3
Linguagem NCL versão 2.0

A linguagem NCL – *Nested Context Language* – é uma linguagem declarativa para autoria de documentos hipermídia baseados no modelo conceitual NCM – *Nested Context Model*. A primeira versão de NCL (Antonacci, 2000a; Antonacci, 2000b) foi especificada através de uma DTD – *Document Type Definition* – XML (W3C, 1998a).

A nova versão de NCL, batizada de NCL 2.0, foi especificada através de XML Schema (W3C, 2001b), uma recomendação mais recente do W3C, que oferece uma linguagem mais rica para a definição de tipos de documentos.

Seguindo padrões atuais, NCL 2.0 foi especificada de forma modular, permitindo a combinação de seus módulos em perfis de linguagem. Cada perfil pode agrupar um subconjunto de módulos de NCL 2.0, possibilitando a criação de linguagens de acordo com a necessidade dos usuários. Além disso, módulos de NCL 2.0 podem ser combinados com módulos de outras linguagens, permitindo a incorporação de características de NCL 2.0 a essas linguagens e vice-versa. Essa abordagem modular para a especificação de linguagens foi adotada nas especificações mais recentes do W3C, como por exemplo em XHTML 1.1 (W3C, 2001a) e SMIL 2.0 (W3C, 2001d).

Comparando NCL 2.0 com a sua versão anterior, além da estrutura modular, NCL 2.0 introduz novas facilidades na linguagem de autoria. Dentre as facilidades, pode-se ressaltar:

- a definição de conectores hipermídia e de bases de conectores;
- o uso de conectores hipermídia para a autoria de elos;
- a definição de bases de elos;
- o reuso de elos e bases de elos em diferentes documentos;
- a definição de portas e mapeamentos para nós de composição, satisfazendo a propriedade de composicionalidade dos documentos;

- a definição de templates de composição hipermídia, permitindo a especificação de restrições em documentos;
- a definição de bases de templates de composição;
- o uso de templates de composição para a autoria de nós de composição;
- o refinamento da especificação de documentos com alternativas de conteúdo, através do elemento *switch*, que agrupa um conjunto de nós alternativos;
- o refinamento da especificação de documentos com alternativas de apresentação, através do elemento *descriptorSwitch*, que agrupa um conjunto de descritores alternativos, como será detalhado mais adiante;
- o uso de um novo modelo de layout espacial (Moura, 2001), que possibilita especificar informações para posicionamento de objetos em um dispositivo de saída.

Antes de apresentar os módulos de NCL 2.0, a próxima seção realiza um breve resumo sobre modelagem de documentos usando o NCM, modelo conceitual no qual NCL se baseia. Esse resumo introduz as entidades principais do modelo NCM, que vão se refletir em elementos da linguagem NCL. Para obter mais detalhes sobre o modelo NCM, o leitor deve consultar a referência (Soares, 2003).

3.1 Documentos Hipermídia no Modelo NCM

Um documento hipermídia no modelo NCM é representado por um nó de composição, podendo conter um conjunto de nós, que podem ser objetos de mídia ou outros nós de composição, recursivamente, e ainda elos relacionando esses nós. Uma restrição do modelo impede que um nó contenha, recursivamente, ele próprio.

Nós de composição no modelo NCM não têm nenhuma semântica embutida, diferente das composições SMIL, por exemplo, que têm semântica temporal. A semântica de um nó de composição é deixada para linguagens de autoria baseadas no modelo. A semântica dos relacionamentos entre os componentes de um documento é feita através dos elos NCM, que podem especificar relações de

referência, relações de sincronização, relações de derivação, relações entre tarefas de um trabalho cooperativo etc. Elos, na nova versão do modelo NCM, são definidos fazendo referência a um conector hipermídia, que pode representar qualquer tipo de relação. Além da referência a um conector, um elo define um conjunto de *binds* associando papéis do conector a nós do documento. Assim sendo, elos podem representar relacionamentos multiponto entre vários nós de um documento.

Para permitir a criação de relacionamentos entre partes internas do conteúdo de um nó, o modelo permite a definição de pontos de interface dos nós, que definem âncoras ou atributos, para qualquer tipo de nó, ou portas, no caso de um nó de composição (veja Seção 2.4). Sendo assim, cada extremidade de um elo indica um ponto de interface de um nó.

A versão anterior do modelo (Soares, 2000) especificava uma outra restrição que exigia que todas as extremidades de um elo fossem componentes da composição onde o elo foi definido. Entretanto, na nova versão (Soares, 2003), essa restrição é deixada para linguagens de autoria baseadas no modelo, permitindo que o NCM também possa ser usado para modelar documentos que não têm esse tipo de restrição, como, por exemplo, documentos web. Vale ressaltar no entanto que, se essa restrição não for obedecida, a propriedade de composicionalidade é perdida.

