Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Ferramenta para criação dinâmica de testes para Ginga-NCL

Stephania Falcão do Nascimento

JUIZ DE FORA
JULHO, 2015

Ferramenta para criação dinâmica de testes para Ginga-NCL

Stephania Falcão do Nascimento

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Marcelo Ferreira Moreno

JUIZ DE FORA

JULHO, 2015

Ferramenta para criação dinâmica de testes para Ginga-NCL

Stephania Falcão do Nascimento

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Marcelo Ferreira Moreno
Doutor em Informática

Eduardo Barrére
Doutor em Engenharia de Sistemas e Computação

Romualdo Monteiro de Resende Costa
Doutor em Informática

JUIZ DE FORA

05 DE JULHO, 2015

Aos meus amigos, irmão e filha

Aos pais, pelo apoio e sustento.

Resumo

Ao agregarem poder computacional, os dispositivos eletrônicos têm se tornado cada vez mais interativos e um dos serviços que tiram grande proveito dessa evolução é a TV Digital. No caso do Sistema Brasileiro de TV Digital, o componente responsável por esta interatividade é o *middleware* Ginga, uma camada de *software* que provê às aplicações a infraestrutura necessária para executá-las, independentemente do *hardware*. Devido ao processo de conformidade baseado em auto certificação adotado pelo Sistema Brasileiro de TV Digital e o consequente atraso na especificação de uma suíte de testes, implementações comerciais do Ginga têm chegado ao mercado sem estar em conformidade com os padrões ABNT. A única suíte de testes de conformidade disponível, proposta para resolver este problema e evitar o mesmo em Serviços IPTV, é eficaz, porém os casos de teste que a compõem possui estrutura complexa e exige dos criadores de teste demasiado conhecimento do assunto; além disso, esta compreende um repositório colaborativo de testes que dificilmente abrangeriam todas as inúmeras entradas possíveis de teste sobre linguagens baseadas em XML, como a NCL, a linguagem declarativa do Ginga. Este trabalho propõe que os casos de teste para Ginga-NCL sejam criados dinamicamente, em conformidade com os padrões estabelecidos, a fim de torná-los mais abrangentes, através de uma ferramenta de simples uso e entendimento por parte dos criadores de teste, abrangendo os principais elementos da linguagem NCL. Para tanto, a mesma disponibiliza: uma biblioteca de mídias, as quais serão utilizadas na construção do caso de teste; o *download* dos arquivos que o compõe; os resultados da validação do mesmo e o escopo do caso de teste.

Palavras-chave: TV Digital, Interatividade, Ginga, Suíte de testes, Geração Automatizada de Aplicações.

Abstract

By aggregating computing power, electronic devices are becoming more and more interactive and one of the services that take great advantage of this evolution is Digital TV. In the case of Brazilian Digital TV System, the component responsible to its interactivity is the Ginga middleware, a software layer that provides to applications the required infrastructure to execute them independently of the underlying hardware. Due to the compliance process based on auto-certification as adopted by the Brazilian Digital TV System and the delay in specifying a common test suite, Ginga comercial implementations have been reaching market without full conformance to the ABNT standards. The only public test suite, proposed to solve these issues and avoid the same to occur in IPTV services, is an efficient one, though it has a complex structure and demands advanced knowledge from the test creators; besides, it comprises a collaborative repository of test cases that very hardly would cover the countless entries that are possible for an XML based language, like NCL, which is Ginga's declarative language. This work proposes that test cases for Ginga-NCL should be dynamically created, in conformance with the stablished standards, in order to make them more comprehensive, through the use of an authoring tool, which addresses the main elements of NCL. For this purpose, the tool provides: a library of sample media objects to be used to compose the test cases; the download of all files that comprise the test case; the results of validation and the scope of the test case.

Keywords: Digital TV, Interactivity, Ginga, Test suite, Automated Application Generation.

Agradecimentos

Aos meus pais, pela paciência e apoio.

Ao professor Marcelo F. Moreno pela orientação, amizade, confiança e principalmente, pela paciência, sem a qual este trabalho não se realizaria. Aos professores Eduardo Barrére e Romualdo Monteiro de Resende Costa, que cederam precioso tempo para a avaliação deste trabalho.

Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

Aos amigos que me acompanharam durante todos esses anos.

"Às vezes, pessoas que nem imaginamos
são capazes de realizar algo surpreendente".
Alan Turing (O jogo da imitação)

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

DCC	Departamento de Ciência da Computação
UFJF	Universidade Federal de Juiz de Fora
NCL	*Nested Context Language*
XML	*eXtensible Markup Language*
PUC-Rio	Pontifícia Universidade Católica do Rio de Janeiro
IPTV	*Internet Protocol Television*
ABNT	Associação Brasileira de Normas Técnicas
UIT	União Internacional de Telecomunicações
RTP	*Real-time Transport Protocol*
AAC	*Advanced Audio Coding*
UFPB	Universidade Federal da Paraíba
USB	*Universal Serial Bus*
SBTVD-T	Sistema Brasileiro de Televisão Digital Terrestre
ISDB-T	*Integrated Services Digital Broadcasting Terrestrial*
NCM	*Nested Context Model*
MPEG	*Moving Picture Experts Group*
HTML	*HyperText Markup Language*
PHP	*Hypertext Preprocessor*
SQL	*Structured Query Language*
DOM	*Document Object Model*
CSS	*Cascading Style Sheets*
Web	*World Wide Web*
URL	*Uniform Resource Locator*

1 Introdução

Sabe-se que, hoje em dia, a grande maioria dos dispositivos eletrônicos vem utilizando a computação de forma a torná-los cada vez mais interativos e conectados. Como exemplo notável, pode-se citar a TV Digital, que adquiriu enorme vantagem com esta junção, em diversos aspectos, notadamente na introdução da interatividade.

A TV digital interativa permite ao usuário um maior controle do que está sendo assistido, uma vez que cria a possibilidade do mesmo interagir com a informação que está sendo exibida, e dessa forma aprimora sua experiência ao assistir televisão.

O grande desafio trazido pela interatividade consiste na adequação de conteúdo por parte das emissoras de TV para atender os diversos públicos existentes, cada qual com sua cultura e contrastes sociais marcantes (Brackmann, 2010).

A interatividade é possível graças à existência de um *middleware*, uma camada intermediária de *software* que vem integrada na maioria dos modelos de equipamentos de recepção de sinal de TV Digital e que provê às aplicações a infraestrutura necessária para executá-las, independentemente do *hardware*. O *middleware* padrão do Sistema Brasileiro de TV Digital(SBTVD) é denominado Ginga (Brackmann, 2008).

As aplicações declarativas interativas para o Ginga são escritas em *Nested Context Language*(NCL), que tem como subsistema de suporte o Ginga-NCL. A linguagem NCL se baseia no modelo conceitual Nested Context Model(NCM) e permite construir aplicações com sincronismo temporal e espacial entre mídias, podendo ser executadas em múltiplos dispositivos de exibição, bem como ser editadas/produzidas ao vivo (Brackmann, 2008).

As implementações comerciais do Ginga-NCL, assim como qualquer produto concebido para interoperar com outros, devem passar por testes que atestem que as mesmas estejam de acordo com certos padrões, assegurando sua qualidade, confiabilidade e interoperabilidade. Tais testes são chamados testes de conformidade.

Um caso de teste de conformidade deve ser elaborado de forma a identificar erros em trechos da implementação diante de uma especificação. Para as implementações Ginga-NCL, os casos de teste devem verificar a conformidade a cada assertiva encontrada

na Recomendação UIT-T H.761 ITU-T (2011), nas Normas ABNT NBR 15606-2 (2007) e Guia operacional do ISDB-TB ABNT NBR 15606-7 (2011). Para tanto, é preciso que o caso de teste contenha a referência ao que deve ser inspecionado, bem como a definição de pré condições, como se deve proceder a execução do teste, dados de entrada e saída(resultado esperado) (Araújo, 2011).

A elaboração de casos de teste para linguagens declarativas, como o NCL, é complexa pois, ao contrário das linguagens imperativas, os construtores da linguagem possuem interdependências. Isto quer dizer que os mesmos devem ser testados de forma integrada (Araújo, 2011).

A um agrupamento de casos de teste dá-se o nome de suíte de testes, que pode conter instruções detalhadas para cada coleção de casos de teste, além da descrição da configuração do sistema usado.

A especificação e implementação de uma suíte de testes comum e padronizada para Ginga-NCL sofreu um atraso devido a auto certificação de produtos adotada pelo Sistema Brasileiro de TV Digital. A auto certificação permite que os próprios fabricantes, por si só, garantam a qualidade dos produtos existentes e com isso, cada um costuma usar sua própria suíte de testes. Por consequência muitas dessas implementações chegam ao mercado sem necessariamente ter passado por testes de conformidade rigorosos com as normas mencionadas anteriormente (Araújo, 2011).

A suíte de testes proposta por Araújo (2011) para a UIT-T foi criada visando poupar serviços IPTV do mesmo problema. Um caso de teste a ser desenvolvido para a suite de testes UIT-T possui estrutura complexa e exige um aprofundado conhecimento do assunto, impossibilitando que programadores inexperientes, mas conhecedores das normas e de suas funcionalidades, colaborem com seus casos de teste.

Outro ponto importante, é que aquela é a única suíte de testes publicamente disponível para o Ginga-NCL, e é em essência um repositório colaborativo de testes, mantido pela UIT-T em parceria com a PUC-Rio e UFJF. Um repositório de testes pode não ser suficiente para abranger todos os casos, devido a vasta gama de entradas possíveis para testes sobre linguagens baseadas em XML, como a NCL, a linguagem declarativa do Ginga (Araújo, 2011).

Percebe-se que a criação de testes poderia ser mais simples, não exigindo um conhecimento avançado do assunto, e os casos de teste poderiam se tornar mais abrangentes, se criados dinamicamente e em nível de abstração mais elevado.

Assim como o atual repositório de testes do Ginga-NCL se encontra *online*, o mesmo poderia ser feito com uma suíte dinâmica de testes, para abranger o maior número de criadores de casos de teste possível.

1.1 Objetivos

Pode-se apontar como objetivo geral do estudo, a proposta de soluções práticas para um serviço de criação dinâmica de casos de testes para o *middleware* Ginga-NCL.

Como objetivo específico, o presente trabalho visa identificar técnicas que possam ser aplicadas para a geração automatizada de aplicações NCL, a partir de especificações em alto nível das características desejáveis para cada caso de teste gerado.

Logo, busca-se uma forma de organizar elementos e atributos da linguagem NCL a fim de facilitar a criação de trechos reutilizáveis de código para a composição de casos de testes. Para tanto, visa-se identificar trechos comuns, ou seja, agrupamentos e características prioritárias, a várias aplicações NCL.

Com o levantamento dessas informações, propõe-se desenvolver uma ferramenta de autoria a ser disponibilizada na *Web*, de forma a permitir aos usuários a especificação, em alto nível, de casos de testes para Ginga NCL, a partir da enumeração das características desejadas como alvo de testes. Os casos serão então criados de forma automatizada, dinamicamente, incluindo objetos de mídia inicialmente armazenados em uma biblioteca de mídias da ferramenta. Nesta monografia, entende-se como usuário da ferramenta de autoria o criador de teste não especializado em programação NCL, porém conhecedor das normas que especificam o Ginga-NCL.

2 Fundamentação Teórica

2.1 Sistema Brasileiro de Televisão Digital Terrestre

O Sistema Brasileiro de Televisão Digital Terrestre (SBTVD-T) é um padrão técnico para radiodifusão digital e foi instituído no dia 26 de novembro de 2003, de acordo com o Decreto Nº 4.901 (Montez, 2005).

Três sistemas de TV Digital foram escolhidos para analíse de desempenho em solo brasileiro: o americano, o europeu e o japonês. Dentre esses, o escolhido para servir de base ao sistema brasileiro, foi o japonês (ISDB-T). O sistema se destacou pela qualidade de recepção de sinal em qualquer ambiente e suporte a recepção e acesso a serviços tanto em terminais móveis quanto fixos (Montez, 2005).

Em comparação ao sistema japonês, o sistema brasileiro apresenta algumas diferenças no que diz respeito ao formato da compressão de vídeo(substituição do MPEG-2 pelo MPEG-4/H.264) e o *middleware* utilizado. Por ser uma especificação mais moderna, o SBTVD-T tem sido adotado em vários países.

Na figura 2.1 podem ser vistos os padrões que compõem o SBTVD-T.

2.2 Middleware Ginga

Ginga é o *middleware*, desenvolvido pela Pontifícia Universidade Católica do Rio de Janeiro(Puc-Rio) em parceria com a Universidade Federal da Paraíba(UFPB), responsável por prover as condições necessárias para a interatividade no Sistema Brasileiro de Televisão Digital.

