



# PUC

ISSN 0103-9741

Monografias em Ciência da Computação  
n° 03/09

**Nested Context Language 3.0**  
**Part 12 – Support to Multiple Exhibition Devices**

**Luiz Fernando Gomes Soares**

Departamento de Informática

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO**

**RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900**

**RIO DE JANEIRO - BRASIL**

## Nested Context Language 3.0

### Part 12 – Support to Multiple Exhibition Devices

**Luiz Fernando Gomes Soares**

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

lfgs@inf.puc-rio.br

**Abstract.** *This technical report describes the support offered by NCL 3.0 to multiple exhibition devices. NCL (Nested Context Language) is an XML application language based on the NCM (Nested Context Model) conceptual model for hypermedia document specification, with temporal and spatial synchronization among its media objects.*

**Keywords:** *multiple devices, digital TV; middleware; declarative environment; NCL.*

**Resumo.** *Este relatório técnico descreve o suporte a múltiplos dispositivos de exibição oferecido por NCL. NCL é uma aplicação XML baseada no modelo conceitual NCM (Nested Context Model) para a especificação de documentos hipermídia com sincronismo espacial e temporal entre seus objetos.*

**Palavras chave:** *múltiplos dispositivos de exibição; TV digital; middleware; linguagem declarativa; NCL.*



## **Nested Context Language 3.0**

### **Part 12 – Support to Multiple Exhibition Devices**

© Laboratório TeleMídia da PUC-Rio – Todos os direitos reservados

Impresso no Brasil

As informações contidas neste documento são de propriedade do Laboratório TeleMídia (PUC-Rio), sendo proibida a sua divulgação, reprodução ou armazenamento em base de dados ou sistema de recuperação sem permissão prévia e por escrito do Laboratório TeleMídia (PUC-Rio). As informações estão sujeitas a alterações sem notificação prévia.

Os nomes de produtos, serviços ou tecnologias eventualmente mencionadas neste documento são marcas registradas dos respectivos detentores.

Figuras apresentadas, quando obtidas de outros documentos, são sempre referenciadas e são de propriedade dos respectivos autores ou editoras referenciados.

Fazer cópias de qualquer parte deste documento para qualquer finalidade, além do uso pessoal, constitui violação das leis internacionais de direitos autorais.

**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.....	5
2. NCL Historical Evolution.....	6
3. Overview of NCL Elements .....	9
4. NCL Model for Multiple Device Exhibition .....	12
4.1. Types of Classes and Class Registration .....	12
4.2. Control of Input Units.....	13
5. Behavior of Exhibition Devices.....	14
5.1. Device Behavior in a Passive Class.....	14
5.2. Device Behavior in a Active Class .....	18
5.3. Device Registered in Passive and Active Classes .....	22
6. Final Remarks .....	23
References.....	26

# Nested Context Language 3.0

## Part 12 – Support to Multiple Exhibition Devices

Luiz Fernando Gomes Soares

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

lfgs@inf.puc-rio.br

***Abstract.** This technical report describes the support offered by NCL 3.0 to multiple exhibition devices. NCL (Nested Context Language) is an XML application language based on the NCM (Nested Context Model) conceptual model for hypermedia document specification, with temporal and spatial synchronization among its media objects.*

### 1. Introduction

An NCL document can be presented in multiple exhibition devices at the same time. These devices can be registered in classes of two different types: those whose members are able to run media players (including imperative and declarative object players) and those called passive classes, whose members are not required to run media players.

This report describes how an NCL document specifies where its media objects will be played and the behavior of the NCL formatter and media players when presenting this NCL document. The report is organized as follows. Section 2 gives an historical evolution of the NCL versions. Section 3 presents a brief overview of the NCL 3.0 elements. Section 4 introduces the NCL hierarchical control model for multiple device support. Section 5 discusses the behavior of devices depending on the classes they are registered; several NCL examples are also given in this section. Section 6 is reserved for final remarks.

## 2. NCL Historical Evolution

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]. 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.

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

- definition of hypermedia connectors and connector bases;
- use of hypermedia connectors for link authoring;
- definition of ports and maps for composite nodes, satisfying the document compositionality 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 <descriptorSwitch> elements 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.

NCL 2.3 introduced two new modules for supporting base and entity reuse, and refined the definition of some elements in order to support the new features.

NCL 2.4 reviewed and refined the reuse support introduced in version 2.3, and the specification of the switch and descriptor switch elements. This version also split the Timing module introduced by NCL 2.1, creating a new module to encapsulate issues related with time-scaling operations (elastic time computation using temporal cost functions) in hypermedia documents.

The NCL 3.0 edition revised some functionalities contained in NCL 2.4. NCL 3.0 is more specific regarding some attribute values. This new version introduced two new functionalities, as well: Key Navigation and Animation functionalities. In addition, NCL 3.0 made depth modifications on the Composite-Node Template functionality and introduces some SMIL based modules to NCL profiles for transition effects in media presentation and for metadata definition. NCL 3.0 also reviewed the hypermedia connector specification in order to have a more concise notation. Relationships among imperative and

declarative objects and other objects in an NCL document are also refined in NCL 3.0, as well as the behavior of imperative and declarative object players. Finally, NCL 3.0 also refined the support to multiple exhibition devices and introduced the support to NCL live editing commands.

