Relações de Sincronização Espaço-Temporal Hipermídia Também Merecem Status de Primeira Classe

Débora Christina Muchaluat-Saade
debora@telemidia.puc-rio.br

Sérgio Colcher
colcher@inf.puc-rio.br

Luiz Fernando Gomes Soares
lfgs@inf.puc-rio.br

PUC–RioInf.MCC25/01 August, 2001

Abstract: This paper introduces the concept of hypermedia connector, presenting how spatio-temporal relations may be defined as a specialization of architecture description language (ADL) connectors. Some features found in ADLs were brought to hypermedia authoring languages, giving first-class status to synchronization relations expressed by links. Among the profits, we can highlight the possibility of reusing a spatio-temporal relation specification in different links with the same behavior and the possibility of defining composite spatio-temporal relations expressed by links, which is an original feature in hypermedia authoring languages.

Keywords: links and connectors, spatio-temporal relations, hypermedia authoring languages, architecture description languages (ADL)

Resumo: Este artigo introduz o conceito de conector hipermídia, apresentando como relações de sincronização espaço-temporal podem ser definidas como uma especialização de conectores usados em linguagens de descrição de arquitetura (Architecture Description Language – ADL). Algumas características existentes em ADLs foram trazidas para linguagens de autoria hipermídia, tratando relações expressas por elos como entidades de primeira classe. Dentre os benefícios, podemos ressaltar a possibilidade de reutilizar a especificação de uma relação espaço-temporal em elos distintos que possuem o mesmo comportamento, e a possibilidade de definir relações espaço-temporais de composição expressas por elos, que é um recurso original em linguagens de autoria hipermídia.

Palavras-chave: elos e conectores, relações espaço-temporais, linguagens de autoria hipermídia, linguagens de descrição de arquitetura

1. Introdução

Linguagens de Descrição de Arquitetura (***A**rchitecture **D**escription **L**anguage* – ADL) são linguagens formais que podem ser usadas para representar a arquitetura de um sistema de software, definindo seus componentes, como eles se comportam e os padrões e mecanismos para interações entre esses componentes [ShGa96]. Normalmente, os blocos principais que constituem uma descrição arquitetural são [MeTa00]:

- componentes: unidades de computação ou armazenamento de dados que podem ser tão pequenos quanto um único procedimento ou tão grandes quanto uma aplicação inteira;
- conectores: usados para modelar interações entre componentes e regras que governam essas interações;
- configurações arquiteturais: grafos conexos de componentes e conectores que descrevem a estrutura da arquitetura. Composição hierárquica é desejável em ADLs, permitindo a descrição de um sistema em diferentes níveis de detalhe. Uma estrutura com um comportamento complexo pode ser representada explicitamente ou pode ser abstraída através de um componente ou conector único.

Em muitas ADLs, conectores desempenham um papel importante na descrição de uma arquitetura de software. Eles são tratados como entidades de primeira classe, ou seja, podem ser nomeados, subtipados e reutilizados [Shaw94, MeMP00], e seu poder de abstração é comparado ao dos componentes.

Existem algumas vantagens de tratar conectores como entidades de primeira classe, tais como:

- Independência – Definição de um conector independente de saber por quem ele é utilizado, ou seja, que configurações o contém e quais componentes interagem através dele.
- Reuso – Já que conectores têm uma definição independente, configurações diferentes podem usar o mesmo conector. Normalmente, uma configuração define instâncias de componentes e conectores, além de especificar quais componentes estão ligados a quais conectores.
- Composição – Conectores compostos representam grupos de conectores e componentes, modelando interações mais complexas entre componentes de uma arquitetura de software.

Estruturas de arquiteturas de software são similares a estruturas de documentos hipermídia. Fazendo uma análise superficial, pode parecer trivial fazer uma correspondência direta entre os elementos estruturais básicos de ADLs e os encontrados em linguagens de autoria hipermídia **(*H**ypermedia **A**uthoring **L**anguage*) – chamadas de **HAL** neste texto. Componentes são análogos a nós, conectores a elos e configurações a nós de composição.

No entanto, analisando com mais detalhes, podemos notar que a analogia não é tão direta e trivial. Existem algumas características interessantes em ADLs que não têm analogias em HALs e vice-versa. A referência [MuSo01] discute as diferenças principais entre ADLs e HALs, apresentando uma comparação detalhada entre algumas linguagens encontradas na literatura.

As diferenças principais entre ADLs e HALs normalmente estão relacionadas a conectores, que podem ser comparados a relações hipermídia expressas por elos. Essas relações não têm o mesmo tratamento que os nós hipermídia têm em HALs, devido às seguintes limitações:

- a definição das relações normalmente depende dos nós que estão relacionados;
- um nó hipermídia pode ser reutilizado em composições diferentes, mas esse não é o caso de relações expressas por elos na maioria das HALs. Em algumas delas, elos são embutidos no conteúdo do nó impendindo o reuso do nó sem os elos, tal como no HTML. Em algumas outras, elos estão contidos em repositórios de elos, que podem estar contidos em nós de composição, como na linguagem NCL [AMRS00]. Nesse caso, somente nós de composição podem ser reutilizados, mas não o conjunto de elos somente. Repositórios de elos também podem ser armazenados de forma independente, como, por exemplo, as bases de elos do sistema Microcosm [CRHH95] e da linguagem XLink [XLIN01]. Nesse caso, bases de elos podem ser reutilizadas, mas não a especificação de uma relação expressa por um elo;
- algumas HALs oferecem nós de composição, mas nenhuma oferece elos representando relações cuja semântica seja dada por composições de outros nós e elos.

O objetivo deste trabalho é dar às relações hipermídia espaço-temporais, expressas por elos, o mesmo tratamento que conectores possuem em ADLs, trazendo as vantagens existentes em ADLs para HALs. Relações hipermídia espaço-temporais também podem ser representadas por nós de composição especiais, tal como as composições paralelas e seqüenciais de SMIL [SMIL01] e NCL, entretanto, este artigo só trata de como expressar essas relações usando elos. Contudo, isso não representa uma limitação, visto que relações representadas por nós de composição podem ser traduzidas em relações representadas por elos, como discutido em [RRMS99]. Para uma discussão sobre o uso de composições e uso de elos para representar relações de sincronização, pode-se consultar a referência [SoRM00].

Além da comparação entre ADLs e HALs, [MuSo01] também define um metamodelo estrutural que pode ser usado para representar tanto arquiteturas de software como estruturas hipermídia. Apesar do metamodelo já oferecer o conceito de conector, ele só considera definições estruturais. Para uma descrição completa das relações hipermídia espaço-temporais, um modelo semântico também é necessário. Este artigo usa eventos como base para especificar a semântica de relações de sincronização expressas por elos, tornando esta proposta genérica para HALs.

