PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 15/06

Nested Context Model 3.0

Part 6 – NCL (Nested Context Language) Main Profile

Luiz Fernando Gomes Soares
Rogério Ferreira Rodrigues
Romualdo Monteiro de Resende Costa

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900
RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 15/06 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena Maio/2006

Nested Context Model 3.0

Part 6 – NCL (Nested Context Language) Main Profile

Luiz Fernando Gomes Soares
Rogério Ferreira Rodrigues
Romulado Monteiro de Resende Costa

Laboratório TeleMídia DI – PUC-Rio
Rua Marquês de São Vicente, 225, Rio de Janeiro, RJ - 22451-900.

lfgs@inf.puc-rio.br, rogerio@telemidia.puc-rio.br, romualdo@telemidia.puc-rio.br

Abstract. *This technical report describes the basic elements and attributes of the Nested Context Language (NCL) Main Profile version 2.3. NCL is an XML language based on the NCM (Nested Context Model) conceptual model for hypermedia document specification.*

Resumo: *Este relatório técnico descreve os elementos e atributos básicos do Perfil Principal da Linguagem NCL (Nested Context Language), em sua versão 2.3. NCL é uma linguagem XML baseada no modelo conceitual NCM (Nested Context Model), concebido para especificação de documentos hipermídia.*

Keywords*: declarative language, hypermedia conceptual model, temporal synchronization, spatial synchronization, compositionality.*

Palavras chave*: linguagem declarative, modelo conceitual hipermídia, sincronização temporal, sincronização espacial, composicionalidade.*

Nested Context Model 3.0

Part 6 – NCL (Nested Context Language) Main Profile

Laboratório TeleMídia

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro

Rua Marquês de São Vicente, 225, Prédio ITS - Gávea

22451-900 – Rio de Janeiro – RJ – Brasil

http://www.telemidia.puc-rio.br

Table of Contents

1. Introduction 6
2. The NCL 2.3 Modularization 9
   2.1. Overview of NCL 2.3 Functional Areas 9
      2.1.1. Structure Functionality 10
      2.1.2. Layout Functionality 10
      2.1.3. Components Functionality 10
      2.1.4. Interfaces Functionality 11
      2.1.5. Presentation Specification Functionality 12
      2.1.6. Linking Functionality 13
      2.1.7. Connectors Functionality 14
      2.1.8. Presentation Control Functionality 15
      2.1.9. Timing Functionality 16
      2.1.10. Composite-Node Templates Functionality 16
      2.1.11. Reuse Functionality 17
      2.1.12. Metainformation Functionality 19
   2.2. NCL 2.3 Modules 19
      2.2.1. Module Elements 19
      2.2.2. NCL 2.3 Modules in XML Schema 25
      Structure Module: NCL23Structure.xsd 25
      Layout Module: NCL23Layout.xsd 26
      Media Module: NCL23Media.xsd 27
      Context Module: NCL23Context.xsd 27
      MediaContentAnchor Module: NCL23MediaContentAnchor.xsd 28
      CompositeNodeInterface Module: NC23CompositeNodeInterface.xsd 29
      AttributeAnchor Module: NCL23AttributeAnchor.xsd 29
      SwitchInterface Module: NCL23SwitchInterface.xsd 30
      Descriptor Module: NCL23Descriptor.xsd 31
      DescriptorBind Module: NCL23DescriptorBind.xsd 32
      Linking Module: NCL23Linking.xsd 32
      TestRule Module: NCL23TestRule.xsd 33
      ContentControl Module: NCL23ContentControl.xsd 34
      DescriptorControl Module: NCL23DescriptorControl.xsd 35
      Timing Module: NCL23Timing.xsd 35

Import Module: NCL23Import.xsd 36
ReuseEntity Module: NCL23ReuseEntity.xsd 37
3. NCL 2.3 Language Profiles 38
3.1. The NCL 2.3 Language Profile 38
3.2. NCL 2.3 Main Profile 38
3.3. NCL 2.3 XConnector Profile 38
3.4. Other NCL 2.3 Language Profiles 38
4. The NCL 2.3 Main Profile XML Schema 40
NCL23.xsd 40
5. Authoring an NCL 2.3 Document: An Example 52
6. The XConnector Profile 60
6.1. The XConnector 2.3 Module Specification in XML Schema 63
XConnector Module: NCL23XConnector.xsd 63
6.2. The XConnector Profile Specification in XML Schema 70
XConnector23.xsd 70
7. Final Remarks 73
References 74
Appendix A: Connector Base for Allen's Relations 76
Appendix B: Another Extensive Connector Base 80

Nested Context Model 3.0

Part 6 – NCL (Nested Context Language) Main Profile

Luiz Fernando Gomes Soares
Rogério Ferreira Rodrigues
Romualdo Monteiro de Resende Costa

Laboratório TeleMídia DI – PUC-Rio
Rua Marquês de São Vicente, 225, Rio de Janeiro, RJ - 22451-900.

lfgs@inf.puc-rio.br, rogerio@telemidia.puc-rio.br, romualdo@telemidia.puc-rio.br

Abstract. *This technical report describes the basic elements and attributes of the Nested Context Language (NCL) Main Profile, version 2.3. NCL is an XML-based language based on the NCM (Nested Context Model) conceptual model for hypermedia document specification.*

1. Introduction

This report describes the entities of the NCL (*Nested Context Language*) Main Profile version 2.3. NCL is an XML language for authoring hypermedia documents, including non-linear TV programs, based on NCM (Nested Context Model) [SoRo05]. The first version of NCL [Anto00, AMRS00] was specified through an XML DTD – Document Type Definition [XML98].

The second version of NCL, named NCL 2.0, was specified using XML Schema [SCHE01], which defines a richer language for specifying document types.

Following recent trends, from version 2.0 on, NCL has been specified in a modular way, allowing the combination of its modules in language profiles. Each profile can group a subset of NCL modules, allowing the creation of languages according to user needs. Moreover, NCL modules can be combined with other language modules, allowing the incorporation of NCL features into those languages and vice-versa.

Besides the modular structure, NCL 2.0 introduced new facilities to its previous version 1.0, among others:

- definition of hypermedia connectors and connector bases;
- use of hypermedia connectors for link authoring;
- reuse of links in different documents;
- definition of ports and maps for composite nodes, satisfying the document compositional property;
- definition of hypermedia composite-node templates, allowing the specification of constraints on documents;
- definition of composite-node template bases;
- use of composite-node templates for authoring composite nodes;
- refinement of document specifications with content alternatives, through the *switch* element, grouping a set of alternative nodes;
- refinement of document specifications with presentation alternatives, through the *descriptorSwitch* element, grouping a set of alternative descriptors;
- use of a new spatial layout model.

NCL 2.1 brought some refinements to the previous version: a module for defining cost functions associated with media object duration was introduced; a module aiming at describing the selection rules of switch and descriptor switch was defined; and refinements in some NCL modules were made, mainly in the XTemplate module.

NCL 2.2 made minor refinements in some NCL 2.1 modules, concerning their element definitions, and introduced a different approach in defining NCL modules and profiles.

This new version of NCL, called NCL 2.3, introduces two new modules for supporting base and entity reuse, and refines the definition of some elements in order to support the new features.

As aforementioned, NCM is the model underlying NCL. However, in its present version 2.3, NCL does not reflect all NCM 3.0 facilities yet. In order to understand NCL facilities, it is necessary to understand the NCM concepts. With the aim of offering a scalable hypermedia model, with characteristics that can be progressively incorporated in hypermedia system implementations, NCM was divided in several parts:

- Part 1: NCM Core – concerned with the main model entities, which should be present in all NCM implementations[1].
- Part 2: NCM Virtual Entities – concerned mainly with the definition of virtual anchors, nodes and links.
- Part 3: NCM Version Control – concerned with model entities and attributes to support versioning.
- Part 4: NCM Cooperative Work – concerned with model entities and attributes to support cooperative document handling.

The Nested Context Language composes Parts 5 and 6 of the collection:

- Part 5: NCL (Nested Context Language) Full Profile – concerned with the definition of an XML-based language for authoring and exchanging of NCM based documents.
- Part 6: NCL (Nested Context Language) Main Profile – concerned with the definition of an XML-based language for authoring and exchanging NCM-based documents. It comes from the Full Profile without the modules for template use and definition; for composite-connector specification; and for metainformation characterization.

In order to understand NCL, the reading of Part 1: NCM Core is recommended.

[1] It is also possible to have NCM implementations that ignore some of the basic entities, but this is not so relevant as to deserve a minimum-core definition.

This technical report is organized as follows. Section 2 presents an overview of NCL 2.3 modules that compose the NCL Main Profile, and after that it introduces their XML Schemas. Section3 discusses some NCL 2.3 language profiles. The NCL 2.3 Main Profile is presented in Section 4 through its XML Schema. Section 5 gives an example of authoring a NCL document, presenting some of the main NCL elements and their attributes. Section 6 describes, in details, the XConnector Profile, introducing its XML Schemas. Section 7 presents the final remarks.

2. The NCL 2.3 Modularization

NCL 2.3 is partitioned into twelve functional areas, which are partitioned again into modules.

NCL 2.3 functional areas and their corresponding modules are the following:

1. Structure
   a. Structure Module
2. Layout
   a. Layout Module
3. Components
   a. Media Module
   b. Context Module
4. Interfaces
   a. MediaContentAnchor Module
   b. CompositeNodeInterface Module
   c. AttributeAnchor Module
   d. SwitchInterface Module
5. Presentation Specification
   a. Descriptor Module
   b. DescriptorBind Module
6. Linking
   a. Linking Module
7. Connectors
   a. XConnector Module
   b. CompositeConnector Module
8. Presentation Control
   a. TestRule Module
   b. ContentControl Module
   c. DescriptorControl Module
9. Timing
   a. Timing Module
10. CompositeNode Templates
   a. XTemplate Module
   b. XTemplateUse Module
11. Reuse
   a. Import Module
   b. ReuseEntity Module
12. Metainformation
   a. Metainformation Module (identical to SMIL 2.0 Metainformation Module)

2.1. Overview of NCL 2.3 Functional Areas

This section briefly describes the main definitions made by each NCL 2.3 functional area, including CompositeNode Templates and Metainformation functionalities, as well as the compositeConnector module, although they are not present in the NCL 2.3 Main

Profile. The complete definition of NCL 2.3 modules, using XML Schemas, is presented in Section 2.2, with the exception of the modules for template use and definition; for composite-connector specification; and for metainformation characterization.. Sections 4 and 6 present the XML Schemas for the NCL 2.3 Main Profile and XConnector Profile, respectively. Any ambiguity found in this text may be clarified by consulting the XML Schemas.

2.1.1. Structure Functionality

The Structure functionality has just one module, called Structure, which defines the basic structure of an NCL document. It defines the root element, called *ncl*, the *head* and the *body*, following the terminology adopted by other W3C standards. The body element of an NCL document may be treated as a context node, as defined in Section 2.1.3 and 2.1.11, containing other NCM nodes and/or links.

2.1.2. Layout Functionality

The Layout functionality has a single module, named Layout, which specifies elements and attributes that define how objects will be presented inside regions of output devices. Indeed, this module defines initial values for homonym attributes defined in media and context elements, discussed in Section 2.1.3.

In short, a *regionBase* element, which should be declared in the NCL document head, defines a set of *region* elements, which may contain another set of nested *region* elements, and so on, recursively.

The interpretation of the region-nesting meaning should be done by the element in charge of the document presentation orchestration (from now on called *formatter*). Just as an example, the first nesting level could be interpreted as defining a device where the presentation would take place; the second nesting levels as windows (or channels) of the parent device; and the other levels as regions inside these windows (or channels).

Regions can define the following attributes: *id*, *title*, *left*, *right*, *top*, *bottom*, *height*, *width*, *background*, *zIndex*, *decorated*, *movable*, and *resizable*. All those attributes have the usual meaning.

When the user specifies *top*, *bottom* and *height* information for the same region, spatial inconsistencies may occur. In this case, *top* and *height* values should have precedence over *bottom* information. Analogously, when the user specifies inconsistent values for *left*, *right* and *width* region attributes, the *left* and *width* values should be used to compute a new *right* value. When any of these attributes is not specified and cannot have its value computed from the other attributes, it must be assumed as “0”.

2.1.3. Components Functionality

The Components functionality is partitioned into two modules, called Media and Context.

The Media module defines basic media object types. For defining media objects, this module defines the *media* element. Each media object has two main attributes, besides its *id* attribute: *src*, which defines a URI [RFC2396] of the object content, and

type, which defines the object type (*animation*, *audio*, *image*, *text*, *textstream*, *video*, *settings*, *time*, *NCLua*, *NCLet*, *etc.*[2]). The type attribute is optional and should be used to guide the player (presentation tool) choice by the formatter. When the type attribute is not specified, the formatter should use the file extension pointed by the *src* attribute to make the player choice.

The Context module is responsible for the definition of context nodes. An NCM context node is a particular type of NCM composite node and is defined as containing a set of nodes and a set of links.

Media and context nodes have other embedded attributes. Examples of these attributes can be found among those that define the object placement during a presentation and its presentation duration: *top*, *left*, *bottom*, *right*, *width*, *height*, *explicitDur*, etc. Whenever not specified in the media or context element specification, these attributes assume as their initial values those defined in homonym attributes of their node-associated descriptor and region, as discussed in 2.1.2 and 2.1.5.

2.1.4. Interfaces Functionality

The Interfaces functionality allows the definition of node (media object or composite node) interfaces that will be used for link creation. This functionality is partitioned into four modules:

1. MediaContentAnchor, which allows media object content anchor (or area) definition;
2. CompositeNodeInterface, which allows composite node content anchor (compositeNodeArea) and port definitions;
3. AttributeAnchor, which allows the definition of node attributes or node attribute groups as node interfaces; and
4. SwitchInterface, which allows the definition of special interfaces for *switch* elements.

The MediaContentAnchor module defines the *area* element, which extends the syntax and semantics of the homonym element defined by SMIL and XHTML. As such it allows the definition of content anchors representing spatial portions, through the *coords* attribute (as in XHTML); the definition of content anchors representing temporal portions, through *begin* and *end* attributes; and the definition of content anchors representing temporal and spatial portions through *coords*, *begin* and *end* attributes (as in SMIL). In addition, the *area* element allows the definition of textual anchors, through the *text* and *position* attributes that define the string and the string initial position in the text, respectively. Besides, the *area* element can also define a content anchor based on the number of audio samples or video frames, through attributes *first* and *last*, which should indicate the initial and final sample/frame. Moreover, the *area* element can also define a content anchor based on an *anchorLabel*, which specifies a string that should be used by the media player to identify a content region.

The CompositeNodeInterface module defines an element named *compositeNodeArea*, which can be used for creating composite node content anchors. The *compositeNodeArea* element has *areaComponent* child elements, each one specifying a component node *id*. Besides the *compositeNodeArea* element, the CompositeNodeInterface module also defines the *port* element, which specifies a

[2] It is the formatter implementation that defines the types supported. Thus, the formatter specification should be consulted before a document authoring.

composite node port with its respective mapping (*interface* attribute) to an interface of one of its components (specified by the *component* attribute). When a language profile uses the CompositeNodeInterface module, it should include the Context module and/or the ContentControl module to comprise the definition of *context* or *switch* nodes, and should declare *compositeNodeArea* and *port* elements as children of the *context* or *switch* elements.

In NCM every node must have an anchor with a region representing the whole content of the node. This anchor is called the *whole content anchor* and is declared by default in NCL. Every time an NCL component is referenced without specifying one of its anchors, the *whole content anchor* is assumed.

The AttributeAnchor module defines an element named *attribute*, which can be used for defining a node attribute or a group of node attributes as one of its interfaces (anchors)[3]. The *attribute* element has an attribute called *name*, which indicates the attribute, or attribute group, name. When a language profile uses the AttributeAnchor module, it should include the Media or ContextNode modules to comprise the definition of nodes, and should declare *attribute* elements as children of *media* or *composite* objects.

As aforementioned, a context, switch, and media nodes may have several embedded attributes (like *top, left, bottom, right, width, height, explicitDur,* etc.). However, when any of these attributes is used in a relationship it must be explicitly declared as an *attribute* (interface) element. Moreover, a group of node attributes can also be explicitly declared as an *attribute* (interface) element. NCL version 2.3 does not allow the definition of attribute groups, yet. However, the following groups must be recognized by an NCL formatter: *location*, grouping (left, top) in this order; *size*, grouping (width, height) in this order; and *bounds*, grouping (left, top, width, height) in this order. When a formatter treats a change in an attribute group it must only test the process consistency at its end.

The SwitchInterface module allows the creation of *switch* element interfaces, which will be mapped to a set of alternative interfaces of internal nodes, allowing a link to anchor on the component chosen when the *switch* is processed (see [SoRo05]). This module introduces the *switchPort* element, which contains a set of *mapping* elements. A *mapping* element defines a path from the *switchPort* to an interface (*interface* attribute) of one of the switch components (specified by its *component* attribute). When a language profile uses this module, it should declare *switchPort* elements as children of *switch* elements, defined by the ContentControl module, which will be described in Section 2.1.8.

It is important to mention that every element representing an object interface (*area*, *compositeNodeArea*, *port*, *attribute*, and *switchPort*) must have an identifier (*id* attribute).

2.1.5. Presentation Specification Functionality

The Presentation Specification functionality is partitioned in two modules, one named Descriptor and another named DescriptorBind. The purpose of these modules is to

[3] It is possible to have NCL document players (formatters) that define some of the nodes attributes as node interfaces, implicitly. However, in general, it is a good practice to explicitly define the interfaces.

specify temporal and spatial information needed to present each document component. This information is modeled by *descriptor* objects, according to NCM.

The Descriptor module allows the definition of *descriptor* elements, which contain a set of optional attributes, grouping all temporal and spatial definitions, which should be used according to the type of object to be presented. The definition of *descriptor* elements should be done in the document head, inside the *descriptorBase* element, which specifies the set of descriptors of a document.

A *descriptor* element has temporal attributes *explicitDur*, *freeze*, and *costFunction* defined by the Timing module (see Section 2.1.9); an attribute named *player*, which identifies the presentation tool to be used; an attribute named *region*, which refers to a region defined by elements of the Layout module (see Section 2.1.2).

Descriptor elements may have *descriptorParam* child elements, which are used to parameterize the *player* specified by the parent descriptor element. These parameters can, for example, redefine some attribute values defined by the region attributes. They can also define new atributes such as *visible*, allowing the object presentation to be seen or hidden; *fit*, indicating how an object will be presented; *scroll*, which allows the specification of how an author would like to configure the scroll in a region; *controlVisible*, used for enabling control buttons and a time progress bar when needed; *style*, which refers to a style sheet [CSS98] with information for text presentation, for example; and also specific attributes for audio objects, such as *soundLevel*, *balanceLevel*, *trebleLevel* and *bassLevel*. Besides, *descriptorParam* child elements can determine if a new player must be instantiated or if a player already instantiated must be used (*reusePlayer*), and specify what will happen to the player instance at the end of the presentation (*playerLife*).

Besides the descriptor element, the Descriptor module defines a homonym attribute, which refers to an element of the document descriptor set. When a language profile uses the Descriptor module, it has to determine how *descriptors* will be associated with document components. In the case of NCL 2.3 Main Profile, following NCM, the *descriptor* attribute is associated with any media or context node, through *media* and *context* elements, through link endpoint (*bind* element) or through parent composite node (*descriptorBind* element).

The DescriptorBind module defines *descriptorBind* elements, which associate a node contained in a composition, identified by the *component* attribute, with a *descriptor* element, identified by the *descriptor* attribute. Any composite node object may contain *descriptorBind* elements.

It should be remarked that the set of descriptors of a document may contain *descriptor* elements or *descriptorSwitch* elements, which allow specifying alternative descriptors, as it will be presented in Section 2.1.8.

It is also worth to mention that this current version of NCL does not allow specifying event descriptions with the same granularity as defined by NCM. In this current NCL version every event description associated with an anchor should be derived by the formatter from the descriptor attributes.

2.1.6. Linking Functionality

The Linking functionality defines the Linking module, responsible for defining links using connectors. A *link* element has an *id* attribute and an *xconnector* attribute, which

refers to a hypermedia connector URI. The reference would have the format: *alias#tconnector_id*, or *documentURI_value#connector_id*, for connectors defined in an external document (as discussed in Section 2.1.11); or simply *#connector_id*, for connectors defined in the document itself. The *link* element also contains child elements called *binds*, which allow to associate nodes with connector roles. In order to make this association, a *bind* element has three basic attributes. The first one is called *role*, which is used for referring to a connector role. The second one is called *component*, which is used for identifying the node, and the third attribute is called *interface*[4], used for making reference to the node interface. If the connector element defines parameters (as discussed in Section 2.1.7), the *bind* and *link* elements should define parameter values, through child elements called *bindParam* and *linkParam*, respectively, with *name* and *value* attributes following definition of parameter types done by the connector.

When a language profile uses the Linking module, it should include the XConnector module for the definition of connectors.

2.1.7. Connectors Functionality

This functionality is partitioned into two modules, one called XConnector and another called CompositeConnector.

The XConnector module allows the definition of hypermedia connectors (*causalConnector* and *constraintConnector* elements) and *connectorBase* elements. Although XConnector was incorporated as a module of the NCL 2.3 language, it is totally independent from other modules of NCL and can be used together with other languages: [Much03] proposes an extension to the XLink language [XLIN01], [Silv05] an extension to the SMIL 2.0 language [SMIL01], and [Cost05] an extension to XMT-O language [ISO04], all these extensions incorporating XConnector facilities. Thus, XConnector is the core by itself of an XML-based language (another NCL 2.3 profile, as discussed in Section 3) for the definition of connectors, which can be used to specify reference and spatio-temporal hypermedia synchronization relations, treating reference relations as a particular case of synchronization relations. The complete XML Schema definition of XConnector module and profile is presented in Section 6. Any ambiguity found in this text may be clarified by consulting the XML Schema.

The *causalConnector* and *constraintConnector* elements represent relations that can be used for creating links. A hypermedia connector specifies a set of interface points, called roles, and another child element called *glue*, which describe how roles interact.

