Referências Bibliográficas

[Araújo et al. 2008] ARAÚJO, E.; AZEVEDO, R. ; SOARES, C.. **Ncl-validator: um processo para validação sintática e semântica de documentos multimídia ncl**. . II Jornada de Informática do Maranhão, 2008. 1.1, 3

[CheckStyle 2002-9] **Checkstyle**. Website, 2002-9. Acessado em 02/05/2009, URL: `http://checkstyle.sourceforge.net/`. 3

[Dieterich et al. 1993] DIETERICH, H.; MALINOWSKI, U.; KÜHME, T. ; SCHNEIDER-HUFSCHMIDT, M.. **State of the art in adaptive user interfaces**. In: ADAPTIVE USER INTERFACES. Elsevier Science Publishers, 1993. 4.1.7

[FindBugs 2002-9] **Findbugs**. Website, 2002-9. Acessado em 29/04/2009, URL: `http://findbugs.sourceforge.net/`. 3

[Fischer 1990] FISCHER, G.; LEMKE, A. C.; MASTAGLIO, T. ; MORCH, A. I.. **Using critics to empower users**. In: CHI '90: PROCEEDINGS OF THE SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, p. 337–347, New York, NY, USA, 1990. ACM. 4.1.3, 4.1.5, 4.1.6

[Fischer 1993] FISCHER, G.; NAKAKOJI, K.; OSTWALD, J.; STAHL, G. ; SUMNER, T.. **Embedding computer-based critics in the contexts of design**. In: CHI '93: PROCEEDINGS OF THE INTERACT '93 AND CHI '93 CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, p. 157–164, New York, NY, USA, 1993. ACM. 1.1

[Fowler et al. 1999] FOWLER, M.; OTHERS. **Refactoring: improving the design of existing code**. Addison-Wesley Professional, 1999. 2.1, 2.1, 2.1, 2.1

[Fowler et al. 2002] FOWLER, M.. **Patterns of Enterprise Application Architecture**. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002. 4.2.4

[Fowler Bliki 2010] **Fluent interfaces**. Website, 2010. Acessado em 10/03/2010, URL: `http://martinfowler.com/bliki/FluentInterface.html`. 4.2.5

[Gamma et al. 1995] GAMMA, E.; HELM, R.; JOHNSON, R. ; VLISSIDES, J.. **Design patterns: elements of reusable object-oriented software**. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995. 3.3, 4.2

[Guimaraes 07] GUIMARÃES, R.. **Composer: um ambiente de autoria de documentos NCL para TV digital interativa**. Master's thesis, PUC-Rio, 2007. 1.1

[Hix et al. 93] HIX, D.; HARTSON, H. R.. **Developing User Interfaces: Ensuring Usability Through Product and Process**. Número 1. John Wiley and Sons, Inc., 1993. 1.1

[Hunt et al. 1999] HUNT, A.; THOMAS, D.. **The pragmatic programmer: from journeyman to master**. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999. 6.1.2, 2

[JDT 2002-9] **Jdt - eclipse java development tools**. Website, 2002-9. Acessado em 29/04/2009, URL: `http://www.eclipse.org/jdt/`. 3

[Lee et al. 2000] LEE, D.; CHU, W. W.. **Comparative analysis of six xml schema languages**. SIGMOD Rec., 29(3):76–87, 2000. 2.2.1

[Miller 1983] MILLER, P. L.. **ATTENDING: Critiquing a physician's management plan.** IEEE TRANS. PATTERN ANAL. MACH. INTELLIG., 5(5):449–461, September 1983. 1.1

[Miller 1986] MILLER, P. L.. **Expert Critiquing Systems**. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1986. 1.1

[NCLSpec 2007] **Digital terrestrial television – data coding and transmission specification for digital broadcasting – part 2: Ginga-ncl for fixed and mobile receivers – xml application language for application coding**. Technical report, Associação Brasileira de Normas Técnicas, São Paulo, SP, Brazil, November 2007. 1

[NCLEclipse 2008] **Ncl eclipse**. Website, 2008. Acessado em 07/05/2009, URL: `http://laws.deinf.ufma.br/~ncleclipse/`. 1.1

[ObjectAsASuperclass] **Object as a superclass**. Website, 2008. Acessado em 02/05/2009, URL: `http://java.sun.com/docs/books/tutorial/java/IandI/objectclass.html`. 3.2

[PMD 2002-9] **Pmd**. Website, 2002-9. Acessado em 29/04/2009, URL: `http://pmd.sourceforge.net/`. 3

[Rosson et al. 2002] ROSSON, M. B.; CARROLL, J. M.. **Usability engineering: scenario-based development of human-computer interaction**. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002. 5.1.3

[SMIL 2008] **W3c smil**. Website, 2008. Acessed in 03/03/2010, available at: `http://www.w3.org/TR/SMIL3/`. 1

[SVG 2009] **W3c svg**. Website, 2009. Acessed in 03/03/2010, available at: `http://www.w3.org/TR/SVG/`. 1

[SchematronSpec] **Iso schematron**. Website, 2004. Acessed in 09/12/2009, available at: `http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html`. 4.1.2

[Soares et al. 2009] DE SALLES SOARES NETO, C.; SOARES, L. F. G.. **Autoria orientada a arquétipos para tv digital: uma abordagem restritiva e direcionada**. WebMedia, 2009. 5.2.8

[Soares e Barbosa 2009] SOARES, L. F. G.; BARBOSA, S. D. J.. **Programando em NCL 3.0**. Campus, 2009. H

[dos Santos et al. 2009] DOS SANTOS, J. A. F.; SAADE, D. C. M.. **Linguagem xtemplate 3.0: Facilitando a autoria de programas ncl para tv digital interativa**. WebMedia, 2009. 5.2.8

[van Deursen et al. 2000] VAN DEURSEN, A.; KLINT, P. ; VISSER, J.. **Domain-specific languages: an annotated bibliography**. SIGPLAN Not., 35(6):26–36, 2000. 4.1.2

A
Compilando o NCL-Inspector a partir do código fonte

Até o momento da escrita desta dissertação nenhuma versão estável do NCL Inspector foi liberada. Porém uma versão usável pode ser baixada diretamente da linha principal de desenvolvimento. A obtenção do arquivo executável poderá ser feita através da compilação deste código fonte. As informações aqui demonstradas tanto servem para sistemas operacionais Windows quanto para Unix.

Para compilar o código fonte do NCL-Inspector, são necessários:

1. Java SE Development Kit 6 (JDK 6). Disponível em http://java.sun.com.
2. Apache Maven 2. Obtido em http://maven.apache.org.
3. Cliente SVN de linha de comando (pode ser outro equivalente, como o Tortoise, porém será demonstrado aqui usando apenas linha de comando). Disponível em http://subversion.tigris.org.
4. Conexão com a internet disponível.