O objetivo final do nosso trabalho é integrar as contribuições das áreas de hipermídia e arquitetura de software e usar os avanços de uma área na outra. Este artigo mostra como algumas facilidades encontradas em linguagens de descrição de arquitetura podem ser aplicadas em linguagens de autoria hipermídia, representando apenas um passo para a realização da meta final.

O artigo está organizado da seguinte forma. A Seção 2 apresenta uma descrição resumida do metamodelo estrutural proposto em [MuSo01]. A Seção 3 mostra como o metamodelo pode ser especializado para representar estruturas hipermídia. A Seção 4 apresenta uma definição detalhada dos conectores hipermídia espaço-temporais. A Seção 5 discute a proposta apresentada, comparando-a com trabalhos relacionados. Finalmente, a Seção 6 é dedicada aos comentários finais e trabalhos futuros.

2. O Metamodelo Estrutural

O metamodelo estrutural que pode ser usado para representar estruturas de arquiteturas de software e de documentos hipermídia é um caso especial de grafo composto.

O modelo define dois tipos de vértices, chamados *componentes (component)* e *conectores (connector)*, cada um contendo um conjunto de *pontos de interface*. O metamodelo também define um tipo básico de arco, chamado *bind*, conectando um ponto de interface de um componente a um ponto de interface de um conector. Binds não podem conectar diretamente componente a componente ou conector a conector [MuSo01]. Uma estrutura simples definida pelo metamodelo é ilustrada na Figura 1.

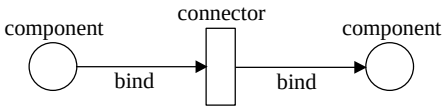


Figura 1 – Exemplo de estrutura simples

Em um grafo composto, vértices podem representar subgrafos também compostos de componentes, conectores e binds, que ainda podem estar aninhados em qualquer profundidade, como ilustrado na Figura 2.

Binds só podem ser definidos entre vértices que têm o mesmo pai ou entre vértices pai e filho. Para permitir conexões indiretas entre vértices dentro e vértices fora de uma composição, outro tipo de arco é definido, chamado *map*. Um map conecta vértices pai e filho de um mesmo tipo, tal como um componente/conector pai a um componente/conector filho. Conexões indiretas podem ser feitas por um caminho de arcos contendo maps. O uso de binds e maps também é ilustrado na Figura 2. É importante salientar que maps não representam interações entre componentes ou entre conectores. Eles servem apenas como meio para exportar pontos de interface internos para o exterior de um vértice composto.

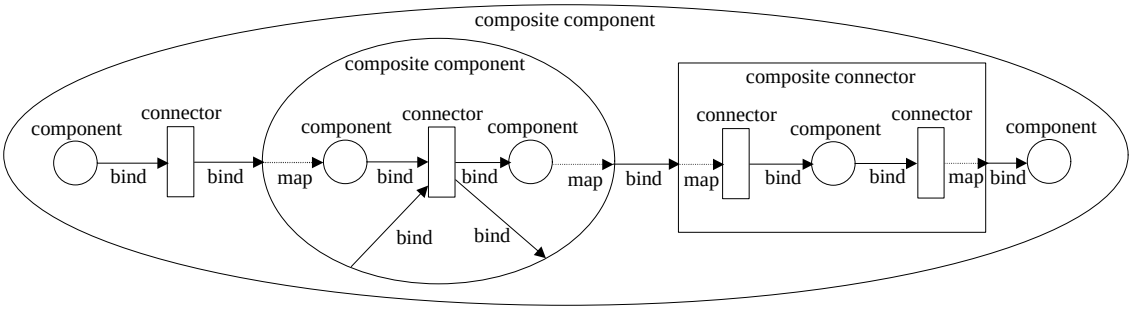


Figura 2 – Vértices compostos, binds e maps

Quando o metamodelo estrutural é aplicado a um domínio particular, algumas restrições provavelmente devem ser definidas para garantir a consistência da estrutura criada. Por exemplo, em um documento hipermídia representado por um componente composto, como veremos na próxima seção, para cada conector constituinte, deve existir pelo menos dois binds ligando esse conector a componentes do documento.

3. Representando Estruturas Hipermídia com o Metamodelo

Para aplicar o metamodelo estrutural ao domínio hipermídia, entidades de modelos hipermídia [HaSc94], tais como objetos de mídia, nós de composição e elos, devem ser descritas em termos de entidades do metamodelo.

Objetos de mídia e nós de composição encontrados em HALs são tipos especiais de componentes. Pontos na interface de um objeto de mídia são âncoras do nó. Pontos na interface de um nó de composição podem ser âncoras ou portas do nó, de acordo com o que segue.

Normalmente, uma âncora representa uma região marcada no conteúdo de um nó. No caso de um objeto de mídia, uma âncora pode ser qualquer conjunto de unidades de informação em seu conteúdo. Uma unidade de informação depende do tipo do objeto de mídia e pode ser um caracter em um objeto de mídia texto, um quadro em um objeto de mídia vídeo, etc. No caso de um nó de composição, uma âncora pode ser qualquer conjunto de seus constituintes. É importante manter âncoras para nós de composição hipermídia, pois um elo pode ancorar em um nó de composição diretamente, para iniciar a apresentação de sua estrutura e não a apresentação de seus componentes, por exemplo.

Uma *porta* de um nó de composição é usada para criar pontos de entrada e saída no nó, permitindo que elos externos ancorem em nós contidos na composição, sem perder a composicionalidade. Isso permite, entre outras coisas, considerar a apresentação da composição como sendo a apresentação de seus componentes. Através do uso de maps, um nó de composição pode tornar um ponto de interface de um de seus nós constituintes visível para qualquer relação com qualquer outro nó fora da composição, associando um ponto de interface daquele constituinte a uma porta da composição. Um exemplo de uso de maps em um nó de composição hipermídia será dado na Seção 4.

Apesar de vários modelos hipermídia tratarem elos como entidades de primeira classe, tais como o modelo Dexter [HaSc94], o modelo AHM [HaBR94], o modelo NCM [SoCr95] e a linguagem para criação de elos XLink [XLIN01], relações expressas por elos hipermídia não recebem o mesmo tratamento. Por isso, uma nova entidade deve ser introduzida para capturar o conceito de conector. Essa entidade foi chamada de *conector hipermídia*.

Um conector hipermídia representa um relacionamento de sincronização espaço-temporal que será usado para criar elos hipermídia entre os nós de um documento. Ele especifica a semântica da relação, mas não especifica as âncoras dos nós que participarão do relacionamento. Âncoras serão especificadas pelos elos hipermídia que usarão essa relação, ou seja, que usarão o conector.

Um conector é definido por um conjunto de pontos de interface, chamados *roles* (papéis), e um atributo chamado *glue* (cola) [Alle97]. Cada papel de um conector especifica a função de um participante da interação, e o glue do conector descreve como os participantes interagem. A Seção 4 dará a definição semântica completa de um papel e do glue baseada no conceito de evento.