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

The Nested Context Model is composed of Parts 1, 2, 3, and 4 of the collection:

- Part 1 – NCM Core  
concerned with the main model entities, which should be present in all NCM implementations<sup>1</sup>.
- 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 NCL (Nested Context Language) specification is composed of Parts 5 to 12 of the collection:

- Part 5 – NCL (Nested Context Language) Full Profile  
concerned with the definition of an XML application language for authoring and exchanging NCM-based documents, using all NCL modules, including those for the definition and use of templates, and also the definition of constraint connectors, composite-connectors, temporal cost functions, transition effects and metainformation characterization.
- Part 6 – NCL (Nested Context Language) XConnector Profile Family  
concerned with the definition of an XML application language for authoring connector bases. One profile is defined for authoring causal connectors, another one for authoring causal and constraint connectors, and a third one for authoring both simple and composite connectors.
- Part 7 – Composite Node Templates  
concerned with the definition of the NCL Composite-Node Template functionality, and with the definition of an XML application language (XTemplate) for authoring template bases.

---

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

- Part 8 – NCL (Nested Context Language) Digital TV Profiles  
concerned with the definition of an XML application language for authoring documents aiming at the digital TV domain. Two profiles are defined: the Enhanced Digital TV (EDTV) profile and the Basic Digital TV (BDTV) profile.
- Part 9 – NCL Live Editing Commands  
concerned with editing commands used for live authoring applications based on NCL.
- Part 10 – Imperative Objects in NCL: The NCLua Scripting Language  
concerned with the definition of objects that contain imperative code and how these objects may be related with other objects in NCL applications.
- Part 11 – Declarative Objects in NCL: Nesting Objects with NCL Code in NCL Documents  
concerned with the definition of objects that contain declarative code (including nested objects with NCL code) and how these objects may be related with other objects in NCL applications.
- Part 12 – Support to Multiple Exhibition Devices (this document)  
concerned with the use of multiple devices for simultaneously presenting an NCL document.

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

### 3. Overview of NCL Elements

NCL is an XML application that follows the modularization approach. The modularization approach has been used in several W3C language recommendations. A *module* is a collection of semantically-related XML elements, attributes, and attribute's values that represents a unit of functionality. Modules are defined in coherent sets. A *language profile* is a combination of modules. Several NCL profiles have been defined, among them those defined by Parts 5, 6, 7, and 8 of the NCL collection presented in Section 2. Of special interest are the profiles defined for Digital TV, the EDTVProfile (*Enhanced Digital TV Profile*) and the BDTVProfile (*Basic Digital TV Profile*). This section briefly describes the elements that compose these profiles. The complete definition of the NCL 3.0 modules for these profiles, using XML Schemas, is presented in [SoRo06]. Any ambiguity found in this text can be clarified by consulting the XML Schemas.

The basic NCL structure module defines the root element, called `<ncl>`, and its children elements, the `<head>` element and the `<body>` element, following the terminology adopted by other W3C standards.

The `<head>` element may have `<importedDocumentBase>`, `<ruleBase>`, `<transitionBase>`, `<regionBase>`, `<descriptorBase>`, `<connectorBase>`, `<meta>`, and `<metadata>` elements as its children.

The `<body>` element may have `<port>`, `<property>`, `<media>`, `<context>`, `<switch>`, and `<link>` elements as its children. The `<body>` element is treated as an NCM context node. In NCM [SoRo05], the conceptual data model of NCL, a node may be a context, a switch or a media object. Context nodes may contain other NCM nodes and links. Switch nodes contain other NCM nodes. NCM nodes are represented by corresponding NCL elements.

The `<media>` element defines a media object specifying its type and its content location. NCL only defines how media objects are structured and related, in time and space. As a glue language, it does not restrict or prescribe the media-object content types. However, some types are defined by the language. For example: the “application/x-ncl-settings” type, specifying an object whose properties are global variables defined by the document author or are reserved environment variables that may be manipulated by the NCL document processing; and the “application/x-ncl-time” type, specifying a special `<media>` element whose content is the Greenwich Mean Time (GMT).

The `<context>` element 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 [SoRo05]. Like the `<body>` element, a `<context>` element may have `<port>`, `<property>`, `<media>`, `<context>`, `<switch>`, and `<link>` elements as its children.

The `<switch>` element allows the definition of alternative document nodes (represented by `<media>`, `<context>`, and `<switch>` elements) to be chosen during presentation time. Test rules used in choosing the switch component to be presented are defined by `<rule>` or `<compositeRule>` elements that are grouped by the `<ruleBase>` element, defined as a child element of the `<head>` element.

