



PUC

ISSN 0103-9741

Monografias em Ciência da Computação
nº 18/05

Nested Context Model 3.0 Part 1 – NCM Core

**Luiz Fernando Gomes Soares
Rogério Ferreira Rodrigues**

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900

RIO DE JANEIRO - BRASIL

Nested Context Model 3.0

Part 1 – NCM Core¹

Luiz Fernando Gomes Soares
Rogério Ferreira Rodrigues

Laboratório TeleMídia DI – PUC-Rio
Rua Marquês de São Vicente, 225, Rio de Janeiro, RJ - 22453-900.
lfgs@inf.puc-rio.br, rogerio@telemidia.puc-rio.br

Resumo. *Este relatório técnico descreve as entidades básicas da versão 3.0 do Modelo de Contextos Aninhados (NCM - Nested Context Model). NCM é um modelo conceitual focado na representação e tratamento de documentos hipermídia.*

Abstract. *This technical report describes the basic entities of Nested Context Model (NCM) version 3.0. NCM is a conceptual model focused on the representation and handling of hypermedia documents.*

¹ Este relatório possui uma versão em inglês sob o mesmo título.



Nested Context Model 3.0

Part 1 – NCM Core

Versão em Português

© 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

22453-900 – Rio de Janeiro – RJ – Brasil

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

Índice

1. Introdução	5
2. Entidades Básicas do NCM	7
2.1. Entidades e Propriedades	7
2.2. Nós e Âncoras	8
2.3. Nós de Conteúdo	9
2.4. Nós de Composição	10
2.5. Nós de Contexto	12
2.6. Nós Switch (Alternativas de Nós)	13
2.7. Eventos	14
2.8. Elos	17
2.8.1. Conectores	17
2.8.2. Binds de Elos	24
2.9. Objetos de Dados versus Objetos de Representação	27
2.10. Descritores Genéricos, Descritores e Switches de Descritores	29
2.11. Trilhas	30
2.12. Hiperbase Pública e Bases Privadas	32
3. Considerações Finais	33
Referências	34
Apêndice A : Exemplos de Uso de Elos NCM	35

Modelo de Contextos Aninhados 3.0: Parte 1

NCM Básico

1. Introdução

Este relatório descreve as entidades básicas da versão 3.0 do NCM (*Nested Context Model*). NCM é um modelo conceitual centrado na representação e tratamento de documentos hipermídia.

Um modelo conceitual hipermídia deve representar os conceitos estruturais dos dados, assim como os eventos e relacionamentos entre os dados. O modelo deve também definir as regras estruturais e as operações sobre os dados para manipulação e atualização das estruturas.

As primeiras descrições do NCM [SoCR91] se ativeram mais aos aspectos da estrutura de dados do modelo básico e das regras de estruturação. As estruturas e operações para controle de versão foram assuntos posteriores de especificação [SoCR95, SSRM99]. Na versão 2.2 [Soar00], as definições anteriores foram revistas e atualizadas, acrescentando ao modelo básico operações para criação, edição e exibição da estrutura do documento. A especificação NCM 2.2 também adicionou novas estruturas de dados para a definição de relações de sincronismo espacial e temporal de documentos, bem como operações para definição e exibição dessas relações de sincronismo.

Para a introdução dos novos conceitos na versão 2.2, a hierarquia de classes original do NCM precisou ser inteiramente revista. Novas classes foram acrescentadas e novos métodos foram incorporados às classes anteriormente especificadas em [SoCR95]. Além disso, foram necessárias modificações na especificação de algumas das classes originais, especialmente da classe *elo* (*link*), através da introdução de novos atributos ou mesmo da redefinição dos antigos atributos. Na verdade, todas as classes do modelo que podem ter objetos instanciados foram parcialmente ou significativamente modificadas.

Essa nova versão do NCM, chamada NCM 3.0, realiza uma nova revisão da hierarquia de classes do modelo. Novamente, a classe *elo* é uma das principais entidades do modelo afetada pela revisão, sendo inteiramente redefinida. Novos suportes para adaptações de documentos e de suas apresentações são também introduzidos ao modelo.

Nessa nova versão 3.0 do NCM, uma estratégia de especificação diferente é também adotada. Com o objetivo de oferecer um modelo hipermídia escalável, com características que possam ser progressivamente incorporadas nas implementações dos sistemas hipermídia, o NCM foi dividido em várias partes:

- Parte 1 – NCM Básico
descreve as principais entidades do modelo, as quais devem estar presentes em todas as implementações do NCM.²
- Parte 2 – Entidades Virtuais no NCM
concentra-se principalmente nas definições de nós, âncoras e elos virtuais.

² É também possível ter implementações NCM que ignorem algumas das entidades básicas, mas isso não é tão relevante na definição de um núcleo mínimo do modelo.

- Parte 3 – Controle de Versões no NCM centrada nas entidades e atributos do modelo que oferecem suporte ao versionamento das entidades hipermídia.
- Parte 4 – Trabalho Cooperativo no NCM centrada nas entidades e atributos do modelo que oferecem suporte ao tratamento de documentos de forma cooperativa.

NCM é o modelo base da linguagem NCL (*Nested Context Language*), uma linguagem de aplicação XML para autoria de documentos hipermídia. A especificação da linguagem NCL é composta das partes 5 a 12 da seguinte coleção:

- Parte 5 – Perfil Completo da NCL (Nested Context Language) concentra-se na definição de uma linguagem de aplicação XML para autoria e troca de documentos baseados no NCM, que faz uso de todos os módulos NCL, incluindo aqueles para definição de templates, de conectores de restrição, de conectores compostos, de funções de custo temporal, de efeitos de transição e de metainformação.
- Parte 6 – Família de Perfis XConnector da NCL (Nested Context Language) concentra-se na definição de uma linguagem XML para autoria de bases de conectores. Um perfil é definido para autoria de conectores causais, outro para autoria tanto de conectores causais quanto de restrição, e ainda um terceiro tanto para a definição de conectores simples quanto conectores compostos.
- Parte 7 – Templates para Nós de Composição concentra-se na definição da funcionalidade NCL Composite-Node Template, e da linguagem de aplicação XML XTemplate para autoria de bases de templates.
- Parte 8 – Perfis NCL (Nested Context Language) para TV Digital concentra-se na definição de uma linguagem de aplicação XML para autoria de documentos voltados para o domínio de TV Digital. Dois perfis são definidos: o perfil Enhanced Digital TV (EDTV) e o perfil Basic Digital TV (BDTV).
- Parte 9 – Comandos para Edição NCL ao Vivo concentra-se na definição de comandos de edição para autoria de aplicações NCL ao vivo.
- Parte 10 – Objetos Imperativos em NCL: A Linguagem de Script NCLua concentra-se na definição de objetos contendo código imperativo e como esses objetos podem relacionar entre si e com outros objetos em documentos NCL.
- Parte 11 – Objetos Hipermídia Declarativos em NCL: Aninhando Objetos com Código NCL em Documentos NCL concentra-se na definição de objetos contendo código declarativo e como esses objetos podem relacionar entre si e com outros objetos em documentos NCL.
- Parte 12 – Suporte a Múltiplos Dispositivos de Exibição concentra-se no uso de múltiplos dispositivos para a apresentação de um mesmo documento NCL.

Este relatório técnico lida com as entidades básicas que formam a base do NCM, conforme discutido ao longo de todas as subseções da Seção 2.

2. Entidades Básicas do NCM

A definição de documentos hiperfídia no NCM é baseada nos conceitos usuais de nós e elos. *Nós* (*nodes*) são fragmentos de informação e *elos* (*links*) são usados para a definição de relacionamentos entre os nós. No entanto, os elos não são a única forma de definição de relacionamentos, como ficará evidente a seguir.

O modelo distingue duas classes básicas de nós, chamados de nós de conteúdo (*content nodes*) e nós de composição (*composite nodes*), sendo esses últimos o ponto central do modelo. A Figura 1 ilustra a visão geral da hierarquia de classes do modelo,³ que será detalhada ao longo desta seção seguindo uma abordagem *top-down*.

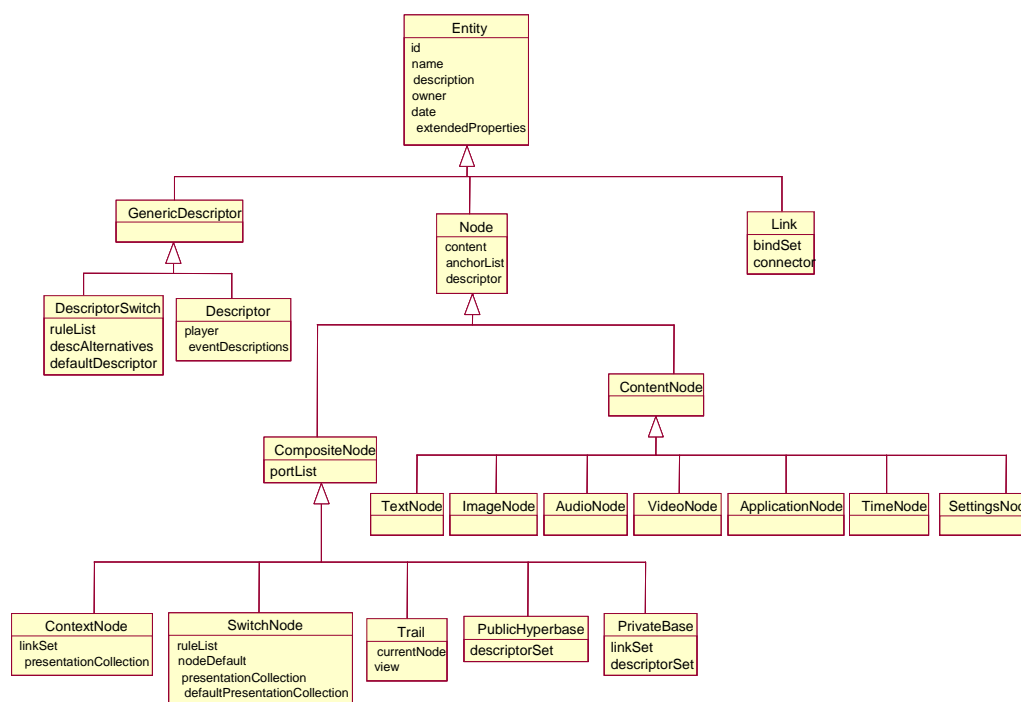


Figura 1 – Visão geral da hierarquia de classes do NCM.⁴

2.1. Entidades e Propriedades

Toda *entidade* (*entity*) do modelo possui como atributos: um *identificador único* (ID), um *nome*, uma *descrição*, a *data de criação* e um *autor*.⁵ Além dessa coleção básica de atributos, uma entidade NCM mantém uma lista de atributos estendidos, para permitir extensões que não se restrinjam apenas de heranças de classes. No NCM, a maioria dos atributos são chamados *propriedades* e devem ser envolvidos (*wrapped*) por uma classe do modelo chamada *propriedade* (*property*). Isso permite ao NCM o suporte para

³ É necessário salientar que, embora seguindo uma especificação orientada a objetos, o NCM não obriga que sua implementação seja orientada a objetos. NCM é apenas um modelo hiperfídia.

⁴ Com o objetivo de não poluir visualmente as figuras, os métodos das classes foram omitidos nos diagramas.

⁵ O NCM define um tipo *usuário* cuja implementação fica a cargo dos sistemas hiperfídia que utilizem as classes do modelo.

manutenção, para cada propriedade da entidade (*básica* ou *estendida*), de informações a cerca de direitos de acesso, do último usuário que modificou o seu valor, da data dessa modificação, se a mudança deve ocasionar ou não um versionamento da entidade etc. Em outras palavras, o NCM prevê um controle granular bastante fino quando da implementação de suporte a controle de versões e controle de acesso das entidades, obrigando que as propriedades mantenham outros atributos. No entanto, sistemas que não tenham interesse em explorar todas as capacidades do modelo podem optar por representar os campos das classes como atributos tradicionais, ao invés de utilizar o *wrapper* propriedade oferecido pelo modelo. Mesmo para aqueles sistemas que implementam controles de acesso e/ou de versões, campos de classes que não necessitem ser monitorados com tal granularidade podem ser representados sem a utilização dos *wrappers*.

Entidades do modelo devem oferecer métodos *get* e *set* para cada propriedade básica (por exemplo, *getId*, *setId*, *getName*, *setName*, etc.),⁶ métodos para adicionar/remover propriedades estendidas, e dois métodos genéricos para consultar (*get*) e modificar (*set*) valores das propriedade estendidas.

2.2. Nós e Âncoras

Um *nó* (*node*) é uma entidade NCM que tem como propriedades básicas adicionais: um conteúdo, um descritor genérico (propriedade opcional) e uma lista ordenada de âncoras.

O *conteúdo* de um nó é composto por uma coleção de unidades de informação. A noção exata do que constitui uma unidade de informação é parte da definição do nó e depende de sua especialização, conforme será exemplificado mais adiante.

Descritores NCM serão explicados nas Seções 2.9 e 2.10. A definição de um descritor como propriedade do nó é opcional. Quando especificado, ele irá conter informações determinando como o nó deverá ser exibido no tempo e no espaço. Infelizmente, as definições de algumas das classes do NCM são interdependentes, o que dificulta a tarefa de construção de um texto estritamente linear. A definição deste modelo é um exemplo típico onde o paradigma de hipertexto seria de grande auxílio.

