Suporte a Metadados Semânticos para o Desenvolvimento de Aplicações Interativas de TV Digital

Aline M. Saettler[1], Roberta L. Gomes[2], Renata S. S. Guizzardi[2]

[1]Departamento de Informática – Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
Rio de Janeiro – RJ – Brasil

[2]Departamento de Informática – Universidade Federal do Espírito Santo (UFES)
Vitória – ES – Brasil

asaettler@inf.puc-rio.br, {rgomes, rguizzardi}@inf.ufes.br

Abstract. *This paper aims to present a way of metadata support in interactive applications for Digital TV, respecting the Brazilian standards. It presents the development of a library in Lua to provide such support, and an application which aims to test its usability. It also presents a study about metadata in the context of Digital TV and about the development of applications according to the current standards, established by SBTVD.*

Resumo. *Este artigo tem por objetivo apresentar um meio de suporte a metadados em aplicações interativas para TV Digital, respeitando os padrões brasileiros. Ele apresenta o desenvolvimento de uma biblioteca em Lua para prover tal suporte, e uma aplicação que visa testar a usabilidade da mesma. Ele também apresenta um estudo sobre metadados no contexto da TV Digital e sobre o desenvolvimento de aplicações de acordo com as normas vigentes, estabelecidas pelo SBTVD.*

1. Introdução

O advento da Televisão Digital no Brasil trouxe diversas vantagens. Além da melhora de qualidade nos sons e imagens em relação a transmissões analógicas, essa mudança permitiu a construção de programas computacionais interativos, que podem ser enviados ao telespectador junto com o áudio e o vídeo de um programa de TV. Assim, ampliou-se a quantidade de serviços disponibilizados através da televisão, entre os quais pode-se citar, por exemplo, os serviços de saúde, educacionais, ou de guias eletrônicos de programação. [Barbosa e Soares 2008]

Aplicações interativas permitem que o telespectador, através do controle remoto, interaja com uma série de serviços. Além disso, dados do telespectador podem ser capturados e enviados aos provedores de serviços, através do canal de retorno. De posse desses dados, as aplicações são capazes de prover conteúdos mais relevantes e específicos a cada telespectador, fornecendo, assim, serviços muito mais eficientes aos usuários de TV Digital. O *middleware* do Sistema Brasileiro de Televisão Digital (SBTVD), o Ginga [Ginga 2011], oferece suporte à interatividade através de linguagens como NCL e Lua.

Uma melhoria considerável nos serviços que podem ser oferecidos para a TV Digital vem da utilização de metadados por parte das aplicações. Metadados podem ser definidos como *dados sobre dados*. No contexto da TV Digital, os programas podem ser

descritos de uma forma muito mais ampla. Pode-se utilizar, por exemplo, um domínio para descrever detalhadamente um programa, ou até mesmo um item vendido pela TV. De posse de informações mais precisas e detalhadas, é mais fácil identificar o conteúdo pelo qual um usuário realmente se interessa. Através de aplicações interativas, esse conteúdo pode ser buscado e apresentado de uma forma eficiente e que corresponda às expectativas dos usuários de TV Digital. O conteúdo apresentado a eles por aplicações que não utilizam metadados pode não refletir os seus interesses. Além disso, as descrições de programas de TV fornecidas pelos provedores de serviços sem a utilização de metadados são muito mais limitadas, tornando a busca de conteúdo mais difícil.

O Ginga oferece suporte a metadados permitindo o uso do *Resource Description Framework* (RDF), um modelo para a representação de metadados usado para descrever páginas Web. Pode-se incluir código em RDF dentro do código de aplicações. Apesar da norma [ABNT NBR 15606-2 2007] definir o uso de RDF como padrão para especificação de metadados em NCL, até o desenvolvimento deste trabalho, não havia nenhum suporte disponível para o processamento desses metadados nas implementações de referência ou abertas do Ginga, sendo esta a motivação para o desenvolvimento da presente iniciativa.

Este trabalho apresenta uma biblioteca em Lua para prover suporte a metadados RDF, levando em conta os padrões e normas estabelecidos pelo SBTVD. As próximas seções estão organizadas do seguinte modo: a seção 2 apresenta trabalhos relacionados; a seção 3 apresenta as ferramentas utilizadas como auxílio no desenvolvimento deste trabalho; a seção 4 apresenta a biblioteca em si e aspectos da sua implementação, bem como uma aplicação utilizada para teste; a seção 5 apresenta conclusões e possíveis trabalhos futuros.