Um elo faz referência a um conector e ainda define um conjunto de binds relacionando papéis do conector a pontos de interface de nós, que podem ser âncoras ou portas. Se o ponto de interface de um nó especificado em um bind for uma porta de uma composição, deve existir um caminho de maps relacionando aquela porta a pontos de interface de seus nós constituintes, até que uma âncora seja encontrada. A definição completa de um elo é obtida por:

- uma referência a um conector hipermídia;
- um conjunto de binds relacionando papéis do conector a pontos de interface de nós;

- uma seqüência de maps para cada bind que faz referência a uma porta de um nó de composição, relacionando a porta a pontos de interface de seus nós constituintes. Cada seqüência de maps define uma seqüência de nós aninhados dentro da composição até que uma âncora do nó mais interno da seqüência seja encontrada.

Um exemplo simples de relação hipermídia espaço-temporal é o elo tradicional hipermídia, exemplificado pelo elo HTML, que causa a navegação para um nó de destino quando uma âncora de um nó de origem for selecionada pelo usuário. Nesse caso, um conector hipermídia definiria dois papéis, identificados por *selection_condition* e *presentation_action*, por exemplo, e o glue do conector daria a semântica causal entre eles, especificando que se uma seleção acontecer no participante desempenhando o papel *selection_condition*, a apresentação do participante desempenhando o papel *presentation_action* deve ser executada. A Figura 3 ilustra um conector hipermídia *CON1* modelando a relação tradicional hipermídia e o documento *COMP1* que possui dois elos distintos referenciando o mesmo conector. O elo l_1 especifica que se a âncora *m* do nó *A* for selecionada, a âncora *l* do nó *B* será apresentada, e o elo l_2 especifica que se a âncora *n* do nó *A* for selecionada, a âncora *l* do nó *C* será apresentada.

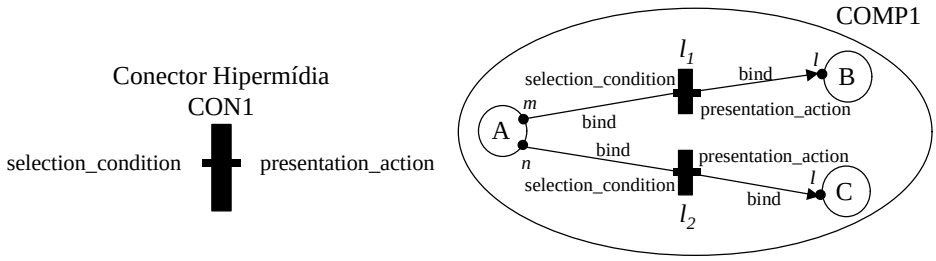


Figura 3 – Exemplo de elos distintos usando o mesmo conector hipermídia

Nesta seção, apresentamos alguns conceitos básicos que podem ser aplicados a qualquer HAL que deseja tratar relações espaço-temporais como entidades de primeira classe. O exemplo anterior ilustrou uma semântica simples para um conector (a semântica do HTML). No entanto, para apresentar a definição detalhada de um conector hipermídia, um modelo semântico mais completo é necessário, como será discutido na próxima seção.

4. Conectores Hipermídia Espaço-Temporais

Usaremos o conceito de evento para especificar relações de sincronização entre nós expressas através de elos hipermídia, pois a semântica de um elo, na maioria das HALs, se não em todas, pode ser expressa por transições em máquinas de estados de eventos.

Evento é a unidade básica para especificar relações de sincronização entre nós de um documento hipermídia. Como afirmado em [PeLi96], um *evento* é uma ocorrência no tempo que pode ser instantânea ou pode ocorrer durante um período de tempo. Algumas HALs especificam dois tipos de evento: *evento de apresentação*, definido pela apresentação de um conjunto de unidades de informação (uma âncora) de um nó e *evento de seleção*, definido pela seleção de uma âncora de um nó. Outras HALs, como NCL, especificam outros tipos de evento, como o *evento de atribuição*, definido pela mudança no valor de um atributo de um nó.

Estados de eventos e transições são usados para definir relações de sincronização entre nós. Um evento de apresentação, por exemplo, pode ser definido pela máquina de

estados mostrada na Figura 4. Inicialmente, o evento se encontra no estado *sleeping*. Ele passa para o estado *preparing* e nele permanece enquanto estiver sendo executado algum procedimento de *prefetch* de suas unidades de informação. Ao final do procedimento, o evento assume o estado *prepared*. Ao iniciar a apresentação de suas unidades de informação, o evento entra no estado *occurring*. Se a exibição for temporariamente suspensa, o evento vai para o estado *paused* e no mesmo permanece enquanto a situação durar. Quando a apresentação de um evento é interrompida abruptamente, o evento passa para o estado *aborted* e automaticamente volta ao estado *prepared*. O estado de um evento de apresentação pode mudar de *occurring* para *prepared* em duas circunstâncias: como conseqüência do término da duração da exibição, ou devido a uma ação que force o encerramento da exibição.

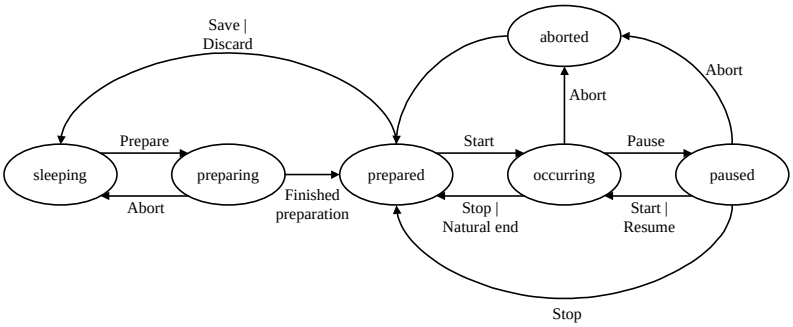


Figura 4 – Exemplo de máquina de estados de um evento de apresentação

Nem todos os modelos hipermídia têm um máquina de estados com a granularidade especificada na Figura 4. Em alguns modelos, o evento de apresentação tem somente os estados *prepared*, *occurring* e *paused*, e em outros, eventos estão restritos aos estados *prepared* e *occurring*. Um evento pode ter um atributo associado, chamado *ocorrências*, que conta o número de vezes que o evento ocorre, ou seja, o número de vezes que seu estado passa de *occurring* para *prepared* no exemplo ilustrado na Figura 4. Eventos de apresentação também podem ter um atributo chamado *repetições*, que indica o número de vezes que o evento deve ocorrer seqüencialmente cada vez que é iniciado. Esse atributo pode receber um valor finito ou infinito (∞).

Conectores hipermídia especificam relacionamentos entre eventos, e possuem os seguintes atributos:

- *conjunto de roles (papéis)*, onde cada papel define como um tipo de evento participa da relação, especificando por exemplo, condições e ações em uma relação causal;
- *glue*, que define como papéis do conector interagem.