Cada elemento da *lista ordenada de âncoras* é chamado âncora do nó, ou simplesmente âncora. Toda *âncora* (*anchor*) é uma entidade NCM, que pode ser especializada, conforme ilustrado no diagrama de classes da Figura 2.⁷ O modelo define dois tipos de âncora (ou interfaces). O primeiro tipo é chamado de *âncora de conteúdo* (*content anchor* ou *area anchor*), que possui um atributo denominado *região* (*region*). A *região* da âncora define uma coleção de unidades de informação do conteúdo do nó. Qualquer subconjunto de unidades de informação do conteúdo de um nó pode definir uma âncora. Tem-se assim que a definição exata da região da âncora é dependente do tipo de conteúdo do nó. No entanto, o modelo requer que todo nó contenha uma âncora de conteúdo com uma região representando a marcação de todo o conteúdo do nó. Essa âncora é chamada de *âncora conteúdo total* e sua região correspondente é representada pelo símbolo λ . A âncora conteúdo total deve sempre ser a primeira âncora na lista de

⁶ Deste ponto em diante, será assumido que as subclasses deverão especificar métodos do tipo *get* e *set* para manipular cada uma de suas propriedades.

⁷ Na verdade, âncoras NCM herdam da classe ponto de interface, que por sua vez herda da classe entidade, como será melhor explicado na Seção 2.4.

âncoras do nó. Âncoras serão extremamente importantes na especificação de relacionamentos entre nós.

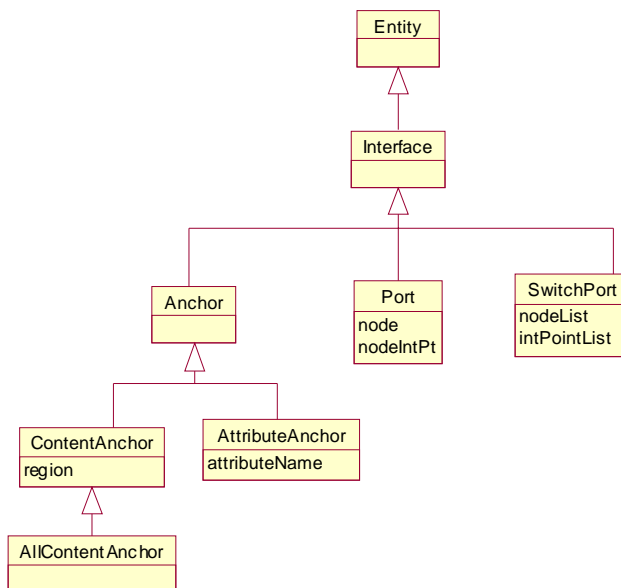


Figura 2 –Hierarquia de classes das interfaces de nós NCM.

O segundo tipo de âncora definido no modelo é a *âncora de atributo (attribute anchor)*. A âncora de atributo aponta para um atributo (ou propriedade) do nó. Como será explicado na Seção 2.9, durante a apresentação de documentos NCM, nós são associados a descritores. Na especificação das âncoras de atributo do nó, autores podem referenciar não apenas atributos do nó, mas também atributos definidos no descritor que venha a ser associado ao nó. Na verdade, a âncora de atributo pode identificar qualquer atributo recursivamente contido no nó, através de referências qualificadas. Como exemplo, um autor pode usar o nome qualificado *descriptorSelecionado.posicaoX* para criar uma âncora de atributo que aponte para um atributo *posicaoX* definido em um descritor selecionado para apresentar o nó.

Além dos já mencionados métodos para consultar e modificar os valores dos atributos, nós devem oferecer métodos para manipulação de suas listas de âncoras (por exemplo, adicionar, remover, recuperar uma âncora, percorrer a lista etc.).

2.3. Nós de Conteúdo

Um *nó de conteúdo (content node)*, também chamado *nó de mídia* ou *objeto de mídia*, é um nó que representa um objeto em uma mídia qualquer. Nós de conteúdo devem ser especializados em subclasses para melhor definir a interpretação do conteúdo (texto, áudio, imagem, vídeo, aplicação etc.). Conforme mencionado anteriormente, a noção exata do que constitui uma unidade de informação do conteúdo é parte da definição da classe do nó. Por exemplo, uma unidade de informação do conteúdo de um nó vídeo pode ser um quadro, enquanto uma unidade de informação do conteúdo de um nó texto pode ser um caractere, ou uma palavra. O conteúdo de um nó de mídia pode ser definido como uma referência (por exemplo, uma URI) para o conteúdo propriamente dito, ou como uma seqüência de bytes (*raw data*). Além disso, cada subclasse de nó de conteúdo pode ser refinada. Como exemplo, nós texto podem ser especializados em nós HTML, nós PDF etc.

Um tipo especial de nó de conteúdo definido pelo modelo é o *nó de tempo* (*time node*). Como o próprio nome sugere, esse nó representa o tempo. Esse nó representa tempos absolutos (baseados, por exemplo, na hora GMT), ou um tempo relativo (baseado em eventos, como por exemplo o início da apresentação de um documento). Cada instante de tempo é considerado uma unidade de informação para o seu conteúdo. Dessa forma, intervalos podem ser definidos como âncoras de um nó de tempo. O nó de tempo vai assim permitir acionar eventos (Seção 2.7) baseados em instantes de tempo específicos.

Outro tipo especial de nó de conteúdo é o *nó de ambiente* (*settings node*). Esse nó representa todas as variáveis controladas pelo formatador (ferramenta responsável pela apresentação de um documento NCM). As variáveis são representadas pelos atributos (propriedades) do nó. Como discutido na Seção 2.8, esses atributos podem ter seus valores modificados por ações dos elos. Como discutido nas Seções 2.6 e 2.10, esses atributos podem também ter seus valores testados pelas regras de nós *switch* e de *switch de descritores*, a fim de realizar a escolha entre alternativas.

2.4. Nós de Composição

Um *nó de composição* (*composite node*), ou simplesmente *composição*, C é um nó cujo conteúdo é uma coleção C_L de nós (de conteúdo ou de composição, recursivamente) que se constituem em suas unidades de informação. Diz-se que um nó N em C_L é um *componente de C* e que N está contido em C . Diz-se também que um nó A está *recursivamente contido em C* se e somente se A está contido em C ou A está contido em um nó recursivamente contido em C . Deve-se também notar que os componentes de C podem ser ordenados (lista ordenada), o que será útil na definição de operações de navegação. Note também que um nó pode estar contido mais de uma vez em C_L . Uma restrição importante é feita, no entanto: um nó não pode estar recursivamente contido em si mesmo.

Nós de composição diferentes podem conter um mesmo nó e ser aninhados em qualquer profundidade, desde que a restrição de um nó não conter recursivamente a si mesmo seja obedecida. Para identificar através de que seqüência de nós de composição aninhados uma dada instância de um nó N está sendo observada, é introduzida a noção de perspectiva de um nó. A *perspectiva* de um nó N é uma seqüência $P = (N_m, \dots, N_1)$, com $m \geq 1$, tal que $N_1 = N$, N_{i+1} é um nó de composição, N_i está contido em N_{i+1} , para $i \in [1, m)$ e N_m não está contido em qualquer nó. Note que pode haver várias perspectivas diferentes para um mesmo nó N , se esse nó estiver contido em mais de uma composição. A *perspectiva corrente* de um nó é aquela percorrida pela última navegação ao nó (as diversas formas de navegação serão definidas a posteriori). Dada a perspectiva $P = (N_m, \dots, N_1)$, o nó N_1 é chamado *nó base da perspectiva*.

Nós de composição são objetos cuja semântica é bem conhecida pelo modelo. Um modelo conceitual deve representar não apenas os conceitos estruturais dos dados, mas também definir operações sobre os dados para manipulação e atualização das estruturas. Assim, todo nó C de composição deve possuir os seguintes métodos:

- Insere nó: deferido (implementação dependente da subclasse de composição).
- Retira nó: retira um nó da coleção de nós da composição.

Com base nas definições de conteúdo de composição e regiões de âncoras de conteúdo (Seção 2.2), conclui-se que a região de uma âncora de conteúdo de uma

composição deve especificar um subconjunto dos componentes da composição. Um subconjunto especial é aquele com todos os nós da composição (âncora conteúdo total (λ) da composição).

Além da lista ordenada de âncoras, nós de composição têm uma outra propriedade chamada *lista ordenada de portas*. Portas e âncoras possuem propósito similar e estendem um tipo comum denominado *interface*. Uma porta (*port*) de uma composição C é uma entidade NCM que possui dois atributos adicionais: um nó N e uma interface ip ; onde N deve estar obrigatoriamente contido em C e ip deve ser uma interface definida em N , ou seja, contido em sua lista de âncoras ou de portas.⁸ Como pode ser percebido, a porta de um nó de composição permite definir mapeamentos entre a composição e seus nós internos. Como consequência, o nó de composição pode tornar a interface de um nó constituinte visível para referências externas (elos hipermídia, por exemplo). O conjunto de interfaces (âncoras ou portas) age como uma proteção para referências a um nó, no sentido de que qualquer entidade só pode ter acesso a segmentos do conteúdo ou atributos de um nó se eles estiverem disponíveis na interface do nó (lista de âncoras ou de portas). Dessa forma, as interfaces podem impedir que modificações internas em um nó se reflitam em outras entidades que o referenciam. Tome como exemplo um nó texto com uma âncora de conteúdo cuja região especifique seu segundo parágrafo. Qualquer mudança no texto, por exemplo, a eliminação do primeiro parágrafo, deve se refletir na região da âncora, mas não deverá afetar as demais entidades que a referenciam, assim mantendo as referências para a parte correta do conteúdo (i.e. o antigo segundo parágrafo agora posicionado como sendo o primeiro). A proteção do nó através de interfaces também trará ao modelo o conceito de composicionalidade, permitindo provas formais de propriedades dos documentos, como a consistência temporal.

Defini-se como *seqüência de mapeamentos da porta* p_k de uma composição N_k a seqüência de nós e interfaces $(N_k, ip_k, \dots, N_1, ip_1)$ com $k > 1$, tal que para $i \in [1, k]$:

- i) N_{i+1} é um nó de composição, e N_i está contido em N_{i+1} ,
- ii) ip_i é uma interface do nó N_i na seqüência de nós da porta, e N_i e ip_i são os valores dos atributos nó e interface, respectivamente, da porta ip_{i+1} . Diz-se que ip_i *pertence* à seqüência de mapeamentos da porta p .

Note que a definição de dois tipos de interfaces de composições (âncoras e portas) permite dois tipos de tratamento para ações de apresentação, muitas vezes desejável na construção de um documento hipermídia. Um autor pode desejar apresentar uma composição para que a estrutura da composição seja visualizada (por exemplo, um desenho mostrando o grafo estrutural da composição). Âncoras de composições são interfaces que a princípio expressam esse tipo de comportamento, e a região da âncora irá enumerar os componentes que devem ser desenhados. Todavia, apresentar uma composição, algumas vezes, pode significar apresentar seus constituintes a partir de um ponto de entrada, sem que a visão estrutural da composição seja exibida. Portas servem exatamente para fornecer pontos de acesso, permitindo que referências externas toquem em nós contidos dentro de um nó de composição, sem se perder a propriedade de composicionalidade do modelo (i.e. pontos de entrada e saída das composições devem ser explicitamente definidos).

⁸ Evidentemente, ip estará contido na lista de portas de N apenas se N for uma composição.

Uma ação sobre um nó de composição deve especificar a interface onde se aplica. Caso não especifique, deve ser considerada como sendo aplicada em todas as suas portas.

Subclasses de composição irão definir semânticas para coleções específicas de nós. Cinco importantes subclasses de composição definidas pelo modelo são: nó de contexto, nó switch, trilha, hiperbase pública e base privada.

2.5. Nós de Contexto

Um *nó de contexto* (*context node*) é um nó de composição tal que seu conteúdo contém um conjunto de nós de conteúdo, nós de contexto ou nós *switch*.⁹ Os nós de contexto possuem como atributos adicionais: um conjunto de elos e uma coleção de apresentação.

Cada *elo* l contido no conjunto de elos de um nó de contexto C define um relacionamento entre nós recursivamente contidos em C ,¹⁰ ou o próprio nó de contexto C . Diz-se que um elo l é um *componente de um nó de contexto* C e que l está contido em C . Diz-se também que um elo l está *recursivamente contido em* C se e somente se l está contido em C ou l está contido em um nó de contexto recursivamente contido em C . Elos são o assunto da Seção 2.8.

Uma *coleção de apresentação* contém para cada nó contido em um nó de contexto C , um grupo de descritores genéricos.¹¹ Como mencionado na Seção 2.2, descritores reúnem as informações referentes às características de apresentação do nó e serão tratados em detalhes nas Seções 2.9 e 2.10. Cada grupo de descritores genéricos deve necessariamente formar um conjunto, ou seja, não pode haver repetição de descritores no grupo¹². No caso do nó contido em C ser um nó de contexto, o grupo de descritores deve conter no máximo um descritor genérico.

Nós de contexto vão servir, entre outras coisas, para definir uma estrutura lógica, hierárquica ou não, para documentos hipermídia. Essa estruturação irá permitir a definição de diferentes visões de um mesmo documento e também melhorar a orientação do usuário na navegação sobre um documento.

Um nó de contexto C deve ter definido o método deferido de nó de composição:

- **Inserir nó:** insere um nó de conteúdo, um nó switch ou um outro nó de contexto no conjunto de nós de C .

Além dos métodos para consultar e modificar os valores dos atributos, dos métodos para manipular as listas de portas e de âncoras, e dos métodos para manipular o conjunto de nós, nós de contexto devem também prover métodos para manipular seus

⁹ Nós switch serão apresentados na Seção 2.6. Nó switch é uma especialização de composição, e permite definir alternativas de nós para documentos adaptativos.

¹⁰ Como será discutido na Seção 2.8, relacionamentos podem ter seus participantes definidos através de mapeamentos para nós recursivamente contidos na composição C .

¹¹ O grupo pode estar vazio para qualquer constituinte do contexto.

¹² A semântica por trás da definição do grupo de descritores selecionados para cada nó N contido em um nó de contexto é permitir uma navegação em profundidade para N especificando várias exibições diferentes simultâneas, como será melhor explicado na Seção 2.9. Além disso, como será comentado na Seção 2.10, um descritor do grupo pode ser o resultado de uma seleção entre alternativas de descritores (*switch de descritores*), cuja escolha poderá depender de parâmetros da plataforma ou do próprio usuário.

conjuntos de elos e suas coleções de apresentação (por exemplo, inserir, remover, consultar, percorrer etc.).

