NCLua SOAP: Acesso a *Web Services* em aplicações de TVDi

Manoel Campos da Silva Filho[1], **Paulo Roberto de Lira Gondim**[2]

[1]Coordenação de Informática
Instituto Federal de Educação, Ciência e Tecnologia do Tocantins (IFTO)
AE 310 Sul, Av LO 05 S/N, Plano Diretor Sul, 77.021-090, Palmas–TO, Brazil

[2]Departamento de Engenharia Elétrica
Universidade de Brasília (UnB)
Campus Universitário Darcy Ribeiro, 70.910-900, Brasília–DF, Brazil

mcampos@ifto.edu.br, pgondim@unb.br

Abstract. *This paper presents a Lua module to access Web Services in interactive Digital TV applications (DTVi). Such applications can be developed using NCL and Lua languages for the Ginga-NCL subsystem of the Ginga middleware. The module allows the convergence between DTVi applications and the Web and the use of existent services in a DTV graphical interface.*

Resumo. *Este artigo apresenta um módulo Lua para acesso a Web Services em aplicações de TV Digital interativa (TVDi). Tais aplicações podem ser desenvolvidas utilizando as linguagens NCL e Lua para o sub-sistema Ginga-NCL do middleware Ginga. O módulo permite a convergência entre aplicações de TVDi e a Web e o uso de serviços já existentes a partir de uma interface gráfica de TVD.*

1. Introdução

Um dos grandes atrativos da TV Digital (TVD) é com certeza a interatividade. O nível de interatividade das aplicações vai desde a chamada interatividade local (onde os usuários podem acessar apenas os dados enviados pela emissora) até a interatividade plena (onde os usuários dispõem de um canal de retorno, permitindo enviar e receber dados em uma rede como a *Internet*) [Soares and Barbosa 2009].

As aplicações com interatividade plena precisam utilizar protocolos de comunicação padrões (como TCP, HTTP e SOAP) para garantir a interoperabilidade com outros sistemas. Utilizando os protocolos citados, é possível garantir a convergência entre *Web* e TV. Com isto, as aplicações de TVDi podem ser enriquecidas com conteúdo proveniente da *Internet*, como o projeto apresentado em [Ghisi et al. 2010].

No entanto, a norma do sub-sistema Ginga-NCL do *middleware* Ginga define apenas a obrigatoriedade do protocolo TCP. Quaisquer protocolos acima da camada de transporte precisam ser implementados pelo desenvolvedor de aplicações.

Assim, são apresentados neste artigo os projetos NCLua HTTP e NCLua SOAP: módulos Lua que implementam HTTP e SOAP para o sub-sistema Ginga-NCL. O desenvolvimento dos projetos partiu da inexistência de versões livres e de código aberto de tais protocolos para o Ginga-NCL.

O artigo está organizado como segue. Na Seção 2 é apresentado o problema de consumo de *Web Services* em aplicações para o Ginga-NCL; na Seção 3 as tecnologias e trabalhos relacionados; na Seção 4 os módulos implementadas; na Seção 6 os resultados alcançados e exemplos de utilização; na Seção 7 as conclusões e por fim os trabalhos futuros.

2. Delimitação do Problema

Apesar de Lua ser uma linguagem extensível [Ierusalimschy et al. 2007], principalmente pela capacidade de utilizar módulos construídos em linguagem C, e de existirem vários destes módulos para as mais diversas finalidades, tais módulos binários não podem ser utilizados em aplicações de TVDi enviadas via *broadcast*, devido a questões de segurança, uma vez que um módulo escrito em linguagem C pode ter acesso a qualquer funcionalidade do sistema operacional (embarcado com o *middleware* no receptor de TVD).

Em [Braga and Restani 2010] são citadas algumas ameaças de segurança em sistemas de TVDi, como transações fraudulentas, pirataria de conteúdo, entre outras. Tais ameaças podem ser potencializadas com o uso de módulos binários. Além do mais, módulos em C possuem código nativo dependente de plataforma, o que não garante que a aplicação executará em qualquer receptor [de Paula Costa 2009]. Somente aplicações residentes podem ser desenvolvidas em linguagens compiladas como C.