Elos hipermídia podem ser usados para expressar relacionamentos causais ou de restrição entre eventos. Em relacionamentos causais, alguma ação é executada quando uma condição específica for satisfeita. Um exemplo seria “se o nó *A* estiver sendo apresentado (condição) e o nó *B* terminou sua apresentação (condição), apresente o nó *C* (ação)”. Outros relacionamentos representam restrições entre eventos. Por exemplo, uma restrição que especifica que dois nós devem terminar suas apresentações juntos. Note que, nesse caso, não existe causalidade envolvida. A ocorrência da apresentação de um nó sem a ocorrência da apresentação do outro também satisfaz a restrição, que só especifica que, se e somente se esses dois nós forem apresentados, os instantes de término de suas apresentações devem coincidir.

Para fornecer a semântica necessária para a definição de relacionamentos causais ou de restrição entre eventos, um conector hipermídia pode ser especializado em conector causal ou conector de restrição. Um conector hipermídia causal tem papéis que podem ser de dois tipos: condição ou ação. E um conector hipermídia de restrição tem papéis de um único tipo, condição.

Um *papel do tipo condição* tem os seguintes atributos:

- *id*, que deve ser único entre os papéis de um conector;
- *tipo do evento*, que pode ser apresentação, seleção, atribuição, etc.;
- *cardinalidade* [*min, max*], onde *max* ≥ *min* ≥ 1 (*max* pode ser ∞). Esse atributo define o número mínimo e máximo de nós que podem desempenhar esse papel em um elo que usa um conector com esse papel. O valor default desse atributo é [1,1].
- *expressão lógica* de condições binárias simples ou compostas, conforme explicado a seguir.

Toda *condição binária simples* é expressa por duas condições unárias: uma *condição prévia*, a ser satisfeita imediatamente antes do instante de tempo em que a condição é avaliada, e uma *condição corrente*, a ser satisfeita no instante de tempo em que a condição é avaliada. Uma condição binária simples é satisfeita se tanto a condição prévia quanto a condição corrente forem satisfeitas. Uma condição unária é uma comparação referente ao estado do evento, ou aos valores de seus atributos ocorrências e repetições, usando os operadores =, ≠, <, ≤, > ou ≥. Tanto a condição prévia quanto a corrente podem receber o valor *VERDADE*, caso elas não sejam relevantes na avaliação da condição simples associada. Uma *condição composta* consiste de uma expressão lógica envolvendo outras condições, simples ou compostas, testando valores de atributos de um mesmo evento e baseada nos operadores ∧ (e), ∨ (ou) e ¬ (negação). Por exemplo, para especificar que um evento ocorreu pela terceira vez, a expressão lógica seria *((estado=occurring, estado=prepared) Ù (verdade, ocorrências=3))*. A principal razão para especificar as condições prévia e corrente quando define-se uma condição binária simples é poder capturar tanto uma mudança de estado de um evento, quanto o intervalo de tempo em que um evento permanece em um mesmo estado.

Um papel do tipo ação tem os seguintes atributos:

- *id*, que deve ser único entre os papéis de um conector;
- *tipo do evento*, que pode ser exibição, atribuição, etc.;
- *tipo da ação*, que especifica a ação que deve ser executada em um participante desempenhando esse papel. Ações causam transições nas máquinas de estados de eventos (veja Figura 4) de participantes desempenhando esse papel. Exemplos de ações são dados na Tabela 1;
- *número de repetições*, que especifica o número de vezes que a ação deve ser repetida. O valor default desse atributo é 0;
- *tempo de espera entre repetições*, que define um tempo que deve ser aguardado antes que a próxima repetição da ação seja executada. O valor default desse atributo é 0 e ele é válido somente se o número de repetições for maior que 0;
- *valor*, obrigatório e válido apenas se o tipo do evento envolve uma operação de atribuição;

- *tempo de espera*, que define um tempo que deve ser aguardado antes que a ação seja executada pela primeira vez. O valor default desse atributo é 0;
- *cardinalidade* [*min*, *max*], onde *max* ≥ *min* ≥ 1 (*max* pode ser ∞). Esse atributo define o número mínimo e máximo de nós que podem desempenhar esse papel em um elo que usa um conector com esse papel. O valor default desse atributo é [1,1].

Tipo do Evento	Tipo da Ação	Descrição
apresentação	*Prepare(E)*	Se o evento *E* estiver no estado *sleeping*, ele passa para o estado *preparing*, caso contrário, nenhuma transição ocorre.
apresentação	*Start(E, n)*	Se o evento *E* estiver nos estados *sleeping*, *preparing*, *prepared* ou *paused*, ele passa para o estado *occurring* através da máquina de estados (veja a Figura 4), iniciando sua apresentação a partir de sua primeira unidade de informação. Caso contrário, nada acontece. Essa ação pode usar o parâmetro opcional *n* para inicializar o atributo repetições do evento.
apresentação	*Stop(E)*	Se o evento *E* estiver no estado *occurring* ou *paused*, ele vai para o estado *prepared*, caso contrário, nada acontece. Essa ação incrementa de uma unidade o valor do atributo ocorrências e decrementa de uma unidade o atributo repetições do evento. Se após decrementado o atributo repetições for maior que zero, o evento é automaticamente iniciado.
apresentação	*Pause(E)*	Se o evento *E* estiver no estado *occurring*, ele vai para o estado *paused*, caso contrário, nada acontece.
apresentação	*Resume(E)*	Se o evento *E* estiver no estado *paused*, ele volta para o estado *occurring*, caso contrário, nada acontece. A apresentação de E recomeça a partir da última unidade de informação apresentada antes do evento ser pausado.
apresentação	*Abort(E)*	Se o evento *E* estiver nos estados *occurring* ou *paused*, ele vai para o estado *aborted*, e imediatamente depois para o estado *prepared*. O atributo ocorrências de *E* não é incrementado e o atributo repetições recebe o valor zero. Se o evento *E* estiver no estado *preparing*, ele vai para o estado *sleeping*, e se *E* estiver em qualquer outro estado, nada acontece.
atribuição	*AbsoluteAssign (E, i)*	Atribui o valor *i* ao atributo correspondente ao evento.
atribuição	*RelativeAssign (E, i)*	Incrementa de *i* o valor do atributo correspondente ao evento

Tabela 1 – Exemplos de ações que podem ser executadas sobre eventos de apresentação e atribuição

Em uma relação causal, o atributo *glue* do conector define uma expressão envolvendo os papéis que são condições e outra envolvendo os papéis que são ações. Quando a expressão com as condições for satisfeita, a expressão com as ações deve ser executada. Em uma relação de restrição, o atributo *glue* do conector define qual a restrição que deve ser satisfeita envolvendo os papéis, que só podem ser condições.

