MARCEL STANLEY ALBUQUERQUE DE MOURA

RELAÇÕES ESPACIAIS EM DOCUMENTOS HIPERMÍDIA

Dissertação de Mestrado

Departamento de Informática

Rio de Janeiro, 31 de agosto de 2001.

Pontifícia Universidade Católica do Rio de Janeiro

MARCEL STANLEY ALBUQUERQUE DE MOURA

RELAÇÕES ESPACIAIS EM DOCUMENTOS HIPERMÍDIA

Dissertação de Mestrado apresentada ao Departamento de Informática da PUC-Rio, como parte dos requisitos para obtenção do título de Mestre em Informática: Ciência da Computação.

Orientador: Luiz Fernando Gomes Soares.

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, 31 de agosto de 2001.

Aos meus pais, Ezequias e Verônica,

por todo amor e compreensão a mim

dedicados durante todos esses anos.

AGRADECIMENTOS

Aos meus pais, Ezequias Viana de Moura e Verônica Maria Albuquerque de Moura, pelo amor, pelo apoio e por terem me proporcionado condições de realizar o Mestrado. Aos meus irmãos, Germman Albuquerque de Moura e Priscila Albuquerque de Moura, pelo amor e pelo apoio.

Ao amigo e orientador, Luiz Fernando Gomes Soares, sem o qual esse trabalho não poderia ter sido realizado.

A Marcel Trigueiro, pelo companheirismo e amizade de longa data.

Aos amigos do laboratório Telemídia, em especial a Rogério Rodrigues e Débora Muchaluat, que contribuíram bastante para a conclusão dessa dissertação.

A Antônio Tadeu, Renata Gorini e Sérgio Colcher pela amizade e pela ajuda durante o período do Mestrado.

Aos amigos e professores Guido Lemos e Thaís Batista, que muito contribuíram para que eu chegasse ao Mestrado.

Aos demais amigos da PUC, em especial a (em ordem alfabética) Milene Siveira, Patrícia Vilain, Pedro Silva e Thyago Mota.

À CAPES, pelo apoio financeiro desde a graduação.

RESUMO

O processo de autoria de documentos hipermídia é bastante complexo. Além da definição detalhada de sua estrutura lógica, é importante que tanto a sincronização temporal quanto a espacial da apresentação dos seus objetos sejam especificadas. A sincronização espacial, em particular, envolve a especificação da disposição espacial dos objetos do documento em relação aos dispositivos de exibição usados durante a apresentação e a definição de como suas características espaciais devem variar durante a apresentação.

Esta dissertação apresenta a definição de um modelo de sincronização espacial para documentos hipermídia, associado à definição de um conjunto de relações espaciais. Além disso, é apresentada a aplicação desse modelo ao Modelo de Contextos Aninhados (NCM) e sua extensão, através da definição de relações espaço-temporais e animações. Também é apresentada a implementação de um ambiente de autoria da sincronização espacial de documentos NCM.

Palavras-chave: relações espaciais, sincronização espacial, documentos hipermídia, modelo de contextos aninhados, sistema HyperProp.

ABSTRACT

Hypermedia Authoring is a very complex process. Along with the definition of the document's logic structure, the specification of its temporal and spatial synchronization plays a major role. Spatial synchronization deals with the document's objects visual placement within their presentation devices and the definition of the spatial changing dynamics that should take place on those objects throughout a presentation.

This work presents a spatial synchronization model for hypermedia documents along with a definition of a set of spatial relationships. An instantiation of this model into NCM (Nested Context Model), and its extension through a definition of spatio-temporal relationships and animations are also presented. Finally, some features of a spatial synchronization tool are showed.

Keywords: *spatial relationships, spatial synchronization, hypermedia documents, nested context model, HyperProp system.*

ÍNDICE ANALÍTICO

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

1 INTRODUÇÃO

O processo de autoria de documentos hipermídia é bastante complexo. Por se tratar de documentos compostos por diversos objetos, é necessária uma definição detalhada de sua estrutura lógica (seja com relação a sua apresentação, contextualização, histórico de versões, etc.).

Além da definição da estrutura lógica do documento, é importante que tanto a sincronização temporal quanto a espacial da apresentação dos seus objetos sejam especificadas. A sincronização temporal diz respeito à definição de como os objetos deverão ser apresentados de acordo com o tempo.

Por sincronização espacial, entende-se a especificação da disposição espacial dos componentes visuais do documento em relação aos dispositivos de exibição usados durante a apresentação e a definição de como suas características espaciais devem variar durante a apresentação.

Todas as definições de sincronização de um documento, sejam espaciais ou temporais, são usadas pelo formatador hipermídia para definir como a apresentação será realizada. Um formatador hipermídia [Rodrigues97] é uma máquina que controla a apresentação de um documento hipermídia. Ela é responsável por compilar as definições lógica, temporal e espacial de um documento e controlar a sua execução. Dentre outras coisas, o formatador é responsável por captar eventos espaciais e/ou temporais assinalados pelos exibidores [Rodrigues00] dos objetos de mídia de um documento e executar ações em resposta. Outra função importante do formatador é o controle elástico do tempo de apresentação dos objetos de um documento, adaptando as especificações do autor do documento às dificuldades encontradas em sua apresentação. O conceito de formatador hipermídia será usado durante toda a dissertação.

Vários trabalhos têm sido conduzidos nos últimos anos, abrangendo aspectos variados da sincronização espacial de documentos hipermídia. Eles abordam:

- a definição do posicionamento e das dimensões de objetos de um documento durante sua apresentação;
- a especificação de relações espaciais entre os objetos de um documento;
- a definição de relações espaço-temporais, envolvendo atributos espaciais e temporais de objetos de um documento; e
- a criação de animações de objetos durante a sua apresentação.

A próxima seção apresenta esses aspectos em maiores detalhes.

1.1 ASPECTOS RELACIONADOS À SINCRONIZAÇÃO ESPACIAL

A sincronização espacial em documentos hipermídia é composta de vários aspectos, que vão desde a definição dos dispositivos que serão usados em uma apresentação até a definição de como os objetos de mídia do documento serão exibidos nesses dispositivos. As subseções a seguir apresentam cada um dos quesitos relacionados à sincronização espacial, segundo uma abordagem semelhante à usada em [Hardman98].