Na nova versão do NCM, elos são sempre agrupados em bases de elos, logo o conjunto de elos de uma composição é dado pela união de conjuntos representados por bases de elos. Além do reuso de nós, permitido em versões anteriores do NCM (Soares, 1995; Soares, 2000), a nova versão também permite o reuso de elos e de bases de elos em outras bases, tornando mais flexível a definição de elos no modelo.

Uma outra entidade introduzida por essa nova versão do modelo NCM é um nó que agrupa um conjunto de nós alternativos, denominado nó *switch*, tal como em SMIL, que pode ser usado para a especificação de documentos hipermídia adaptáveis a uma plataforma de exibição ou a um perfil de usuário específico. Um nó *switch* simplifica a idéia de utilizar entidades virtuais, mais gerais, como comentado em versões anteriores do modelo (Soares, 2000a). Um nó *switch* pode definir tipos especiais de pontos de interface, que permitem ser mapeados em um

conjunto de pontos de interface alternativos de seus nós componentes, onde a escolha pode ser feita em tempo de execução. Esse tipo especial de ponto de interface permite a ancoragem de elos no componente que será escolhido pela execução de um nó *switch*.

A especificação de apresentação de um nó no modelo NCM é feita de forma independente da definição do nó, sendo representada por outro objeto do modelo, chamado descritor (Soares, 2000). Com isso, pode-se definir diferentes especificações de apresentação para um mesmo nó, associando-se diferentes descritores a esse nó. Um nó pode ter um descritor definido por seu tipo (áudio, vídeo, texto, imagem, composição etc.), um outro descritor definido pelo próprio nó (normalmente especificado pelo seu autor), um outro descritor definido por cada composição a que pertence, um outro descritor definido por cada elo que o toca e ainda, um descritor definido pelo usuário final (leitor). No caso de um nó possuir todos esses descritores especificados, o descritor resultante a ser associado a esse nó, para especificar sua apresentação final, será a combinação das especificações de todos esses descritores, de acordo como seguinte cascateamento: atributos do descritor definido pelo usuário sobrescrevem atributos do descritor definido pelo elo, que, por sua vez, sobrescrevem atributos do descritor definido pela composição, que, por sua vez, sobrescrevem atributos do descritor definido pelo próprio nó, que, por último, sobrescrevem atributos do descritor da classe do nó. Além disso, visando a especificação de apresentações multimídia adaptativas, cada especificação de apresentação pode, na verdade, ao invés de especificar um único descritor, especificar um conjunto de descritores alternativos, apresentando opções de apresentação para o mesmo nó. Dentre as opções especificadas, uma delas será escolhida pelo formatador de documentos (Rodrigues, 2003), dependendo de características da plataforma de exibição e do usuário final.

Como mencionado anteriormente, um nó *switch* pode especificar alternativas de componentes para um documento, sendo a escolha feita pelo formatador de documentos em tempo de execução. Além disso, um nó *switch* também pode especificar alternativas de apresentação para cada um de seus componentes, através de conjuntos de descritores alternativos, de forma análoga ao que um nó de composição qualquer pode realizar.

Utilizando as entidades básicas do modelo NCM, que são nó objeto de mídia, nó de composição, nó *switch*, conector, elo, base de elos e descritor, a linguagem NCL 2.0 foi desenvolvida, introduzindo ainda outros recursos para facilitar a autoria de documentos NCM, como por exemplo, o conceito de template de composição hipermídia, abordado anteriormente na Seção 2.5. A próxima seção apresenta a estrutura modular de NCL 2.0.

3.2 Modularização da linguagem NCL 2.0

Seguindo a mesma abordagem de modularização da linguagem SMIL 2.0, a funcionalidade de NCL 2.0 está particionada em onze áreas funcionais, que, por sua vez, estão particionadas novamente em módulos. Como NCL 2.0 tem algumas facilidades idênticas às de SMIL 2.0, alguns módulos de SMIL 2.0 são usados por NCL 2.0.

