NCL 3.1 Enhanced DTV Profile

Luiz Fernando Gomes Soares
Lab. TeleMídia - DI – PUC-Rio
Rua Marquês de São Vicente 225
22453-900 Rio de Janeiro, RJ
+55-21-3527-1500 Ext: 4330
lfgs@inf.puc-rio.br

Guilherme Lima
Lab. TeleMídia - DI – PUC-Rio
Rua Marquês de São Vicente 225
22453-900 Rio de Janeiro, RJ
+55-21-3527-1500 Ext: 3503
gflima@telemidia.puc-rio.br

Carlos de Salles Soares Neto
Lab. TeleMídia - DI – PUC-Rio
Rua Marquês de São Vicente 225
22453-900 Rio de Janeiro, RJ
+55-21-3527-1500 Ext: 3503
csalles@telemidia.puc-rio.br

ABSTRACT
NCL (Nested Context Language) in its current version 3.0 has been evaluated both through empirical and analytic methods, which have provided important insights on how to interpret the meaning and impact of syntactic and semantic features of NCL on human cognition. The several evaluations raised some very interesting issues that lead to the design of the new NCL 3.1 version.

This paper presents the new features of the NCL 3.1 Enhanced Digital TV profile and gives good reasons for them, recognizing the evaluation results.

Categories and Subject Descriptors
D.3.3 [**Programming Languages**]: Language Constructs and Features – *control structures.*

General Terms
Standardization, Languages.

Keywords
Nested Context Language – NCL, Ginga-NCL presentation environment, NCL EDTV profile.

1. INTRODUCTION
NCL (Nested Context Language), in its current version 3.0, has been evaluated both through empirical methods (involving empirical observations of how people actually use the NCL features in real situations or realistic lab settings) [9], and through analytic methods (derived from theories, models or frameworks, in varying degrees of formality) [11]. In particular, analytic methods have helped us to detect specific features of NCL that we (its designers) were not aware of.

An analytic evaluation [11] of the usability of NCL, based on the Cognitive Dimensions of Notations (CDN) Framework [2; 3], has provided important insights on how to interpret the meaning and impact of syntactic and semantic features of NCL on human cognition.

The several evaluations raised some very interesting issues to be taken into consideration in the design of the next NCL version. As stated in the NCL specification under ITU-T Recommendation H.761 [6] and ABNT (ISDB-T_B) Standard NBR 15606-2 [1]: "The version number of NCL consists of a major number and a minor number, separated by a dot. New NCL versions shall be released in accordance to the following versioning policy: If receivers that conform to older versions can still receive a document based on the revised specification, in relation to error corrections, operational reasons, or the addition of a new concise syntax notation ("syntax sugar") that can be translated at compile time to the old one, the new version of NCL shall be released with the minor number updated. If receivers that conform to older versions cannot receive a document based on the revised specifications, the major number shall be updated."

This paper focuses on some changes made on the current NCL 3.0 version [7], which are brought about by the previously mentioned empirical and analytic analysis [9; 11], such that receivers that conform to older versions can still receive a document based on the revised specification. More precisely, it presents the new features of the NCL 3.1 Enhanced Digital TV (EDTV) profile and gives good reasons for them.

The next sections are organized as follows. Section 2 discusses how some redundant entities established to provide reuse features can cause usability problems, especially in large documents. Section 3 introduces some small changes in the NCL 3.0 syntax to solve part of the problems raised in Section 2. Section 4 discusses the new syntactic sugar[1] added to the NCL 3.1 EDTV profile to solve the other part of problems raised in Section 2. Section 5 discusses the impact of the new NCL EDTV profile in the Ginga-NCL presentation engine. Finally, Section 6 is reserved for conclusions.

2. REUSE FEATURES
The general NCL structure is composed of a header and a body. The header has several information bases and the body specifies the organizational structure and the presentation semantics of

WebMedia 2010, October 5–8, 2010, Belo Horizonte, Minas Gerais, Brazil.

[1] In the context of this paper, a syntactic sugar refers to a syntax within NCL that is designed to allow things to be expressed more clearly, more concisely, or in an alternative style that someone may prefer, while alternative ways of expressing them exist.