The NCL Interfaces functionality allows the definition of node interfaces that are used in relationships with other node interfaces. The `<area>` element allows the definition of content anchors representing spatial portions, temporal portions, or temporal and spatial portions of a media object (`<media>` element) content. The `<port>` element specifies a composite node (`<context>`, `<body>` or `<switch>` element) port with its respective mapping to an interface of one of its child components. The `<property>` element is used for defining a node property or a group of node properties as one of the node's interfaces. The `<switchPort>` element allows the creation of `<switch>` element interfaces that are mapped to a set of alternative interfaces of the switch's internal nodes.

The `<descriptor>` element specifies temporal and spatial information needed to present each document component. The element may refer a `<region>` element to define the initial position of the `<media>` element (that is associated with the `<descriptor>` element) presentation in some output device. The definition of `<descriptor>` elements shall be included in the document head, inside the `<descriptorBase>` element, which specifies the set of descriptors of a document. Also inside the document `<head>` element, the `<regionBase>` element defines a set of `<region>` elements in a class of exhibition devices, each of which may contain another set of nested `<region>` elements, and so on, recursively; regions define device areas (e.g. screen windows) and are referred by `<descriptor>` elements, as previously mentioned.

A `<causalConnector>` element represents a relation that may be used for creating `<link>` elements in documents. In a causal relation, a condition shall be satisfied in order to trigger an action. A `<link>` element binds (through its `<bind>` elements) a node interface with connector roles, defining a spatio-temporal relationship among objects (represented by `<media>`, `<context>`, `<body>` or `<switch>` elements).

The `<descriptorSwitch>` element contains a set of alternative descriptors to be associated with an object. Analogous to the `<switch>` element, the `<descriptorSwitch>` choice is done during the document presentation, using test rules defined by `<rule>` or `<compositeRule>` elements.

In order to allow an entity base to incorporate another already-defined base, the `<importBase>` element may be used. Additionally, an NCL document may be imported through the `<importNCL>` element. The `<importedDocumentBase>` element specifies a set of imported NCL documents, and shall also be defined as a child element of the `<head>` element.

Some important NCL element's attributes are defined in other NCL modules. The EntityReuse module allows an NCL element to be reused. This module defines the *refer* attribute, which refers to an element URI that will be reused. Only `<media>`, `<context>`, `<body>` and `<switch>` may be reused. The KeyNavigation module provides the extensions necessary to describe focus movement operations using a control device like a remote control. Basically, the module defines attributes that may be incorporated by `<descriptor>` elements. The Animation module provides the extensions necessary to describe what happens when a property value is changed. The change may be instantaneous, but it may also be carried out during an explicitly declared duration, either linearly or step by step. Basically, the Animation module defines attributes that may be incorporated by actions, defined as child elements of `<causalConnector>` elements.

Some SMIL functionalities are also incorporated by NCL. The <transition> element and some transition attributes have the same semantics of homonym element and attributes defined in the SMIL BasicTransitions module and the SMIL TransitionModifiers module. The NCL <transitionBase> element specifies a set of transition effects, defined by <transition> elements, and shall be defined as a child element of the <head> element.

Finally, the MetaInformation module is also incorporated, inheriting the same semantics of the SMIL MetaInformation module. Meta-information does not contain content information that is used or display during a presentation. Instead, it contains information about content that is used or displayed. The Metainformation module contains two elements that allow describing NCL documents. The <meta> element specifies a single property/value pair. The <metadata> element contains information that is also related to meta-information of the document. It acts as the root element of an RDF tree: RDF element and its sub-elements (for more details, refer to W3C metadata recommendations [RDF99]).

## 4. NCL Model for Multiple Device Exhibition

An NCL application may have its exhibition spread over multiple exhibition devices exhibition spread over multiple exhibition devices working. A hierarchical control model establishes the basis of the *modus operandi*.

Any device able to process exhibition tasks is called a *processing device*. Processing devices may have output and input units. A processing device together with its output units is called an *exhibition device*. A processing device together with its input and output units is called an *exhibition-interaction device*. Note that an exhibition-interaction device is also an exhibition device that is also, in its turn, a processing device. The reciprocal is not true.

An NCL application specifies all its processing devices joining them into classes. The processing device that runs an NCL formatter responsible for processing the application's NCL document is called the *base device*. All processing devices that jointly run an NCL application compound what is called the application's *device domain*.

Since NCL applications can embed NCL media objects, that is, media objects whose content are other NCL applications, each NCL media object has its own base device and its own domain. This domain is indeed a subset of its parent NCL application's domain; and the NCL media object base device is not necessarily the base device of the parent NCL application.

### 4.1. Types of Classes and Class Registration

There is no limit for the number of device classes in an NCL application. However, there are only two types of device classes: passive and active.

Active classes are those whose members are able to run media players (including imperative and declarative object players).

Passive classes are those whose members are not required to run media players. However, the members shall be able to present the video memory map and the audio samples they receive from other processing devices, called *parent device*.