Some attribute values of connector roles and glue can be parameterized. The definition of connector parameters can be done declaring *connectorParam* child elements. In order to specify which attributes receive parameter values defined by the connector, we should specify their values as the parameter name preceded by the *$* symbol. For instance, in order to parameterize the delay attribute to be waited before executing a link action, we define a parameter called *actionDelay* (<*connectorParam name=“actionDelay” type=“unsignedLong”*/>) and use the value *“$actionDelay”* in the corresponding action-expression attribute (*delay*=“$*actionDelay*”).

[4] Note that the *interface* attribute can refer to any node interface, that is, an anchor, an attribute or a port, if it is a composite node.

Besides the *causalConnector* and *constraintConnector* elements, the XConnector module defines another element named *connectorBase*, which allows the definition of *xconnector* bases. As an example, Appendix A shows a connector base for Allen's relations. Other XConnector elements are discussed in Section 6.

The CompositeConnector module allows the definition of composite connectors. A composite connector is defined by the *compositeConnector* element, which contains *role* elements and a *glue* element, such as a simple connector. Each *role* element represents an interface point of the composite connector, which is mapped to an interface point of an occurrence of an internal connector (called partially-defined link). Thus, each *role* element has an *id* attribute, a *partialLink* attribute and a *role* attribute. The *partialLink* attribute refers to a partially-defined link and the *role* attribute refers to one of the roles of the connector used by this partial link. Besides the set of *role* elements, a composite connector defines a *glue* element. The *glue* of a composite connector is defined with the same content as a composite node, added with a set of partially-defined links, called *partialLinks*.

2.1.8. Presentation Control Functionality

The Presentation Control functionality purpose is to specify content alternatives, and also presentation alternatives for a document. This functional area is partitioned in three modules, named TestRule, ContentControl and DescriptorControl.

The TestRule module allows the definition of rules that, when satisfied, select alternatives for document presentation. The specification of rules in NCL 2.3 was done in a separate module, because they will be useful for defining either alternative components or alternative descriptors.

The *ruleBase* element specifies a set of rules, and should be defined as a child element of the *head* element. These rules can be simple, defined by the *rule* element, or composite, defined by the *compositeRule* element. Simple rules define an identifier (*id* attribute), a variable (*var* attribute), a value (*value* attribute), and an operator (*op* attribute) relating the variable to the value. Composite rules have an identifier (*id* attribute) and a Boolean operator ("and" or "or") relating their child rules.

Finally, the *bindRule* element is used to associate rules to components of a *switch* or *descriptorSwitch* element, through its *rule* and *component* attributes, respectively.

The ContentControl module specifies the *switch* element, allowing the definition of alternative document nodes to be chosen during presentation time. It should be noted that NCL *switch* element has not the same content as the other language homonyms (as for example the switch element defined by SMIL 2.0 BasicContentControl module), because it can contain, besides media objects, *context* elements defined by the Context module of NCL 2.3. Test rules used to choose the switch component to be presented are defined by the TestRules module, as previously mentioned. For this reason, when a language profile uses the ContentControl module, it should include the TestRules module.

In order to allow link anchorings on the component chosen after evaluating the rules of a *switch*, a language profile should also include the SwitchInterface module, which allows the definition of special interfaces, named *switchPort*. Moreover, in order to allow the definition of presentation alternatives for components of a switch node, a language profile should also include the DescriptorBind module, which introduces the

descriptorBind element, allowing its inclusion in a *switch* element, analogous to what is done in *context* elements (see Section 2.1.5).

The DescriptorControl module specifies the *descriptorSwitch* element, which contains a set of alternative descriptors to be associated with an object. Analogous to the *switch* element, the *descriptorSwitch* choice is done during presentation time, using test rules defined by the TestRule module. For this reason, when a language profile uses the DescriptorControl module, it should include the TestRule module, besides the Descriptor module, to comprise the definition of *descriptor* elements.

2.1.9. Timing Functionality

The Timing functionality defines a single module called Timing, which allows the definition of temporal properties for document components. Basically, this module defines attributes for specifying what will happen with an object at the end of its presentation (*freeze*), the ideal duration of an object (*explicitDur*), and the allowed duration variation with its associated cost (*costFunction*). This last attribute must refer to a *costFunction* element, defined in the next paragraph. The three attributes will be used by *descriptor* elements, which group all component presentation specifications, as defined in Section 2.1.5.

In order to define cost functions associated with object durations, the Timing module defines the *costFunction* element. The definition of *costFunction* elements should be done in the document head, inside the *costFunctionBase* element, which specifies the set of cost functions of a document.

The *costFunction* element has an identifier (*id* attribute), a type (*type* attribute) and a reference to a mathML file [W3C03] (*mathMLRef* attribute).

Cost function parameters can be defined for a *costFunction* element, through its child *costFunctionParam* element. Parameters are specified through a *name* attribute and a *value* attribute.

2.1.10. Composite-Node Templates Functionality

The Composite-Node Templates functionality is partitioned into two modules, called XTemplate and XTemplateUse.

The XTemplate module allows the definition of hypermedia composite-node templates (*xtemplate* elements) and *templateBase* elements. A composite-node template specifies types of components, types of relations, components and relationships that a composition has or may have, without identifying the instances of the components and relationships. This task must be carried out by specific hypermedia compositions that use the template. A *templateBase* groups *xtemplate* elements.

XTemplate is the core by itself of another XML-based language (another NCL 2.3 profile, as discussed in Section 3) in charge of providing the definition of hypermedia composite-node templates. The complete definition of XTemplate module and profile using XML Schema, as well as the specification of the NCL 2.3 Full Profile, are presented in [SoRC05]. Any ambiguity found in this text may be clarified by consulting the XML Schema that describes the NCL 2.3 Full Profile.

An *xtemplate* element has a *vocabulary* that defines types of components, types of connectors and types of interfaces. It may also define *constraints* on the vocabulary elements and on the inclusion of instances of components and connectors, representing

relationships among components in the composite node. The defined structure is inherited by a composite node that refers to the template.

The XTemplateUse module allows NCL composite nodes to use templates. This module defines the *xtemplate* attribute, which refers to a template URI, and the *type* attribute, which specifies a type defined by the template. The reference would have the format: *alias#template_id*, or *documentURI_value#template_id*, for templates defined in an external document (as discussed in Section 2.1.11); or simply *#templateid*, for templates defined in the document itself. When a language profile uses this module, it should add *xtemplate* attribute to *context* and *switch* elements and *type* attribute to *context*, *switch* and *Interfaces functionality* elements. It should be remarked that the type attribute of media objects must also be able to reference one of the types defined by the template, when templates are used. As aforementioned, the media type should be used to guide the player (presentation tool) choice by the formatter, however, when a media content type is defined by a template, the corresponding *ctype* attribute in the template component type definition should be used instead.

2.1.11. Reuse Functionality

Following NCM principles, NCL allows intensive reuse of its elements. The NCL Reuse functionality is partitioned into two modules: Import and ReuseEntity.

In order to allow an entity base to incorporate another already defined base, the Import module defines the *importBase* element, which has two attributes: *documentURI* and *alias*. The *documentURI* refers to a URI corresponding to the NCL document containing the base to be imported. The *alias* attribute specifies a name to be used as prefix when referring an element of this imported base. The name must be unique (type=ID) and its scope is constrained to the document that has defined the alias attribute. The reference would have the format: *alias#element_id*, or *documentURI_value#element_id*. The import operation has the transitive property, that is, if *baseA* imports *baseB* that imports *baseC*, then *baseA* imports *baseC*. However, the *alias* defined for *baseC* inside *baseB* must be desconsidered by *baseA*.

When a language profile uses the Import module, the following specifications are allowed:

- The *descriptorBase* element may have a child *importBase* element referring to a URI corresponding to another NCL document containing the descriptor base (in fact its child elements) to be imported and nested. When a descriptor base is imported, the region base, the rule base and the cost function base, when present in the imported document, are also automatically imported to the corresponding region, rule and cost function bases of the importing document.

- The *connectorBase* element may have a child *importBase* element referring to a URI corresponding to another connector base (in fact its child elements) to be imported and nested.

- The *ruleBase* element may have a child *importBase* element referring to a URI corresponding to another NCL document containing the rule base (in fact its child elements) to be imported and nested.

- The *costFunctionBase* element may have a child *importBase* element referring to a URI corresponding to another NCL document containing the cost function base (in fact its child elements) to be imported and nested.

- The *templateBase* element may have a child *importBase* element referring to a URI corresponding to another NCL document containing the template base (in fact its child elements) to be imported and nested.

- The *regionBase* element may have a child *importBase* element referring to a URI corresponding to another NCL document containing the region base (in fact its child elements) to be imported and nested. Although NCL defines its layout model, nothing prevents an NCL document from using other layout models, since they define regions where objects can be presented, as for example SMIL 2.0 layout models.

The *importedDocumentBase* element specifies a set of imported NCL documents, and should be defined as a child element of the *head* element. An NCL document can be imported through the *importNCL* element. All bases defined inside an NCL document, as well as the document body element are imported all at once through the *importNCL* element. The bases will be treated as if each one is imported by an *importBase* element. The imported body element will be treated as a context node. The imported body as well as any of its contained nodes or links can be reused inside the body element of the importing NCL document. It must be remarked that the imported elements (the body and its nested child elements) can only be reused: new relationships can be defined among new nodes created by reuse (see next paragraph), but no new relationships can be defined inside imported context nodes[5].

The *importNCL* element has two attributes: *documentURI* and *alias*. The *documentURI* refers to a URI corresponding to the document to be imported. The *alias* attribute specifies a name to be used when referring an element of this imported document. As in the *importBase* element, the name must be unique (type=ID) and its scope is contrained to the document that has defined the alias attribute. The reference would have the format: *alias#element_id*, or *documentURI_value#element_id*. It is important to note that the same alias should be used when referring to elements defined in the imported document bases (*regionBase*, *connectorBase*, *descriptorBase*, etc.). The importNCL operation has also the transitive property, that is, if *documentA* imports *documentB* that imports *documentC*, then *documentA* imports *documentC*. However, the *alias* defined for *documentC* inside *documentB* must be desconsidered by *documentA*.

The ReuseEntity module allows an NCL element to be reused. This module defines the *refer* attribute, which refers to an element URI that will be reused. That is, the referred element and the element that refers to it must be considered the same[6]. When a language profile uses this module, it may add the *refer* attribute to:

- A *media* or *switch* element. In this case, the referred element must be, respectively, a media or switch element, which will represent the same node previously defined in the document body itself or in an external imported body. This referred element must directly contain the definition of all its attributes.

- A *context* element. In this case, the referred element must be a context or a body element that will represent the same context, which is previously defined in the document body itself or in an external imported body. This referred element must directly contain the definition of all its attributes.

---

[5] In other words, when reused, a node cannot have its content modified, but can be related to other nodes.

[6] Note that, although the language still uses the id attribute as type=ID, the formatter must consider the reffered and the referring elements as if they had the same id, without any conflict problem, as guaranteed by NCM.

- A *link* element. In this case, the referred element must be a link element that will represent the same link, which is previously defined in the document body itself or in an external imported body. This referred element must directly contain the definition of all its attributes.

- A *descriptor* or *descriptorSwitch* element. In this case, the referred element must be, respectively, a descriptor or a descriptorSwitch, which is previously defined in the document descriptorBase or in an external descriptorBase imported by the document, including those inherited from external NCL document imports. This referred element must directly contain the definition of all its attributes.

- A *rule* or *compositeRule* element. In this case, the referred element must be, respectively, a rule or a compositeRule, which is previously defined in the document ruleBase or in an external ruleBase imported by the document, including those inherited from external NCL document imports or descriptor base imports. This referred element must directly contain the definition of all its attributes.

- A *region* element. In this case, the referred element must be a region, which is previously defined in the document regionBase or in an external regionBase imported by the document, including those inherited from external NCL document imports or descriptor base imports. This referred element must directly contain the definition of all its attributes.

- A *costFunction* element. In this case, the referred element must be a costFunction, which is previously defined in the document costFunctionBase or in an external costFunctionBase imported by the document, including those inherited from external NCL document imports or descriptor base imports. This referred element must directly contain the definition of all its attributes.

When an element declares a *refer* attribute, all of its other attributes and child elements must be ignored by the formatter.

2.1.12. Metainformation Functionality

The Metainformation functionality is identical to the SMIL 2.0 homonym functionality. It defines a single module named Metainformation, which allows the definition of metadata about a document and about its elements, following the RDF – Resource Description Framework recommendation [RDF99]. All metadata information should be included in the document *head*.

2.2. NCL 2.3 Modules

2.2.1. Module Elements

The following tables indicate elements of each NCL 2.3 module (the Metainformation and the Composite-Node Templates functionalities, as well as the CompositeConnector module, are not detailed in this series of tables). As attributes and contents of elements can be defined in the module itself or in the language profile that groups the modules, the tables show the attributes and contents come also from the NCL Main Profile, as discussed in Section 4, besides those defined in the modules. In the content specification, the following symbols were used: (?) optional (zero or one occurrence), (|) or, (*) zero or more occurrences, (+) one or more occurrences.

Table 1 – Extended Structure Module

Elements	Attributes	Content
ncl	*id*	*(head?, body?)*
head		*(metainformation*, regionBase?, descriptorBase?, costFunctionBase?, ruleBase?, connectorBase?, importedDocumentBase?)*
body	*id*	*(compositeNodeArea| port| attribute| descriptorBind|(media|context|switch|link)**

Table 2 - Extended Layout Module

Elements	Attributes	Content
regionBase	*id, type*	*(importBase| region)**
region	*id, title, left, right, top, bottom, height, width, background, zIndex, decorated, movable, resizable, refer*	*(region)**

Table 3 – Extended Media Module

Elements	Attributes	Content
media	*id, src, refer, type, descriptor*	*(area|attribute)**

Table 4 – Extended Context Module

Elements	Attributes	Content
context	*id, refer, descriptor*	*(compositeNodeArea| port| attribute|descriptorBind|media|context|link|switch)**

Table 5 Extended MediaContentAnchor Module

Elements	Attributes	Content
area	*id, coords, begin, end, dur, text, position, first, last, anchorLabel*	*empty*

Table 6 – Extended CompositeNodeInterface Module

Elements	Attributes	Content
compositeNodeArea	*id*	*areaComponent+*
areaComponent	*componentId*	*empty*
port	*id, component, interface*	*empty*

Table 7 – Extended AttributeAnchor Module

Elements	Attributes	Content
attribute	*id, name*	*empty*

Table 8 – Extended SwitchInterface Module

Elements	Attributes	Content
switchPort	*id*	*mapping+*
mapping	*component, interface*	*empty*

Table 9 – Extended Descriptor Module

Elements	Attributes	Content
descriptor	*id, player, explicitDur, freeze, costFunction, refer, region*	*(descriptorParam)**
descriptorParam	*name, value*	
descriptorBase	*id*	*(importBase| (descriptor|descriptorSwitch)**

Table 10 – Extended DescriptorBind Module

Elements	Attributes	Content
descriptorBind	*component, descriptor*	*empty*

Table 11 - Extended Linking Module

Elements	Attributes	Content
bind	*role, component, interface, descriptor*	*(bindParam)**
bindParam	*name, value*	*empty*
linkParam	*name, value*	*empty*
link	*id, xconnector, refer*	*(linkParam*, bind+)*

Table 12 – Extended XConnector Module

Elements	Attributes	Content
causalConnector	*id, name, description*	*(connectorParam*, conditionRole+, assessmentRole*, actionRole+, causalGlue)*
constraintConnector	*id, name, description*	*(connectorParam*, assessmentRole+, constraintGlue)*
actionRole	*id, eventType, min, max*	*(presentationAction | assignmentAction)*
presentationAction	*actionType*	*empty*
assignmentAction	*value*	*empty*
conditionRole	*id, eventType, min, max, key*	*(eventStateTransitionCondition | attributeCondition | compoundCondition)*
compoundCondition	*isNegated, operator*	*(eventStateTransitionCondition* | attributeCondition* | compoundCondition*)+*
eventStateTransitionCondition	*transition*	*empty*
attributeCondition	*comparator, attributeType, value*	*empty*
assessmentRole	*id, eventType, min, max, key*	*(eventStateTransitionAssessment | attributeAssessment)*
eventStateTransitionAssessment	*transitionName*	*empty*
attributeAssessment	*attributeType*	*empty*
causalGlue		*((simpleTriggerExpression |compoundTriggerExpression), (simpleActionExpresssion | compoundActionExpresssion))*

constraintGlue		*(assessmentStatement | assessmentValueStatement | assessmentEventStatement | compoundStatement)*
simpleTriggerExpression	*minDelay, maxDelay, coditionRole, qualifier*	
compoundTriggerExpression	*isNegated, minDelay, maxDelay, logicalOperatorType*	*((simpleTriggerExpression | compoundTriggerExpression)+ , (assessmentStatement | assessmentValueStatement | assessmentEventStatement | compoundStatement)*)*
assessmentStatement	*comparator, firstAssessmentRole, firstAssessmentOffset firstRoleQualifier, secondAssessmentRole secondAssessmentOffset, secondRoleQualifier,*	
assessmentValueStatement	*comparator, assessmentRole, assesmentOffset, roleQualifier, value*	
compoundStatement	*operator, isNegated*	*(assessmentStatement | assessmentValueStatement | assessmentEventStatement | compoundStatement)+*
simpleActionExpresssion	*delay,actionRole, qualifier, repeat, repeatDelay*	
compoundActionExpresssion	*delay,operator*	*(simpleActionExpresssion | compoundActionExpresssion)+*
connectorParam	*name, type*	*Empty*
connectorBase	*id, name, description*	(*causalConnector**, *constraintConnector**)

Table 13 – Extended TestRule Module

Elements	Attributes	Content
ruleBase	*id*	*(importBase|rule|compositeRule)**
rule	*id, var, op, value, refer*	*empty*
compositeRule	*id, op, refer*	*(rule | compositeRule)+*

Table 14 – Extended ContentControl Module

Elements	Attributes	Content
switch	*id, refer*	*(switchPort| port| descriptorBind|bindRule|media|context|switch)**
bindRule	*component, rule*	*empty*

Table 15 – Extended DescriptorControl Module

Elements	Attributes	Content
descriptorSwitch	*id, refer*	*(descriptor, bindRule)**
bindRule	*component, rule*	*empty*

Table 16 –Timing Module

Elements	Attributes	Content
costFunctionBase	*id*	*(importBase| costFunction)**
costFunction	*id, type, refer, mathMLRef*	*(costFunctionParam)**
costFunctionParam	*name, value*	*empty*

Table 17 – Import Module

Elements	Attributes	Content
importBase	*alias, documentUri*	*empty*
importedDocumentBase	*id*	*(importNCL)+*
importNCL	*alias, documentUri*	*empty*

Table 18 – ReuseEntity Module

Elements	Attributes	Content
empty		

2.2.2. NCL 2.3 Modules in XML Schema

This section presents all NCL 2.3 module schemas, with the exception of Metainformation module; the XConnector module, which is presented in Section 6; and the CompositeConnector module and XTemplate functionalities, which are presented in [SoRC05].

