácil desenvolvimento, além de esconder os detalhes das camadas inferiores, bem como a heterogeneidade entre os diferentes sistemas operacionais e o *hardware*. Pode-se dizer que, do ponto de vista do *software*, ao definir-se o *middleware*, está de fato, se definindo a “Televisão Digital Brasileira”.

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”.

Um conteúdo declarativo é baseado (especificado) em uma linguagem declarativa, isto é, em uma linguagem que enfatiza a descrição declarativa do problema, ao invés da sua decomposição em uma implementação algorítmica. Enquanto que um conteúdo procedural é baseado em uma linguagem não declarativa. Linguagens não declarativas podem seguir diferentes paradigmas. Tem-se assim, as linguagens baseadas em módulos, orientadas a objetos etc. A literatura sobre TV Digital, no entanto, cunhou o termo procedural para representar todas as linguagens que não são declarativas. Numa programação procedural, o computador deve ser informado sobre cada passo a ser executado. Pode-se afirmar que, em linguagens procedurais, o programador possui um maior poder sobre o código, sendo capaz de estabelecer todo o fluxo de controle e execução de seu programa. Entretanto, para isso, ele deve ser bem qualificado e conhecer bem os recursos de implementação. A linguagem mais usual encontrada nos ambientes procedurais de um sistema de TV Digital é Java [26].

Linguagens declarativas são linguagens de mais alto nível de abstração, usualmente ligadas a um domínio ou objetivo específico. Nas linguagens declarativas, o programador fornece apenas o conjunto das tarefas a serem realizadas, não estando preocupado com os detalhes de como o executor da linguagem (interpretador, compilador ou a própria máquina real ou virtual de execução) realmente implementará essas tarefas. Linguagens declarativas resultam em uma declaração do resultado desejado, e, portanto, normalmente não necessitam de tantas linhas de código para definir uma certa tarefa. Entre as linguagens declarativas mais comuns estão a NCL (*Nested Context Language*) [24], SMIL (*Synchronized Multimedia Integration Language*) [27] e XHTML (*eXtensible Hypertext Markup Language*) [28].

Aplicações para TV Digital usualmente lidam com objetos (a partir de agora chamados de objetos de mídia) que são gerados individualmente, baseados em ferramentas de terceiros, mais apropriadas à edição de cada mídia específica.

Grande parte de uma aplicação multimídia (interativa ou não) para TV Digital é baseada na sincronização espacial e temporal entre os seus diversos objetos de mídia e, possivelmente, na escolha entre alternativas de objetos para apresentação.

Uma linguagem de “união” entre objetos que permita a definição de seus sincronismos e suas adaptabilidades se torna assim a solução ideal, para a geração desse conteúdo multimídia, ou aplicação, tipicamente declarativo.

Mesmo quando tais aplicações são suportadas por um tipo procedural (Java), o que se nota é que as ferramentas de autoria tentam esconder do programador toda parte procedural, oferecendo uma interface declarativa ao gerador de aplicações (*JAME Author* [11]; *Cardinal Studio* [5]; *AltComposer* [1] etc.). A dificuldade e limitação dessas ferramentas são, porém, evidentes, por não terem como base uma linguagem declarativa eficiente, capaz de facilitar a organização de programas e sistemas e preocupar-se com o **que fazer**, e não com o **como fazer**.

A princípio, poder-se-ia pensar que o uso de uma linguagem declarativa é sempre mais vantajoso do que o uso de linguagens não declarativas, entretanto, como já mencionado, as linguagens declarativas têm de ser definidas com um foco específico. Quando o foco da aplicação não “casa” com o da linguagem, o uso de uma linguagem procedural não é apenas vantajoso, mas se faz necessário pois ajuda na organização de programas e sistemas.

Uma aplicação não precisa ser puramente declarativa ou puramente procedural. Uma aplicação declarativa pura é aquela em que todas as suas entidades, e não apenas a “inicial”, é do tipo conteúdo declarativo (especificado segundo uma linguagem declarativa). Analogamente, uma aplicação procedural pura é aquela em que todas as suas entidades, e não apenas a “inicial”, é do tipo conteúdo procedural.

Uma aplicação híbrida (procedural ou declarativa) é aquela cujo conjunto de entidades contém tanto conteúdo do tipo declarativo quanto procedural.