O NCL-Inspector depende do componente NCL-Parser, que foi desenvolvido em conjunto por mim e outro aluno de mestrado (Hildebrando Trannin), que também não possui uma versão estável. Portanto, antes de compilar o código do NCL-Inspector, precisaremos fazer o mesmo para o NCL-Parser. Crie um diretório para armazenar as cópias de trabalho do código fonte do NCL-Inspector e do NCL-Parser. Iremos chamar essa pasta de `nclinspector-workspace`. Na linha de comando, entre dentro da pasta criada e chame o comando (lembrando que o cliente SVN de linha de comando deverá estar declarado na variável de ambiente `PATH` para esse comando funcionar):

```
svn checkout http://ncl-parser.googlecode.com/svn/trunk/ncl-parser
```

Após o término da execução deste comando, o próximo passo é compilar o NCL-Parser utilizando a ferramenta Maven 2. A versão desta ferramenta que utilizamos foi a 2.2.1. Para instalar o Maven, basta ir até a seção de downloads do site, fazer o download. Ao término descompacte o arquivo em um determinado diretório. Inclua a pasta `bin` que vem junto com o Maven na variável de ambiente `PATH`, para que você possa executar o comando de qualquer local do sistema. Tendo feito isto, no console, basta entrar na pasta do NCL-Parser onde está o arquivo `pom.xml` (deverá estar na pasta `nclinspector-workspace/ncl-parser`) e digitar o comando:

```
mvn install
```

Este comando irá baixar todas as bibliotecas e dependências necessárias para a compilação do código fonte e irá executar a compilação do NCL-Parser. Tenha paciência pois o comando leva um certo tempo para realizar a compilação. Ao terminar, execute o comando `cd ..` e deixe a pasta onde está o NCL-Parser, voltando pra raiz das cópias de trabalho (`nclinspector-workspace`). Então baixe o código fonte do NCL-Inspector utilizando o seguinte comando:

```
svn checkout http://ncl-inspector.googlecode.com/svn/trunk/nclinspector
```

Analogamente à compilação do NCL-Parser, para compilar o NCL-Inspector entre na pasta `nclinspector` (que deverá estar o arquivo `pom.xml` e execute o comando `mvn install`. Ao término deste comando, a versão de linha de comando do NCL-Inspector estará dentro do arquivo zipado:

```
nclinspector-console-<Versão>.zip
```

localizado na pasta:

```
nclinspector/nclinspector-console/target
```

Dentro deste `zip`, existe uma pasta `bin`. Nesta pasta, possuem dois scripts para iniciar a execução do NCL-Inspector. Para executar, os seguintes argumentos deverão ser passados:

```
ncl-inspector-console <configFile> <inspectedFile> [rulelib path]
```

Os argumentos entre <> são obrigatórios e os entre [] são opcionais. O primeiro argumento serve para especificar o arquivo de configuração do NCL-Inspector que informa ao programa quais regras de verificação serão executadas. O NCL-Inspector vem com um arquivo com todas as regras disponibilizadas por ele, na pasta `conf`. Este arquivo poderá ser usado caso não deseje personalizar as regras que serão executadas. Se desejar uma execução de apenas algumas regras, outro arquivo pode ser criado tomando este como modelo.

O segundo argumento deve ser o arquivo que será inspecionado pelo NCL-Inspector. O terceiro é opcional e é recomendado deixar em branco, a não ser que deseje especificar outra pasta que contenha implementações de regras de inspeção.

B
Preparando o ambiente de desenvolvimento de novas regras

Antes de preparar o ambiente de desenvolvimento de novas regras, é necessário que todos as operações descritas no Apêndice A tenham sido realizadas com sucesso.

B.1
Desenvolvendo novas regras

Para desenvolver novas regras para o NCL-Inspector usando XSL é necessário apenas um editor de texto. Caso seja necessário o desenvolvimento em Java, também será necessário o compilador da linguagem e o arquivo `jar` correspondente ao mecanismo de regras.

Apesar de ser perfeitamente possível a criação de regras utilizando apenas as ferramentas mencionadas no parágrafo anterior, para criar o conjunto de regras padrão do NCL-Inspector, utilizamos um ambiente um pouco mais sofisticado. Usar um ambiente mais sofisticado nos dá um custo maior de configuração, porém fornece uma maior facilidade na criação das regras. Escolher qual ambiente utilizar é uma decisão muito subjetiva. Se o desenvolvedor necessita escrever muitas regras, talvez seja interessante gastar um tempo configurando um ambiente mais sofisticado. Caso contrário, se apenas algumas poucas regras forem escritas, talvez o custo de configurar um ambiente mais sofisticado não valha a pena.

Para configurar o ambiente de criação de regras, necessitaremos dos seguintes componentes:

1. Eclipse IDE for Java Developers 3.5 (Galileo) SR1.
2. m2eclipse (plugin do Maven para o Eclipse)

Vá até o site http://www.eclipse.org/downloads/ e baixe a versão do Eclipse de acordo com a figura B.1. Descompacte o arquivo, isto resultará na criação de uma pasta chamada `eclipse`. Dentro desta pasta, execute a aplicação através do arquivo `eclipse.exe` se o seu sistema for Windows ou `eclipse` se o seu sistema for GNU/Linux.

Figura B.1: Página de download do Eclipse

Ao iniciar, o Eclipse irá pedir para selecionar uma pasta para armazenar seu *workspace*. Selecione a pasta que desejar e clique em OK.

O próximo passo, será a instalação do plugin do Eclipse para integração com o Maven. Para prosseguir, clique na opção de menu "Help" > "Install new software...". Irá aparecer a janela mostrada na figura B.2 e então clique no botão "Add". Na nova janela digite no campo "Location" a URL `http://m2eclipse.sonatype.org/update/` e clique em "OK", como mostrado na figura B.3.

Selecione todos os plugins, exceto o "Maven SCM handler for Subclipse" e "Maven Integration for AJDT", como mostra a figura B.4. Caso sua instalação do Eclipse possua o Subclipse ou o ApectJ Development Tooling (AJDT), você poderá deixar marcada as respectivas opções. Terminada a seleção, clique en "Next". Prossiga pelo assistente e aceite os termos da licença e clique em "Finish". É possível que o Eclipse apresente uma mensagem pedindo a