Structure Module: NCL23Structure.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23Structure.xsd
Author: TeleMidia Laboratory
Revision: 29/06/2005

Schema for the Structure module namespace,
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:structure="http://www.telemidia.puc-rio.br/specs/xml/NCL23/Structure"
  targetNamespace="http://www.telemidia.puc-rio.br/specs/xml/NCL23/Structure"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <!-- ================================================================= -->
  <!-- define the top down structure of an NCL language document.          --
>
  <!-- ================================================================= -->

  <complexType name="nclPrototype">
    <sequence>
      <element ref="structure:head" minOccurs="0" maxOccurs="1"/>
      <element ref="structure:body" minOccurs="0" maxOccurs="1"/>
    </sequence>
    <attribute name="id" type="ID" />
  </complexType>

  <complexType name="headPrototype">
  </complexType>

  <complexType name="bodyPrototype">
    <attribute name="id" type="ID" use="optional"/>
  </complexType>

  <!-- declare global elements in this module -->
  <element name="ncl" type="structure:nclPrototype"/>
  <element name="head" type="structure:headPrototype"/>
  <element name="body" type="structure:bodyPrototype"/>
</schema>
```

Layout Module: NCL23Layout.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.telemidia.puc-
rio.br/specs/xml/ncl/modules/NCL23Layout.xsd
Author: TeleMidia Laboratory
Revision: 29/06/2005

Schema for the NCL Layout module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:layout="http://www.telemidia.puc-rio.br/specs/xml/NCL23/Layout"
  targetNamespace="http://www.telemidia.puc-rio.br/specs/xml/NCL23/Layout"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <!-- define the layout element prototype -->
  <complexType name="regionBasePrototype">
    <choice minOccurs="0" maxOccurs="unbounded">
      <element ref="layout:region" />
    </choice>
    <attribute name="id" type="ID" use="optional" />
    <attribute name="type" type="string" use="optional" />
  </complexType>

  <!-- define the device Component element type -->
  <complexType name="regionPrototype">
    <sequence minOccurs="0" maxOccurs="unbounded">
      <element ref="layout:region" />
    </sequence>
    <attribute name="id" type="ID" use="optional" />
    <attribute name="title" type="string" use="optional" />
    <attribute name="background" type="string" use="optional" />
    <attribute name="height" type="string" use="optional" />
    <attribute name="left" type="string" use="optional"  />
    <attribute name="right" type="string" use="optional"  />
    <attribute name="top" type="string" use="optional"  />
    <attribute name="bottom" type="string" use="optional"  />
    <attribute name="width" type="string" use="optional"  />
    <attribute name="zIndex" type="integer" use="optional" />
    <attribute name="decorated" type="boolean" use="optional"  />
    <attribute name="movable" type="boolean" use="optional" />
    <attribute name="resizable" type="boolean" use="optional" />
  </complexType>

  <!-- define the global region attribute -->

  <!-- define the region attributeGroup -->
  <attributeGroup name="regionAttrs">
    <attribute name="region" type="string" use="optional"/>
  </attributeGroup>

  <!-- define the global layout elements -->
  <element name="regionBase" type="layout:regionBasePrototype"/>
  <element name="region" type="layout:regionPrototype"/>

</schema>
```

Media Module: NCL23Media.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23Media.xsd
Author: TeleMidia Laboratory
Revision: 29/06/2005

Schema for the NCL Media module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:media="http://www.telemidia.puc-rio.br/specs/xml/NCL23/Media"
  targetNamespace="http://www.telemidia.puc-rio.br/specs/xml/NCL23/Media"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <!--
      define the media object element prototype
      animation, audio, img, text, textstream, video
      are all identical and use this prototype
  -->
  <complexType name="mediaPrototype">
    <attribute name="id" type="ID" use="optional" />
    <attribute name="type" type="string" use="optional"/>
    <attribute name="src" type="anyURI" use="optional"/>
  </complexType>

  <!-- define the global media elements -->
  <element name="media" type="media:mediaPrototype"/>
</schema>
```

Context Module: NCL23Context.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23Context.xsd
Author: TeleMidia Laboratory
Revision: 29/06/2005

Schema for the NCL Context module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:context="http://www.telemidia.puc-rio.br/specs/xml/NCL23/Context"
  targetNamespace="http://www.telemidia.puc-rio.br/specs/xml/NCL23/Context"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <!-- define the compositeNode element prototype -->
  <complexType name="contextPrototype">
    <attribute name="id" type="ID" use="optional"/>
  </complexType>

  <!-- declare global elements in this module -->
  <element name="context" type="context:contextPrototype"/>
</schema>
```

MediaContentAnchor Module: NCL23MediaContentAnchor.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23MediaContentAnchor.xsd
Author: TeleMidia Laboratory
Revision: 29/06/2005

Schema for the NCL Media Content Anchor module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:mediaAnchor="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/MediaContentAnchor"
  targetNamespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/MediaContentAnchor"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <!-- define the temporalAnchorAttrs attribute group -->
  <attributeGroup name="temporalAnchorAttrs">
    <attribute name="begin" type="string"/>
    <attribute name="end" type="string"/>
  </attributeGroup>

  <!-- define the textAnchorAttrs attribute group -->
  <attributeGroup name="textAnchorAttrs">
    <attribute name="text" type="string"/>
    <attribute name="position" type="unsignedLong"/>
  </attributeGroup>

  <!-- define the sampleAnchorAttrs attribute group -->
  <attributeGroup name="sampleAnchorAttrs">
    <attribute name="first" type="unsignedLong"/>
    <attribute name="last" type="unsignedLong"/>
  </attributeGroup>

  <!-- define the coordsAnchorAttrs attribute group -->
  <attributeGroup name="coordsAnchorAttrs">
    <attribute name="coords" type="string"/>
  </attributeGroup>

  <!-- define the labelAnchorAttrs attribute group -->
  <attributeGroup name="labelAnchorAttrs">
    <attribute name="anchorLabel" type="string"/>
  </attributeGroup>

  <complexType name="componentAnchorPrototype">
    <attribute name="id" type="ID" use="required"/>
    <attributeGroup ref="mediaAnchor:coordsAnchorAttrs" />
    <attributeGroup ref="mediaAnchor:temporalAnchorAttrs" />
    <attributeGroup ref="mediaAnchor:textAnchorAttrs" />
    <attributeGroup ref="mediaAnchor:sampleAnchorAttrs" />
    <attributeGroup ref="mediaAnchor:labelAnchorAttrs" />
  </complexType>

  <!-- declare global elements in this module -->
  <element name="area" type="mediaAnchor:componentAnchorPrototype"/>
</schema>
```

CompositeNodeInterface Module: NC23CompositeNodeInterface.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23CompositeNodeInterface.xsd
Author: TeleMidia Laboratory
Revision: 29/06/2005

Schema for the NCL Composite Node Interface module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:compositeInterface="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/CompositeNodeInterface"
  targetNamespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/CompositeNodeInterface"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <complexType name="compositeNodeAnchorPrototype">
    <sequence>
      <element name="areaComponent" minOccurs="1" maxOccurs="unbounded">
        <complexType>
          <attribute name="componentId" type="string" use="required" />
        </complexType>
      </element>
    </sequence>
    <attribute name="id" type="ID" use="required" />
  </complexType>

  <complexType name="compositeNodePortPrototype">
    <attribute name="id" type="ID" use="required" />
    <attribute name="component" type="string" use="required"/>
    <attribute name="interface" type="IDREF" use="optional" />
  </complexType>

  <!-- declare global elements in this module -->
  <element name="compositeNodeArea"
type="compositeInterface:compositeNodeAnchorPrototype" />
  <element name="port" type="compositeInterface:compositeNodePortPrototype" />
</schema>
```

AttributeAnchor Module: NCL23AttributeAnchor.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23AttributeAnchor.xsd
Author: TeleMidia Laboratory
Revision: 29/06/2005

Schema for the NCL Attribute Anchor module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:attributeAnchor="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/AttributeAnchor"
```

```
  targetNamespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/AttributeAnchor"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <complexType name="attributeAnchorPrototype">
    <attribute name="id" type="ID" use="required" />
    <attribute name="name" type="string" use="required" />
  </complexType>

  <!-- declare global elements in this module -->
  <element name="attribute" type="attributeAnchor:attributeAnchorPrototype"/>
</schema>
```

SwitchInterface Module: NCL23SwitchInterface.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23SwitchInterface.xsd
Author: TeleMidia Laboratory
Revision: 29/06/2005

Schema for the NCL Switch Interface module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:switchInterface="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/SwitchInterface"
  targetNamespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/SwitchInterface"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <complexType name="mappingPrototype">
    <attribute name="component" type="string" use="required"/>
    <attribute name="interface" type="IDREF" use="optional"/>
  </complexType>

  <complexType name="switchPortPrototype">
    <sequence>
      <element ref="switchInterface:mapping" minOccurs="1"
maxOccurs="unbounded"/>
    </sequence>
    <attribute name="id" type="ID" use="required"/>
  </complexType>

  <!-- declare global elements in this module -->
  <element name="mapping" type="switchInterface:mappingPrototype"/>
  <element name="switchPort" type="switchInterface:switchPortPrototype" />
</schema>
```

Descriptor Module: NCL23Descriptor.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23Descriptor.xsd
Author: TeleMidia Laboratory
Revision: 29/06/2005

Schema for the NCL Descriptor module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:descriptor="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/Descriptor"
  targetNamespace="http://www.telemidia.puc-rio.br/specs/xml/NCL23/Descriptor"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <complexType name="descriptorParamPrototype">
    <attribute name="name" type="string" use="required" />
    <attribute name="value" type="string" use="required"/>
  </complexType>

  <complexType name="descriptorPrototype">
    <sequence minOccurs="0" maxOccurs="unbounded">
      <element ref="descriptor:descriptorParam"/>
    </sequence>
    <attribute name="id" type="ID" use="optional" />
    <attribute name="player" type="string" use="optional"/>
  </complexType>

<!--
Formatters should support the following descriptorParam names.
* For audio players: soundLevel; balanceLevel; trebleLevel; bassLevel;
* For text players: style, which refers to a style sheet with information for
text presentation;
* For continuous media players: enableTimeBar, used for enabling a time
progress bar;
* For visual media players: scroll, which allows the specification of how an
author would like to configure the scroll in a region; fit, indicating how an
object will be presented (hidden, fill, meet, meetBest, slice).
* For players in general: reusePlayer, which determines if a new player must
be instatiated or if a player already instatiated must be used; and
playerLife, whic specifies what will happen to the player instance at the end
of the presentation.

-->

  <complexType name="descriptorBasePrototype">
    <choice minOccurs="0" maxOccurs="unbounded">
      <element ref="descriptor:descriptor" />
    </choice>
    <attribute name="id" type="ID" use="optional" />
  </complexType>

  <!-- declare global elements in this module -->
  <element name="descriptorParam" type="descriptor:descriptorParamPrototype"/>
  <element name="descriptor" type="descriptor:descriptorPrototype"/>
  <element name="descriptorBase" type="descriptor:descriptorBasePrototype"/>

  <!-- define the descriptor global attribute -->
```

```
  <attributeGroup name="descriptorAttrs">
    <attribute name="descriptor" type="string" use="optional"/>
  </attributeGroup>

</schema>
```

DescriptorBind Module: NCL23DescriptorBind.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23DescriptorBind.xsd
Author: TeleMidia Laboratory
Revision: 29/06/2005

Schema for the NCL DescriptorBind module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:descriptorBind="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/DescriptorBind"
  targetNamespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/DescriptorBind"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <complexType name="descriptorBindPrototype">
    <attribute name="component" type="string" use="required"/>
    <attribute name="descriptor" type="string" use="required"/>
  </complexType>

  <!-- declare global elements in this module -->
  <element name="descriptorBind"
type="descriptorBind:descriptorBindPrototype"/>

</schema>
```

Linking Module: NCL23Linking.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23Linking.xsd
Author: TeleMidia Laboratory
Revision: 29/06/2005

Schema for the NCL Linking module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:linking="http://www.telemidia.puc-rio.br/specs/xml/NCL23/Linking"
  targetNamespace="http://www.telemidia.puc-rio.br/specs/xml/NCL23/Linking"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <complexType name="paramPrototype">
    <attribute name="name" type="string" use="required"/>
    <attribute name="value" type="anySimpleType" use="required"/>
```

```
  </complexType>

  <complexType name="bindPrototype">
    <sequence minOccurs="0" maxOccurs="unbounded">
      <element ref="linking:bindParam"/>
    </sequence>
    <attribute name="role" type="string" use="required"/>
    <attribute name="component" type="string" use="required"/>
    <attribute name="interface" type="IDREF" use="optional"/>
  </complexType>

  <complexType name="linkPrototype">
    <sequence>
      <element ref="linking:linkParam" minOccurs="0" maxOccurs="unbounded"/>
      <element ref="linking:bind" minOccurs="2" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="id" type="ID" use="optional"/>
    <attribute name="xconnector" type="string" use="required"/>
  </complexType>

  <!-- declare global elements in this module -->
  <element name="linkParam" type="linking:paramPrototype"/>
  <element name="bindParam" type="linking:paramPrototype"/>
  <element name="bind" type="linking:bindPrototype" />
  <element name="link" type="linking:linkPrototype" />

</schema>
```

TestRule Module: NCL23TestRule.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23TestRules.xsd
Author: TeleMidia Laboratory
Revision: 29/06/2005

Schema for the NCL TestRule module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:testRule="http://www.telemidia.puc-rio.br/specs/xml/NCL23/TestRule"
  targetNamespace="http://www.telemidia.puc-rio.br/specs/xml/NCL23/TestRule"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <complexType name="rulePrototype">
    <attribute name="id" type="ID" use="optional" />
    <attribute name="var" type="string" use="optional"/>
    <attribute name="value" type="string" use="optional"/>
    <attribute name="op" use="optional">
      <simpleType>
        <restriction base="string">
          <enumeration value="eq"/>
          <enumeration value="ne"/>
          <enumeration value="gt"/>
          <enumeration value="ge"/>
          <enumeration value="lt"/>
          <enumeration value="le"/>
        </restriction>
      </simpleType>
    </attribute>
  </complexType>
```

```
  <complexType name="compositeRulePrototype">
    <choice minOccurs="2" maxOccurs="unbounded">
      <element ref="testRule:rule"/>
      <element ref="testRule:compositeRule"/>
    </choice>
    <attribute name="id" type="ID" use="optional" />
    <attribute name="op" use="optional">
      <simpleType>
        <restriction base="string">
          <enumeration value="and"/>
          <enumeration value="or"/>
        </restriction>
      </simpleType>
    </attribute>
  </complexType>

  <complexType name="ruleBasePrototype">
    <choice minOccurs="0" maxOccurs="unbounded">
      <element ref="testRule:rule"/>
      <element ref="testRule:compositeRule"/>
    </choice>
    <attribute name="id" type="ID" use="optional" />
  </complexType>

  <!-- declare global elements in this module -->
  <element name="rule" type="testRule:rulePrototype"/>
  <element name="compositeRule" type="testRule:compositeRulePrototype"/>
  <element name="ruleBase" type="testRule:ruleBasePrototype"/>

</schema>
```

ContentControl Module: NCL23ContentControl.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23ContentControl.xsd
Author: TeleMidia Laboratory
Revision: 29/06/2005

Schema for the NCL ContentControl module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:contentControl="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/ContentControl"
  targetNamespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/ContentControl"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <!-- define the switch element prototype -->

  <complexType name="switchPrototype">
    <attribute name="id" type="ID" use="required" />
  </complexType>

  <!-- declare global elements in this module -->
  <element name="switch" type="contentControl:switchPrototype"/>

</schema>
```

DescriptorControl Module: NCL23DescriptorControl.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23DescriptorControl.xsd
Author: TeleMidia Laboratory
Revision: 29/06/2005

Schema for the NCL DescriptorControl module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:descriptorControl="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/DescriptorControl"
  targetNamespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/DescriptorControl"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <!-- define the descriptor switch element prototype -->
  <complexType name="descriptorSwitchPrototype">
    <attribute name="id" type="ID" use="optional"/>
  </complexType>

  <!-- declare global elements in this module -->
  <element name="descriptorSwitch"
type="descriptorControl:descriptorSwitchPrototype"/>

</schema>
```

Timing Module: NCL23Timing.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23Timing.xsd
Author: TeleMidia Laboratory
Revision: 29/06/2005

Schema for the NCL Timing module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:timing="http://www.telemidia.puc-rio.br/specs/xml/NCL23/Timing"
  targetNamespace="http://www.telemidia.puc-rio.br/specs/xml/NCL23/Timing"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <!-- declare global attributes in this module -->

  <!-- define the costFunctionAttrs attribute group -->
  <attributeGroup name="costFunctionAttrs">
    <attribute name="costFunction" type="string" use="optional"/>
  </attributeGroup>

  <!-- define the explicitDur attribute group -->
  <attributeGroup name="explicitDurAttrs">
    <attribute name="explicitDur" type="string" use="optional"/>
```

```
  </attributeGroup>

  <!-- define the freeze attribute group -->
  <attributeGroup name="freezeAttrs">
    <attribute name="freeze" type="boolean" use="optional"/>
  </attributeGroup>

  <complexType name="costFunctionPrototype">
    <sequence>
      <element name="costFunctionParam" minOccurs="0" maxOccurs="unbounded">
        <complexType>
          <attribute name="name" type="string" use="required"/>
          <attribute name="value" type="anySimpleType" use="required"/>
        </complexType>
      </element>
    </sequence>
    <attribute name="id" type="ID" use="optional" />
    <attribute name="type" type="string" use="optional"/>
  </complexType>

  <complexType name="costFunctionBasePrototype">
    <choice minOccurs="0" maxOccurs="unbounded">
      <element ref="timing:costFunction"/>
    </choice>
  </complexType>

  <!-- declare global elements in this module -->
  <element name="costFunction" type="timing:costFunctionPrototype"/>
  <element name="costFunctionBase" type="timing:costFunctionBasePrototype"/>

</schema>
```

Import Module: NCL23Import.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23Import.xsd
Author: TeleMidia Laboratory
Revision: 29/06/2005

Schema for the NCL Import module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:import="http://www.telemidia.puc-rio.br/specs/xml/NCL23/Import"
  targetNamespace="http://www.telemidia.puc-rio.br/specs/xml/NCL23/Import"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <complexType name="importBasePrototype">
      <attribute name="alias" type="ID" use="optional"/>
      <attribute name="documentURI" type="anyURI" use="required"/>
  </complexType>

  <complexType name="importNCLPrototype">
      <attribute name="alias" type="ID" use="optional"/>
      <attribute name="documentURI" type="anyURI" use="required"/>
  </complexType>

  <complexType name="importedDocumentBasePrototype">
    <sequence minOccurs="1" maxOccurs="unbounded">
      <element ref="import:importNCL" />
```

```
    </sequence>
    <attribute name="id" type="ID" use="optional" />
  </complexType>

  <!-- declare global elements in this module -->
  <element name="importBase" type="import:importBasePrototype"/>
  <element name="importNCL" type="import:importNCLPrototype"/>
  <element name="importedDocumentBase"
type="import:importedDocumentBasePrototype"/>

</schema>
```

ReuseEntity Module: NCL23ReuseEntity.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23ReuseEntity.xsd
Author: TeleMidia Laboratory
Revision: 29/06/2005

Schema for the NCL ReuseEntity module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:reuseEntity="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/ReuseEntity"
  targetNamespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/ReuseEntity"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <!-- define the reuseEntity global attribute -->
  <attributeGroup name="reuseEntityAttrs">
    <attribute name="refer" type="string" use="optional"/>
  </attributeGroup>

</schema>
```

3. NCL 2.3 Language Profiles

NCL 2.3 modules can be combined to define different NCL Language Profiles. This section presents some of the main profiles already defined.

3.1. The NCL 2.3 Language Profile

The *NCL 2.3 Full Profile*, also called *NCL 2.3 Language profile*, is the "complete profile" of NCL 2.3 language. It includes all NCL modules and provides all facilities for declarative authoring of NCL documents. The XML Schema for this profile is presented and discussed in [SoRC05].

3.2. NCL 2.3 Main Profile

NCL 2.3 Main Profile includes the Structure, Layout, Media, Context, MediaContentAnchor, CompositeNodeInterface, AttributeAnchor, SwitchInterface, Descriptor, DescriptorBind, XConnector, Linking, TestRule, ContentControl, DescriptorControl, Timing, Import and ReuseEntity modules. Section 4 presents the definition of this profile using XML Schema. Section 5 presents an example of a document in compliance with this profile.

3.3. NCL 2.3 XConnector Profile

XConnector Profile allows the creation of simple hypermedia connectors. This profile includes only the XConnector and Structure modules. Section 6 presents the definition of this profile using XML Schema. Appendix A shows a connector base for Allen's relations and Appendix B another extensive connector base, both using the XConnector profile.

3.4. Other NCL 2.3 Language Profiles

Besides the NCL 2.3 Language Profile, other profiles may be built, such as:

- *Basic Media Profile* – allows the creation of hypermedia documents without composite nodes and composite connectors. This profile includes the Structure, Layout, Media, MediaContentAnchor, AttributeAnchor, Descriptor, XConnector, and Linking modules.
- *Simple NCL Profile* – This profile includes the Structure, Layout, Media, Context, MediaContentAnchor, CompositeNodeInterface, Descriptor, XConnector, Linking and ReuseEntity modules.
- *Basic NCM Profile* – This profile includes the Structure, Layout, Media, Context, MediaContentAnchor, CompositeNodeInterface, AttributeAnchor, SwitchInterface, Descriptor, DescriptorBind, XConnector, Linking, TestRule, ContentControl, DescriptorControl, Timing, and ReuseEntity modules.
- *High NCL Profile* –This profile includes the Structure, Layout, Media, Context, MediaContentAnchor, CompositeNodeInterface, AttributeAnchor, SwitchInterface, Descriptor, DescriptorBind, XConnector, Linking, TestRule, ContentControl, DescriptorControl, Timing, XTemplate, XTemplateUse, Import and ReuseEntity modules.

- *XTemplate Profile* – allows the creation of hypermedia composite-node templates. This profile includes Structure, and XTemplate modules. The XML Schema for this profile is presented and discussed in [SoRC05]. Other profiles using XTemplate are also discussed in [SoRC05].

4. The NCL 2.3 Main Profile XML Schema

This section presents the definition of the NCL 2.3 Main Profile using XML Schema.

NCL23.xsd

```
<!--
XML Schema for the NCL Language

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights
Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.telemidia.puc-
rio.br/specs/xml/NCL23/profiles/NCL23.xsd
Author: TeleMidia Laboratory
Revision: 29/06/2005
-->

<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:attributeAnchor="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/AttributeAnchor"
  xmlns:compositeInterface="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/CompositeNodeInterface"
  xmlns:connector="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/XConnector"
  xmlns:contentControl="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/ContentControl"
  xmlns:context="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/Context"
  xmlns:descriptor="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/Descriptor"
  xmlns:descriptorBind="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/DescriptorBind"
  xmlns:descriptorControl="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/DescriptorControl"
  xmlns:import="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/Import"
  xmlns:layout="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/Layout"
  xmlns:linking="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/Linking"
  xmlns:media="http://www.telemidia.puc-rio.br/specs/xml/NCL23/Media"
  xmlns:mediaAnchor="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/MediaContentAnchor"
  xmlns:reuseEntity="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/ReuseEntity"
  xmlns:structure="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/Structure"
  xmlns:switchInterface="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/SwitchInterface"
  xmlns:testRule="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/TestRule"
  xmlns:timing="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/Timing"
  xmlns:profile="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/profiles"
```

```
  targetNamespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/profiles"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <!-- import the definitions in the modules namespaces -->
  <import namespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/AttributeAnchor"
    schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23AttributeAnchor.xsd"/>
  <import namespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/CompositeNodeInterface"
    schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23CompositeNodeInterface.xsd"/>
  <import namespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/XConnector"
    schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23XConnector.xsd"/>
  <import namespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/ContentControl"
    schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23ContentControl.xsd"/>
  <import namespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/Context"
    schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23Context.xsd"/>
  <import namespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/Descriptor"
    schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23Descriptor.xsd"/>
  <import namespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/DescriptorBind"
    schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23DescriptorBind.xsd"/>
  <import namespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/DescriptorControl"
    schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23DescriptorControl.xsd"/>
  <import namespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/Import"
    schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23Import.xsd"/>
  <import namespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/Layout"
    schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23Layout.xsd"/>
  <import namespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/Linking"
    schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23Linking.xsd"/>
  <import namespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/Media"
    schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23Media.xsd"/>
  <import namespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/MediaContentAnchor"
    schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23MediaContentAnchor.xsd"/>
  <import namespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/ReuseEntity"
    schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23ReuseEntity.xsd"/>
```

```
  <import namespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/Structure"
    schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23Structure.xsd"/>
  <import namespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/SwitchInterface"
    schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23SwitchInterface.xsd"/>
  <import namespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/TestRule"
    schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23TestRule.xsd"/>
  <import namespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/Timing"
    schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23Timing.xsd"/>

  <!-- ============================================================ -->
  <!-- Structure                                                    -->
  <!-- ============================================================ -->
  <!-- extends ncl element -->