2. Uso de Metadados na TV Digital

Nesta seção, são apresentados diversos padrões de metadados utilizados no contexto da TV Digital (TVD) e tecnologias de TVD no Brasil. São mostrados também exemplos de aplicações para a televisão que utilizam metadados para fornecer serviços eficientes aos usuários e ferramentas disponíveis atualmente para o tratamento de metadados.

2.1 TV Digital e o *middleware* Ginga

Um *middleware* é uma camada de software posicionada entre o código das aplicações e a infra-estrutura de execução [Soares e Castro 2008]. Assim, um *middleware* contém bibliotecas e ambientes para a construção de aplicações. O SBTVD Terrestre apresenta um *middleware* próprio, o Ginga. O Ginga possui ambientes tanto para a construção de aplicações declarativas quanto para a construção de aplicações procedurais. Em aplicações procedurais, há a necessidade de se especificar cada passo executado pelo algoritmo da aplicação, enquanto em aplicações declarativas, o programador pode concentrar-se nos resultados que deseja obter. Assim, em uma aplicação declarativa, são definidos, por exemplo, as regiões da tela em que algo irá aparecer, o tipo de conteúdo que será mostrado (áudio, vídeo, etc) e a ordem em que irão aparecer. Não há a necessidade de se preocupar com a maneira como esse conteúdo será executado. Uma aplicação procedural, por outro lado, é capaz de executar diversas operações que seriam dificultadas ou até mesmo não permitidas por linguagens declarativas, como manipulação de *strings* e operações matemáticas.

O ambiente para a construção de aplicações declarativas no Ginga é o Ginga-NCL [Soares 2007], que utiliza a linguagem NCL (*Nested Context Language*) para a construção das aplicações. O ambiente para a construção de aplicações procedurais é o Ginga-J [Souza Filho 2007], mais recente e com poucos ambientes de teste. Este trabalho foi desenvolvido para ser utilizado na construção de aplicações declarativas feitas para o Ginga-NCL. Como mencionado anteriormente, linguagens declarativas não são ideais para a execução de certas operações. Assim, juntamente com a linguagem declarativa NCL pode-se utilizar uma linguagem com um paradigma procedural, como a linguagem Lua.

Lua foi criada no Brasil por Roberto Ierusalimschy [Lua.org 2011] e é uma linguagem leve e de tipagem dinâmica, comumente utilizada no desenvolvimento de jogos.

O NCL permite que *scripts* Lua sejam chamados durante a execução de aplicações. Um arquivo Lua é capaz de se comunicar com um documento NCL, através dos chamados eventos. Um evento pode representar, por exemplo, a ação de pressionar um botão no controle remoto ou o início ou término de um vídeo. Um *script* Lua pode tanto receber quanto enviar eventos, se comunicando desse modo com o documento NCL.

A biblioteca desenvolvida neste trabalho foi implementada em Lua, pois, como dito anteriormente, a linguagem permite a execução de determinadas operações que seriam inviáveis somente com o uso de NCL.

2.2 Metadados em TV Digital

Metadados são utilizados para descrever dados. Metadados sobre uma música, por exemplo, podem incluir o nome do compositor e do intérprete da música em questão, entre outras informações relevantes. Um dos contextos que mais explora o uso de metadados, hoje em dia, é a Web. Tomando ainda o exemplo da música, tais informações disponibilizadas pelos metadados podem auxiliar na compra da música em questão por um usuário, ou mesmo dar suporte a um programa recomendador a inferir informação sobre a música, para sugeri-la a alguém que aprecie tal compositor ou intérprete.

Vários padrões de metadados são utilizados atualmente na construção de aplicações para TV Digital. No Brasil, o RDF é o padrão adotado pelo SBTVD. No entanto, muitas aplicações são desenvolvidas em outros países utilizando-se padrões como o MPEG-7, o MPEG-21 e o TV-Anytime [Alves, Silva e Bressan 2008]. [Alves et al. 2006] fornece uma análise comparativa sobre esses três padrões. Já OWL (*Web Ontology Language*), uma linguagem que utiliza o conceito de ontologias para a descrição de conteúdo na Web, é utilizada em diversos trabalhos para a descrição de metadados na área de TV Digital [Alam et al. 2009] [Jr et al. 2009], embora originalmente não tenha sido criada para este propósito.