In NCL, the <regionBase> element can be associated with a device class where the presentation will take place. In order to identify the class (i) of devices, the *device* attribute can receive the "systemScreen (i)" or the "systemAudio(i)" value. When this attribute is not specified, the presentation must take place in the same exhibition device in which the NCL document is running.

Processing devices can register themselves in classes ("systemScreen (i)" and "systemAudio(i)") of a given domain. The number of classes of a given domain can be obtained from the settings node's "system.classNumber" property. A processing device may register itself in more than one class.

An specific class (i) defines the following properties of the NCL document's <media type="application/x-ncl-settings">, in which the class is defined: system.screenSize(i), system.screenGraphicSize(i) and system.audioType(i), as defined in [SoRo 06].

A class of active type shall define which media object players are available in all its registered devices.

In SBTVD, the Ginga middleware defines some default classes, for simplicity. The “systemScreen (1)” and the “systemAudio(1)” classes are reserved as classes of passive type. The “systemScreen (2)” and “systemAudio(2)” classes are reserved as classes of active type, in which all registered devices must be able to exhibit any media object type specified in the SBTVD Standard, including imperative NCLua and NCLet players and declarative NCL and HTML players. Moreover, when there is just one exhibition device in the device domain, no register is necessary. Indeed, a base device has one class only for it, where no other device can be registered and which is declared by default.

Ginga reference implementation [ref] allows creating classes and class registers for the device domain where its NCL formatter runs. A resident application is responsible for this task.

In NCL, the video memory map (or the audio sample sequence) created by a parent device to be exhibited in a passive class may also be exhibited in a region of the parent device. This region is specified in the *region* attribute of the <regionBase> element that defines the passive class. This attribute shall refer to a region defined for the parent exhibition device.

In the NCL hierarchical exhibition control model, independent from the class type, exhibition devices in one class can only exhibit media objects coming from the same parent device (explicitly or embedded in video memory maps or audio sample sequences). A class may not have more than one parent device in a given moment. Moreover, a base device may not receive objects (explicitly or embedded in video memory maps or audio sample sequences) from other devices of the same device domain. In other words, it is not possible to have a device as an ascendant or a descendent of itself.

## 4.2. Control of Input Units

In the beginning of an NCL document presentation, all input units associated with devices of the document device domain are under control of the domain base device.

When a <media> element in exhibition in a device of an active class receives the presentation focus and is selected, the device in charge of the exhibition gains control of all its input units and all input units of devices that are in classes that will be its descendents (classes for which it will be the parent device). The media player can then follow its own navigational rules. When the “BACK” key is pressed, the control of all previously mentioned input units is returned to the parent device. As usual in NCL, the focus will go to the element identified by the *service.currentFocus* attribute of the settings node (<media type=“application/x-ncl-settings”).

It must be noted that there can be more than one device controlling the key navigation, but each one controlling different input units from the others.

## 5. Behavior of Exhibition Devices

In order to exemplify the behavior of exhibition devices when playing an NCL application, let it be the following scenario:

1. A video, an animation, about a famous soccer player exhibited in a primary device (a TV set, for example)
2. During the video an advertisement about a soccer shoes will be played in a class of secondary devices.
  - a. When the advertisement is ready to be presented an icon will appear in the TV set, during a certain period of time, in order to notify the existence of secondary exhibition.
  - b. At the same period of time a soccer shoes icon will appear on secondary devices. If a viewer selects this icon, an advertisement video and an HTML form will appear on secondary devices over a background picture; the icon presentation in these devices will stop.
  - c. The end of the advertisement video will end the exhibition on secondary devices.