  <element name="ncl" substitutionGroup="structure:ncl"/>

  <!-- extends head element -->

  <complexType name="headType">
    <complexContent>
      <extension base="structure:headPrototype">
        <all>
          <element ref="profile:importedDocumentBase" minOccurs="0"/>
          <element ref="profile:regionBase" minOccurs="0"/>
          <element ref="profile:descriptorBase" minOccurs="0"/>
          <element ref="profile:ruleBase" minOccurs="0"/>
          <element ref="profile:costFunctionBase" minOccurs="0"/>
          <element ref="profile:connectorBase" minOccurs="0"/>
        </all>
      </extension>
    </complexContent>
  </complexType>

  <element name="head" type="profile:headType"
substitutionGroup="structure:head"/>

  <!-- extends body element -->

  <complexType name="bodyType">
    <complexContent>
      <extension base="structure:bodyPrototype">
        <choice minOccurs="0" maxOccurs="unbounded">
          <group ref="profile:contextInterfaceElementGroup"/>
          <element ref="profile:descriptorBind"/>
          <element ref="profile:media"/>
          <element ref="profile:context"/>
          <element ref="profile:switch"/>
          <element ref="profile:link"/>
        </choice>
      </extension>
    </complexContent>
  </complexType>
```

```
  <element name="body" type="profile:bodyType"
substitutionGroup="structure:body"/>

  <!-- ============================================================ -->
  <!-- Layout                                                       -->
  <!-- ============================================================ -->
  <!-- extends regionBase element -->

  <complexType name="regionBaseType">
    <complexContent>
      <extension base="layout:regionBasePrototype">
        <choice minOccurs="0" maxOccurs="unbounded">
          <element ref="profile:importBase"/>
        </choice>
      </extension>
    </complexContent>
  </complexType>

  <complexType name="regionType">
    <complexContent>
      <extension base="layout:regionPrototype">
        <attributeGroup ref="reuseEntity:reuseEntityAttrs"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="regionBase" type="profile:regionBaseType"
substitutionGroup="layout:regionBase"/>
  <element name="region" type="profile:regionType"
substitutionGroup="layout:region"/>

  <!-- ============================================================ -->
  <!-- Media                                                        -->
  <!-- ============================================================ -->
  <!-- extends Media elements -->

  <!-- media interface element groups -->
  <group name="mediaInterfaceElementGroup">
    <choice>
      <element ref="profile:area"/>
      <element ref="profile:attribute"/>
    </choice>
  </group>

  <complexType name="mediaType">
    <complexContent>
      <extension base="media:mediaPrototype">
        <choice minOccurs="0" maxOccurs="unbounded">
          <group ref="profile:mediaInterfaceElementGroup"/>
        </choice>
        <attributeGroup ref="descriptor:descriptorAttrs"/>
        <attributeGroup ref="reuseEntity:reuseEntityAttrs"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="media" type="profile:mediaType"
substitutionGroup="media:media"/>

  <!-- ============================================================ -->
  <!-- Context                                                      -->
```

```
  <!-- ========================================================== -->
  <!-- extends context element -->

  <!-- composite node interface element groups -->
  <group name="contextInterfaceElementGroup">
    <choice>
      <element ref="profile:compositeNodeArea"/>
      <element ref="profile:port"/>
      <element ref="profile:attribute"/>
    </choice>
  </group>

  <complexType name="contextType">
    <complexContent>
      <extension base="context:contextPrototype">
        <choice minOccurs="0" maxOccurs="unbounded">
          <group ref="profile:contextInterfaceElementGroup"/>
          <element ref="profile:descriptorBind"/>
          <element ref="profile:media"/>
          <element ref="profile:context"/>
          <element ref="profile:link"/>
          <element ref="profile:switch"/>
        </choice>
        <attributeGroup ref="descriptor:descriptorAttrs"/>
        <attributeGroup ref="reuseEntity:reuseEntityAttrs"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="context" type="profile:contextType"
substitutionGroup="context:context"/>

  <!-- ========================================================== -->
  <!-- MediaContentAnchor                                         -->
  <!-- ========================================================== -->
  <!-- extends area element -->

  <complexType name="componentAnchorType">
    <complexContent>
      <extension base="mediaAnchor:componentAnchorPrototype">
      </extension>
    </complexContent>
  </complexType>

  <element name="area" type="profile:componentAnchorType"
substitutionGroup="mediaAnchor:area"/>

  <!-- ========================================================== -->
  <!-- CompositeNodeInterface                                     -->
  <!-- ========================================================== -->
  <!-- extends compositeNodeArea element -->

  <complexType name="compositeNodeAnchorType">
    <complexContent>
      <extension
base="compositeInterface:compositeNodeAnchorPrototype">
      </extension>
    </complexContent>
  </complexType>
```

```
  <element name="compositeNodeArea"
type="profile:compositeNodeAnchorType"
substitutionGroup="compositeInterface:compositeNodeArea"/>

  <!-- extends port element -->

  <complexType name="compositeNodePortType">
    <complexContent>
      <extension base="compositeInterface:compositeNodePortPrototype">
      </extension>
    </complexContent>
  </complexType>

  <element name="port" type="profile:compositeNodePortType"
substitutionGroup="compositeInterface:port"/>

  <!-- ========================================================== -->
  <!-- AtributeAnchor                                             -->
  <!-- ========================================================== -->
  <!-- extends attribute element -->

  <complexType name="attributeAnchorType">
    <complexContent>
      <extension base="attributeAnchor:attributeAnchorPrototype">
      </extension>
    </complexContent>
  </complexType>

  <element name="attribute" type="profile:attributeAnchorType"
substitutionGroup="attributeAnchor:attribute"/>

  <!-- ========================================================== -->
  <!-- SwitchInterface                                            -->
  <!-- ========================================================== -->
  <!-- extends switchPort element -->

  <complexType name="switchPortType">
    <complexContent>
      <extension base="switchInterface:switchPortPrototype">
      </extension>
    </complexContent>
  </complexType>

  <element name="mapping"
substitutionGroup="switchInterface:mapping"/>
  <element name="switchPort" type="profile:switchPortType"
substitutionGroup="switchInterface:switchPort"/>

  <!-- ========================================================== -->
  <!-- Descriptor                                                 -->
  <!-- ========================================================== -->

  <!-- substitutes descriptorParam element -->

  <element name="descriptorParam"
substitutionGroup="descriptor:descriptorParam"/>


  <!-- extends descriptor element -->

  <complexType name="descriptorType">
```

```
    <complexContent>
      <extension base="descriptor:descriptorPrototype">
        <attributeGroup ref="layout:regionAttrs"/>
        <attributeGroup ref="timing:costFunctionAttrs"/>
        <attributeGroup ref="timing:explicitDurAttrs"/>
        <attributeGroup ref="timing:freezeAttrs"/>
        <attributeGroup ref="reuseEntity:reuseEntityAttrs"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="descriptor" type="profile:descriptorType"
substitutionGroup="descriptor:descriptor"/>

  <!-- extends descriptorBase element -->
  <complexType name="descriptorBaseType">
    <complexContent>
      <extension base="descriptor:descriptorBasePrototype">
        <choice>
          <element ref="profile:importBase" minOccurs="0"
maxOccurs="unbounded"/>
          <element ref="profile:descriptorSwitch" minOccurs="0"
maxOccurs="unbounded"/>
        </choice>
      </extension>
    </complexContent>
  </complexType>

  <element name="descriptorBase" type="profile:descriptorBaseType"
substitutionGroup="descriptor:descriptorBase"/>

  <!-- ============================================================ -->
  <!-- CompositeNodedDescriptorBind                                 -->
  <!-- ============================================================ -->
  <!-- extends descriptorBind element -->

  <element name="descriptorBind"
substitutionGroup="descriptorBind:descriptorBind"/>

  <!-- ============================================================ -->
  <!-- Linking                                                      -->
  <!-- ============================================================ -->

  <!-- substitutes linkParam and bindParam elements -->
  <element name="linkParam" substitutionGroup="linking:linkParam"/>
  <element name="bindParam" substitutionGroup="linking:bindParam"/>


  <!-- extends bind element and link element, as a consequence-->

  <complexType name="bindType">
    <complexContent>
      <extension base="linking:bindPrototype">
        <attributeGroup ref="descriptor:descriptorAttrs"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="bind" type="profile:bindType"
substitutionGroup="linking:bind"/>
```

```
  <!-- extends link element -->
  <complexType name="linkType">
    <complexContent>
      <extension base="linking:linkPrototype">
        <attributeGroup ref="reuseEntity:reuseEntityAttrs"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="link" type="profile:linkType"
substitutionGroup="linking:link"/>

  <!-- ========================================================== -->
  <!-- XConnector                                                 -->
  <!-- ========================================================== -->
  <!-- extends causalConnector element -->
  <complexType name="causalConnectorType">
    <complexContent>
      <extension base="connector:causalConnectorPrototype">
        <attributeGroup ref="reuseEntity:reuseEntityAttrs"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="causalConnector" type="profile:causalConnectorType"
substitutionGroup="connector:causalConnector"/>

  <!-- extends constraintConnector element -->
  <complexType name="constraintConnectorType">
    <complexContent>
      <extension base="connector:constraintConnectorPrototype">
        <attributeGroup ref="reuseEntity:reuseEntityAttrs"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="constraintConnector"
type="profile:constraintConnectorType"
substitutionGroup="connector:constraintConnector"/>

  <!-- extends connectorBase element -->
  <complexType name="connectorBaseType">
    <complexContent>
      <extension base="connector:connectorBasePrototype">
        <choice minOccurs="0" maxOccurs="unbounded">
          <element ref="profile:importBase"/>
        </choice>
      </extension>
    </complexContent>
  </complexType>

  <element name="connectorBase" type="profile:connectorBaseType"
substitutionGroup="connector:connectorBase"/>

  <element name="connectorParam"
substitutionGroup="connector:connectorParam"/>

  <element name="conditionRole"
substitutionGroup="connector:conditionRole"/>
  <element name="assessmentRole"
substitutionGroup="connector:assessmentRole"/>
```

```
  <element name="actionRole"
substitutionGroup="connector:actionRole"/>

  <element name="compoundCondition"
substitutionGroup="connector:compoundCondition"/>

  <element name="eventStateTransitionCondition"
substitutionGroup="connector:eventStateTransitionCondition"/>
  <element name="attributeCondition"
substitutionGroup="connector:attributeCondition"/>
  <element name="eventStateTransitionAssessment"
substitutionGroup="connector:eventStateTransitionAssessment"/>
  <element name="attributeAssessment"
substitutionGroup="connector:attributeAssessment"/>

  <element name="presentationAction"
substitutionGroup="connector:presentationAction"/>
  <element name="assignmentAction"
substitutionGroup="connector:assignmentAction"/>

  <element name="causalGlue"
substitutionGroup="connector:causalGlue"/>
  <element name="constraintGlue"
substitutionGroup="connector:constraintGlue"/>

  <element name="simpleTriggerExpression"
substitutionGroup="connector:simpleTriggerExpression"/>
  <element name="compoundTriggerExpression"
substitutionGroup="connector:compoundTriggerExpression"/>
  <element name="simpleActionExpression"
substitutionGroup="connector:simpleActionExpression"/>
  <element name="compoundActionExpression"
substitutionGroup="connector:compoundActionExpression"/>
  <element name="assessmentStatement"
substitutionGroup="connector:assessmentStatement"/>
  <element name="assessmentValueStatement"
substitutionGroup="connector:assessmentValueStatement"/>
  <element name="compoundStatement"
substitutionGroup="connector:compoundStatement"/>

  <!-- ============================================================ -->
  <!-- TestRule                                                     -->
  <!-- ============================================================ -->
  <!-- extends rule element -->
  <complexType name="ruleType">
    <complexContent>
      <extension base="testRule:ruleePrototype">
        <attributeGroup ref="reuseEntity:reuseEntityAttrs"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="rule" type="profile:ruleType"
substitutionGroup="testRule:rule"/>

  <!-- extends compositeRule element -->
  <complexType name="compositeRuleType">
    <complexContent>
      <extension base="testRule:compositeRulePrototype">
        <attributeGroup ref="reuseEntity:reuseEntityAttrs"/>
      </extension>
```

```
    </complexContent>
  </complexType>

  <element name="compositeRule" type="profile:compositeRuleType"
substitutionGroup="testRule:compositeRule"/>

  <!-- extends ruleBase element -->
  <complexType name="ruleBaseType">
    <complexContent>
      <extension base="testRule:ruleBasePrototype">
        <choice minOccurs="0" maxOccurs="unbounded">
          <element ref="profile:importBase"/>
        </choice>
      </extension>
    </complexContent>
  </complexType>

  <element name="ruleBase" type="profile:ruleBaseType"
substitutionGroup="testRule:ruleBase"/>

  <!-- ========================================================= -->
  <!-- ContentControl                                            -->
  <!-- ========================================================= -->
  <!-- extends switch element -->

  <!-- switch interface element groups -->
  <group name="switchInterfaceElementGroup">
    <choice>
      <element ref="profile:port"/>
      <element ref="profile:switchPort"/>
    </choice>
  </group>

  <complexType name="switchType">
    <complexContent>
      <extension base="contentControl:switchPrototype">
        <choice minOccurs="0" maxOccurs="unbounded">
          <group ref="profile:switchInterfaceElementGroup"/>
          <element ref="profile:descriptorBind"/>
          <element name="bindRule">
            <complexType>
              <attribute name="rule" type="string" use="required"/>
              <attribute name="component" type="string"
use="required"/>
            </complexType>
          </element>
          <element ref="profile:switch"/>
          <element ref="profile:media"/>
          <element ref="profile:context"/>
        </choice>
        <attributeGroup ref="reuseEntity:reuseEntityAttrs"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="switch" type="profile:switchType"
substitutionGroup="contentControl:switch"/>

  <!-- ========================================================= -->
  <!-- DescriptorControl                                         -->
  <!-- ========================================================= -->
```

```
  <!-- extends descriptorSwitch element -->

  <complexType name="descriptorSwitchType">
    <complexContent>
      <extension base="descriptorControl:descriptorSwitchPrototype">
        <sequence minOccurs="0" maxOccurs="unbounded">
          <element ref="profile:descriptor"/>
          <element name="bindRule">
            <complexType>
             <attribute name="rule" type="string" use="required"/>
             <attribute name="component" type="string"
use="required"/>
            </complexType>
          </element>
        </sequence>
        <attributeGroup ref="reuseEntity:reuseEntityAttrs"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="descriptorSwitch" type="profile:descriptorSwitchType"
substitutionGroup="descriptorControl:descriptorSwitch"/>

  <!-- ========================================================== -->
  <!-- Timing                                                     -->
  <!-- ========================================================== -->

  <!-- extends costFunction element -->
  <complexType name="costFunctionType">
    <complexContent>
      <extension base="timing:costFunctionPrototype">
        <attributeGroup ref="reuseEntity:reuseEntityAttrs"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="costFunction" type="profile:costFunctionType"
substitutionGroup="timing:costFunction"/>

  <!-- extends costFunctionBase element -->
  <complexType name="costFunctionBaseType">
    <complexContent>
      <extension base="timing:costFunctionBasePrototype">
        <choice minOccurs="0" maxOccurs="unbounded">
          <element ref="profile:importBase"/>
        </choice>
      </extension>
    </complexContent>
  </complexType>

  <element name="costFunctionBase" type="profile:costFunctionBaseType"
substitutionGroup="timing:costFunctionBase"/>


  <!-- ========================================================== -->
  <!-- Import                                                  -->
  <!--========================================================== -->
  <complexType name="importBaseType">
    <complexContent>
      <extension base="import:importBasePrototype">
      </extension>
```

```
    </complexContent>
  </complexType>

  <complexType name="importNCLType">
    <complexContent>
      <extension base="import:importNCLPrototype">
      </extension>
    </complexContent>
  </complexType>

  <complexType name="importedDocumentBaseType">
    <complexContent>
      <extension base="import:importedDocumentBasePrototype">
      </extension>
    </complexContent>
  </complexType>

  <element name="importBase" type="profile:importBaseType"
substitutionGroup="import:importBase"/>

  <element name="importNCL" type="profile:importNCLType"
substitutionGroup="import:importNCL"/>
  <element name="importedDocumentBase"
type="profile:importedDocumentBaseType"
substitutionGroup="import:importedDocumentBase"/>

  <!-- ============================================================ -->
  <!-- ReuseEntity                                                  -->
  <!-- ============================================================ -->
</schema>
```

5. Authoring an NCL 2.3 Document: An Example

This section presents a step-by-step specification of an NCL document, following the language profile specification of Section 4. The example comprises a main video (a track of the MATRIX film with a dialog between two characters: Morpheus and Neo), with two advertisement videos synchronized with the main video. The first one is a battery advertisement synchronized with the moment that Morpheus raises a battery. The second one is presented when a user interacts with an icon placed over Morpheus' eyeglasses. In this last case, the main video is paused and resized during the advertisement presentation and a text form is presented at the end of the advertisement to offer user the opportunity to initiate a transaction to buy the eyeglasses. Depending on the selected language, the English or the Portuguese text form is presented to the user.

The specification begins declaring in the body element which media objects will take part of the document, as shown in Figure 1. A port element in the body gives the start point for the document presentation. This entry point is bind to the maestroTV element and to its default "whole content anchor" (since the maestroTV interface is not specified), indicating the image TV800.jpg as the starting point. TV800.jpg is only an image that simulates a 3:4 aspect ratio TV device where the presentation will take place.

The setting element (media type equals to setting) groups all environment (context aware) variables. In NCL, any variable type can be specified. Every variable used in relationship specifications must be explicitly declared in the setting type media element. In the example, the variable "language" is explicitly declared as an attribute of the setting element.

The matrix video definition has two anchors specifying video tracks used to synchronize the battery video presentation and the icon image presentation. As mentioned before, the icon will be placed over the matrix video presentation in a region that matches the position of Morpheus' eyeglasses at this time. The video matrix definition has also explicitly declared its positioning attributes, since they will be used when resizing the video. Again, every media attribute used in relationship specifications must be explicitly declared. Note also that all media objects reference descriptors used to control their presentation. Descriptor specifications are made in the document head, as will be discussed further on.

A context element (r) is used to group all media objects related to the glasses advertisement: the video, the icon for interaction and the text forms. The grouping can be very useful, for example, if one wants to repeat (reuse) the advertisement in another part of the film. It is also useful as a structuring facility. The context element has three port elements, exporting context internal element interfaces to be used in relationships declared by links.

A switch element allows the selection of the form text element to be presented, depending on which rule is satisfied. Rule specifications are made in the document head, as will be discussed further on. Similar to the context element, switches also have port elements exporting internal element interfaces to be used in relationships declared by links.

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <ncl id="matrixExample"
    xmlns="http://www.telemidia.puc-rio.br/specs/xml/NCL23/profiles"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/profiles http://www.telemidia.puc-
rio.br/specs/xml/NCL23/profiles/NCL23.xsd">

03 <head>


40 </head>

41 <body>
42   <port id="entryPoint" component="maestroTV"/>

43   <media id="maestroTV" type="image" descriptor="maestroDesc"
      src="example/TV800.jpg"/>
44   <media type="setting" id="setting" >
45    <attribute id="language" name="language" type="xsi:string"/>
46   </media>

47   <media id="matrix" type="video" descriptor="matrixDesc"
      src="../../videos/matrix/50-whatisthematrix.mpg">
48     <area id="anchorBattery" begin="18s" end="19s"/>
49     <area id="anchorGlasses" begin="27s" end="32s" />
50     <attribute id="left" name="left" type="xsi:string"/>
51     <attribute id="top" name="top" type="xsi:string"/>
52     <attribute id="height" name="height" type="xsi:string"/>
53     <attribute id="width" name="width" type="xsi:string"/>
54   </media>
55   <media id="battery" type="video" descriptor="batteryDesc"
      src="../../videos/matrix/51-battery.mpg"/>

56   <context id="glassesPub">
57     <port id="glassesIconInt" component="glassesIcon"/>
58     <port id="glassesInt" component="glasses"/>
59     <port id="glassesFormInt" component="form" interface="entryForm"/>

60     <media id="glassesIcon" type="video" descriptor="glassesIconDesc"
        src="../../videos/matrix/glasses-blinks.avi"/>
61     <media id="glasses" type="video" descriptor="glassesDesc"
        src="../../videos/matrix/52-glassesAd.mpg"/>

62     <switch id="form">
63       <switchPort id="entryForm">
64          <mapping component="ptForm"/>
65          <mapping component="enForm"/>
66       </switchPort>
67       <bindRule rule="rPt" component="ptForm"/>
68       <bindRule rule="rEn" component="enForm"/>
69       <media id="ptForm" type="text" descriptor="formDesc"
            src="example/ptForm.html"/>
70       <media id="enForm" type="text" descriptor="formDesc"
          src="example/enForm.html"/>
71     </switch>


76   </context>


136 </body>
137 </ncl>
```

Figure 1 – Component specification

The layout, declaring the positioning of each media object presentation, is then specified in the *head* element, as shown in Figure 2. One region representing the device (the simulated MaestroTV) is specified, whose background is set to black. Another region (TVRegion) specifies the exhibition area (MaestroTV screen). Five (sub) regions are then specified for the document media object placements.