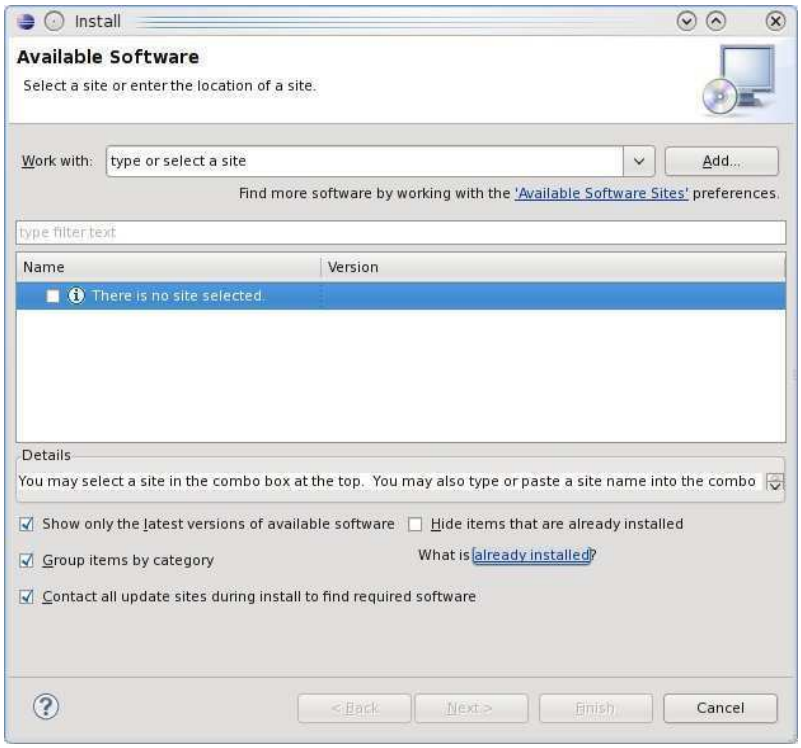

Figura B.2: Janela de instalação de plugins do Eclipse

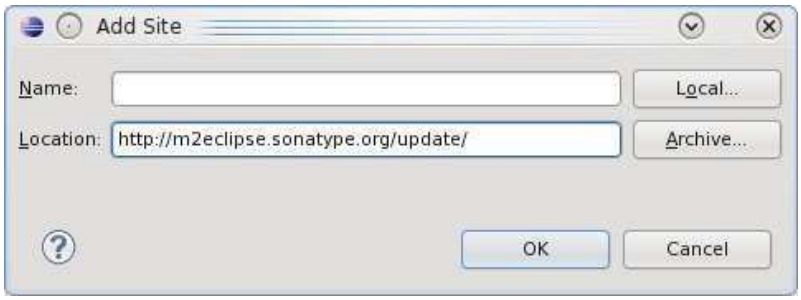

Figura B.3: Adicionando o site de instalação do Plugin do Maven.

confirmação para instalar um conteúdo não-assinado. Se isto ocorrer, permita que o conteúdo seja instalado.

Prossiga criando um novo projeto através da opção de menu "File" > "New" > "Other", ou pressionando a tecla `Ctrl+N`. Como no projeto NCL-Inspector, também usaremos o Maven para criar e gerenciar o projeto. Por isso, devemos criar um novo "Maven Project", como mostrado na figura B.5 e clique em "Next".

No passo seguinte, marque a opção "Create a simple project (skip archetype selection)" e clique novamente em "Next". A seguir aparecerá uma tela com diversos campos a serem preenchidos, a descrição do significado de cada campo está a seguir.

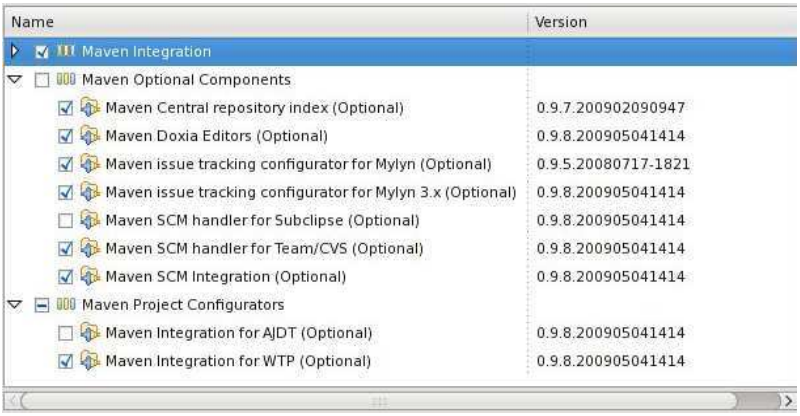

Figura B.4: Seleção dos plugins da integração com o Maven que serão instalados.

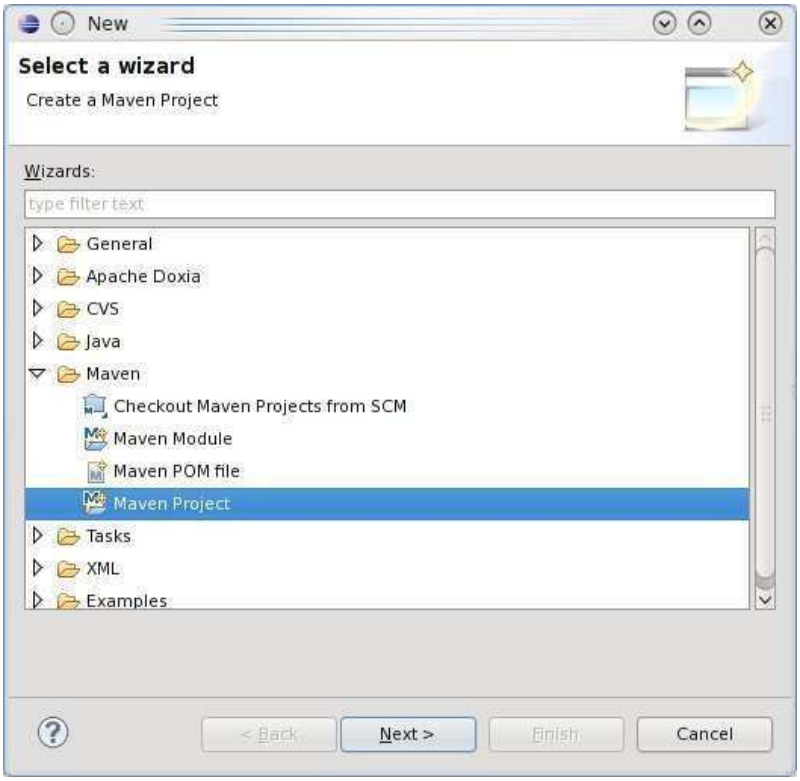

Figura B.5: Criando um novo projeto Maven

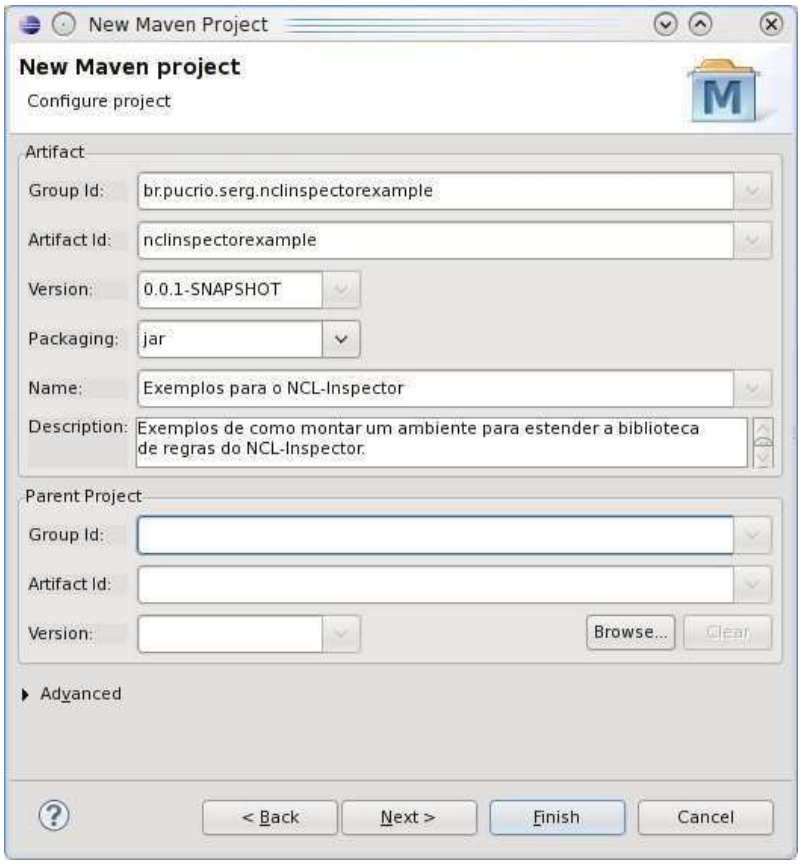

Figura B.6: Tela de configuração do projeto Maven

GroupId é um identificador de uma coleção de módulos relacionados. É comumente uma hierarquia que se inicia com a organização que produziu esses módulos. Muito parecido (ou idêntico) ao sistemas de pacotes de Java.