As áreas funcionais de NCL 2.0 e seus módulos correspondentes são os seguintes:

1. *Structure*
    a. *Structure Module*
2. *Components*
    a. *BasicMedia Module* (idêntico ao *BasicMedia Module* de SMIL 2.0)
    b. *BasicComposite Module*
3. *Interfaces*
    a. *MediaInterface Module*
    b. *CompositeInterface Module*
    c. *AttributeInterface Module*
    d. *SwitchInterface Module*
4. *Linking*
    a. *Linking Module*
5. *Connectors*
    a. *XConnector Module*
    b. *CompositeConnector Module*
6. *Composite Templates*

a. *XTemplate Module*
b. *XTemplateUse Module*

7. *Timing*
   a. *BasicTiming Module*
8. *Layout*
   a. *BasicLayout Module*
9. *Presentation Specification*
   a. *BasicDescriptor Module*
   b. *CompositeDescriptor Module*
10. *Presentation Control*
   a. *TestAttributes Module*
   b. *ContentControl Module* (similar ao *BasicContentControl Module* de SMIL 2.0)
   c. *DescriptorControl Module*
11. *Metainformation*
   a. *Metainformation Module* (idêntico ao *Metainformation Module* de SMIL 2.0)

As próximas subseções descrevem, de maneira sucinta, as principais definições feitas por cada área funcional da linguagem e seus módulos correspondentes. A definição completa dos módulos de NCL 2.0, usando XML Schema, está disponível no Apêndice D.

No fim deste capítulo, são comentadas algumas possibilidades de combinação de módulos de NCL 2.0, gerando diferentes perfis de linguagem, além do perfil completo, denominado NCL 2.0 *Language Profile*.

3.2.1 Área Funcional *Structure*

A área funcional *Structure* contém apenas um módulo, chamado *Structure*, que define a estrutura básica de um documento NCL. São definidos o elemento raiz, chamado *ncl*, o cabeçalho e o corpo do documento, chamados respectivamente de *head* e *body*, seguindo a terminologia adotada por outras linguagens padronizadas pelo W3C. O corpo de um documento NCL pode ser tratado como um nó de composição, contendo outros nós e/ou bases de elos.

3.2.2
Área Funcional *Components*

A área funcional *Components* contém dois módulos, chamados *BasicMedia* e *BasicComposite*.

O módulo *BasicMedia* define os tipos básicos de objetos de mídia que um documento NCL pode conter. Esse módulo é idêntico ao módulo *BasicMedia* do SMIL, definido os seguintes elementos *animation*, *audio*, *img*, *text*, *textstream*, *video* e *ref*. O elemento *ref* permite definir uma referência a um outro objeto, permitindo o reuso de objetos em diferentes documentos. Cada objeto de mídia possui dois atributos principais: o atributo *src* que define uma URI (Berners-Lee, 1998) com o conteúdo do objeto e o atributo *type*, que define o tipo MIME (Freed, 1996) do objeto, além do atributo *id*.

O módulo *BasicComposite* é responsável pela definição de nós de composição em documentos NCL. Um nó de composição no modelo NCM, como já mencionado na Seção 3.1, é definido por um conjunto de nós, que podem ser de composição recursivamente, e de bases de elos. Assim sendo, o elemento *composite*, que representa o nó de composição, pode conter elementos definidos pelo módulo *BasicMedia*, outros elementos *composite* e ainda elementos *linkBase*, que agrupam a definição dos elos da composição. O conteúdo de elementos *linkBase* será definido na Seção 3.2.4, que aborda o módulo *Linking*.

O uso do módulo *BasicComposite*, por um determinado perfil de linguagem, exige o uso dos módulos *BasicMedia* e *Linking*, para contemplar a definição de objetos de mídia e bases de elos.

3.2.3
Área Funcional *Interfaces*

A área funcional *Interfaces* permite a definição de pontos de interface de nós que serão usados para a criação de elos. Essa área funcional está particionada em quatro módulos, um chamado *MediaInterface*, que permite a definição de âncoras para objetos de mídia, outro chamado *CompositeInterface*, que permite a definição de âncoras e portas para nós de composição, um terceiro chamado *AttributeInterface*, que permite a definição de atributos de nós como pontos de interface, e um último módulo chamado *SwitchInterface*, que permite a definição

de pontos de interface especiais para elementos *switch*, que serão comentados na Seção 3.2.10.

O módulo *MediaInterface* define o elemento *area*, que estende a sintaxe e semântica do elemento homônimo definido por SMIL e XHTML. Além de permitir a definição de âncoras representando porções espaciais, através do atributo *coords* (como em XHTML), e a definição de âncoras representando porções temporais, através dos atributos *begin*, *end* e *dur*, como em SMIL, o elemento *area* permite a definição de âncoras textuais, através do atributo *text*, que define a cadeia de caracteres, e do atributo *position*, que define a posição inicial da cadeia no texto. Além das âncoras textuais, o módulo *MediaInterface* permite a definição de âncoras baseadas em número de amostras, para mídia áudio, ou número de quadros, para mídia vídeo, através dos atributos *first* e *last*, que devem indicar o número da amostra/quadro inicial e o final. Um perfil de linguagem que usa esse módulo deve acrescentar o elemento *area* como elemento-filho dos elementos que definem objetos de mídia, especificados pelo módulo *BasicMedia*.

