Universidade Federal Fluminense
Escola de Engenharia
Curso de Engenharia de Telecomunicações
Programa de Educação Tutorial

Tutorial sobre a linguagem de programação NCL (*Nested Context Language*)

Autor: Erick Martins Ratamero (PET-Tele)
Orientador: Profa. Débora C. M. Saade (UFF/CTC/TCE/TET)

Niterói-RJ
Novembro / 2007

Capítulo 1

O que é NCL?

A linguagem NCL é uma linguagem do tipo declarativa, isto é, ela especifica de maneira imperativa o que deve ser feito, em oposição às linguagens procedurais, que descrevem como fazer alguma coisa. Como exemplos destes dois tipos de linguagens, temos HTML representando a abordagem declarativa, usada para descrever o conteúdo de uma página web, e Java ou C representando a abordagem procedural, usadas para especificar um algoritmo para executar determinada tarefa.

A NCL é baseado no modelo NCM (*Nested Context Model*, ou Modelo de Contextos Aninhados). Este modelo usa os conceitos de nós (*nodes*) e elos (*links*) para descrever documentos hipermídia (documentos que contêm diversos tipos de mídia, além de interação com o usuário). Podemos exemplificar este modelo pela Figura 1.

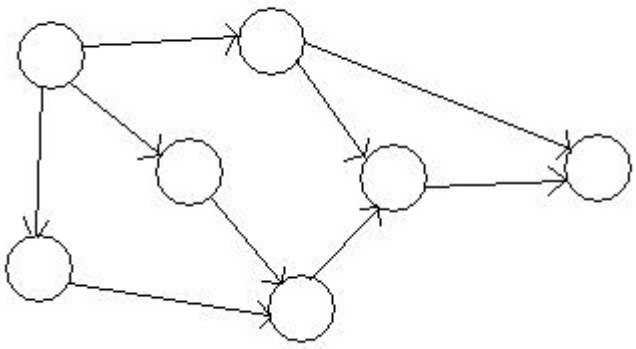

Figura 1.1: Estrutura de um documento hipermídia

Neste modelo, estes grafos podem ser aninhados, ou seja, cada nó pode ser, na verdade, um conjunto de nós e elos. Isto permite tornar a estrutura de um documento mais enxuta e organizada. Este aninhamento pode ser feito através de um tipo especial de nó, chamado de nó de composição (*composite node*) ou de contexto (*context node*). Assim, fica claro que podemos trabalhar com dois tipos de nós:

- nós de mídia: representam figuras, textos, vídeos e demais tipos de mídia;
- nós de contexto: representam um conjunto de nós e elos.

Podemos ver como estes nós se relacionam entre si na Figura 2.

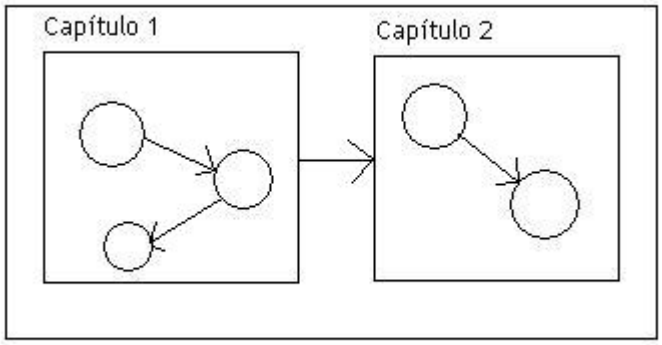

Figura 1.2: Estrutura de um documento NCM com composições

Para a autoria de documentos usando o modelo NCM, foi criada a linguagem NCL para a elaboração de documentos hipermídia. Um programa formatador é utilizado para interpretar um documento NCL e apresentar o conteúdo audiovisual interativo representado por ele.

Capítulo 2

Estrutura básica de um documento NCL

Um documento NCL deve possuir um cabeçalho de arquivo NCL, um cabeçalho do programa, o corpo do programa e o encerramento do documento. Os nós (sejam de mídia ou de contexto), elos e outros elementos que definem a estrutura e o conteúdo do programa são definidos na parte do corpo do programa.