Com isto, para haver, em aplicações NCLua de TVDi, algumas das funcionalidades dos módulos binários citados, é preciso implementar módulos inteiramente em linguagem Lua, cujo ciclo de vida é controlado pelo *middleware* Ginga[ABNT 2008].

3. Trabalhos Relacionados

A seguir são apresentadas as tecnologias envolvidas na solução proposta e os trabalhos relacionados.

3.1. Lua e os *scripts* NCLua

Lua é a linguagem imperativa utilizada pelo sub-sistema Ginga-NCL para o desenvolvimento de aplicações procedurais.

Como a linguagem NCL é apenas declarativa, a inclusão de características imperativas a uma aplicação de TVDi para o Ginga-NCL é possibilitada por meio dos chamados NCLua, *scripts* Lua funcionando como nós de mídia dentro de um documento NCL [ABNT 2008] [Soares et al. 2007]. O que diferencia os *scripts* NCLua de *scripts* Lua convencionais é a possibilidade de comunicação bidirecional, por meio de eventos, entre este e o documento NCL [ABNT 2008].

O tratamento de eventos e as particularidades associadas a aplicações Lua para TVDi são implementadas por módulos adicionais à Linguagem[ABNT 2008]. Os módulos de NCLua utilizados no presente artigo são: ***event***, para comunicação bidirecional entre um NCL e um NCLua; e ***canvas***, para desenhar imagens e primitivas gráficas.

3.2. Protocolo TCP no Ginga-NCL

A norma do Ginga-NCL [ABNT 2008] define a disponibilidade do protocolo TCP por meio da classe de eventos *tcp* do módulo *event*. Este permite o tratamento de eventos e a comunicação assíncrona entre o formatador NCL e os *scripts* NCLua.

Devido à característica assíncrona do módulo *event*, o tratamento de requisições TCP em NCLua não é trivial, necessitando do uso de co-rotinas da linguagem Lua, como implementado no módulo *tcp.lua* em [Sant'Anna 2010]. Co-rotinas são similares a um *thread* [Ierusalimschy 2006].

A Figura 1 apresenta um gráfico de máquinas de estados da realização de uma conexão TCP no Ginga-NCL, utilizando o módulo *tcp.lua*. A aplicação estabelece uma conexão a um servidor, envia uma requisição e fica aguardando o retorno. A função *tcp.receive* é executada até que não hajam mais dados a receber, realizando a desconexão.

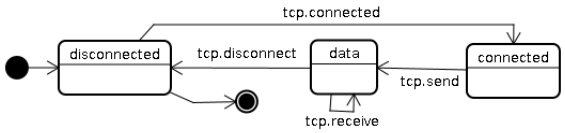


Figura 1. Diagrama de Máquinas de Estados do Módulo *tcp.lua*

A Figura 2 apresenta um gráfico do ponto de vista das co-rotinas em execução. O módulo inicia criando uma co-rotina. A função *coroutine.resume* inicia a execução da co-rotina. Quando é solicitada uma tentativa de conexão, a co-rotina é suspensa (aguardando até uma conexão ser estabelecida). Após a conexão, a co-rotina é novamente resumida (continuando a execução), permitindo que seja enviada uma requisição ao servidor (*tcp.send*). Para a obtenção da resposta da requisição, usa-se a função *tcp.receive*, suspendendo a co-rotina novamente, até que algum dado seja obtido.

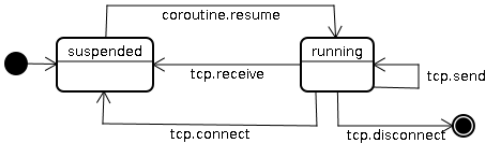


Figura 2. Diagrama de Máquinas de Estados do Módulo *tcp.lua* (corotinas em execução)

A implementação realizadas simplificam diversas chamadas de funções e co-rotinas necessárias à realização de uma conexão TCP no Ginga-NCL, encapsulando diversos detalhes deste processo.