2.6. Nós Switch (Alternativas de Nós)

O NCM possui várias características que oferecem suporte à adaptação de documentos. Uma importante facilidade é o grupo de nós alternativos, cuja seleção é feita com base em *regras* (*rules*) associadas ao documento. A **Error! Reference source not found.** descreve o diagrama de classes para regras NCM.

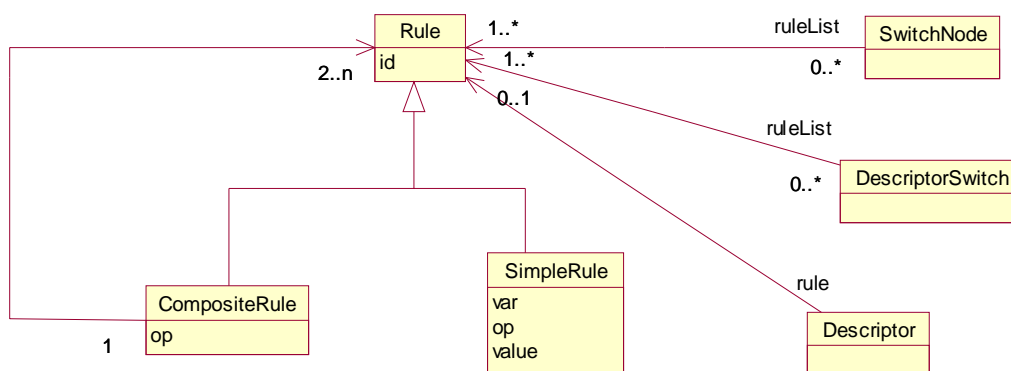


Figura 3 – Diagrama de classes para regras.

Baseado nas informações contextuais (por exemplo, preferências do usuário, características da plataforma de exibição etc.)¹³, o formatador de documentos NCM deve avaliar cada regra para descobrir se uma determinada entidade associada à regra deve ou não ser considerada na apresentação do documento. A forma que entidades e regras são associadas será explicada adiante.

Uma regra pode ser simples ou composta. Uma *regra simples* (*simple rule*) é análoga à expressão-assertiva do conector que compara uma avaliação com um valor (Seção 2.8.1) e possui três atributos: um identificador (*var*) da variável a ser testada, um operador de comparação (=, ≠, <, ≤, >, ≥) e um valor. A *regra composta* (*compound rule*) é uma expressão lógica compreendendo duas ou mais regras (simples ou compostas) relacionadas através de operadores lógicos *AND* e *OR*.

Com o objetivo de permitir que um autor especifique alternativas de nós dependendo da informação contextual (atributos do contexto de exibição), o NCM define uma entidade chamada nó switch. O *nó switch* (*switch node*) é uma especialização de nós de composição. O conteúdo de um nó switch é um conjunto que pode incluir nós de contexto e de conteúdo. O nó switch tem um atributo adicional que define, para cada nó contido no seu conjunto de nós, uma regra associada. As regras são definidas em uma lista ordenada. O formatador de documentos deve avaliar cada uma das regras conforme a ordem na lista. O primeiro nó que tenha a sua regra avaliada como verdadeira deve ser eleito a *alternativa selecionada*.

¹³ Como anteriormente mencionado, as informações contextuais podem ser representadas por atributos (propriedades) do nó de ambiente (*settings node*).

O nó switch pode conter um nó default. Durante a apresentação do documento, esse nó deve ser eleito a alternativa selecionada do switch se nenhuma das regras for avaliada como verdadeira. Na ausência de um nó default e de uma regra avaliada como verdadeira, nenhum dos nós contidos no nó switch deve ser exibido.

Além da propriedade de lista ordenada de portas comum a toda composição (Seção 2.4), nós switch têm uma propriedade adicional chamada *lista ordenada de portas-switch* (*ordered switch port list*). Cada *porta-switch* (*switch port*) é um ponto de interface (Figura 2) que define um conjunto de mapeamentos para interfaces de nós contidos no nó switch.

A definição de portas-switch permite que elos (relacionamentos discutidos na Seção 2.8) sejam definidos ancorando em nós switch, independente do nó que será selecionado. As portas tradicionais de composição permitem que um elo seja associado a uma alternativa específica. Se essa alternativa não for selecionada, o elo será simplesmente ignorado durante a apresentação do documento.

Como os nós de contexto (Seção 2.5), um nó switch tem uma *coleção de apresentação*, cujo objetivo é permitir que sejam definidos grupos de descritores para cada nó contido no nó switch.¹⁴ Na verdade, o grupo deve formar um conjunto de descritores, pois não pode haver mais de uma instância de um mesmo descritor em um grupo. Se o nó contido no nó switch for um nó de contexto, o grupo de descritores deverá ser composto de no máximo um descritor genérico. Se o nó switch contiver um nó default, também pode ser especificado um grupo de descritores para esse nó, desde que as regras mencionadas neste parágrafo sejam seguidas.

O formatador de documentos NCM deve decidir quando avaliar as regras a fim de resolver os nós switch.

2.7. Eventos

Seguindo a definição de Pérez-Luque e Little [PéLi96], um evento é uma ocorrência no tempo que pode ser instantânea ou ter uma duração mensurável. O NCM define algumas classes básicas de evento que podem ser estendidas: evento de exibição, evento de composição, evento de seleção, evento de superposição, evento de arraste, evento de foco e evento de atribuição.

Um *evento de exibição* (*presentation event*), também chamado *evento de apresentação*, representa a exibição de uma âncora de conteúdo (segmento de mídia) de um nó de conteúdo em uma dada perspectiva seguindo as diretrizes de um dado descritor. Dessa forma, perspectivas distintas ou descritores diferentes para um mesmo nó geram diferentes eventos de apresentação NCM. Os eventos de apresentação também podem ser definidos sobre nós de contexto e switch, representando a apresentação das unidades de informação de qualquer nó dentro desses nós de composição.

Eventos de composição são definidos pela apresentação da estrutura de um nó de composição. Os eventos de composição são utilizados para apresentar o mapa da composição (organização da composição).

¹⁴ O grupo pode ser vazio para qualquer constituinte do nó switch.

Um *evento de seleção* (*selection event*) representa a seleção de uma âncora de conteúdo de um nó em uma dada perspectiva seguindo as diretrizes de um dado descritor. A forma mais comum para o usuário selecionar uma âncora é através de dispositivos de entrada como mouse e teclado, no entanto outros dispositivos também podem ser utilizados na geração desse evento, como um controle remoto de TV. Os eventos de superposição (*hovering event*), de arraste (*drag event*) e de foco (*focus event*) também representam a ação de interação correspondente sobre uma âncora de conteúdo de um nó em uma dada perspectiva seguindo as diretrizes de um dado descritor.

Um *evento de atribuição* (*attribution event*) refere-se a uma âncora de atributo de um nó, dada uma perspectiva e um descritor específico.¹⁵

No NCM, cada evento define uma máquina de estados que deve ser mantida pela máquina de controle da apresentação dos documentos, denominada formatador [Rodr03]. A Figura 4 mostra a máquina de estados geral dos eventos NCM.

Um evento NCM pode estar em um dos seguintes estados: *dormindo* (*sleeping*), *ocorrendo* (*occurring*) ou *suspensão* (*paused*). Todo evento possui um atributo denominado *ocorrências* (*occurrences*), que conta o número de vezes que o mesmo muda do estado *ocorrendo* para o estado *dormindo* durante a apresentação de um documento. Eventos como os de exibição e de atribuição também possuem um atributo denominado *repetições* (*repetitions*), que determina o número de vezes seguidas que o mesmo deve ocorrer. Esse atributo pode conter um valor finito ou o valor *indefinido*, que levará a uma execução em loop do evento, até que a mesma seja interrompida.

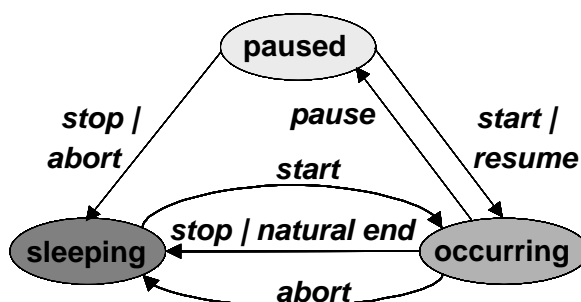


Figura 4 – Máquina de estados dos eventos NCM.

Intuitivamente, considerando um evento de exibição como exemplo (Figura 4), o evento inicia no estado *dormindo*. Ao iniciar a exibição de suas unidades de informação, o evento passa para o estado *ocorrendo*. Se a apresentação for temporariamente suspensa, o evento vai para o estado *pausado* e no mesmo permanece enquanto a situação durar. Ao final da apresentação, o evento retorna para o estado *dormindo*, seu atributo *ocorrências* é incrementado de uma unidade, e o atributo *repetições* é decrementado de uma unidade. Se após ser decrementado o atributo *repetições* possuir um valor maior que zero, a apresentação do evento é reiniciada automaticamente. Quando uma apresentação de um evento é interrompida abruptamente, através de um comando de aborto da exibição, o evento passa para o *dormindo*, sem que o atributo *ocorrências* seja incrementado e tornando zero o valor do atributo *repetições*. Eventos de seleção permanecem no estado *ocorrendo* enquanto a âncora correspondente estiver sendo selecionada. De modo similar, eventos de arraste, foco e superposição

¹⁵ É importante lembrar que, como definido na Seção 2.2, uma âncora de atributo do nó pode referenciar tanto um atributo do nó propriamente dito como um atributo do descritor associado ao nó para apresentá-lo, como discutido na Seção 2.9.

permanecem no estado *ocorrendo* enquanto a respectiva operação sobre a âncora durar. Já os eventos de atribuição permanecem no estado *ocorrendo* enquanto os valores dos atributos estiverem sendo modificados. Evidentemente, eventos instantâneos, como uma simples atribuição de valor, podem permanecer por um tempo infinitesimal no estado *ocorrendo*.

Um evento de apresentação pode mudar do estado ocorrendo para dormindo em duas situações: como consequência de um término natural da exibição de suas unidades de informação, ou devido a uma ação que force o término do evento.

A duração de um evento é o tempo que ele permanece no estado ocorrendo. No caso de um evento de apresentação, essa duração pode ser intrínseca ao objeto de mídia, ou especificada pelo descritor do evento. A duração de um evento de apresentação será escolhida pelo formatador de documentos levando em consideração parâmetros intrínsecos ao conteúdo, parâmetros do descritor, relacionamentos do documento (principalmente os elos) e outras informações externas, como características da plataforma de exibição.

Um evento de apresentação associado com um nó de composição permanece no estado ocorrendo enquanto pelo menos um evento de apresentação associado com qualquer um dos nós filhos dessa composição estiver no estado ocorrendo, ou enquanto pelo menos um elo-filho do nó de composição estiver sendo avaliado.

Um evento de apresentação associado com um nó de composição está no estado pausado se pelo menos um evento de apresentação associado com qualquer um dos nós filhos da composição estiver no estado pausado e todos os outros eventos de apresentação associados com os nós filhos da composição estiverem no estado dormindo ou pausado. Do contrário, o evento de apresentação está no estado dormindo.

Um evento de apresentação associado com um nó *switch* permanece no estado ocorrendo enquanto o elemento-filho do *switch*, escolhido (nó selecionado) através das regras de ligação (*bind rules*), estiver no estado ocorrendo. Ele está no estado pausado se o nó selecionado estiver no estado pausado. Do contrário, o evento de apresentação está no estado dormindo.

Um evento de composição permanece no estado ocorrendo enquanto o mapa da composição estiver sendo apresentado.

Elos definidos nos nós de contexto, na verdade, especificam relacionamentos entre eventos definidos nas âncoras dos nós, mais precisamente, entre máquinas de estados dos eventos, como será discutido na próxima seção. Com o objetivo de facilitar a explicação dos elos NCM, a Tabela 1 define nomes para as transições de estados e também para as ações que produzem uma determinada transição de estado nas máquinas de estados dos eventos NCM.

Tabela 1 – Nomes de transições e ações para a máquina de estados dos eventos NCM.

Transição (causado pela ação)	Nome da Transição
<i>sleeping</i> → <i>occurring</i> (<i>start</i>)	<i>starts</i>
<i>occurring</i> → <i>sleeping</i> (<i>stop</i>)	<i>stops</i>
<i>occurring</i> → <i>sleeping</i> (<i>abort</i>)	<i>aborts</i>
<i>occurring</i> → <i>paused</i> (<i>pause</i>)	<i>pauses</i>
<i>paused</i> → <i>occurring</i> (<i>resume or start</i>)	<i>resumes</i>
<i>paused</i> → <i>sleeping</i> (<i>stop</i>)	<i>stops</i>
<i>paused</i> → <i>sleeping</i> (<i>abort</i>)	<i>aborts</i>

2.8. Elos

Um *elo* (*link*) é uma entidade NCM que possui duas propriedades adicionais: um conector e um conjunto de associações a esse conector.

O conector é uma entidade introduzida nesta versão 3.0 do NCM, cujo objetivo é definir as semânticas das relações hipermídia, independentes dos participantes que irão efetivamente interagir [MuRS02].

Conectores recebem o status de entidades de primeira classe no modelo [MuSo01], isto é, conectores podem ser definidos independentemente de outras entidades do modelo.

Elos representando o mesmo tipo de relação, mas interconectando participantes (nós) diferentes, podem reusar o mesmo conector.

Um conector especifica um conjunto de pontos de acesso de interface, chamados papéis. Um elo NCM referencia um conector e define um conjunto de *binds* que relacionam cada extremidade do elo (ponto de interface de um nó) com um papel do conector referenciado.

A Figura 5 apresenta a hierarquia de classes do elo NCM, discutida nas subseções seguintes.