Figure 1 presents a structural view of this scenario

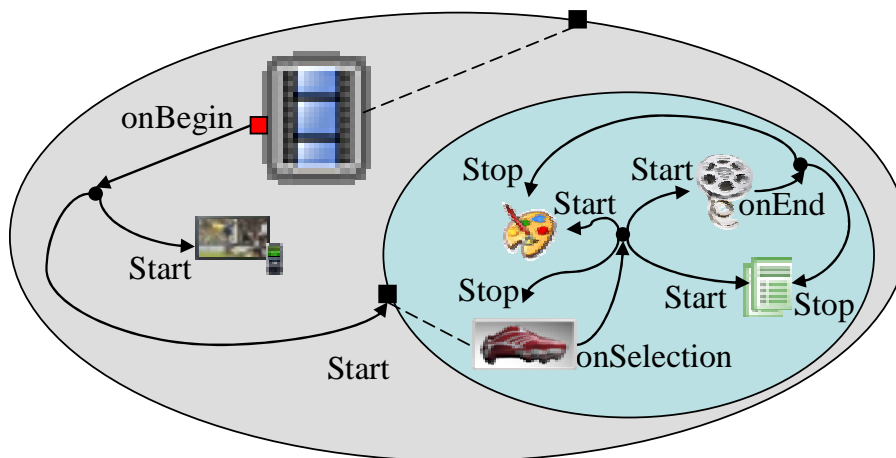


Figure 1 – Structural view of an NCL application

### 5.1. Device Behavior in a Passive Class

Let us assume that secondary devices of Figure 1 will be declared in a passive class, as shown in Figure 2.

```

<regionBase>
  <region id="screenReg" width="100%" height="100%" zIndex="1">
    <region id="secIconReg" left="87.5%" top="11.7%" width="8.45%"
      height="6.7%" zIndex="2"/>
  </region>
</regionBase>
<regionBase device="systemScreen(1)">
  <region id="backgroundReg" width="100%" height="100%" zIndex="1">
    <region id="iconReg" width="100%" height="100%" zIndex="2"/>
    <region id="shoesReg" left="5%" top="30%" width="40%" height="40%"
      zIndex="2"/>
    <region id="formReg" left="50%" top="5%" width="45%" height="90%"
      zIndex="2"/>
  </region>
</regionBase>

```

Figure 2 – Passive “systemScreen(1)” class specification

In Figure 2 two region bases, and thus two classes of devices, are defined. The first one defines presentation regions for the primary device (the base device declared by default). In this device, the video occupies the whole TV screen and the icon is positioned according to its left, top, width and height attributes. The second class defines presentation regions for secondary devices. The background presentation region and the icon presentation region occupy the whole secondary screen. The soccer shoes advert and the form are positioned according to their left, top, width and height attributes. As aforementioned, in Ginga, “systemScreen(1)” is defined as a passive class, by default.

As mentioned in Section 4.1, the parent device is in charge of running media players for media objects to be exhibited in a passive class it controls. The parent device must transmit the video memory map of the exhibition plan (frame buffer), in the case of visual output, and audio sample sequences, in the case of audio output to its child devices (in the passive class). On the other hand, a device in the passive class must be able to scan the video memory map and the mixed audio samples received and present them.

When a media object has to be exhibited on devices included in a passive class, only one exhibition instance must be created. This instance, created by the parent device, is then shared by all child devices in the passive class.

It should be mentioned that there is no zIndex associated with the created video memory maps or audio samples. They will be presented with zIndex = 0, that is, the lowest zIndex value. If a device receives more than one video memory map only the last one will be presented. The same procedure works for mixed audio sample sequences.

If a <media> element in exhibition in a passive class receives the navigational focus, the media player in the parent device gains control of all input units controlled by the parent device. From then on this media player can follow its own navigational rules until the control is returned to the parent device by pressing the “BACK” key.

Turning back to the example of Figure 1, let us assume that all advertisement content (the soccer shoes icon, the background picture and the soccer shoes video) is within an NCL

<context id="advert"...> element. It is this context that must be exhibited in exhibition devices registered in the passive "systemScreen(1)" class defined in Figure 2.

Figure 3 presents the complete NCL document for the example. Note that if the soccer shoes icon is selected, the soccer shoes video, the HTML form and the background picture will appear in every device registered in "systemScreen(1)". If the HTML form is selected (note that it begins its presentation with the focus) in any of these devices, any navigation inside the form made by any input unit associated with any exhibition device will be displayed in every device registered in "systemScreen(1)". If an individual navigation on the form is desired, the solution presented in the next section must be assumed.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--Multiple exhibition devices in a passive class: an example -->
<ncl id="devices" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
  <head>
    <regionBase>
      <region id="screenReg" width="100%" height="100%" zIndex="1">
        <region id="secIconReg" left="87.5%" top="11.7%" width="8.45%"
          height="6.7%" zIndex="2"/>
      </region>
    </regionBase>
    <regionBase device="systemScreen(1)">
      <region id="backgroundReg" width="100%" height="100%" zIndex="1">
        <region id="iconReg" width="100%" height="100%" zIndex="2"/>
        <region id="shoesReg" left="5%" top="30%" width="40%" height="40%"
          zIndex="2"/>
        <region id="formReg" left="50%" top="5%" width="45%" height="90%"
          zIndex="2"/>
      </region>
    </regionBase>
    <descriptorBase>
      <descriptor id="backgroundDesc" region="backgroundReg"/>
      <descriptor id="screenDesc" region="screenReg"/>
      <descriptor id="iconDesc" region="iconReg" explicitDur="6s"/>
      <descriptor id="secIconDesc" region="secIconReg" explicitDur="6s"/>
      <descriptor id="shoesDesc" region="shoesReg"/>
      <descriptor id="formDesc" region="formReg" focusIndex="1"/>
    </descriptorBase>
    <connectorBase>
      <importBase documentURI="../../causalConnBase.ncl" alias="conEx"/>
    </connectorBase>
  </head>
  <body>
    <port id="entry" component="animation"/>
  </body>
</ncl>
```