Frequentemente, aplicações declarativas fazem uso de conteúdos em forma de *scripts*, que são de natureza procedural. Mais ainda, uma aplicação declarativa pode referenciar um código procedural embutido (no caso usual de sistemas de TV Digital).

Por ser uma linguagem onde o programador especialista é capaz de estabelecer todo o fluxo de controle e execução de seu programa, uma linguagem procedural pode especificar de forma procedural (algorítmica) qualquer conteúdo declarativo. A recíproca não é verdadeira, visto que as linguagens declarativas não têm o foco geral, mas ao contrário, usualmente são projetadas para facilitar o desenvolvimento de aplicações com um foco específico. Da mesma forma, aplicações procedurais podem referenciar conteúdos declarativos, ou até construir e iniciar a apresentação de um conteúdo declarativo.

Assim, sem medo de errar pode-se afirmar que, nos sistemas de TV Digital, os dois tipos de aplicação irão coexistir, sendo então conveniente que o dispositivo receptor

integre o suporte aos dois tipos em seu *middleware*. Esse é o caso do *middleware* Ginga do Sistema Brasileiro de TV Digital Terrestre (ISDTV-T).

Após esta rápida introdução aos paradigmas de estilos de programação utilizados nos diversos *middleware* para sistemas de TV Digital, este trabalho tem como objetivo apresentar um breve histórico do desenvolvimento dos *middleware*, bem como o *middleware* padrão do sistema brasileiro, o Ginga. Além de aprofundar um pouco mais a discussão sobre o ambiente declarativo do Ginga, o NCL (*Nested Context Language*), e por fim as considerações finais.

Um Breve Histórico

Um dos primeiros padrões abertos usados nos sistemas de TV Digital (DTV) foi definido pelo ISO-MHEG (*Multimedia and Hypermedia Experts Group*) em 1997 [15], conhecido como MHEG-1, que usava a notação sintática ASN-1 para a definição de aplicações multimídia baseadas no relacionamento entre objetos. Em 1991, o modelo de contextos aninhados NCM (*Nested Context Model*) [23], modelo conceitual de dados base da linguagem NCL, propôs uma solução para o problema então em aberto, sobre aninhamento de composições [12], de onde derivou seu nome, que foi em seguida adotada pelo MHEG como sua estrutura de composições, na reunião do grupo de trabalho em 1992.

Desde o início, tanto NCM quanto o MHEG apresentavam uma linguagem declarativa que incluía o suporte a objetos procedurais, estendendo seus modelos declarativos básicos. NCM, no entanto, tinha como foco apenas apresentações na *Web* e não no ambiente de TV Digital.

O avanço da especificação MHEG se deu junto ao sucesso da portabilidade da linguagem Java e, assim, em 1998 MHEG incorporava o uso de Java na definição de seus objetos *scripts*, aliando sua força declarativa ao poder computacional de Java.

Embora essa versão MHEG-6 nunca tenha sido implantada, ela formou a base para o padrão de TV interativa DAVIC (*Digital Audio Visual Council*), que teve várias de suas APIs (*Application Programming Interface*) adotadas pelo MHP (*Multimedia Home Platform*) [10].

MHP foi o primeiro padrão de *middleware* puramente em Java, evoluindo posteriormente para a harmonização GEM (*Globally Executable MHP*) [8]. Esta mudança do paradigma declarativo em direção a Java, principalmente pela portabilidade do Java, não significou, entretanto, o abandono do paradigma declarativo. Pouco a pouco *middleware* baseados em Java reincorporaram o ambiente declarativo. No MHP, ele foi incluído pelo uso do HTML e *plug-ins* para outros formatos. A demora na definição de um perfil padrão HTML, entretanto, levou a várias implementações diferentes, fazendo com que a maioria das aplicações que usavam o HTML, no presente momento, o façam através de *browsers* HTML que devem ser obtidos por *download*, de forma a garantir a consistência da apresentação em diferentes receptores.

A tentativa de padronização de um ambiente declarativo pelo MHP, que abrangesse os diversos legados HTML, levou à complexidade excessiva do DVBHTML [9]. Assim, embora padrão na versão MHP 1.1, o uso do DVB-HTML é questionado por inúmeras implementações, que continuam a trabalhar com *browsers* HTML obtidos por *download*.