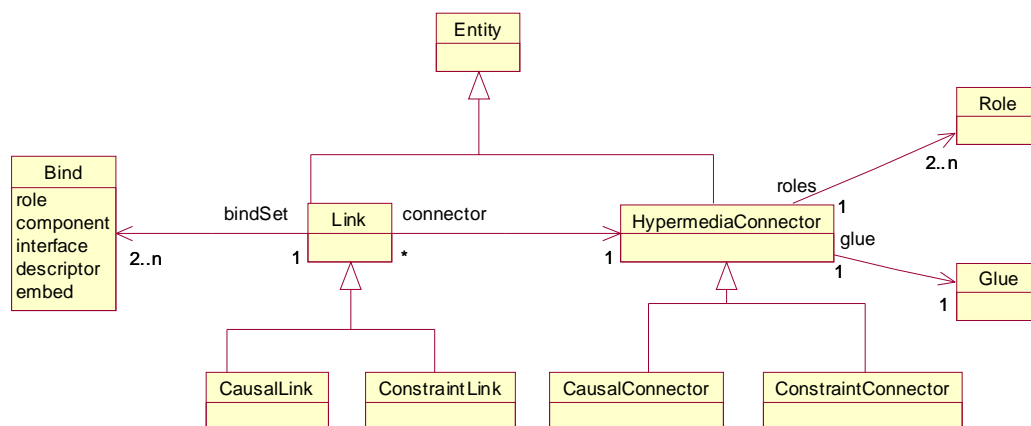


Figura 5 – Hierarquia de classes do elo NCM.

2.8.1. Conectores

A Figura ilustra um conector R representando uma relação com três papéis distintos, que significam três tipos de participantes da relação. A figura também mostra dois elos diferentes, l_1 e l_2 , reusando R . Enquanto o conector define o tipo de relação, o conjunto de binds de um elo define os participantes. O elo l_1 especifica três binds, interligando os nós A , B e C aos papéis de R . O elo l_2 também especifica três binds, só que interligando um conjunto diferente de nós (B , C e D). Os elos l_1 e l_2 definem relacionamentos diferentes, já que interligam conjuntos distintos de nós, mas representam o mesmo tipo de relação, pois usam o mesmo conector. Na especificação de um documento NCM, um elo deve fazer referência a uma instância de conector.

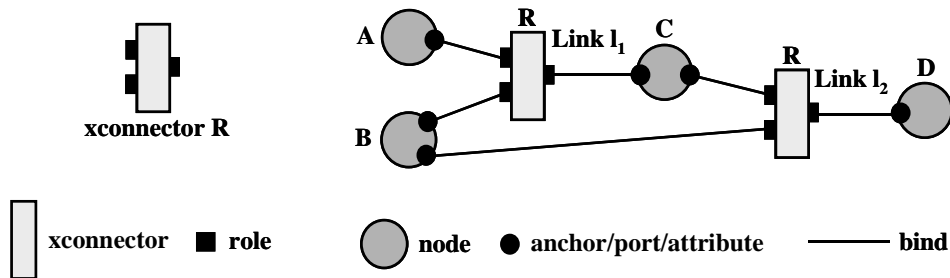


Figura 6 – Exemplos de elos que utilizam o mesmo conector *R*.

Conceitualmente, conectores podem representar qualquer tipo de relação hipermídia, tal como relações de referência, relações de sincronização, relações semânticas, relações de derivação etc. Esta versão NCM concentra seus esforços na especificação de relações de sincronização espaço-temporal assim como relações de referência,¹⁶ oferecendo o suporte necessário para a criação de documentos hipermídia.

O *conector (connector)* NCM permite a definição de relações multiponto com semântica causal ou de restrição. Em uma relação causal, uma condição deve ser satisfeita para que uma ação seja executada. Um exemplo de relação causal é a tradicional relação de referência hipermídia, que causa a navegação para um nó de destino quando uma âncora de um nó de origem for selecionada pelo usuário. Um outro exemplo de relação causal pode iniciar a apresentação de um nó, quando a apresentação de outro terminar. Além de relações causais, relações de restrição, sem nenhuma causalidade envolvida, também podem ser especificadas por conectores NCM. Considere, por exemplo, uma restrição especificando que um nó deve terminar sua apresentação ao mesmo tempo que outro começa a dele. A ocorrência de uma apresentação sem a ocorrência da outra também satisfaz a restrição, que especifica que, se e somente se esses dois nós forem apresentados, seus tempos de fim e início, respectivamente, devem coincidir.

Para capturar relações causais e de restrição, conectores são especializados em conectores causais e conectores de restrição. Em ambos os tipos, a definição de um conector é feita por um conjunto de papéis (*roles*), que determinam a função dos participantes da relação, e um atributo *glue*, que descreve como os papéis interagem. A definição de papéis é baseada no conceito de evento (Seção 2.7). Cada papel descreve um evento associado a um participante da relação e o *glue* descreve a combinação entre os eventos de acordo com a semântica de causalidade ou de restrição.

Cada papel de um conector define um identificador único (*id*) no conjunto de papéis do conector, um tipo de evento e sua cardinalidade. O tipo de evento (*event type*) especifica o nome de uma das especializações da classe evento. A Tabela 2 descreve os nomes para os tipos de evento NCM. A cardinalidade de um papel indica o número mínimo e máximo de participantes que podem desempenhar o papel (número de binds), quando esse conector for usado na criação de um elo, como será definido posteriormente.

¹⁶ Na realidade, as relações de referência são tratadas como casos particulares de relações de sincronização espaço-temporal.

Tabela 2 – Definições dos nomes para especificação dos tipos de evento nos conectores NCM

Especialização do Evento	Nome para o Tipo de Evento
<i>Evento de apresentação</i>	<i>presentation</i>
<i>Evento de seleção</i>	<i>selection</i>
<i>Evento de superposição do dispositivo apontador</i>	<i>pointOver</i>
<i>Evento de arraste</i>	<i>drag</i>
<i>Evento de atribuição</i>	<i>attribution</i>
<i>Evento de foco</i>	<i>focus</i>

Papéis são especializados em condição (*condition*), ação (*action*) e avaliação (*assessment*). Diferentes tipos de papéis são usados de acordo com o tipo de conector. Em conectores de restrição, somente papéis do tipo avaliação devem ser usados. Em conectores causais, qualquer tipo de papel pode ser usado. A Figura 7 apresenta a hierarquia de classe dos papéis de conectores NCM.

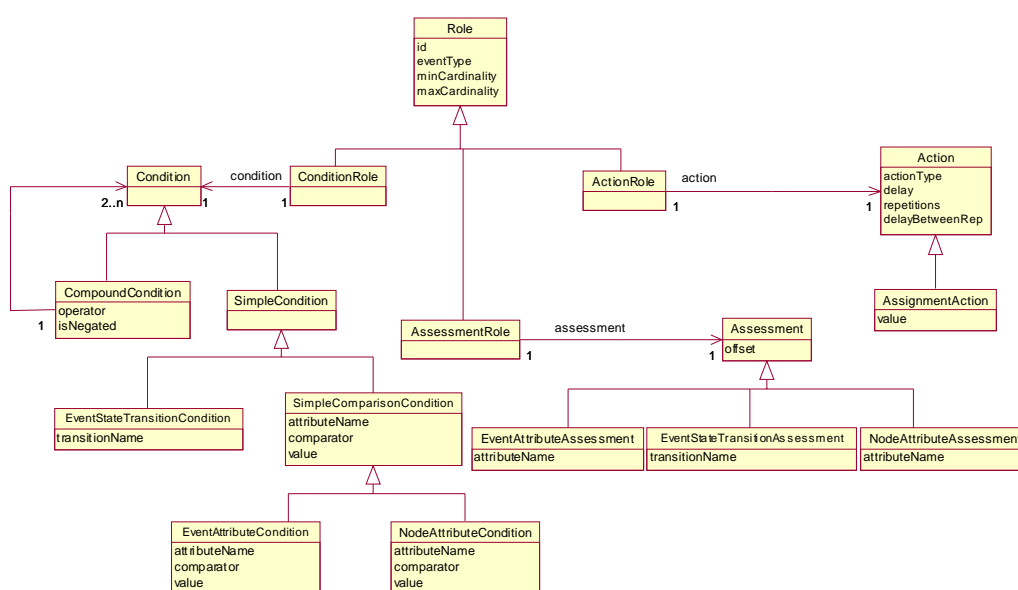


Figura 7 – Hierarquia de classes dos papéis de conectores NCM

Papéis do tipo ação (*action roles*) capturam ações que devem ser executadas em uma relação causal. Os tipos de ação estão ilustrados na Figura 4 por arcos rotulados, que causam transições na máquina de estados. Além de seu tipo, a ação de um papel pode definir um valor (*value*) a ser imposto a um atributo participante (no caso de eventos de atribuição). Um exemplo de papel do tipo ação é “pause a apresentação de um nó de conteúdo”.

Ações NCM podem ser estendidas. Por exemplo, ações de animação podem ser especificadas, definindo um valor inicial, um valor final e uma duração para que uma atribuição seja realizada (por exemplo, a posição x do objeto na tela, em um eixo de coordenadas x e y).

Em conectores causais, condições devem ser satisfeitas para disparar as ações. As condições são capturadas por papéis do tipo condição (*condition role*), que definem expressões lógicas avaliando transições nos estados dos eventos, valores de atributos dos eventos ou valores de atributos dos nós. Quando uma condição é avaliada, ela retorna um valor booleano.

As condições podem ser simples ou compostas. Uma condição simples (*simple condition*) pode testar uma transição de estado de um evento, o valor de um atributo de um evento ou o valor de um atributo de um nó. No caso da *condição sobre transição de estado do evento* (chamada *event state-transition condition*), o resultado do teste é considerado verdadeiro apenas no momento em que a transição ocorre, especificada em seu atributo *nome da transição* (*transition name*), conforme definido na Tabela 1. A *condição sobre atributo* (*attribute condition*) deve especificar o tipo do atributo a ser testado (*attributeType*): estado de um evento ou atributo de um evento (*occurrences* ou *repetitions*), associado por um elo a uma âncora de um nó; ou atributo de um nó (*nodeAttribute*), associado por um elo a um atributo qualquer de um nó. O atributo referenciado será comparado com o atributo valor (*value*) especificado na condição, usando um dos seguintes comparadores: =, ≠, <, ≤, >, ≥. A *condição sobre atributo do nó* obriga que o tipo do evento definido no papel seja de atribuição, conforme discutido na Seção 2.7. Para os eventos de seleção, o papel condição pode especificar, adicionalmente, a que dispositivo de seleção ele se refere (por exemplo, teclado ou teclas de um controle remoto), através do atributo *Key*.

Uma condição composta (*compound condition*) de um papel do tipo condição consiste de uma expressão lógica baseada nos operadores *and* ou *or* envolvendo duas ou mais condições que irão atuar sobre o mesmo evento. Um exemplo de papel com condição composta é “a apresentação de um participante terminou pela segunda vez”, que seria especificada como “[*(eventType = “presentation”), ((transition = “stops”) AND (occurrences = “2”))*]¹⁷. Note que qualquer condição composta pode ser negada utilizando o atributo booleano *está negada* (*isNegated*).

Enquanto uma condição sempre retorna um valor booleano, um papel de avaliação (*assessment role*) contém uma avaliação que retorna qualquer tipo de valor, dependendo do tipo do alvo da avaliação. Uma *avaliação de atributo* (*attribute assessment*) retorna o valor de um atributo do evento (*attributeType* igual a um dos atributos de evento: *occurrences* ou *repetitions*), ou o valor de um estado de evento (*attributeType* igual a *state*), quando associado por um elo a uma âncora de um nó; ou retorna um valor de atributo de um nó (*attributeType* igual a *nodeAttribute*), associado por um elo. Uma *avaliação de transição de estado do evento* (*event-state transition assessment*) retorna o instante de tempo em que uma transição de estado do evento, especificada no atributo *nome da transição* (*transitionName*), ocorre. Ao se referir a um evento de seleção, um papel de avaliação pode especificar, adicionalmente, a que dispositivo de seleção ele se refere, através do atributo *key*.

Como mencionado anteriormente, um conector é definido por um conjunto de papéis e um *glue*, que especifica como os papéis interagem. Todo papel de um conector deve ser usado em seu *glue*. Um conector de restrição tem um *glue de restrição* (*constraint glue*), que define uma *expressão-assertiva* (*statement expression*) relacionando papéis do tipo avaliação. Um conector causal tem um *glue causal* (*causal glue*), que define tanto uma *expressão de disparo* (*trigger expression*), relacionando papéis do tipo condição ou avaliação, quanto uma *expressão de ações* (*action expression*), relacionando papéis do tipo ação. Quando a expressão de disparo for

¹⁷ Operadores de condições compostas podem ser estendidos com outros tipos, como operadores de lógica temporal. Evidentemente, esses operadores terão de ser corretamente interpretados pelos formatadores dos documentos.

satisfeita, a expressão de ações deve ser executada. A Figura ilustra a hierarquia de classes definida pelo modelo para as expressões dos conectores NCM.

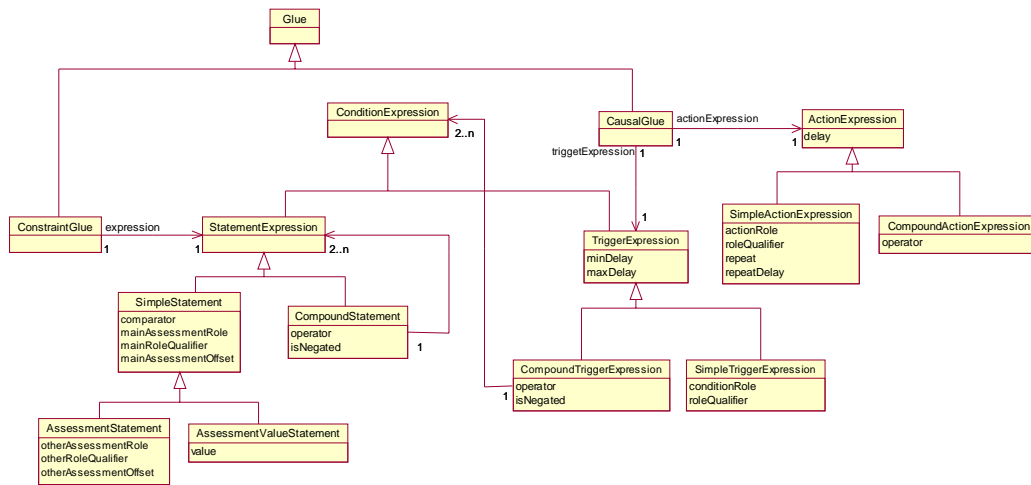


Figura 8 – Hierarquia de classes das expressões nos conectores NCM.