3.3. Implementações de SOAP

SOAP é um protocolo do *World Wide Web Consortium* (W3C), baseado em XML, que permite a aplicações disponibilizarem serviços na *Web*, em uma arquitetura distribuída no modelo cliente/servidor. Tais serviços, denominados *Web Services* (WS), podem ser providos e consumidos por aplicações desenvolvidas em diferentes linguagens e plataformas [W3C 2007] [Curbera et al. 2002] [Newcomer 2002]. O protocolo possui uma linguagem para descrição dos serviços disponibilizados, a *Web Service Description Language* (WSDL). De forma padronizada (manual ou automatizadamente), uma aplicação cliente pode conhecer os serviços de um *Web Service* lendo o WSDL do mesmo[W3C 2007].

A seguir são analisados alguns *toolkits* para provimento e consumo de *WS's* SOAP.

Para acesso a *WS's* SOAP a partir de aplicações Lua, pode-se utilizar o módulo LuaSOAP [Ierusalimschy et al. 2004]. O projeto utiliza o *Expat* [Jr and Waclawek 2007][Cooper 1999] para tratamento de XML. Este é um *parser* XML

desenvolvido em linguagem C, cujos problemas de uso em aplicações de TVDi foram apresentados na Seção 2. Ele usa a biblioteca *LuaSocket*, que depende de módulos em C. Uma das vantagens do projeto é que as chamadas aos métodos remotos no *WS* são síncronas, o que facilita bastante o uso.

O gSOAP é um projeto que permite a aplicações desenvolvidos em linguagens C, C++ ou Fortran, consumirem *WS's*[Van Engelen and Gallivan 2005]. O projeto possui um utilitário para gerar *stubs* em C/C++, a partir do WSDL do serviço. Em tal *stub* são incluídos métodos *proxies* para realizar chamadas aos métodos remotos do serviço. Por ser desenvolvido em C, o mesmo só pode ser utilizado em aplicações de TVD residentes.

Em [Davis and Parashar 2005] são apresentados outros *toolkits* SOAP, como o Apache Axis, uma implementação de SOAP para Java e C. No entanto, esta não atende a um dos requisitos desejados: ser implementado em Lua para uso direto no Ginga-NCL.

4. Proposta

A seguir são apresentados os módulos implementados. O NCLua SOAP está disponível em `http://ncluasoap.manoelcampos.com`. O NCLua HTTP está disponível em `http://ncluahttp.manoelcampos.com`. Tais URL's contêm os módulos, exemplos e toda a documentação dos mesmos. Para montagem do ambiente de desenvolvimento e testes de aplicações que utilizem os módulos implementados são necessários:

- máquina virtual Ginga Virtual Set-top Box 0.12.1 rev 34 ou superior (a ser executada com o VMWare Player);
- IDE Eclipse 3.6.1 para criação de projetos de aplicações;
- plugin NCLEclipse 1.5.1 para dar suporte à linguagem NCL no Eclipse e poder transferir os arquivos das aplicações para a máquina virtual Ginga Virtual Set-top Box;
- plugin LuaEclipse 1.3.1 para dar suporte à linguagem Lua no Eclipse.

4.1. NCLua HTTP

O NCLua HTTP implementa alguns dos principais recursos do protocolo HTTP/1.0. Ele é um módulo escrito inteiramente em linguagem Lua para ser utilizado em *scripts* NCLua. O mesmo utiliza o protocolo TCP da forma como especificado na norma [ABNT 2008].

Pela simplicidade do protocolo HTTP, o módulo possui apenas algumas funções que permitem a geração de requisições e tratamento de respostas. Atualmente existem os seguintes recursos implementados: autenticação básica; download de arquivos; requisições *GET* e *POST*; passagem de cabeçalhos HTTP e definição de *User-Agent*; separação automática dos dados do cabeçalho e do corpo da resposta de uma requisição.

4.2. NCLua SOAP

O NCLua SOAP implementa as principais funcionalidades do protocolo SOAP 1.1 e 1.2. Ele também é um módulo inteiramente escrito em Lua, que faz o *parse* de arquivos XML, realizando o *marshalling* e *unmarshalling* de/para tabelas Lua, permitindo que o desenvolvedor Lua trabalhe com a estrutura de dados principal da linguagem: o tipo *table*.