Metadados podem ser aplicados de diversas formas no contexto da TV Digital. [Lui et al. 2009] apresenta uma aplicação que utiliza metadados para a recomendação personalizada de conteúdo. A aplicação utiliza uma ontologia para a descrição de conteúdo e divide os programas em gêneros, descritos na ontologia. Os usuários buscam e especificam o tipo de conteúdo que querem assistir, e a aplicação retorna recomendações com base no que foi especificado por eles. [Blanco-Fernández et al.

2009] utiliza metadados para descrever produtos e serviços. Os telespectadores podem realizar buscas de acordo com seus interesses, comprar produtos pela internet através da televisão e requisitar informações sobre determinados produtos, entre outros. [Alves, Silva e Bressan 2008] apresenta uma arquitetura com base na participação colaborativa entre os usuários. Metadados descrevem programas e também características pessoais e preferências de usuários. A participação colaborativa se dá na forma de avaliação dos programas. Os usuários classificam o conteúdo visto e essa avaliação fica disponível a outros usuários.

Em mais detalhes, esse tipo de aplicação pode funcionar da seguinte maneira: uma ontologia define os conceitos relevantes para determinado contexto. Por exemplo, em aplicações que recomendam programas de TV, habitualmente se define que Telespectador *prefere* Programa *classificado em* um determinado Tipo. Os conceitos Telespectador, Programa e Tipo têm determinadas propriedades (ex.: Programa tem **título, horário, faixa etária recomendada** etc.) A medida que os usuários vão escolhendo programas para assistir na TV, a aplicação vai reunindo informações que indiquem a preferência do usuário, instanciando, assim, o modelo ontológico para determinado usuário, criando uma espécie de perfil. Tanto os programas quanto esse perfil podem ser descritos a partir de metadados definidos a partir dos conceitos ontológicos, em arquivos textuais ou bancos de dados específicos para metadados.

3. APIs para o tratamento de Metadados

Nesta seção, serão descritas as duas ferramentas utilizadas para o desenvolvimento da biblioteca: o Lua XML Parser e o Jena. A primeira foi utilizada para estruturar documentos na linguagem RDF em tabelas em Lua. A segunda foi utilizada para o estudo das principais funções a serem posteriormente implementadas em Lua.

3.1 *Parsers* XML desenvolvidos em Lua

Como parte do desenvolvimento desta biblioteca, foram estudadas maneiras já existentes de se estruturar documentos XML em Lua. Existem duas técnicas que podem ser utilizadas para processar esses documentos: *Simple API for XML* (SAX) e *Document Object Model* (DOM). Enquanto o DOM estrutura o documento XML em uma árvore, o SAX o transforma em uma sucessão de chamadas de funções [Veloso 2007]. Entre as ferramentas estudadas, podemos destacar o LuaExpat [LuaExpat 2011] e o Lua XML Parser. [LuaXML 2011]

O LuaExpat é uma API que utiliza SAX para tratar os documentos XML e é baseado em outro *parser*, escrito em C, para XML: o Expat. O LuaExpat define funções para o início e fim de *tags*, para texto entre as *tags*, entre outras. Uma tabela contém referências para essas funções, que são chamadas quando um determinado elemento é reconhecido pelo *parser*. Diferentemente do Lua XML Parser, porém, o LuaExpat exige uma pré-instalação no Ginga, inclusive de outros componentes necessários para o seu funcionamento, o que dificultaria o uso da biblioteca RDF. Por isso, o Lua XML Parser foi escolhido como base para o desenvolvimento deste trabalho.

3.2 APIs e Frameworks para tratamento de dados RDF

Como parte dos estudos realizados para o desenvolvimento da biblioteca, foram analisadas ferramentas já disponíveis para o tratamento de documentos que utilizam a

linguagem RDF. A maioria das ferramentas disponíveis utiliza a linguagem Java, entre as quais podemos citar o JRDF [JRDF 2011] e o Jena [Jena 2011].