ArtifactId é um identificador único para os módulos de um grupo.

Version é usado para identificar uma *release* ou número do *build* de um projeto.

Packaging é o formato do artefato que será gerado. Nosso caso, somente faz sentido ser formato `jar`.

Name é um nome mais amigável para o projeto.

Description como o próprio nome já diz, é uma breve descrição do projeto.

Preencha os campos desta tela de acordo com a figura B.6. Com o Maven é possível estruturar os módulos de forma hierárquica, porém este projeto possui apenas um módulo, portanto, ignore o "Parent Project". Clique em "Finish" e finalize a criação do projeto.

Dentro do projeto criado é possível ver um arquivo chamado `pom.xml` (Project Object Model) com o conteúdo apresentado na listagem B.1. Esse arquivo fornece subsídios para o Maven gerenciar o projeto.

```
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="
    http://www.w3.org/2001/XMLSchema-instance"
2   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
      maven.apache.org/maven-v4_0_0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4   <groupId>br.pucrio.serg.nclinspectorexamples</groupId>
5   <artifactId>nclinspectorexamples</artifactId>
6   <name>Exemplos para o NCL-Inspector.</name>
7   <version>0.0.1-SNAPSHOT</version>
8   <description>Exemplos de como montar um ambiente para
      estender a biblioteca de regras do NCL-Inspector.</
      description>
9 </project>
```

Listagem B.1: pom.xml gerado.

Para escrever regras em Java, é necessário incluir o componente nclinspector-engine como dependência do projeto. Isso é feito adicionando o código mostrado na listagem B.2 após o elemento `<description>` do arquivo `pom.xml` (linha 8 da listagem B.1).

Note que o número da versão, especificado pelo elemento `<version>` deve ser correspondente ao número de versão do código fonte compilado no apêndice A. Para descrobrir qual a versão que foi compilada, veja o arquivo `pom.xml` na pasta que contém os fontes do NCL-Inspector.

```
1 <dependencies>
2   <dependency>
3     <groupId>br.pucrio.inf.serg.nclinspector</groupId>
4     <artifactId>nclinspector-engine</artifactId>
5     <version>0.0.1-SNAPSHOT</version>
6     <scope>provided</scope>
7   </dependency>
8 </dependencies>
```

Listagem B.2: Trecho de código para adicionar dependência do nclinspector-engine.

B.2
Testando as regras

B.3
Fazendo a implantação de novas regras

Para fazer a implantação basta rodar o Maven a partir do Eclipse utilizando o *goal* package, como mostra a figura B.7. Ao término, um arquivo `jar` será criado dentro do diretório `target`. Copie esse `jar` para o diretório `rulelib` dentro da distribuição binária da versão de linha de comando do NCL-Inspector.

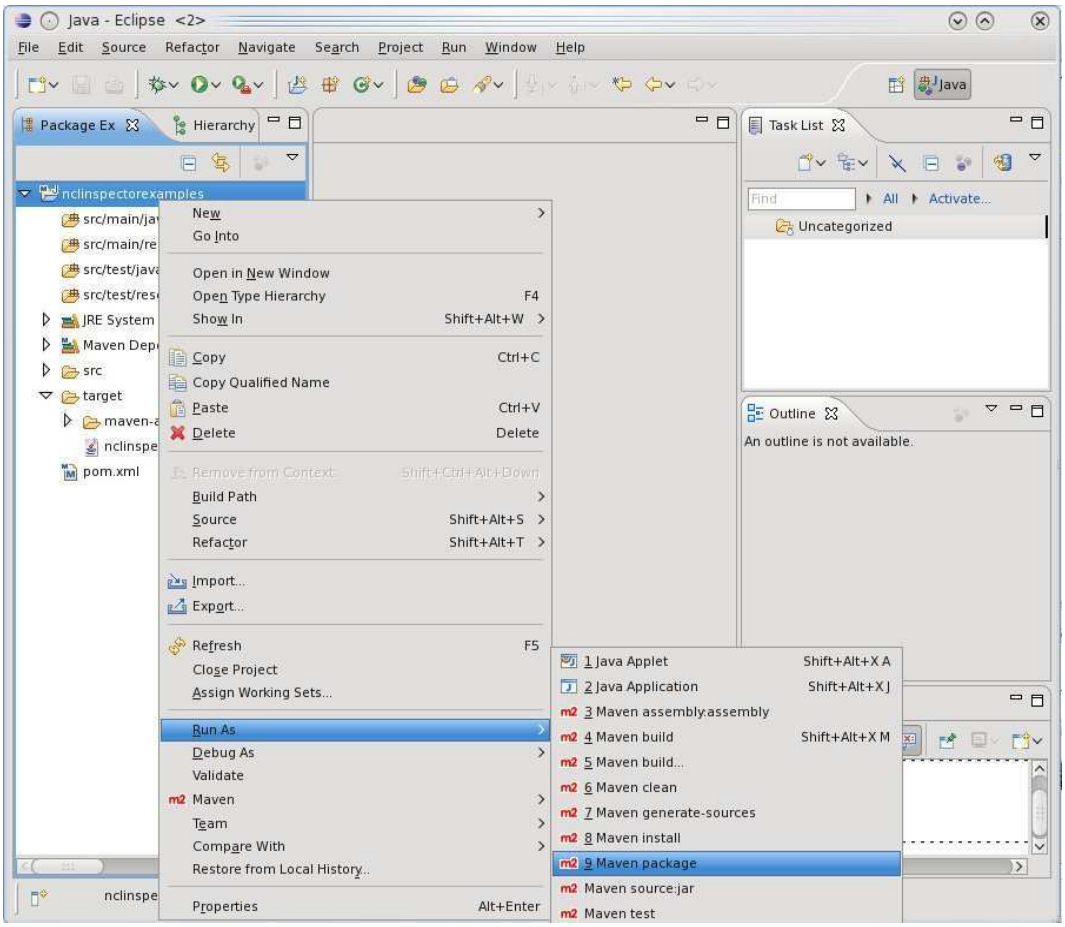

Figura B.7: Gerando um pacote jar para implantação das regras.

Após isso, é necessário criar o arquivo de configuração do NCL-Inspector. Se já tiver sido criado, basta adicionar as entradas das regras que deverão ser utilizadas, como mostrado na seção 4.2.4, página 40.