Para definirmos por onde a apresentação do programa será iniciada, devemos criar portas. Estas portas servem como ponto de entrada em um documento ou nó de contexto. No momento da exibição do documento, devemos informar por que porta deseja-se iniciar a apresentação, propiciando assim que um único documento NCL tenha diversos pontos de entrada. Caso a porta de início não seja informada, o formatador usará uma porta-padrão que depende da implementação do mesmo.

A seguir, temos um exemplo de documento NCL:

```
1:  <?xml version="1.0" encoding="ISO-8859-1"?>
2:
3:  <ncl id="exemplo01"
 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/NCL23.xsd">
4:  <head>
5:       <regionBase>
6:          <!-- regiões da tela onde as mídias são apresentadas -->
7:       </regionBase>
8:    <descriptorBase>
9:       <!-- descritores que definem como as mídias são apresentadas
 -->
10:   </descriptorBase>
11:   <connectorBase>
12:      <!-- conectores que definem como os elos são ativados e o que
 eles disparam -->
13:   </connectorBase>
```

```
14: </head>
15: <body>
16:   <port id="pInicio" component="ncPrincipal" interface="iInicio"/>
17:      <!-- contextos, nós de mídia, elos e outros elementos -->
18: </body>
19: </ncl>
```

As linhas de 1 a 3 definem o cabeçalho de arquivo NCL. Da linha 4 à linha 14 temos o cabeçalho do programa. As linhas 15 a 18 contêm o corpo do programa, e a linha 19 o encerra.

Um passo-a-passo bastante prático para a construção de um documento NCL consiste em, primeiramente, escrever o cabeçalho de arquivo NCL, definir as regiões, conectores e descritores (*regionBase, connectorBase, descriptorBase* - serão explicados mais adiante no tutorial), definir os nós de conteúdo e de mídia, associando-os aos descritores, definir eventuais âncoras, definir a porta de entrada e as portas dos contextos e, finalmente, definir os elos.

2.1 Regiões

Uma região nada mais é do que uma área na tela (ou outro dispositivo de saída) onde será exibido um determinado nó de mídia. Estas regiões podem ser aninhadas (regiões dentro de regiões), tornando a estrutura mais organizada. Todas as regiões devem ser definidas no cabeçalho do programa (*regionBase*). Um exemplo de definição de regiões é o seguinte:

```
<region id="rgTV" width="1920" height="1080">
    <region id="rgVideo1" left="448" top="156" width="1024"
 height="768" />
</region>
```

Os atributos *height, width, left e top* definem a altura, largura, coordenada esquerda e coordenada superior da região. O atributo *id* dá um nome único a esta região, nome este que será referenciado, por exemplo, nos descritores das mídias associadas a esta região. Podemos definir ainda os atributos background, que atribui uma cor de fundo, e *zIndex*, que indica quais regiões aparecerão sobre quais no caso de regiões sobrepostas.

2.2 Descritores

Um descritor define como será apresentado um nó de mídia, incluindo em que região ele aparecerá. Os descritores devem ser definidos no cabeçalho do programa (*descriptorBase*). Um exemplo de descritor é o seguinte:

```
<descriptor id="dVideo1" region="rgVideo1" />
```

O atributo *id*, como nas regiões, dá um nome único a este descritor, que será referenciado quando da criação de um nó de mídia relacionado a este descritor. O atributo *region* associa uma região a este descritor. Além destes atributos, pode-se definir os atributos *player*, que diz qual a ferramenta de apresentação que será utilizada para mostrar nós de mídia associados a este descritor, e *explicitDur*, que diz qual será a duração temporal da apresentação dos nós de mídia relacionados a este descritor.

2.3 Portas

Portas servem para garantir acesso externo ao conteúdo de um contexto. Assim sendo, para que um elo aponte para um nó interno a um contexto, este nó deve apontar para uma porta que leve ao nó interno desejado.