O JRDF é um Framework de código aberto, que dispõe de uma série de funcionalidades para o tratamento de dados em RDF. Ele oferece uma API para a criação e manipulação de grafos, incluindo a criação de recursos, *statements*, e outros elementos. O JRDF também provê suporte a SPARQL, uma linguagem de consultas para RDF.

O Jena também é um Framework de código aberto, bastante utilizado para a construção de aplicações baseadas em metadados tais como RDF e OWL. Esse framework possui uma API para tratamento de RDF com um grande número de funcionalidades e se mostrou completo e simples de se utilizar, sendo, portanto, escolhido como foco dos estudos.

3.3 Lua XML Parser

O Lua XML Parser é um *parser* que fornece meios de se estruturar um documento XML em tabelas em Lua. Ele se mostrou extremamente eficiente para as necessidades deste trabalho, além de ser composto apenas de arquivos em Lua que podem ser simplesmente incluídos em outros arquivos, sem qualquer necessidade de instalação de programas e conseqüente alteração da configuração inicial do Ginga Virtual Set-top Box[1], utilizado para os testes. Assim, ele foi escolhido como foco de estudos.

O Lua XML Parser não é especifico para o tratamento de metadados, e sim um *parser* para documentos XML. No entanto, o RDF e o NCL utilizam a sintaxe do XML. Por isso, metadados podem ser inseridos em arquivos NCL sem qualquer problema para a execução de aplicações declarativas.

O Lua XML Parser permite que diferentes elementos de um arquivo XML (tais como *tags* e atributos) sejam armazenados em uma única estrutura de tabelas em Lua. Utilizando-se o Lua XML Parser, uma estrutura de árvore é gerada a partir de um documento XML pela função *domHandler*. Essa estrutura contém um nó raiz e um vetor de filhos, que são nós que contêm os seguintes campos:

- **_name:** nome do elemento
- **_type:** ROOT, ELEMENT, TEXT, etc
- **_attr:** atributos do nó
- **_parent:** nó pai
- **_children:** vetor de nós filhos

A figura 2 mostra a estrutura gerada para armazenar o documento da figura 1:

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:inf="http://www.exemplo/informacoes">
  <rdf:Description rdf:about="http://exemplo/aline">
    <inf:idade>21</inf:idade>
    <inf:curso>ciencia da computacao</inf:curso>
  </rdf:Description>
</rdf:RDF>
```

Figura 1. Trecho de código em RDF

[1] Máquina virtual utilizada para testar aplicações declarativas

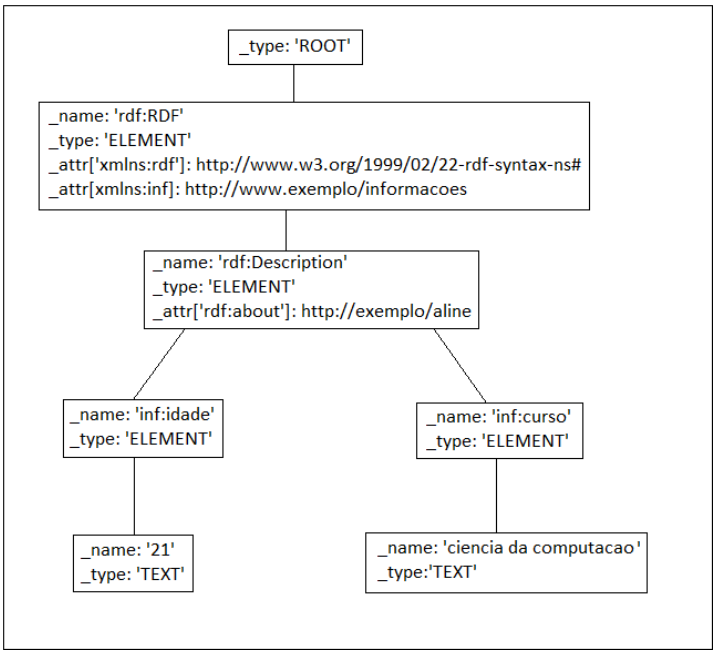

Figura 2. Árvore do documento XML

Durante este trabalho, foi necessário compreender o modo como documentos XML são estruturados na árvore gerada pela função *domHandler* e como um trecho de código RDF fica armazenado de modo que cada um dos seus elementos possa ser facilmente acessado.

3.4 Jena Framework

A API para RDF fornecida pelo Jena inclui diversas funções para a criação e edição de modelos em RDF. Um modelo é um conjunto de recursos, propriedades e valores. Na API, esses recursos, propriedades e outros elementos presentes na linguagem são definidos como classes em Java. O Jena utiliza, portanto, uma abordagem orientada a objetos. Recursos e *statements*, por exemplo, são classes e possuem seus próprios métodos.

No RDF, um recurso representa algo a ser descrito (por exemplo, um indivíduo). Um recurso é descrito através de suas propriedades (tais como *idade* e *sobrenome*), que possuem valores. Uma tripla (*statement*) é composta por um recurso, uma propriedade e um valor.

Assim, no Jena existem funções para a criação de recursos, de propriedades e de *statements*, para recuperar propriedades de um dado recurso, para adicionar *statements* a modelos e diversas outras, baseadas nos elementos do RDF.

O Jena foi utilizado para o estudo do comportamento das funções oferecidas para a criação e manipulação desses elementos, que foram posteriormente reimplementadas em Lua.

4. Desenvolvimento de uma biblioteca em Lua para RDF

Para a implementação da biblioteca em Lua, foram utilizados como base o Lua XML Parser e o Jena, mencionados na seção anterior.

Com o Lua XML Parser, foi montada uma estrutura de árvore que segue um padrão único. Todos os elementos de um documento RDF são estruturados de acordo com esse padrão. Na raiz, ficam armazenados os *namespaces* utilizados em um documento RDF e seus prefixos. No segundo nível, ficam os recursos; no terceiro, as propriedades; e no quarto e último nível, os valores dessas propriedades (que podem ser também outros recursos).

Tanto recursos como propriedades são representados como tabelas em Lua. Caso um recurso seja adicionado como valor de uma propriedade *p*, uma *string* com o seu identificador é adicionada à tabela associada à *p*, e uma nova tabela é criada e inserida no primeiro nível da árvore, representando esse recurso. Todas as mudanças a serem realizadas posteriormente (como por exemplo, a adição de uma nova propriedade) com relação ao recurso são feitas na tabela adicionada ao primeiro nível da árvore. Com essa escolha de organização, toda a estrutura representando os recursos, propriedades e valores, possui no máximo quatro níveis de profundidade. Como conseqüência, a implementação das funções é simplificada e há uma melhora na eficiência, visto que os percursos pelos nós da árvore podem ser realizados rapidamente.

As diversas funções disponibilizadas pela API para RDF do Jena foram estudadas e analisadas e, com base na sua documentação e no estudo da linguagem RDF, um subconjunto delas foi escolhido para ser implementado, porém utilizando a linguagem Lua e a estrutura fornecida pelo Lua XML Parser. Foram implementadas funções para a criação, edição e manipulação de diversos elementos da linguagem, como recursos, propriedades e *statements*, entre outras, além de funções para a leitura e escrita de documentos.

A principal dificuldade encontrada no desenvolvimento deste trabalho foi a mudança de paradigma, ao se passar do paradigma de orientação a objetos em Java para um paradigma procedural em Lua. No Jena, cada classe possui seus próprios métodos. Assim, um método chamado por um recurso no Jena passou, na implementação em Lua, a ser invocado com este recurso na sua lista de parâmetros, para que se pudesse obter os resultados desejados. No entanto, uma mudança de paradigma afeta muito mais do que as chamadas de funções, e tal mudança levou a diversas adequações na implementação das funções ao longo do desenvolvimento da biblioteca.