A *expressão de condições* do glue de um conector causal é qualquer expressão baseada nos operadores lógicos ∧ (e), ∨ (ou) e ¬ (negação). Além desses operadores, existe um outro operador ⊕ (retardo) que pode ser aplicado a qualquer operando. Um operador retardo aplicado a uma condição é denotado por $C \oplus [t_1, t_2]$, onde $t_1, t_2 \in \Re$ e $0 \leq t_1 \leq t_2$. Dado que uma condição *C* é verdade num instante *t*, uma condição *C'*, definida como $C \oplus [t_1, t_2]$, é verdade no intervalo de tempo $[t+t_1, t+t_2]$. Por exemplo, para especificarmos

um retardo de dois segundos em uma condição composta envolvendo dois papéis, a expressão da condição seria (*(papel1 Ú papel2) Å [2,2]).*

Papéis do tipo condição, que podem ser desempenhados por mais de um nó (*max* > 1), devem ter um qualificador na expressão de condições do glue com uma das semânticas a seguir:

- ALL – todas as condições devem ser verdadeiras;
- ANY – pelo menos uma das condições deve ser verdadeira.

A *expressão de ações* do glue de um conector causal é definida por uma expressão baseada nos operadores | (paralelo) e → (seqüencial) envolvendo papéis do tipo ação, definindo a ordem de execução de cada ação da expressão.

Papéis do tipo ação, que podem ser desempenhados por mais de um nó (*max* > 1), devem ter um qualificador na expressão de ações do glue com uma das semânticas a seguir:

- ALL – todas as ações devem ser executadas;
- ONE – somente uma dentre as ações deve ser executada.

O exemplo da Figura 5 ilustra a definição completa de um conector causal. Suponha um documento representado pelo nó de composição *COMP2*, usando o documento *COMP1* já mostrado na Figura 3. *COMP2* define o elo *l3*, que usa o conector hipermídia causal *CON2*. A definição completa do elo é discutida na Seção 4.1. Primeiramente, vamos focar nossa atenção em seu conector.

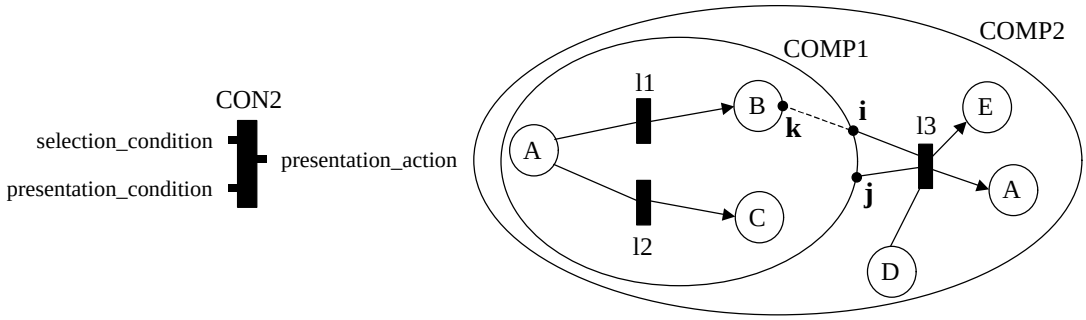


Figura 5 – Exemplo de documento hipermídia usando um conector causal n-ário

O conector *CON2* especifica uma relação hipermídia n-ária, onde a seleção de uma âncora, descrita pelo papel *selection_condition*, acontecendo durante a segunda apresentação de uma outra âncora, descrita pelo papel *presentation_condition*, provoca a apresentação de uma terceira âncora, descrita pelo papel *presentation_action*. A ação deve ser retardada de 2 segundos e deve ser repetida duas vezes com um segundo de retardo entre as repetições. A definição do conector *CON2* é dada por:

➸ Conjunto de papéis:

1. Papel do tipo condição:
   - id => *selection_condition*
   - tipo do evento => *seleção*
   - expressão lógica => *(estado = occurring, estado = prepared)*

- cardinalidade => *[1,1]*

2. Papel do tipo condição:
   - id => *presentation_condition*
   - tipo do evento => *apresentação*
   - expressão lógica => *((estado = occurring, estado = occurring) Ù (verdade, ocorrências = 2))*
   - cardinalidade => *[1, ∞]*
3. Papel do tipo ação:
   - id => *presentation_action*
   - tipo do evento => *apresentação*
   - tipo da ação => *start*
   - número de repetições => *2*
   - tempo de espera entre repetições => *1*
   - tempo de espera => *2*
   - cardinalidade => *[1, ∞]*

➻ Glue:

- Expressão de condições => *((selection_condition) Ù (ALL(presentation_condition))*
- Expressão de ações => *ALL(presentation_action)*

4.1 Elos Hipermídia

Um *elo* usa um conector hipermídia, especificando os nós que desempenharão cada papel do conector. Um elo tem os seguintes atributos:

- conector hipermídia;
- conjunto de *binds*, onde cada bind define um ponto de interface de um nó que desempenhará um papel específico no conector hipermídia usado.

Em um elo, cada papel do conector hipermídia deve ter pelo menos o valor mínimo da cardinalidade (*min*) de binds associados e no máximo o ser valor máximo (*max*).

Um bind tem os seguintes atributos:

- nó *N;*
- ponto de interface de N^1*;*

[1] Pontos de interface de nós ancorados por elos normalmente são portas ou âncoras definindo eventos de apresentação ou seleção. Portas são usadas em nós de composição para exportar âncoras de nós internos, como explicado anteriormente. No entanto, podem existir outros tipos de pontos de interface para definir outros tipos de eventos, como por exemplo, atributos dos nós, no caso de um evento de atribuição. Sem perder a generalidade, este artigo considera o uso de portas e âncoras como pontos de interface de nós.

- especificação de apresentação do nó *N* (opcional);
- conector hipermídia;
- papel do conector hipermídia usado.

A especificação de apresentação de um nó pode ser definida por um elo ancorando no nó para adaptar as características de apresentação de acordo com a navegação feita, por exemplo, associando uma nova folha de estilo a uma página HTML que é destino de um elo. Algumas HALs oferecem essa facilidade, como NCL.

Voltando ao exemplo da Figura 3, o conjunto de binds do elo l_1 é {(*A, m, CON1, selection_condition*), (*B, l, CON1, presentation_action*)} e conjunto de binds do elo l_2 é {(*A, n, CON1, selection_condition*), (*C, l, CON1, presentation_action*)}, onde a âncora *l* representa todo o conteúdo de um nó.

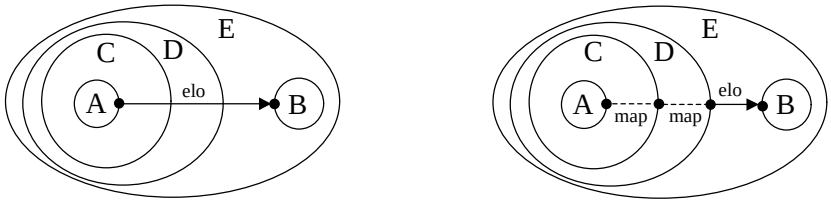
Em HALs que permitem que um nó seja reutilizado em nós de composição distintos e que um elo atravesse uma composição (veja a Figura 5), um ponto terminal de um elo deve identificar a seqüência de nós aninhados usada para atingir a ocorrência do nó ancorado pelo elo.