```

<media id="animation" src="../../media/animGar.mp4"
                                descriptor="screenDesc">
  <area id="segIcon" begin="45s" end="51s"/>
</media>
<media id="secIcon" src="../../media/secIcon.png"
                                descriptor="secIconDesc"/>
<context id="advert">
  <port id="pIcon" component="icon"/>
  <media id="icon" src="../../media/icon.png" descriptor="iconDesc"/>
  <media id="background" src="../../media/background.png"
                                descriptor="backgroundDesc"/>
  <media id="shoes" src="../../media/shoes.mp4" descriptor="shoesDesc"/>
  <media id="ptForm" src="../../media/ptForm.htm" descriptor="formDesc"/>
  <link id="lBegingAdvert"
        xconnector="conEx#onKeySelectionStopStart">
    <bind role="onSelection" component="icon">
      <bindParam name="keyCode" value="RED"/>
    </bind>
    <bind role="stop" component="icon"/>
    <bind role="start" component="background"/>
    <bind role="start" component="ptForm"/>
    <bind role="start" component="shoes"/>
  </link>
  <link id="lEndshoes" xconnector="conEx#onEndStop">
    <bind role="onEnd" component="shoes"/>
    <bind role="stop" component="background"/>
    <bind role="stop" component="ptForm"/>
  </link>
</context>
<link id="lIcon" xconnector="conEx#onBeginStart">
  <bind role="onBegin" component="animation" interface="segIcon"/>
  <bind role="start" component="secIcon"/>
  <bind role="start" component="advert" interface="pIcon"/>
</link>
</body>
</ncl>

```

Figure 3 – NCL application using multiple devices registered in a passive class

Figure 3 shows how easily presentation in multiple devices can be defined. Only one element (the <regionBase device="systemScreen(1)">) had to be added to distinguish the multiple device presentation from the one presenting all media objects in the TV set.

In NCL, the video memory map presented in secondary devices can also be shown in a region of the parent device. For that, it is sufficient to add an attribute to the <regionBase>

element that defines the “systemScreen(i)” class, referring to a region defined in the <regionBase> element associated with the parent device, as shown in bold face in Figure 4 for the example of Figure 3.

```
<regionBase>
  <region id="screenReg" width="100%" height="100%" zIndex="1">
    <region id="memoryMap" left="87.5%" top="87.5%" width="10%"
      height="10%" zIndex="2"/>
  ...
  </region>
</regionBase>
<regionBase device="systemScreen(1)" region="memoryMap">
  ...
</regionBase>
```

Figure 4 – Video memory map presented in the parent device

## 5.2. Device Behavior in a Active Class

When dealing with an active class, the required processing for playing media object contents is transferred from the parent device to the class’ processing devices.

Each device in an active class must create its own media-object presentation instance for each media object it controls. In a <link> element definition, <bind> elements referring to this media-object must have its cardinality *max* attribute set to “unbounded”. Every presentation or attribution *action* (*stop*, *abort*, *pause* e *resume*) [ref] on any of these instances must result in identical actions in all other instances (in all devices in the class) of the same media object (actions shall be performed in any order). Any *condition* [ref] derived from these media object presentations shall be considered fulfilled, if it is fulfilled by all instances (not necessarily simultaneously), in the case its <bind> elements’ *qualifier* attribute is set to “and”, or if it is fulfilled by any instance, in the case its <bind> element’s *qualifier* attribute is set to “or”.

While (upon receiving the focus [SoRo06]) no <media> element controlled by devices in an active class is select to gain the key navigation control, all viewer interactions through these devices’ input units shall be passed to the parent device. If a <media> element controlled by devices in an active class is select to gain the key navigation control in a specific device X, the particular media player of this device X gains control of all input units associated with X and with all devices in descendent classes controlled by X. From then on the media player can follow its own navigational rules until the control is returned to the parent device by pressing the “BACK” key.

Turning back to the example of Figure 1, let us substitute the <context id="advert"> element by an NCL media node (<media id="NCLAdvert" src=advert.ncl ...>). This NCL media object, defined as an NCL application apart (advert.ncl), will be presented in devices registered in an active class, as defined by Figure 5.

```
<regionBase device="systemScreen(2)">
  <region id="NCLAdvertReg" width="100%" height="100%" zIndex="1"/>
</regionBase>
```

Figure 5 – Active “systemScreen(2)” class specification

On starting the NCLAdvert object presentation, the key navigation control is passed to it by the same <link> element used for its starting, setting the NCL “service.currentKeyMaster” global property to this object id, as shown in Figure 6. This means that, from then on each device in “systemScreen(2)” can navigate inside NCLAdvert independently.