A Tabela 1 apresenta as funções desenvolvidas para a biblioteca.

4.1. Teste de Usabilidade

Para verificar a usabilidade da biblioteca, foi desenvolvida uma aplicação de teste. A aplicação foi testada utilizando-se um *set-top box* virtual disponibilizado pela PUC-Rio, o Ginga-NCL Virtual Set-top Box, na versão 0.12.1. Além dos arquivos do Lua XML Parser e da API desenvolvida no trabalho foram utilizados os seguintes arquivos:

- Um arquivo *main.ncl;*
- Um script Lua, chamado g*oogle.lua;*
- Um arquivo com informações sobre um domínio, chamado *dominio.rdf;*
- Uma pasta com três vídeos exibidos durante a execução da aplicação.

Tabela 1. Funções implementadas

Nome	Comportamento
createModel	Cria um novo modelo
readModel	Lê um arquivo com dados no formato RDF/XML
writeModel	Escreve modelo na saída padrão
addStatement	Adiciona um *statement* ao modelo
containsStatement	Verifica se um modelo contém um determinado *statement*
containsResourceandProperty	Verifica se um modelo contém um *statement* que possui um recurso e uma propriedade específicos
containsResource	Verifica se um modelo contém um recurso específico
createResource	Cria um novo recurso, dado um URI
createResourceinResource	Cria um novo recurso como sendo uma propriedade de um outro recurso
getResource	Recebe um URI e um modelo e retorna o recurso que possui esse URI
getURI	Retorna o URI de um recurso
listProperties	Retorna um iterador com todas as propriedades de um recurso (em forma de lista de *statements*)
createProperty	Cria uma nova propriedade
addProperty	Adiciona uma propriedade (e o seu valor) a um recurso
getRequiredProperty	Retorna um *statement* no modelo que possui uma propriedade e um recurso específicos
getLocalName	Retorna o *local name* de uma propriedade
getNamespace	Retorna o *namespace* de uma propriedade
listStatements	Organiza todos os *statements* de um modelo em uma lista de *statements*
nextStatement	Recupera o próximo *statement* numa lista de *statements*
getSubject	Retorna o recurso de um *statement*
getPredicate	Retorna a propriedade de um *statement*
getObject	Retorna o valor da propriedade de um *statement*

A aplicação foi desenvolvida tendo como base um exemplo disponibilizado em [PUC-Rio 2011], posteriormente alterado. A aplicação disponibilizada originalmente realiza uma busca na internet, através da televisão, por um ou mais termos fornecidos pelo usuário. Essa aplicação exibe dois campos. No primeiro, o usuário digita os termos

que deseja pesquisar, com o auxílio do controle remoto. Assim que ele pressiona a tecla *Enter*, uma pesquisa é realizada no Google (*http://www.google.com.br*) com os termos digitados pelo usuário. A URL do primeiro resultado encontrado é exibida no segundo campo, como mostra a figura 3.

Nessa aplicação, se o usuário estivesse assistindo algo na TV e quisesse fazer uma pesquisa sobre o conteúdo exibido, ele teria que incluir (por conta própria) no campo de busca os termos relacionados ao que ele estivesse assistindo. Com o uso de metadados, esse cenário se altera.

A aplicação original foi, então, alterada para incluir o uso de metadados. Como no exemplo original havia apenas dois campos, foram incluídos três vídeos para serem exibidos enquanto o usuário digita os termos da busca, assim como metadados relacionados a eles, utilizados pela biblioteca. A aplicação também busca informações relacionadas aos metadados em um domínio, presente no arquivo *dominio.rdf*.

Na aplicação alterada, os três vídeos são exibidos seqüencialmente. Todos eles são descritos por trechos de código em RDF (metadados), presentes no arquivo *main.ncl*. Além disso, associado ao primeiro vídeo, há um arquivo chamado *domínio.rdf*, contendo informações sobre um domínio relacionado ao conteúdo do vídeo.