Alguns modelos hipermídia, como o NCM em sua definição anterior [SoCR95, SoRM00], permitem que elos ancorem diretamente em nós contidos em composições distintas através da definição de uma seqüência de nós aninhados em cada ponto terminal do elo, como ilustrado na Figura 6(a). Apesar de ser uma característica importante que deve ser oferecida por um modelo conceitual hipermídia, isso também impede a composicionalidade, pois elos podem atravessar uma composição sem notificá-la. Para permitir que elos externos possam ancorar indiretamente em nós dentro de uma composição e ainda oferecer composicionalidade, a definição de portas em nós de composição é necessária.

No entanto, portas somente não permitem que elos ancorem dentro de uma composição. Seguindo o metamodelo, nós de composição devem ainda definir mapeamentos relacionando suas portas a pontos de interface de seus nós constituintes. Esses mapeamentos são definidos em um atributo de nó de composição, chamado conjunto de maps. Um *map* tem os seguintes atributos:

- nó de composição
- porta do nó de composição;
- nó constituinte;
- ponto de interface do nó constituinte, que pode ser uma âncora ou porta.

Cada porta de nó de composição deve ter um map associado. Um elo pode ancorar indiretamente em um constituinte da composição através de uma porta e um caminho de maps, que o leva até o nó desejado, como ilustrado na Figura 6(b).



(a) elo ancorando diretamente em nós internos (b) elo ancorando indiretamente em nós internos

Figura 6 – Definições de elo

Voltando ao exemplo da Figura 5, o nó de composição *COMP1* define um map entre sua porta *i* e a âncora *k* do nó *B*, representado por (*COMP1, i, B, k*) e ilustrado por uma linha pontilhada na figura. O elo *l3* especifica que uma seleção na âncora *k* do nó *B*, durante a apresentação da âncora *j* do nó de composição *COMP1* e durante a apresentação do objeto de mídia *D*, provoca a execução da apresentação dos nós *E* e *A*. O conjunto de binds de *l3* é {(*COMP1, i, CON2, selection_condition*), (*COMP1, j, CON2, presentation_condition*), (*D, l, CON2, presentation_condition*), (*E, l, CON2, presentation_action*), (*A, l, CON2, presentation_action*)}. O bind relacionado a porta *i* do nó de composição *COMP1* associado ao map (*COMP1, i, B, k*) faz a âncora *k* do nó *B* desempenhar o papel *selection_condition* do conector *CON2*.

Como um conector hipermídia pode ser reutilizado em elos distintos, se a definição do conector mudar, todos os elos que o referenciam devem ser checados para verificar qualquer inconsistência.

4.2 Conectores Hipermídia Compostos

Outra conseqüência de aplicar o conceito de conector a linguagens de autoria hipermídia é fornecer conectores hipermídia compostos, que é uma facilidade interessante, como veremos a seguir, não encontrada em nenhuma HAL. Um *conector hipermídia composto* tem os seguintes atributos:

- *conjunto de roles (papéis)*, que possuem identificadores únicos (*id*) no conjunto e que serão associados a papéis de conectores constituintes, através do uso de maps. A definição de um papel do conector composto é dada pelo papel do conector constituinte mapeado;
- *glue*, que define:
  - *conjunto de nós*;
  - *conjunto de elos*;
  - *conjunto de elos parcialmente definidos*, que podem não ter binds para todos os papéis de seus conectores, mas que devem ter maps para estes papéis, definidos no conjunto de maps a seguir;
  - *conjunto de maps*, onde cada map relaciona um papel do conector composto a um papel de uma ocorrência de um conector constituinte, representada por um elo parcialmente definido. Cada papel do conector composto deve ter um map associado.

O grafo de nós, elos e elos parcialmente definidos contidos em um conector composto deve ser conexo.

Para ilustrar a definição e a utilidade de conectores hipermídia compostos, considere o conector *CON3* ilustrado na Figura 7. *CON3* representa uma relação hipermídia tradicional que requer uma confirmação do usuário antes de realizar a navegação ao nó de destino. Esse é um procedimento usual em browsers web comerciais, quando o usuário navega para uma página segura, por exemplo. A definição do conector *CON3* é dada por:

➸ Conjunto de papéis:

1. Papel do tipo condição

- id => *selection_condition*

2. Papel do tipo ação:
   - id => *presentation_action*

➽ Glue:

- conjunto de nós => {*X*}
- conjunto de elos => {}
- conjunto de elos parcialmente definidos => {*l4*, *l5*}
- conjunto de maps => {(*CON3, selection_condition, l4, selection_condition*), (*CON3, presentation_action, l5, presentation_action*)}

O conjunto de elos parcialmente definidos especifica os elos *l4* e *l5*, que referenciam o conector CON1 apresentado na Figura 3 e respectivamente definem os seguintes conjuntos de binds {(*X, l, CON1, presentation_action*)} e {(*X, r, CON1, selection_condition*)}, onde *r* é uma âncora de *X*.

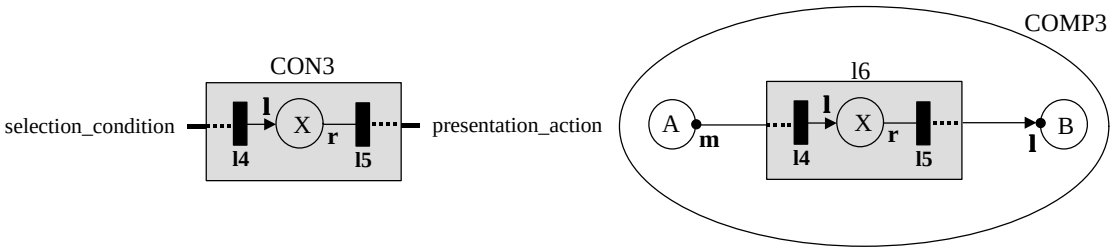


Figura 7 – Exemplo de documento hipermídia usando um conector hipermídia composto

Um documento *COMP3* contendo o elo *l6*, que usa o conector *CON3*, também é ilustrado na Figura 7. O conjunto de binds de *l6* é {(*A, m, CON3, selection_condition*), (*B, l, CON3, presentation_action*)}. Os nós *A* e *B* desempenham os papéis da relação tradicional hipermídia, mas essa relação envolve a apresentação de um nó intermediário X. Quando a âncora *m* do nó *A* é selecionada, o elo *l4* provoca a apresentação do nó *X*. Então, quando a âncora *r* do nó *X* é selecionada, o elo *l5* provoca a apresentação do nó *B*. Considerando o exemplo dado anteriormente, o nó X seria a mensagem do browser que avisa ao usuário que a próxima página será trazida usando um protocolo de comunicação seguro.