1.1.1 Definição de dispositivos de apresentação

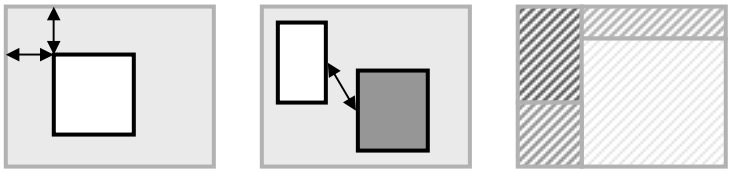
Um documento hipermídia é composto de vários fragmentos de informação, identificados como objetos de mídia. Esses objetos, codificados em mídias de representação diversas (texto, áudio, vídeo, imagem estática) precisam ser associados a dispositivos de exibição para que possam ser apresentados apropriadamente. Exemplos de dispositivos de exibição são um terminal de vídeo, um canal de áudio, uma televisão, o visor de um telefone celular, etc.

Um modelo de documentos hipermídia deve permitir a definição dos dispositivos de exibição a serem usados na apresentação de um documento hipermídia.

1.1.2 Posicionamento de objetos de mídia

Ao ser associado a um dispositivo de exibição, deve ser definido de que forma um objeto de mídia deverá ser posicionado. Existem diversas abordagens que podem ser usadas para definir o posicionamento de um objeto em uma apresentação.

Dado que um dispositivo de apresentação possui dimensões, a posição de um objeto pode ser definida de forma relativa a ele (Figura 1.1(a)). Por exemplo, pode-se especificar que uma imagem estática seja exibida a 10 mm da esquerda e a 20 mm do topo de um monitor de vídeo. Um objeto pode ser posicionado também com relação a outro objeto que esteja sendo exibido no mesmo dispositivo (Figura 1.1(b)).



(a) (b) (c)

Figura 1.1 – Posicionamento de objetos de mídia em um dispositivo de apresentação

Ao invés de determinar o posicionamento de cada objeto de uma apresentação, pode-se usar de **estruturas de apresentação** predefinidas. Essas estruturas determinam canais associados a um dispositivo de apresentação que podem ser usados para a apresentação de qualquer objeto de um documento (Figura 1.1(c)).

A posição de um objeto também pode ser determinada dinamicamente com relação ao tempo. A Figura 1.2 ilustra um objeto sendo movido (animado) durante sua apresentação.

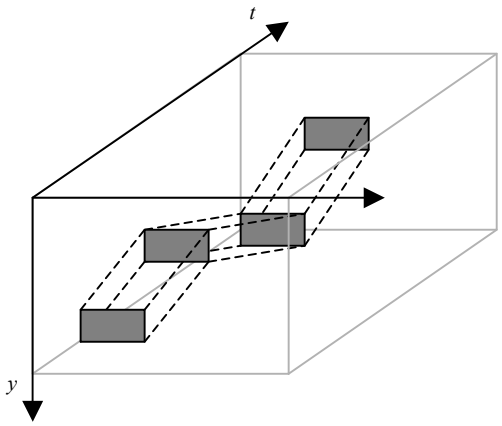


Figura 1.2 – Animação de um objeto.

A orientação de um objeto de mídia também pode ser especificada, em adição à definição do seu posicionamento. Pode-se definir que um objeto seja exibido horizontalmente ou verticalmente invertido.

1.1.3 Dimensões de objetos de mídia

Além de definir o posicionamento dos objetos de mídia de um documento, deve-se determinar suas dimensões. Elas podem ser definidas de forma implícita, explícita ou com relação a outros objetos, dispositivos ou canais de uma estrutura de apresentação.

Por exemplo, objetos de vídeo e imagens estáticas têm, em geral, dimensões definidas intrinsecamente, não necessitando de qualquer especificação. Ao contrário, objetos de mídia texto devem ter suas dimensões de apresentação explicitadas. Por exemplo, pode-se definir que um objeto de texto, correspondente à legenda de um objeto de vídeo, tenha sua largura igual à do vídeo.

1.1.4 Redimensionamento de objetos de mídia

As dimensões originais de um objeto podem não ser apropriadas para a sua apresentação. Pode ser necessário ajustá-las para que o objeto seja apresentado a contento. Quando as dimensões do objeto forem menores que o esperado, o objeto pode ser aumentado ou ser repetido. Quando elas forem maiores do que deveriam, o objeto pode ser cortado (*cropping*) ou ser diminuído (*shrinking*). Ao alterar as dimensões de um objeto, a proporção (*aspect ratio*) entre elas deve ser considerada. Em objetos de vídeo, por exemplo, pode ser desejável manter essa proporção quando algum ajuste for feito.

1.1.5 Prioridade de apresentação

A apresentação simultânea de vários objetos em um mesmo dispositivo de exibição pode ocasionar sobreposições. Deve-se prover um meio de definir prioridades de apresentação em um documento hipermídia, de forma que objetos mais importantes de um documento não sejam sobrepostos por outros durante sua apresentação.

1.1.6 Relações entre objetos

Como mencionado na Seção 1.1.2, o posicionamento de objetos em uma apresentação pode ser determinado com base em outros objetos. Relações entre objetos são importantes na definição da sincronização de documentos hipermídia. Elas podem envolver características/atributos de somente um objeto (“intraobjeto”) ou de dois ou mais objetos (“interobjetos”).

Em particular, quando envolvem somente características espaciais, essas relações são chamadas de relações espaciais. Outros atributos, como as dimensões de objetos e suas prioridades de apresentação, podem estar envolvidos na definição de relações espaciais. Exemplos de relações espaciais são apresentadas abaixo:

- "dois objetos devem estar centralizados horizontalmente durante sua apresentação" (interobjetos);
- "a largura de um objeto deve ser o dobro de sua altura" (intraobjeto).