A especificação é aberta(*open source*), uma vez que foi concebida segundo os preceitos de inclusão social/digital e a obrigação de compartilhamento de conhecimento de forma livre (Brackmann, 2008).

Atualmente, ele é considerado o *middleware* mais avançado do mercado, devido a tecnologias e inovações brasileiras que o tornaram um padrão mundial reconhecido pela

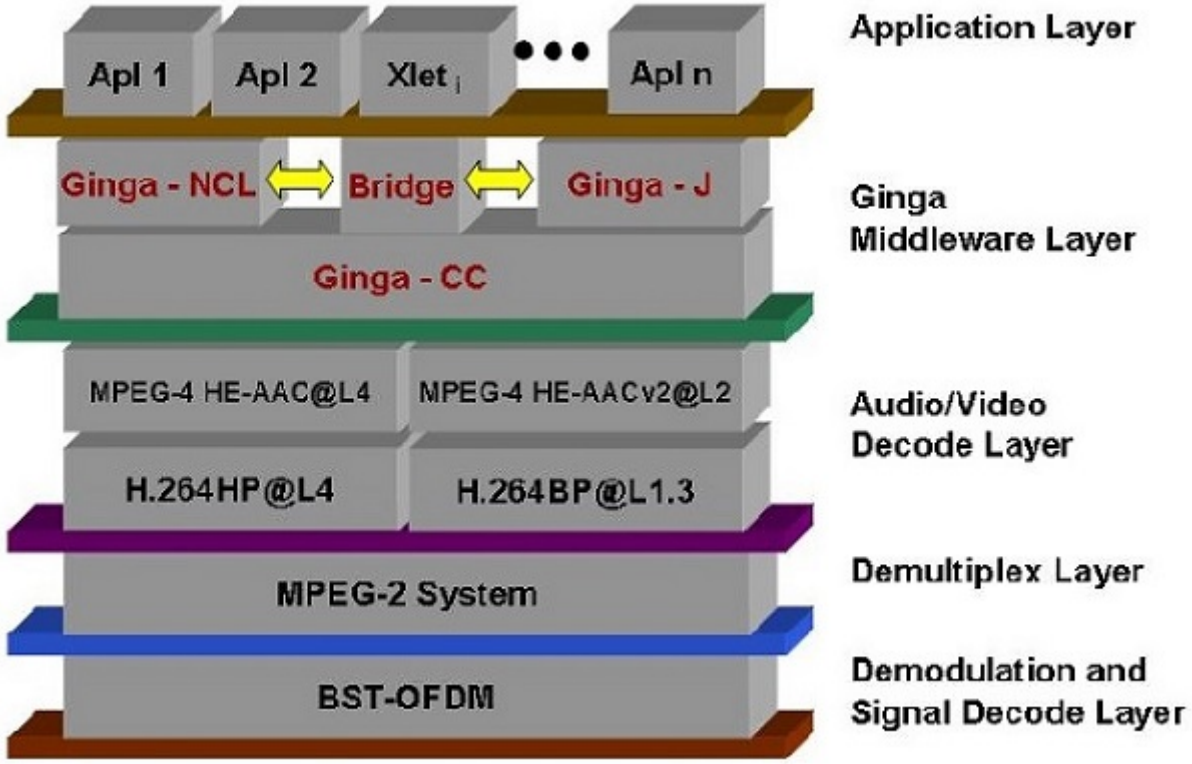

Figura 2.1: Camadas do Sistema Brasileiro de TV Digital(Brackmann, 2008)

União Internacional de Telecomunicações (UIT-T) (Brackmann, 2008).

O *middleware* Ginga pode ser dividido em três subsistemas principais: Ginga-NCL, Ginga-J e Ginga-CC (Brackmann, 2008).

Como mencionado anteriormente, Ginga-NCL é o subsistema Ginga responsável por executar aplicações escritas em NCL, oferecendo a infraestrutura necessária. Lua é a linguagem imperativa de suporte do NCL.

Por sua vez, Ginga-J é o subsistema responsável por executar aplicações baseadas na linguagem Java.

Já Ginga-CC é reponsável por prover suporte aos dois subsistemas mencionados acima, além de controlar acessos a camada de rede, meios de obtenção de conteúdo, entre outros (Brackmann, 2008).

Na figura 2.2 é mostrada a arquitetura do *middleware* Ginga.

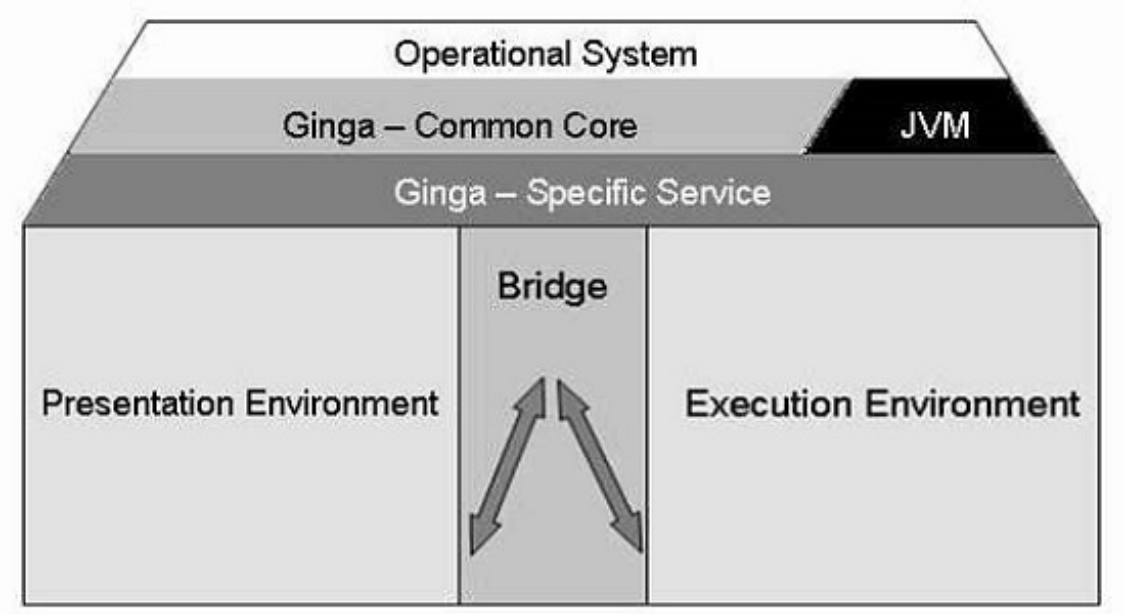

Figura 2.2: Arquitetura do *middleware* Ginga(Soares, 2007)

2.3 Nested Context Language

A linguagem declarativa NCL, baseada no modelo conceitual NCM, caracteriza-se por ser uma linguagem de marcação e sua estrutura se assemelha a da linguagem HTML (Ratamero, 2007).

A estrutura básica de um arquivo NCL é:

- Cabeçalho do arquivo XML;
- Cabeçalho do programa:
  - Base de regiões;
  - Base de descritores;
  - Base de conectores
- Corpo do programa:
  - Nó de conteúdo;
  - Nó de contexto;
  - Âncoras.

No cabeçalho do arquivo XML são definidas a versão do XML bem como a codificação usada, de forma a definir o tipo de caracteres que serão exibidos na tela, além da tag que marca o início do programa (Ratamero, 2007).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <ncl xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile" id="nclEx">
```

Figura 2.3: Exemplo de cabeçalho do documento NCL (Soares, 2009)

Já no cabeçalho do programa, encontram-se a definição das regiões, descritores e conectores.

Regiões definem onde as mídias ou nós de conteúdo serão exibidas. Na linguagem NCL as regiões são definidas pela tag <*region*> e se encontram na base de regiões. A base de regiões inicia-se com a tag <*regionBase*> e finaliza com a tag </*regionBase*> (Ratamero, 2007).

A figura 2.4 ilustra a estrutura da base de regiões e exemplifica a definição de algumas.

```
<regionBase>
    <region id="bottomRightReg" zIndex="2" height="45%" width="50%"
        bottom="5%" right="0"/>
    <region id="topRightReg" zIndex="2" height="45%" width="50%" right="0"
        top="5%"/>
    <region id="bottomLeftReg" zIndex="2" height="45%" width="50%" bottom="5%"
        left="0"/>
</regionBase>
```

Figura 2.4: Exemplo de base de regiões (Soares, 2009)

O atributo id identifica unicamente o elemento <*region*> e é usado na definição dos descritores. O atributo *z-index* define a sobreposição de regiões. Nesse sentido, regiões com *z-index* maiores, sobrepõem as com *z-index* menores. Os atributos *height* e *width* definem a área da região. Já os atributos *bottom*, *top*, *right* e *left* definem o posicionamento da região (NCLHandbook, 2012).

Descritores associam as mídias às regiões previamente definidas no <*regionBase*>, além de poder definir comportamentos adicionais para as mesmas. Na linguagem NCL os descritores são definidos pela tag <*descriptor*> e se encontram na base de descritores. A base de descritores inicia-se com a tag <*descriptorBase*> e finaliza com a tag

</*descriptorBase*> (Ratamero, 2007).

A figura 2.5 ilustra a estrutura da base de descritores e exemplifica a definição de alguns.

```
<descriptorBase>
  - <descriptor id="audiomp3Desc" explicitDur="45">
        <descriptorParam value="50%" name="soundLevel"/>
    </descriptor>
    <descriptor id="imagejpgDesc" moveDown="videomp4Idx" moveRight="imagegifIdx"
        moveLeft="imagegifIdx" moveUp="imagepngIdx" focusIndex="imagejpgIdx"
        region="bottomRightReg"/>
    <descriptor id="imagepngDesc" focusIndex="imagepngIdx" region="bottomLeftReg"/>
    <descriptor id="imagegifDesc" focusIndex="imagegifIdx" region="bottomLeftReg"/>
    <descriptor id="videomp4Desc" focusIndex="videomp4Idx" region="bottomLeftReg"/>
</descriptorBase>
```

Figura 2.5: Exemplo de base de descritores (Soares, 2009)

O atributo id identifica unicamente o elemento <*descriptor*> e é usado na definição das mídias. Para atribuir uma duração explicíta a mídia, é usado o atributo *explicitDur*. Já o atributo *focusIndex* é usado para referenciar uma mídia na navegação por teclas. Os atributos *moveUp*, *moveRight*, *moveDown* e *moveLeft*, contêm o *focusIndex* da mídia que ficará em foco quando as respectivas teclas forem pressionadas: "seta para cima", "seta para direita", "seta para baixo"e "seta para esquerda". Para a manipulação do volume da mídia é usado o elemento <*descriptorParam*>, que tem como atributos *name* e *value*. O atributo *name* contém a propriedade da mídia a ser manipulada, no caso *soundLevel*, e o atributo *value* contém um valor válido inicial, que para a mesma propriedade, deve estar entre 0 e 100 (NCLHandbook, 2012). Conectores associam condições a ações. Tais associações acontecem devido a conectores causais, que na linguagem NCL, são definidos pela tag <*causalConnector*>. Uma condição pode estar associada a uma ou mais ações. As ações estão vinculadas a tag <*simpleAction*>, e no caso de mais ações associadas a uma condição, as mesmas estarão contidas na tag <*compoundAction*>. Todos os elementos citados compõem a base de conectores, que inicia-se com a tag <*connectorBase*> e finaliza com a tag </*connectorBase*> (Ratamero, 2007).

A figura 2.6 ilustra a estrutura da base de conectores e exemplifica a definição de ações simples e compostas.

O atributo id identifica unicamente o elemento <*casualConnector*> e é usado na

```
<connectorBase>
  - <causalConnector id="OnBeginStopNStartN">
        <simpleCondition role="OnBegin"/>
      - <compoundAction operator="seq">
            <simpleAction role="stop" qualifier="par" max="unbounded"/>
            <simpleAction role="start" qualifier="par" max="unbounded"/>
        </compoundAction>
    </causalConnector>
  - <causalConnector id="OnBeginStartN">
        <simpleCondition role="OnBegin"/>
        <simpleAction role="start" qualifier="par" max="unbounded"/>
    </causalConnector>
</connectorBase>
```

Figura 2.6: Exemplo de base de conectores (Soares, 2009)

definição dos links. O atributo *role*, está presente em <*simpleCondition*> e <*simpleAction*>. No elemento <*simpleCondition*> o mesmo contém a condição, já no <*simpleAction*> contém a ação. O atributo *qualifier* do elemento <*simpleAction*>, assim como o atributo *operator* do elemento <*compoundAction*>, definem se as ações serão executadas paralelamente(par) ou sequencialmente(seq). Por fim, temos o atributo max do elemento <*simpleAction*>, que define o número máximo de elementos que realizarão a ação.

No corpo do programa estão contidos os nós, portas, links ou elos, âncoras, etc.

Como dito anteriormente, os nós de conteúdo correspondem as mídias da aplicação que podem ser vídeos, imagens, áudio, etc.