NCL documents. This general structure shows the designers' concern with reuse in NCL, in which language elements in the body frequently refer to elements in the header.

The NCL language design and its conceptual model drive application authors to create documents with high reuse degree [10], and to conjecture that sustained use of NCL leads to good programming practices. However, in order to move from supporting to promoting reuse, a specification language must itself have a number of usability merits. If it doesn't, programmers (or multimedia document authors in our specific context) are likely to abandon it for a more usable specification tool.

To promote reuse, NCL 3.0 introduced a series of syntactic sugars to the language. As an example, let us take the positioning of media-object content in a screen. Being a media-object property, the positioning may be defined in a <property> element. In order to be reused by other media objects, the positioning may also be defined in a <descriptor> element. Moreover, to be reused by descriptors, the positioning may be defined in a <region> element.

The use of these syntactic sugars can bring about some problems, however. First, the diffuseness of NCL code can create problems when reusing an element introduces dependency chains (i. e., one element reuses another, which reuses another, and so on and so forth). Some common element chains are:

media -> descriptor -> region

media -> descriptor -> transition

switch -> rule -> bindRule

Second, there are several examples of hidden dependency problems specifically related to reuse. For example, when an attribute of a region changes, all associated descriptors collaterally receive the effects of this change. Likewise, changes in descriptors have impact on media objects that refer to them.

Third, especially in large documents, elements that refer others can introduce visibility cognitive dimension problems. That is, sometimes all needed information is not easily identified and accessible when authors are editing part of an NCL document.

Of course, all mentioned cognitive problems can be avoided if authors specify all attributes of an element without referring to other element. For example, all presentation attributes should use, in this case, <property> elements.

It is important to stress that NCL allows for, but does not impose, reuse practices. However, does NCL 3.0 really have the same expressiveness without using the reuse syntactic sugars? Unfortunately, it does not. To solve this problem, some minor changes must be carried out in the new NCL 3.1 EDTV profile, as presented in Section 3, and a new syntactic sugar must be introduced, as discussed in Section 4.

3. MINOR CHANGES ON NCL OBJECT PROPERTIES

To provide the same expressiveness when reuse facilities are not employed and also to maintain a cognitive coherence over all NCL entity, some minor chances must be introduced in the NCL 3.1 EDTV profile.

3.1 Specification of Presentation Properties

When specifying presentation properties for NCL media objects, references to <descriptor> and <region> elements are optional. A <descriptor> element could define a region using <descriptorParam> elements, thus without referring to <region> elements. A <media> element can also define all necessary presentation properties using <property> elements, without reference to a <descriptor> element. Indeed, descriptors are used only to initialize these properties.

However, by default, presentation properties defined in <descriptor> and <region> elements are not visible for external reference by using <link> elements. On the other hand, presentation properties defined in <property> elements are visible for external reference by using <link> elements, by default. There is no way to define in a <property> element a property that is not externalized.

To solve this problem, NCL 3.1 maintains the same definition of NCL 3.0 but introduces a new attribute, named *externable*, to the <property> element. Changing the value of this attribute, the possibility of using the property as a <link>'s role can be controlled.

Moreover, in NCL 3.0 specification, if no <descriptor> element is defined to be associated to a <media> object, a default descriptor (specific to each media object type) should be used to initialize all properties with default values. In other words, default values are defined inside the default descriptor.

In a different way, NCL 3.1 EDTV specification does not use the concept of "default descriptor". If no value is attributed to a specific property, no matter how it is defined, a default value should be applied. Defaults are part of the NCL 3.1 specification and not defined apart. Simple like that, but simplifying a lot the Ginga-NCL implementation.

3.2 Transition Specification

Unlike what happens with regions, a <descriptor> element must always refer to a transition if the effect is requested. There is no way to define transitions as parameters of <descriptor> elements. Likewise, it is impossible to define transitions as presentation properties in <media> elements.

This means that transitions are more prone to visibility cognitive problems than regions and descriptors. Although this has not been considered relevant in the NCL 3.0 EDTV profile, the concept was revised in NCL 3.1 EDTV profile. Several new attributes were defined for specifying transitions and are able to be manipulated as any other attribute of a <property> element. Appendix A shows the reserved name of this predefined attributes and their meaning.