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <ncl id="matrixExample"
    xmlns="http://www.telemidia.puc-rio.br/specs/xml/NCL23/profiles"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/profiles http://www.telemidia.puc-
rio.br/specs/xml/NCL23/profiles/NCL23.xsd">

03 <head>

04   <regionBase>
05    <region background="black" id="maestroRegion" title="Matrix Example"
       top="0" left="0" height="640" width="810" zIndex="4"/>
06    <region background="black" id="TVRegion" top="35" left="5" height="600"
       width="800" zIndex="3">
07      <region background="black" id="matrixRegion" left="0%" top="0%"
         height="540" width="800" zIndex="2"/>
08      <region background="black" id="pubBatteryRegion" left="68%" top="68%"
         height="32%" width="32%" zIndex="1"/>
09      <region background="black" id="glassesIconRegion" left="12%"
         top="24%" height="39" width="68" zIndex="1"/>
10      <region background="black" id="pubGlassesRegion" left="53%" top="43%"
         height="37%" width="47%" zIndex="1"/>
11      <region id="formRegion" left="53%" top="43%" height="37%" width="47%"
         zIndex="1"/>
12    </region>
13   </regionBase>


40 </head>

41 <body>


136 </body>
137 </ncl>
```

Figure 2 – Layout specification

In Figure 3, descriptor elements bind media objects and their presentation regions, besides specifying other media presentation characteristics. Descriptors are grouped in descriptor bases. Similarly, presentation rules are grouped in rule bases.

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <ncl id="matrixExample"
    xmlns="http://www.telemidia.puc-rio.br/specs/xml/NCL23/profiles"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/profiles http://www.telemidia.puc-
rio.br/specs/xml/NCL23/profiles/NCL23.xsd">

03 <head>


14   <descriptorBase>
15     <descriptor id="maestroDesc" region="maestroRegion"/>
16     <descriptor id="matrixDesc" region="matrixRegion">
```

```
17         <descriptorParam name="soundLevel" value="1"/>
18         <descriptorParam name="controlVisible" value="false"/>
19       </descriptor>
20       <descriptor id="batteryDesc" region="pubBatteryRegion">
21         <descriptorParam name="soundLevel" value="1"/>
22         <descriptorParam name="controlVisible" value="false"/>
23       </descriptor>
24       <descriptor id="glassesDesc" region="pubGlassesRegion">
25         <descriptorParam name="soundLevel" value="1"/>
26         <descriptorParam name="controlVisible" value="false"/>
27       </descriptor>
28       <descriptor id="formDesc" region="formRegion" explicitDur="3s"/>
29       <descriptor id="glassesIconDesc" region="glassesIconRegion">
30         <descriptorParam name="controlVisible" value="false"/>
31       </descriptor>
32    </descriptorBase>

33    <ruleBase>
34       <rule id="rEn" var="language" op="eq" value="en" />
35       <rule id="rPt" var="language" op="eq" value="pt" />
36    </ruleBase>


40  </head>

41  <body>

136 </body>
137 </ncl>
```

Figure 3 – Descriptor and presentation rule specification

Finally, synchronization relationships among document components are specified in the context element and in the body element, through their link child elements, as shown in Figure 4. Connectors referenced by links (xconnector attribute) are imported from external connector bases whose URI are specified in connectorBase elements declared as child elements of the head element. The imported connector base is presented in Appendix B.

In the figure, Link00 starts the matrix video and sets the language variable to pt (Portuguese). Link01 starts the battery video when the matrix video enters the track specified as anchorBattery. Link 02 starts the icon presentation when the matrix video enters the track specified as anchorGlasses. Link03 stops the icon presentation when the track corresponding to the anchorGlasses finishes its presentation. Link04 specifies that the icon selection (by the yellow key of the TV remote control) resizes and pauses the matrix video, stops the icon presentation, and starts the glasses publicity video. Link05 defines that when the eyeglasses publicity finishes, the matrix video is resumed with its original size. Link07 (inside the context element) starts the text form presentation when the eyeglasses publicity video finishes and Link06 finishes TVMaestro image presentation when the matrix video also finishes. This ends the document presentation.

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <ncl id="matrixExample"
    xmlns="http://www.telemidia.puc-rio.br/specs/xml/NCL23/profiles"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/profiles http://www.telemidia.puc-
rio.br/specs/xml/NCL23/profiles/NCL23.xsd">

03 <head>
```

```
37    <connectorBase>
38      <importBase alias="itv" baseURI="maestroConnectorBase.ncl"/>
39    </connectorBase>

42  </head>

43  <body>
43    <port id="entryPoint" component="maestroTV"/>


56    <context id="glassesPub">


72      <link id="Link07" xconnector="itv#onEnd1Start1">
73        <bind role="onEnd" component="glasses"/>
74        <bind role="start" component="form" interface="entryForm"/>
75      </link>

76    </context>


77    <link id="Link00" xconnector="itv#onBegin1SetVar1Start1">
78      <bind component="maestroTV" role="onBegin"/>
79      <bind component="setting" interface="language" role="set">
80        <bindParam name="var" value="pt"/>
81      </bind>
82      <bind component="matrix" role="start"/>
83    </link>

84    <link id="Link01" xconnector="itv#onBegin1Start1">
85      <bind role="onBegin" component="matrix" interface="anchorBattery"/>
86      <bind role="start" component="battery"/>
87    </link>

88    <link id="Link02" xconnector="itv#onBegin1Start1">
89      <bind role="onBegin" component="matrix" interface="anchorGlasses"/>
90      <bind role="start" component="glassesPub" interface="glassesIconInt"/>
91    </link>

92    <link id="Link03" xconnector="itv#onEnd1Stop1">
93      <bind role="onEnd" component="matrix" interface="anchorGlasses"/>
94      <bind role="stop" component="glassesPub" interface="glassesIconInt"/>
95    </link>

96    <link id="Link04"
        xconnector="itv#onKeySelection1Stop1ResizePause1Start1">
97      <bind role="onKeySelection" component="glassesPub"
          interface="glassesIconInt">
98        <bindParam name="keyCode" value="YELLOW"/>
99      </bind>
100     <bind component="glassesPub" interface="glassesIconInt" role="stop"/>
101     <bind component="matrix" interface="width" role="setWidth">
102       <bindParam name="width" value="424"/>
103     </bind>
104     <bind component="matrix" interface="height" role="setHeight">
105       <bindParam name="height" value="286"/>
106     </bind>
107     <bind component="matrix" interface="left" role="setLeft">
108       <bindParam name="left" value="0"/>
109     </bind>
110     <bind component="matrix" interface="top" role="setTop">
111       <bindParam name="top" value="20"/>
112     </bind>
113     <bind role="pause" component="matrix"/>
114     <bind role="start" component="glassesPub" interface="glassesInt"/>
115   </link>

116   <link id="Link05" xconnector="itv#onEnd1ResizeResume1">
```

```
117     <bind role="onEnd" component="glassesPub" interface="glassesFormInt"/>
118     <bind component="matrix" interface="left" role="setLeft">
119        <bindParam name="left" value="0"/>
120     </bind>
121     <bind component="matrix" interface="top" role="setTop">
122        <bindParam name="top" value="0"/>
123     </bind>
124     <bind component="matrix" interface="width" role="setWidth">
125        <bindParam name="width" value="800"/>
126     </bind>
127     <bind component="matrix" interface="height" role="setHeight">
128        <bindParam name="height" value="540"/>
129     </bind>
130     <bind component="matrix" role="resume"/>
131   </link>

132   <link id="Link06" xconnector="itv#onEnd1Stop1">
133     <bind role="onEnd" component="matrix"/>
134     <bind role="stop" component="maestroTV"/>
135   </link>

136 </body>
137 </ncl>
```

Figure 4 – Relationship specification

It is worth to be noted the orders in which binds are declared in Link04 and Link05. Indeed, these orders must be obeyed in the connector definition and are very important. The region is altered step by step, that is attribute by attribute, by the formatter. If in any time of this process the region does not fit in its parent region an inconsistency is reached and the process fail, even if the initial and final region definitions are consistent. In order to avoid this problem, group of attributes should be used. In the case of Link04 and Link05, the attribute group named *bounds* is the most appropriate. When a formatter treats an attribute group it only tests the consistency at the end of the process, as mentioned in Section 2.1.4.

The complete document specification, with the modifications in Link04, Link05 and matrix attributes, is shown in Figure 5.

```
<?xml version="1.0" encoding="UTF-8"?>
<ncl id="matrixExample"
  xmlns="http://www.telemidia.puc-rio.br/specs/xml/NCL23/profiles"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.telemidia.puc-rio.br/specs/xml/NCL23/profiles
  http://www.telemidia.puc-rio.br/specs/xml/NCL23/profiles/NCL23.xsd">

  <head>

    <regionBase>
       <region background="black" id="maestroRegion" title="Matrix Example"
            top="0" left="0" height="640" width="810" zIndex="3"/>
       <region background="black" id="TVRegion" top="35" left="5" height="600"
            width="800" zIndex="3">
         <region background="black" id="matrixRegion" left="0%" top="0%"
             height="540" width="800" zIndex="2"/>
         <region background="black" id="pubBatteryRegion" left="68%" top="68%"
             height="32%" width="32%" zIndex="1"/>
         <region background="black" id="glassesIconRegion" left="12%" top="24%"
             height="39" width="68" zIndex="1"/>
         <region background="black" id="pubGlassesRegion" left="53%" top="43%"
             height="37%" width="47%" zIndex="1"/>
         <region id="formRegion" left="53%" top="43%" height="37%" width="47%"
             zIndex="1"/>
```

```
      </region>
   </regionBase>

   <descriptorBase>
      <descriptor id="maestroDesc" region="maestroRegion"/>
      <descriptor id="matrixDesc" region="matrixRegion">
        <descriptorParam name="soundLevel" value="1"/>
        <descriptorParam name="controlVisible" value="false"/>
      </descriptor>
      <descriptor id="batteryDesc" region="pubBatteryRegion">
        <descriptorParam name="soundLevel" value="1"/>
        <descriptorParam name="controlVisible" value="false"/>
      </descriptor>
      <descriptor id="glassesDesc" region="pubGlassesRegion">
        <descriptorParam name="soundLevel" value="1"/>
        <descriptorParam name="controlVisible" value="false"/>
      </descriptor>
      <descriptor id="formDesc" region="formRegion" explicitDur="3s"/>
      <descriptor id="glassesIconDesc" region="glassesIconRegion">
        <descriptorParam name="controlVisible" value="false"/>
      </descriptor>
   </descriptorBase>

   <ruleBase>
      <rule id="rEn" var="language" op="eq" value="en" />
      <rule id="rPt" var="language" op="eq" value="pt" />
   </ruleBase>

   <connectorBase>
     <importBase alias="itv" baseURI="maestroConnectorBase.ncl"/>
   </connectorBase>

 </head>

 <body>
  <port id="entryPoint" component="maestroTV"/>

  <media id="maestroTV" type="image" descriptor="maestroDesc"
     src="example/TV800.jpg"/>
  <media type="setting" id="setting" >
    <attribute id="language" name="language" type="xsi:string"/>
  </media>

  <media id="matrix" type="video" descriptor="matrixDesc"
     src="../../videos/matrix/50-whatisthematrix.mpg">
    <area id="anchorBattery" begin="18s" end="19s"/>
    <area id="anchorGlasses" begin="27s" end="32s" />
    <attribute id="bounds" name="bounds" type="xsi:string"/>
  </media>

  <media id="battery" type="video" descriptor="batteryDesc"
     src="../../videos/matrix/51-battery.mpg"/>

  <context id="glassesPub">
    <port id="glassesIconInt" component="glassesIcon"/>
    <port id="glassesInt" component="glasses"/>
    <port id="glassesFormInt" component="form" interface="entryForm"/>

    <media id="glassesIcon" type="video" descriptor="glassesIconDesc"
     src="../../videos/matrix/glasses-blinks.avi"/>
    <media id="glasses" type="video" descriptor="glassesDesc"
     src="../../videos/matrix/52-glassesAd.mpg"/>

    <switch id="form">
      <switchPort id="entryForm">
         <mapping component="ptForm"/>
         <mapping component="enForm"/>
      </switchPort>
```

```
      <bindRule rule="rPt" component="ptForm"/>
      <bindRule rule="rEn" component="enForm"/>
      <media id="ptForm" type="text" descriptor="formDesc"
        src="example/ptForm.html" />
      <media id="enForm" type="text" descriptor="formDesc"
        src="example/enForm.html" />
    </switch>

    <link id="Link07" xconnector="itv#onEnd1Start1">
      <bind role="onEnd" component="glasses"/>
      <bind role="start" component="form" interface="entryForm"/>
    </link>

  </context>

  <link id="Link00" xconnector="itv#onBegin1SetVar1Start1">
     <bind component="maestroTV" role="onBegin"/>
     <bind component="setting" interface="language" role="set">
        <bindParam name="var" value="pt"/>
     </bind>
     <bind component="matrix" role="start"/>
  </link>

  <link id="Link01" xconnector="itv#onBegin1Start1">
     <bind role="onBegin" component="matrix" interface="anchorBattery"/>
     <bind role="start" component="battery"/>
  </link>

  <link id="Link02" xconnector="itv#onBegin1Start1">
     <bind role="onBegin" component="matrix" interface="anchorGlasses"/>
     <bind role="start" component="glassesPub" interface="glassesIconInt"/>
  </link>

  <link id="Link03" xconnector="itv#onEnd1Stop1">
     <bind role="onEnd" component="matrix" interface="anchorGlasses"/>
     <bind role="stop" component="glassesPub" interface="glassesIconInt"/>
  </link>

  <link id="Link04" xconnector="itv#onKeySelection1Stop1ResizePause1Start1">
    <bind component="glassesPub" interface="glassesIconInt"
     role="onKeySelection">
       <bindParam name="keyCode" value="YELLOW"/>
    </bind>
    <bind component="glassesPub" interface="glassesIconInt" role="stop"/>
    <bind component="matrix" interface="bounds" role="setBounds">
       <bindParam name="bounds" value="0,20,424,286"/>
    </bind>
    <bind role="pause" component="matrix"/>
    <bind role="start" component="glassesPub" interface="glassesInt"/>
  </link>

  <link id="Link05" xconnector="itv#onEnd1ResizeResume1">
    <bind role="onEnd" component="glassesPub" interface="glassesFormInt"/>
    <bind component="matrix" interface="bounds" role="setBounds">
       <bindParam name="bounds" value="0,0,800,540"/>
    </bind>
    <bind component="matrix" role="resume"/>
  </link>

  <link id="Link06" xconnector="itv#onEnd1Stop1">
    <bind role="onEnd" component="matrix"/>
    <bind role="stop" component="maestroTV"/>
  </link>

 </body>
</ncl>
```

Figure 5 – Complete document specification

6. The XConnector Profile

A hypermedia connector [SoRo05] represents a relation that can be used for creating links in hyperdocuments. A connector specifies a relation independently of relationships, that is, it does not specify which nodes will interact through the relation. Links representing the same type of relation, but interconnecting different nodes, can reuse the same connector, reusing all previous specifications. A hypermedia connector specifies a set of interface points, called roles. A link refers to a connector and defines a set of binds, which associate each link endpoint (node interface) to a role of the connector used.

XConnector Profile is an XML-based language for the definition of connectors that can be used to specify reference and spatio-temporal hypermedia synchronization relations, treating reference relations as a particular case of synchronization relations.

The XConnector profile allows the definition of multipoint relations with causal or constraint semantics, as discussed in [SoRo05]. That reference gives the basis for all concepts used in this section.

Relations in XConnector are event based. As stated in [SoRo05], events in NCM may be instantaneous or have a measurable duration. Relations are defined based on event states or changes on the event state machine, as shown in Figure 5. Table 20 defines transition names for an event state machine.

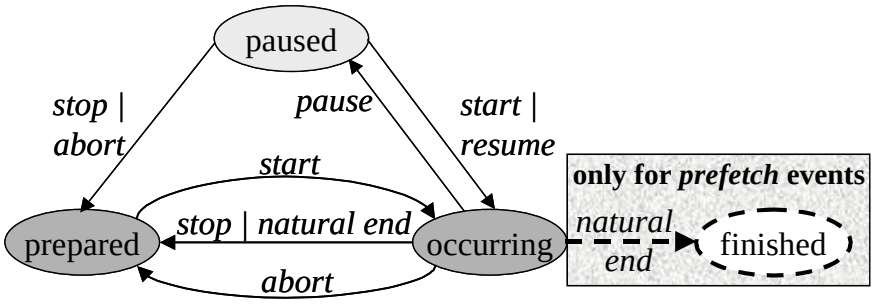

Figure 5 - Event state machine

Table 20 - Transition names for an event state machine

Transition (caused by action)	Transition Name
prepared→occurring (start)	*starts*
occurring→prepared (stop or natural end)	*stops*
occurring→prepared (abort)	*aborts*
occurring→finished (natural end)	*ends*[7]
occurring→paused (pause)	*pauses*
paused→occurring (resume or start)	*resumes*
paused→prepared (stop or abort)	*abortsFromPaused*

Each connector *role* defines an id, which has to be unique in the connector role set, an event type (*eventType*) and its cardinality. The role cardinality specifies the minimal (*min*) and maximal (*max*) number of participants that may play this role

[7] Valid only for *prefetch* type events.

(number of binds) when this connector is used for creating a link, as will be defined later. If a role event type is *attribution*, the role must also define the corresponding attribute name. If a role type is *selection*, the role may also define to which selection apparatus (for example, keyboard or remote control keys) it refers, through its key attribute. Roles are specialized in action roles, condition roles and assessment roles. Different types of roles are used depending on the connector type. In constraint connectors, only assessment roles are allowed. In causal connectors, any type of role may be used. Action roles (*actionRole* element) capture actions that can be executed in causal relations.

When the maximal cardinality value of a role is greater than one, several participants may play the same role. In this case, a *qualifier* must be specified each time the role is used in the glue expressions. Table12 presents possible qualifier values.

Table 21 - Role qualifier values

Role type	Qualifier	Semantics
Condition	*all*	All conditions must be true
Condition	*any*	At least one condition must be true
Assessment	*all*	All assessments must be considered
Assessment	*any*	At least one assessment must be considered
Action	*par*	All actions must be executed in parallel
Action	*excl*	Only one of the actions must be executed

The XConnector *glue* specifies how connector roles interact. Every connector role must be used in the glue. Glues are specialized in causal glues and constraint glues.

A causal connector has causal glue, which defines a *triggerExpression*, relating condition and assessment roles, and an *actionExpression*, relating action roles. When the trigger expression is satisfied, the action expression must be executed.

Statement expressions relate assessment roles and are used in constraint connectors for defining spatio-temporal constraints.

Some attribute values from connector roles and glues can be parameterized. The definition of connector parameters can be done declaring *connectorParam* child-elements, which have *name* and *type* attributes. In order to specify which attributes receive parameter values defined by the connector, their values are specified as the parameter name preceded by the *$* symbol. For instance, in order to parameterize the delay attribute, a parameter called *actionDelay* is defined (<*connectorParam name=”actionDelay” type=”unsignedLong”*/>) and the value *“$actionDelay”* is used in the attribute (*delay*=”$*actionDelay*”).

Since the definition of connectors is not easily done by naïve users, the idea is to have expert users defining connectors, storing them in libraries (*connectorBases*) and making them available to others for creating links. As an example of connector base, the well-known synchronization relations defined by Allen [Alle83] are specified in Appendix A. Appendix B gives another extensive example of connector base.

Table 22 reshows the Connector Module already presented in Table 12. In the content specification, the same symbols of Table 12 were used: (?) optional, (|) or, (*) zero or more occurrences, (+) one or more occurrences.

Section 6.1 presents the XConnector module specification using XML Schema, and Section 6.2, the XConnector 2.3 Language Specification Schema (XConnector Profile).

Table 22 – XConnector Module

Elements	Attributes	Content
causalConnector	*id, name, description*	*(connectorParam*, conditionRole+, assessmentRole*, actionRole+, causalGlue)*
constraintConnector	*id, name, description*	*(connectorParam*, assessmentRole+, constraintGlue)*
actionRole	*id, eventType, min, max*	*(presentationAction | assignmentAction)*
presentationAction	*actionType*	*empty*
assignmentAction	*value*	*empty*
conditionRole	*id, eventType, min, max, key*	*(eventStateTransitionCondition | attributeCondition | compoundCondition)*
compoundCondition	*isNegated, operator*	*(eventStateTransitionCondition* | attributeCondition* | compoundCondition*)+*
eventStateTransitionCondition	*transition*	*empty*
attributeCondition	*comparator, attributeType, value*	*empty*
assessmentRole	*id, eventType, min, max, key*	*(eventStateTransitionAssessment | attributeAssessment)*
eventStateTransitionAssessment	*transitionName*	*empty*
attributeAssessment	*attributeType*	*empty*
causalGlue		*((simpleTriggerExpression |compoundTriggerExpression), (simpleActionExpresssion | compoundActionExpresssion))*
constraintGlue		*(assessmentStatement | assessmentValueStatement | assessmentEventStatement | compoundStatement)*
simpleTriggerExpression	*minDelay, maxDelay, coditionRole, qualifier*	
compoundTriggerExpression	*isNegated, minDelay, maxDelay, logicalOperatorType*	*((simpleTriggerExpression | compoundTriggerExpression)+ , (assessmentStatement | assessmentValueStatement |*

		assessmentEventStatement | compoundStatement))*
assessmentStatement	*comparator, firstAssessmentRole, firstAssessmentOffset firstRoleQualifier, secondAssessmentRole secondAssessmentOffset, secondRoleQualifier,*	
assessmentValueStatement	*comparator, assessmentRole, assesmentOffset, roleQualifier, value*	
compoundStatement	*operator, isNegated*	*(assessmentStatement | assessmentValueStatement | assessmentEventStatement | compoundStatement)+*
simpleActionExpresssion	*delay,actionRole, qualifier, repeat, repeatDelay*	
compoundActionExpresssion	*delay,operator*	*(simpleActionExpresssion | compoundActionExpresssion)+*
connectorParam	*name, type*	*Empty*
connectorBase	*id, name, description*	(*causalConnector**, *constraintConnector**)

6.1. The XConnector 2.3 Module Specification in XML Schema

XConnector Module: NCL23XConnector.xsd

```
<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23XConnector.xsd
Author: TeleMidia Laboratory
Revision: 29/06/2005

Schema for the NCL XConnector module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:connector="http://www.telemidia.puc-rio.br/specs/xml/NCL23/XConnector"
  targetNamespace="http://www.telemidia.puc-rio.br/specs/xml/NCL23/XConnector"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

<complexType name="parameterType" >
```

```
  <attribute name="name" type="string" use="required"/>
  <attribute name="type" type="string" use="required"/>
</complexType>

<element name="connectorParam" type="connector:parameterType"/>

<complexType name="hypermediaConnectorType" abstract="true">
  <sequence>
    <element ref="connector:connectorParam" minOccurs="0"
maxOccurs="unbounded"/>
  </sequence>
  <attribute name="id" type="ID" use="required"/>
  <attribute name="name" type="string"/>
  <attribute name="description" type="string"/>
</complexType>