Podemos ver todo o *body* do documento como um grande contexto. Assim, precisamos de uma porta de entrada que aponte para o primeiro nó de mídia ou contexto a ser apresentado quando da execução do documento. Um exemplo de definição de porta segue abaixo:

```
<port id="pInicio" component="video1" />
```

O atributo *id* atribui um nome único, pelo qual esta porta será referenciada sempre que necessário. O atributo *component* diz a qual nó de mídia ou contexto esta porta está associada. Há ainda o atributo *interface*. Este atributo indica a qual porta esta porta deve ser relacionada, no caso de *component* ser um nó de contexto, ou a qual âncora ela deve ser relacionada, caso *component* seja um nó de mídia.

2.4 Contextos

Os contextos servem para estruturar o documento NCL. Desta forma, eles podem ser aninhados, procurando sempre refletir a estrutura do documento e tornar a organização do programa mais intuitiva.

Define-se contexto da seguinte forma:

```
<context id="ctxNome">
...
</context>
```

O atributo *id*, como nos demais itens, define um nome único pelo qual o contexto será referenciado. Além deste, temos os atributos *descriptor*, que identifica qual descritor definirá a apresentação do contexto, e *refer*, que faz referência a outro contexto já definido, do qual este contexto herdará tudo menos o atributo *id*.

2.5 Nós de mídia

Um nó de mídia caracteriza o objeto de conteúdo propriamente dito, seja ele um vídeo, um texto, um áudio, etc. O nó de mídia deve identificar o arquivo com o conteúdo da mídia, além do descritor usado para regular a apresentação deste objeto de mídia. Um exemplo de nó de mídia é o seguinte:

```
<media type="video" id="video1" src="media/video1.mpg"
 descriptor="dVideo1"/>
```

O atributo *type* define de que tipo de mídia se trata: vídeo, áudio, texto, etc. O atributo *id* dá um nome único ao nó de mídia. O atributo *src* indica onde está o arquivo-fonte daquele nó de mídia, e o atributo *descriptor* indica qual descritor será usado para a execução daquele objeto. Outro atributo que pode ser utilizado é o *refer*, que referencia um outro nó de mídia do qual este nó usará todos os atributos, menos o *id*.

Capítulo 3

Elos e conectores

Os nós de mídia ou contexto possuem relações entre si, caracterizando um documento hipermídia. Estas relações podem ser de diversos tipos. Desta forma, surge o conceito de conectores, objetivando definir a relação semântica contida nos elos em NCL.

Assim sendo, o conector (do tipo causal) serve para definir a relação entre um ou mais nós de origem (chamados assim por ativar o elo) e um ou mais nós de destino (que serão afetados pela ativação do elo). Um conjunto de conectores já foi bem-definido pela grupo de trabalho da PUC-RJ que desenvolveu a linguagem NCL.

Geralmente, um arquivo externo contendo todos os conectores (chamado de base de conectores) pode ser importado pelo documento NCL. Assim, o autor de um documento hipermídia deste tipo não precisa se preocupar com a criação e desenvolvimento de conectores se ele não tiver esta necessidade. Para importar uma base de conectores, podemos incluir o seguinte código no cabeçalho de programa:

```
<connectorBase>
       <importBase alias="connectors" baseURI="connectorBase.ncl" />
</connectorBase>
```

O parâmetro *alias* dá um 'àpelido" à base de conectores. Este *alias* será referenciado quando da criação de elos. O parâmetro *baseURI* diz ao documento NCL onde ele deve procurar pela base de conectores a ser importada. Neste caso, a base será um arquivo chamado *"connectorBase.ncl"*, contido no mesmo diretório do documento NCL.

Um elo é criado utilizando-se os conectores e aplicando nós de mídia ou contexto a papéis (ou *roles*) estabelecidos pelo conector. Vejamos um exemplo:

```
    <link id="lVideo1Titulo1Start"
 xconnector="connectors#onBegin1StartN">
      <bind component="video1" role="onBegin" />
      <bind component="titulo1" role="start" />
    </link>
```

(visão estrutural, temporal, etc)

O atributo *id* dá um nome único ao elo estabelecido. O parâmetro *xconnector* indica qual será o conector utilizado. Ele pode ser usado de duas formas: no caso de uma base de conectores importada, dá-se o *alias* do conector e o nome do conector após o *sharp*

(#). Neste caso, a base de conectores tem como *alias "connectors"* e o conector usado é o *"onBegin1StartN"*. Como o próprio nome indica, este conector liga vários nós de forma que, quando um determinado nó começa, diversos outros são iniciados simultaneamente.