O módulo utiliza o NCLua HTTP para transportar as mensagens SOAP. Com isto, a implementação do SOAP fica bastante simplificada, tornando o código fácil de

ser mantido. O NCLua SOAP encarrega-se apenas de gerar o XML da requisição SOAP, utilizando o NCLua HTTP para enviar tal XML no corpo da mensagem. O *parse* e *unmarshalling* do XML para uma tabela Lua é todo encapsulado pelo módulo LuaXML[1].

Um importante recurso, que facilita bastante a utilização do módulo, é a simplificação do XML retornando como resposta, que é convertido (*unmarshalling*) automaticamente para uma tabela Lua. Para demonstrar este recurso, utilizar-se-á um *WS* de consulta de endereço a partir de um CEP. Tal WS possui um método chamado ”cep”, que recebe um determinado CEP e retorna o endereço referente ao mesmo. O XML de retorno do método ”cep”, apresentado como tabela Lua, é semelhante ao da Listagem 1.

Listagem 1. Tabela Lua gerada a partir de um XML de resposta a uma requisição SOAP

```
1 {cepResult =  {diffgr = {NewDataSet = {tbCEP = {
2    nome="Cln 407", bairro="Asa Norte", UF="DF", cidade="Brasilia" }}}}}
```

Como pode ser visto, o elemento da tabela que contém de fato os dados do endereço retornado (tbCEP) está envolvido em várias outras tabelas que não contém dado algum, sendo estruturas completamente desnecessárias para a aplicação NCLua. Com isto, para o desenvolvedor poder acessar, por exemplo, a cidade do CEP indicado, precisará conhecer toda a estrutura retornada, utilizando uma instrução como *result.cepResult.diffgr.NewDataSet.tbCEP.cidade*.

Para esconder estes detalhes do desenvolvedor, o NCLua SOAP simplifica qualquer resultado que contenha estruturas desnecessárias, como o mostrado na Listagem 1. Desta forma, para o exemplo citado, a tabela Lua (gerada a partir do XML de retorno da requisição) ficará como apresentado na Listagem 2, o que simplifica o acesso aos elementos da estrutura retornada, permitindo, por exemplo, que o campo cidade seja acessado utilizando-se apenas a instrução *result.cidade*.

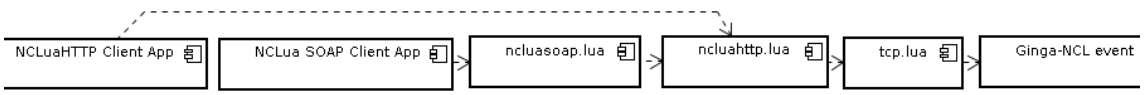
Listagem 2. Simplificação de retorno de uma requisição SOAP feita pelo NCLua SOAP

```
1 {   nome="Cln 407", bairro="Asa Norte", UF="DF", cidade="Brasilia"   }
```

O módulo ainda possui um *script* (*wsdlparser.lua*) que realiza o *parse* de um WSDL e obtém algumas das informações que precisa-se passar para que ele realize a requisição (como o *namespace* do serviço e o nome do método desejado).

Para resumir, as características principais do NCLua SOAP são: suporte a SOAP 1.1 e 1.2; parâmetros de entrada e saída *struct* e *array*; facilidade para manipular chamadas assíncronas; simplicidade na obtenção do retorno de uma chamada remota; SOAP *Fault* para captura de erros SOAP; SOAP *Header*[W3C 2007] para passagem de parâmetros específicos da aplicação (como informações de autenticação, pagamento, etc).

A Figura 3 apresenta um diagrama de componentes dos módulos implementados, mostrando a dependência entre os componentes. Todos os módulos já foram apresentados, exceto o *NCLua HTTP Client App* e o *NCLua SOAP Client App*, que representam, respectivamente, uma aplicação cliente usando HTTP e uma aplicação consumindo *WS's*.