A expressão-assertiva do *glue* de restrição pode ser simples ou composta. Uma *assertiva simples* (*simple statement*) pode comparar papéis de avaliação do mesmo tipo (*assertiva entre avaliações – assessment statement*), ou um papel de avaliação com um valor, do mesmo tipo, do resultado da avaliação (*assertiva de valor de avaliação – assessment value statement*). Um valor de deslocamento (*offset*) pode ser adicionado a um papel de avaliação antes da comparação. Por exemplo, um deslocamento pode ser adicionado a uma avaliação especificando: “5 segundos após o instante de tempo em que um evento de apresentação termina” ou ainda “a posição vertical na tela mais 50 pixels”. A comparação pode usar os mesmos comparadores definidos para as condições simples. Por exemplo, suponha que um papel de avaliação de transição de estado de evento P especifique o tipo de evento como apresentação e a transição “starts” (início da ocorrência do evento), e que um outro papel de avaliação de transição de estado de evento Q especifique o tipo de evento como apresentação e a transição “stops” (término da ocorrência do evento). Se uma assertiva entre avaliações S_1 definir que “ $P = Q$ ”, S_1 será avaliada como verdadeira se um evento de apresentação associado a P iniciar ao mesmo tempo em que um outro evento de apresentação associado a Q terminar.¹⁸ Como um outro exemplo, suponha que um papel de avaliação H contenha uma avaliação de atributo do nó que especifique a posição horizontal na tela. Se uma assertiva de valor de avaliação S_2 definir que “ $H \geq 100$ ”, S_2 será avaliada como verdadeira se a posição horizontal de um participante desempenhando o papel H for maior que “100”. Quando o valor da cardinalidade máxima de um papel é maior do que um, vários participantes podem desempenhar o mesmo papel. Nesse caso, um *qualificador* (*qualifier*) precisará ser definido cada vez que o papel for utilizado nas expressões do *glue*, como será explicado na descrição da Tabela 4.

Uma *assertiva composta* (*compound statement*) consiste de uma expressão lógica, baseada nos operadores *and* ou *or*, envolvendo duas ou mais outras expressões-assertivas. Assertivas compostas podem opcionalmente ser negadas.

¹⁸ Como comentado, se o primeiro evento de apresentação não for iniciado ou o segundo evento de apresentação não terminar, a expressão permanece verdadeira.

Apesar de expressões-assertivas poderem ser utilizadas em conectores causais, sua principal utilidade está na especificação de conectores de restrição. A semântica de um conector de restrição é a de que a expressão-assertiva deve ser mantida verdadeira durante a apresentação. A Tabela 3 ilustra um exemplo de conector de restrição expressando uma relação de sincronização espacial especificando que “dois nós devem ser alinhados pelo topo (o atributo *top* de seus descritores devem ser idênticos)”.

Tabela 3 – Exemplo de conector de restrição

Tipo de Papel e Id	Tipo do Evento	Cardinalidade (min,max)	Nome do Atributo
Avaliação P_1	<i>atribuição</i>	(1,1)	<i>descriptor.top</i>
Avaliação P_2	<i>atribuição</i>	(1,1)	<i>descriptor.top</i>

Tipo do Glue	Expressão-Assertiva
Restrição	$P_1 = P_2$

Uma expressão de disparo de um *glue* causal também pode ser simples ou composta. Uma *expressão de disparo simples* (*simple trigger expression*) se refere a um papel do tipo condição. Uma *expressão de disparo composta* (*compound trigger expression*) consiste de uma expressão lógica, baseada nos operadores *and* ou *or*, envolvendo duas ou mais outras expressões de disparo ou de assertiva. Uma expressão de disparo composta pode ser opcionalmente negada. Qualquer expressão de disparo (simples ou composta) pode especificar retardos mínimo (*minimum delay*) e máximo (*maximum delay*) para sua avaliação. Por exemplo, dado que uma expressão de disparo C é verdadeira no instante t , C' definida com *minimum delay*="t1" e *maximum delay*="t2" é verdadeira no intervalo $[t+t1, t+t2]$.¹⁹

Expressões de disparo compostas podem relacionar qualquer número de papéis de condição e de avaliação (através das expressões-assertivas e expressões de condição). Entretanto, uma restrição é necessária para garantir a consistência de relações causais. Toda expressão de disparo associada a um conector causal deve ser satisfeita somente em um instante de tempo infinitesimal, exigindo que pelo menos um papel de condição de cada conector causal defina uma condição sobre uma transição de estado de um evento.

Uma *expressão de ações* (*action expression*) também pode ser simples ou composta. Uma *expressão de ações simples* (*simple action expression*) refere-se a um papel do tipo ação. Se o papel do tipo ação de uma expressão de ações simples é exercido por um evento do tipo de apresentação ou de atribuição, um atributo *repeat* pode ter seu valor imposto ao atributo repetições (*repetitions*) do evento. Um retardo entre repetições da ação (*repeat delay*) também pode ser especificado. Uma *expressão de ações composta* (*compound action expression*) consiste de uma expressão, baseada nos operadores *par*, *seq* ou *excl*, envolvendo duas ou mais outras expressões de ações. Expressões compostas de ações paralelas (*par*) ou seqüenciais (*seq*) especificam que a execução das ações deve ser feita em qualquer ordem ou em uma ordem específica, respectivamente. Expressões compostas de ações exclusivas (*excl*) especificam que somente uma das ações deve ser executada. No último caso, o formatador do documento deve decidir qual das ações deve ser disparada ou por sua conta ou com o auxílio do usuário.

¹⁹ Note que o comportamento temporal das relações NCM também pode ser obtido utilizando o nó de tempo (Section 2.3), ao invés de explorar os parâmetros de retardo do conector.

Uma *expressão de ações* pode também especificar um retardo (*delay*) que deve ser respeitado antes que a ação seja efetivamente executada.²⁰

Como dito anteriormente, quando a cardinalidade máxima de um papel for maior que um, vários participantes podem desempenhar o mesmo papel. Nesse caso, um qualificador (*qualifier*) deve ser especificado toda vez que esse papel for usado em expressões do *glue*. A Tabela 4 apresenta os possíveis valores para qualificadores.

Tabela 4 – Valores para os qualificadores dos papéis com cardinalidade máxima maior que um.

Tipo do Papel	Qualificador	Semântica
Condição	<i>all</i>	Todas as condições devem ser verdadeiras
Condição	<i>any</i>	Ao menos uma condição deve ser verdadeira
Avaliação	<i>all</i>	Todas as avaliações devem ser consideradas
Avaliação	<i>any</i>	Ao menos uma avaliação deve ser considerada
Ação	<i>par</i>	Todas as ações devem executar em paralelo
Ação	<i>seq</i>	Todas as ações devem executar em paralelo, mas respeitando a ordem que os participantes foram associados ao papel.
Ação	<i>excl</i>	Apenas uma das ações deve ser executada

A Tabela 5 ilustra um exemplo de conector causal expressando uma relação de sincronização temporal. A especificação do conector pode ser interpretada como “se um grupo de participantes estiver sendo apresentado (C_1) e outro participante for selecionado (C_2), pare a apresentação do grupo de participantes (A_1) e inicie a apresentação de outro participante (A_2)”. Para parar a apresentação do mesmo grupo de participantes que desempenhou o papel C_1 , um elo usando esse conector deve criar dois binds para cada participante do grupo, um para o papel C_1 e outro para o papel A_1 .

Tabela 5 – Exemplo de conector causal.

Tipo do Papel e Id	Tipo do Evento	Cardinalidade (min,max)	Condição	Ação
Condição C_1	<i>apresentação</i>	$(1, \text{unbounded})$	<i>state=occurring</i>	
Condição C_2	<i>seleção</i>	$(1, 1)$	<i>transition=stops</i>	
Ação A_1	<i>apresentação</i>	$(1, \text{unbounded})$		<i>stop</i>
Ação A_2	<i>apresentação</i>	$(1, 1)$		<i>start</i>

Tipo do Glue	Expressão de Disparo	Expressão de Ações
Causal	$all(C_1) \text{ AND } C_2$	$seq(par(A_1), A_2)$

Como a definição de conectores não é simples de ser feita por um usuário leigo, pois ele precisaria conhecer os conceitos de estados e transições de estados de eventos, a idéia é fazer com que usuários experientes definam conectores, os armazenem em bibliotecas, chamadas de *bases de conectores (connector bases)*, e as tornem disponíveis para a criação de elos.²¹

²⁰ Uma aplicação baseada no NCM pode permitir a parametrização desse valor e de outros atributos presentes nas expressões de um glue e nos papéis. Na linguagem NCL, por exemplo, uma mesma especificação de conector pode ser reusada, com valores diferentes de parâmetros derivando conectores NCM diferentes. De fato, a parametrização NCL é usada não apenas para atributos de ações, mas também para atributos de condições e de avaliações, especificados tanto nos papéis do conector quanto nas expressões de um glue. Parametrização, contudo, é uma questão de implementação e não de modelo. Por isso, ela é deixada para as definições das aplicações, como discutido na Part5: NCL (Nested Context Language).

²¹ Como exemplo de base de conectores, considere um conjunto contendo as treze famosas relações de sincronização temporal propostas por Allen [Alle83]. Apesar de Allen ter especificado um conjunto

2.8.2. Binds de Elos

Conforme já mencionado, elos são definidos em uma composição (na realidade, em um nó de contexto). Um elo referencia um conector e define um conjunto de binds que associam cada extremidade do elo (interface dos nós) a um papel do conector referenciado. Binds são limitados a conectar interfaces de nós que estejam diretamente contidos em uma composição C onde o elo é definido, ou âncoras da própria composição C . No entanto, uma vez que a porta de um nó de composição pode ser mapeada para a porta de um outro nó de composição interno, e assim por diante até que a âncora de um nó seja alcançada, elos definidos em uma composição C podem indiretamente associar aos papéis do seu conector, eventos definidos em qualquer nó recursivamente contido em C . Isso traz a noção de elos visíveis e elos contextuais, definidos a seguir.

Lembrando que no NCM nós de composição diferentes podem conter o mesmo nó, e que elos podem ser definidos em qualquer composição da perspectiva de um nó, é necessário identificar quais elos efetivamente ancoram em um nó, ou passam por um nó (no caso de nós de composição), em uma dada perspectiva. Ao conjunto de elos que ancoram em um nó, em uma dada perspectiva P , chamamos de *elos contextuais de P* ; ao conjunto de elos que ancoram ou passam por um nó de composição, em uma dada perspectiva, chamamos de *elos visíveis de P* .

Mais precisamente, dado um nó N_l e uma perspectiva $P = (N_m, \dots, N_l)$, com $m > 0$:

1. Um elo l é *visível em P* se e somente se existe um nó de composição N_i , para $i \in [1, m]$, tal que N_i ocorre em P , N_i contém l e:
 - i) se $i > 1$ então $(N_{i-1}, p, \dots, N_l, p_l)$ define uma seqüência de mapeamentos de uma porta (Seção 2.4) p da composição N_{i-1} , e p é diretamente associada a um papel do conector usado por l ;
 - ii) senão, N_l é um nó que possui uma interface diretamente associado a um papel do conector usado por l .
2. Um elo l é *contextual em P* se e somente se existe um nó de composição N_i , para $i \in [1, m]$, tal que N_i ocorre em P , N_i contém l e:
 - i) se $i > 1$ então $(N_{i-1}, p, \dots, N_l, p_l)$ define uma seqüência de mapeamentos de uma porta p da composição N_{i-1} , N_l contém uma âncora que pertence à seqüência de mapeamentos da porta p , e p é diretamente associada a um papel do conector usado por l ;

completo de todas as relações possíveis entre dois intervalos temporais, elas não expressam, precisamente, a semântica causal ou de restrição que deveria existir entre os intervalos [DuKe95]. Por exemplo, a relação chamada *meet* especifica que o fim de um intervalo x deve coincidir com o início do intervalo y , o que dá margem a várias interpretações diferentes. A relação *meet* poderia ser considerada como uma simples restrição, ou então uma causalidade onde o término de x provoca o início de y , ou ainda uma causalidade onde o início de y provoca o término de x . Ao especificar relações com conectores NCM, o autor pode escolher e definir a semântica exata que deseja expressar através da relação, permitindo uma definição sem ambigüidades.

ii) senão, N_l é um nó que possui uma âncora diretamente associada a um papel do conector usado por l .

Por exemplo, suponha que os nós A e Z contêm o nó B , que por sua vez contém os nós C , D , E e F , com elos ilustrados na Figura 9. Então, a exibição do nó C , através da perspectiva (A, B, C) , vai mostrar um elo da âncora i de C para a âncora j de E e um elo da âncora m de C para âncora n de F , definidos em A e B , respectivamente. Ambos são elos contextuais e visíveis em (A, B, C) . A exibição do nó B , através da perspectiva (A, B) , vai mostrar um elo de B para B , que é definido no nó A . Esse é o único elo visível em (A, B) ; não há elos contextuais em (A, B) .