O uso da linguagem HTML como linguagem declarativa é bastante questionável, pelo fato de ter seu foco exclusivamente na interatividade, relegando o tratamento do sincronismo temporal, em sua forma mais geral, a scripts procedurais, usualmente escritos em ECMAScript [7]. Tal é o caso do DVB-HTML, do padrão DVB (*Digital Video Broadcasting*), e do ACAP-X (*Advanced Common Application Platform - XHTML*) [4], do padrão ATSC (*Advanced Television System Committee*).

Reconhecendo a importância do ambiente declarativo e do suporte que deve ser dado a aplicações com foco no sincronismo espacial e temporal de seus objetos de mídia,

discussões sobre o *middleware* do padrão japonês ISDB levaram em conta a existência da linguagem SMIL, padrão W3C para sincronismo de mídia na *Web*.

Entretanto, abandonaram a idéia, pela simples razão que assim reportaram: “SMIL é um esquema de representação bastante estático. A linguagem está pronta para temporizações pré-programadas, mas não em tempo real. Ela é inconveniente para programas ao “vivo”. Na verdade, apenas recentemente, em 2006, um perfil SMIL, específico para TV Digital, começou a ser estudado, em uma possível junção com a linguagem NCL. Assim, tal qual nos sistemas DVB (*Digital Video Broadcasting*) e ATSC (*Advanced Television System Committee*), o sistema ISDB adota um perfil XHTML como base de sua linguagem declarativa, chamada BML (*Better Markup Language*) [3].

Em BML, assim como nos outros padrões, o sincronismo de mídia e a adaptabilidade são conseguidos através de entidades procedurais, escritas por meio da linguagem ECMAScript. Sincronismo em programas gerados ao vivo é obtido através de funções ECMAScript chamadas por eventos de fluxo (*stream events*) DSM-CC (*Digital Storage Media - Command and Control*) [14], denominados *b-events* em BML.

O padrão ISDB também previu o uso do GEM (*Globally Executable MHP*) como seu ambiente procedural, mas esse nunca foi implementado e nem parece ter perspectiva de ser a médio prazo. Assim, BML, através de seus objetos ECMAScript, assume todas as funções procedurais necessárias em um *middleware*, fazendo com que uma possível futura versão com o GEM integrado seja, provavelmente, ineficiente, devido as redundâncias de funções que terá de oferecer.

Por ser mais recente, o sistema brasileiro de TV Digital teve por obrigação procurar as alternativas tecnológicas mais recentes e entre elas estava a concepção de um *middleware* onde a convivência dos ambientes declarativos e procedurais fosse a mais eficiente possível, em termos de custo e desempenho, além de dar suporte a aplicações declarativas de forma mais eficiente possível e, portanto, tendo como foco: o sincronismo de mídia na sua forma mais ampla, tendo a interatividade do usuário como caso particular;

a adaptabilidade do conteúdo a ser apresentado; e o suporte a múltiplos dispositivos de interação e exibição. Nasce assim o *middleware* Ginga, incorporando o ambiente procedural GEM estendido, e o ambiente declarativo baseado na linguagem NCL-Lua.

A seguir serão descritos os principais aspectos do ambiente declarativo Ginga-NCL.

Ginga-NCL

O Ginga-NCL é o subsistema “lógico” do *middleware* Ginga que processa documentos NCL. Entre seus módulos chaves está o “Formatador” NCL, que é o responsável por receber um documento NCL e controlar sua apresentação, fazendo com que as relações de sincronismo entre os objetos de mídia existentes sejam respeitadas.

Diferente do HTML, ou XHTML, a linguagem NCL não mistura a definição do conteúdo de um documento com sua estruturação, oferecendo um controle não invasivo, tanto do *layout* do documento, quanto da sua apresentação temporal. Como tal, NCL não define nenhum objeto de mídia, mas apenas a “cola” que mantém esses objetos semanticamente juntos em uma apresentação multimídia.

Objetos de vídeo (MPEG, etc), áudio (AAC, etc), imagem (JPEG, GIF, etc) e texto (TXT, HTML, etc) são exemplos de objetos de mídia que devem ser definidos e tratados por ferramentas de terceiros integradas ao Ginga. Entre esses objetos ressaltam-se os objetos de vídeo e áudio MPEG-4 que, no Sistema Brasileiro de TV Digital, são tratados por exibidores em *hardware*.