3.3 Rule Specification

In NCL 3.0 specification, it is impossible to reuse a rule in order to define other rules. Moreover, rules can refer to global properties (properties of a special media object type: "application/x-ncl-settings") without having them explicitly declared. On the other hand, any other use of a property requires that the property is explicitly declared. Rules are rare exceptions in the NCL 3.0 reuse coherence.

In NCL 3.1 EDTV profile, to reuse a rule in order to define other rules is not considered an important subject. Therefore, the same NCL 3.0 specification was kept.

On the other hand, the language coherence requiring that any property to be referred must be explicitly declared is considered an important issue. Therefore, in NCL 3.1 EDTV profile, all global properties (properties of a special media object type: "application/x-ncl-settings") used in rule definitions must be explicitly declared in a <property> element.

4. LINK SPECIFICATION

Unlike other XML languages, NCL detaches the relation and the relationship concepts [4; 7], as it is usual in ADLs (Architecture Description Languages [4]). Relations can be considered a type definition for relationships, and relationships relation instantiations.

Relations are defined in a relation base in the head part of a document (<head> element). Relations define roles and the glue relating roles. NCL allows defining any kind of relation, but reserved words has been defined to simplify the definition of causal temporal and spatial relations. Causal relations are defined by <causalConnector> elements. In causal relations, conditions defined over roles must be satisfied in order to trigger actions to be applied in roles (the same or others).

Relationships (represented by NCL <link> elements) can be defined referring to a relation and defining actors to play the relation roles. Thus relation reuse is natural in NCL.

The disconnection between hypermedia relations, on the one side, and relationships, on the other, is an important feature for NCL expressiveness and reuse. The definition of relation types is the most difficult task in authoring a document. However, once relations are defined, they can be reused in several relationships. It is common to see iDTV producers having a well defined base of relations made up by expert programmers and shared among naïve document authors.

NCL documents typically have several elements that refer to other elements. This is a welcome feature for reuse but it can also create the need to check non-visible code. When specifying a link, for example, the author needs to check the role cardinality in the referred connector specification. This information is not immediately available and can be misleading. The referring mechanism can lead to various other kinds of mistakes. Avoiding this is a matter of language design and programming infrastructure.

Since relationships in NCL 3.0 are always defined referring to a previously defined relation, this can cause visibility cognitive problems. Visibility problems in using a predefined relation can be minimized using a good naming strategy for relation identification. Terminology that is well known by authors is an asset in all authoring tasks. However, for the rare cases where new relation definitions are needed, a visibility problem can still persist, since relationships are created far from where relations are defined in the text.

As opposed to other optional reuse features of NCL 3.0 (like the layout definition) there is no option in this case. In order to bypass this problem, a syntactic sugar is introduced by the NCL 3.1 EDTV profile. This approach allows for the joint definition of a relationship and its relation in a pseudo code, written in quasi natural language, which composes the content of a link element.

Different from NCL 3.0, in the NCL 3.1 EDTV profile, a <link> element may have a content, but only if the link does not refer to a <causalConnector> element and does not have any child element.

4.1 Link Content Specification

In the definition of the <link> element's content, the following EBNF meta-characters are used:

;	production terminator
{}	group of symbols
|	alternatives
-	character range

?	zero or one
*	zero or more
+	one or more

The syntax for <link>'s content is very simple and always refers to a condition list that when satisfied triggers the execution of an action list:

<link>condList **then** actList **end</link>**

In the proposed syntax, NCL assessments of connector's roles are usual conditions. The BNF definition of a link is as follows:

```
link     = condlist "then" actlist "end";

condlist = "(" condlist ")" withparams
           {lop condlist}?
         | condition {lop condlist}?;

condition = condname perspective withparams
          | assessment withparams;

assessmt = assessexpr rop assesexpr;

assesxpr = perspective {"+" string}?
         | string {"+" perspective}?;

condname = "onAbort" | "onBegin"
         | "onBeginAttribution"
         | "onEndAttribution" | "onEnd"
         | "onPause"
         | "onResume" | "onSelection";

actlist  = "(" actlist ")" withparams
           {aop actlist}?
         | action {aop actlist}?;

action   = actnoset perspective withparams
         | "set" perspective "=" string withparams
         | "set" perspective "=" perspective
           withparams;

actnoset = "abort" | "pause" | "resume" | "start"
         | "stop";

perspective = idref {"." idref}?;

withparams  = {"with" {parameter,}*
              parameter{","}?}?;

parameter   = idref "=" string;

string = "'"character sequence"'"
       | """character sequence""";

aop = "||" | {";"}?;

lop = "and" | "or";

rop = "lt" | "gt" | "lte" | "gte" | "eq" | "ne";
```

```
idref = {"a"-"z" | "A"-"Z" | "_" | ":"}
      | {"a"-"z" | "A"-"Z" | "_" | ":"
      | "." | "-" | "0"-"9"}+;
```

4.1.1 Examples

In order to show the simplicity of this new syntax notation and how it benefits the authoring process let us see some example of link specifications.

1) NCL 3.1 new notation:

<link>onBegin a then start b with delay="2s" end**</link>**

The same relationship using connector and link:

```
<connectorBase>
    <causalConnector id="onBeginStart">
        <connectorParam name="delay"/>
        <simpleCondition role="onBegin"/>
        <simpleAction role="start" delay="$delay"/>
    </causalConnector>
</connectorBase>

<link xconnector="onBeginStart">
    <bind role="onBegin" component="a" />
    <bind role="start" component="b">
        <bindParam name="delay" value="2s"/>
    </bind>
</link>
```

2) NCL 3.1 new notation:

<link>onBegin a and settings.user.age gt "18" then start b end **</link>**

The same relationship using connector and link:

```
<causalConnector id="onBeginTestStart">
    <compoundCondition operator="and">
        <simpleCondition role="onBegin"/>
        <assessmentStatement comparator="gt">
            <attributeAssessment role="nodeTest"
            eventType="attribution" attributeType="nodeProperty"/>
            <valueAssessment value="18"/>
        </assessmentStatement>
    </compoundCondition>
    <simpleAction role="start"/>
</causalConnector>

<link xconnector="onBeginTestStart">
    <bind role="onBegin" component="a" />
    <bind role="nodeTest" component="settings"
                                    interface="user.age" />
    <bind role="start" component="b" />
</link>
```

3) NCL 3.1 new notation:

<link>onEnd game and game.score gt game.bestScore then set game.bestScore=game.score end **</link>**

The same relationship using connector and link:

```
<causalConnector id="onEndTestSet">
    <connectorParam name="var"/>
    <compoundCondition operator="and">
        <simpleCondition role="onEnd"/>
        <assessmentStatement comparator="gt">
            <attributeAssessment role="thisScore"
            eventType="attribution" attributeType="nodeProperty"/>
            <attributeAssessment role="bestScore"
            eventType="attribution" attributeType="nodeProperty"/>
        </assessmentStatement>
    </compoundCondition>
    <simpleAction role="set" value="$var"/>
</causalConnector>

<link xconnector="onEndTestSet">
    <bind role="onEnd" component="game" />
    <bind role="thisScore" component="game"
                                    interface="score"/>
    <bind role="bestScore" component="game"
                                    interface="bestScore" />
    <bind role="set" component="game" interface="bestScore">
        <bindParam name="var" value="$get"/>
    </bind>
    <bind role="get" component="game" interface="score"/>
</link>
```

4) NCL 3.1 new notation:

<link>onBegin enForm and system.language eq "pt" then stop enForm **;** start ptForm end**</link>**

a) The same relationship using connector and link:

```
<causalConnector id="onBeginTestStopStart">
    <connectorParam name="var"/>
    <compoundCondition operator="and">
        <simpleCondition role="onBegin"/>
        <assessmentStatement comparator="eq">
            <attributeAssessment role="test"
            eventType="attribution" attributeType="nodeProperty"/>
            <valueAssessment value="$var"/>
        </assessmentStatement>
    </compoundCondition>
    <compoundAction operator="seq">
        <simpleAction role="stop"/>
        <simpleAction role="start"/>
    </compoundAction>
</causalConnector>

<link xconnector="onBeginTestStopStart">
```

```
    <bind role="onBegin" component="enForm" />
    <bind role="test" component="settings"
                                interface="system.language">
        <bindParam name="var" value="pt"/>
    </bind>
    <bind role="stop" component="enForm" />
    <bind role="start" component="ptForm" />
</link>
```

b) The same relationship using switch:

```
<rule id="rPt" var="system.language" comparator="eq"
      value="pt"/>

<switch id="s">
          <bindRule constituent="ptForm" rule="rPt"/>
          <defaultComponent component="enForm"/>
          <media id="enForm" ... />
          <media id="ptForm" ... />
</switch>
```

5. IMPACT ON GINGA-NCL

Ginga-NCL is the presentation engine of NCL (the NCL player) [8]. The impact of the NCL 3.1 new features discussed in Section 3 in Ginga-NCL implementation is almost nil.

All the new properties and the new attribute created are already in internal use by the presentation engine. The only thing that must be changed is to allow for manipulating these properties and attribute like any other. This change indeed simplifies the Ginga-NCL implementation since now no especial case exists.

Likewise, properties referred by <rule> elements will have now the same requirements of any property that must be explicitly defined to external use.

The impact of NCL 3.1 new <link> element definition is also very simple. An NCL parser can be responsible for translating the <link> content into NCL <causalConnector>, and <link> elements with child <bind> elements.

6. CONCLUSIONS

The empirical and analytic evaluation of the NCL 3.0 EDTV profile raised several cognitive problems whose solutions should be the focus of new NCL versions.

The NCL 3.1 EDTV profile does not aim to introducing any new expressiveness to NCL, but only to solve the raised cognitive problems and make even easier the authoring of NCL applications, as exemplified in Section 4.

The NCL 3.1 EDTV profile is the initial step towards the NCL 4.0 version, which will include support to 3D objects, among other features; in this case extending the NCL expressiveness indeed.

The work in the TeleMídia Lab, where NCL and Ginga-NCL were conceived, on a new presentation engine is evolving with high priority and we expect to run a complete solution for the NCL 3.1 EDTV profile at the beginning of 2011.

7. ACKNOWLEDGMENTS

The authors would like to thank Dr. Marcelo Moreno and the TeleMídia Lab team who provided a thoughtful discussion of this work, tracked down and fixed problems in the initial new implementation of Ginga-NCL. The authors also thank CNPq, CAPES, MCT and CETIC/RNP for their support.

8. REFERENCES

[1] ABNT NBR Associação Brasileira de Normas Técnicas. *Digital Terrestrial Television Standard 06: Data Codification and Transmission Specifications for Digital Broadcasting, Part 2 – GINGA-NCL: XML Application Language for Application Coding* (São Paulo, SP, Brazil, November, 2007). http://www.abnt.org.br/imagens/Normalizacao_TV_Digital/ABNTNBR15606-2_2007Ing_2008.pdf

[2] Blackwell, A.F., Green, T.R.G.. Notational systems – the cognitive dimensions of notations framework. In: *Carroll, J.M. (Ed.), HCI Models, Theories and Frameworks: Toward a multidisciplinary science*. Morgan Kaufmann, San Francisco, pp. 103-134, 2003.

[3] Blackwell, A.F., 2006. Ten years of cognitive dimensions in visual languages and computing. *Journal of Visual Languages and Computing* 17 (4), 285-287.

[4] Clements, P. C. A Survey of Architecture Description Languages. In *8th International Workshop on Software Specifications & Design.* IEEE Computer Society, Washington, DC, USA, 1996.

[5] D.C. Muchaluat-Saade, R.F. Rodrigues, L.F.G. Soares. XConnector: Extending XLink to Provide Multimedia Synchronization. In *II ACM Symposium on Document Engineering – DocEng2002,* McLean, USA, 2002.

[6] ITUITU-T Recommendation H.761. *Nested Context Language (NCL) and Ginga-NCL for IPTV Services*. Geneva, April, 2009.