```
<link id="lBeginAdvert" xconnector="conEx#onBeginSetStart">
  <bind role="onBegin" component="animation" interface="segIcon"/>
  <bind role="start" component="secIcon"/>
  <bind role="start" component="NCLAdvert"/>
  <bind role="set" component="globalVar"
    interface="service.currentKeyMaster">
    <bindParam name="var" value="NCLAdvert"/>
  </bind>
</link>
```

Figure 6 – Starting the NCL media object to be presented by devices in an active class

Figure 7 presents the NCL media object specification, very similar to the context specification of Figure 3.

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- NCLAdvert "application/x-ginga-NCL" specification-->
<ncl id="advert" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
  <head>
    <regionBase>
      <region id="backgroundReg" width="100%" height="100%" zIndex="1">
        <region id="iconReg" width="100%" height="100%" zIndex="2"/>
        <region id="shoesReg" left="5%" top="30%" width="40%" height="40%"
          zIndex="2"/>
        <region id="formReg" left="50%" top="5%" width="45%" height="90%"
          zIndex="2"/>
      </region>
    </regionBase>
    <descriptorBase>
      <descriptor id="backgroundDesc" region="backgroundReg"/>
      <descriptor id="shoesDesc" region="shoesReg"/>
      <descriptor id="formDesc" region="formReg" focusIndex="1"/>
      <descriptor id="iconDesc" region="iconReg" explicitDur="6s"/>
    </descriptorBase>
    <connectorBase>
      <importBase documentURI="../causalConnBase.ncl" alias="conEx"/>
    </connectorBase>
  </head>
  <body>
    <port id="pIcon" component="icon"/>
    <media id="icon" src="../media/icon.png" descriptor="iconDesc"/>
    <media id="background" src="../media/background.png"
      descriptor="backgroundDesc"/>
    <media id="shoes" src="../media/shoes.mp4" descriptor="shoesDesc"/>
    <media id="ptForm" src="../media/ptForm.htm" descriptor="formDesc"/>
    <link id="lBegingAdvert" xconnector="conEx#onKeySelectionStopStart">
      <bind role="onSelection" component="icon">
        <bindParam name="keyCode" value="RED"/>
      </bind>
      <bind role="stop" component="icon"/>
      <bind role="start" component="background"/>
      <bind role="start" component="ptForm"/>
      <bind role="start" component="shoes"/>
    </link>
    <link id="lEndshoes" xconnector="conEx#onEndStop">
      <bind role="onEnd" component="shoes"/>
      <bind role="stop" component="background"/>
      <bind role="stop" component="ptForm"/>
    </link>
  </body>
</ncl>

```

Figure 7 – NCL specification for NCLAdvert object

The NCL master application is shown in Figure 8.

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!--Multiple devices with independent navigation -->
<ncl id="indDevices" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
  <head>
    <regionBase>
      <region id="screenReg" width="100%" height="100%" zIndex="1">
        <region id="secIconReg" left="87.5%" top="11.7%" width="8.45%"
          height="6.7%" zIndex="2"/>
      </region>
    </regionBase>

    <regionBase device="systemScreen(2)">
      <region id="NCLAdvertReg" width="100%" height="100%" zIndex="1"/>
    </regionBase>

    <descriptorBase>
      <descriptor id="screenDesc" region="screenReg"/>
      <descriptor id="secIconDesc" region="secIconReg" explicitDur="6s"/>
      <descriptor id="NCLAdvertDesc" region="NCLAdvertReg"/>
    </descriptorBase>
    <connectorBase>
      <importBase documentURI="../causalConnBase.ncl" alias="conEx"/>
    </connectorBase>
  </head>
  <body>
    <port id="entry" component="animation"/>
    <media id="animation" src="../media/animGar.mp4"
      descriptor="screenDesc">
      <area id="segIcon" begin="45s" end="51s"/>
    </media>
    <media id="globalVar" type="application/x-ginga-settings">
      <property name="service.currentKeyMaster"/>
    </media>
    <media id="secIcon" src="../media/icon.png" descriptor="secIconDesc"/>
    <media id="NCLAdvert" src="advert.ncl" descriptor="NCLAdvertDesc"/>
    <link id="lBeginAdvert" xconnector="conEx#onBeginSetStart">
      <bind role="onBegin" component="animation" interface="segIcon"/>
      <bind role="start" component="secIcon"/>
      <bind role="start" component="NCLAdvert"/>
      <bind role="set" component="globalVar"
        interface="service.currentKeyMaster">
        <bindParam name="var" value="NCLAdvert"/>
      </bind>
    </link>
  </body>
</ncl>

```

Figure 8 – NCL application using multiple devices registered in an active class

Figure 9 shows screens of multiple devices after running Figure 1 application in Ginga, the middleware of the Brazilian Terrestrial Digital TV System.



Figure 9 – Running multiple device applications in Ginga-NCL

Following the NCL hierarchical exhibition control model, a media object is sent to a device in an active class from its parent device one by one, when the moment of the object presentation arrives. However there is an optional exception of this procedure.

If devices in the active class are able to play NCL applications (that is, if devices implement the NCL formatter), set of objects can be received in block.

Based on the Hypermedia Temporal Graph [CoMS 08] defined by an NCL application, an NCL formatter running on the parent device may compute which part of a temporal chain (sequence of actions on object presentations), derived from temporal graph, will be exhibit by devices in an active class. If devices in this class are able to run NCL applications, the NCL formatter running in the parent device may choose to pass the entire chain and the referred objects to them. Based on the received chain, NCL formatters are then able to instantiate objects they must control; instead of receive command to instantiate each object in the chain one by one.

### 5.3. Device Registered in Passive and Active Classes

As aforementioned, a device can be registered in many active and passive classes simultaneously, since all classes are controlled by the same parent device in each moment in time. In this case, the device inherits the behavior of both types of classes:

- An exhibition device can only present objects passed (explicitly, or nested in other explicitly received object, or embedded in video memory maps/audio sample sequences) to classes in which it is registered;
- A base device may not receive objects (explicitly or embedded in video memory maps/audio samples) passed by other devices in the device domain it defines;
- When an object must be presented by devices in an active class, individual instances are created by each device in the class. Devices in the class must then behavior as specified in Section 5.2;

- When an object must be presented by devices in a passive class, a video memory map and an audio sample sequence are created and controlled by the parent device, which will create only one instance for each object to be presented. Child devices must behavior as specified in Section 5.1;
- There is no zIndex definition for the video memory map or the audio sample sequence received from the parent device. They must be presented with zIndex=0 in child devices. If more than one video memory map or more than one sequence of audio samples is received, only the last one must be present. Every object presentation received from an active class has its presentation placed on the top of any already received video memory map or sequence of audio samples.

#### **5.4. Presentation Adaptation**

As aforementioned, to work with class of devices during the specification phase allows authors to be unaware of the number of devices in a given class, which can vary over time. However this number can be zero. In this case, content forwarded to this class would not be exhibited. It would be nice if an author could specify an alternative presentation for this situation.

As discussed in [SoRo06], a media object initial positioning is determined in NCL when a <descriptor> element is associated with a <region> element, which in its turn is associated with a class of devices. Moreover, NCL allows specifying alternative descriptors for a media object presentation using <descriptorSwitch> elements. An alternative is chosen during runtime based on rule evaluations. Rules can be defined over global variables maintained by the NCL formatter. Figure 10 shows an example of an alternative presentation when the number of devices in the `systemScreen(2)` class (defined by the `system.devNumber(2)` global variable) is zero. In this case, the advertisement is deviated to be presented in the TV set, superposed to the animation.