<complexType name="causalConnectorPrototype">
  <complexContent>
    <extension base="connector:hypermediaConnectorType">
      <sequence>
        <element ref="connector:conditionRole" minOccurs="1"
maxOccurs="unbounded"/>
        <element ref="connector:assessmentRole" minOccurs="0"
maxOccurs="unbounded" />
        <element ref="connector:actionRole" minOccurs="1"
maxOccurs="unbounded"/>
        <element ref="connector:causalGlue"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<complexType name="constraintConnectorPrototype">
  <complexContent>
    <extension base="connector:hypermediaConnectorType">
      <sequence>
        <element ref="connector:assessmentRole" minOccurs="2"
maxOccurs="unbounded"/>
        <element ref="connector:constraintGlue"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<element name="causalGlue" type="connector:causalGlueType"/>
<element name="constraintGlue" type="connector:constraintGlueType"/>

<element name="conditionRole" type="connector:conditionRoleType"/>
<element name="assessmentRole" type="connector:assessmentRoleType" />
<element name="actionRole" type="connector:actionRoleType" />

<complexType name="roleType" abstract="true">
  <attribute name="id" type="string" use="required"/>
  <attribute name="eventType" type="connector:eventType" use="required"/>
  <attributeGroup ref="connector:cardinality"/>
</complexType>

<simpleType name="eventType">
  <restriction base="string">
    <enumeration value="presentation" />
    <enumeration value="selection" />
    <enumeration value="attribution" />
    <enumeration value="pointOver" />
    <enumeration value="prefetch" />
    <enumeration value="focus" />
  </restriction>
</simpleType>
```

```
<attributeGroup name="cardinality">
  <attribute name="min" type="positiveInteger" default="1"/>
  <attribute name="max" type="connector:maxUnion" default="1"/>
</attributeGroup >

<simpleType name="maxUnion">
  <union memberTypes="positiveInteger connector:unboundedString"/>
</simpleType>

<simpleType name="unboundedString">
  <restriction base="string">
    <pattern value="unbounded"/>
  </restriction>
</simpleType>

<complexType name="actionRoleType">
  <complexContent>
    <extension base="connector:roleType">
       <group ref="connector:actionGroup"/>
    </extension>
  </complexContent>
</complexType>

<group name="actionGroup">
  <choice>
    <element ref="connector:presentationAction"/>
    <element ref="connector:assignmentAction"/>
  </choice>
</group>

<element name="presentationAction" type="connector:presentationActionType"/>
<element name="assignmentAction" type="connector:assignmentActionType"/>

<complexType name="presentationActionType">
  <attribute name="actionType" type="connector:presentationActionNameType"
use="required"/>
</complexType>

<complexType name="assignmentActionType">
  <attribute name="value" type="anySimpleType" use="required"/>
</complexType>

<simpleType name="presentationActionNameType">
  <restriction base="string">
    <enumeration value="start" />
    <enumeration value="stop" />
    <enumeration value="pause" />
    <enumeration value="resume" />
    <enumeration value="abort" />
  </restriction>
</simpleType>

<simpleType name="unsignedLongParamUnion">
  <union memberTypes="unsignedLong string"/>
</simpleType>

<simpleType name="repeatUnion">
  <union memberTypes="nonNegativeInteger connector:unboundedString string"/>
</simpleType>

<complexType name="conditionRoleType">
  <complexContent>
    <extension base="connector:roleType">
      <group ref="connector:conditionGroup"/>
      <attribute name="key" type="string" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

```
<group name="conditionGroup">
  <choice>
    <element ref="connector:eventStateTransitionCondition"/>
    <element ref="connector:attributeCondition"/>
    <element ref="connector:compoundCondition"/>
  </choice>
</group>

<element name="eventStateTransitionCondition"
type="connector:eventStateTransitionConditionType"/>
<element name="attributeCondition" type="connector:attributeConditionType"/>
<element name="compoundCondition" type="connector:compoundConditionType"/>

<complexType name="eventStateTransitionConditionType">
  <attribute name="transition" type="connector:transitionType"
use="required"/>
</complexType>

<complexType name="attributeConditionType">
  <attribute name="comparator" type="connector:comparatorType"
use="required"/>
  <attribute name="attributeType" type="connector:attributeType"
use="required"/>
  <attribute name="value" type="anySimpleType" use="required"/>
</complexType>

<complexType name="compoundConditionType" >
  <sequence>
      <group ref="connector:conditionGroup" minOccurs="2" maxOccurs="2"/>
  </sequence>
  <attribute name="operator" type="connector:logicalOperatorType"
use="required"/>
  <attribute name="isNegated" type="boolean" default="false"/>
</complexType>

<simpleType name="logicalOperatorType">
  <restriction base="string">
    <enumeration value="and" />
    <enumeration value="or" />
  </restriction>
</simpleType>

<simpleType name="comparatorType">
  <restriction base="string">
    <enumeration value="eq" />
    <enumeration value="ne" />
    <enumeration value="gt" />
    <enumeration value="lt" />
    <enumeration value="gte" />
    <enumeration value="lte" />
  </restriction>
</simpleType>

<simpleType name="stateType">
  <restriction base="string">
    <enumeration value="prepared" />
    <enumeration value="occurring" />
    <enumeration value="paused" />
    <enumeration value="finished" />
  </restriction>
</simpleType>

<simpleType name="transitionType">
  <restriction base="string">
    <enumeration value="starts" />
    <enumeration value="stops" />
    <enumeration value="pauses" />
```

```
    <enumeration value="resumes" />
    <enumeration value="aborts" />
    <enumeration value="abortsFromPaused" />
    <enumeration value="ends" />
  </restriction>
</simpleType>

<simpleType name="attributeType">
  <restriction base="string">
    <enumeration value="repeat" />
    <enumeration value="occurences" />
    <enumeration value="state" />
    <enumeration value="nodeAttribute" />
  </restriction>
</simpleType>

<complexType name="assessmentRoleType">
  <complexContent>
    <extension base="connector:roleType">
      <group ref="connector:assessmentGroup"/>
      <attribute name="key" type="string" use="optional"/>
    </extension>
  </complexContent>
</complexType>

<group name="assessmentGroup">
  <choice>
    <element ref="connector:eventStateTransitionAssessment"/>
    <element ref="connector:attributeAssessment"/>
  </choice>
</group>

<element name="eventStateTransitionAssessment"
type="connector:eventStateTransitionAssessmentType"/>
<element name="attributeAssessment" type="connector:attributeAssessmentType"/>

<complexType name="eventStateTransitionAssessmentType">
  <attribute name="transition" type="connector:transitionType"
use="required"/>
</complexType>

<complexType name="attributeAssessmentType">
  <attribute name="attributeType" type="connector:attributeType"
use="required"/>
</complexType>

<complexType name="causalGlueType">
  <sequence>
    <group ref="connector:triggerExpressionGroup"/>
    <group ref="connector:actionExpressionGroup"/>
  </sequence>
</complexType>

<complexType name="constraintGlueType">
    <group ref="connector:statementExpressionGroup"/>
</complexType>

<complexType name="triggerExpressionType" abstract="true">
  <attribute name="minDelay" type="connector:unsignedLongParamUnion"
default="0"/>
  <attribute name="maxDelay" type="connector:unsignedLongParamUnion"
default="0"/>
</complexType>

<complexType name="simpleTriggerExpressionType">
  <complexContent>
    <extension base="connector:triggerExpressionType">
      <attribute name="conditionRole" type="string" use="required"/>
```

```
      <attribute name="qualifier" type="connector:conditionQualifierType"/>
    </extension>
  </complexContent>
</complexType>

<simpleType name="conditionQualifierType">
  <restriction base="string">
    <enumeration value="all" />
    <enumeration value="any" />
  </restriction>
</simpleType>

<complexType name="compoundTriggerExpressionType" >
  <complexContent>
    <extension base="connector:triggerExpressionType">
      <sequence>
        <group ref="connector:triggerExpressionGroup"/>
        <choice minOccurs="1" maxOccurs="unbounded">
          <group ref="connector:triggerExpressionGroup"/>
          <group ref="connector:statementExpressionGroup"/>
        </choice>
      </sequence>
      <attribute name="operator" type="connector:logicalOperatorType"
use="required"/>
      <attribute name="isNegated" type="boolean" default="false"/>
    </extension>
  </complexContent>
</complexType>

<element name="simpleTriggerExpression"
type="connector:simpleTriggerExpressionType"/>
<element name="compoundTriggerExpression"
type="connector:compoundTriggerExpressionType"/>

<group name="triggerExpressionGroup">
  <choice>
    <element ref="connector:simpleTriggerExpression"/>
    <element ref="connector:compoundTriggerExpression"/>
  </choice>
</group>

<complexType name="statementExpressionType" abstract="true"/>

<complexType name="simpleStatementType" abstract="true">
  <complexContent>
    <extension base="connector:statementExpressionType">
      <attribute name="comparator" type="connector:comparatorType"
use="required"/>
    </extension>
  </complexContent>
</complexType>

<complexType name="assessmentStatementType">
  <complexContent>
    <extension base="connector:simpleStatementType">
      <attribute name="firstAssessmentRole" type="string" use="required"/>
      <attribute name="firstRoleQualifier"
type="connector:conditionQualifierType"/>
      <attribute name="firstAssessmentOffset" type="string" use="optional"/>
      <attribute name="secondAssessmentRole" type="string" use="required"/>
      <attribute name="secondRoleQualifier"
type="connector:conditionQualifierType"/>
      <attribute name="secondAssessmentOffset" type="string" use="optional"/>
    </extension>
  </complexContent>
</complexType>

<complexType name="assessmentValueStatementType">
```

```
  <complexContent>
    <extension base="connector:simpleStatementType">
      <attribute name="assessmentRole" type="string" use="required"/>
      <attribute name="roleQualifier"
type="connector:conditionQualifierType"/>
      <attribute name="assessmentOffset" type="string" use="optional"/>
      <attribute name="value" type="anySimpleType" use="required"/>
    </extension>
  </complexContent>
</complexType>

<complexType name="compoundStatementType">
  <complexContent>
    <extension base="connector:statementExpressionType">
        <sequence>
          <group ref="connector:statementExpressionGroup" minOccurs="2"
maxOccurs="unbounded"/>
        </sequence>
      <attribute name="operator" type="connector:logicalOperatorType"
use="required"/>
      <attribute name="isNegated" type="boolean" default="false"/>
    </extension>
  </complexContent>
</complexType>

<element name="assessmentStatement" type="connector:assessmentStatementType"/>
<element name="assessmentValueStatement"
type="connector:assessmentValueStatementType"/>
<element name="compoundStatement" type="connector:compoundStatementType"/>

<group name="statementExpressionGroup">
  <choice>
    <element ref="connector:assessmentStatement"/>
    <element ref="connector:assessmentValueStatement"/>
    <element ref="connector:compoundStatement"/>
  </choice>
</group>

<complexType name="actionExpressionType" abstract="true">
  <attribute name="delay" type="connector:unsignedLongParamUnion"
default="0"/>
</complexType>

<complexType name="simpleActionExpressionType">
  <complexContent>
    <extension base="connector:actionExpressionType">
      <attribute name="actionRole" type="string" use="required"/>
      <attribute name="qualifier" type="connector:actionQualifierType"/>
      <attribute name="repeat" type="connector:repeatUnion" default="0"/>
      <attribute name="repeatDelay" type="connector:unsignedLongParamUnion"
default="0"/>
    </extension>
  </complexContent>
</complexType>

<simpleType name="compoundActionOperatorType">
  <restriction base="string">
    <enumeration value="par" />
    <enumeration value="seq" />
    <enumeration value="excl" />
  </restriction>
</simpleType>

<simpleType name="actionQualifierType">
  <restriction base="string">
    <enumeration value="all" />
    <enumeration value="one" />
  </restriction>
```

```
</simpleType>

<complexType name="compoundActionExpressionType" >
  <complexContent>
    <extension base="connector:actionExpressionType">
        <choice minOccurs="2" maxOccurs="2">
          <group ref="connector:actionExpressionGroup"/>
        </choice>
      <attribute name="operator" type="connector:compoundActionOperatorType"
use="required"/>
    </extension>
  </complexContent>
</complexType>

<element name="simpleActionExpression"
type="connector:simpleActionExpressionType"/>
<element name="compoundActionExpression"
type="connector:compoundActionExpressionType"/>

<group name="actionExpressionGroup">
  <choice>
    <element ref="connector:simpleActionExpression"/>
    <element ref="connector:compoundActionExpression"/>
  </choice>
</group>

<complexType name="connectorBasePrototype">
  <choice minOccurs="0" maxOccurs="unbounded">
    <element ref="connector:causalConnector"/>
    <element ref="connector:constraintConnector"/>
  </choice>
  <attribute name="id" type="ID" use="optional"/>
  <attribute name="description" type="string" use="optional"/>
</complexType>

<element name="causalConnector" type="connector:causalConnectorPrototype"/>
<element name="constraintConnector"
type="connector:constraintConnectorPrototype"/>
<element name="connectorBase" type="connector:connectorBasePrototype"/>

</schema>
```

6.2. The XConnector Profile Specification in XML Schema

XConnector23.xsd

```
<!--
XML Schema for the NCL Language

This is NCL
Copyright: 2000-2005 LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.telemidia.puc-
rio.br/specs/xml/NCL23/profiles/XConnector23.xsd
Author: TeleMidia Laboratory
Revision: 29/06/2005

-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
```

```
        xmlns:connector="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/XConnector"
        xmlns:structure="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/Structure"
        xmlns:profile="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/profiles"
        targetNamespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/profiles"
        elementFormDefault="qualified" attributeFormDefault="unqualified" >

    <!-- import the definitions in the modules namespaces -->

    <import namespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/XConnector"
schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23XConnector.xsd"/>
    <import namespace="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/Structure" schemaLocation="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/modules/NCL23Structure.xsd"/>

    <!-- ========================================================== -->
    <!-- Structure                                                  -->
    <!-- ========================================================== -->
    <!-- extends ncl element -->

    <element name="ncl" substitutionGroup="structure:ncl"/>

    <!-- extends head element -->

    <complexType name="headType">
      <complexContent>
        <extension base="structure:headPrototype">
          <all>
            <element ref="profile:connectorBase" />
          </all>
        </extension>
      </complexContent>
    </complexType>

    <element name="head" type="profile:headType"
substitutionGroup="structure:head"/>


    <!-- ========================================================================= -
->
    <!-- XConnector                                                                -
->
    <!-- ========================================================================= -
->
    <!-- extends causalConnector element -->

    <complexType name="causalConnectorType">
        <complexContent>
            <extension base="connector:causalConnectorPrototype">
            </extension>
        </complexContent>
    </complexType>

    <element name="causalConnector" type="profile:causalConnectorType"
substitutionGroup="connector:causalConnector"/>

    <!-- extends constraintConnector element -->

    <complexType name="constraintConnectorType">
        <complexContent>
            <extension base="connector:constraintConnectorPrototype">
            </extension>
        </complexContent>
```

```
    </complexType>

    <element name="constraintConnector" type="profile:constraintConnectorType"
substitutionGroup="connector:constraintConnector"/>

    <!-- extends connectorBase element -->

    <complexType name="connectorBaseType">
        <complexContent>
            <extension base="connector:connectorBasePrototype">
            </extension>
        </complexContent>
    </complexType>

<element name="connectorBase" type="profile:connectorBaseType"
substitutionGroup="connector:connectorBase"/>

<element name="connectorParam" substitutionGroup="connector:connectorParam"/>

    <element name="conditionRole"
substitutionGroup="connector:conditionRole"/>
    <element name="assessmentRole"
substitutionGroup="connector:assessmentRole"/>
    <element name="actionRole" substitutionGroup="connector:actionRole"/>

    <element name="compoundCondition"
substitutionGroup="connector:compoundCondition"/>

    <element name="eventStateTransitionCondition"
substitutionGroup="connector:eventStateTransitionCondition"/>
    <element name="attributeCondition"
substitutionGroup="connector:attributeCondition"/>
    <element name="eventStateTransitionAssessment"
substitutionGroup="connector:eventStateTransitionAssessment"/>
    <element name="attributeAssessment"
substitutionGroup="connector:attributeAssessment"/>

    <element name="presentationAction"
substitutionGroup="connector:presentationAction"/>
    <element name="assignmentAction"
substitutionGroup="connector:assignmentAction"/>

    <element name="causalGlue" substitutionGroup="connector:causalGlue"/>
    <element name="constraintGlue"
substitutionGroup="connector:constraintGlue"/>

    <element name="simpleTriggerExpression"
substitutionGroup="connector:simpleTriggerExpression"/>
    <element name="compoundTriggerExpression"
substitutionGroup="connector:compoundTriggerExpression"/>
    <element name="simpleActionExpression"
substitutionGroup="connector:simpleActionExpression"/>
    <element name="compoundActionExpression"
substitutionGroup="connector:compoundActionExpression"/>
    <element name="assessmentStatement"
substitutionGroup="connector:assessmentStatement"/>
    <element name="assessmentValueStatement"
substitutionGroup="connector:assessmentValueStatement"/>
    <element name="compoundStatement"
substitutionGroup="connector:compoundStatement"/>

</schema>
```

7. Final Remarks

In order to offer a scalable hypermedia model, with characteristics that can be progressively incorporated in hypermedia system implementations, NCM was divided in several parts. This technical report deals with the declarative language for authoring documents based on NCM, which comprises Part 6: NCL (Nested Context Language) Main Profile.

NCL is a very powerful authoring language that comes with a substantial learning curve. However, one of the most requested features of any authoring system is simplicity. In this sense, many users become overwhelmed with the capabilities provided within NCL. The use of pre built templates (see [SoRC05]) and connectors, made by language experts, can render this task less difficult. Providing a comprehensive visual interface to control the presentation workflow creation process and exposing all of NCL's functionality is a goal that should be followed.

Acknowledgements

Many people have contributed to the NCL definition. Chief among these are Meire Juliana Antonacci, Marcel Stanley, Débora Muchaluat, Heron Vilela de Oliveira e Silva and Romualdo Rezende Costa, who have worked on the language for nearly six years.

References

[Alle83] Allen J.F. Maintaining Knowledge about Temporal Intervals. Communications of the ACM, 26(11), November 1983, pp. 832-843.

[Anto00] Antonacci M.J. NCL: Uma Linguagem Declarativa para Especificação de Documentos Hipermídia com Sincronização Temporal e Espacial. **Master Dissertation, Departamento de Informática, PUC-Rio**, April 2000.

[AMRS00] Antonacci M.J., Muchaluat-Saade D.C., Rodrigues R.F., Soares L.F.G. NCL: Uma Linguagem Declarativa para Especificação de Documentos Hipermídia na Web, **VI Simpósio Brasileiro de Sistemas Multimídia e Hipermídia - SBMídia2000**, Natal, Rio Grande do Norte, June 2000.

[Cost05] Costa, R.M.R. Integração e Interoperabilidade de Documentos MPEG-4 e NCL. **Master Dissertation, Departamento de Informática, PUC-Rio**, April 2005.

[CSS98] Cascading Style Sheets, level 2 – CSS2 Specification, **W3C Recommendation**, disponível em http://www.w3.org/TR/REC-CSS2/, May 1998.

[ISO04] ISO/IEC International Organisation for Standardisation. 14496-11:2004. Coding of Audio-Visual Objects – Part 11: Scene description and application engine/Amd 4. XMT & MPEG-J extensions. 2004.

[Much03] Muchaluat D.C. Relações em Linguagens de Autoria Hipermídia: Aumentando Reuso e Expressividade. **PhD Thesis, Departamento de Informática**, PUC-Rio, Rio de Janeiro, Brasil, March 2003.

[RDF99] Lassila O., Swick R. Resource Description Framework (RDF) Model and Syntax Specification, **W3C Recommendation**, in http://www.w3.org/TR/REC-rdf-syntax/, February 1999.

[RFC2046] Freed N., Borenstein N. Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types, **RFC 2046, IETF**, in ftp://ftp.isi.edu/in-notes/rfc2046.txt, November 1996.

[RFC2396] Berners-Lee T., Fielding R., Masinter L. Uniform Resource Identifiers (URI): Generic Syntax, **RFC 2046, IETF**, in http://www.ietf.org/rfc/rfc2396.txt, August 1998.

[SCHE01] XML Schema Part 0: Primer, **W3C Recommendation**, in http://www.w3.org/TR/xmlschema-0/, May 2001.

[Silv05] Silva H.V.O. X-SMIL: Aumentando Reuso e Expressividade em Linguagens de Autoria Hipermídia. **Master Dissertation, Departamento de Informática, PUC-Rio**, April 2005.

[SMIL01] Synchronized Multimedia Integration Language (SMIL 2.0), **W3C Recommendation,** in http://www.w3.org/TR/smil20, August 2001.

[SoRo05] Soares L.F.G; Rodrigues R.F. Nested Context Model 3.0: Part 1 – NCM Core, **Technical Report, Departamento de Informática PUC-Rio**, May 2005, ISSN: 0103-9741.

[SoRC05] Soares L.F.G; Rodrigues R.F. Nested Context Model 3.0: Part 5 – NCL Full Profile, **Technical Report, Laboratório TeleMídia PUC-Rio**, June 2005.

[W3C03] Mathematical Markup Language (MathML 2.0), W3C Recommendation, in http://www.w3.org/TR/MathML2, October 2003.

[XHTM01] XHTML™ 1.1 - Module-based XHTML, **W3C Recommendation**, in http://www.w3.org/TR/xhtml11, May 2001.

[XML98] Bray T., Paoli J., Sperberg-McQueen C.M., Maler E. Extensible Markup Language (XML) 1.0 (Second Edition), **W3C Recommendation**, in http://www.w3.org/TR/REC-xml, February 1998.

[XLIN01] XML Linking Language (XLink) Version 1.0, **W3C Recommendation**, in http://www.w3.org/TR/xlink, June 2001.

[XLNS99] W3C - World-Wide Web Consortium. **Namespaces in XML**. W3C Recommendation, in < http://www.w3.org/TR/REC-xml-names>, January 1999.

[XPAT99] XML Path language (XPath) version 1.0. **W3C Recommendation**, in http://www.w3.org/TR/xpath, November 1999.

[XSL01] Extensible Stylesheet Language (XSL) Version 1.0. **W3C Recommendation**, in http://www.w3.org/TR/xsl. October 2001.