Note que um elo que referencia um conector hipermídia composto é definido da mesma forma que um elo usando um conector causal ou de restrição. O elo deve sempre referenciar o conector hipermídia e definir o conjunto de binds relacionando papéis do conector a pontos de interface de nós.

5. Discussão sobre o Uso de Conectores

O ponto chave para tratar relações espaço-temporais como entidades de primeira classe é fornecer a definição da relação independente da especificação de quais nós estarão relacionados.

Algumas HALs já fazem essa separação quando definem uma relação. Em NCL, por exemplo, o primeiro elemento de um elo, similar a um conector, é chamado *meeting point* e o segundo, similar ao conjunto de binds, é chamado *endpoint set*. Entretanto,

NCL não permite que um *meeting point* possa ser reutilizado em elos distintos e trata ambos os elementos como atributos embutidos no elo.

Separar a especificação da relação, do elo que efetivamente a usará, fornece algumas vantagens a linguagens de autoria hipermídia, tais como:

- Reuso da mesma relação espaço-temporal (conector hipermídia) em elos distintos, que possuem a mesma semântica, mas especificam uma interação entre nós diferentes.
- Facilidade de definição de relações de alto-nível que uma HAL poderia fornecer. Agora uma linguagem não precisa oferecer somente um conjunto pré-definido de relações de alto-nível para facilitar o processo de autoria, mas ela também oferece meios para os usuários criarem suas próprias relações. Essas relações, representadas por conectores hipermídia, funcionam como *templates* de elos e podem ser reutilizadas por autores para criar elos em seus documentos. Essa característica fornece poder de expressão e facilidade de uso a uma linguagem.
- Tratamento das relações espaço-temporais como entidades de primeira classe, que podem até ser compostas por outros nós e elos.

Por outro lado, existe uma desvantagem de especificar elos usando conectores hipermídia. Na definição do elo NCM [SoRM00], uma única condição unária poderia relacionar eventos de nós diferentes. Usando conectores, no atributo expressão lógica de um papel do tipo condição, todas as condições unárias devem estar relacionadas a um único tipo de evento (que será associado a uma única âncora).

Para HALs que oferecem meios de especificar relações espaço-temporais através de especificações de baixo nível baseadas em eventos, tais como NCL e IMAP [VaMo93], o conceito de conector hipermídia pode ser diretamente aplicado, fornecendo as vantagens salientadas anteriormente. Para outras HALs que oferecem um conjunto de relações de alto nível, tal como Madeus [JLRS98], o conceito de conector hipermídia também pode ser aplicado, considerando suas relações como conectores contidos em uma biblioteca de conectores pré-definida pela linguagem.

A linguagem para criação de elos XLink [XLIN01] também fornece meios para a criação de elos complexos, envolvendo vários participantes, através de seu conceito de elo estendido (elementos do tipo *extended*). Um elo estendido em XLink possui um conjunto de participantes e um conjunto de regras de navegação. Os participantes podem ser recursos locais (elementos do tipo *resource*) ou remotos (elementos do tipo *locator*) em relação ao recurso onde é feita a definição do elo. As regras de navegação (elementos do tipo *arc*) determinam quais participantes são origens ou destinos do elo. Comparando um elo XLink com elos usando conectores, cada regra de navegação pode ser representada por um conector hipermídia causal, sendo seu conjunto de participantes representado pelo conjunto de binds de um elo. Da mesma forma que um bind relaciona um participante de um elo a um papel do conector hipermídia usado, cada participante de um elo XLink possui um rótulo (atributo chamado *label*) que vai ser usado na especificação de regras de navegação. Entretanto, note que um bind especifica o papel que o participante desempenhará na relação, enquanto que um rótulo é simplesmente uma referência a um participante, não determinando o papel que será desempenhado por ele. Isso será especificado por regras de navegação que fazem referência ao rótulo. Analogamente aos conectores apresentados, um mesmo rótulo pode ser usado por mais de um participante em um elo XLink. Uma regra de navegação XLink relaciona um

rótulo de origem (atributo *from*) e um rótulo de destino (atributo *to*), além de especificar quando a navegação deve ser feita (atributo *actuate*) e como a apresentação do destino deve ser realizada (atributo *show*). A navegação através do elo pode ser disparada por um evento de seleção do usuário (*actuate="onRequest"*) ou feita automaticamente quando o recurso de origem for carregado (*actuate="onLoad"*). A apresentação do recurso de destino pode ser feita em uma nova janela (*show="new"*), ou substituindo a apresentação do recurso de origem (*show="replace"*) ou ainda podendo ser embutida dentro da apresentação do mesmo (*show="embed"*). Os valores dos atributos *actuate* e *show* do XLink representam respectivamente subconjuntos pré-definidos de papéis do tipo condição e de papéis do tipo ação de um conector hipermídia causal, envolvendo eventos de seleção ou apresentação. Uma regra de navegação XLink é representada por um conector hipermídia causal que relaciona apenas dois papéis, um do tipo condição e outro do tipo ação, determinados pelos valores dos atributos *actuate* e *show*. O conceito de conector proposto neste artigo fornece bem mais expressividade para a especificação de relações, permitindo a definição de expressões em seu atributo glue que relacionam um número qualquer de papéis. Além disso, as condições e ações do conector proposto podem atuar sobre outros tipos de evento, além dos tipos mais usuais que são seleção e apresentação. Como exemplo, condições e ações envolvendo eventos de atribuição podem ser usadas para a especificação de relações espaciais entre objetos, o que não é permitido em XLink. Outra limitação do XLink é que a linguagem só permite a especificação de relacionamentos causais. Os conectores apresentados neste trabalho também permitem a especificação de relações de restrição, podendo representar, por exemplo, a semântica dos *sync arcs* [HaBR94] do modelo AHM. Por fim, XLink também não permite a definição de regras de navegação de forma independente, ou seja, a definição da relação sem especificar seus participantes, o que impede o reuso de uma relação em elos distintos.

Este trabalho trouxe características encontradas no domínio de engenharia de software para o domínio hipermídia. Existem outros trabalhos no sentido oposto estudando a possibilidade de usar linguagens de autoria de documentos no domínio de arquitetura de software, tal como o trabalho de [PSSC00], que discute o mérito de usar XML [XML00] com uma metalinguagem para a definição de ADLs. Além desse objetivo geral, os autores também desenvolveram um esquema XML para uma ADL chamada ACME e também um protótipo para a criação de representações XML para arquiteturas definidas em ACME.