C
Questionário Pré-Avaliação do NCL-Inspector

Qual é o seu nível de conhecimento nas seguintes tecnologias? E há quanto tempo as utiliza?

Tecnologia	Nível de conhecimento	Tempo de utilização
Java	Especialista ( ) ( ) ( ) ( ) ( ) Novato	___ anos e ___ meses
XML	Especialista ( ) ( ) ( ) ( ) ( ) Novato	___ anos e ___ meses
XSL	Especialista ( ) ( ) ( ) ( ) ( ) Novato	___ anos e ___ meses
NCL	Especialista ( ) ( ) ( ) ( ) ( ) Novato	___ anos e ___ meses
Eclipse	Especialista ( ) ( ) ( ) ( ) ( ) Novato	___ anos e ___ meses

Quantas aplicações você já desenvolveu utilizando as seguintes tecnologias?

Tecnologia	Número de aplicações					
Java	( )20+	( )15-19	( )10-14	( )6-10	( )1-5	( )0
XML	( )20+	( )15-19	( )10-14	( )6-10	( )1-5	( )0
XSL	( )20+	( )15-19	( )10-14	( )6-10	( )1-5	( )0
NCL	( )20+	( )15-19	( )10-14	( )6-10	( )1-5	( )0
Eclipse	( )20+	( )15-19	( )10-14	( )6-10	( )1-5	( )0

Como você adquiriu seu conhecimento em NCL? (Marque todas as aplicáveis)

Item	**Percentual lido/estudado de cada item**
Lendo a norma da ABNT	100%( )( )( )( )0%
Lendo o livro de NCL	100%( )( )( )( )0%
Lendo o tutorial de NCL disponível em www.ncl.org.br	100%( )( )( )( )0%
Estudando as aplicações do Clube NCL	100%( )( )( )( )0%
Outras fontes. Quais?	100%( )( )( )( )0%

Caso já tenha elaborado alguma aplicação NCL, que ferramentas costuma utilizar? (Marque todas as aplicáveis)

( ) Composer
( ) NCL-Eclipse
( ) Editores de texto avançados (e.g. Text-Mate, Emacs, vi)
( ) Editores de texto básicos (e.g. Bloco de notas)
( ) Nunca elaborei uma aplicação NCL
( ) Outros. Quais?

Caso já tenha elaborado alguma aplicação em Java, que ferramentas costuma utilizar pra realizar os testes? (Marque todas as opções aplicáveis)

( ) JUnit
( ) TestNG
( ) Não utilizo as ferramentas ou realizo testes
( ) Outros. Quais?

Caso já tenha elaborado alguma aplicação em Java, indique quais técnicas você costuma utilizar na construção desses programas (Marque todas as opções aplicáveis)

Técnica	**Uso**
Classes ou métodos abstratos	Frequentemente( ) Às vezes ( ) Nunca ( ) Não conheço( )
Generics (Templates)	Frequentemente( ) Às vezes ( ) Nunca ( ) Não conheço( )
Reflexão e Introspecção	Frequentemente( ) Às vezes ( ) Nunca ( ) Não conheço( )
Classes Internas (Inner Classes)	Frequentemente( ) Às vezes ( ) Nunca ( ) Não conheço( )
Uso de Interfaces	Frequentemente( ) Às vezes ( ) Nunca ( ) Não conheço( )

D
Exemplo da Descrição do Problema usado no Experimento

D.1
Descrição

A norma NCL indica que elementos `<media>` que possuem o valor do atributo `type` igual a `application/x-ginga-NCL` não podem ter definidos os atributos `instance` e `refer`. Além disso, o elemento `<media>` deste tipo, não pode definir elementos filhos, isto é, aninhados.

D.2
Exemplos de elementos media que violam essa restrição

```
1 <media id="mirrorProg1" type="application/x-ginga-NCL" src="
    prog1.ncl" descriptor="descPDA" refer="xpto" instance="
    instSame" />
```

Listagem D.1: Exemplo 1

```
1 <media id="mirrorProg1" type="application/x-ginga-NCL" src="
    prog1.ncl" descriptor="descPDA">
2    <area id="a1" begin="1s" end="3s" />
3    <area id="a2" begin="6s" end="9s" />
4 </media>
```

Listagem D.2: Exemplo 2

```
1 <media id="mirrorProg1" type="application/x-ginga-NCL" src="
    prog1.ncl" descriptor="descPDA" refer="xpto" instance="
    instSame">
2    <area id="a1" begin="1s" end="3s" />
3    <area id="a2" begin="6s" end="9s" />
4 </media>
```

Listagem D.3: Exemplo 3

E
Ficha de Apoio à Observação de Uso

Qual a tecnologia escolhida para elaborar a regra?

( ) Java ( ) XSL

Quais dúvidas surgiram? Como foram esclarecidas?

_____

_____

_____

_____

_____

Sober a conclusão do teste

( ) Conseguiu com apenas uma leitura
( ) Conseguiu depois de mais de uma leitura
( ) Conseguiu depois do avaliador apontar no manual onde estava a resposta
( ) Conseguiu depois do avaliador dar mais explicações
( ) Não conseguiu
Detalhes:

_____

_____

_____

Observações gerais

F
Roteiro da Entrevista Pós-Avaliação do NCL-Inspector

F.1
Por que optou por utilizar a tecnologia escolhida (XSL ou Java)?

F.2
O que achou do mecanismo de regras?

F.2.1
O que achou mais fácil?

F.2.2
O que achou mais complicado ou chato?

F.3
O que você modificaria no processo de criação de regras?

F.4
Compare o esforço de elaboração de uma regra com o de inspeção manual dos programas NCL?

Obs: Se mencionar que "para uma aplicação grande vale a pena implementar uma regra" deve ser perguntado: (1) o que vem a ser uma aplicação grande? (2) Tem algum exemplo em mente de uma aplicação grande? (3) Tem alguma métrica para avaliar uma aplicação grande (no. de elementos, contextos, elos, linhas) ?

F.5
Você teria alguma sugestão de caráter geral?

]

G
Problemas encontrados no XML Schema do NCL

#	Descrição	Como foi contornada
1	A informação dada pelo XML Schema é que o atributo `id` do elemento `<ncl>` é requerido. No entanto, na norma é informado que este atributo é opcional.	Foi adotado o que foi apresentado na norma, ou seja, o atributo `id` do elemento `<ncl>` é opcional.
2	No XML Schema o elemento `<regionBase>` possui um atributo `type`. Na norma, este atributo não existe.	Foi adotado determinado pela norma, ou seja, para o elemento `<regionBase>` o atributo `type` não existe.
3	No arquivo `NCL30KeyNavigation.xsd` existe um aparente erro de digitação dos atributos `focusSrc` e `focusSelSrc`. Esses foram digitados como `focusScr` e `focusSelScr`.	Foi corrigido este problema e adotado o nome do atributo que foi entendido como correto (i.e. `focusSrc` e `focusSelSrc`).
4	Em `NCL30CausalConnectorFun...` existe uma quebra de linha na URL dos módulos importados que não está permitindo que o Schema seja validado corretamente.	
5	Aparentemente o `focusIndex` do elemento `<descriptor>`, de acordo com o tutorial, deveria permitir uma `xs:string` como valor, mas no schema está permitindo apenas `xs:positiveInteger`. Isto também se aplica aos atributos `moveRight`, `moveLeft`, `moveUp`, `moveDown` do elemento `<descriptor>`.	No parser implementado por mim, considerei o tipo do valor de `focusIndex` como `xs:string`.