```
<media id="audiomp3" descriptor="audiomp3Desc" src="media/audio.mp3"/>
<media id="imagejpg" descriptor="imagejpgDesc" src="media/image.jpg"/>
<media id="imagepng" descriptor="imagepngDesc" src="media/image.png"/>
<media id="imagegif" descriptor="imagegifDesc" src="media/image.gif"/>
<media id="videomp4" descriptor="videomp4Desc" src="media/video.mp4"/>
```

Figura 2.7: Exemplo de definição de mídias (Soares, 2009)

O atributo id identifica unicamente o elemento <*media*> no documento NCL, já o atributo src indica a localização da mesma. Através do atributo *descriptor*, que contém o id do descritor, faz-se a associação da mídia à região.

O nó de contexto corresponde a portas, elos(links) e âncoras.

A porta, como próprio nome diz, permite o acesso de um elemento externo ao conteúdo de um contexto. Inicia-se com a tag <*port*> e finaliza com a tag </> (Ratamero, 2007).

```
<port id="entry" component="imagejpg"/>
```

Figura 2.8: Exemplo de definição de porta (Soares, 2009)

O atributo id identifica unicamente o elemento <*port*> no documento NCL, já o atributo *component* contém o id do nó de entrada para o contexto (NCLHandbook, 2012).

Os elos(links), associados com os conectores definidos no <*connectorBase*>, são os responsáveis pelo sincronismo de eventos em uma aplicação NCL. Inicia-se com a tag <*link*> e finaliza com a tag </*link*> (Ratamero, 2007).