[1] *Parser* XML escrito inteiramente em Lua, adaptado para funcionar com Lua 5

Figura 3. Diagrama de Componentes do NCLua SOAP e NCLua HTTP

5. Estudos de Caso

As implementações apresentadas agilizam a construção de aplicações com interatividade plena, realizando acesso ao canal de retorno por meio de protocolos padronizados. Existem diversas categorias de aplicações interativas que podem ser construídas com uso dos protocolos implementados, tais como jogos, informações (notícias, horóscopo, previsão do tempo, etc), educação (*T-Learning*), governo eletrônico (*T-Government*), comércio eletrônico (*T-Commerce*), saúde (*T-Health*), bancárias (*T-Banking*) e outras, como apresentado em [Fernández and Goldenberg 2008], [de Paiva Teixeira 2006] e [de Carvalho Barbosa et al. 2010].

Dentre as categorias de aplicações apresentadas, as aplicações de governo eletrônico, saúde e educação pública tem como objetivo fornecer serviços à população em geral, democratizando e tornando mais acessíveis serviços que normalmente são disponibilizados em sedes de órgãos do governo ou pela *Internet*. Com o uso dos recursos de interatividade da TV Digital, aplicações destas categorias servem para promover a inclusão digital e social.

Alguns dos serviços que podem ser providos pela TV Digital são:

- marcação de consultas em hospitais públicos;
- enquetes/consultas à população;
- campanhas de vacinação e para controle de epidemias;
- educação à distância, tutores inteligentes;
- informações de previdência social (como o projeto TV Digital Social[2])

Apesar de a TV estar em cerca de 96% dos lares brasileiros[IBGE 2009], em 2009, apenas 25% dos lares tinham acesso à Internet[CETIC.br 2009]. O uso da interatividade plena requer acesso à *Internet*, e como pôde-se ver, o acesso residencial à mesma, no Brasil, é bastante restrito. No entanto, com a criação do Plano Nacional de Banda Larga (PNBL) por meio do Decreto número 7.175, de 12 de maio de 2010, visando prover *Internet* banda larga a um custo reduzido, este cenário tende a mudar. Desta forma, o uso de serviços da *Internet*, inclusive de governo eletrônico, por meio de protocolos como HTTP e SOAP, se tornará cada vez mais popular, considerando todos os benefícios de agilidade que estes trazem, sem a necessidade de locomoção do cidadão até uma agência do governo.

6. Resultados

Foram realizados diversos testes de interoperabilidade com servidores *Web* e *WS's*, estes últimos desenvolvidos em diferentes linguagens. A atual versão dos módulos conseguiu fazer a interoperação com tais serviços normalmente. As subseções a seguir apresentam exemplos de uso dos módulos implementados.

6.1. Exemplos de uso do NCLua HTTP

O exemplo da Listagem 3 envia uma requisição HTTP *GET* para uma página *Web*, com o parâmetro ”voto”igual a ”sim”, obtendo o resultado e exibindo no terminal.

[2]`http://clube.ncl.org.br/node/101`

A linha 1 adiciona o módulo NCLua HTTP. A linha 7 chama a função *http.request* que envia a requisição HTTP. Devido à particularidade assíncrona do protocolo TCP no Ginga-NCL (que é utilizado para transportar as mensagens HTTP), para facilitar o envio de requisições e recebimento de respostas, o módulo NCLua HTTP utiliza co-rotinas de Lua. Por este motivo, a obtenção do retorno deve ser feita dentro de uma função, como a definida na linha 5, contendo os parâmetros *header* e *body* (comentados na definição da mesma). Tal função deve ser passada como parâmetro à *http.request*, para que seja chamada por esta quando a resposta for obtida.

Listagem 3. Exemplo de envio de requisição GET com NCLua HTTP

```
1 require "http"
2 --Funcao de callback executada de forma assincrona pelo modulo
3 --@param header Headers HTTP enviados na resposta
4 --@param body Corpo da mensagem HTTP
5 function getResponse(header, body) print("Resposta obtida\n", body) end
6
7 http.request("http://manoelcampos.com/voto.php?voto=sim", getResponse)
```

O envio de requisições *POST* é bastante semelhante ao exemplo apresentado anteriormente. Neste caso, os parâmetros devem ser passados à função *http.request* por meio de uma tabela Lua. A formatação destes valores, como exigido pelo protocolo HTTP, é feita automaticamente pelo NCLua HTTP.