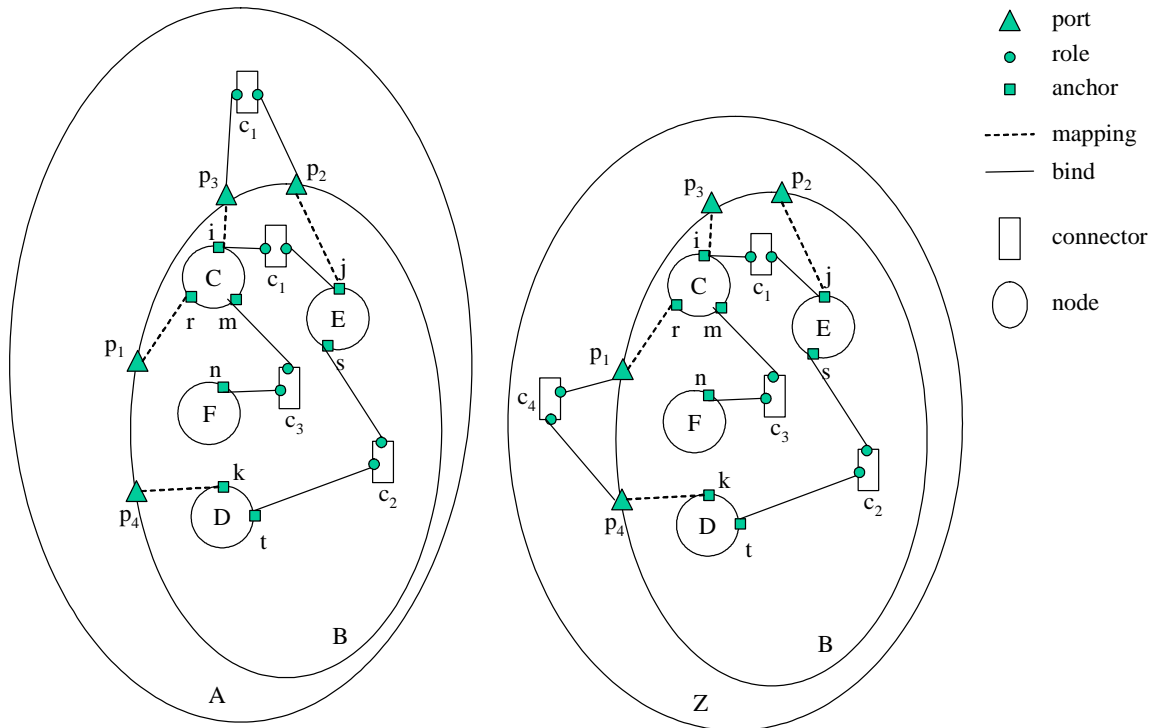


Figura 9 – Exemplos de elos visíveis em contextuais no NCM.

Por outro lado, a exibição do nó C , através da perspectiva (Z, B, C) , vai mostrar um elo a partir da âncora r de C para a âncora k de D e um elo a partir da âncora m de C para a âncora n de F , definidos em Z e B , respectivamente. Ambos são elos contextuais e visíveis em (Z, B, C) . A exibição do nó B , através da perspectiva (Z, B) , vai mostrar um elo de B para B , definido no nó Z . Esse é o único elo visível em (Z, B) , não havendo elos contextuais nessa perspectiva.

Binds de um elo possuem outros atributos além daqueles utilizados para associar uma interface a um papel: *descriptor* e *embed*. Um atributo *descriptor* é opcional e especifica um descriptor genérico para o nó associado com o papel do conector. Se o nó associado for um nó de composição N , o descriptor deve ser nulo. Note que vários binds para o mesmo nó de conteúdo com diferentes descriptors, levam a apresentações simultâneas do mesmo nó com diferentes características de exibição, similar ao que é proporcionado com o grupo de descriptors e a navegação em profundidade discutida na Seção 2.4.

O atributo *embed* é um atributo booleano que só deve ser usado na associação entre um papel de ação (somente para as ações *start* ou *prepare*) com um evento de apresentação *E*. Se o descritor do bind referenciar uma ferramenta de exibição (Seção 2.10) que já esteja sendo utilizada para controlar a apresentação de um outro evento *F*, a ferramenta de exibição é solicitada a também controlar *E*, sem parar *F*, se o atributo *embed* for verdadeiro, caso contrário, se *embed* for igual a falso, a ferramenta de exibição deverá ser solicitada a substituir (parar) *F* por *E*. Quando não especificado esse atributo deve ser considerado como igual a falso.

A definição de portas-switch apresentada na Seção 2.6 permite que elos (relacionamentos) sejam definidos ancorando em nós switch, independente do nó que será selecionado, como ilustrado na Figura 3. As portas de composição tradicionais permitem que um elo seja associado a uma alternativa específica. Se essa alternativa não for selecionada, o elo será simplesmente ignorado durante a apresentação do documento.

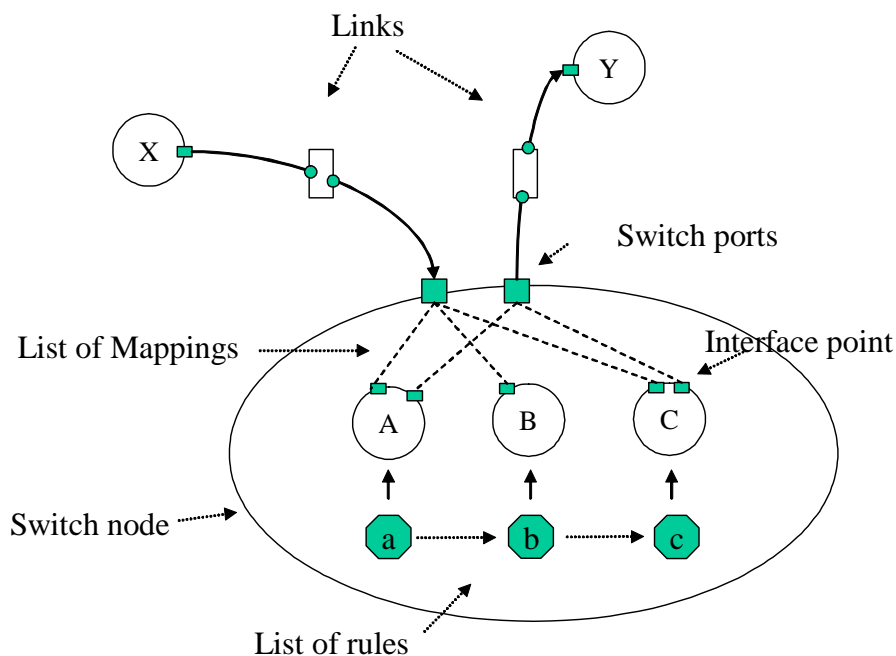


Figura 3 – Nós switch, portas-switch e elos.

Baseado nas entidades do modelo nós switch, portas-switch e binds de elos, um autor pode especificar documentos que contemplem adaptação de elos. Para isso, basta colocar em um mesmo nó switch cópias de um mesmo nó, com diferentes elos associados às várias ocorrências do nó. A Figura 4 oferece um exemplo de uso das alternativas para adaptar os relacionamentos. No exemplo, o elo que inicia a apresentação do nó *Z* só estará habilitado no documento se a regra *a* for avaliada como falsa e a regra *b* for avaliada como verdadeira.

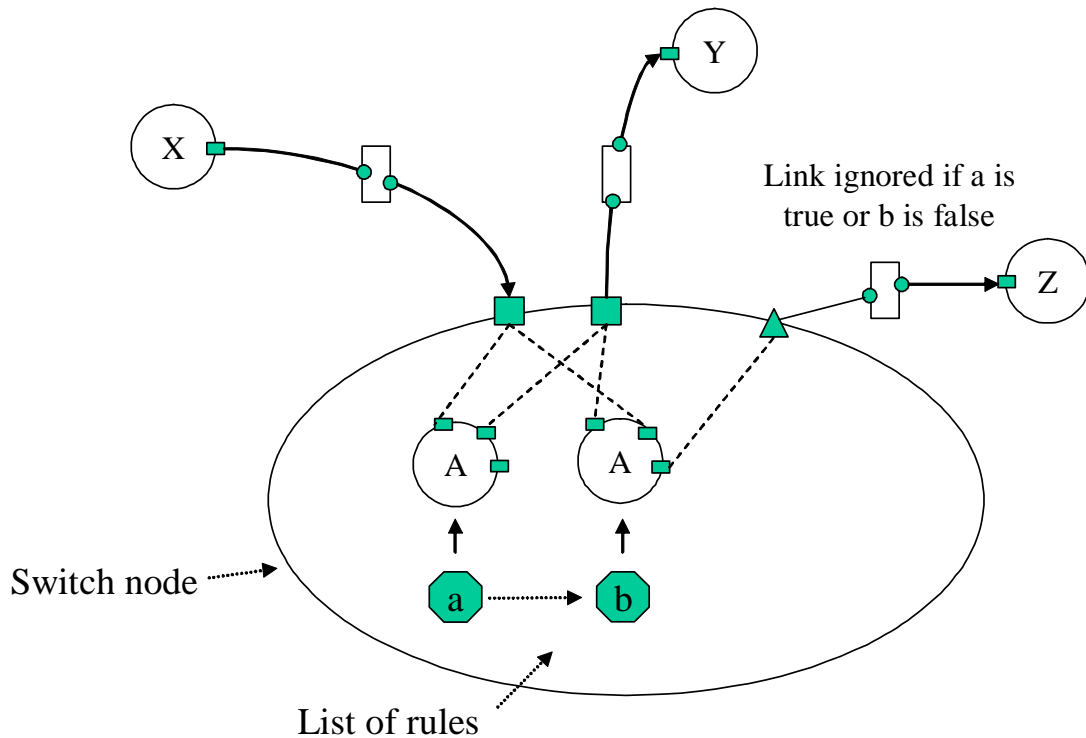


Figura 4 – Suporte à adaptação de elos no NCM.

2.9. Objetos de Dados versus Objetos de Representação

O NCM define um *objeto de dados* (*data object*) como uma entidade que compreende um nó NCM e todas as operações para manipulação desse nó, exceto as operações relacionadas com a apresentação do seu conteúdo (suas unidades de informação, conforme definido na Seção 2.2). A funcionalidade de apresentar o conteúdo do nó é dada por uma instância de descritor que deve ser associada ao nó (objeto de dados).

A agregação de um objeto de dados e um descritor NCM é chamada um *objeto de representação*. A associação entre objetos de dados e descritores é representada na Figura 5 por linhas conectando os objetos de dados no plano intermediário com os objetos de representação, desenhados no plano superior. Na figura, nós são representados por círculos, elos por arcos e composições pela inclusão de círculos e arcos em círculos maiores.

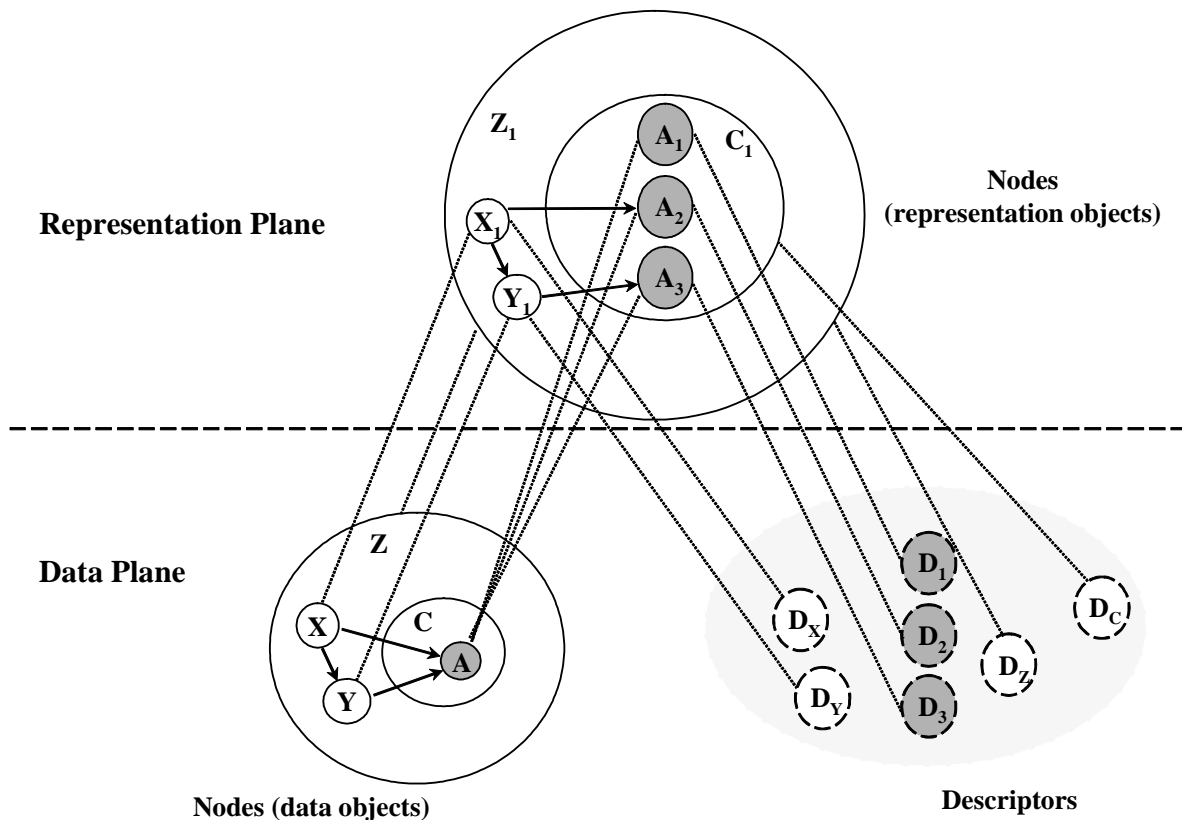


Figura 5 – Associação entre objetos de dados e descritores, gerando objetos de representação.