Relações podem envolver além de atributos espaciais, atributos temporais e/ou ações associadas à apresentação de objetos (inicia, interrompe ou termina exibição). Essas relações são chamadas de relações espaço-temporais. Exemplos desse tipo de relação são descritos abaixo:

- "quando um objeto estiver na posição (100,130) sua exibição deve ser encerrada";
- "dois objetos devem permanecer alinhados à direita enquanto um terceiro estiver sendo apresentado";
- "ao término da apresentação de um objeto, dois outros devem ser posicionados justapostos à esquerda".

1.1.7 Transições

Em documentos hipermídia ou multimídia é comum que o fim da apresentação de um objeto ocasione o início da apresentação de outro. Efeitos de transição podem ser aplicados quando isso acontece.

Exemplos de efeito de transição são o aparecimento/desaparecimento gradativo de um objeto, bastante úteis na definição de apresentações de *slides*.

1.2 OBJETIVOS DA DISSERTAÇÃO

Essa dissertação tem como objetivos:

- a definição de um modelo de sincronização espacial para documentos hipermídia que permita a definição das características espaciais dos objetos de um documento de uma forma uniforme. Esse modelo englobará a definição de um conjunto de relações espaciais

que auxiliem a compor a sincronização espacial da apresentação de documentos hipermídia;

- a definição de relações espaço-temporais que permitam associar a sincronização espacial de um documento à sua sincronização temporal;
- a aplicação do modelo espacial a um modelo de documentos hipermídia, mais precisamente, ao Modelo de Contextos Aninhados (NCM); e
- a implementação de um ambiente que permita a autoria da sincronização espacial e temporal de documentos NCM.

1.3 ESTRUTURA DA DISSERTAÇÃO

Esta dissertação encontra-se organizada como se segue.

O Capítulo 2 introduz alguns conceitos relacionados à sincronização espacial de documentos hipermídia, como a definição de posicionamento de objetos e de relações espaciais. Após são apresentados vários trabalhos que permitem a definição de algum mecanismo de sincronização espacial.

O Capítulo 3 apresenta um modelo genérico de definição de estruturas de apresentação e como essas estruturas podem ser usadas para definir a apresentação espacial de documentos hipermídia. Além disso, nesse capítulo, é definido como relações espaciais e espaço-temporais entre objetos de um documento podem ser criadas.

O Capítulo 4 apresenta a aplicação do modelo definido no Capítulo 3 na especificação de sincronização espacial de documentos definidos pelo Modelo de Contextos Aninhados. É mostrado como é possível criar estruturas de apresentação através de objetos descritores e como relações espaciais e espaço-temporais entre objetos podem ser definidas através de elos. No final do capítulo, é apresentada como a animação de objetos NCM pode ser definida.

O Capítulo 5 apresenta a implementação do *Browser* de Sincronismo do sistema HyperProp, uma ferramenta para a definição e acompanhamento da sincronização espacial e temporal de documentos NCM.

Finalmente, o Capítulo 6 mostra algumas conclusões sobre o trabalho realizado, compara esse trabalho com os apresentados no Capítulo 2 e apresenta alguns possíveis trabalhos futuros.

2 TRABALHOS RELACIONADOS

Sincronização espacial envolve a definição das características espaciais dos objetos que compõem um documento e como eles interagem espacialmente durante sua apresentação. Este capítulo apresenta uma série de trabalhos relacionados à sincronização espacial de documentos hipermídia. Outros trabalhos poderiam ter sido incluídos, entretanto, somente são apresentados aqueles que tiveram maior influência na definição dessa dissertação.

O capítulo é estruturado com se segue. A Seção 2.1 apresenta a linguagem SMIL. A Seção 2.2 apresenta a linguagem utilizada pelo sistema Madeus e suas extensões. A Seção 2.3 apresenta o método de especificação SSTS. A Seção 2.4 apresenta um conjunto de trabalhos desenvolvido por Vazirgiannis. A Seção 2.5 apresenta algumas linguagens de definição de estilos, usadas em documentos hipermídia. Finalmente, a Seção 2.6 apresenta um breve resumo do capítulo.

2.1 SMIL

SMIL[1] (*Synchronized Multimedia Integration Language*, pronunciada *smile*) [SMIL2] é uma linguagem declarativa, baseada na versão 1.0 de XML [XML], criada pelo *World Wide Web Consortium* (W3C)[2] para a definição de apresentações multimídia interativas.

SMIL permite definir o comportamento temporal de uma apresentação multimídia, criar elos entre os objetos de mídia envolvidos e definir uma **estrutura de apresentação**, associada a um dispositivo de apresentação único. Essa linguagem é composta de um conjunto de módulos, cada um responsável por um conjunto de propriedades da linguagem.

[1] SMIL encontra-se atualmente na versão 2.0 e, durante o período de escrita dessa dissertação, passava pelo estágio final de avaliação dos membros do W3C, estando prestes a se tornar uma recomendação.
[2] O World Wide Web Consortium (www.w3.org) foi criado em outubro de 1994 com o objetivo de promover a evolução da World Wide Web e promover sua interoperabilidade.

A definição da estrutura de apresentação de documentos SMIL é feita através de elementos e atributos definidos em quatro módulos de *layout*.

Uma estrutura de apresentação SMIL é definida no cabeçalho de um documento SMIL e é composta de um conjunto de janelas, cada uma delas composta de uma estrutura hierárquica de regiões (*canais*), às quais os objetos de mídia de um documento devem ser associados para que sejam apresentados.

A especificação de uma estrutura de apresentação é delimitada pelo elemento, único e não-vazio, `layout`, definido no módulo básico. A definição de uma janela é feita através do elemento não-vazio `topLayout` e deve estar contida no elemento `layout`. As regiões de uma janela são definidas internamente aos elementos `topLayout` correspondentes por elementos `region`. Regiões podem conter outras regiões, que são definidas internamente às regiões que as contém por elementos `region`. A Figura 2.1 apresenta um exemplo de definição de estrutura de apresentação SMIL.

```
<layout>
   <topLayout id="Window1" width="320" height="240"/>
      <region id="pictures" title="pictures" height="100%">
         <region id="picture1" height="100" fit="meet"/>
         <region id="picture2" height="80" fit="slice"/>
      <region/>
   </topLayout>
   <topLayout id="Window2" title="Captions" width="320" height="60">
      <region id="captions" title="caption text" top="20" fit="meet"/>
   </topLayout>
</layout>
```

Figura 2.1 – Exemplo de uma estrutura de apresentação SMIL.