```
<link id="lAudiomp3" xconnector="OnBeginStopNStartN">
    <bind role="OnBegin" component="audiomp3"/>
    <bind role="stop" component="audiomp3"/>
    <bind role="stop" component="imagepng"/>
    <bind role="start" component="imagepng"/>
</link>
```

Figura 2.9: Exemplo de definição de links (Soares, 2009)

O atributo id identifica unicamente o elemento <*link*> no documento NCL, já o atributo *xconnector* contém o id do conector ao qual o *link* está associado. Os atributos *role* e *component* do elemento <*bind*> contêm, respectivamente, a condição ou ação e o id do nó (NCLHandbook, 2012).

Por último , não menos importante, estão as âncoras. As âncoras são encontradas dentro dos nós e podem servir tanto para ativar *links* quanto para a manipulação de propriedades dos nós (Ratamero, 2007).

Segundo Ratamero (2007), dois tipos de âncoras podem ser encontrados numa aplicação NCL: âncoras de conteúdo e de atributo.

A âncora de conteúdo, contida no elemento <*media*>, corresponde a tag <*area*> e é responsável pela ativação de links. A âncora de atributo define os atributos dos nós que podem ser manipulados por outros elementos, e corresponde a tag <*property*>.

Ressalta-se que os elementos referidos ao longo deste trecho possuem outros atributos e propriedades, mas não foram abordados por não serem relevantes ao atual contexto do trabalho.

2.4 Trabalhos Relacionados

A única suíte de testes para o Ginga-NCL disponível publicamente foi desenvolvida por Araújo (2011) para a UIT-T e tem por objetivo prover aos criadores de teste a metodologia para a estruturação dos casos de teste para Ginga-NCL, bem como o acesso aos casos de teste existentes no repositório e, ainda, a possibilidade de contribuírem com os seus próprios casos de teste.

Na suíte de testes UIT-T é possível encontrar uma lista de assertivas que foram retiradas das normas que especificam o Ginga-NCL. Tais assertivas encontram-se organizadas por elementos NCL, e cada qual contêm regras, condições e alvos dos casos de teste relativas ao mesmo. Também é possível acessar as instruções para a construção do caso de teste, que foi inspirado em um formato padrão IEEE Std 829 (1998) com algumas adaptações para linguagens baseadas em XML. A estrutura de um caso de teste de conformidade para o Ginga-NCL, requer:

- o identificador da assertiva, que indica a assertiva a ser considerada;
- a Seção de Recomendação UIT-T H.761 do ITU-T (2011), que especifica o formato padrão da assertiva;
- a assertiva, que contêm as características intrínsecas ao elemento;
- o *Validation Type*, que define o retorno do caso de teste em questão: Falha ou Sucesso;
- o *Target*, que define mais especificadamente qual elemento dentro da assertiva que será testado;
- o *Prescription Level*, que define se a assertiva será testada ou não;
- o Identificador da instrução, identifica a instrução;
- a Instrução, que determina os passos a serem seguidos durante a execução do teste;
- o documento NCL a ser testado.

Para que o caso de teste seja válido, o documento em questão deve conter além do código habitual em NCL, a identificação do caso de teste, bem como a descrição do comportamento esperado de saída e a identificação do autor do caso de teste.

Percebe-se que a estrutura é bastante complexa, e exige experiência de programação NCL por parte dos criadores de casos de teste. Logo, torna-se interessante oferecer a possibilidade dos mesmos serem criados como nos objetivos do presente trabalho. Pode-se destacar a lista de assertivas contida na suíte de testes UIT-T como algo a ser aproveitado na construção do caso de teste por meio da ferramenta de autoria proposta neste trabalho.

3 Solução Proposta

3.1 Requisitos

Alguns requisitos puderam ser identificados para a ferramenta de criação de testes proposta neste trabalho, à partir dos objetivos apresentados no capítulo 1.

- Requisitos Funcionais:
  - o sistema deve permitir a criação de casos de testes para Ginga-NCL;
  - o sistema deve oferecer uma biblioteca de mídias que serão usadas para compor o caso de teste;
  - o sistema deve oferecer abstrações gráficas em alto nível ao criador dos casos de teste;
  - o sistema deve construir o caso de teste a partir das escolhas do criador dos casos de teste
  - o sistema deve permitir o download de arquivos que compõem o caso de teste resultante pelo criador dos casos de testes.
- Requisitos Não Funcionais:
  - o sistema deve se comunicar com o banco de dados;
  - o sistema deve ser disponibilizado *online*;
  - o sistema deve ser de fácil manipulção por parte do criador dos casos de teste;
  - o sistema deve garantir o encapsulamento do código NCL durante sua construção;

3.2 Arquitetura

O padrão de arquitetura MVC ou *Model-View-Controller*, foi concebido no início dos anos 80 e se tornou um dos mais conhecidos, sendo utilizado em *frameworks* de inúmeras

linguagens orientadas a objetos. Na camada *View* encontram-se as telas do sistema, o que será visualizado pelo usuário. Para camada *Model* são enviados os dados do sistema que devem ser persistidos, ou seja, é através desta que tem-se acesso ao banco de dados. A camada *Controller* controla o acesso as camadas mencionadas acima.

Segundo Luciano (2011), a grande vantagem no uso desta arquitetura é a independência proporcionada aos componentes, o que tem o efeito de facilitar a manutenção do sistema, no sentido de que na hipotése da necessidade de alterações, assim como melhorias, a localização do ponto desejado é facilmente realizada.

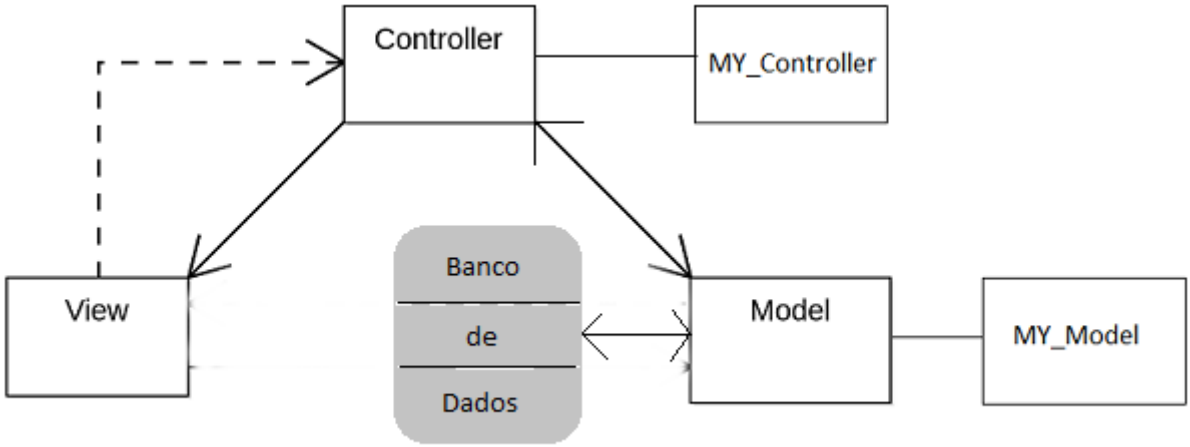

Figura 3.1: Arquitetura MVC da ferramenta proposta

Nota-se que a *View* não acessa diretamente o *Model*, nem este a *View*. Ambas se comunicam por meio do *Controller*. Basicamente, a *View* efetua uma requisição ao *Controller* através da chamada de uma função do mesmo. O *Controller*, caso necessário, segue o mesmo procedimento para acessar o *Model*. O *Model* se comunica com o banco de dados e, ao final do processo, devolve a resposta obtida ao *Controller* que o encaminha corretamente a *View*.

Pode-se observar a existência de dois elementos adicionais acoplados ao *Controller* e ao *Model*: o *MY_Controller* e o *MY_Model*. O *MY_Controller* contêm arquivos (*javascript* e css) que são comuns as *Views*, como também o carregamento dos *models* integrantes do sistema. Neste contexto, cada *model* corresponde a uma tabela do banco, que será apresentado em detalhes ainda nesta seção. Já o *MY_Model* contêm funções que realizam operações básicas no banco de dados, como consultas, inserção, etc. Os *controllers* e