```

<head>
  <ruleBase>
    <rule id="single" var="system.devNumber(2)" value="0" comparator="eq"/>
  </ruleBase>
  <regionBase>
    <region id="screenReg" width="100%" height="100%" zIndex="1">
      ...
      <region id="NCLAdvertSingleReg" left="5%" top="5%"
        width="30%" height="30%" zIndex="2"/>
    </region>
  </regionBase>
  <regionBase device="systemScreen(2)">
    <region id="NCLAdvertMultiReg" width="100%" height="100%" zIndex="1"/>
  </regionBase>

  <descriptorBase>
    ...
    <descriptorSwitch id="NCLAdvertDesc">
      <bindRule constituent="NCLAdvertSingleDesc" rule="single"/>
      <defaultDescriptor descriptor="NCLAdvertMultiDesc"/>
      <descriptor id="NCLAdvertSingleDesc" region="NCLAdvertSingleReg"/>
      <descriptor id="NCLAdvertMultiDesc" region="NCLAdvertMultiReg"/>
    </descriptorSwitch>
  </descriptorBase>
  ...
</head>

```

Figure 10 – Alternative presentation when there are no devices in the class to where an object is addressed

## **6. Final Remarks**

In order to offer a scalable hypermedia model, with characteristics that may be progressively incorporated in hypermedia system implementations, NCM was divided in several parts, and also its declarative XML application language: NCL. This technical report describes the support offered by NCL 3.0 to multiple exhibition devices, which comprises Part 12: Support to Multiple Exhibition Devices.

### **Acknowledgements**

Many people have contributed to the definition of the NCL multiple device support. Chief among them are Romualdo Rezende Costa, Marcio Ferreira Moreno, and Marcelo Ferreira Moreno.

## References

- [Anto00] Antonacci M.J. NCL: Uma Linguagem Declarativa para Especificação de Documentos Hiperímia 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 Hiperímia na Web, **VI Simpósio Brasileiro de Sistemas Multimímia e Hiperímia - SBMímia2000**, Natal, Rio Grande do Norte, June 2000.
- [CoMS 08] Costa R. R; Moreno, M. F.; Soares, L. F. G. Intermedia Synchronization Management in DTV Systems. *Proceedings of the ACM Symposium on Document Engineering*. São Paulo, Brazil. September 2008; pp. 289-297. ISBN: 978-1-60558-081-4.
- [RDF99] Resource Description Framework (RDF) Model and Syntax Specification, Ora Lassila and Ralph R. Swick. W3C Recommendation, 22 February 1999. Available at <http://www.w3.org/TR/REC-rdf-syntax/>
- [SCHE01] XML Schema Part 0: Primer, **W3C Recommendation**, in <http://www.w3.org/TR/xmlschema-0/>, May 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.
- [SoRo06] Soares L.F.G; Rodrigues R.F. Nested Context Language 3.0: Part 8 – NCL Live Editing Commands, **Technical Report, Departamento de Informática PUC-Rio**, December 2006, ISSN: 0103-9741.
- [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.