Nesse exemplo, é definida uma estrutura de apresentação composta de duas janelas. A primeira delas é composta de uma região que, por sua vez, contém outras duas. A segunda janela é composta de uma só região. A Figura 2.2 ilustra essa estrutura.

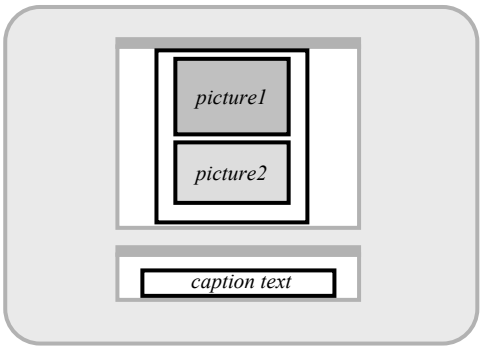


Figura 2.2 – Ilustração do exemplo da Figura 2.1.

Como qualquer elemento da linguagem SMIL, os elementos de uma estrutura de apresentação possuem os atributos básicos `id` (identificador único na definição do documento) e `title` (título do elemento). Todo os elementos possuem atributos adicionais. O elemento `topLayout` possui os seguintes atributos adicionais: `backgroundColor`, que define uma cor de fundo para a janela; `close`, que define em que situações a apresentação de uma janela deve ser terminada; `height`, que define a altura da janela; `open`, que define quando a apresentação da janela deve ser iniciada; e `width`, que define a largura da janela. O elemento `region` possui os seguintes atributos: `backgroundColor`, `height` e `width`, definidos de forma idêntica ao feito para o elemento `topLayout`; `bottom`, que define a distância da borda inferior de uma região com relação à janela ou região que a contém; `fit`, que define como o objeto de mídia associado à região deve se adequar a ela durante sua exibição; `left`, que define a distância da borda da esquerda de uma região com relação à janela ou região que a contém; `regionName`, que define um nome para a região; `right`, que define a distância da borda da direita de uma região com relação à janela ou região que a contém; `showBackground`, que define em que situações a janela deve mostra sua cor de fundo; `top`, que define a distância da borda superior de uma região com relação à janela ou região que a contém; e `z-index`, que define a prioridade de apresentação da janela.

Como mencionado anteriormente, objetos de mídia de um documento SMIL são associados a regiões para que sejam exibidos. Adicionalmente, a linguagem provê pontos de registro para auxiliar o posicionamento de um objeto de mídia visual em sua região associada. Cada ponto de registro identifica um ponto e é definido pelo elemento vazio `regPoint` que deve ser definido dentro do elemento `layout` e pode ser utilizado por qualquer uma das regiões de uma estrutura de *layout*. Ele possui os seguintes atributos: `bottom`, `left`, `right` e `top`, que definem, respectivamente, as distâncias do ponto de registro com relação à borda inferior,

à borda da esquerda, à borda da direita e à borda superior da região; `left`, que define a distância do ponto de registro à borda da região; e `regAlign`, que define o algoritmo de alinhamento a ser utilizado no posicionamento do objeto de mídia. Esse atributo determina um ponto do objeto de mídia que deve se posicionar nas coordenadas definidas pelos outros atributos do ponto de registro. Seus valores possíveis são definidos na Figura 2.3. Cada um dos pontos define que parte do objeto deve ser posicionada sobre o ponto de registro.

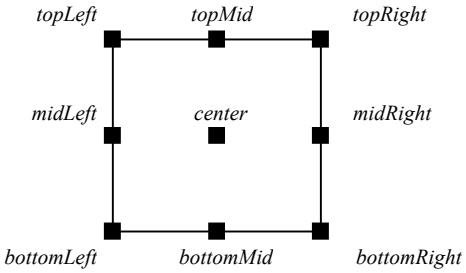


Figura 2.3 – Pontos de registro *default*.

A associação de objetos de mídia é feita através do atributo `regPoint` do elemento que define o objeto. A Figura 2.4 mostra um exemplo de definição de um ponto de registro e de seu uso no posicionamento de uma imagem (`img`) em uma região. A imagem é definida no corpo do documento SMIL, definido pelo elemento `body`. Mais informações a respeito da definição da estrutura lógica de documentos SMIL podem ser encontrados na especificação do conjunto de módulos de estrutura de SMIL [SMIL2]. O ponto de registro usado para posicionar a imagem é ilustrado na Figura 2.5.

```
<layout>
   <regPoint id="p1" top="50" left="50" regAlign="topLeft"/>
   <window id ="window1" width="300" height="200" left="10" top="10">
      <region id ="region1" width="300" height="200">
   </window>
</layout>
...
<body>
   ...
   <img region="region1" regPoint="p1">
   ...
</body>
```

Figura 2.4 – Posicionamento de uma região com o auxílio de um ponto de registro.

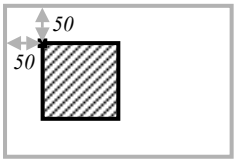


Figura 2.5 – Ilustração do exemplo da Figura 2.4.

Além da definição da estrutura de apresentação de seus documentos, em SMIL é possível definir animações, ou variações, dos valores de atributos de objetos de um documento. A linguagem define dois módulos de animação com esse propósito.

O modelo de animação de SMIL é baseado na definição de uma função contínua baseada no tempo que, aplicada a um atributo de um elemento (atributo alvo), define seu valor para qualquer tempo durante o período de tempo definido por sua duração simples[3]. A essa função, dá-se o nome de **função de animação simples**.