Tabela G.1: Problemas encontrados no XML Schema da norma com data de 14/04/2009

H
Catálogo de problemas de código NCL

Durante a execução desse trabalho, foi necessário ter um conhecimento maior dos problemas de código que ocorrem em programas NCL. A partir de uma análise detalhada da Norma ABNT da NCL e do livro de NCL (Soares e Barbosa 2009), foi feito um levantamento de quase uma centena de problemas de código NCL. Todos esses problemas aqui descritos não são detectados por uma validação baseada no XML Schema. Os problemas estão divididos de acordo com os módulos da linguagem NCL.

H.1
Módulo Structure Functionality

Neste módulo, nenhum problema de código foi encontrado que não pudesse ser detectado a partir de uma validação baseada no XML Schema.

H.2
Módulo Layout functionality

1. O valor do atributo `device` de `<regionBase>` deve ser `"systemScreen(i)"` ou `"systemAudio(i)"`.

   Referência: p.34§4

2. O atributo `region` de `<regionBase>` deve ser formado de acordo com a norma (vide referência).

   Referência: p.34§6

3. Os atributos `left`, `right`, `top`, `bottom`, `height`, `width` e `zIndex` de uma `<region>` devem estar formados de acordo com a norma (vide referência).

   Referência: p.34§9

4. Quando uma `<region>` não especifica atributos de tamanho e posicionamento (`left`, `right`, `height`, `width`, etc), é assumido que essa `<region>` terá o mesmo tamanho e posição da `<region>` pai. Caso seja uma

`<region>` raiz, seu tamanho será igual a toda área do dispositivo de apresentação.

Referência: p.35§1

5. Quando uma region especifica simultaneamente os atributos `top`, `bottom` e `height` ou `left`, `right` e `width` pode ocasionar em inconsistências.

   Referência: p.35§2

   Estado: Terminado

6. Uma `<region>` filha não pode estar fora de uma área estabelecida pela `<region>` pai.

   Referência: p.35§2,

7. Quando dois elementos `<region>` sobrepostas, isto é com o mesmo `zIndex`, e com a mesma ordem temporal irá ocasionar uma `<region>` aparecer sobre a outra de forma aleatória.

   Referência: p.35§3

8. Caso não seja especificado um valor do atributo `zIndex`, para um elemento `<region>`, será atribuído a ele um valor 0 como padrão.

   Referência: p.35§3

H.3
Módulo Components functionality

1. A URI do atributo `src` de um elemento `<media>` devem estar bem formadas de acordo com a tabela (vide referência).

   Referência: p.36 tab.11

2. A referências feitas pelo atributo `src` de um elemento `<media>` devem existir. (e.g. Verificar se o arquivo referenciado por src existe)

   Referência: Não possui

3. Não pode haver mais de um elemento `<media>` com atributo `type="application/x-ginga-settings"`.

   Referência: p.36§ultimo

   Estado: Terminada

4. As variáveis de ambiente reservadas do elemento `<media>` do `type="x-ginga-settings"` devem estar bem formadas, de acordo com a tabela. (vide referência)"

Referência: p.37 tab. 12

5. Não pode haver mais de um elemento `<media>` com atributo `type="application/x-ginga-time"`.

   Referência: p.40§1

   Estado: Terminada

6. O conteúdo de `<media> type="application/x-ginga-time"` deve ser validado de acordo com o parágrafo da norma (vide referências).

   Referência: p.40§2

7. O atributo `type` de `<media>` deve ser um mimetype válido de acordo com a tabela (vide referências).

   Referência: p.41 tab.13

8. O atributo `type` de um elemento `<media>` é opcional, mas uma vez que é usado, o mimetype deve corresponder a extensão do arquivo especificado no atributo `src`.

   Referência: p.41 tab.13

9. Um elemento `<media>` com o atributo `type="application/x-ginga-NCL"` ou `"application/x-NCL-NCL"` não podem ter atributos `instance` e `refer`. Também não pode ter elementos filhos.

   Referência: p.41§4

10. Não pode haver atributos `refer` e `instance` para um elemento `<media>` com o `type` com o valor `"application/x-ginga-settings"` e `"application/x-ginga-time"`.

    Referência: Não possui

11. Um elemento `<context>` não pode conter como valor do atributo `refer` um valor de `id` de um `<context>` pai ou ele próprio.

    Referência: Livro pág. 163

12. Os elementos `<media>` com `type="application/x-ginga-time"`, `"application/x-ginga-settings"`, `"application/x-NCL-time"`, `"application/x-NCL-settings"` não devem possuir o atributo `src`.

    Referência: Não possui

H.4
Módulo Interfaces Functionality

1. Um elemento `<area>` de uma `<media` definida com atributo `type="application/x-ginga-time"` ou `"application/x-NCL-time"` deve possuir sempre os atributos `begin` e `end`.

   Referência: p.42§3

2. Um elemento `<area>` de uma `<media>` definida com atributo `type="application/x-ginga-time"` ou `"application/x-NCL-time"` deve ser sintaticamente bem formado de acordo com UTC.

   Referência: p.42§4 e 5 (nota 1 e 2)

3. Um elemento `<area>` deve possuir os atributos `text` e `position` apenas se o elemento `<media>` pai for do tipo texto.

   Referência: Livro pág. 168 e 169

4. Os atributos `begin`, `end`, `first`, `last`, `position`, `coords`, etc. do elemento `<area>` devem ser bem formados (vide referência).

   Referência: Livro pág. 168 e 169

5. Deve haver ao menos um elemento `<port>` aninhado em cada elemento `<context>` e `<body>`.

   Referência: Livro pág. 165

6. O atributo `component` de um elemento `<port>` deverá referenciar apenas um `id` de um elemento `<media>` ou `<context>`.

   Referência: Livro pág. 164

7. O atributo interface do elemento `<port>` segue a regra: se o atributo `component` for um elemento `<media>`, o atributo `interface` deverá ser um identificador de um elemento `<area>` ou o valor de um atributo `name` de uma `<property>`. Caso o atributo `component` seja um elemento `<context>`, o atributo `interface` deverá ter um identificador de um elemento `<port>`. Porém esse elemento é opcional.

   Referência: Livro pág. 164

8. O elemento `<property>` com o atributo `name` reservados, devem possuir seus atributos `value` validados de acordo com a tabela.

   Referência: p.45 tab.20

9. O atributo `name` de um elemento `<property>` não deve conter valores repetidos para um mesmo elemento `<media>`.

   Referência: Não possui

   Estado: Terminada

10. O "group" property usados em conjunto com suas respectivas "single" property pode gerar inconsistências. Em outras palavras o valor `location` não pode ser usada com `left` e `top`, o valor `size` não pode ser usada com `width` e `height` e `bounds` não pode ser usada com `top`, `left`, `width` e `height` para uma mesmo elemento `<media>`.

    Referência: p.43§ult.