O uso do módulo *MediaInterface*, por um determinado perfil de linguagem, exige o uso do módulo *BasicMedia*, para contemplar a definição de objetos de mídia.

O segundo módulo da área funcional *Interfaces* é o módulo *CompositeInterface*, que define um elemento chamado *areaComposite*, que pode ser usado para a criação de âncoras em nós de composição. O elemento *areaComposite* tem um atributo chamado *componentList*, que contém uma lista de identificadores de nós componentes do nó de composição. Além do elemento *areaComposite*, o módulo *CompositeInterface* também especifica o elemento *port*, que define uma porta da composição com seu respectivo mapeamento para um ponto de interface (atributo *port*) de um de seus nós componentes (atributo *component*). Um perfil de linguagem que usa esse módulo deve acrescentar os elementos *areaComposite* e *port* como elementos-filho do elemento *composite*, especificado pelo módulo *BasicComposite*.

O uso do módulo *CompositeInterface*, por um determinado perfil de linguagem, exige o uso do módulo *BasicComposite*, para contemplar a definição de nós de composição.

O terceiro módulo da área funcional *Interfaces* é o módulo *AttributeInterface*, que define um elemento chamado *attribute*, que pode ser usado para definir um atributo de um nó como um de seus pontos de interface. O elemento *attribute* tem um atributo chamado *name*, que indica o nome do atributo em questão. Um perfil de linguagem que usa esse módulo deve acrescentar o elemento *attribute* como elemento-filho dos elementos que definem objetos de mídia, especificados pelo módulo *BasicMedia*, e do elemento *composite*, especificado pelo módulo *BasicComposite*.

O uso do módulo *AttributeInterface*, por um determinado perfil de linguagem, exige o uso do módulo *BasicMedia* ou *BasicComposite*, para contemplar a definição de nós.

O módulo *SwitchInterface* permite a especificação de pontos de interface de um elemento *switch*, que podem ser mapeados em um conjunto de alternativas de pontos de interface de seus nós internos, permitindo a ancoragem de elos no componente escolhido durante a exibição de um nó *switch* (veja Seção 3.1). Esse módulo introduz o elemento *portSwitch*, que contém um conjunto de elementos *port*, de acordo com a definição feita pelo módulo *CompositeInterface*, mencionada anteriormente. Um perfil de linguagem que usa esse módulo deve acrescentar os elementos *portSwitch* e *port* como elementos-filho de um elemento *switch*, especificado pelo módulo *ContentControl*, descrito na Seção 3.2.10.

A definição de elementos *port* e *portSwitch* introduzidos pelos módulos *CompositeInterface* e *SwitchInterface* são novas facilidades que NCL 2.0 oferece.

3.2.4
Área Funcional *Linking*

A área funcional *Linking* define o módulo *Linking*, responsável pela definição de elementos para a criação de elos usando conectores. Um elo é definido pelo elemento *link*, que possui, além do atributo *id*, um atributo chamado *xconnector*, que faz referência à URI de um conector hipermídia. A definição de um conector hipermídia será feita em detalhes no Capítulo 4. Se o conector referenciado definir parâmetros, o elo deve associar valores para esses parâmetros através de elementos-filho *param* com atributos nome (*name*) e valor (*value*), seguindo as definições de tipo dos parâmetros feitas pelo conector. O elemento

link contém ainda elementos-filho chamados *bind*, que permitem associar os nós relacionados pelo elo a papéis do conector usado (veja Seção 2.4). Para realizar tal associação um elemento *bind* possui três atributos básicos. O primeiro é chamado *role*, sendo usado para fazer referência a um papel do conector usado. O segundo é chamado *component*, que deve fazer referência ao identificador único de um nó, e o terceiro é chamado *port*[8], usado para fazer referência a um ponto de interface do nó associado.

Além do elemento *link*, o módulo *Linking* define um outro elemento chamado *linkBase*, que permite a definição de bases de elos. Além disso, esse módulo ainda especifica o elemento *lref*, que pode ser usado para fazer referência a um elo ou uma base de elos já definidos, permitindo o reuso de elos e bases de elos em diferentes documentos. O elemento *lref*, análogo ao elemento *ref*, definido pelo módulo *BasicMedia*, tem um atributo *src* que identifica o elemento sendo reusado. Um elemento *linkBase* pode conter elementos *link* ou *lref*.