[XSLT99] W3C - World-Wide Web Consortium. **XSL Transformations (XSLT) Version 1.0**. W3C Recommendation, in < http://www.w3.org/TR/xslt>, November 1999.

Appendix A: Connector Base for Allen's Relations

```
<?xml version="1.0"?>
<ncl id="Allen_Temporal_Relations"
xmlns="http://www.telemidia.puc-rio.br/specs/xml/NCL23/profiles"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.telemidia.puc-rio.br/specs/xml/NCL23/profiles
http://www.telemidia.puc-rio.br/specs/xml/NCL23/profiles/XConnector.xsd">

<head>
<connectorBase>

<constraintConnector id="meetsConstraint" >
   <assessmentRole id="xEnd" eventType="presentation">
      <eventStateTransitionAssessment transition="stops"/>
   </assessmentRole>
   <assessmentRole id="yBegin" eventType="presentation">
      <eventStateTransitionAssessment transition="starts"/>
   </assessmentRole>
   <constraintGlue>
      <assessmentStatement comparator="eq" mainAssessmentRole="xEnd"
       otherAssessmentRole="yBegin"/>
   </constraintGlue>
</constraintConnector >

<causalConnector id="meetsStart">
   <conditionRole id="onEnd" eventType="presentation">
      <eventStateTransitionCondition transition="stops"/>
   </conditionRole>
   <actionRole id="start" eventType="presentation">
      <presentationAction actionType="start"/>
   </actionRole>
   <causalGlue>
      <simpleTriggerExpression conditionRole="onEnd"/>
      <simpleActionExpression actionRole="start"/>
   </causalGlue>
</causalConnector>

<causalConnector id="meetsStop">
   <conditionRole id="onBegin" eventType="presentation">
      <eventStateTransitionCondition transition="starts"/>
   </conditionRole>
   <actionRole id="stop" eventType="presentation">
      <presentationAction actionType="stop"/>
   </actionRole>
   <causalGlue>
      <simpleTriggerExpression conditionRole="onBegin"/>
      <simpleActionExpression actionRole="stop"/>
   </causalGlue>
</causalConnector>

<constraintConnector id="startsConstraint" >
   <assessmentRole id="xBegin" eventType="presentation">
      <eventStateTransitionAssessment transition="starts"/>
   </assessmentRole>
   <assessmentRole id="yBegin" eventType="presentation">
      <eventStateTransitionAssessment transition="starts"/>
   </assessmentRole>
   <constraintGlue>
      <assessmentStatement comparator="eq" mainAssessmentRole="xBegin"
       otherAssessmentRole="yBegin"/>
   </constraintGlue>
</constraintConnector >

<causalConnector id="starts">
   <conditionRole id="onBegin" eventType="presentation">
```

```
      <eventStateTransitionCondition transition="starts"/>
   </conditionRole>
   <actionRole id="start" eventType="presentation">
      <presentationAction actionType="start"/>
   </actionRole>
   <causalGlue>
      <simpleTriggerExpression conditionRole="onBegin"/>
      <simpleActionExpression actionRole="start"/>
   </causalGlue>
</causalConnector>

<constraintConnector id="finishesConstraint" >
   <assessmentRole id="xEnd" eventType="presentation">
      <eventStateTransitionAssessment transition="stops"/>
   </assessmentRole>
   <assessmentRole id="yEnd" eventType="presentation">
      <eventStateTransitionAssessment transition="stops"/>
   </assessmentRole>
   <constraintGlue>
      <assessmentStatement comparator="eq" mainAssessmentRole="xEnd"
       otherAssessmentRole="yEnd"/>
   </constraintGlue>
</constraintConnector >

<causalConnector id="finishes">
   <conditionRole id="onEnd" eventType="presentation">
      <eventStateTransitionCondition transition="stops"/>
   </conditionRole>
   <actionRole id="stop" eventType="presentation">
      <presentationAction actionType="stop"/>
   </actionRole>
   <causalGlue>
      <simpleTriggerExpression conditionRole="onEnd"/>
      <simpleActionExpression actionRole="stop"/>
   </causalGlue>
</causalConnector>

<causalConnector id="before">
   <param name="delay" type="unsignedLong"/>
   <conditionRole id="afterEnd" eventType="presentation">
      <eventStateTransitionCondition transition="stops"/>
   </conditionRole>
   <actionRole id="start" eventType="presentation">
      <presentationAction actionType="start"/>
   </actionRole>
   <causalGlue>
      <simpleTriggerExpression conditionRole="afterEnd"
       minDelay="$delay" maxDelay="$delay"/>
      <simpleActionExpression actionRole="start"/>
   </causalGlue>
</causalConnector>

<constraintConnector id="beforeConstraint" >
   <assessmentRole id="xEnd" eventType="presentation">
      <eventStateTransitionAssessment transition="stops"/>
   </assessmentRole>
   <assessmentRole id="yBegin" eventType="presentation">
      <eventStateTransitionAssessment transition="starts"/>
   </assessmentRole>
   <constraintGlue>
      <assessmentStatement comparator="lt" mainAssessmentRole="xEnd"
       otherAssessmentRole="yBegin"/>
   </constraintGlue>
</constraintConnector >

<constraintConnector id="during">
   <assessmentRole id="xBegin" eventType="presentation">
      <eventStateTransitionAssessment transition="starts"/>
```

```
   </assessmentRole>
   <assessmentRole id="xEnd" eventType="presentation">
      <eventStateTransitionAssessment transition="stops"/>
   </assessmentRole>
   <assessmentRole id="yBegin" eventType="presentation">
      <eventStateTransitionAssessment transition="starts"/>
   </assessmentRole>
   <assessmentRole id="yEnd" eventType="presentation">
      <eventStateTransitionAssessment transition="stops"/>
   </assessmentRole>
   <constraintGlue>
      <compoundStatement operator="and">
         <assessmentStatement
          mainAssessmentRole="yBegin" otherAssessmentRole="xBegin"
comparator="lt"/>
         <assessmentStatement
          mainAssessmentRole="xEnd" otherAssessmentRole="yEnd"
comparator="lt"/>
      </propertyExpression>
   </constraintGlue>
</constraintConnector>

<constraintConnector id="overlaps">
   <assessmentRole id="xBegin" eventType="presentation">
      <eventStateTransitionAssessment transition="starts"/>
   </assessmentRole>
   <assessmentRole id="xEnd" eventType="presentation">
      <eventStateTransitionAssessment transition="stops"/>
   </assessmentRole>
   <assessmentRole id="yBegin" eventType="presentation">
      <eventStateTransitionAssessment transition="starts"/>
   </assessmentRole>
   <assessmentRole id="yEnd" eventType="presentation">
      <eventStateTransitionAssessment transition="stops"/>
   </assessmentRole>
   <constraintGlue>
      <compoundStatement operator="and">
         <assessmentStatement
          mainAssessmentRole="xBegin" otherAssessmentRole="yBegin"
comparator="lt"/>
         <assessmentStatement
          mainAssessmentRole="xEnd" otherAssessmentRole="yEnd"
comparator="lt"/>
      </propertyExpression>
   </constraintGlue>
</constraintConnector>

<constraintConnector id="equals">
   <assessmentRole id="xBegin" eventType="presentation">
      <eventStateTransitionAssessment transition="starts"/>
   </assessmentRole>
   <assessmentRole id="xEnd" eventType="presentation">
      <eventStateTransitionAssessment transition="stops"/>
   </assessmentRole>
   <assessmentRole id="yBegin" eventType="presentation">
      <eventStateTransitionAssessment transition="starts"/>
   </assessmentRole>
   <assessmentRole id="yEnd" eventType="presentation">
      <eventStateTransitionAssessment transition="stops"/>
   </assessmentRole>
   <constraintGlue>
      <compoundStatement operator="and">
         <assessmentStatement
          mainAssessmentRole="xBegin" otherAssessmentRole="yBegin"
comparator="eq"/>
         <assessmentStatement
          mainAssessmentRole="xEnd" otherAssessmentRole="yEnd"
comparator="eq"/>
```

```
      </propertyExpression>
   </constraintGlue>
</constraintConnector>

</connectorBase>
</head>
</ncl>
```

