

Ambiente para Desenvolvimento de Aplicações Declarativas para a TV Digital Brasileira

Laboratório TeleMídia
Depto. Informática PUC-Rio
Rua Marquês de São Vicente 225; 22453-900 Rio de Janeiro, RJ
ncl@telemidia.puc-rio.br

Resumo:

Aplicações com o foco no sincronismo de mídia e adaptabilidade devem se constituir na maior parte das aplicações de um sistema de TV digital. Prover um bom suporte para a execução e apresentação de tais aplicações é função do ambiente declarativo de um *middleware*. Nos *middlewares* atuais, tais funções, com exceção da interatividade, têm sido resolvidas através de scripts em uma linguagem procedural embutida no ambiente declarativo, e não por um suporte verdadeiramente declarativo da linguagem.

Sem perder a compatibilidade com os outros padrões, ao contrário das implementações correntes, Ginga, o *middleware* padrão do Sistema Brasileiro de TV Digital, oferece um ambiente puramente declarativo, através da linguagem NCL, para a definição e tratamento do sincronismo de mídia e da adaptabilidade, bem como para o suporte à utilização de múltiplos dispositivos de interação e exibição.

1- Introdução

Middleware é a camada de software localizada entre as aplicações (programas de uso final) e o sistema operacional. Seu objetivo é oferecer às aplicações 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 *hardwares*, definindo, para os que produzem conteúdo, uma visão única de aparelho. Esse papel confere à definição do "*middleware* Brasileiro" grande relevo, pois, na prática, é ele quem regulará as relações entre duas indústrias de fundamental importância no país: a de produção de conteúdos e a de fabricação de aparelhos receptores. Do ponto de vista do software, podemos dizer, sem exagero, que, ao definir o *middleware*, estamos, de fato, definindo a "televisão brasileira".

Dominar o conhecimento dessa tecnologia é estratégico para o país, pois, o não domínio certamente acarretaria no não domínio de seu uso.

O universo das aplicações para TV digital pode ser particionado 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.

Um conteúdo procedural é baseado (especificado) 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 [1].

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) [2], SMIL [3] e XHTML [4].

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.

Exemplos de tais ferramentas encontradas em ambientes televisivos são o AVID [5], Final Cut [6], Pro-Tools [7], ferramentas para gerações gráficas, para geração de objetos de texto etc.

Grande parte das aplicações multimídia (interativas 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 “cola” 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)¹, 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 [8]; Cardinal Studio [9]; AltiComposer [10] etc.). A dificuldade e limitação dessas ferramentas são porém evidentes, por não terem como base uma linguagem declarativa eficiente.

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.

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 script, que são de natureza procedural. Mais ainda, uma aplicação declarativa pode referenciar um código procedural embutido (no caso usual de sistemas de TV digital,

¹ 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.

um XLet JavaTV). Da mesma forma, aplicações procedurais podem referenciar conteúdos declarativos, ou até construir e iniciar a apresentação de um conteúdo declarativo.

Assim, sem erro 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 *middlewares* para sistemas de TV digital, este artigo segue, na Seção 2, apresentando um breve histórico do desenvolvimento dos *middlewares*, situando neste histórico o *middleware* padrão do sistema brasileiro Ginga. A Seção 3 aprofunda um pouco mais a discussão sobre o ambiente declarativo do Ginga, em particular sua linguagem declarativa NCL (Nested Context Language), deixando para a Seção 4 as considerações finais.

2 – 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) in 1997 [11], conhecido como MHEG-1 (MHEG part 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 [12], modelo conceitual de dados base da linguagem NCL, propôs uma solução para o problema então em aberto, sobre aninhamento de composições [13], 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 adotadas pelo (Multimedia Home Platform) [14].

MHP foi o primeiro padrão de *middleware* puramente em Java, evoluindo posteriormente para a harmonização GEM (Globally Executable MHP) [15]. Esta mudança do paradigma declarativo em direção a Java, principalmente pela portabilidade de Java, não significou, entretanto, o abandono do paradigma declarativo. Pouco a pouco *middlewares* 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 usam o HTML, no presente momento, o faça 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 abraçasse os diversos legados HTML, levou à complexidade excessiva do DVB-HTML [16]. 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 [17]. Tal é o caso do DVB-HTML, do padrão DVB, e do ACAP-X [18], do padrão ATSC.

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 e ATSC, o sistema ISDB adota um perfil XHTML como base de sua linguagem declarativa, chamada BML [19].

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 [20], denominados b-events em BML.

O padrão ISDB também previu o uso do GEM 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 a 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 ambiente declarativo e procedural 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. É sobre esse ambiente declarativo que trata o restante deste artigo.