6.2. Exemplo de uso do NCLua SOAP

O exemplo apresentado na Listagem 4 consume um *WS* para obtenção de um endereço a partir do CEP. Na função *getResponse* (linha 2), o resultado retornado é um tipo composto, que é acessado como uma tabela Lua. Tal tabela conterá campos com os valores armazenados no XML enviado pelo *Web Service*.

Listagem 4. Exemplo de consumo de WS de consulta de endereço a partir do CEP

```
1 require "ncluasoap"
2 function getResponse(r) print(r.nome,r.bairro,r.cidade,r.UF) end
3 local msg = {
4   address = "http://www.bronzebusiness.com.br/webservices/wscep.asmx",
5   namespace = "http://tempuri.org/",
6   operationName = "cep", params = {  strcep = "70855530"  }
7 }
8 ncluasoap.call(msg, getResponse)
```

7. Conclusão

Os módulos implementados facilitam a convergência entre *Web* e TV, escondendo os detalhes de implementação dos protocolos HTTP e SOAP do desenvolvedor de aplicações de TVDi, permitindo o surgimento de novas aplicações interativas que fazem uso de conteúdo da *Internet*.

As implementações apresentadas trazem ainda como benefícios para os desenvolvedores de aplicações NCLua:

- encapsulamento de toda a complexidade dos protocolos HTTP e SOAP;
- encapsulamento dos detalhes do procotolo TCP do sub-sistema Ginga-NCL;

- facilidade e agilidade no desenvolvimento de aplicações com interatividade plena;
- serem implementações de código aberto que permitem a evolução das mesmas pela comunidade de desenvolvedores de aplicações de TV Digital.

Algumas aplicações foram desenvolvidas como prova de conceito do uso dos módulos. Algumas delas foram: NCLua RSS Reader (leitor de RSS); Enquete TVD; TVD Quizz; NCLua Tweet (cliente de *Twitter*); Rastreador de Encomendas dos Correios. O NCLua SOAP tem permitido, recentemente, o desenvolvimento de aplicações tais como: *T-Government* [de Carvalho Barbosa et al. 2010], recomendação de conteúdo [Gatto 2010], entre outras.

Atualmente, o processo de obtenção dos dados para realizar a chamada a um método remoto em um *WS* ainda é praticamente todo manual, no entanto, após o desenvolvedor obter tais dados para o método remoto desejado, a realização da requisição é bem simples. O *feedback* dos usuários tem mostrado que o módulo está sendo bastante útil para a comunidade, além de permitir a evolução do mesmo.

Uma tabela comparativa e análise de desempenho dos módulos foram construídas, no entanto, não foram apresentadas aqui por restrições de espaço.

8. Trabalhos Futuros

Como trabalhos futuros, pretende-se: concluir a implementação do *parse* automático do documento WSDL e geração de *stubs* Lua, contendo funções *proxies* para realizar a chamada aos métodos remotos (semelhante a ferramentas como o *wsdl2java*[3]); incluir tratamento de exceções para permitir que as aplicações de TVDi possam emitir mensagens amigáveis ao usuário quando uma requisição HTTP falhar; e implementar o envio de anexos em formato *Multipurpose Internet Mail Extensions* (MIME) [IETF 1996].

No módulo NCLua HTTP, pretende-se realizar testes de conformidade utilizando-se os métodos HTTP *OPTIONS, HEAD, PUT* e *DELETE*. Pretende-se também implementar mais funcionalidades no módulo, como realizar redirecionamentos automaticamente a partir de respostas HTTP 301 e 302, e implementar recursos de conexões persistentes.

Referências

[ABNT 2008] ABNT, N. (2008). 15606-2. Televisão digital terrestre–Codificação de dados e especificações de transmissão para radiodifusão digital Parte 2: Ginga-NCL para receptores fixos e móveis–Linguagem de aplicação XML para codificação de aplicações.

[Braga and Restani 2010] Braga, A. and Restani, G. (2010). Introdução à segurança de aplicações para a TV digital interativa brasileira. Simpósio Brasileiro de Segurança.

[CETIC.br 2009] CETIC.br (2009). Tic Domicílios e Usuários - Pesquisa sobre o Uso das Tecnologias da Informação e da Comunicação no Brasil.