11. Mesmo quando o elemento `switch` seleciona objetos inteiros, é uma boa prática identificar esse tipo de seleção explicitamente através de elemento `<switchPort>`.

    Referência: Livro pag. 228

12. O atributo `component` do elemento `<mapping>` aninhado com `<switchPort>` deve ter um identificador de um elemento `<media>`, `<context>` ou `<switch>` internos ao `<switch>` que o contém.

    Referência: Não possui

13. O atributo interface do elemento `<mapping>` aninhado com o elemento `<switchPort>` deve ter um identificador de um elemento `<area>`.

    Referência: Não possui

14. O atributo `coords` do elemento `<area>` só deve estar presente se o elemento `<media>` pai for uma mídia visual (e.g. vídeo, imagem)

    Referência: Livro pág. 169

H.5
Módulo Presentation Specification Functionality

1. O `<descriptor>` de uma `<media>` do tipo `"application/x-ginga-NCL"` ou `"application/x-NCL-NCL"` não devem ter o attributo `player`.

   Referência: p.44§9

2. Os atributos de `<descriptor>` devem ser validados de acordo com a norma (vide referências).

   Referência: (p.44§8) (p.46§1)

3. É possível redefinir os atributos de uma `<region>` pelo `<descriptorParam>`. Os atributos value desses `<descriptorParam>` devem ser validados (vide referências).

   Referência: p.44§10

4. Os `<descriptorParam>` com o atributo `name` reservados, devem possuir seus atributos `value` validados de acordo com a tabela (vide referência).

   Referência: p.45 tab.20

   Estado: Testes pendentes

5. O atributo `name` de um `<descriptorParam>` não deve conter valores repetidos para um mesmo `<descriptor>`.

   Referência: Não possui

   Estado: Terminada

6. O "group"`<descriptorParam>` usados em conjunto com suas respectivas "single"`<descriptorParam>` pode gerar inconsistências. Em outras palavras o valor `location` não pode ser usada com `left` e `top`, o valor `size` não pode ser usada com `width` e `height` e `bounds` não pode ser usada com `top`, `left`, `width` e `height` para uma mesmo elemento `<descriptor>`.

   Referência: Não possui

H.6
Módulo Linking Functionality

1. O atributo `xconnector` do elemento `<link>` deve ser bem formado de acordo com a norma. (e. g. avisar que se um elemento `<link>` referenciar um `xconnector` inexistente, esse `<link>` será ignorado) (vide referência)

   Referência: p.46§ult.

2. O atributo `role` do elemento `<bind>` deve estar definido no `xconnector` o qual o elemento `<link>` que o contém faz referência.

   Referência: p.47§ 1 e 2

3. O atributo `component` de um elemento `<bind>` deve referenciar apenas um elemento `<media>` ou `<context>`.

   Referência: Não possui

4. O atributo `interface` do elemento `<bind>` deverá ser validado seguindo a regra: se o atributo `component` for um elemento `<media>`, o atributo `interface` deverá ser um identificador de um elemento `<area>` ou o valor de um atributo `name` de uma `<property>`. Caso o atributo `component` seja um elemento `<context>`, o atributo `interface` deverá ter um identificador de um elemento `<port>`. Porém esse elemento é opcional.

   Referência: Não possui

5. Os atributos `name` de `<linkParam>` e `<bindParam>` se referenciam a parâmetros definidos no connector referenciado pelo atributo `xconnector` do elemento `<link>` que o contém.

   Referência: p.47§3

6. Todos os `<connectorParam>` de um `<causalConnector>` devem ser utilizados no elemento `<link>` que o referencia.

   Referência: Não possui

H.7 Módulo Connectors Functionality

1. O atributo `role` do elemento `<simpleCondition>` deve ser único para o conjunto de roles de um elemento `<causalConnector>`.

   Referência: p.49§ ult.

   Estado: Faltando testes

2. Os atributos `eventType` e `transition` do elemento `<simpleCondition>` são opcionais se o `role` possui um valor reservado (e.g. `onBegin`, `onEnd`, etc), caso contrário, deverão ser especificados.

   Referência: p.49§ ult.

3. O atributo `key` do elemento `<simpleCondition>` só deve existir se o valor do atributo `eventType` for igual a `"selection"` (e.g. `onSelection`). Vide tabela 24.

   Referência: p.50 § 1

4. Validar o atributo `key` do elemento `<simpleCondition>` para aceitar apenas os valores válidos de acordo com a norma.

   Referência: p.50 § 1

5. O atributo `min` (que indica cardinalidade) do elemento `<simpleCondition>` deve ser validado: deve ser um inteiro maior que zero e menor ou igual ao valor especificado pelo atributo `max`. O atributo `min` não aceita valores `"unbounded"`.

   Referência: p.50 § 2

6. O atributo `max` (que indica cardinalidade) do elemento `<simpleCondition>` deve ser validado: deve ser maior ou igual ao atributo `min` e aceita o valor `"unbounded"`.

   Referência: p.50 § 2

7. Caso o elemento `<simpleCondition>` declarado possua uma cardinalidade maior que um (i.e. o atributo `min` ou `max` forem maiores do que 1) deverá ser obrigatoriamente especificado o atributo `qualifier`.

   Referência: p.50 § 2

8. O número de elementos `<bind>` para um determinado valor de `role` em um `<link>` deverá respeitar os valores definidos para os atributos `min` e `max` do elemento `<simpleCondition>` que define aquele `role`.

   Referência: Não possui

9. O atributo `role` do elemento `<simpleAction>` deve ser único para o conjunto de roles de um elemento `<causalConnector>`.

   Referência: p.51 § 2

   Estado: Faltando testes

10. Os atributos `eventType` e `transition` do elemento `<simpleAction>` são opcionais se o `role` possui um valor reservado (e.g. `onBegin`, `onEnd`, etc), caso contrário, deverão ser especificados.

    Referência: p.51 § 3

11. Se o valor do atributo `eventType` do elemento `<simpleAction>` é `"attribution"` (e.g. `set`), o elemento `<simpleAction>` deve especificar o valor da atribuição através do elemento `value`. (Vide tabela 26 e referências).

    Referência: p.51 § 3

12. O atributo `min` (que indica cardinalidade) do elemento `<simpleAction>` deve ser validado: deve ser um inteiro positivo, maior que zero e menor que o valor especificado pelo atributo `max`. O atributo `min` não aceita valores `"unbounded"`.

Referência: p.51 § penult.

13. O atributo `max` (que indica cardinalidade) do elemento `<simpleAction>` deve ser validado: deve ser maior que o atributo `min` e aceita o valor `"unbounded"`.

    Referência: p.51 § penult.

14. Caso o elemento `<simpleAction>` declarado possua uma cardinalidade maior que um (i.e. o atributo `min` ou `max` forem maiores do que 1) deverá ser obrigatoriamente especificado o atributo `qualifier`.

    Referência: p.51 § penult.

15. O elemento `<simpleAction>` só deve possuir os atributos `animation` e `by` caso valor do atributo `eventType` for `"attribution"`.

    Referência: p.52 § 1

16. O atributo `role` do elemento `<attributeAssessment>` deve ser único para o conjunto de roles de um elemento `<causalConnector>`.