Para indicar quais nós desempenharão os "papéis" determinados pelo conector, usa-se estruturas do tipo *bind*. Nestas estruturas, o parâmetro *component* deve indicar a *id* de um nó (no caso de um nó de contexto, deve-se ainda especificar uma porta usando o atributo *interface*), enquanto o parâmetro *role* deve indicar qual será o papel que este nó terá na execução deste elo. No exemplo dado, o nó video1 tem o papel *onBegin* e o nó titulo1 tem o papel *start*. Assim sendo, quando a execução do nó video1 for iniciada, este nó será disparado, iniciando assim a execução do nó titulo1.

Dois tipos de conectores podem ser definidos: conectores causais e conectores de restrição.

Os conectores causais definem dois ou mais papéis. Estes papéis indicarão quais as condições sob as quais o elo será ativado e que ações serão efetuadas quando da ativação deste elo. A seguir, é dado um exemplo de conector causal:

```
<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>
```

O atributo *id* dá um nome ao conector. Em seguida, criam-se os papéis: um *conditionRole* estabelece uma condição sob a qual o elo que utilize este conector será ativado. Vê-se que foi criado um papel *"onBegin"*, equivalente à transição *"starts"* em um evento do tipo *"presentation"*. Ou seja, isto significa que o elo será ativado quando a apresentação do nó que ocupa este papel for iniciada.

Logo depois, cria-se um papel de nome *"start"*. Ele é um papel do tipo *actionRole*, o que significa que ele corresponde a uma ação que será efetuada por um elo que utilize este conector. Este papel corresponde a uma ação *"start"* no evento do tipo *"presentation"*. Isto é, quando o elo for ativado, a apresentação do nó que ocupa este papel será iniciada.

Por fim, é preciso mostrar claramente qual é a lógica que liga os papéis: para isso, criamos o campo *causalGlue*. Este campo indicará qual é a "cola" que liga os papéis criados anteriormente. Associa-se o *conditionRole* ao *"Trigger"* do evento, e o *actionRole* ao *"Action"*. Assim, fica claro que o *conditionRole* será o gatilho do elo, enquanto o *actionRole* será a ação a ser executada pelo elo.

Capítulo 4

Âncoras

Uma âncora é um ponto de entrada em um nó, seja ele de mídia ou de contexto. Quando se usa âncoras, objetiva-se usar um segmento de um nó como origem ou destino de um elo. Há dois tipos de âncoras: âncora de conteúdo e âncora de atributo.

Uma âncora de conteúdo delimita uma parte da mídia a ser utilizada como ativadora de um elo. Assim sendo, ela consiste em uma seleção de um determinado "pedaço" de uma mídia. Por exemplo, podemos usar uma certa área de uma imagem, um certo trecho de um áudio ou vídeo, etc. A âncora é definida como um elemento area dentro de um nó de mídia. Um exemplo é dado a seguir:

```
<media type="video" id="video1" src="media/video1.mpg"
 descriptor="dVideo1">
   <!-- âncoras de conteúdo no vídeo que devem ser sincronizadas
 com a legenda -->
   <area id="aVideoLegenda01" begin="5s" end="9s"/>
   <area id="aVideoLegenda02" begin="10s" end="14s"/>
   <area id="aVideoLegenda03" begin="15s" end="19s"/>
</media>
```

O atributo *id* dá um identificador à âncora. Além do *id*, a âncora deve delimitar a qual parte da mídia ela se refere. Assim, há atributos como *coords, begin, end, dur, first, last, text e position*, todos eles servindo ao mesmo objetivo: delimitar uma parte de uma imagem (no caso de *coords*), de um áudio ou vídeo (no caso de *begin, end, dur, first e last*) ou de um texto (*text e position*).

Já as âncoras de atributo referem-se a determinados atributos de nós que serão manipulados por elos. Estes atributos podem ser, por exemplo, o volume do som em um áudio ou vídeo, dimensões de exibição de uma imagem, etc. Estas âncoras devem ser definidas como elementos do tipo *attribute* dentro de um nó de mídia ou contexto.