6. Considerações Finais

Este trabalho propôs como o conceito de conectores pode ser introduzido em hipermídia, trazendo algumas contribuições de ADLs para HALs. Analogamente a [Shaw94], este artigo discutiu porque relações de sincronização hipermídia, expressas por elos, merecem status de primeira classe. Dentre suas contribuições, podemos ressaltar:

- a possibilidade de reutilizar a especificação de uma relação espaço-temporal em elos distintos, mas que têm o mesmo comportamento;
- a possibilidade de permitir que usuários experientes especifiquem relações complexas, representadas por conectores modelando relações hipermídia de alto nível, que podem ser facilmente usadas por outros autores menos experientes para a criação de elos;

- a possibilidade de definir relações espaço-temporais de composição expressas por elos, que é uma característica original não encontrada em nenhuma linguagem ou modelo hipermídia;
- a definição de portas e conjuntos de maps para nós de composição hipermídia, tornando um modelo hipermídia composicional, sem perder a facilidade de elos poderem ancorar em nós aninhados dentro de um nó de composição. Composicionalidade é desejável em HALs, pois facilita a verificação de consistência de documentos.

O conceito de conector hipermídia foi incorporado ao modelo hipermídia NCM e à linguagem de autoria NCL, herdando as vantagens discutidas. O sistema de autoria HyperProp [SoRM00], baseado no NCM, está sendo atualizado para fornecer a definição de conectores hipermídia.

Um trabalho futuro, que já está em andamento, é propor como as facilidades do conector hipermídia poderiam ser tornadas disponíveis para qualquer documento XML. Os objetivos desse trabalho em andamento podem ser comparados aos do XLink combinados com todas as vantagens discutidas neste artigo.

Outro trabalho futuro é estudar como o conceito de estilo arquitetural oferecido em ADLs poderia ser generalizado para o domínio hipermídia. Estilos fornecem a definição de restrições que toda configuração que segue o estilo deve satisfazer. Esses conceitos aplicados em documentos hipermídia permitiriam que estruturas de documentos com características particulares pudessem ser reutilizadas em vários documentos distintos. Essa estruturas poderiam ser usadas como *templates* de documentos representando tipos de documentos bem conhecidos.

Referências

[Alle97] R. Allen. A Formal Approach to Software Architecture, *PhD Thesis*, School of Computer Science, Carnegie Mellon University, Maio de 1997.

[AMRS00] M.J. Antonacci, D.C. Muchaluat-Saade, R. Rodrigues, L.F.G. Soares. NCL: Uma Linguagem Declarativa para Especificação de Documentos Hipermídia na Web, *Anais do VI Simpósio Brasileiro de Sistemas Multimídia e Hipermídia - SBMídia2000*, Natal, Rio Grande do Norte, Junho de 2000.

[CRHH95] L. Carr, D. Roure, W. Hall, G. Hill. The Distributed Link Service: A Tool for Publishers, Authors and Readers, *Proceedings of the Fourth International World Wide Web Conference*, Boston, USA, 1995.

[HaBR94] L. Hardman, D.C.A Bulterman, G. Van Rossum. The Amsterdam Hypermedia Model, *Communications of the ACM*, Vol. 37, No. 2, pp. 50-62, Fevereiro de 1994.

[HaSc94] F. Halasz, M. Schwartz. The Dexter Hypertext Reference Model, *Communications of the ACM*, Vol. 37, No. 2, pp. 30-39, Fevereiro de 1994.

[JLRS98] M. Jourdan, N. Layaïda, C. Roisin, L. Sabry-Ismail; L. Tardif. Madeus, an Authoring Environment for Interactive Multimedia Documents, *Proceedings of the ACM Multimedia Conference 98*, pp. 267-272, England, Setembro de 1998.

[MeMP00] N. Mehta, N. Medvidovic, S. Phadke. Towards a Taxonomy of Software Connectors, *Proceedings of the 22nd International Conference on Software Engineering*, Limerick, Ireland, Junho de 2000.

[MeTa00] N. Medvidovic, R. Taylor. A Classification and Comparison Framework for Software Architecture Description Languages, *IEEE Transactions on Software Engineering*, Vol. 26, No. 1, Janeiro de 2000.

[MuSo01] D.C. Muchaluat-Saade, L.F.G. Soares. Towards the Convergence between Hypermedia Authoring Languages and Architecture Description Languages, *Proceedings of the ACM Symposium on Document Engineering – SDE'2001*, Atlanta, Georgia, Novembro de 2001.

[PeLi96] M. Pérez-Luque, T. Little. A Temporal Reference Framework for Multimedia Synchronization. *IEEE Journal on Selected Areas in Communications (Special Issue: Synchronization Issues in Multimedia Communication)*, 14(1), Janeiro de 1996. pp. 36-51.

[PSSC00] S. Pruitt, D. Stuart, W. Sull, T.W. Cook. The Merit of XML as an Architecture Description Language Meta-Language, *Relatório Técnico de MCC (Microelectronics and Computer Technology Corp.)*, disponível em http://www.mcc.com/projects/ssepp/papers/meritofxml.html, Austin, TX, Janeiro de 2000.

[RRMS99] L.M. Rodrigues, R.F. Rodrigues, D.C. Muchaluat-Saade, L.F.G. Soares. Acrescendo Facilidades NCM a Documentos SMIL, *Anais do V Simpósio Brasileiro de Sistemas Multimídia e Hipermídia – SBMÍDIA'99*, Goiânia, Goiás, Junho de 1999.

[Shaw94] M. Shaw. Procedure Calls are the Assembly Language of Software Interconnections: Connectors Deserve First-Class Status, *Proceedings of the Workshop on Studies of Software Design*, LNCS, Springer-Verlag, 1994.

[ShGa96] M. Shaw, D. Garlan. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall, Abril de 1996.

[SMIL01] "Synchronized Multimedia Integration Language (SMIL 2.0)" — *W3C Recommendation*. Agosto de 2001. disponível em http://www.w3.org/TR/smil20.

[SoCR95] L. Soares, M. Casanova, N. Rodriguez. Nested Composite Nodes and Version Control in an Open Hypermedia System, *International Journal on Information Systems; Special issue on Multimedia Information Systems*, Vol. 20, No. 6, Elsevier Science Ltd. England, pp. 501-520, Setembro de 1995.

[SoRM00] L. Soares, R. Rodrigues, D. Muchaluat-Saade. Authoring and Formatting Hypermedia Documents in the HyperProp System, *ACM Multimedia Systems Journal*, Vol. 8, No. 2, pp. 118-134, Springer-Verlag, Março de 2000.

[VaMo93] M. Vazirgiannis, C. Mourlas. An Object Oriented Model for Interactive Multimedia Applications. *The Computer Journal, British Computer Society*, 36(1), Janeiro de 1993.

[XLIN01] "XML Linking Language (XLink) Version 1.0". *W3C Recommendation*, Junho de 2001. disponível em http://www.w3.org/TR/xlink

[XML00] "Extensible Markup Language (XML) 1.0 (Second Edition)". *W3C Recommendation*, Outubro de 2000. disponível em http://www.w3.org/TR/REC-xml