Note que um nó (objeto de dados) pode ser combinado a diferentes descritores, originando diferentes representações (objetos de representação) da mesma entidade. A figura mostra essa característica com a associação dos descritores D_1 , D_2 e D_3 ao objeto de dados A , criando os objetos de representação A_1 , A_2 e A_3 . O nó A possui três diferentes representações porque existem, por exemplo, três diferentes formas de navegação até ele, através dos dois elos ou através da hierarquia de composições. Note assim que, devido ao fato de um mesmo objeto de dados poder gerar vários objetos de representação, um contexto objeto de representação pode conter um número de elementos diferente do contexto objeto de dados, por exemplo, o contexto C_1 da figura possui três nós, ao passo que o nó objeto de dados correspondente (contexto C) só tem um.

Pelas definições do NCM, um descritor pode ser definido em uma propriedade do nó, em um grupo de descritores da composição que contém o nó ou como um atributo de binds entre o nó e elos. Além disso, descritores default podem ser especificados para as classes dos nós (texto, imagem etc.), ou explicitamente sugeridos pelos usuário leitor do documento. Se um nó possuir mais de um desses descritores especificados, o sistema de apresentação (formatador de documentos NCM) deverá construir um descritor resultante baseado na regra de cascadeamento definida a seguir.

Suponha um nó N da classe C com a perspectiva corrente ($C_k \dots, C_l, N$) alcançada através de uma navegação por elo (elo l). Seja D_1 um descritor definido para a classe do nó C , D_2 o descritor definido pela propriedade descritor de N , D_3 o único membro do grupo de descritores especificado para N em C_l , e D_4 um descritor especificado no bind feito para associar N ao conector referenciado pelo elo l . O descritor resultante será formado pela soma de todos os atributos/propriedades dos seguintes descritores (D_1, D_2, D_3, D_4). Se dois ou mais descritores definirem o mesmo

atributo/propriedade com diferentes valores, D_4 (descriptor do elo) terá prioridade sobre D_3 (descriptor da composição), que terá prioridade sobre D_2 (descriptor do nó), que terá prioridade sobre D_1 (descriptor da classe). Se o usuário especificar um quinto descriptor ao navegar para N , esse descriptor será incluído na lista de cascadeamento com precedência sobre D_4 .

2.10. Descritores Genéricos, Descriptores e Switches de Descritores

Conforme mencionado anteriormente, descritores NCM agrupam informações das características de apresentação, objetivando separar essas informações do conteúdo do documento e de sua estrutura.

NCM define uma classe *descriptor genérico* (*generic descriptor*) que é especializada nas classes *descriptor* (*descriptor*) e *switch de descritores* (*descriptor switch*). O descriptor NCM é uma entidade que possui como propriedades adicionais: uma *especificação de início de apresentação*, uma *especificação de fim de apresentação* e uma *coleção de descrições de eventos*.

A propriedade *especificação de início de apresentação* deve conter um identificador da ferramenta de exibição (*player*) utilizada pelo formatador. Essa ferramenta será responsável por processar e exibir o conteúdo do nó durante a apresentação. Quando essa propriedade não estiver definida ou o seu valor apontar para uma ferramenta inexistente no formatador, o formatador de documentos deve escolher um exibidor default com base no tipo de conteúdo do nó.

A especificação de início de apresentação pode também conter um atributo especificando se uma nova ferramenta de exibição deve ser instanciada ou se uma ferramenta já instanciada pode ser usada. Além disso, a especificação de início de apresentação pode conter um *conjunto de atributos particulares* que serão interpretados pelo exibidor selecionado para controlar a apresentação do nó. Alguns exemplos desses parâmetros são:

- para nós com apresentações visíveis, como nós de texto, vídeo e imagem, um parâmetro *dispositivo* (*device*) especifica o dispositivo onde a apresentação ocorrerá; um parâmetro *região espacial* (*spatial region*) especifica uma localização para apresentar o conteúdo do nó, identificando a posição na tela do dispositivo especificado; etc. Quando esses parâmetros não forem especificados, o formatador deve escolher um dispositivo e uma área default para apresentação do nó.
- para nós de áudio, um parâmetro *dispositivo* (*device*) especifica o dispositivo de áudio onde o som deve ser apresentado; um parâmetro *volume* especifica o volume inicial de apresentação; etc.²² Quando esses parâmetros não forem especificados, o formatador deve escolher um dispositivo e uma área default para apresentação do nó.
- Para um nó de texto que deva ser exibido por uma ferramenta TtS (*text to speech*), parâmetros podem especificar a voz desejada, o sotaque etc.

²² O autor pode escolher exibir alguma informação visual associada ao áudio, por exemplo, uma barra de progressão temporal. Nesse caso, os parâmetros descritos no primeiro item também poderiam ser especificados.

A propriedade *especificação de fim de apresentação* define ações que devem ser executadas ao final de uma apresentação. Ela deve conter um atributo especificando o que acontecerá com a ferramenta de exibição ao final da apresentação, isto é, se a ferramenta será fechada, ou permanecerá aberta, pronta para uma próxima apresentação. Nesse último caso, deve também ser especificado se a ferramenta de exibição permanecerá escondida ou não.

Uma *descrição de um evento*, em um descritor, por sua vez, consiste da tupla $\langle Id\grave{A}ncora, Dur\grave{E}xp\grave{L}icita, Fun\grave{c}\tilde{a}oCusto, Rep \rangle$. O parâmetro *IdÂncora* é um identificador de uma âncora do objeto de dados (nó) ao qual o descritor será associado para a formação do objeto de representação (essa âncora pode ser genericamente identificada por um rótulo ou por sua posição na lista de âncoras para permitir que um mesmo descritor seja reusado por mais de um nó). *DurExplícita* é opcional, somente pode ser utilizada para eventos do tipo exibição, e sobrescreve a duração de apresentação ideal intrínseca ao conteúdo do nó. *FunçãoCusto* também é opcional, também só pode ser usada em eventos do tipo exibição, e especifica uma métrica para guiar o formatador em ajustes que precisem ser feitos na duração de apresentação da âncora do nó. *Rep* especifica um valor para iniciar o atributo repetições do evento (Seção 2.7). Em particular, todo descritor contém ao menos a descrição de evento $\langle \lambda, Dur\grave{E}xp\grave{L}icita, Fun\grave{c}\tilde{a}oCusto, Rep \rangle$ associada a sua âncora de conteúdo total, onde, como já mencionado, *DurExplícita*, *FunçãoCusto* são opcionais.

De forma similar à entidade nó switch (Seção 2.6), o switch de descritores contém uma lista ordenada de regras, estando cada regra associada a um descritor. O formatador de documentos deve percorrer a lista avaliando cada regra e selecionando o primeiro descritor cuja regra tiver sido satisfeita. Se nenhuma das regras for verdadeira, o switch de descritores pode especificar um *descriptor default* para ser selecionado.

A entidade switch de descritores permite que formatadores de documentos NCM adaptem características da apresentação dos nós independente das informações estruturais e dos conteúdos. Obviamente, ambos os tipos de flexibilidade (adaptação de apresentação e adaptação de conteúdo) podem ser combinadas. Juntas com os ajustes de duração, essas características do modelo proporcionam um suporte flexível para que autores especifiquem documentos e apresentações de documentos hipermídia adaptativos.

2.11. Trilhas

Características como grande número de nós e grande número de elos, muitas mudanças no documento, tempo de resposta ruim para as ações do usuário, diferenças visuais insuficientes entre elos e nós, e usuários não orientados visualmente se combinam para dificultar os mecanismos de navegação em um documento. Usuários desorientados precisam de informações de escopo para restabelecerem a noção de localização. Em particular, informações do escopo temporal são necessárias para responder a questões do tipo “como eu cheguei aqui?” Essas questões encontram suas respostas com a introdução do conceito de trilha.

Dada uma composição *C*, do tipo nó de contexto, base privada ou hiperbase pública,²³ uma *trilha T para C* é um nó de composição cujo conteúdo é uma lista ordenada de nós de conteúdo, nós de contexto ou outros nós de trilha, tais que: todos os

²³ Bases privadas e hiperbase pública serão definidas na próxima seção.

nós, que não são trilhas, estão recursivamente contidos em C , e todos os seus nós de trilhas são trilhas para C . Mais ainda, T tem um atributo básico adicional denominado *nó-corrente*, cujo valor indica a posição de um nó na lista ordenada de T , chamado de *entidade corrente de T* . Trilhas possuem um outro atributo básico adicional denominado *visão*, cujo valor associa a cada ocorrência de um nó N , na lista de T , um aninhamento de nós (N_m, \dots, N_1) , com $m \geq 1$, e um descritor D , tal que: $N_1 = N$, $N_m = C$, N_{i+1} é um nó de composição, N_i está contido em N_{i+1} , para $i \in [1, m)$. Diz-se que a trilha T é *associada a C* .

Toda trilha deve implementar o método deferido da classe composição:

- **Inserir nó:** insere um nó na lista de nós da trilha, em uma posição especificada, com uma visão associada.

Adicionalmente, toda trilha deve implementar os seguintes métodos:

- *próximo:* se o atributo *nó-corrente* não apontar para o último nó, incrementa o atributo *nó-corrente*;
- *anterior:* se o atributo *nó-corrente* não apontar para o primeiro nó, decrementa o atributo *nó-corrente*;
- *primeiro:* coloca o atributo *nó-corrente* apontando para o primeiro nó da lista;
- *último:* coloca o atributo *nó-corrente* apontando para o último nó da lista;
- *ativa:* habilita os métodos *próximo*, *anterior*, *primeiro* e *último*; e inibe os métodos, *insere nó* e *retira nó*;
- *desativa:* desabilita os métodos *próximo*, *anterior*, *primeiro* e *último*; e habilita os métodos *insere nó* e *retira nó*.

A razão dos métodos *ativa* e *desativa* é não permitir que uma trilha seja usada ao mesmo tempo para navegação (pela trilha) e para manter o histórico da navegação. Ou ela é usada para a realização de uma tarefa ou da outra.

Note que um nó pode aparecer mais de uma vez na lista ordenada de T . Além disso, cada ocorrência do nó é associada com um aninhamento de nós relativos a C . Dessa forma, trilhas são úteis para linearização de documentos hipermídia e para implementação de *viagens guiadas*. Como no sistema Intermédia [YaMe85], uma *trilha especial do sistema* pode manter a história de navegação do usuário durante uma sessão de trabalho, de tal modo que o usuário possa mover-se aleatoriamente entre os nós e depois recordar a navegação realizada. No entanto, se um nó N contido em um nó switch S pertencer a uma trilha, esse nó deverá obrigatoriamente ser a alternativa selecionada de S quando o usuário navegar pela trilha, independente se a regra associada ao nó não for mais verdadeira no momento da navegação através da trilha.

A definição de trilhas é importante para sistemas hipermídia, pois elas representam travessias ordenadas. Através delas, os autores podem fornecer ordens de leitura, que auxiliam leitores não familiarizados com o material informativo a navegar pelo hiperdocumento, ou determinar uma ordem apropriada de apresentação para uma certa audiência. Os usuários sentem-se menos desorientados quando seguem uma trilha já definida, pois têm limitado o número de opções que possuem para percorrer o documento. Note, no entanto

Seguindo o modelo NCM, uma implementação possível para trilhas que mantenham o histórico de navegação é criar uma *trilha principal* (ou *trilha do sistema*).

Toda vez que um usuário navegar para um nó, esse nó, sua perspectiva corrente, e o descritor resultante (descritor cascadeado) são inseridos na trilha principal pelo sistema. Se o usuário decidir navegar através da trilha, é criada uma cópia da trilha principal e o método ativa é chamado sobre essa cópia. Mesmo durante a navegação nessa cópia, a trilha especial do sistema é atualizada, de forma a sempre manter o histórico da navegação. Se o usuário fizer qualquer navegação fora da dada pela trilha cópia, a cópia é destruída.

2.12. Hiperbase Pública e Bases Privadas

A hiperbase pública é um conceito do NCM que representa o repositório público global das entidades disponíveis em um sistema hipermídia. A *hiperbase pública* (*public hyperbase*) é um nó de composição único que, estando um nó de composição *C* nele contido, então todos os nós recursivamente contidos em *C* devem também pertencer à hiperbase. A composição hiperbase pública tem como propriedade básica adicional um conjunto de descritores. Os descritores desse conjunto são aqueles utilizados para criação dos objetos de representação (veja Seção 2.9), a partir do conjunto de nós contidos na hiperbase pública.

Uma *base privada* é um tipo especial de nó de composição, tal que:

- i) ela pode conter nós de conteúdo, nós de contexto, nós switch, trilhas e outras bases privadas;
- ii) uma base privada pode pertencer a no máximo uma base privada;
- iii) se um nó de composição estiver contido em uma base privada *PB*, seus componentes podem estar ou contidos em *PB*; ou contidos na hiperbase pública; ou contidos em alguma base privada recursivamente contida em *PB*.

Bases privadas possuem como propriedades básicas adicionais um conjunto de elos e um conjunto de descritores. Cada elo contido no conjunto de elos de um nó base privada *PB* define um relacionamento entre nós recursivamente contidos em *PB*. Os descritores no conjunto de descritores de uma base privada são aqueles utilizados para criar os objetos de representação (veja Seção 2.9) a partir dos nós recursivamente contidos na base privada.

Intuitivamente, uma base privada reúne todas as entidades utilizadas durante uma sessão de trabalho do usuário.

3. Considerações Finais

Com o objetivo de oferecer um modelo hipermídia escalável, com características que possam ser progressivamente incorporadas nas implementações dos sistemas hipermídia, o NCM foi dividido em várias partes. Este relatório técnico lida com as entidades básicas do modelo que forma o núcleo do NCM.

Por fim, é importante mencionar mais uma vez que é possível ter implementações do modelo que ignorem algumas das entidades básicas (ou alguns atributos), mas isso não é relevante na definição desta versão do modelo.

Acknowledgements

Muitas pessoas contribuíram na definição do modelo NCM. Dentre elas merecem destaque Marco Antônio Casanova e Débora Muchaluat, que trabalharam na definição do modelo por quase uma década.