De fato, um atributo não tem seu valor alterado com a aplicação de uma função de animação. SMIL define que durante a apresentação de um documento, para cada atributo animado, deve ser mantido um **valor de apresentação**. Por exemplo, considere um objeto *o* qualquer que possui um atributo *x* com valor inicial igual a 10. Caso seja aplicada uma animação a esse atributo que defina que ele deve variar entre 0 e 100 em um período de 10 segundos, o valor original do atributo é mantido. A animação ocorre com um valor de apresentação do atributo. Esse valor é iniciado com o valor 0 e é incrementado até 100 durante o tempo especificado. Durante toda a animação, caso o valor do atributo *x* seja consultado, será retornado 10, seu valor original.

Animações podem ser definidas de forma a sobrepor (animações não-aditivas) o *valor base* (o valor base de um atributo corresponde ao seu valor original ou ao resultado da aplicação da última função de animação) de um atributo ou adicionar-lhe algo (animações aditivas). Além disso, elas podem ser repetidas ou congeladas (quando o seu valor não deve se alterar). Uma função mais geral, a **função de efeito de animação**, definida para toda a duração ativa[4] do objeto envolvido, engloba a definição de como os valores devem ser gerados durante suas repetições e do período em que ela deve permanecer congelada. Essa função é definida com base na definição da função de animação simples do atributo.

---

[3] A duração simples de um atributo de um objeto qualquer corresponde à duração da apresentação desse objeto, excluindo-se suas possíveis repetições e o período em que sua apresentação tenha sido interrompida.

[4] A duração ativa de um objeto corresponde a soma de todas as suas durações simples e ao período em que ela se encontra interrompida.

A repetição de uma animação pode ser definida de forma cumulativa, onde o valor de uma repetição deve ser acumulado, servindo de base para a aplicação da função de animação à repetição seguinte, ou não-cumulativa, quando a cada repetição o valor de apresentação inicial é atualizado com relação ao valor base do atributo.

Os módulos de animação são compostos de elementos e atributos usados na definição de animações. Todos eles utilizam o modelo de animação descrito brevemente acima. Os elementos existentes são: `animate`, `set`, `animateMotion` e `animateColor`. Todos eles possuem os seguintes atributos:

- `attributeName` e `attributeType` que definem, respectivamente, o nome do atributo a ser animado e seu tipo;
- `dur` que define a duração simples da animação;
- `targetElement` e `href`, que podem ser usados alternativamente para definir o elemento cujo atributo especificado deve ser animado. Esses atributos são opcionais, no caso do elemento de animação ser definido internamente ao elemento alvo.

O elemento `animate` define uma animação genérica. Ele pode ser definido através dos seguintes atributos adicionais:

- `from`, `to` e `by`, correspondentes ao valor inicial, ao valor final e à variação a ser aplicada ao valor do atributo a ser animado. Eles podem ser combinados para definir os seguintes tipos de animação: *from-to*, *from-by*, *by* e *to*. Por exemplo, pode-se definir as seguintes animações:

```
<animate attributeName="width" from="50" to="100" dur="10s">
<animate attributeName="width" from="50" by="25" dur="10s">
<animate attributeName="width" by="30" dur="10s">
<animate attributeName="width" to="100" dur="10s">
```

  A largura de um objeto é alterada de quatro formas diferentes durante um mesmo período de 10 segundos. Na primeira delas, a largura varia de um valor inicial de 50 até um valor final de 100 *pixels*. Na segunda, o mesmo valor inicial é usado, no entanto, ao invés de um valor final, uma diferença de 25 *pixels* do valor da largura é aplicada. Na terceira definição a largura varia de seu valor base de uma diferença de 30 *pixels*. Na quarta, a mesma largura varia de seu valor base, qualquer que seja, até o valor final de 100 *pixels*.

- `values` e `calcMode`, que definem um conjunto de valores intermediários pelos quais o atributo alvo deve passar durante sua animação e o modo de interpolação que deve ser

seguido para que esses valores sejam cumpridos. O atributo `calcMode` pode ter os valores *discrete*, indicando que a função deve saltar entre os valores definidos sem qualquer tipo de interpolação; *linear* (valor *default*), indicando que deve ser aplicada uma interpolação linear simples; *paced*, que define que deve ser aplicada uma interpolação uniforme durante toda a animação; e *spline*, que define que a interpolação deve ser feita de acordo com uma função definida por uma curva cúbica de Bezier (*cubic Bezier spline*) [Foley], usando como pontos os valores especificados no atributo `keyTimes` e como pontos de controle para cada intervalo os valores do atributo `keySplines`, apresentados logo adiante. A Figura 2.6 apresenta esse modos de interpolação. Nela é considerado o seguinte exemplo de definição de animação:

```
<animate dur="30s" values="0; 1; 2; 4; 8; 15" calcMode="a, b ou c">
```

A Figura 2.6 apresenta a aplicação dos três possíveis modos de cálculo. Em (a), a animação é feita usando-se o modo *discrete*; O atributo permanece com cada um dos valores relacionados pelo mesmo período de tempo. Em (b), o modo de cálculo utilizado é *linear*; a duração é dividida em cinco partes iguais e os valores são interpolados linearmente, determinando variações uniformes entre os valores parciais do atributo. Em (c), utiliza-se o modo *paced*; o valor do atributo varia uniformemente entre o valor inicial e final.

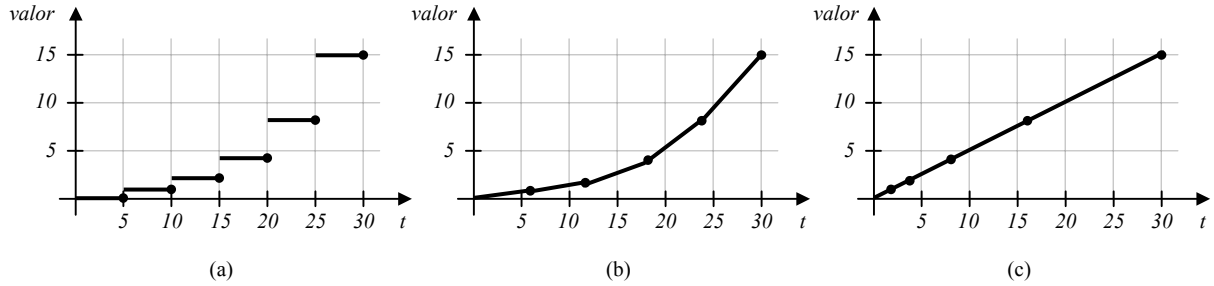


Figura 2.6 – Modos de interpolação em SMIL.