A definição de bases de elos e o reuso de elos introduzidos pelo módulo *Linking* são duas novas facilidades que NCL 2.0 oferece.

O uso do módulo *Linking*, por um determinado perfil de linguagem, exige que um módulo que define conectores, tal como *XConnector*, seja utilizado também, pois a semântica de um elo é dada pela definição de um conector, como será detalhado no Capítulo 4.

3.2.5 Área Funcional *Connectors*

A área funcional *Connectors* é uma nova facilidade, que NCL 2.0 oferece, não encontrada em nenhuma outra linguagem hipermídia. Essa área funcional tem dois módulos, um chamado *XConnector* e outro chamado *CompositeConnector*.

O módulo *XConnector* permite a definição de conectores hipermídia, através do elemento *xconnector*, e de bases de conectores, através do elemento *connectorBase*. O módulo *XConnector* será descrito em detalhes no Capítulo 4.

8 Note que o atributo *port* em um elemento *bind*, pode fazer referência a qualquer ponto de interface de um nó, ou seja, uma âncora, um atributo ou uma porta, no caso de um nó de composição.

O módulo *CompositeConnector* permite a definição de conectores compostos, tal como discutido na Seção 2.4. Um conector composto é definido pelo elemento *compositeConnector*, que contém elementos *role* e um elemento *glue*, tal como um conector simples (veja Capítulo 4). Cada elemento *role* representa um ponto de interface do conector composto, que é mapeado em um ponto de interface de uma ocorrência de um conector interno (chamado de elo parcialmente definido, de acordo com a Seção 2.4). Sendo assim, cada elemento *role* tem um atributo *id*, um atributo chamado *partialLink* e outro chamado *role*, que fazem referência, respectivamente, a um elo parcialmente definido, interno ao conector, e um papel do conector usado por esse elo. Além do conjunto de elementos *role*, um conector composto especifica um elemento *glue*, como um nó de composição, contendo o mesmo conteúdo que um elemento *composite* e ainda um conjunto de elos parcialmente definidos, chamado de *partialLinkBase*. O elemento *partialLinkBase* contém elementos *link* somente.

O uso do módulo *CompositeConnector*, por um determinado perfil de linguagem, exige o uso do módulo *BasicComposite*, para contemplar a definição de nós de composição, que já contemplam a definição de objetos de mídia e bases de elos.

3.2.6 Área Funcional *Composite Templates*

A área funcional *CompositeTemplates* também é uma nova facilidade, que NCL 2.0 oferece, não encontrada em nenhuma outra linguagem hipermídia. Essa área funcional tem dois módulos, um chamado *XTemplate* e outro chamado *XTemplateUse*.

O módulo *XTemplate* permite a definição de templates de composição hipermídia, através do elemento *xtemplate* e de bases de templates, através do elemento *templateBase*. O módulo *XTemplate* é descrito em detalhes no Capítulo 5.

O uso do módulo *XTemplate*, por um determinado perfil de linguagem, exige que um módulo que define conectores, tal como *XConnector*, seja utilizado também, pois a semântica de elos definidos em um template de composição é dada pela definição de conectores hipermídia.

O módulo *XTemplateUse* permite a utilização de templates por nós de composição em documentos NCL. Para isso, esse módulo define o atributo *xtemplate*, que faz referência à URI de um template de composição e define o atributo *label*, que especifica um rótulo. Um perfil de linguagem que use esse módulo deve acrescentar o atributo *xtemplate* à definição do elemento *composite* e o atributo *label* à definição de objetos de mídia, de elementos *composite* e de pontos de interface de nós. Por esses motivos, o uso do módulo *XTemplateUse*, por um determinado perfil de linguagem, exige o uso dos módulos *BasicMedia*, *BasicComposite*, *MediaInterface* e *CompositeInterface*.

3.2.7 Área Funcional *Timing*

A área funcional *Timing* define um único módulo chamado *BasicTiming*, que permite a definição de especificações temporais para componentes de um documento. Basicamente, esse módulo define atributos para especificação da duração ideal (*dur*), duração mínima (*min*) e duração máxima (*max*) de um objeto qualquer, que serão utilizados por descritores, que, por sua vez, agrupam toda a especificação de apresentação de componentes do documento, como será definido na Seção 3.2.9.

Essa área funcional pode incluir um outro módulo que permite a especificação de funções de custo para informar ao formatador de documentos a melhor forma de realizar ajustes nas durações do objetos, caso seja necessário realizar esse tipo de adaptação durante a apresentação (Rodrigues, 2003). Um função de custo poderia ser associada à especificação temporal de cada objeto. Entretanto, a definição desse outro módulo foi deixada para trabalho futuro.