    Referência: p.52 § 5

    Estado: Faltando testes

17. O atributo `key` de `<attributeAssessment>` só deve estar definido se o valor do atributo `eventType` for igual a `"selection"` (e.g. onSelection). Vide tabela 24.

    Referência: p.52 § 6

18. O atributo `key` do elemento `<attributeAssessment>` deve ser validado de acordo com a norma (vide referência).

    Referência: p.52 § 6

19. Se o valor do atributo `eventType` de `<attributeAssessment>` for `"presentation"` o valor do `attributeType` deverá `"occurrences"`, `"repetition"` ou `"state"`.

    Referência: p.52 § 6

20. Se o valor do atributo `eventType` de `<attributeAssessment>` for `"selection"` o atributo `attributeType` pode ser opcional, mas se estiver presente, deverá possui os valores `"occurrences"` ou `"state"`.

    Referência: p.52 § 6

21. Se o valor do atributo `eventType` de `<attributeAssessment>` for `"attribution"` o atributo `attributeType` pode ser opcional, mas se estiver presente, deverá possui os valores: `"nodeProperty"`, `"occurrences"`, `"repetition"` ou `"state"`.

    Referência: p.52 § 6

22. O atributo `name` do elemento `<connectorParam>` deve ser único para o conjunto de names de um elemento `<causalConnector>`.

    Referência: Não possui

    Estado: Faltando testes

23. Um elemento `<causalConnector>` não pode possuir elementos filhos `<simpleAction>` e `<compoundAction>` simuntâneamente.

    Referência: Não possui

24. Um elemento `<causalConnector>` não pode possuir elementos filhos `<simpleCondition>` e `<compoundCondition>` simuntâneamente.

    Referência: Não possui

H.8 Módulo Presentation Control Functionality

1. O atributo `var` de um elemento `<rule>` deve ter o mesmo valor do atributo `name` de um elemento `<property>` filho de um elemento `<media>` `type="application/x-ginga-settings"` e `"application/x-NCL-settings"`.

    Referência: p.54§ 4

2. O atributo `rule` do elemento `<bindRule>` deve conter um identificador de um elemento `<rule>` ou `<compositeRule>`.

    Referência: Não possui

3. O atributo `constutient` de `<bindRule>` deve conter um identificador de um elemento `<media>`, `<context>` ou `<switch>` internos ao `<switch>` onde está.

    Referência: Não possui

4. O elemento `<defaultComponent>` de um `<switch>` não for definido e caso nenhum elemento `<bindRule>` definido nesse `<descriptorSwitch>` seja avaliado como `true`, nenhum componente será selecionado e o formatador irá tratar como se o componente não existisse.

Referência: Guia de Operações p.39§4

5. O atributo `component` de um elemento `<defaultComponent>` deverá referenciar apenas um `id` de um elemento `<media>`, `<context>` ou `<switch>` internos ao `<switch>` de onde está.

   Referência: Não possui

6. O elemento `<defaultDescriptor>` de um `<descriptorSwitch>` não for definido e caso nenhum elemento `<bindRule>` definido nesse `<switch>` seja avaliado como `true`, nenhum componente será selecionado e o formatador irá tratar como se o componente não existisse.

   Referência: Não possui

H.9
Módulo Timing Functionality

Neste módulo, nenhum problema de código foi encontrado que não pudesse ser detectado a partir de uma validação baseada no XML Schema.

H.10
Módulo Reuse Functionality

1. O atributo `alias` do elemento `<importBase>` deve ter um valor único em todo o documento.

   Referência: p.56§ 5

   Estado: Faltando testes

2. O valor do atributo `alias` de `<importBase>` tem escopo de documento.

   Referência: p.56§ 5

3. O arquivo do documento NCL referenciado pelo atributo `documentURI` do elemento `<importBase>` deve existir.

   Referência: Não possui

4. O atributo `region` do elemento `<importBase>` só deve existir se esse for filho de uma `<regionBase>`.

   Referência: Não possui

5. O valor do atributo `region` do elemento `<importBase>` deve ser um `id` de um elemento `<region>` do documento que fez a importação.

   Referência: p.56§ ult.

6. Os documentos referenciados pelo atributo `documentURI` dos elementos `<importBase>` e `<importNCL>` devem ser validados se existem.

   Referência: Não possui

7. Um atributo `refer` nunca poderá conter um `id` de outro elemento que também possui um `refer`. Em outras palavras, `refer` de `refer` não é permitido.

   Referência: p.57§ 6

8. Um elemento `<media>` e `<switch>` só poderão ter um atributo `refer` referenciando um `id` de um elemento `<media>` e `<switch>` respectivamente. Os elementos referidos deverão estar definidos no `<body>` do documento corrente ou em um `<body>` externo exportado.

   Referência: p.57§ penult.

9. Um elemento `<context>` e `<body>` só poderão ter um atributo `refer` referenciando um `id` de um elemento `<context>` ou `<body>`. Os elementos referidos deverão estar definidos no `<body>` do documento corrente ou em um `<body>` externo exportado.

   Referência: Guia de Operações p.40§9

10. Quando um elemento define um atributo `refer`, todos os atributos são herdados do elemento referido. Quando o elemento que referencia define atributos ou elementos filhos, estes são ignorados, com exceção de: (1) Atributo `id`; (2) Elementos `<property>` e `<area>` filhos de `<media>`; (3) Atributo `<instance>` de `<media>`.

    Referência: p.58§ 1

11. Se um novo elemento `<property>` adicionado tem o mesmo atributo name de um elemento `<property>` já existente (definido em um elemento `<media>` reusado através do `refer`), o novo elemento `<property>` adicionado deve ser ignorado.

    Referência: p.58§ 2

12. Similarmente ao elemento `<property>`, se um novo elemento `<area>` possui o mesmo atributo `id` de um elemento `<area>` já existente (definida em um elemento `<media>` reusado), o novo elemento `<area>` deverá ser ignorado.

    Referência: p.58§ 2

H.11
Módulo Navigation Key Functionality

1. O valor do atributo `focusIndex` deve ser único em todo o documento.

   Referência: p.59§ 1

   Estado: Faltando testes

2. Os valores dos atributos `moveRight`, `moveLeft`, `moveDown`, `moveTop` devem ser um `focusIndex` válido.

   Referência: p.59§ 2,3

3. O valor do atributo `focusSrc` e `focusSelSrc` do elemento `<descriptor>` deve ser válidado, verificando se o arquivo existe, similar ao atributo `src` do elemento `<media>`.

   Referência: p.59§ 5

4. Os atributos `focusBorderColor`, `focusBorderWidth` e `focusBorderTransparency` do elemento `<descriptor>` devem estar de acordo com a norma (vide referência).

   Referência: p.59§ 6

5. Quando um elemento `<media>` faz referencia a outro elemento `<media>` (usando o atributo `refer`), o atributo `focusIndex` especificado no elemento `<descriptor>` associado com o elemento `<media>` referida será ignorado.

   Referência: Guia de Operações p.42§2

H.12
Módulo Animation Key Functionality

Neste módulo, nenhum problema de código foi encontrado que não pudesse ser detectado a partir de uma validação baseada no XML Schema.