models estendem, respectivamente, *MY_Controller* e *MY_Model*, ou seja, herdam todas as funções e atributos destas classes.

3.3 Projeto de interface

Projeto de interface visa facilitar a interação do usuário com o sistema, criando um ambiente de fácil visualização e de uso intuitivo. De acordo com Mandel (1997), existem três regras de ouro para esse fim:

1. Deixar o usuário no comando: permitir que o mesmo decida o que ele quer e como quer fazer, sem exigir conhecimentos computacionais específicos

2. Reduzir a carga de memória do usuário: projetar a interface de forma a deixá-la bem intuitiva e organizada. Além disso, o sistema deve guardar informações previamente inseridas pelo usuário de forma a disponibilizá-las facilmente, quando necessário

3. Tornar a interface consistente: manter o padrão de interface ao longo da aplicação, bem como manter o usuário a par das opções disponíveis

Para que a interface da ferramenta de autoria proposta pelo presente trabalho atendesse aos críterios apresentados acima, a mesma se baseou nas quatro perguntas fundamentais no desenvolvimento de uma aplicação NCL. São elas: o que? (*what*), onde? (*where*), como? (*how*) e quando? (*when*)(Soares, 2009).

A pergunta o que(*what*), corresponde as mídias; onde(*where*) corresponde às regiões; como(*how*) corresponde aos descritores e quando(*when*) corresponde aos *links* e conectores. Percebe-se que com o uso das perguntas, abrange-se os principais elementos da linguagem NCL de forma intuitiva. A interface e as tecnologias utilizadas em sua implementação são apresentadas no capítulo 4.

Nas figuras 3.2, 3.3, 3.4 e 3.5 são mostrados os esboços das telas da ferramenta.

3.4 Escopo possível dos casos de teste

O escopo dos casos de teste corresponde aos elementos e atributos da linguagem NCL cobertos pelo protótipo desenvolvido. Tais elementos e atributos são apresentados na

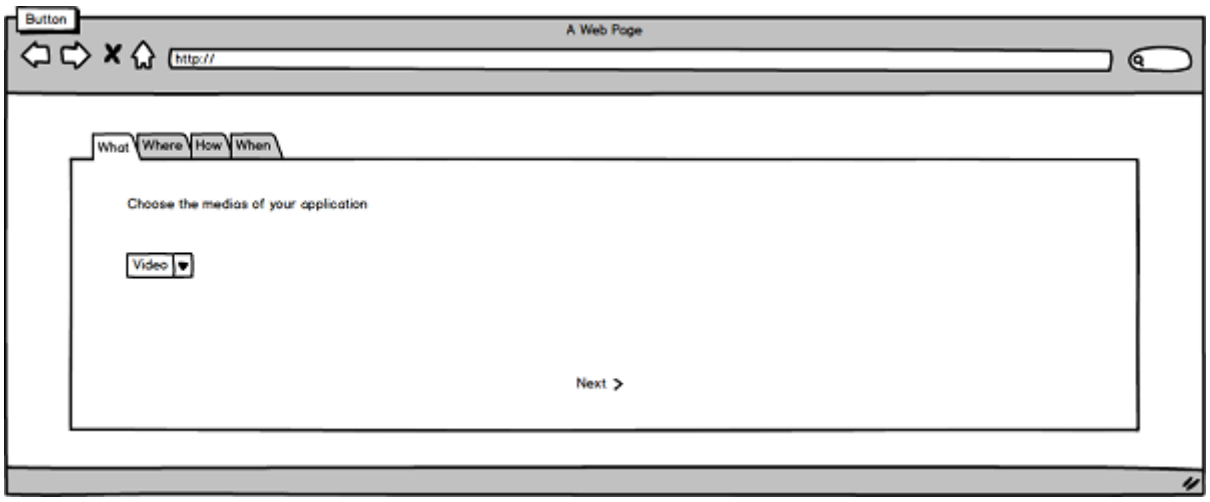

Figura 3.2: Esboço da tela de mídias

tabela 3.1.

Descrições sobre esses elementos e atributos podem ser encontrados no capítulo 2, onde abordou-se a linguagem NCL.

3.5 Banco de dados

O banco de dados foi modelado a fim de abranger os elementos apresentados na seção anterior e seus atributos, como também permitir que o mesmo seja expandido, sem dificuldades. O modelo do banco de dados é mostrado na figura 3.6.

Nota-se a existência de 10 tabelas: ncl, *media*, *port*, *descriptor*, *navigation*, *condition*, *action*, *link*, *link_has_media* e *action_has_media*. As tabelas *media*, *condition* e *action*, contêm valores iniciais necessários para a contrução do caso de teste, e serão explicitados ao longo desta Seç ao. Ressalta-se que os identificadores contidos nas tabelas não correspondem ao atributo id dos elementos NCL.

Na tabela ncl são armazenados os identificadores dos casos de teste. O campo idncl corresponde ao identificador do caso de teste e está contido, além desta, nas tabelas *port*, *link* e *descriptor*. Com esse identificador é possível recuperar todos os dados de um caso de teste. Já o campo ncl é usado apenas para que o identificador do caso de teste seja gravado no banco.

Na tabela *media* encontram-se armazenados os dados relativos às mídias contidas

na biblioteca: arquivos de áudio, nos formatos AAC e MP3; vídeos nos formatos MPG e MP4 e imagens nos formatos PNG, JPEG e GIF. O campo *idmedia* corresponde ao identificador da mídia e é encontrado nas tabelas *descriptor*, *port*, *link* e *action_has_media*. O nome genérico da mídia encontra-se no campo *media_name*. Nesse sentido, este último pode assumir os valores áudio, vídeo ou imagem. É possível que haja valores duplicados neste campo, uma vez que, cada mídia contida na biblioteca com seu respectivo formato é considerada diferente. Os formato das mídias encontram-se no campo *media_extension* e a duração das mesmas em *media_duration*. A duração da mídia é utilizada como limite superior para o atributo *explicitDur* do elemento *descriptor*, como será mostrado no capítulo 4.

Nas tabelas *descriptor* e *navigation* são armazenados os dados relativos ao descritor. O identificador do *descriptor* corresponde ao *iddescriptor*, que pode ser encontrado também na tabela *navigation*. Com esse identificador é possível recuperar as informações relativas à navegação por teclas. A duração explícita definida para uma mídia encontra-se armazenada no campo *descriptor_explicit_dur*. O volume, correspondente a propriedade *soundLevel* do elemento media encontra-se no campo *descriptor_sound_level*. Como mostrado no capítulo 2, essa propriedade pode ser manipulada pelo descritor através do *descriptorParam*. O campo *region_name* contêm a região onde a mídia referenciada pelo *media_idmedia* encontra-se. A tabela *navigation* tem como identificador o campo *idnavigation*, e nela estão contidos os dados relativos a navegação por teclas do descritor. Os dados do campo *navigation_position* correspondem aos atributos *moveLeft*, *moveRight*, *moveUp* e *moveDown*, e armazena exatamente esses valores. O *focusIndex* da mídia que ficará em foco encontra-se no campo *navigation_focus_index*.

A tabela *port* contêm a referência a mídia(*media_idmedia*) pela qual o contexto é acessado por elementos externos. O campo *idport* corresponde ao identificador da porta.

Na tabela *condition* encontram-se armazenadas as condições pré definidas: *OnBegin*, *OnEnd* e *OnSelection*. Essas condições estão contidas no campo *condition_name*. O identificador da condição(*idcondition*), bem como o da mídia, são encontrados na tabela *link*.

Na tabela *action* encontram-se armazenadas as ações pré definidas: *Stop* e *Start*.

Essa ações estão contidas no campo *action_name*. O identificador da ação(*idaction*) é encontrado na tabela *link_has_action*.

Nas tabelas *link*, *link_has_action* e *action_has_media* são armazenados os dados de sincronismo do caso de teste.

Na tabela *link* é feita a associação da condição à mídia que dispara os eventos, através de seus identificadores. A associação de um *link*(*idlink*) a uma ou mais ações é feita na tabela *link_has_action*. As mídias relacionadas a cada ação são referenciadas na tabela *action_has_media*.

3.6 Metodologia de teste

A metodologia que deve ser seguida pelo criador de caso de teste é a seguinte:

1. Estudar as normas ABNT NBR 15606-2 (2007), ABNT NBR 15606-5 (2008) e ABNT NBR 15606-7 (2011); e a Recomendação UIT-T H.761 ITU-T (2011)