Durante a exibição dos vídeos, o usuário pode digitar um ou mais termos no primeiro campo utilizando o controle remoto, assim como na aplicação utilizada como base. Quando ele pressiona a tecla *Enter*, uma variável é setada para guardar um valor que determina qual o vídeo exibido no momento. Essa variável é passada para um *script* Lua, que utiliza as funções da biblioteca desenvolvida neste trabalho para capturar os termos relacionados ao vídeo sendo exibido no momento, presentes no arquivo NCL. Se o segundo ou o terceiro vídeo estiverem em exibição quando a tecla *Enter* for pressionada, então serão buscados os termos buscados pelo usuário junto com os termos relacionados ao vídeo exibido.

Se a pesquisa for realizada durante a exibição do primeiro vídeo, no entanto, a aplicação busca no arquivo *domínio.rdf* termos relacionados às informações no arquivo NCL sobre o primeiro vídeo. Os recursos (elemento da linguagem que representa algo a ser descrito) do arquivo são analisados e o recurso *http://exemplo/cleopatra* é encontrado. A palavra *cleopatra* está presente em *main.ncl*, como sendo um objeto (no caso, uma *string*) que descreve o recurso *http://www.example.org/egito*. Dentro do arquivo com o domínio, as informações relacionadas ao recurso *http://exemplo/cleopatra* são *cesar*, *ptolomeu* e *alexandria*. Logo, a busca será realizada com os termos digitados pelo usuário, junto com essas palavras. As figuras 3 e 4 mostram o resultado da execução da aplicação, quando a busca é realizada durante a exibição do primeiro e do segundo vídeo, respectivamente:

Figura 3. Tecla *Enter* pressionada durante a exibição do primeiro vídeo

Figura 4. Tecla *Enter* pressionada durante a exibição do segundo vídeo

Na figura 3, a palavra *aula* foi digitada. Se a busca fosse realizada somente por esse termo, o resultado seria a página *www.aulavaga.com.br/*. Porém, foram buscados os seguintes termos, nessa ordem: *cesar*, *ptolomeu*, *alexandria*, *cleopatra*, *egito* e *aula*, o que retornou a página *www.brasilescola.com/historia/cleopatra.htm*. Já na figura 4, se fosse buscado somente o termo *imperador*, o resultado exibido seria pt.wikipedia.org/wiki/Imperador. Entretanto, a busca foi realizada sobre *roma*, *cesar* e *imperador*.

Por fim, na aplicação foram utilizadas as seguintes funções da biblioteca: a) **listStatements:** lista os *statements* tanto do arquivo *main.ncl* quanto do arquivo *dominio.rdf; b)* **getObject:** recupera os valores das propriedades dos *statements* listados; c) **readModel:** lê o arquivo *dominio.rdf;* d) **getSubject:** recupera os recursos dos *statements* listados; e) **nextStatement:** iterador para percorrer todos os *statements* encontrados

5. Conclusões e trabalhos futuros

Este trabalho apresentou uma biblioteca para o tratamento de metadados seguindo os padrões estabelecidos pelo SBTVD. Essa biblioteca torna possível a leitura e edição de arquivos, criação de modelos, recursos, propriedades e outros elementos. Com a existência de tal ferramenta, os programadores que desenvolvem aplicações para TV Digital não precisam mais de se preocupar com a captura e manipulação de metadados, e sim com a aplicação em si. Outro ponto é que ela pode ser importada em scripts NCLua, sem qualquer necessidade de instalação. A biblioteca encontra-se disponível publicamente no site http://www.lprm.inf.ufes.br. A aplicação de teste exemplificou como as aplicações interativas para TV Digital podem tirar proveito do uso de metadados para apresentar ao telespectador conteúdos mais relevantes de acordo com o que eles buscam. Pode-se conceber várias outras aplicações que se beneficiem do uso de metadados, em tempo real de execução do vídeo, possibilitada por este trabalho. Por exemplo, no contexto de comércio via televisão, imagine que os vídeos exibissem, durante a programação, *merchandizen* de determinados produtos para serem comprados em tempo real pelo telespectador, o que, aliás há muito tempo é comum na TV. Mas suponhamos, agora, que o controle remoto tenha um recurso que permita ao telespectador apontar para determinado produto para procurar informações de preço, características técnicas, entrega etc. e, se interessado, que o permita adquiri-lo. O metadado desses produtos também poderiam ficar embutidos no arquivo NCL dos vídeos, caracterizando cada produto disponível para compra.Como trabalho futuro, pode-se estender a biblioteca, para abarcar mais funções para o controle de metadados em RDF e permitir a criação de operações mais complexas com os elementos da linguagem, tornando-a mais completa. Pode-se, no futuro, desenvolver vários outros tipos de aplicações que utilizam metadados com o auxílio da biblioteca desenvolvida. Além disso, essas aplicações podem ser desenvolvidas utilizando-se como referência o *middleware* brasileiro e o padrão de metadados adotado pelo SBTVD, o RDF.