Outro objeto importante no sistema brasileiro é aquele baseado em XHTML, tratado como um caso particular de objeto de mídia. Note assim que NCL não substitui XHTML, mas a complementa naquilo que ela é incapaz de cumprir como uma linguagem declarativa. Qual o objeto baseado em XHTML terá suporte na NCL depende da

implementação. De fato, depende de que *browser* XHTML será embutido no formatador NCL.

Dependendo do *browser* escolhido, tem-se compatibilidade com os padrões europeu, americano ou japonês, ou então com a harmonização definida pelo ITU-T na sua recomendação J-201 [16]. Note que o *browser* XHTML pode ter suporte a ECMAScript, e também a eventos de sincronismo carregados pelo fluxo DSM-CC, mantendo compatibilidade com os demais padrões. No entanto, a definição de relacionamentos temporais usando o XHTML (*script*, *links* ou eventos de sincronismo) é desencorajada em NCL, por razões de independência de estruturação, tão bem discutidas na literatura técnica.

Além do objeto XHTML com sua linguagem procedural ECMAScript, outros objetos de execução são permitidos em NCL como objetos de mídia. Entre eles, objetos *XLet* (Java TV), que fazem parte da ponte entre o ambiente declarativo e procedural do Ginga.

Outro objeto procedural que tem suporte em Ginga é o objeto LUA. Lua [13] é uma linguagem de programação poderosa e leve, projetada para estender aplicações.

A linguagem 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, 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.

Lua é uma máquina virtual (*engine*) acoplada ao formatador NCL. Isso significa que, além de sintaxe e semântica, Lua fornece uma API que permite às aplicações NCL trocar dados com programas Lua. É importante também destacar a integração entre Lua e Java, através da biblioteca LuaJava, que permite o acesso a qualquer classe de Java a partir de Lua, de forma similar ao que acontece com ECMAScript. Além disso, o LuaJava

permite que a manipulação do ambiente de Lua a partir de Java, tornando-se, assim, parte da ponte entre os ambientes declarativo e procedural do *middleware* Ginga.

A máquina Lua (código livre e aberto [19]) está implementada como uma pequena biblioteca de funções C, escritas em ANSI C, que compila sem modificações em todas as plataformas conhecidas. Os objetivos da implementação são simplicidade, eficiência, portabilidade e baixo impacto de inclusão em aplicações. Isso faz de Lua uma linguagem muitíssimo mais eficiente que ECMAScript e Java, tanto em termos de tempo de CPU quanto de utilização de memória [20].

Lua é hoje uma das linguagens mais utilizadas no mundo na área de entretenimento (*LucasArts, BioWare, Microsoft, Relic Entertainment, Absolute Studios, Monkeystone Games, etc.*). Naturalmente, NCL-Lua se tornou o “casamento” ideal para o ambiente declarativo do Sistema Brasileiro de TV Digital.

Pode-se entender agora um pouco melhor a operação do “Formatador” NCL. Durante a exibição dos conteúdos dos vários objetos de mídia, exibição esta efetuada pelos diversos exibidores (MPEG, JPEG, HTML, Lua, etc.), eventos são gerados. Exemplos de eventos são: a) apresentação de um segmento marcado (um trecho) de um objeto de mídia (por exemplo, um trecho de um vídeo); b) seleção de um segmento marcado (por exemplo, a seleção de uma âncora em um texto, ou de um botão - uma imagem) etc. Eventos podem gerar ações (de sincronismo) em outros objetos de mídia, tais como parar, iniciar ou pausar suas apresentações.

Assim, os eventos devem ser reportados pelos diversos exibidores ao “Formatador” NCL que, por sua vez, gerará ações a serem aplicadas em outros objetos de mídia.

O padrão Ginga define uma API padrão que todo o exibidor acoplado ao sistema deve obedecer para reportar seus eventos e serem comandados por ações geradas pelo “Formatador”. Exibidores de outros fabricantes, incluindo os *browsers* HTML, usualmente

necessitam de um módulo adaptador para realizar essas funções e se integrarem ao Ginga. Todo relacionamento entre condições de eventos e ações é especificado usando a linguagem NCL.

O formatador NCL foi implementado em C, para integração aos diversos tipos de receptores e em Java (código livre e aberto [25]). A implementação em Java pode ser enviada por *download* para receptores que não estejam prontos para usarem a tecnologia Ginga, fazendo com que aplicações desenvolvidas em NCL tenham suporte em outros sistemas. Consegue-se assim o duplo sentido da interoperabilidade: tanto aplicações geradas em outros sistemas (baseadas em XHTML ou no GEM) têm suporte no Ginga, como aplicações geradas em NCL terão suporte em sistemas que não o brasileiro.