3 – O Ambiente Declarativo do Middleware Ginga

Ginga-NCL [21] 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 leiaute do documento (apresentação espacial), 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 (coladas) 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, teremos compatibilidade com os padrões europeu, americano ou japonês, ou então com a harmonização definida pelo ITU-T na sua recomendação J-201 [22].

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 [23] é uma linguagem de programação poderosa 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, 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 a 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 [24]) 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 [25].

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); a 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 terceiros fabricantes, incluindo aí 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 está implementado em C, para integração aos diversos tipos de receptores e em Java (código livre e aberto [26]). A implementação em Java pode ser enviada por *download* para receptores que não são conformes com o 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.

4 – Considerações Finais

A maioria das aplicações para TV digital deverá ter como função principal o sincronismo espacial e temporal dos diversos objetos de mídia que as compõem. A adaptabilidade do conteúdo de uma aplicação e de sua apresentação ao perfil do usuário telespectador, ao local onde se encontra o usuário e ao tipo de dispositivo usado para exibição, também deverá ser o foco de grande parte das aplicações que, além disso, deverá permitir a interação simultânea a partir de vários dispositivos (controle remoto, celulares, PDAs etc) e a exibição da resposta a esta interação em dispositivos outros que não simplesmente o aparelho de TV (o visor do celular, a tela do PDA, etc.). 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. No momento atual, Ginga é o único *middleware* que provê tal suporte e, por isso, se constitui na grande inovação e contribuição do Sistema Brasileiro.

NCL, a linguagem declarativa padrão do *middleware* Ginga, é uma linguagem de cola que tem como seu 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 a 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. 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.

Ginga-NCL é fruto de 15 anos de pesquisa no laboratório TeleMídia do Departamento de Informática da PUC-Rio, e é apenas um dos muitos exemplos do que a tecnologia nacional gerada na academia brasileira é capaz, se oportunidades a ela forem dadas.

Agradecimento: O autor gostaria de agradecer a toda equipe do laboratório TeleMídia, desde seu primeiro time, que deu partida à criação do modelo NCM e da linguagem NCL, até a equipe atual, pela garra e competência demonstrada na concretização do *middleware* Ginga. Ao longo dos últimos anos, o apoio recebido do laboratório SERG da PUC-Rio e da equipe Lua, também da PUC-Rio, foi fundamental e sem ele o Ginga-NCL não seria hoje uma realidade.

Referências

- [1] Eckel, B. Thinking in Java. 2a. Edição. Pearson Education. Maio de 2000.
- [2] 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.
- [3] World-Wide Web Consortium. “Synchronized Multimedia Integration Language (SMIL 2.1)”. W3C Recommendation. Dezembro de 2005.
- [4] World-Wide Web Consortium. “XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition)”. W3C Recommendation. Agosto de 2002.
- [5] <http://www.avid.com/>
- [6] <http://www.apple.com/finalcutstudio/>
- [7] <http://www.digidesign.com/>
- [8] 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.
- [9] Cardinal Information Systems Ltd, Cardinal Studio 4.0 - Finland - Disponível em <http://www.cardinal.fi>. Acesso em 22 de fevereiro de 2007.
- [10] Alticast Inc, Alticomposer 2.0 - USA - Disponível em <http://www.alticast.com>. Acesso em 22 de fevereiro de 2007.

- [11] ISO/IEC 13522-5. Information Technology – Coding of multimedia and hypermedia information – Part 5: Support for base-level interactive applications. *ISO Standard*. 1997.
- [12] Soares L.F.G; Rodrigues R.F. Nested Context Model 3.0: Part 1 – NCM Core, *Technical Report, Departamento de Informática PUC-Rio*. Maio de 2005, ISSN: 0103-9741. Also available in www.telemedia.puc-rio.br/maestro.
- [13] 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.
- [14] ETSI Multimedia Home Platform (MHP) Especification 1.1.1. *ETSI Standard*. 2003.
- [15] 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.
- [16] 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.
- [17] ECMA Standardizing Information and Communication Systems. “ECMAScript Language Specification”, Standard ECMA 262, 3ª Edição. Dezembro de 1999.
- [18] ATSC - Advanced Television Systems Committee “ATSC Standard: Advanced Common Application Platform (ACAP)”, Padrão A/101, Washington, EUA. Agosto de 2005.
- [19] 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.
- [20] 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).
- [21] Soares, L.F.G. (2006). 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.
- [22] ITU-T Recommendation J.201 Harmonization of declarative content format for interactive television applications. Julho de 2004.
- [23] 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.
- [24] <http://www.lua.org>
- [25] Veja e teste em: <http://shootout.alioth.debian.org/>
- [26] <http://www.ncl.org.br>