[7] Soares L.F.G., Rodrigues R.F. Nested Context Language 3.0 Part 8 – NCL Digital TV Profiles. *Technical Report. Informatics Department of PUC-Rio*, MCC 35/06. Rio de Janeiro, October, 2006. ISSN 0103-9741 http://www.ncl.org.br/documentos/NCL3.0-DTV.pdf.

[8] Soares, L.F.G.; Moreno, M.F.; Soares Neto, C.S.; Moreno, M.F. Ginga-NCL: Declarative Middleware for Multimedia IPTV Services. *IEEE Communications Magazine*. Vol. 48, No. 6, pp. 74-81. June, 2010. ISSN: 0163-6804.

[9] Soares Neto, C. S.; Souza, C. S.; Soares, L. F. G. Linguagens Computacionais como Interfaces: Um Estudo com Nested Context Language. In: Simpósio Brasileiro de Fatores Humanos em Sistemas Computacionais – IHC 2008, Porto Alegre, Brasil. Outubro de 2008.

[10] Soares Neto, C.S.; Soares, L.F.G. Reúso e Importação em Nested Context Language. *Anais do XV Simpósio Brasileiro de Sistemas Multimídia e Hipermídia*, Fortaleza, Ceará. Outubro de 2009; pp. 155-162. ISSN: 2175-9642.

[11] Soares Neto, C. S.; Soares, L. F. G.; Souza, C. S. *Journal of Brazilian Computer Science*. To be published.

APPENDIX A: Predefined Properties for Media Objects

Property name	Meaning
top, left, bottom, right, width, height	Screen positions
location	Screen positions
size	Screen size
bounds	Screen positions
baseDeviceRegion	A region on the screen
deviceClass	Class of secondary devices
plan	Graphical plan
explicitDur	Explicit duration of a media content
background	Background color
visible	Visibility control
transparency	Transparency level
rgbChromakey	RGB color for chromakey
fit	Way to fulfill a region
scroll	Scroll enabling
style	Reference to a style sheet
soundLevel, trebleLevel, bassLevel	Sound control
balanceLevel	Sound control
zIndex	Superposition index
fontColor	Font color
fontAlign	Font Align
fontFamily	Font Family
fontStyle	Font Style
fontSize	Font Size
fontVariant	Font Variant
fontWeight	Font Weight
player	Player identifier for media exhibition content
reusePlayer	Player life cycle control
playerLife	Player life cycle control
moveLeft, moveRight, moveUp, moveDown, focusIndex	Key navigation control properties
focusBorderColor;	Border color for media object in focus
selBorderColor	Border color for media object in selected

Property name	Meaning
focusBorderWidth	Border width for media object in focus
focusBorderTransparency	Border transparency for object in focus
focusSrc, focusSelSrc	Content to be display for object in focus
freeze	Control of the presentation end of a continuous content
transInType	Type of the transition-in
transInSubtype	Subtype of the transition-in
transInDur	Transition-in duration
transInStartProgress	Amount of progress through the transition at which to begin execution
transInEndProgress	Amount of progress through the transition at which to end execution
transInDirection	The direction the transition will run
transInFadeColor	Fade color for the transition-in
transInHorRepeat	Specifies how many times to perform the transition pattern along the horizontal axis
transInVertRepeat	Specifies how many times to perform the transition pattern along the vertical axis
transInBorderWidth	Specifies the border width along a wipe edge
transInBorderColor	Specifies the border color along a wipe edge
transOutType	Type of the transition-out
TransOutSubtype	Subtype of the transition-out
transOutDur	Transition-out duration
transOutStartProgress	Amount of progress through the transition at which to begin execution
transOutEndProgress	Amount of progress through the transition at which to end execution
transOutDirection	The direction the transition will run
transOutFadeColor	Fade color for the transition-out
transOutHorRepeat	Specifies how many times to perform the transition pattern along the horizontal axis
transOutVertRepeat	Specifies how many times to perform the transition pattern along the vertical axis
transOutBorderWidth	Specifies the border width along a wipe edge
transBorderColor	Specifies the border color along a wipe edge