GINGA-J

É a plataforma que utiliza outras API's para o processamento de classes compiladas. Estas, são consideradas como componentes e cada uma é definida para um tipo de serviço, conforme descrito a seguir:

a) **API Java** definida pela Sun: Utilizada para apresentação, seleção de serviços, controle dos gráficos na tela, que será explicada no item seguinte;

b) **API DAVIC** (*Digital Audio-Visual Council*): Criada pela associação DAVIC, essa API especifica formatos de conteúdo para objetos como áudio, vídeo, textos e hipertexto e ainda controla o acesso ao aplicativo e a língua adotada (áudio e legenda);

c) **API Havi** (*Home Audio Video*): Criada por uma associação de companhias de produtos eletrônicos, com objetivo de atuar na apresentação e interface gráfica do usuário, sendo mais robusta do que API Java Sun. Ou seja, é uma API que possui um padrão para interconexão e interoperação de áudio e vídeo digital, a fim de interagirem entre si na rede.

Alem disso, pode gerenciar a rede, a interface do usuário e a comunicação dos componentes. Essa API permite que usuários controlem a aplicação através de botões de um controle remoto [21].

d) **API** definida pelo DVB (*Digital Video Broadcasting*): É uma API relacionada ao padrão DVB, necessária para segurança, acesso de dados e para dispositivos de I/O (entrada/saída).

API Java TV

A API Java TV foi criada pela Sun Microsystems e desenvolvida no ambiente J2ME – Plataforma Java 2 Micro Edition [17], sendo uma extensão da plataforma Java. É uma API utilizada no desenvolvimento de conteúdo para Televisão Digital Interativa, pois provê as funcionalidades necessárias num receptor de TVD, o STB. [18]

A API apresenta um alto nível de abstração, isso é uma característica que facilita no desenvolvimento, pois o desenvolvedor não se preocupa com as camadas mais baixas, que se referem aos protocolos de serviços, transmissão e rede. Funciona como uma espécie de *middleware*, pois se situa entre o Sistema Operacional e as aplicações.

Podem ser encontradas no Java TV, a JVM (*Java Virtual Machine*) e várias bibliotecas destinadas a TVDI (TV Digital e Interativa), que contém no STB (Sistema Televisão Brasileiro). Isso permite ao desenvolvedor escrever apenas uma única vez o código, pois a JVM torna compatível para os receptores, sem se preocupar em saber qual o hardware e *software*. Esta é a vantagem do Java, pois torna a aplicação portátil e compatível.

O Java TV oferece serviços e informações de serviços (SI – *Service Information*), onde serviço pode ser considerado um programa de televisão, ou seja, um conjunto de conteúdo (vídeo, áudio e dados) para apresentação no STB.

SI é uma coleção de informações que especificam o conteúdo dos serviços, que são armazenadas em uma base de dados denominada SI *database*.

Além disso, o Java especifica pacotes que são utilizados para o desenvolvimento de interfaces, navegação, serviços e transportes, podemos citar alguns exemplos tais como:

- **javax.tv.carousel**: Fornece acesso a arquivos de radio difusão e diretório de dados;
- **javax.tv.graphics**: Permite que *Xlets* possam obter seu repositório principal;
- **javax.tv.locator**: Oferece formas para referenciar dados ou aplicativos acessíveis pela API Java TV;
- **javax.tv.xlet**: Provê interfaces para o desenvolvimento e comunicação entre aplicações, oferecendo um gerenciamento.

Considerações Finais

Dada a “imaturidade” da tecnologia da TV Digital novos componentes são adicionados, removidos ou substituídos por versões mais robustas ou mais apropriadas ao *hardware* ou ao sistema operacional com os quais o sistema tenha que operar.

O uso desta abordagem propicia a prototipação e a experimentação, que são fundamentais em sistemas colaborativos voltados para a TV Digital, visto que ainda são escassos e pouco documentados os casos de sucesso. O uso de componentes auxilia a adaptação dinâmica do ambiente e do suporte à colaboração através da recomposição e reconfiguração do sistema.