- `keyTimes`, que define um conjunto de valores em ponto flutuante (entre 0 e 1) usado para controlar o ritmo da animação. Cada valor corresponde, na ordem em que são definidos, a um dos valores do atributo `values` e define um percentual da duração simples do atributo onde o valor do atributo deve ser alterado. De acordo com o modo de interpolação especificado, o valor é alterado de forma diferente. Por exemplo, pode-se definir a seguinte animação:

```
<animate attributeName="x" dur="10s" values="0; 50; 100"
keyTimes="0; 0.8; 1" calcMode="paced">
```

São dois intervalos de animação. O valor do atributo `x` é iniciado com valor 0 (zero). Após 8s (80% da duração) ele deve ter o valor 50 e, ao final da animação, 100. O valor varia vagarosamente no primeiro intervalo e rapidamente, no segundo.

- `keySplines`, que define um conjunto de pontos de controle Bezier (*Bezier control points*). Esses pontos definem uma função cúbica de Bezier que controla o ritmo de variação do valor do atributo animado dentro do intervalo especificado. Os valores dos pontos devem estar entre 0 e 1.
- `accumulate` e `additive`, que definem respectivamente se a função definida deve ser cumulativa e aditiva. O atributo `accumulate` pode recebe o valor `sum`, para uma animação cumulativa, ou `none` (valor *default*), para uma animação não-cumulativa. O atributo `additive` pode receber o valor `sum`, para uma animação aditiva, ou `replace` (valor *default*), para uma animação não-aditiva.

O elemento `set` é usado para definir um valor constante para um atributo que deve permanecer durante um tempo definido. Define apenas um elemento adicional, `to`, que deve conter o valor do atributo animado.

O elemento `animateMotion` permite uma animação do posicionamento de um objeto durante sua apresentação. Ele possui os seguintes atributos adicionais:

- `from`, `to`, `by`, `values`, `keyTimes`, `keySplines`, `calcMode`, `accumulate` e `additive`, semelhantes aos definidos para o elemento `animate`.
- `origin`, que define as coordenadas da origem a ser utilizada para a animação.
- `path`, que permite definir um caminho a ser percorrido pelo objeto durante a animação.

O elemento `animateColor` permite definir a animação de um atributo de cor. Ele também possui os atributos `from`, `to`, `by`, `values`, `keyTimes`, `keySplines`, `calcMode`, `accumulate` e `additive`, definidos de forma semelhante aos outros elementos.

A definição do congelamento de animações é feita através da adição do atributo `fill` ao elemento que define a animação. Seu valor deve ser `freeze`. Esse atributo é definido nos módulos de temporização e sincronização de SMIL e define o comportamento do valor do atributo após o término da animação. Por exemplo, a animação definida por

```
<animate attributeName="x" from="15" to="100" dur="10s" fill="freeze">
```

especifica que o atributo `x` deve permanecer com o valor 100 após o término da animação. A ação do atributo `fill` é limitada pela duração ativa do elemento que contiver a definição do elemento de animação. Caso o valor do atributo `fill` seja `remove` (valor *default*), o valor base do atributo animado é restaurado ao fim da duração ativa da animação.

Em SMIL também é permitido definir repetições de animações. Existem dois atributos, `repeatCount` e `repeatDur`, que definem, respectivamente, quantas vezes uma animação deve ser repetida e a duração total de todas as repetições de uma animação. Esses atributos podem ser aplicados a qualquer um dos elementos definidos anteriormente.

SMIL também possui um conjunto de módulos usados para a definição de transições. Há dois métodos para a definição de transições:

- através de **estilos de transição**. O autor define uma série de classes de transição que podem ser usadas em vários objetos de um mesmo documento; ou
- através de **transições *inline***, quando cada objeto tem sua transição definida. O autor tem controle total sobre as propriedades da transição.

Um estilo é definido através do elemento `transition`. Ele contém uma série de atributos que permitem definir as características da transição. Os mais importantes são:

- `id`. Define uma identificação única para a transição;
- `type`. Define o tipo da transição. SMIL define uma taxonomia de tipos de transições que podem ser usados [SMIL2];
- `subtype`. Define o subtipo de transição. É opcional e também e é definido para cada um dos tipos da taxonomia de transições;
- `dur`. Define a duração da transição;

O uso de transições por objetos de mídia é feito através dos atributos `transIn` e `transOut`. Eles definem as transições a serem usadas no início e no fim da apresentação de um objeto. A apresentação de tanto a transição de início com da de fim, ocupam parte da duração ativa do objeto. Por exemplo, considerando a apresentação de um objeto, cuja duração ativa é de 10 s, uma transição de início com duração de um segundo e uma transição de fim com duração de dois segundos, reduziria a apresentação "limpa" do objeto a sete segundos; os três segundos restantes seriam usados na aplicação dos efeitos de transição.

A Figura 2.7 apresenta um trecho de documento SMIL onde são definidos dois estilos de transição, *wipe1* e *xfade*. Esses estilos são usados, posteriormente, por um objeto do documento, *img1*. *wipe1* define uma transição inicial com duração de dois segundos, enquanto *xfade*, uma transição final de um segundo. Ambas as transições tem seu tipo de findo através do atributo `type`. A segunda transição, em particular possui um subtipo, definido pelo atributo `subtype`.

```
<transition id="wipe1" type="barWipe" dur="2s"…/>
<transition id="xfade" type="fade" subtype="crossfade" dur="1s"…/>
...
<img id="img1" src="img1.jpg" dur="15s" transIn="wipe1" transOut="xfade" …/>
```

Figura 2.7 – Exemplo de definição e uso de transições em SMIL.

A transição de início tem precedência maior que a de fim. Assim, quando as durações dos efeitos de transição de início e fim se sobrepuserem, somente a de início é apresentada. SMIL define outras regras para a aplicação de efeitos de transição. Elas são detalhadas em [SMIL2].