[Cooper 1999] Cooper, C. (1999). Using expat. Disponível em: `http://www.xml.com/pub/a/1999/09/expat/`. Acessado em: 28 out. 2010.

[3] `http://ws.apache.org/axis/java/user-guide.html`

[Curbera et al. 2002] Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., and Weerawarana, S. (2002). Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. IEEE Internet computing, 6(2):86–93.

[Davis and Parashar 2005] Davis, D. and Parashar, M. (2005). Latency performance of SOAP implementations. In Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on, page 407. IEEE.

[de Carvalho Barbosa et al. 2010] de Carvalho Barbosa, R., Kutiishi, S. M., and de Lima, V. (2010). Desenvolvendo serviços de governo eletrônico multiplataforma para TV interativa utilizando Web Services. WebMedia.

[de Paiva Teixeira 2006] de Paiva Teixeira, L. (2006). Usabilidade e Entretenimento na TV Digital Interativa. UNIrevista, 1(3).

[de Paula Costa 2009] de Paula Costa, L. C. (2009). Segurança para o Sistema Brasileiro de Televisão Digital: Contribuições à Proteção de Direitos Autorais e à Autenticação de Aplicativos. Escola Politécnica da Universidade de São Paulo.

[Fernández and Goldenberg 2008] Fernández, F. and Goldenberg, S. (2008). Aplicaciones interactivas para la televisión digital en Chile. Cuadernos de información, I(22):12.

[Gatto 2010] Gatto, E. C. (2010). Personalização de Programas de TV no contexto da TV Digital Portátil Interativa. UFScar.

[Ghisi et al. 2010] Ghisi, B. C., Lopes, G. F., and Siqueira, F. (2010). Integração de Aplicações para TV Digital Interativa com Redes Sociais. WebMedia.

[IBGE 2009] IBGE (2009). Pesquisa Nacional por Amostra de Domicílios 2009 - PNAD.

[Ierusalimschy 2006] Ierusalimschy, R. (2006). Programming in Lua. Lua.org, 2nd. edition.

[Ierusalimschy et al. 2004] Ierusalimschy, R., Carregal, A., and Guisasola, T. (2004). LuaSOAP - SOAP interface to the Lua programming language. Disponível em: `http://www.keplerproject.org/luasoap/`. Acessado em: 28 out. 2010.

[Ierusalimschy et al. 2007] Ierusalimschy, R., de Figueiredo, L., and Celes, W. (2007). The evolution of Lua. In Proceedings of the third ACM SIGPLAN conference on History of programming languages, page 2. ACM.

[IETF 1996] IETF (1996). MIME. Disponível em: `http://tools.ietf.org/html/rfc2045`. Acessado em: 02 nov. 2010.

[Jr and Waclawek 2007] Jr, F. L. D. and Waclawek, K. (2007). The Expat XML Parser. Disponível em: `http://www.libexpat.org`. Acessado em: 28 out. 2010.

[Newcomer 2002] Newcomer, E. (2002). Understanding Web Services: XML, Wsdl, Soap, and UDDI. Addison-Wesley Professional.

[Sant'Anna 2010] Sant'Anna, F. (2010). Documentação NCLua. Disponível em: `http://www.lua.inf.puc-rio.br/~francisco/nclua/`. Acessado em: 31 out. 2010.

[Soares and Barbosa 2009] Soares, L. and Barbosa, S. (2009). Programando em NCL 3.0: Desenvolvimento de aplicações para o middleware Ginga. Campus, Rio de Janeiro, RJ.

[Soares et al. 2007] Soares, L., Rodrigues, R., and Moreno, M. (2007). Ginga-NCL: the declarative environment of the Brazilian digital TV system. Journal of the Brazilian Computer Society, 12:37–46.

[Van Engelen and Gallivan 2005] Van Engelen, R. and Gallivan, K. (2005). The gSOAP toolkit for web services and peer-to-peer computing networks. In Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on, page 128. IEEE.

[W3C 2007] W3C (2007). SOAP Specification. Disponível em: `http://www.w3.org/TR/soap/`. Acessado em: 02 nov. 2010.