Ao integrar tanto com Ginga-J quanto com o Ginga-NCL, oferece-se ao desenvolvedor da aplicação de TV Digital a possibilidade de trabalhar tanto com uma linguagem procedimental quanto com uma linguagem declarativa.

Acredita-se que todos os sistemas e aplicações para TV Digital deverão ter como função principal o sincronismo espacial e temporal dos diversos objetos de mídia de que serão compostos.

A adaptabilidade do conteúdo de uma aplicação e de sua apresentação ao tipo de usuário telespectador, ao local onde se encontra este usuário e ao tipo de dispositivo utilizado para exibição do conteúdo, também deverá ser o foco de grande parte das aplicações que, além disso, deverá permitir a interação simultânea a partir de vários dispositivos, como por exemplo controle remoto, celulares, PDAs e a exibição da resposta a esta interação em outros dispositivos que não simplesmente o aparelho de TV.

A existência de um ambiente declarativo para dar suporte a essas funcionalidades não só permitirá uma rápida, confiável e eficiente geração de conteúdos, mas também propiciará uma execução eficiente no ambiente do receptor, tanto em termos de tempo de CPU quanto de memória.

Atualmente, o **Ginga** é o único *middleware* que provê tal suporte e, por isso, se constitui na grande inovação e contribuição do Sistema Brasileiro de TV Digital. NCL, a linguagem declarativa padrão do *middleware* **Ginga**, é uma linguagem de “união” que tem como foco, o sincronismo de mídias, a adaptabilidade e o suporte a múltiplos dispositivos.

Inicialmente, a linguagem teve foco no sincronismo de objetos para documentos gerados para a *Web*. Nos dois últimos anos, entretanto, estendeu seu domínio para atender também às aplicações para TV Digital. Desde o início de seu desenvolvimento, NCL tem contribuído com inovações que têm sido incorporadas por outros padrões internacionais, como o caso citado do padrão MHEG (*Multimedia and Hypermedia information coding Expert Group*). Recentemente, em acordo firmado entre o CWI, PUC-Rio e W3C, NCL e SMIL vêm sendo estudadas no sentido a minimizarem suas diferenças e tornar possível, senão sua junção em uma nova linguagem, pelo menos que exibidores de documentos NCL possam facilmente exibir documentos SMIL, sem grandes alterações em suas funções, e vice-versa.

Um novo padrão internacional de *middleware* declarativo pode daí surgir, indo ao encontro do que tem sido buscado tanto no padrão europeu quanto no americano para seus *middlewares* declarativos.

O **Ginga-NCL** foi desenvolvido pelo Departamento de Pesquisa de Informática da PUC-Rio e é resultado de muitos anos de pesquisa e é apenas um dos muitos sistemas desenvolvidos com tecnologia nacional criados na academia brasileira.

Referências Bibliográficas

- [1] **Alticast Inc, AltComposer 2.0** - USA - Disponível em <http://www.alticast.com>. Acesso em 22 de fevereiro de 2007.
- [2] **Ambiente para Desenvolvimento de Aplicações Declarativas para a TV Digital Brasileira** – Disponível em <http://www.ncl.org.br/documentos/MDIC2007.pdf> acesso em 03 de março de 2008 às 18:24;
- [3] **ARIB** – Association of Radio Industries and Businesses “Data Coding and Transmission Specifications for Digital Broadcasting Volume 2: XML-Based Multimedia Coding Schema”, STD-B24 Versão 4. Fevereiro de 2004.
- [4] **ATSC - Advanced Television Systems Committee** “ATSC Standard: Advanced Common Application Platform (ACAP)”, Padrão A/101, Washington, EUA. Agosto de 2005.
- [5] **Cardinal Information Systems Ltd**, Cardinal Studio 4.0 - Finland - Disponível em <http://www.cardinal.fi>. Acesso em 22 de fevereiro de 2007.
- [6] **Eckel, B.** Thinking in Java. 2a. Edição. Pearson Education. Maio de 2000.
- [7] **ECMA Standardizing Information and Communication Systems**. “ECMAScript Language Specification”, Standard ECMA 262, 3a Edição. Dezembro de 1999.
- [8] **ETSI**. TS 102 819 V1.3.1: Digital Video Broadcasting (DVB) Globally Executable MHP version 1.0.2 (GEM 1.0.2). ETSI Standard. 2005.
- [9] **ETSI – European Telecommunications Standards Institute**. “Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.1.1”, Especificação Técnica ETSI TS 102 B12. Maio de 2005.
- [10] **ETSI Multimedia Home Platform (MHP) Especification 1.1.1**. ETSI Standard. 2003.
- [11] **Fraunhofer Institute for Media Communication IMK**, JAME Author 1.0 – Schloss Birlinghoven - Germany – Disponível em <http://www.jame.tv>. Acesso em 22 de fevereiro de 2007.
- [12] **Halasz, F.G.** “Reflexions on Notecards: Seven Issues for the Next Generation of Hypermedia Systems”. Communications of ACM, Vol.31, No. 7. Julho de 1988.