2. Fazer o levantamento das assertivas a serem endereçadas pelo caso de teste a ser criado

3. Estruturar os alvos dos testes segundo a especificação das assertivas

4. Utilizar a ferramenta de autoria para a criação do caso de teste

O conhecimento das normas que especificam o Ginga-NCL é de fundamental importância para a construção de bons casos de teste, uma vez que elas contêm as regras e condições inerentes a cada elemento NCL. A essas regras e condições dá-se o nome de assertivas. Através das assertivas é possível definir os alvos dos testes, ou seja, o que será efetivamente testado. Nesse sentido a consulta à lista de assertivas da atual suíte de testes UIT-T pode ser interessante. Como mencionado no capítulo 2, as assertivas contidas na lista foram retiradas das normas e encontram-se organizadas por elementos NCL. Cada assertiva contêm o alvo do teste do elemento em questão. Com todos os alvos dos testes definidos, basta utilizar a ferramenta para a criação do caso de teste.

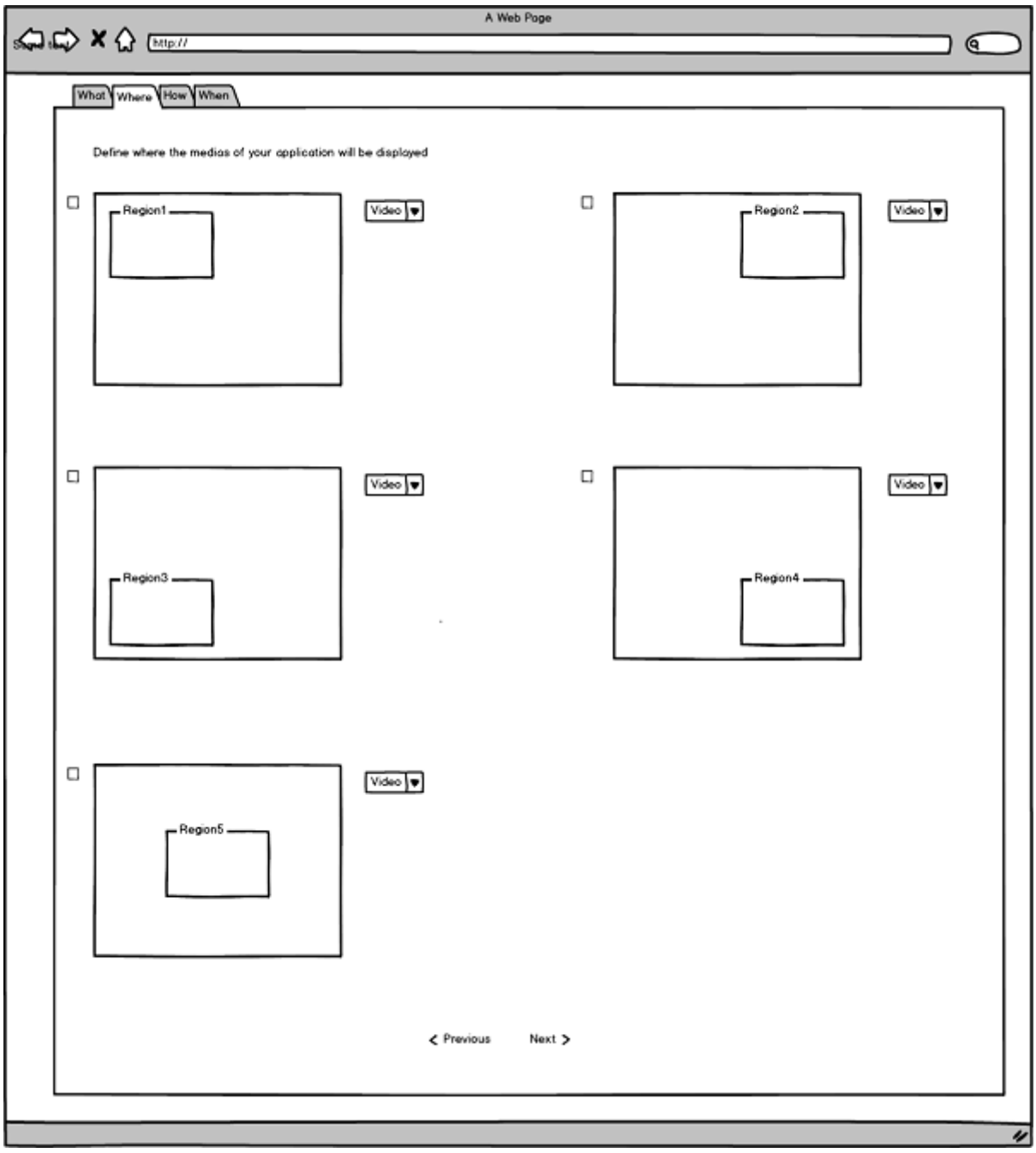

Figura 3.3: Esboço da tela de regiões

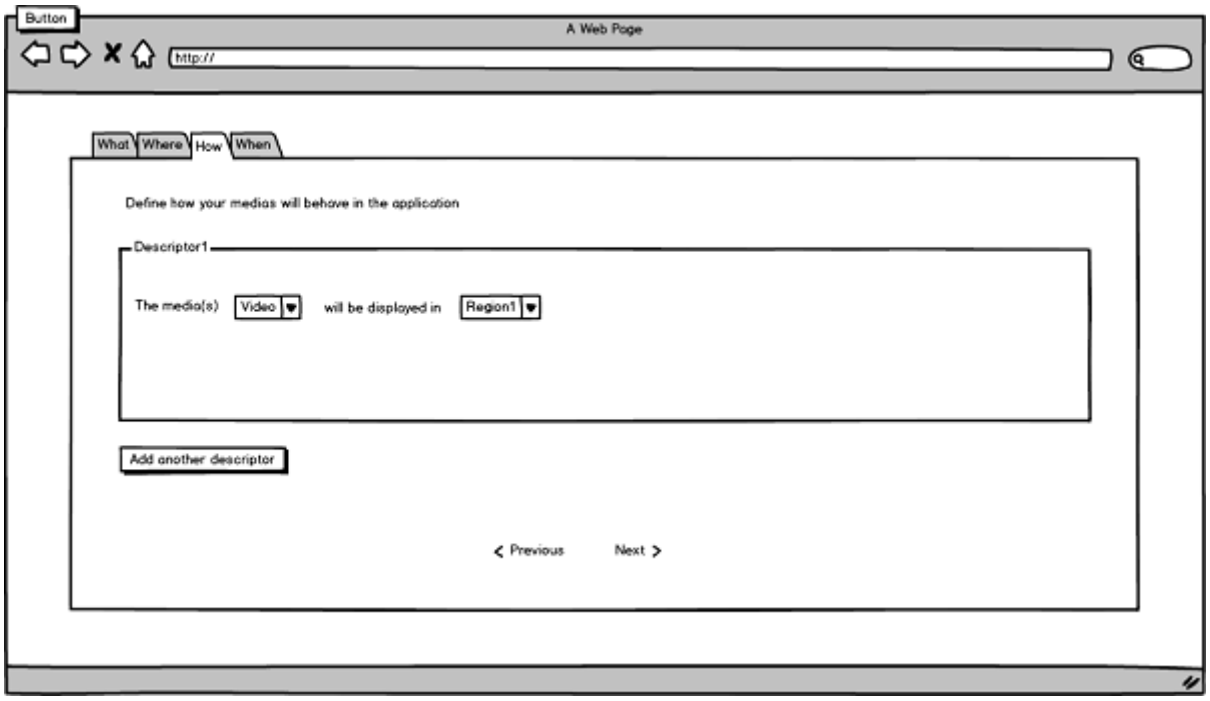

Figura 3.4: Esboço da tela de descritores

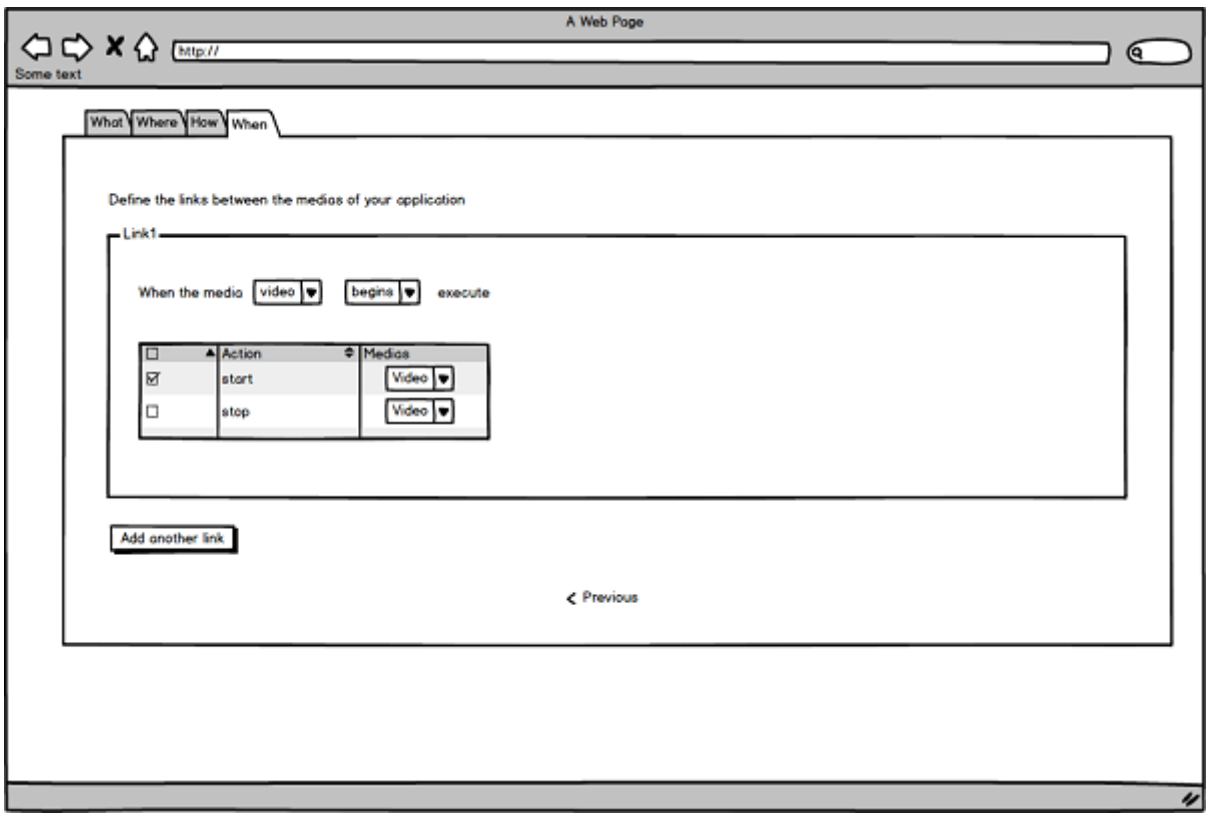

Figura 3.5: Esboço da tela de links

Tabela 3.1: Elementos e atributos cobertos pelos casos de teste

Elementos	Atributos	Conteúdo
ncl	id xmlns	head body
head		
regionBase		region
region	id width height zIndex top left right bottom	
descriptorBase		descriptor
descriptor	id region explicitDur moveLeft moveRight moveUp moveDown focusIndex	descriptorParam
descriptorParam	name(soundLevel) value	
connectorBase		causalConnector
causalConnector	id	simpleCondition compoundAction simpleAction
compoundAction	operator(seq)	simpleAction
simpleAction	role max(unbounded) qualifier(par)	
body		port media link
port	id component	
media	id src	
link	id xconnector	bind
bind	role component	

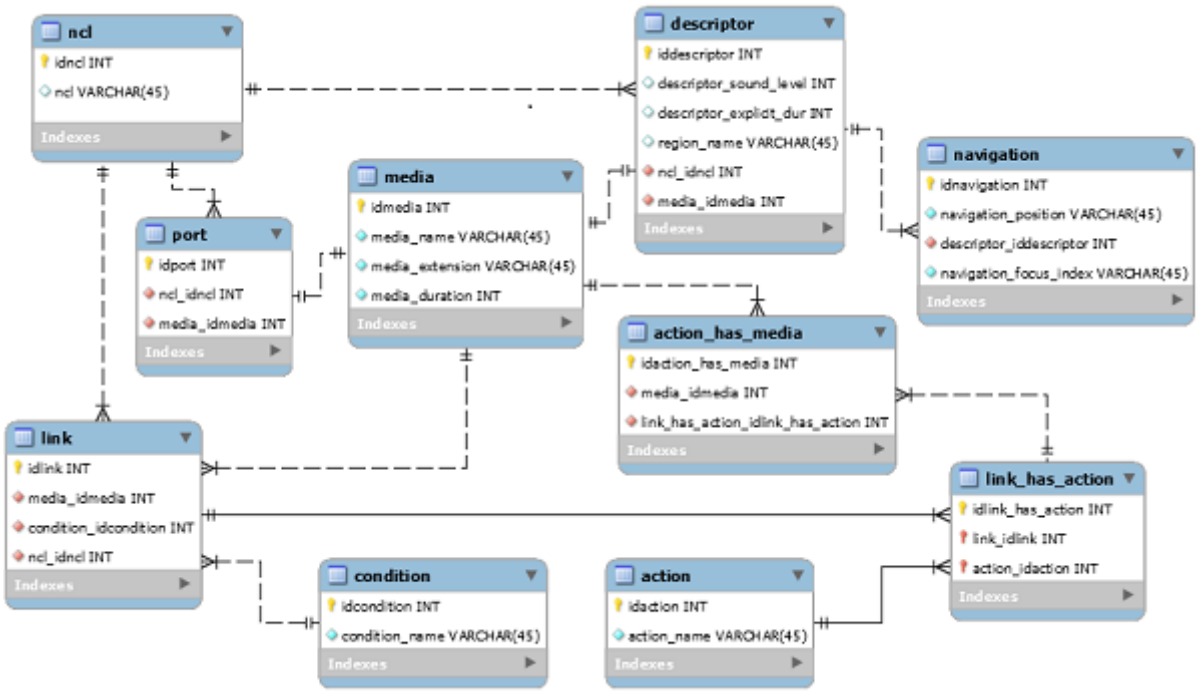

Figura 3.6: Modelo do banco de dados

4 Implementação

A ferramenta foi implementada para acesso via *Web*, de forma a tornar qualquer navegador da Internet uma plataforma de execução da ferramenta no lado cliente.

O servidor usado para a construção foi o XAMPP 5.6.8 (Apache Friends, 2005). Tal servidor oferece facilidades para instalação do servidor *Web* Apache (Apache Foundation, 1999), contendo o banco de dados *MySQL* (MySQL, 2009), interpretadores PHP (PHP, 1995) e Perl (Perl, 1987).

Para o desenvolvimento foi utilizado o *Codeigniter* (Codeigniter, 2006), um *framework* PHP que utiliza o conceito de orientação a objetos e a arquitetura MVC, explicada anteriormente, além de possuir uma documentação vasta e de fácil entendimento.

Para a construção da interface foi utilizado o *Bootstrap* (Bootstrap, 2011), um *framework* HTML, CSS e *Javascript*. O mesmo se mostrou útil devido ao fato de possuir estruturas prontas para serem usadas e a possibilidade de tornar a ferramenta responsiva.

Também foi utilizado o AngularJS (AngularJS, 2009), um *framework Javascript* que se mostrou muito útil para o desenvolvimento da interface da ferramenta, devido ao *data-binding* e o uso de diretivas para a manipulação do DOM. O *data-binding* trata-se de um sincronismo bidirecional automático entre o DOM(*view*) e os dados mostrados no mesmo, com os quais o usuário interage(*model*).

O banco foi contruído utilizando-se o *MySQL Workbench* (Workbench, 2005), pois através dela é possível construir o banco modelando-o graficamente, além da possibilidade de sincronismo com o gerenciador do banco de dados.

Para a validação semântica e sintática dos casos de teste gerados foi utilizado o NCL-validator (2008), uma implementação Java que, após sua execução, exibe os alertas e/ou erros encontrados. O validador foi instalado no servidor e suas mensagens podem ser visualizadas em português(pt_BR), inglês(en_US) ou espanhol(es_ES) (NCL-validator, 2008).

A seguir será apresentada a ferramenta e seu funcionamento:

1. Seleção das mídias da aplicação(*What*): Nesta tela são exibidas as mídias disponníveis na biblioteca. Pelo menos uma delas deve ser selecionada. Dentre as selecionadas uma deve ser escolhida como a mídia de entrada para o contexto. Definidas as mídias que comporão o caso de teste, a criação do mesmo pode prosseguir.

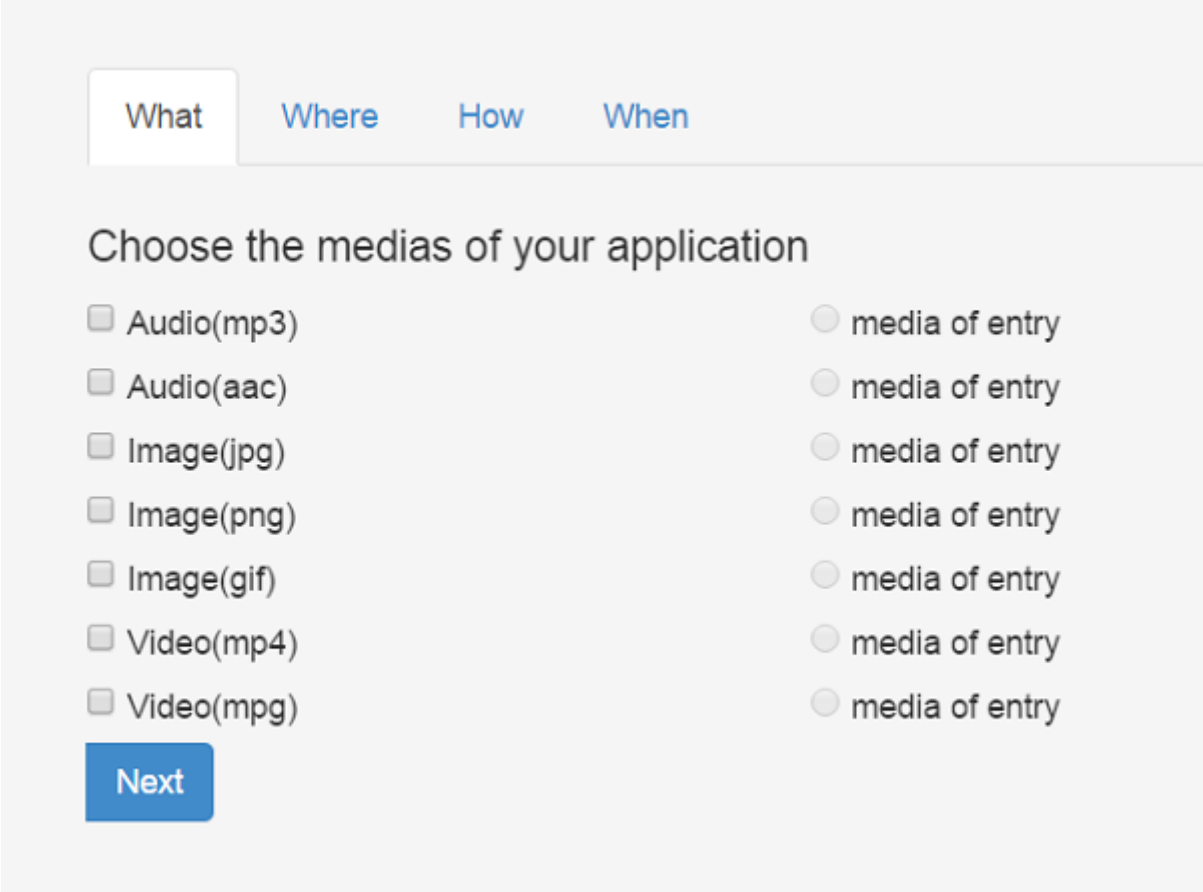

Figura 4.1: Tela de escolha de mídias

2. Associação das mídias às regiões(*When*): Nesta tela são exibidas as regiões.

   São 6 as regiões possíveis:

   - *top left*: definida pela área amarela da tela;
   - *top right*: definida pela área vermelha da tela;
   - *bottom left*: definida pela área azul da tela;
   - *bottom right*: definida pela área verde da tela;
   - *center*: definida pela área branca;
   - *background*: definida pela área preta que se encontra atrás das demais;

As mídias escolhidas, exceto as do tipo Áudio, são exibidas em cada uma das regiões. Ao se associar uma mídia a uma região, a mesma fica indisponível para as demais regiões. Se somente mídias do tipo Áudio tiverem sido selecionadas, a execução do teste prossegue, sem restrições. Em caso contrário, a associação às regiões é obrigatória.

Na figura 4.2 é mostrada a tela de associação de mídias à regiões e como as mesmas são exibidas.

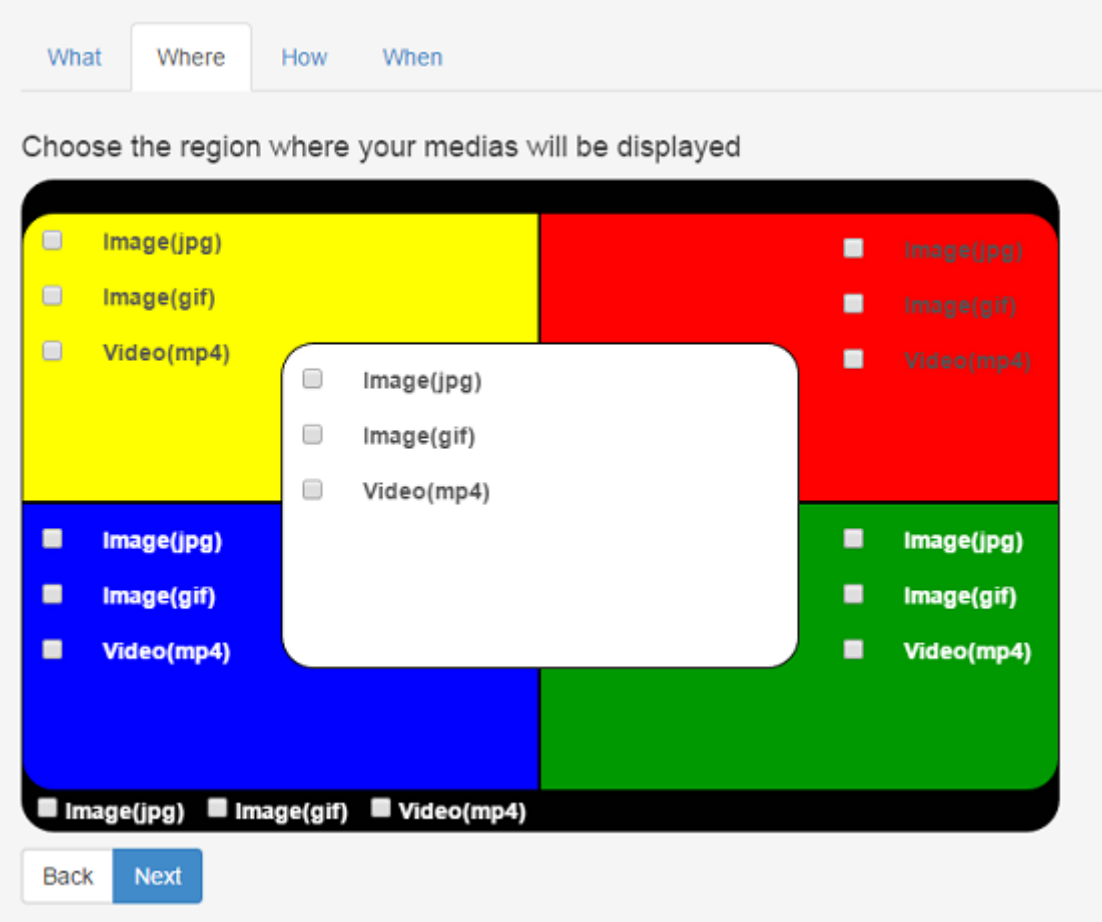

Figura 4.2: Tela de associação de mídias à regiões

Nesta tela, a princípio, é possível visualizar as regiões, bem como se uma ou mais mídias do tipo áudio foram selecionadas.

Caso uma região contenha alguma mídia, a mesma aparecerá colorida, seguindo o mesmo esquema de cores da tela anterior; e ao selecioná-la serão exibidas as mídias contidas na mesma e as opções do descritor para cada uma.

As opções do descritor são as seguintes: *Duration*, *Sound Level* e *Navigation*. O

campo da opção *Duration* aceita apenas números inteiros tendo como limite superior a duração da mídia em questão. Já o campo *Sound Level*, assim como o descrito acima, aceita apenas números inteiros entre 0 e 100, sendo que o volume da mídia permanece inalterado se o mesmo não for preenchido. O campo *Navigation* apresenta todas as mídias anteriormente selecionadas, inclusive a que está em foco, mas a mesma não se encontra habilitada para a seleção.

Como dito, tais opções serão exibidas segundo as mídias. Por exemplo, *Navigation* não se enquadra as mídias do tipo áudio, bem como *Sound Level* às do tipo imagem (NCLHandbook, 2012).

A definição de descritores não é obrigatória, ou seja, o criador do caso de teste não precisa definí-los para prosseguir com a criação do mesmo.