Transições *inline* permitem uma definição mais detalhada de efeitos de transição. Elas são criadas através do elemento `transitionFilter`, para objetos específicos de um documento. Além dos mesmos atributos do elemento `transition`, com o número de repetições (`repeatCount`) ou a duração de uma seqüência de repetições (`repeatDur`). Outros atributos são definidos em [SMIL2].

2.2 MADEUS

O modelo Madeus [Villard00a e Layaïda97] é um modelo de documentos hipermídia cujos documentos são organizados segundos quatro dimensões diferentes: lógica, temporal, espacial e hipermídia. Um sistema de autoria [Layaïda97 e Jourdan97] de mesmo nome permite a criação e edição de desses documentos de forma interativa e declarativa (a sintaxe dos documentos é definida segundo uma DTD XML [Villard00b]). O sistema Madeus é baseado na definição de restrições. Tanto as características espaciais como as temporais são traduzidas em restrições [Carcone97a, Jourdan97, Jourdan98a, Layaïda97 e Tardif00] e mantidas pelo sistema durante a apresentação do documento.

A Figura 2.8 apresenta a estrutura básica de um documento Madeus. Ela é formada por quatro seções básicas: *mediaContent*, *mediaUse*, *temporal* e *spatial*.

```
<madeus>
   <mediaContent>...</mediaContent>
   <mediaUse>...</mediaUse>
   <temporal>...</temporal>
   <spatial>...</spatial>
</madeus>
```

Figura 2.8 – Estrutura de um documento Madeus.

As seções *mediaContent* e *mediaUse* definem a estrutura lógica do documento. Na seção *mediaContent* são definidos os objetos de mídia que farão parte do documento. Cada objeto de mídia pode ser definido de forma estruturada, por exemplo, objetos de vídeo podem ser subdivididos em seqüências, cenas, etc. [Roisin99]. Cada objeto de mídia deve identificar propriedades intrínsecas, como suas dimensões e duração. A seção *mediaUse* define usos específicos dos objetos definidos na seção anterior, adicionando características de estilo, como a cor de fonte a ser utilizada, por exemplo.

O modelo lógico permite que os objetos de mídia sejam organizados de forma hierárquica, através de elementos (*C-Group* e *U-Group*) que definem agrupamentos.

A seção *temporal* define como os objetos de um documento devem ser sincronizados temporalmente durante sua apresentação. O modelo temporal é baseado na idéia de que a apresentação de cada objeto é definida por um intervalo [Jourdan99c, Layaïda97 e Jourdan97] definido por um início, uma duração e um fim. Cada um desses valores é definido por uma tupla [*min*, *pref*, *max*], cujos valores podem variar entre zero e *indefinido*, permitindo que o intervalo seja adaptado pelo formatador.

Um objeto de mídia, definido na seção *mediaUse*, é associado a um elemento (*Interval*) que define o intervalo no qual o objeto deve ser apresentado. A duração de um intervalo pode se sobrepor à duração intrínseca do objeto associado.

A sincronização temporal da apresentação pode ser definida através de nós de composição e relações temporais. Nós de composição são definidos pelo elemento *T-Group* (a restrição *during* [Allen83] é definida implicitamente entre um intervalo filho e seu pai). Relações temporais podem ser definidas entre os intervalos de um nó de composição. Por exemplo, pode-se definir que dois objetos sejam apresentados em seqüência definindo-se a relação

temporal *meets* entre os intervalos usados em sua apresentação. [Allen83] apresenta as relações temporais usadas por Madeus. Adicionalmente, cada grupo temporal pode ser associado a um operador temporal *par* ou *seq*, assim como em SMIL [SMIL1 e SMIL2].

A seção *spatial* define como os objetos de um documento podem ser organizados espacialmente durante sua apresentação. A sincronização espacial de um documento é definida através de uma hierarquia de regiões (elemento *shape*), que podem ser agrupadas (elemento *S-Group*) e relacionadas espacialmente. Cada região possui um conjunto de propriedades que definem seu posicionamento, suas dimensões e sua prioridade de apresentação. Cada região é associada a um objeto, definido na seção *mediaUse*, identificando onde ele deve ser apresentado.

[Carcone97b e Carcone97a] apresentam um conjunto de relações espaciais que podem ser aplicadas às regiões de um mesmo grupo:

- *top-aligned*, *bottom-aligned*, *left-aligned* e *right-aligned*, que definem que os objetos apresentados nas regiões relacionadas devem permanecer alinhados com relação ao topo, à base, à esquerda e à direita, respectivamente;
- *vert-center* e *horiz-center*, que definem que os objetos apresentados nas regiões relacionadas devem permanecer centralizados verticalmente ou horizontalmente;
- *left-of(d)*, *right-of(d)*, *above(d)* e *below(d)*, que definem respectivamente que um objeto, apresentado em uma região, deve permanecer à esquerda, à direita, acima ou abaixo de outro objeto (de referência), apresentado em outra região, de uma distância *d*;
- *equal-width* e *equal-height*, que definem que objetos associados a regiões relacionadas devem ter a mesma largura ou altura durante sua apresentação.

A dimensão hipermídia de um documento Madeus define como os objetos se relacionam através de elos. Eles são definidos tanto na seção *mediaUse* como na seção *temporal*.

Um documento Madeus pode conter relações espaço-temporais, definidas como relações espaciais que dependem do tempo. Por exemplo, a relação

```
<S-Relation name="left_align" shapes="all" interval="interval1">
```

define que todas as regiões de um grupo devem permanecer alinhadas à esquerda enquanto o objeto associado ao intervalo *interval1* estiver sendo apresentado.