6. Agradecimentos

Agradecemos à CAPES – Brasil pelo financiamento parcial deste projeto (Projeto RH-TVD #225/2008). Esta pesquisa também foi parcialmente financiada pela FAPES (Números de Processo 47697253 e 52272362), e por uma bolsa de Iniciação Científica PIBIC-UFES (Edital 2010-2011).

Referências

ABNT NBR 15606-2 (2007) “Televisão digital terrestre – Codificação de dados e especificações de transmissões para radiodifusão digital – Parte 2: Ginga-NCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações”

Alan et al. (2009) “Semantic Personalization Framework for Connected Set-Top Box Environment” , In: CENTRIC '09 Proceedings of the 2009 Second International Conference on Advances in Human-Oriented and Personalized Mechanisms, Technologies, and Services

Alves, L. G. P. ; Silva, F. S. ; Bressan, G. (2008) “CollaboraTVware: Uma proposta de Infra-estrutura Ciente de Contexto para Suporte a Participação Colaborativa no Cenário da TV Digital Interativa.” In: XIV Simpósio Brasileiro de Sistemas

Multimídia e Web (WebMedia 2008), 2008, Vila Velha. XIV Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia 2008).

Barbosa e Soares. (2008) "TV digital interativa no Brasil se faz com Ginga: Fundamentos, Padrões, Autoria Declarativa e Usabilidade", In: T. Kowaltowski & K. Breitman (orgs) Atualizações em Informática 2008. XXVIII Congresso da Sociedade Brasileira de Computação. Jornadas de Atualização em Informática (JAI), JAI/SBC 2008. Julho de 2008.

Blanco-Fernández et al. (2009) "Semantic Reasoning and Mashups: An Innovative Approach to Personalized E-Commerce in Digital TV", In: SMAP '09 Proceedings of the 2009 Fourth International Workshop on Semantic Media Adaptation and Personalization.

Ginga 2011 GINGA. *Início | Ginga*. 2011. Disponível em <http://www.ginga.org.br/en/inicio>. Acessado pela última vez em Janeiro de 2012.

Jena 2011 JENA. *Jena Semantic Web Framework*. 2011. Disponível em <http://jena.sourceforge.net/>. Acessado pela última vez em Junho de 2011.

Jr et al. (2009) "Context Information Exchange and Sharing in a Peer-to-Peer Community: a Video Annotation Scenario", In: SIGDOC '09 Proceedings of the 27th ACM International Conference on Design of Communication.

JRDF 2011 JRDF. *JRDF – An RDF Library in Java*. 2011. Disponível em <http://jrdf.sourceforge.net/>. Acessado pela última vez em Junho de 2011.

Lua.org 2011 LUA.ORG. *Lua Authors*. 2011. Disponível em <http://www.lua.org/authors.html>. Acessado pela última vez em Junho de 2011.

Lui et al. (2009) "Ontology Based Content Management for Digital Television Services", In: 2009 IEEE International Conference on E-Business Engineering.

LuaExpat 2011 LUAEXPPAT. *LuaExpat*. 2011. Disponível em < http://matthewwild.co.uk/projects/luaexpat/>. Acessado pela última vez em Junho de 2011.

LuaXML 2011 LUAXML. *Lua XML Parser*. 2011. Disponível em <http://lua-users.org/wiki/LuaXml>. Acessado pela última vez em Junho de 2011.

PUC-Rio 2011 PUC-RIO. *Consulta ao Google*. 2011. Disponível em <http://www.lua.inf.puc-rio.br/~francisco/nclua/tutorial/exemplo_06.html>. Acessado pela última vez em Junho de 2011.

Soares e Castro (2008) "As múltiplas possibilidades do middleware Ginga", Revista de Comunicação e Técnica Audiovisual, São Paulo, SP. p. 76-83.

Soares, L. F. G. ; Rodrigues, R. F. ; Ferreira, M. (2007) "Ginga-NCL: the Declarative Environment of the Brazilian Digital TV System" Journal of the Brazilian Computer Society, v. 12, p. 37-46.

Souza Filho, G. L. ; Leite, Luiz Eduardo Cunha ; BATISTA, C. E. C. F. (2007) "Gnga-J: The Procedural Middleware for the Brazilian Digital TV System" Journal of the Brazilian Computer Society, v. 12, p. 47-56.

Veloso, R. R. (2007) "Java e XML – Processamento de Documentos XML com Java", São Paulo, SP, Brasil: Novatec Editora Ltda