Referências

- [Alle83] Allen J.F. “Maintaining Knowledge about Temporal Intervals”. Communications of the ACM, 26(11), November 1983, pp. 832-843.
- [BMRS04] Bachelet B., Mahey P., Rodrigues R.F., Soares L.F.G. “Elastic Time Computation in QoS-Driven Hypermedia Presentations”, Technical Report of TeleMídia Lab., Departamento de Informática, PUC-Rio, Brazil, May 2004.
- [DuKe95] Duda A., Keramane C. “Structured Temporal Composition of Multimedia Data”. Proceedings of the IEEE International Workshop on Multimedia Database Management Systems, Blue Mountain Lake, USA, August 1995.
- [HaSc90] Halasz F.G., Schwartz M. “The Dexter Hypertext Reference Model”. NIST Hypertext Standardization Workshop. Gaithersburg. January 1990.
- [MuSo01] Muchaluat-Saade D.C., Soares L.F.G. Hypermedia Spatio-Temporal Synchronization Relations Also Deserve First Class Status, VIII Multimedia Modeling Conference - MMM'2001, Amsterdam, Netherlands, November 2001.
- [MuRS02] Muchaluat-Saade D.C., Rodrigues R.F., Soares L.F.G. “XConnector: Extending XLink to Provide Multimedia Synchronization”. ACM Symposium on Document Engineering - DocEng'02, Virginia, USA, November 2002.
- [PéLi96] Pérez-Luque M.J., Little T.D.C. “A Temporal Reference Framework for Multimedia Synchronization”. IEEE Journal on Selected Areas in Communications (Special Issue: Synchronization Issues in Multimedia Communication), 14(1), January 1996, pp. 36-51.
- [Rodr03] Rodrigues R.F. “Formatação e Controle de Apresentações HiperMídia com Mecanismos de Adaptação Temporal”. PhD Thesis, Departamento de Informática, PUC-Rio, March 2003.
- [Soar00] Soares L.F.G. et al. “Modelo de Contextos Aninhados versão 2.2”, Relatório Técnico do Laboratório TeleMídia, PUC-Rio, Rio de Janeiro, Brazil, 2000. (*in Portuguese*)
- [SoCR95] Soares L.F.G., Casanova M.A., Rodriguez N.R. “Nested Composite Nodes and Version Control in an Open Hypermedia System”, International Journal on Information Systems; Special issue on Multimedia Information Systems, 20(6):501-520, Elsevier Science Ltd. England, September 1995.
- [SoCR91] Soares L.F.G., Casanova M.A., Rodriguez N.R. Modelo de Contextos Aninhados. Relatório Técnico PUC-Rio - Departamento de Informática. Rio de Janeiro. May de 1991. (*in Portuguese*)
- [SSRM99] Soares L.F.G., Souza G.L., Rodrigues R.F., Muchaluat-Saade D.C. “Versioning Support in the HyperProp System”. Multimedia Tools & Applications, 8(8), May 1999.
- [XLin01] “XML Linking Language (XLink) Version 1.0”, W3C Recommendation, June 2001.

Apêndice A : Exemplos de Uso de Elos NCM

Considere um documento de trabalho do Grupo de Pesquisa de Poesias Inglesas do Século XVI. Suponha que o documento é modelado como um contexto E contendo um contexto S , agrupando peças de Shakespeare, e um outro contexto M , agrupando sonetos de Christopher Marlowe. Suponha também que S contém os nós de texto H e L representando as peças “Hamlet” e “King Lear” e que M contém um nó de texto F representando “Dr. Faustus”. Pode ser que o grupo deseje registrar uma conexão entre “Hamlet” e “Dr. Faustus” (tal elo poderia ser usado, por exemplo, para registrar uma conexão entre peças onde o tema principal fosse o conflito). Esse elo, chamado l_1 , poderia ser definido com o seguinte comportamento: quando o usuário selecionar uma âncora i (por exemplo, uma frase) de H , a frase onde o mesmo conceito aparece pela primeira vez (âncora j) em F é apresentada, substituindo a exibição de H . Esse elo pode ser definido em E , mas requer antes a criação de um conector hipermídia e a especificação de duas portas nas composições S e M . A Tabela 6 mostra a especificação do conector cH , enquanto a Figura 6 ilustra a organização do contexto que representa o documento (sem os elos), assim como as portas criadas e os mapeamentos.

Tabela 6 – Exemplo de conector causal para um hiperelo

Tipo do Papel e Id	Tipo do Evento	Cardinalidade (min,max)	Condição	Ação
Condição C	<i>seleção</i>	(1, 1)	<i>transition=stops</i>	
Ação A_1	<i>apresentação</i>	(1, 1)		<i>stop</i>
Ação A_2	<i>apresentação</i>	(1, 1)		<i>start</i>

Tipo do Glue	Expressão de Trigger	Expressão de Ação
Causal	C	$seq(A_1, A_2)$

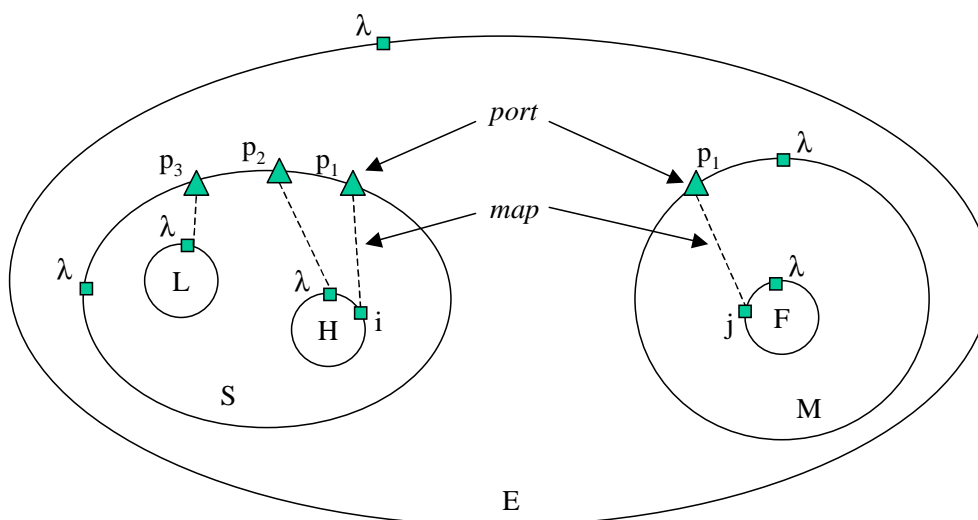


Figura 6 – Exemplo do documento do Grupo de Pesquisa de Poesias Inglesas do Século XVI

O elo l_1 poderia então ser criado em E usando o conector cH e estabelecendo três binds: $\langle (S, p_1), C \rangle$, $\langle (S, p_2), A_1 \rangle$ e $\langle (M, p_1), A_2 \rangle$, conforme ilustrado na Figura 7.

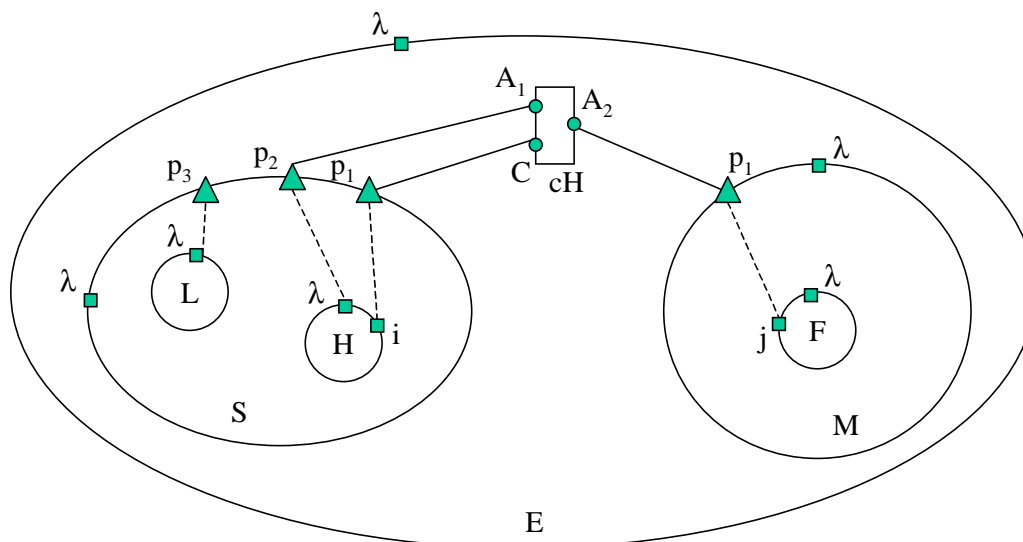


Figura 7 – Criação de elo no exemplo do documento do Grupo de Pesquisa de Poesias Inglesas do Século XVI.

Em outro exemplo, ilustrado na Figura 8, uma vez que S contém H e L , um elo conectando esses nós poderia, em princípio, ser definido em S com um conector c_1 e os seguintes binds para esses conector: $\langle (H, \lambda), C \rangle$ e $\langle (L, \lambda), A \rangle$, onde C e A são identificadores de papéis definidos em c_1 . Se fosse criado um elo em E conectando H e L usando o mesmo conector, o elo teria os seguintes binds: $\langle (S, p_2), C \rangle$ e $\langle (S, p_3), A \rangle$. Note a diferença em se definir o elo em S e em E . O elo definido em S será herdado por todos os documentos que contiverem S (o contexto que agrupa as peças de Shakespeare será provavelmente compartilhado por vários documentos), enquanto que o elo definido em E será herdado apenas para os leitores do documento E .

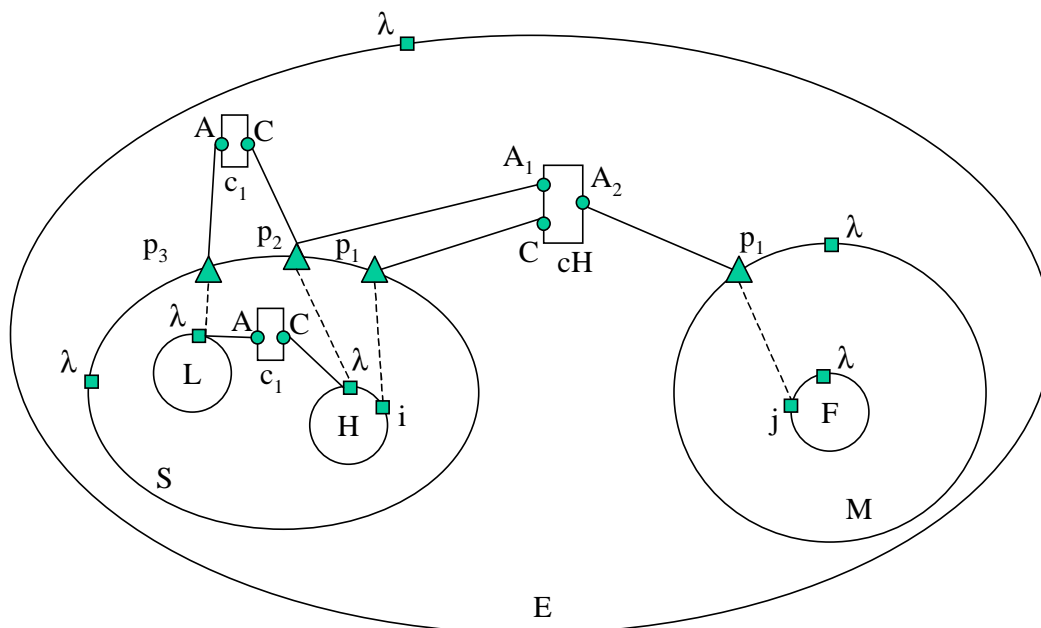


Figura 8 – Definição de elos com o mesmo comportamento em contextos diferentes.

A Figura 9 oferece uma visão menos poluída do documento apresentado na Figura 8. No desenho, conectores e mapeamentos de portas são implicitamente

definidos nas setas dos elos. Uma vez que são utilizados somente conectores causais, é possível especificar um sentido das condições para as ações (setas dos elos).

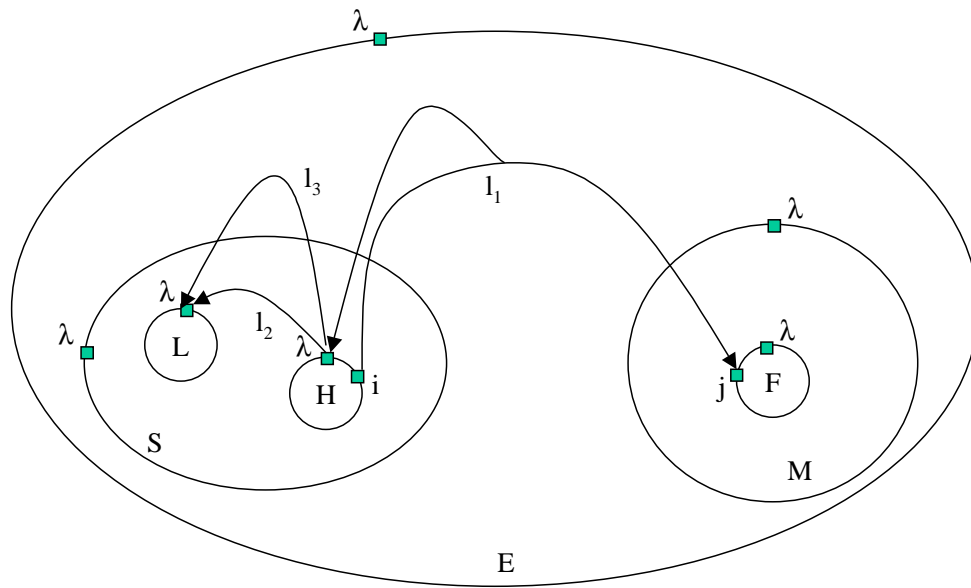


Figura 9 – Elos causais NCM ilustrados abstraído os conceitos de conectores e portas.