3.2.8 Área Funcional *Layout*

A área funcional *Layout* define um único módulo, chamado *BasicLayout*, que especifica elementos e atributos que definem como os objetos serão apresentados em regiões na tela de um dispositivo de exibição.

O modelo de layout usado por NCL 2.0 é apresentado em detalhes em (Moura, 2001). Em resumo, um elemento *layout*, que deve ser declarado no

cabeçalho de um documento, define um conjunto de janelas (*topLayout*), que contêm um conjunto de regiões (*region*), que, por sua vez, podem conter outro conjunto de regiões aninhadas. Tanto janelas quanto regiões podem definir os seguintes atributos: *id*, *title*, *left*, *top*, *height*, *width*, *backgroundColor*, *zIndex*, *scroll*, *open*, *close*. Além disso, regiões também podem definir o atributo *fit*, indicando como um objeto deve ser apresentado na região indicada. Todos esses atributos têm o mesmo significado e os mesmos valores possíveis dos atributos homônimos definidos nos módulos de layout da linguagem SMIL 2.0, exceto o atributo *scroll*, que permite ao usuário especificar como deseja configurar o *scroll* de uma região (Rodrigues, 2003).

Apesar de NCL definir seu modelo de layout, nada impede que um documento NCL possa utilizar outros modelos de layout, desde que eles definam regiões onde objetos devam ser exibidos, como por exemplo os modelos de layout definidos por SMIL 2.0. Sendo assim, o elemento *layout* pode especificar um atributo *src*, que indica a URI do documento contendo a especificação de layout, e não precisa conter, nesse caso, conteúdo algum.

3.2.9 Área Funcional *Presentation Specification*

A área funcional *Presentation Specification* define dois módulos, um chamado *BasicDescriptor* e outro chamado *CompositeDescriptor*. O objetivo desses módulos é especificar as informações temporais e espaciais necessárias para apresentar cada componente de um documento, modeladas, de acordo com o NCM, por objetos chamados descritores.

O módulo *BasicDescriptor* permite a definição do elemento *descriptor*, que contém um conjunto de atributos opcionais, agrupando todas as definições temporais e espaciais, que devem ser utilizados de acordo com o tipo do objeto a ser apresentado. A definição de elementos *descriptor* deve ser feita no cabeçalho de um documento (*head*), dentro do elemento *descriptorBase*, que especifica o conjunto de descritores do documento.

Um elemento *descriptor* possui os atributos temporais *dur*, *min* e *max* definidos pelo módulo *BasicTiming* (veja Seção 3.2.7); um atributo chamado *player*, que identifica a ferramenta de exibição a ser utilizada; um atributo

chamado *region*, que faz referência a uma região definida por elementos do módulo *BasicLayout* (veja Seção 3.2.8); e ainda um atributo chamado *enableTimeBar*, usado para habilitar uma barra de rolagem de tempo, caso se aplique. Além desses atributos, um elemento *descriptor* também pode definir o atributo *style*, que faz referência a uma folha de estilo (W3C, 1998b), com informações para apresentação de textos, por exemplo; e ainda atributos específicos para objetos com áudio, definindo *soundLevel*, *balanceLevel*, *trebleLevel* e *bassLevel*.

O uso do módulo *BasicDescriptor*, por um determinado perfil de linguagem, exige o uso dos módulos *BasicTiming* e *BasicLayout*.

O módulo *BasicDescriptor*, além do elemento *descriptor*, define um atributo homônimo, que faz referência a um elemento do conjunto de descritores do documento. Um perfil de linguagem, que use o módulo *BasicDescriptor*, pode determinar como é feita a associação do atributo *descriptor* a um componente do documento. No caso do perfil NCL 2.0 *Language Profile*, seguindo o modelo NCM, o atributo *descriptor* é associado a um objeto de mídia qualquer, um nó de composição, ou ainda a uma extremidade de um elo (veja Seção 3.1).

O módulo *CompositeDescriptor* define o elemento *componentPresentation*, que associa um nó componente de uma composição, identificado pelo atributo *component*, a um elemento do conjunto de descritores do documento, identificado pelo atributo *descriptor*. Qualquer elemento *composite* passa, então, a poder conter elementos *componentPresentation*. O uso do módulo *CompositeDescriptor*, por um determinado perfil de linguagem, exige o uso dos módulos *BasicDescriptor* e *BasicComposite*.

Vale ressaltar que o conjunto de descritores de um documento pode conter elementos *descriptor* ou ainda elementos *descriptorSwitch*, que permitem especificar descritores alternativos, como será apresentado na Seção 3.2.10.