Na figura 4.3 é mostrada a tela de definição de descritores. Já na figura 4.4 é mostrada a tela de definição de descritores, com ênfase no campo *Navigation*.

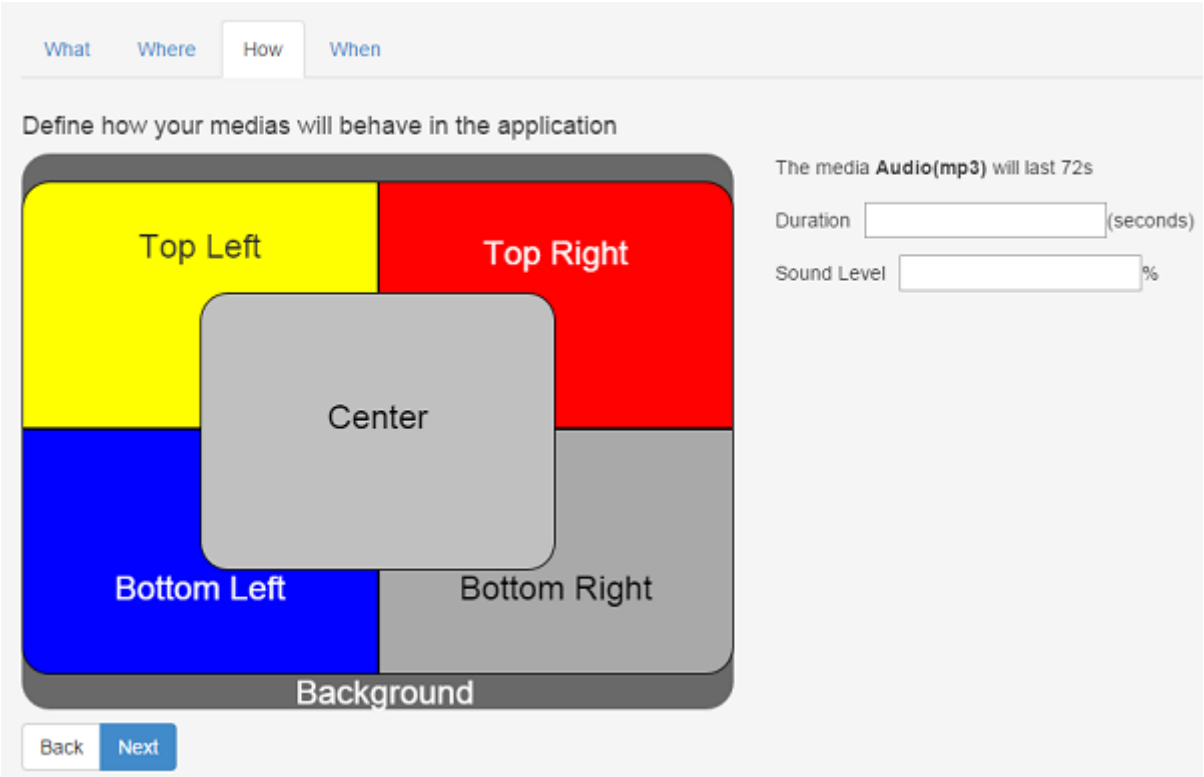

Figura 4.3: Tela de definição de descritores

3. Sincronismo de mídias(*When*): Nesta tela são apresentadas as condições e ações para o sincronismo de mídias. Como mencionado no capítulo 3, as condições e as

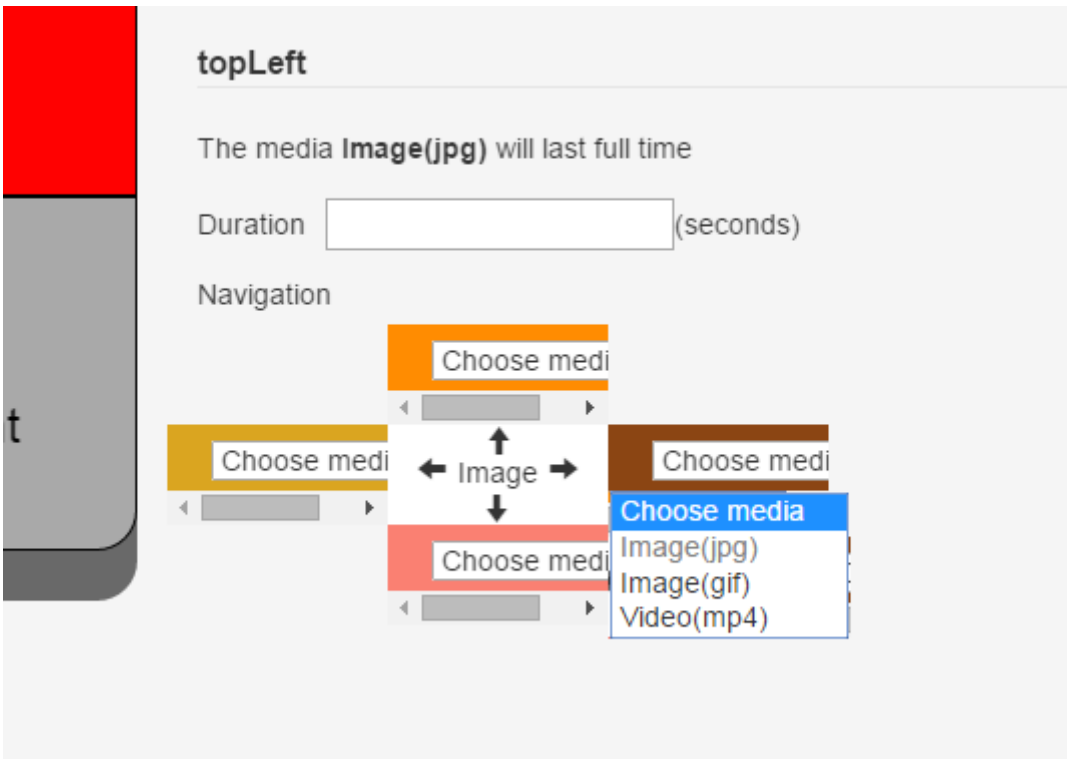

Figura 4.4: Tela de definição de descritores - Navigation

ações possuem valores pré definidos, relembrados a seguir:

- Condições: *OnBegin*, *OnEnd* e *OnSelection*
- Ações: *Start* e *Stop*

Para definir um *link*, deve-se escolher uma condição, bem como a mídia que dará início ao sincronismo. Após este passo, devem ser definidas as ações que a mesma dispara, bem como, as mídias vinculadas a cada ação. Para que os *links* sejam criados com sucesso e exibidos no painel *Links*, que se encontra na parte superior da tela, é necessário que o criador do teste siga exatamente os passos descritos acima.

É permitido ainda excluir *links* já definidos, bem como definir sincronismos com ações simples ou compostas. No caso de sincronismos com ações compostas, estas serão executadas sequencialmente, sendo que a ação *Stop* tem precedência a ação *Start*.

O sincronismo não é obrigatório, ou seja, o criador do caso de teste não precisa definir nenhum *link* para prosseguir com a criação do mesmo.

Na figura 4.5 é mostrada a tela de inicial de sincronismo, sem nenhum *link* definido. Já na figura 4.6 é mostrada a tela de sincronismo, com *links* já definidos.

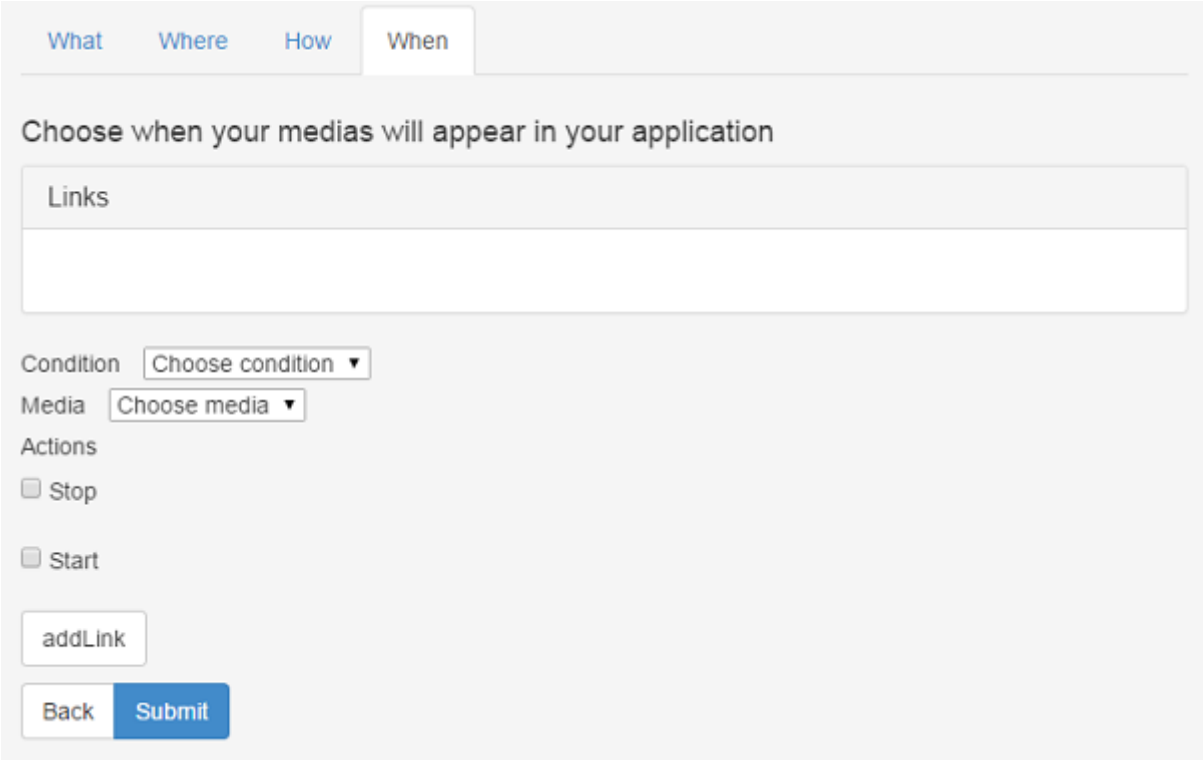

Figura 4.5: Tela de sincronismo

4. Envio de dados e construção do caso de teste: Ao pressionar o botão *submit*, os dados são enviados ao servidor e gravados no banco. Caso tudo ocorra como o esperado, o criador do caso de teste é encaminhado para a página de resultados, onde o mesmo pode fazer o *download* do caso de teste, escrito em NCL, e das mídias que o compõem. Nesta tela também é possível que o mesmo visualize o resultado da validação e os elementos e atributos que seu caso de teste cobre. Grande parte dos dados desta página, exceto o *download* de mídias, são recuperados à partir do identificador do caso de teste gerado. Na figura 4.7 é mostrada a tela de resultados.

   Ao se clicar sobre o botão *Download* NCL, os dados do caso de teste são recuperados do banco. De acordo com estes dados e, em conformidade com a Recomendação H.761 ITU-T (2011), as Normas ABNT NBR 15606-2 (2007) e Guia operacional do ISDB-TB ABNT NBR 15606-7 (2011), o documento NCL correspondente é construído, linha por linha, à partir de um *script* PHP. Nesse *script* os trechos do código

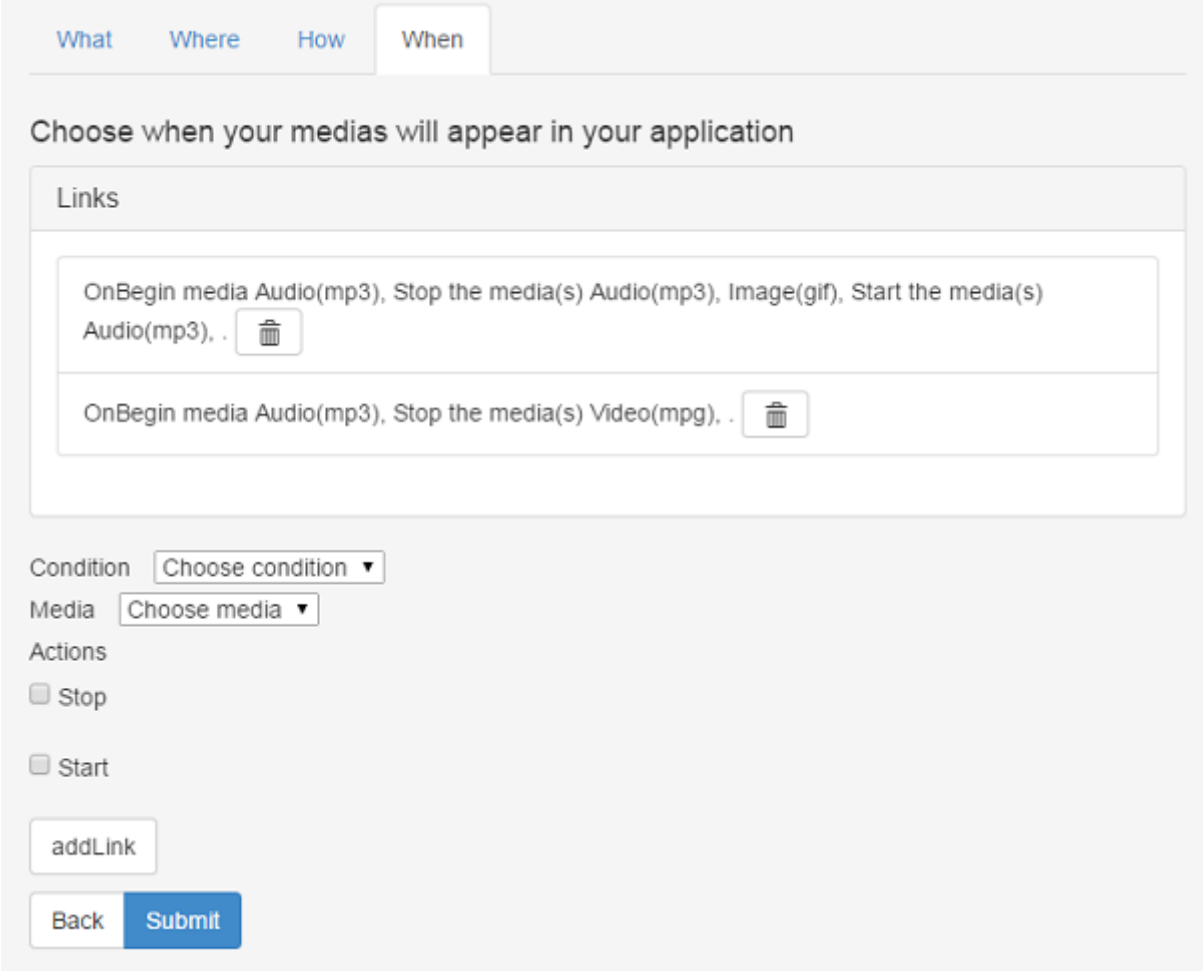

Figura 4.6: Tela de sincronismo - com links já definidos

gerados são reservados ao longo da execução e concatenados. O documento NCL resultante é encaminhado para *download*.

O *download* da mídia contida na biblioteca de mídias é feita ao se clicar sobre o botão correspondente a mesma. O identificador da mídia é repassado ao servidor que recupera seus dados e encaminha o arquivo correspondente para *download*.

No canto direito da tela encontram-se dois botões: *See the scope of your test case* e *See the results of the validation*.

Pressionando o primeiro botão, são exibidos os elementos e atributos abrangidos pelo caso de teste gerado. Os dados são recuperados do banco de acordo com o identificador do caso de teste, agrupados de acordo com o tipo (descritores, regiões, etc) e encaminhados à tela de resultados. Na figura 4.8 é mostrado um exemplo de escopo de um caso de teste gerado.

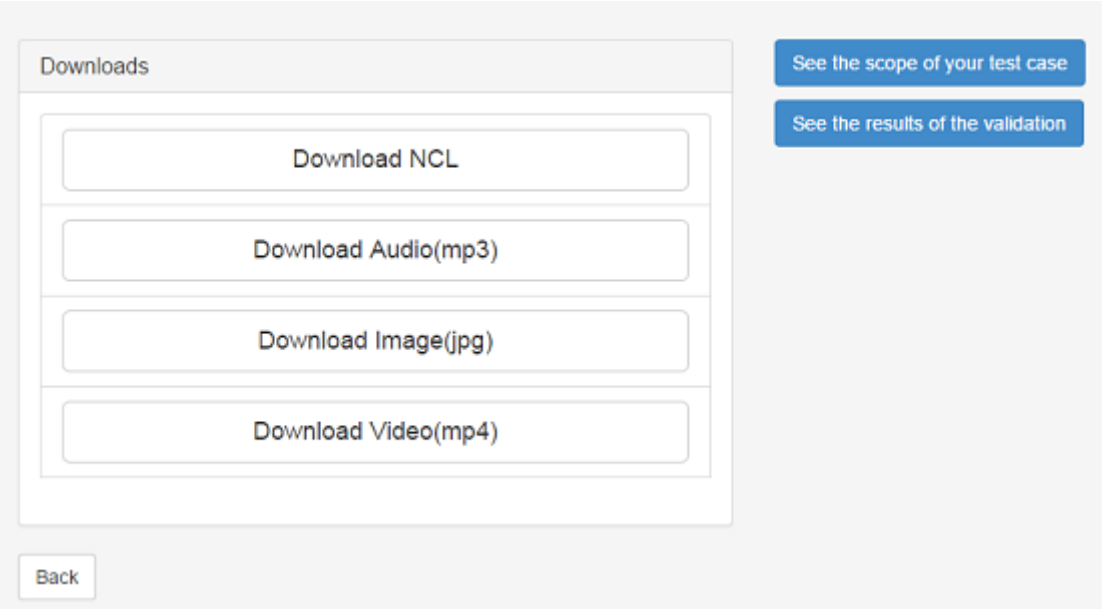

Figura 4.7: Tela de resultados do teste

Já pressionando o segundo botão, são exibidos os resultados da validação do caso de teste gerado. Com mencionado no começo da seção, a validação é feita utilizando-se o NCL-validator (2008), que é executado, pelo *script* PHP correspondente, via linha de comando da seguinte maneira: "java -jar ncl-validator.jar -nl pt_BR nome-arquivo.ncl". O arquivo ncl do caso de teste é criado com nome único no mesmo diretório onde se encontra o validador. Feita a validação do arquivo, o mesmo é apagado do diretório, e os alertas e/ou erros obtidos são encaminhados a tela de resultados. A definição de um nome único para o arquivo, bem como sua exclusão ao final do processo de validação, é de extrema importância, uma vez que a ferramenta será disponibilizada na *Web* e sofrerá diversos acessos ao mesmo tempo. Na figura 4.9 é mostrado um exemplo de retorno de validação de um caso de teste gerado.

Na figura 4.10 e 4.11 é mostrado um exemplo de um caso de teste gerado. Pode-se observar que o mesmo segue a estrutura apresentada no capítulo 2.

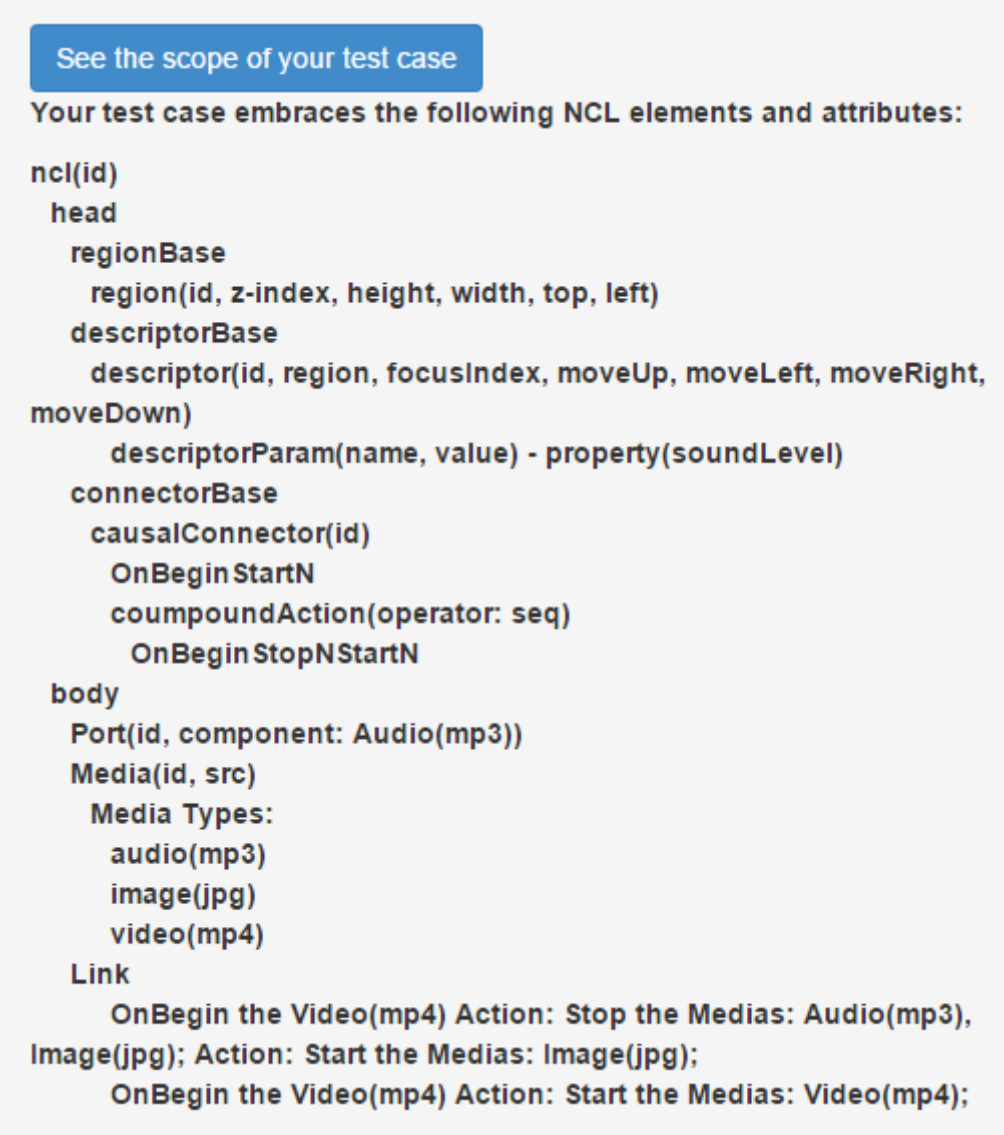

Figura 4.8: Exemplo de escopo de caso de teste gerado

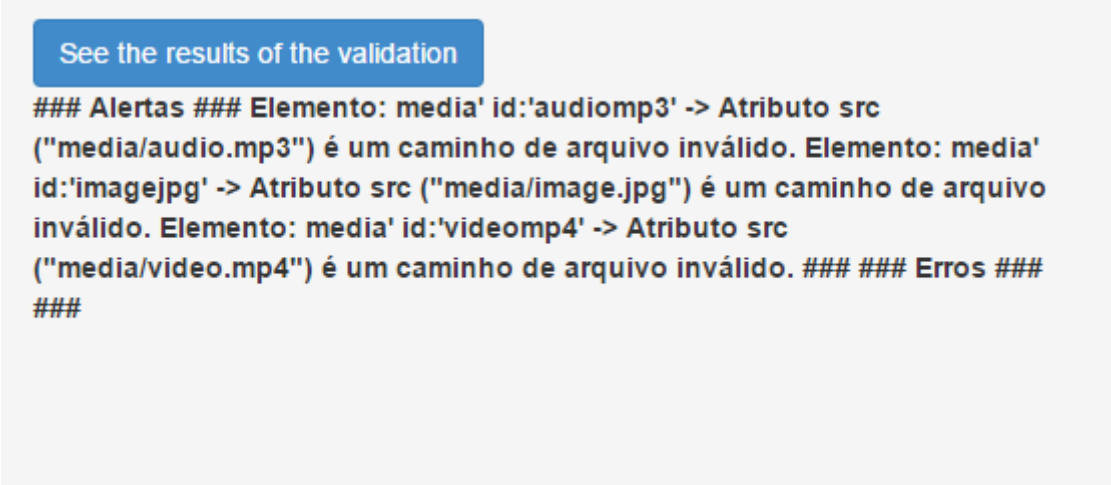

Figura 4.9: Exemplo de resultado de validação de um caso de teste gerado

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <ncl xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile" id="nclEx">
  - <head>
    - <regionBase>
        <region id="topLeftReg" zIndex="2" height="45%" width="50%" top="5%"
          left="0"/>
        <region id="BackgroundReg" zIndex="1" height="100%" width="100%"/>
      </regionBase>
    - <descriptorBase>
      - <descriptor id="audiomp3Desc" explicitDur="34s">
          <descriptorParam value="67%" name="soundLevel"/>
        </descriptor>
        <descriptor id="imagejpgDesc" moveDown="videomp4Idx"
          moveRight="videomp4Idx" moveLeft="videomp4Idx"
          moveUp="videomp4Idx" focusIndex="imagejpgIdx"
          region="topLeftReg"/>
        <descriptor id="videomp4Desc" focusIndex="videomp4Idx"
          region="BackgroundReg"/>
      </descriptorBase>
    - <connectorBase>
      - <causalConnector id="OnBeginStartN">
          <simpleCondition role="OnBegin"/>
          <simpleAction role="start" qualifier="par" max="unbounded"/>
        </causalConnector>
      - <causalConnector id="OnBeginStopNStartN">
          <simpleCondition role="OnBegin"/>
        - <compoundAction operator="seq">
            <simpleAction role="stop" qualifier="par" max="unbounded"/>
            <simpleAction role="start" qualifier="par" max="unbounded"/>
          </compoundAction>
        </causalConnector>
      </connectorBase>
    </head>
```