[13] **Ierusalimschy, R.; Figueiredo, L.H.; Celes, W.** “Lua 5.0 Reference Manual”. Technical Report MCC -14/03, PUC-Rio. Rio de Janeiro. Maio de 2003. ISSN 0103-9741.

[14] **ISO/IEC International Organization for Standardization**. 13818-6:1998, Generic Coding of Moving Pictures and Associated Audio Information – Part 6: Extensions for DSM-CC, 1998.

[15] **ISO/IEC 13522-5**. Information Technology – Coding of multimedia and hypermedia information – Part 5: Support for base-level interactive applications. ISO Standard. 1997.

[16] **ITU-T Recommendation J.201** Harmonization of declarative content format for interactive television applications. Julho de 2004.

[17] **Java TV**. Disponível em <<http://www.java.sun.com/products/javatv>> Acesso em Ago. 2005.

[18] **Lemos, G.; Fernandes, J.; Silveira, G.** - Introdução à Televisão Digital Interativa: Arquitetura, Protocolos, Padrões e Práticas. In: Jornada de Atualização em Informática do Congresso da Sociedade Brasileira de Comunicação, Salvador. Anais. Salvador: [s.e.], 2004.

[19] **Loureiro, J. A.** - Interfaces de Programação para o Desenvolvimento de Aplicações para TV Digital. Pernambuco: UFPE, 2004. Monografia, Faculdade em Ciência da Computação, Centro de Informática, Universidade Federal de Pernambuco, 2004.

[20] **Monteiro, M.B.** - Uma Proposta de Categorização para Aplicações de TV Digital. Pernambuco: UFPE, 2004. Monografia, Faculdade em Ciência da Computação, Centro de Informática, Universidade Federal de Pernambuco, 2004.

[21] **Silva, J. Q.** - TV Digital Interativa. São Leopoldo: UVRS. Monografia, Curso de Especialização em Redes de Computadores, Centro de Ciências Exatas e Tecnológicas, Universidade do Vale do Rio dos Sinos.

[22] **Soares, L.F.G.** - Standard 06 - ISDTV-T Data Codification and Transmission Specifications for Digital Broadcasting, Volume 2 – GINGA-NCL: Environment for the execution of declarative applications. São Paulo, SP, Brazil. ISDTV-T Forum, 2006.

[23] **Soares L.F.G; Rodrigues R.F.** - Nested Context Model 3.0- Part 1 – NCM Core, Technical Report, Monografias em Ciência da Computação do Departamento de Informática da PUC-Rio, maio de 2005, ISSN: 0103-9741.

[24] **Soares L.F.G.; Rodrigues R.F.** - Nested Context Language 3.0 - Part 8 – NCL Digital TV Profiles”. Monografias em Ciência da Computação do Departamento de Informática da PUC-Rio, MCC 35/06, 2006.

[25] **Sourceforge** - Disponível em <<http://sourceforge.net/project/>> Acesso em 05 Ago. 2007.

[26] **Tajra**, S. F. - Informática na Educação. 5ª ed. São Paulo, Editora Erica, 2004.

[27] **World Wide Web Consortium** - “Synchronized Multimedia Integration Language (SMIL 2.1)”. W3C Recommendation. – dezembro de 2005.

[28] **World Wide Web Consortium** - “XHTML™ 1.0 - The Extensible HyperText Markup Language (Second Edition)”. W3C Recommendation - Agosto de 2002.

[29] **Zuffo**, M. K. - TV Digital Aberta no Brasil: Políticas Estruturais para um Modelo Nacional - Departamento de Engenharia de Sistemas Eletrônicos, Escola Politécnica – Universidade de São Paulo, 2004.