3.2.10 Área Funcional *Presentation Control*

Os objetivos da área funcional *Presentation Control* são similares aos da área funcional *Content Control* da linguagem SMIL 2.0, ou seja, especificar alternativas de conteúdo, e ainda alternativas de exibição para um documento.

Essa área funcional define três módulos, um chamado *TestAttributes*, outro chamado *ContentControl*, e um último módulo chamado *DescriptorControl*.

O módulo *TestAttributes* permite a definição de atributos de teste usados para especificar alternativas para a apresentação de um documento. Os atributos de teste são exatamente os mesmos pré-definidos pelo módulo *BasicContentControl* de SMIL 2.0 (W3C, 2001d). A especificação dos atributos de teste foi feita em um módulo separado de NCL, pois será útil tanto para definir componentes alternativos, como para definir descritores alternativos em documentos NCL, como apresentado a seguir.

O módulo *ContentControl* especifica o elemento *switch*, tal como em SMIL, permitindo a definição de nós componentes alternativos do documento, a serem escolhidos em tempo de execução. Entretanto, o elemento *switch* de NCL não tem conteúdo idêntico ao elemento homônimo de SMIL (definido pelo seu módulo *BasicContentControl*), pois pode conter, além de objetos de mídia, elementos *composite*, definidos pelo módulo *BasicComposite* de NCL 2.0. Os atributos de teste para a escolha do componente do *switch* a ser exibido são os definidos pelo módulo *TestAttributes*, ou seja, os mesmos de SMIL 2.0. Por essa razão, o uso do módulo *ContentControl*, por um determinado perfil de linguagem, exige o uso do módulo *TestAttributes*, além dos módulos *BasicMedia* e *BasicComposite*, para contemplar a definição de objetos de mídia e nós de composição. Para permitir a ancoragem de elos no componente selecionado na execução de um nó *switch*, um perfil de linguagem deve, também, incluir o módulo *SwitchInterface*, que possibilita a definição de pontos de interface especiais para elementos *switch*, chamados de *portSwitch*, como descrito na Seção 3.2.3. Além disso, para permitir a definição de alternativas de apresentação para componentes de um nó *switch*, um perfil de linguagem deve, ainda, incluir o módulo *CompositeDescriptor*, que introduz o elemento *componentPresentation*, permitindo sua inclusão em um nó *switch*, de forma análoga ao que é feito por um nó de composição qualquer (veja Seção 3.2.9).

O módulo *DescriptorControl* especifica o elemento *descriptorSwitch*, que contém um conjunto de descritores alternativos a serem associados a um objeto. De forma análoga ao elemento *switch*, a escolha do descritor a ser usado é feita em tempo de execução, usando os mesmo atributos de teste, definidos pelo

módulo *TestAttributes*. Por essa razão, o uso do módulo *DescriptorControl*, por um determinado perfil de linguagem, exige o uso do módulo *TestAttributes*, além do módulo *BasicDescriptor*, para contemplar a definição de descritores.

3.2.11 Área Funcional *Metainformation*

A área funcional *Metainformation* é idêntica à área funcional de mesmo nome da linguagem SMIL 2.0. Ela define um único módulo chamado *Metainformation*, que permite a definição de metadados sobre um documento ou sobre elementos de um documento, seguindo as recomendações do padrão RDF – *Resource Description Framework* (W3C, 1999a) – para a definição de metadados. Toda informação sobre metadados de um documento deve ser incluída em seu cabeçalho (elemento *head*).

3.3 Perfil de Linguagem NCL 2.0 *Language Profile*

O perfil de linguagem chamado NCL 2.0 *Language Profile*, tal como SMIL 2.0 *Language Profile*, é o perfil completo da linguagem NCL 2.0, ou seja, que inclui todos os seus módulos e oferece todas as facilidades para a autoria declarativa de documentos NCL. As tabelas seguintes indicam os elementos e atributos, considerando seus modelos de conteúdo especificados no perfil completo de NCL 2.0, separados por área funcional. Na especificação do conteúdo, os símbolos indicam o seguinte: (?) opcional, (|) ou, (*) zero ou mais ocorrências, (+) uma ou mais ocorrências.