Appendix B: Another Extensive Connector Base

```
<?xml version="1.0" encoding="UTF-8"?>
<ncl id="connBase" xmlns="http://www.telemidia.puc-
rio.br/specs/xml/NCL23/profiles"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.telemidia.puc-rio.br/specs/xml/NCL22/profiles
http://www.telemidia.puc-rio.br/specs/xml/NCL23/profiles/XConnector.xsd">

<head>
<connectorBase>

<!-- Causal Connectors 1 to 1 -->

<!-- OnBegin -->

<causalConnector id="onBegin1Start1">
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation">
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin"/>
    <simpleActionExpression actionRole="start"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1Stop1">
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin" />
    <simpleActionExpression actionRole="stop"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1Pause1">
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="pause" eventType="presentation">
    <presentationAction actionType="pause"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin" />
    <simpleActionExpression actionRole="pause"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1Resume1">
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="resume" eventType="presentation">
    <presentationAction actionType="resume"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin" />
    <simpleActionExpression actionRole="resume"/>
```

```
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1SetVar1">
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="set" eventType="attribution">
    <assignmentAction value="$var"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin"/>
    <simpleActionExpression actionRole="set"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1Resize1">
  <connectorParam name="bounds" type="xsi:string"/>
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="setBounds" eventType="attribution">
    <assignmentAction value="$bounds"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin"/>
    <simpleActionExpression actionRole="setBounds"/>
  </causalGlue>
</causalConnector>

<!-- OnEnd -->

<causalConnector id="onEnd1Start1">
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation">
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd"/>
    <simpleActionExpression actionRole="start"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1Stop1">
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd" />
    <simpleActionExpression actionRole="stop"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1Pause1">
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="pause" eventType="presentation">
    <presentationAction actionType="pause"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd" />
```

```
    <simpleActionExpression actionRole="pause"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1Resume1">
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="resume" eventType="presentation">
    <presentationAction actionType="resume"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd" />
    <simpleActionExpression actionRole="resume"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1SetVar1">
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="set" eventType="attribution">
    <assignmentAction value="$var"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd"/>
    <simpleActionExpression actionRole="set"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1Resize1">
  <connectorParam name="bounds" type="xsi:string"/>
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="setBounds" eventType="attribution">
    <assignmentAction value="$bounds"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd"/>
    <simpleActionExpression actionRole="setBounds"/>
  </causalGlue>
</causalConnector>


<!-- OnMouseSelection -->

<causalConnector id="onMouseSelection1Start1">
  <conditionRole id="onMouseSelection" eventType="selection">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation">
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onMouseSelection"/>
    <simpleActionExpression actionRole="start"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onMouseSelection1Stop1">
  <conditionRole id="onMouseSelection" eventType="selection">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation">
    <presentationAction actionType="stop"/>
  </actionRole>
```

```
  <causalGlue>
    <simpleTriggerExpression conditionRole="onMouseSelection"/>
    <simpleActionExpression actionRole="stop"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onMouseSelection1Pause1">
  <conditionRole id="onMouseSelection" eventType="selection">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="pause" eventType="presentation">
    <presentationAction actionType="pause"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onMouseSelection"/>
    <simpleActionExpression actionRole="pause"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onMouseSelection1Resume1">
  <conditionRole id="onMouseSelection" eventType="selection">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="resume" eventType="presentation">
    <presentationAction actionType="resume"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onMouseSelection"/>
    <simpleActionExpression actionRole="resume"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onMouseSelection1SetVar1">
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onMouseSelection" eventType="selection">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="set" eventType="attribution">
    <assignmentAction value="$var"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onMouseSelection"/>
    <simpleActionExpression actionRole="set"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onMouseSelection1Resize1">
  <connectorParam name="bounds" type="xsi:string"/>
  <conditionRole id="onMouseSelection" eventType="selection">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="setBounds" eventType="attribution">
    <assignmentAction value="$bounds"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onMouseSelection"/>
    <simpleActionExpression actionRole="setBounds"/>
  </causalGlue>
</causalConnector>


<!-- OnKeySelection -->

<causalConnector id="onKeySelection1Start1">
  <connectorParam name="keyCode" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
```

```
  <actionRole id="start" eventType="presentation">
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onKeySelection"/>
    <simpleActionExpression actionRole="start"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onKeySelection1Stop1">
  <connectorParam name="keyCode" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onKeySelection"/>
    <simpleActionExpression actionRole="stop"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onKeySelection1Pause1">
  <connectorParam name="keyCode" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="pause" eventType="presentation">
    <presentationAction actionType="pause"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onKeySelection"/>
    <simpleActionExpression actionRole="pause"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onKeySelection1Resume1">
  <connectorParam name="keyCode" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="resume" eventType="presentation">
    <presentationAction actionType="resume"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onKeySelection"/>
    <simpleActionExpression actionRole="resume"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onKeySelection1SetVar1">
  <connectorParam name="keyCode" type="xsi:string"/>
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="set" eventType="attribution">
    <assignmentAction value="$var"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onKeySelection"/>
    <simpleActionExpression actionRole="set"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onKeySelection1Resize1">
  <connectorParam name="keyCode" type="xsi:string"/>
```

```
  <connectorParam name="bounds" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="setBounds" eventType="attribution">
    <assignmentAction value="$bounds"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onKeySelection"/>
    <simpleActionExpression actionRole="setBounds"/>
  </causalGlue>
</causalConnector>

<!-- Causal Connectors 1 to N -->

<!-- OnBegin -->

<causalConnector id="onBegin1StartN">
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin" />
    <simpleActionExpression actionRole="start" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1StopN">
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin" />
    <simpleActionExpression actionRole="stop" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1PauseN">
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="pause" eventType="presentation" max="unbounded">
    <presentationAction actionType="pause"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin" />
    <simpleActionExpression actionRole="pause" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1ResumeN">
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="resume" eventType="presentation" max="unbounded">
    <presentationAction actionType="resume"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin" />
    <simpleActionExpression actionRole="resume" qualifier="all"/>
  </causalGlue>
</causalConnector>
```

```
<causalConnector id="onBegin1SetVarN">
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="set" eventType="attribution" max="unbounded">
    <assignmentAction value="$var"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin"/>
    <simpleActionExpression actionRole="set" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1StartNStopN">
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin" />
    <compoundActionExpression operator="seq">
        <simpleActionExpression actionRole="start" qualifier="all"/>
        <simpleActionExpression actionRole="stop" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1StartNPauseN">
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <actionRole id="pause" eventType="presentation" max="unbounded">
    <presentationAction actionType="pause"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin" />
    <compoundActionExpression operator="seq">
        <simpleActionExpression actionRole="start" qualifier="all"/>
        <simpleActionExpression actionRole="pause" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1StartNResumeN">
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <actionRole id="resume" eventType="presentation" max="unbounded">
    <presentationAction actionType="resume"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin" />
    <compoundActionExpression operator="seq">
        <simpleActionExpression actionRole="start" qualifier="all"/>
        <simpleActionExpression actionRole="resume" qualifier="all"/>
```

```
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1StartNSetVarN">
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <actionRole id="set" eventType="attribution" max="unbounded">
    <assignmentAction value="$var"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin"/>
    <compoundActionExpression operator="seq">
         <simpleActionExpression actionRole="start" qualifier="all"/>
         <simpleActionExpression actionRole="set" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1StartNResize">
  <connectorParam name="bounds" type="xsi:string"/>
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <actionRole id="setBounds" eventType="attribution">
    <assignmentAction value="$bounds"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin"/>
    <compoundActionExpression operator="seq">
      <simpleActionExpression actionRole="start" qualifier="all"/>
      <simpleActionExpression actionRole="setBounds"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1StartNDelayStopN">
  <connectorParam name="delay" type="xsi:string"/>
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start" />
  </actionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin"/>
    <compoundActionExpression operator="seq">
      <simpleActionExpression actionRole="start" qualifier="all"/>
      <simpleActionExpression actionRole="stop" qualifier="all" delay="$delay"
/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1StopNStartN">
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
```

```
  </conditionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin" />
    <compoundActionExpression operator="seq">
        <simpleActionExpression actionRole="stop" qualifier="all"/>
        <simpleActionExpression actionRole="start" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1StopNPauseN">
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <actionRole id="pause" eventType="presentation" max="unbounded">
    <presentationAction actionType="pause"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin" />
    <compoundActionExpression operator="seq">
        <simpleActionExpression actionRole="stop" qualifier="all"/>
        <simpleActionExpression actionRole="pause" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1StopNResumeN">
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <actionRole id="resume" eventType="presentation" max="unbounded">
    <presentationAction actionType="resume"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin" />
    <compoundActionExpression operator="seq">
        <simpleActionExpression actionRole="stop" qualifier="all"/>
        <simpleActionExpression actionRole="resume" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1StopNSetVarN">
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <actionRole id="set" eventType="attribution" max="unbounded">
    <assignmentAction value="$var"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin"/>
    <compoundActionExpression operator="seq">
```

```
        <simpleActionExpression actionRole="stop" qualifier="all"/>
        <simpleActionExpression actionRole="set" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1StopNResize">
  <connectorParam name="bounds" type="xsi:string"/>
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <actionRole id="setBounds" eventType="attribution">
    <assignmentAction value="$bounds"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin"/>
    <compoundActionExpression operator="seq">
      <simpleActionExpression actionRole="stop" qualifier="all"/>
      <simpleActionExpression actionRole="setBounds"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1StopNDelayStartN">
  <connectorParam name="delay" type="xsi:string"/>
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start" />
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin"/>
    <compoundActionExpression operator="seq">
      <simpleActionExpression actionRole="stop" qualifier="all"/>
      <simpleActionExpression actionRole="start" qualifier="all"
delay="$delay" />
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1SetVarNDelayStopN">
  <connectorParam name="delay" type="xsi:string"/>
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="set" eventType="attribution" max="unbounded">
    <assignmentAction value="$var"/>
  </actionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin"/>
    <compoundActionExpression operator="seq">
      <simpleActionExpression actionRole="set" qualifier="all"/>
      <simpleActionExpression actionRole="stop" qualifier="all" delay="$delay"
/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>
```

```
<!-- OnEnd -->

<causalConnector id="onEnd1StartN">
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd" />
    <simpleActionExpression actionRole="start" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1StopN">
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd" />
    <simpleActionExpression actionRole="stop" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1PauseN">
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="pause" eventType="presentation" max="unbounded">
    <presentationAction actionType="pause"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd" />
    <simpleActionExpression actionRole="pause" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1ResumeN">
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="resume" eventType="presentation" max="unbounded">
    <presentationAction actionType="resume"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd" />
    <simpleActionExpression actionRole="resume" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1SetVarN">
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="set" eventType="attribution" max="unbounded">
    <assignmentAction value="$var"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd"/>
    <simpleActionExpression actionRole="set" qualifier="all"/>
  </causalGlue>
```

```
</causalConnector>

<causalConnector id="onEnd1StartNStopN">
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd" />
    <compoundActionExpression operator="seq">
        <simpleActionExpression actionRole="start" qualifier="all"/>
        <simpleActionExpression actionRole="stop" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1StartNPauseN">
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <actionRole id="pause" eventType="presentation" max="unbounded">
    <presentationAction actionType="pause"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd" />
    <compoundActionExpression operator="seq">
        <simpleActionExpression actionRole="start" qualifier="all"/>
        <simpleActionExpression actionRole="pause" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1StartNResumeN">
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <actionRole id="resume" eventType="presentation" max="unbounded">
    <presentationAction actionType="resume"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd" />
    <compoundActionExpression operator="seq">
        <simpleActionExpression actionRole="start" qualifier="all"/>
        <simpleActionExpression actionRole="resume" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1StartNSetVarN">
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <actionRole id="set" eventType="attribution" max="unbounded">
```

```
    <assignmentAction value="$var"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd"/>
    <compoundActionExpression operator="seq">
         <simpleActionExpression actionRole="start" qualifier="all"/>
         <simpleActionExpression actionRole="set" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1StartNResize">
  <connectorParam name="bounds" type="xsi:string"/>
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <actionRole id="setBounds" eventType="attribution">
    <assignmentAction value="$bounds"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd"/>
    <compoundActionExpression operator="seq">
      <simpleActionExpression actionRole="start" qualifier="all"/>
      <simpleActionExpression actionRole="setBounds"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1StartNDelayStopN">
  <connectorParam name="delay" type="xsi:string"/>
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start" />
  </actionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd"/>
    <compoundActionExpression operator="seq">
      <simpleActionExpression actionRole="start" qualifier="all"/>
      <simpleActionExpression actionRole="stop" qualifier="all" delay="$delay"
/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1StopNStartN">
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd" />
    <compoundActionExpression operator="seq">
        <simpleActionExpression actionRole="stop" qualifier="all"/>
        <simpleActionExpression actionRole="start" qualifier="all"/>
    </compoundActionExpression>
```

```
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1StopNPauseN">
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <actionRole id="pause" eventType="presentation" max="unbounded">
    <presentationAction actionType="pause"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd" />
    <compoundActionExpression operator="seq">
        <simpleActionExpression actionRole="stop" qualifier="all"/>
        <simpleActionExpression actionRole="pause" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1StopNResumeN">
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <actionRole id="resume" eventType="presentation" max="unbounded">
    <presentationAction actionType="resume"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd" />
    <compoundActionExpression operator="seq">
        <simpleActionExpression actionRole="stop" qualifier="all"/>
        <simpleActionExpression actionRole="resume" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1StopNSetVarN">
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <actionRole id="set" eventType="attribution" max="unbounded">
    <assignmentAction value="$var"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd"/>
    <compoundActionExpression operator="seq">
         <simpleActionExpression actionRole="stop" qualifier="all"/>
         <simpleActionExpression actionRole="set" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1StopNResize">
  <connectorParam name="bounds" type="xsi:string"/>
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
```

```
  </actionRole>
  <actionRole id="setBounds" eventType="attribution">
    <assignmentAction value="$bounds"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd"/>
    <compoundActionExpression operator="seq">
      <simpleActionExpression actionRole="stop" qualifier="all"/>
      <simpleActionExpression actionRole="setBounds"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1StopNDelayStartN">
  <connectorParam name="delay" type="xsi:string"/>
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start" />
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd"/>
    <compoundActionExpression operator="seq">
      <simpleActionExpression actionRole="stop" qualifier="all"/>
      <simpleActionExpression actionRole="start" qualifier="all"
delay="$delay" />
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1DelayStopNStopN">
  <connectorParam name="delay" type="xsi:string"/>
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <actionRole id="delayStop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd" />
    <compoundActionExpression operator="seq">
       <simpleActionExpression actionRole="stop" qualifier="all"/>
       <simpleActionExpression actionRole="delayStop" qualifier="all"
delay="$delay" />
    </compoundActionExpression>
  </causalGlue>
</causalConnector>


<!-- OnMouseSelection -->

<causalConnector id="onMouseSelection1StartN">
  <conditionRole id="onMouseSelection" eventType="selection">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onMouseSelection"/>
    <simpleActionExpression actionRole="start" qualifier="all"/>
```

```
  </causalGlue>
</causalConnector>

<causalConnector id="onMouseSelection1StopN">
  <conditionRole id="onMouseSelection" eventType="selection">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onMouseSelection"/>
    <simpleActionExpression actionRole="stop" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onMouseSelection1PauseN">
  <conditionRole id="onMouseSelection" eventType="selection">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="pause" eventType="presentation" max="unbounded">
    <presentationAction actionType="pause"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onMouseSelection"/>
    <simpleActionExpression actionRole="pause" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onMouseSelection1ResumeN">
  <conditionRole id="onMouseSelection" eventType="selection">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="resume" eventType="presentation" max="unbounded">
    <presentationAction actionType="resume"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onMouseSelection"/>
    <simpleActionExpression actionRole="resume" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onMouseSelection1SetVarN">
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onMouseSelection" eventType="selection">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="set" eventType="attribution" max="unbounded">
    <assignmentAction value="$var"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onMouseSelection"/>
    <simpleActionExpression actionRole="set" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onMouseSelection1StartNStopN">
  <conditionRole id="onMouseSelection" eventType="selection">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onMouseSelection"/>
```

```
    <compoundActionExpression operator="seq">
      <simpleActionExpression actionRole="start" qualifier="all"/>
      <simpleActionExpression actionRole="stop" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onMouseSelection1StartNSetVarN">
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onMouseSelection" eventType="selection">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="set" eventType="attribution" max="unbounded">
    <assignmentAction value="$var"/>
  </actionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onMouseSelection"/>
    <compoundActionExpression operator="seq">
        <simpleActionExpression actionRole="start" qualifier="all"/>
        <simpleActionExpression actionRole="set" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onMouseSelection1StopNSetVarN">
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onMouseSelection" eventType="selection">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <actionRole id="set" eventType="attribution" max="unbounded">
    <assignmentAction value="$var"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onMouseSelection"/>
    <compoundActionExpression operator="seq">
        <simpleActionExpression actionRole="set" qualifier="all"/>
        <simpleActionExpression actionRole="stop" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onMouseSelection1SetVar1StopNStartN">
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onMouseSelection" eventType="selection">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="set" eventType="attribution">
    <assignmentAction value="$var"/>
  </actionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onMouseSelection"/>
    <compoundActionExpression operator="seq">
        <simpleActionExpression actionRole="set"/>
        <simpleActionExpression actionRole="stop" qualifier="all"/>
      <simpleActionExpression actionRole="start" qualifier="all"/>
    </compoundActionExpression>
```

```
  </causalGlue>
</causalConnector>

<causalConnector id="onMouseSelectionSetVarResizeStartStop">
  <connectorParam name="bounds" type="xsi:string"/>
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onMouseSelection" eventType="selection">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="set" eventType="attribution">
    <assignmentAction value="$var"/>
  </actionRole>
  <actionRole id="setBounds" eventType="attribution">
    <assignmentAction value="$bounds"/>
  </actionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onMouseSelection"/>
    <compoundActionExpression operator="seq">
      <simpleActionExpression actionRole="set"/>
      <simpleActionExpression actionRole="setBounds"/>
      <simpleActionExpression actionRole="start" qualifier="all"/>
      <simpleActionExpression actionRole="stop" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>


<!-- OnKeySelection -->

<causalConnector id="onKeySelection1StartN">
  <connectorParam name="keyCode" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onKeySelection"/>
    <simpleActionExpression actionRole="start" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onKeySelection1StopN">
  <connectorParam name="keyCode" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onKeySelection"/>
    <simpleActionExpression actionRole="stop" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onKeySelection1PauseN">
  <connectorParam name="keyCode" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
```

```
  <actionRole id="pause" eventType="presentation" max="unbounded">
    <presentationAction actionType="pause"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onKeySelection"/>
    <simpleActionExpression actionRole="pause" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onKeySelection1ResumeN">
  <connectorParam name="keyCode" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="resume" eventType="presentation" max="unbounded">
    <presentationAction actionType="resume"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onKeySelection"/>
    <simpleActionExpression actionRole="resume" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onKeySelection1SetVarN">
  <connectorParam name="keyCode" type="xsi:string"/>
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="set" eventType="attribution" max="unbounded">
    <assignmentAction value="$var"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onKeySelection"/>
    <simpleActionExpression actionRole="set" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onKeySelection1StartNStopN">
  <connectorParam name="keyCode" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onKeySelection"/>
    <compoundActionExpression operator="seq">
      <simpleActionExpression actionRole="start" qualifier="all"/>
      <simpleActionExpression actionRole="stop" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onKeySelection1StartNSetVarN">
  <connectorParam name="keyCode" type="xsi:string"/>
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="set" eventType="attribution" max="unbounded">
    <assignmentAction value="$var"/>
  </actionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
```

```
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onKeySelection"/>
    <compoundActionExpression operator="seq">
        <simpleActionExpression actionRole="start" qualifier="all"/>
        <simpleActionExpression actionRole="set" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onKeySelection1StopNSetVarN">
  <connectorParam name="keyCode" type="xsi:string"/>
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <actionRole id="set" eventType="attribution" max="unbounded">
    <assignmentAction value="$var"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onKeySelection"/>
    <compoundActionExpression operator="seq">
        <simpleActionExpression actionRole="set" qualifier="all"/>
        <simpleActionExpression actionRole="stop" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onKeySelection1SetVar1StopNStartN">
  <connectorParam name="keyCode" type="xsi:string"/>
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="set" eventType="attribution">
    <assignmentAction value="$var"/>
  </actionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onKeySelection"/>
    <compoundActionExpression operator="seq">
        <simpleActionExpression actionRole="set"/>
        <simpleActionExpression actionRole="stop" qualifier="all"/>
      <simpleActionExpression actionRole="start" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onKeySelectionSetVarResizeStartStop">
  <connectorParam name="keyCode" type="xsi:string"/>
  <connectorParam name="bounds" type="xsi:string"/>
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="set" eventType="attribution">
    <assignmentAction value="$var"/>
  </actionRole>
  <actionRole id="setBounds" eventType="attribution">
```

```
    <assignmentAction value="$bounds"/>
  </actionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onKeySelection"/>
    <compoundActionExpression operator="seq">
      <simpleActionExpression actionRole="set"/>
      <simpleActionExpression actionRole="setBounds"/>
      <simpleActionExpression actionRole="start" qualifier="all"/>
      <simpleActionExpression actionRole="stop" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onKeySelection1Stop1ResizePause1Start1">
  <connectorParam name="keyCode" type="xsi:string"/>
  <connectorParam name="bounds" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation">
    <presentationAction actionType="stop"/>
  </actionRole>
  <actionRole id="setBounds" eventType="attribution">
    <assignmentAction value="$bounds"/>
  </actionRole>
  <actionRole id="pause" eventType="presentation" >
    <presentationAction actionType="pause"/>
  </actionRole>
  <actionRole id="start" eventType="presentation" >
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onKeySelection"/>
    <compoundActionExpression operator="seq">
      <simpleActionExpression actionRole="stop" />
      <simpleActionExpression actionRole="setBounds"/>
      <simpleActionExpression actionRole="pause" />
      <simpleActionExpression actionRole="start" />
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1ResizeResume1">
  <connectorParam name="bounds" type="xsi:string"/>
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <actionRole id="setBounds" eventType="attribution">
    <assignmentAction value="$bounds"/>
  </actionRole>
  <actionRole id="resume" eventType="presentation" >
    <presentationAction actionType="resume"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onEnd"/>
    <compoundActionExpression operator="seq">
      <simpleActionExpression actionRole="setBounds"/>
      <simpleActionExpression actionRole="resume" />
    </compoundActionExpression>
  </causalGlue>
</causalConnector>
```

```
<causalConnector id="onBegin1SetVar1Start1">
  <connectorParam name="var" type="xsi:string"/>
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="set" eventType="attribution">
    <assignmentAction value="$var"/>
  </actionRole>
  <actionRole id="start" eventType="presentation">
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onBegin"/>
    <compoundActionExpression operator="seq">
      <simpleActionExpression actionRole="start"/>
      <simpleActionExpression actionRole="set"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<!-- Causal Connectors N to 1 -->

<causalConnector id="onOrKeySelectionNResize1">
  <connectorParam name="keyCode" type="xsi:string"/>
  <connectorParam name="bounds" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode"
max="unbounded">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="setBounds" eventType="attribution">
    <assignmentAction value="$bounds"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onKeySelection" qualifier="any"/>
    <simpleActionExpression actionRole="setBounds"/>
  </causalGlue>
</causalConnector>

<!-- Causal Connectors N to N -->

<causalConnector id="onKeySelectionNStartN">
  <connectorParam name="keyCode" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode"
max="unbounded">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
     <simpleTriggerExpression conditionRole="onKeySelection" qualifier="any"/>
     <simpleActionExpression actionRole="start" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onKeySelectionNStopN">
  <connectorParam name="keyCode" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode"
max="unbounded">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
     <simpleTriggerExpression conditionRole="onKeySelection" qualifier="any"/>
     <simpleActionExpression actionRole="stop" qualifier="all"/>
  </causalGlue>
```

```
</causalConnector>

<causalConnector id="onKeySelectionNPauseN">
  <connectorParam name="keyCode" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode"
max="unbounded">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="pause" eventType="presentation" max="unbounded">
    <presentationAction actionType="pause"/>
  </actionRole>
  <causalGlue>
     <simpleTriggerExpression conditionRole="onKeySelection" qualifier="any"/>
     <simpleActionExpression actionRole="pause" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onKeySelectionNResumeN">
  <connectorParam name="keyCode" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode"
max="unbounded">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="resume" eventType="presentation" max="unbounded">
    <presentationAction actionType="resume"/>
  </actionRole>
  <causalGlue>
     <simpleTriggerExpression conditionRole="onKeySelection" qualifier="any"/>
     <simpleActionExpression actionRole="resume" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onKeySelectionNStartNStopN">
  <connectorParam name="keyCode" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode"
max="unbounded">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
     <simpleTriggerExpression conditionRole="onKeySelection" qualifier="any"/>
     <compoundActionExpression operator="seq">
        <simpleActionExpression actionRole="start" qualifier="all"/>
        <simpleActionExpression actionRole="stop" qualifier="all"/>
     </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onKeySelectionStopStart">
  <connectorParam name="keyCode" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode"
max="unbounded">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
    <simpleTriggerExpression conditionRole="onKeySelection" qualifier="any"/>
    <compoundActionExpression operator="seq">
```

```
        <simpleActionExpression actionRole="stop" qualifier="all"/>
        <simpleActionExpression actionRole="start" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1AttNodeTest1StartN">
<connectorParam name="value" type="xsi:string"/>
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <assessmentRole id="attNodeTest" eventType="attribution">
    <attributeAssessment attributeType="nodeAttribute"/>
  </assessmentRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
    <compoundTriggerExpression operator="and">
      <simpleTriggerExpression conditionRole="onBegin"/>
      <assessmentValueStatement assessmentRole="attNodeTest" comparator="eq"
value="$value"/>
    </compoundTriggerExpression>
    <simpleActionExpression actionRole="start" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1AttNodeTest1StopN">
<connectorParam name="value" type="xsi:string"/>
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <assessmentRole id="attNodeTest" eventType="attribution">
    <attributeAssessment attributeType="nodeAttribute"/>
  </assessmentRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
    <compoundTriggerExpression operator="and">
      <simpleTriggerExpression conditionRole="onBegin"/>
      <assessmentValueStatement assessmentRole="attNodeTest" comparator="eq"
value="$value"/>
    </compoundTriggerExpression>
    <simpleActionExpression actionRole="stop" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1AttNodeTest1StartN">
<connectorParam name="value" type="xsi:string"/>
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <assessmentRole id="attNodeTest" eventType="attribution">
    <attributeAssessment attributeType="nodeAttribute"/>
  </assessmentRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
    <compoundTriggerExpression operator="and">
      <simpleTriggerExpression conditionRole="onEnd"/>
      <assessmentValueStatement assessmentRole="attNodeTest" comparator="eq"
value="$value"/>
    </compoundTriggerExpression>
    <simpleActionExpression actionRole="start" qualifier="all"/>
  </causalGlue>
</causalConnector>
```

```
<causalConnector id="onEnd1AttNodeTest1StopN">
<connectorParam name="value" type="xsi:string"/>
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <assessmentRole id="attNodeTest" eventType="attribution">
    <attributeAssessment attributeType="nodeAttribute"/>
  </assessmentRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
    <compoundTriggerExpression operator="and">
      <simpleTriggerExpression conditionRole="onEnd"/>
      <assessmentValueStatement assessmentRole="attNodeTest" comparator="eq"
value="$value"/>
    </compoundTriggerExpression>
    <simpleActionExpression actionRole="stop" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onKeySelectionNAttNodeTest1StartN">
  <connectorParam name="keyCode" type="xsi:string"/>
  <connectorParam name="value" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode"
max="unbounded">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <assessmentRole id="attNodeTest" eventType="attribution">
    <attributeAssessment attributeType="nodeAttribute"/>
  </assessmentRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
     <compoundTriggerExpression operator="and">
        <simpleTriggerExpression conditionRole="onKeySelection"
qualifier="any"/>
        <assessmentValueStatement assessmentRole="attNodeTest" comparator="eq"
value="$value"/>
     </compoundTriggerExpression>
     <simpleActionExpression actionRole="start" qualifier="all"/>
  </causalGlue>
</causalConnector>

<causalConnector id="onKeySelectionNAttNodeTest1StopN">
  <connectorParam name="keyCode" type="xsi:string"/>
  <connectorParam name="value" type="xsi:string"/>
  <conditionRole id="onKeySelection" eventType="selection" key="$keyCode"
max="unbounded">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <assessmentRole id="attNodeTest" eventType="attribution">
    <attributeAssessment attributeType="nodeAttribute"/>
  </assessmentRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
     <compoundTriggerExpression operator="and">
        <simpleTriggerExpression conditionRole="onKeySelection"
qualifier="any"/>
        <assessmentValueStatement assessmentRole="attNodeTest" comparator="eq"
value="$value"/>
     </compoundTriggerExpression>
     <simpleActionExpression actionRole="stop" qualifier="all"/>
  </causalGlue>
```

```
</causalConnector>

<causalConnector id="onBegin1AttNodeTest1StartNStopN">
<connectorParam name="value" type="xsi:string"/>
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <assessmentRole id="attNodeTest" eventType="attribution">
    <attributeAssessment attributeType="nodeAttribute"/>
  </assessmentRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
    <compoundTriggerExpression operator="and">
      <simpleTriggerExpression conditionRole="onBegin"/>
      <assessmentValueStatement assessmentRole="attNodeTest" comparator="eq"
value="$value"/>
    </compoundTriggerExpression>
    <compoundActionExpression operator="seq">
        <simpleActionExpression actionRole="stop" qualifier="all"/>
        <simpleActionExpression actionRole="start" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1AttNodeTest1StartNStopN">
<connectorParam name="value" type="xsi:string"/>
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <assessmentRole id="attNodeTest" eventType="attribution">
    <attributeAssessment attributeType="nodeAttribute"/>
  </assessmentRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
    <compoundTriggerExpression operator="and">
      <simpleTriggerExpression conditionRole="onEnd"/>
      <assessmentValueStatement assessmentRole="attNodeTest" comparator="eq"
value="$value"/>
    </compoundTriggerExpression>
    <compoundActionExpression operator="seq">
        <simpleActionExpression actionRole="stop" qualifier="all"/>
        <simpleActionExpression actionRole="start" qualifier="all"/>
    </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onEnd1AttNodeTest1Resize1StopN">
  <connectorParam name="bounds" type="xsi:string"/>
  <connectorParam name="value" type="xsi:string"/>
  <conditionRole id="onEnd" eventType="presentation">
    <eventStateTransitionCondition transition="stops"/>
  </conditionRole>
  <assessmentRole id="attNodeTest" eventType="attribution">
    <attributeAssessment attributeType="nodeAttribute"/>
  </assessmentRole>
  <actionRole id="setBounds" eventType="attribution">
    <assignmentAction value="$bounds"/>
  </actionRole>
```

```
  <actionRole id="stop" eventType="presentation" max="unbounded">
    <presentationAction actionType="stop"/>
  </actionRole>
  <causalGlue>
     <compoundTriggerExpression operator="and">
        <simpleTriggerExpression conditionRole="onEnd"/>
        <assessmentValueStatement assessmentRole="attNodeTest" comparator="eq"
value="$value"/>
     </compoundTriggerExpression>
     <compoundActionExpression operator="seq">
       <simpleActionExpression actionRole="setBounds"/>
       <simpleActionExpression actionRole="stop" qualifier="all"/>
     </compoundActionExpression>
  </causalGlue>
</causalConnector>

<causalConnector id="onBegin1AttNodeTest1Resize1StartN">
  <connectorParam name="bounds" type="xsi:string"/>
  <connectorParam name="value" type="xsi:string"/>
  <conditionRole id="onBegin" eventType="presentation">
    <eventStateTransitionCondition transition="starts"/>
  </conditionRole>
  <assessmentRole id="attNodeTest" eventType="attribution">
    <attributeAssessment attributeType="nodeAttribute"/>
  </assessmentRole>
  <actionRole id="setBounds" eventType="attribution">
    <assignmentAction value="$bounds"/>
  </actionRole>
  <actionRole id="start" eventType="presentation" max="unbounded">
    <presentationAction actionType="start"/>
  </actionRole>
  <causalGlue>
     <compoundTriggerExpression operator="and">
        <simpleTriggerExpression conditionRole="onBegin"/>
        <assessmentValueStatement assessmentRole="attNodeTest" comparator="eq"
value="$value"/>
     </compoundTriggerExpression>
     <compoundActionExpression operator="seq">
       <simpleActionExpression actionRole="setBounds"/>
       <simpleActionExpression actionRole="start" qualifier="all"/>
     </compoundActionExpression>
  </causalGlue>
</causalConnector>



<!-- Constraint Conectores 1 to 1 / comparator "eq" -->


<constraintConnector id="trStopsTest1trStartsTest1" >
   <assessmentRole id="trStopsTest" eventType="presentation">
      <eventStateTransitionAssessment transition="stops"/>
   </assessmentRole>
   <assessmentRole id="trStartsTest" eventType="presentation">
      <eventStateTransitionAssessment transition="starts"/>
   </assessmentRole>
   <constraintGlue>
      <assessmentStatement comparator="eq" firstAssessmentRole="trStopsTest"
       secondAssessmentRole="trStartsTest"/>
   </constraintGlue>
</constraintConnector >

<constraintConnector id="trStarts1Test1trStarts2Test1" >
   <assessmentRole id="trStarts1Test" eventType="presentation">
      <eventStateTransitionAssessment transition="starts"/>
   </assessmentRole>
   <assessmentRole id="trStarts2Test" eventType="presentation">
      <eventStateTransitionAssessment transition="starts"/>
```

```
   </assessmentRole>
   <constraintGlue>
      <assessmentStatement comparator="eq" firstAssessmentRole="trStarts1Test"
       secondAssessmentRole="trStarts2Test"/>
   </constraintGlue>
</constraintConnector >

<constraintConnector id="trStops1Test1trStops2Test1" >
   <assessmentRole id="trStops1Test" eventType="presentation">
      <eventStateTransitionAssessment transition="stops"/>
   </assessmentRole>
   <assessmentRole id="trStops2Test" eventType="presentation">
      <eventStateTransitionAssessment transition="stops"/>
   </assessmentRole>
   <constraintGlue>
      <assessmentStatement comparator="eq" firstAssessmentRole="trStops1Test"
       secondAssessmentRole="trStops2Test"/>
   </constraintGlue>
</constraintConnector >

<constraintConnector
id="trStarts1Test1trStarts2Test1_AND_trStops1Test1trStops2Test1">
   <assessmentRole id="trStarts1Test" eventType="presentation">
      <eventStateTransitionAssessment transition="starts"/>
   </assessmentRole>
   <assessmentRole id="trStops1Test" eventType="presentation">
      <eventStateTransitionAssessment transition="stops"/>
   </assessmentRole>
   <assessmentRole id="trStarts2Test" eventType="presentation">
      <eventStateTransitionAssessment transition="starts"/>
   </assessmentRole>
   <assessmentRole id="trStops2Test" eventType="presentation">
      <eventStateTransitionAssessment transition="stops"/>
   </assessmentRole>
   <constraintGlue>
      <compoundStatement operator="and">
         <assessmentStatement
          firstAssessmentRole="trStarts1Test"
secondAssessmentRole="trStarts2Test" comparator="eq"/>
         <assessmentStatement
         firstAssessmentRole="trStops1Test"
secondAssessmentRole="trStops2Test" comparator="eq"/>
      </compoundStatement>
   </constraintGlue>
</constraintConnector>


<!-- Constraint Conectores 1 to 1 / comparator different from "eq" -->

<constraintConnector id="trStopsTestLTtrStartsTest" >
   <assessmentRole id="trStopsTest" eventType="presentation">
      <eventStateTransitionAssessment transition="stops"/>
   </assessmentRole>
   <assessmentRole id="trStartsTest" eventType="presentation">
      <eventStateTransitionAssessment transition="starts"/>
   </assessmentRole>
   <constraintGlue>
      <assessmentStatement comparator="lt" firstAssessmentRole="trStopsTest"
       secondAssessmentRole="trStartsTest"/>
   </constraintGlue>
</constraintConnector >

</connectorBase>

</head>
</ncl>
```