Figura 4.10: Exemplo de código NCL gerado pelo teste - cabeçalho

```
  - <body>
      <port id="entry" component="audiomp3"/>
      <media id="audiomp3" descriptor="audiomp3Desc" src="media/audio.mp3"/>
      <media id="imagejpg" descriptor="imagejpgDesc" src="media/image.jpg"/>
      <media id="videomp4" descriptor="videomp4Desc" src="media/video.mp4"/>
    - <link id="lVideomp4" xconnector="OnBeginStopNStartN">
        <bind role="OnBegin" component="videomp4"/>
        <bind role="stop" component="audiomp3"/>
        <bind role="stop" component="imagejpg"/>
        <bind role="start" component="imagejpg"/>
      </link>
    - <link id="lVideomp41" xconnector="OnBeginStartN">
        <bind role="OnBegin" component="videomp4"/>
        <bind role="start" component="videomp4"/>
      </link>
    </body>
</ncl>
```

Figura 4.11: Exemplo de código NCL gerado pelo teste - corpo

5 Conclusões

Como discutido ao longo do texto, testes de conformidade são de extrema importância para garantir a qualidade das implementações comerciais do Ginga oferecidas pelos fabricantes. Por essa razão foi criada uma suíte de testes de conformidade. Tal suíte é eficaz, porém possui estrutura complexa, além de exigir o conhecimento da linguagem NCL, o que dificulta seu uso por criadores de teste que não possuem experiência na linguagem. Outro ponto importante é que a mesma se resume a um repositório colaborativo de testes, que, dificilmente, cobriria as infinitas possíveis entradas de teste para linguagens baseadas em XML, como a linguagem NCL.

Nesse sentido, este trabalho propôs a disponibilização de uma ferramenta de autoria onde casos de testes para Ginga-NCL sejam criados de forma simples e dinâmica, à partir das escolhas do criador do teste, oferecendo abstrações gráficas em alto nível dos elementos da linguagem NCL. Como explicitado no capítulo 4, não é necessário o conhecimento prévio da linguagem para a construção dos casos de teste através da ferramenta, apenas das normas relativas a especificação Ginga-NCL. Encontra-se disponibilizada na mesma uma biblioteca de mídias, as quais são usadas na construção dos casos de teste, o que isenta o criador destes de fazer o *upload* das mídias. A ferramenta apresenta, de forma organizada, os elementos e atributos que o caso de teste criado abrange, bem como os resultados da validação. Tais casos são validados sintática e semanticamente utilizando-se o NCL-validator (2008), o que garante a concordância deles às normas referidas acima.

Embora a ferramenta atenda aos requisitos apresentados no capítulo 3, há alguns pontos a serem melhorados: adição de elementos da linguagem NCL à ferramenta, melhora na construção do código NCL, criação de vídeo simulando o caso de teste gerado, adaptação da interface da ferramenta proposta para a criação de aplicações NCL em geral.

Para a adição de novos elementos da linguagem NCL, o banco de dados foi modelado para se adaptar às mudanças necessárias para tal, como demonstrado no capítulo 3.

Já a melhoria na contrução do código NCL do caso de teste pode ser alcançada

através da identificação de técnicas em Engenharia de *Software* que podem ser aplicadas para definir trechos reutilizáveis de código.

A criação do vídeo envolve a pesquisa de extensões, bibliotecas ou plugins para a linguagem PHP. Outra forma possível de atingir esse objetivo, é a junção do PHP a outra linguagem, que ofereça os recursos necessários para tanto, através de um *Web Service*.

Para a que a ferramenta seja utilizada na geração de aplicações NCL, a interface precisaria de alguns ajustes, principalmente a tela da definição de sincronismo de mídias(*when*), para que a mesma ofereça uma abstração gráfica ainda maior dos elementos NCL contidos nela.

Referências Bibliográficas

ABNT, N. 15606-2 (2007)–associação brasileira de normas técnicas. **Televisão digital terrestre -Codificação de dados e especificações de transmissão para radiodifusão digital–Parte**, v.2, p. 15606–2, 2007.

ABNT, N. 15606-5: 2008. **Televisão digital terrestreâCodificação de dados e especificações de transmissão para radiodifusão digital Parte**, v.5, p. 15606–5, 2008.

ABNT, N. 15606-7 (2011)–associação brasileira de normas técnicas. **Televisão digital terrestre - Codificação e transmissão de dados para radiodifusão digital–Parte**, v.7, 2011.

Angularjs. disponível em: https://angularjs.org/. acesso em: 15 jun.2015.

Xampp. disponível em: https://www.apachefriends.org/. acesso em: 16 jun.2015.

Apache. disponível em: http://www.apache.org/. acesso em: 16 jun.2015.

ARAÚJO, E. C. e. a. Suíte de testes de conformidade para o ginga-ncl. 2011.

Bootstrap. disponível em: http://getbootstrap.com/. acesso em: 15 jun.2015.

Brackmann, C. P. Sistema brasileiro de tv digital. **UNIVERSIDADE CATÓLICA DE PELOTAS, Pelotas**, 2008.

BRACKMANN, C. Usabilidade em tv digital. **Trabalho de Mestrado em Ciência da Computação na Universidade Católica de Pelotas**, 2010.

Codeigniter. disponível em: http://www.codeigniter.com/. acesso em: 15 jun.2015.

Committee, S. E. T.; others. Ieee standard for software test. ieee standard ieee std 829-1998. **IEEE Computer Society**, v.345, p. 10017–2394.

ITU-T. H.761 (2011)–international telecommunication union. **Nested context language (NCL) and Ginga-NCL**, p. H.761–112, 2011.

LUCIANO, J.; ALVES, W. J. B. Padrão de arquitetura mvc: Model-view-controller.

Mandel, T. **The elements of user interface design**, volume 20. Wiley New York, 1997.

Becker, V.; Montez, C. **TV digital interativa: conceitos, desafios e perspectivas para o Brasil**. I2TV, 2004.

Mysql. disponível em: https://docs.oracle.com/cd/e17952$_0$1/.*acessoem* : 16*jun*.2015.

Ncl handbook. disponível em: http://handbook.ncl.org.br/doku.php. acesso em: 22 jun.2015.

Ncl validator. disponível em: http://laws.deinf.ufma.br/nclvalidator/index.html. acesso em: 22 jun.2015.

Php. disponível em: http://php.net/docs.php. acesso em: 16 jun.2015.

Perl. disponível em: http://perldoc.perl.org/. acesso em: 16 jun.2015.

Ratamero, E. M. **Tutorial sobre a linguagem de programaçao ncl (nested context language)**, 2007.

Soares, L. F. G.; Souza Filho, G. L. d. **Interactive television in brazil: System software and the digital divide**. In: European Conference on Interactive TV (EuroITV), Áustria, 2007.

Soares, L. F. G. S. **Programando em NCL 3.0: desenvolvimento de aplicaçoes para middleware Ginga: TV digital e Web**. Elsevier, 2009.

Mysql workbench. disponível em: https://www.mysql.com/products/workbench/. acesso em: 15 jun.2015.