Tabela 2. Área Funcional *Structure*

Elementos	Atributos	Conteúdo
ncl	*id*	*(head?, body)*
head		*(Metainformation*, layout?, descriptorBase?)*
body		*(BasicMedia|composite|switch|linkBase)**

Tabela 3. Área Funcional *Components*

Elementos	Atributos	Conteúdo
animation, audio, img, text, textstream, vídeo, ref	*id, src, type, descriptor, label, TestAttributes*	*area*, attribute**
composite	*id, descriptor, xtemplate, label, TestAttributes*	*(areaComposite*, port*, attribute*, componentPresentation*, (BasicMedia|composite|linkBase|switch)*)*

Tabela 4. Área Funcional *Interfaces*

Elementos	Atributos	Conteúdo
area	*id, coords, begin, end, dur, text, position, first, last, label*	*vazio*
areaComposite	*id, componentList, label*	*vazio*
port	*id, component, port, label*	*vazio*
attribute	*id, name*	*vazio*
portSwitch	*id, label*	*port+*

Tabela 5. Área Funcional *Linking*

Elementos	Atributos	Conteúdo
bind	*role, component, port*	*vazio*
param	*name, value*	*vazio*
link	*id, xconnector*	*(param*, bind+)*
lref	*id, src*	*vazio*
linkBase	*id*	*(link|lref)+*

Tabela 6. Área Funcional *Connectors*

Elementos	Atributos	Conteúdo
xconnector	*id*	*(param*, role+, glue)*
param	*name, type*	*vazio*
connectorBase	*id*	*xconnector+*

compositeConnector	*id*	*(role+, glue)*

Tabela 7. Área Funcional *Composite Templates*

Elementos	Atributos	Conteúdo
xtemplate	*id*	*(vocabulary, constraints?)*
templateBase	*id*	*xtemplate+*

Tabela 8. Área Funcional *Layout*

Elementos	Atributos	Conteúdo
layout	*src, type*	*topLayout**
topLayout	*id, title, left, top, height, width, backgroundColor, zIndex, scroll, open, close, movable, resizable, visible*	*region**
region	*id, title, left, top, height, width, backgroundColor, zIndex, scroll, open, close, fit, visible*	*region**

Tabela 9. Área Funcional *Presentation Specification*

Elementos	Atributos	Conteúdo
descriptor	*id, player, dur, min, max, region, enableTimeBar, style, soundLevel, balanceLevel, trebleLevel, bassLevel, TestAttributes*	*vazio*
descriptorBase		*(descriptor|descriptorSwitch)**
componentPresentation	*component, descriptor*	*vazio*

Tabela 10. Área Funcional *Presentation Control*

Elementos	Atributos	Conteúdo
switch	*id, TestAttributes*	*(portSwitch*, componentPresentation*, (BasicMedia|composite|switch)+)*
descriptorSwitch	*id*	*descriptor+*

O Apêndice D apresenta a definição do perfil NCL 2.0 *Language Profile* em XML Schema. A próxima seção dá exemplos de outros perfis que podem ser construídos, agrupando algumas das facilidades de NCL 2.0.

3.4 Exemplos de Outros Perfis de Linguagem Agrupando Módulos de NCL 2.0

Além do perfil NCL 2.0 *Language Profile*, outros perfis podem ser construídos, como, por exemplo:

- *Simple NCL Language Profile* – permite a criação de documentos NCL sem contemplar conectores compostos, controle de conteúdo (elemento *switch*), descritores alternativos, atributos como pontos de interface e definição de metadados. Esse perfil inclui os módulos *Structure*, *BasicMedia*, *MediaInterface*, *BasicComposite*, *CompositeInterface*, *XConnector*, *Linking*, *BasicDescriptor*, *XTemplate* e *XTemplateUse*. Esse perfil foi o usado na implementação atual do sistema HyperProp, que será comentada no Capítulo 6.
- *Basic Media Profile* – permite a criação de documentos hipermídia sem nós de composição e sem conectores compostos. Esse perfil precisa incluir os módulos *Structure*, *BasicMedia*, *MediaInterface*, *XConnector*, *Linking* e *BasicDescriptor*.
- *Basic NCM Profile* – permite a criação de documentos NCM sem templates de composição hipermídia e sem conectores compostos. Esse perfil precisa incluir os módulos *Structure*, *BasicMedia*, *MediaInterface*, *AttributeInterface*, *BasicComposite*, *CompositeInterface*, *XConnector*, *Linking*, *BasicDescriptor*, *CompositeDescriptor*, *TestAttributes*, *ContentControl*, *SwitchInterface* e *DescriptorControl*.
- *Basic Linking Profile* – permite a criação de elos e bases de elos sem conectores compostos. Esse perfil precisa incluir os módulos *Structure*, *XConnector* e *Linking*.
- *XConnector Profile* – permite a criação de conectores hipermídia sem composição. Esse perfil precisa incluir apenas o módulo *XConnector*, que é independente dos demais módulos de NCL 2.0.

- *XTemplate Profile* – permite a criação de templates de composição hipermídia e conectores sem composição. Esse perfil precisa incluir os módulos *XConnector* e *XTemplate*.