[Villard98] explora a definição de relações espaço-temporais (não presentes na versão atual do modelo Madeus). São definidos os seguintes tipos de relação espaciais dependentes do tempo:

- relações espaciais pontuais. Por convenção a relação espacial se aplica sobre o instante final do intervalo do primeiro objeto da relação. Por exemplo, a relação abaixo é válida quando o intervalo *A.I1* é terminado.

```
A.I1 Top_Align B
```

- relações espaciais permanentes. São válidas enquanto os objetos relacionados estejam sendo apresentados. Caracterizam-se pelo prefixo *Always*.

```
A Always_Top_Align B
```

- relações espaciais temporizadas. São válidas durante o intervalo especificado entre parênteses, que pode ser definido com base na duração da apresentação de outros objetos.

```
A Left_Align(10s) B

A Left_Align(C.during) B
```

Além das **relações espaciais dependentes de tempo**, é possível definir **relações temporais dependentes do espaço**. Elas são criadas através de eventos dentro de composições e possuem a seguinte sintaxe:

```
<EVENT> { <SOURCE> Objeto
          <CONDITION> condições
          <ACTIONS> ações }
```

Cada definição de evento é composta de:

- um objeto de origem;
- uma lista de condições, definidas sobre os atributos do objeto de origem; e
- uma lista de ações que devem ser executadas quando as condições forem satisfeitas.

A Figura 2.9 apresenta um exemplo, onde o intervalo *B.I2* se inicia assim que o objeto *A* se posiciona em (50,50).

```
<EVENTS> {
   <SOURCE> A
   <CONDITION> <POS> 50 50
   <ACTIONS> : start B.I2
}
```

Figura 2.9 – Definição de uma relação temporal dependente do espaço em Madeus.

[Tran_Thuong01] apresenta extensões ao modelo Madeus. São definidos sub-elementos, que permitem uma especificação mais detalhada da sincronização espacial e temporal de um documento.

A criação de sub-elementos na estrutura lógica de um documento corresponde à definição de *sub-objetos de mídia*, que permitem uma estruturação mais apurada de um objeto de mídia para efeitos de sincronização. Como mencionado anteriormente, um vídeo pode ser estruturado em seqüências, cenas, etc., que podem ser identificados como sub-objetos de mídia. Sub-objetos de mídia não são considerados objetos de mídia comuns, devidos às suas restrições temporais (seu intervalo de apresentação, identificado por um *subintervalo*) e espaciais (seu posicionamento, identificado por uma *sub-região*) intrínsecas com relação ao objeto de que fazem parte.

O modelo temporal é estendido, podendo-se definir subintervalos. Um subintervalo é um elemento temporal, sendo definido por um início, duração e um fim. Cada subintervalo apresenta uma porção do conteúdo de um objeto de mídia (sub-objeto de mídia) ou uma animação (como será explicado adiante). Por definição, um subintervalo contém uma restrição *during* com relação ao seu intervalo pai. Subintervalos não podem conter outros subintervalos.

No modelo espacial são definidas sub-regiões, usadas na apresentação de parcelas de um objeto de mídia. Uma sub-região é um elemento espacial, possuindo propriedades de posicionamento e dimensão, que apresenta o conteúdo de um sub-objeto de mídia de um objeto de mídia associado à sua região mãe. Por exemplo, pode-se usar uma sub-região na apresentação de um objeto (como uma pessoa) de uma cena de um objeto de vídeo ou uma palavra de um objeto de texto. Esse sub-elemento facilita a localização de partes de um objeto de mídia em uma região.

Também é definido um modelo de animação que permite especificar a animação de atributos de um objeto durante sua apresentação. Animações são consideradas objetos de mídia

contínua pelo modelo, definidos através de elementos de animação, como *animate animateMotion*, *animateColor* e *animateTransformation*[5], semelhantes aos definidos em SMIL (ver Seção 2.1).

A definição de uma animação separa a especificação de seu comportamento (da variação dos valores dos atributos) de sua temporização. Ela é feita em duas partes: a criação de um elemento de animação, que é identificado como um objeto de mídia (na seção *mediaUse*); e sua associação a um subintervalo (como ocorre com qualquer objeto de mídia) que esteja contido no intervalo associado ao objeto de mídia onde a animação deve ocorrer.

A definição do comportamento de uma animação considera o intervalo de tempo abstrato [0, 1]. A associação de uma animação a um intervalo adapta esse intervalo abstrato ao intervalo real, gerando a função que define o efeito de animação final. A Figura 2.10 ilustra a adaptação de uma animação cujo comportamento é definido pelo atributo *keyTimes* (ver Seção 2.1). Esse atributo contém o valor "0; 0.2; 0.8; 1". O intervalo abstrato é mapeado em sub-intervalos concretos, com durações definidas explicitamente.

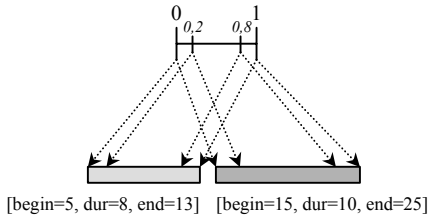


Figura 2.10 – Definição de animações no Madeus.

Essa forma de definição de animações permite que uma animação seja associada a vários objetos, em diferentes contextos.

2.3 SSTS

SSTS (*Synchronization Specification method for Temporal and Spatial events*) [Nang97] é um método de especificação da sincronização espacial e temporal da apresentação de objetos multimídia. Esse método permite especificar eventos temporais e espaciais e relações temporais entre esses eventos de uma maneira uniforme.

[5] Na verdade, o elemento *animateTransformation* é definido pelo modelo de animação de SVG (*Scalable Vector Graphics*) [Bowler01] que é uma extensão ao modelo de animação usado em SMIL.

O método é baseado na construção de um grafo de especificação de sincronização, onde os nós, representando eventos espaciais ou temporais dos objetos envolvidos, são relacionados temporalmente através de arestas.

Um evento temporal representa o início ou o fim da exibição de um objeto enquanto que um evento espacial, uma modificação de alguma das características espaciais de um objeto, como, por exemplo, o movimento de um objeto.

O grafo de especificação de sincronização é um grafo orientado composto de um conjunto de nós de início, um conjunto de nós de fim, um conjunto de nós espaciais, um conjunto de arestas de exibição e um conjunto de arestas de controle.

Um nó de início representa o evento de início da exibição de um objeto, enquanto que um nó de fim, o seu término. O número de nós de início e de fim de um grafo de sincronização deve corresponder ao número de objetos de mídia envolvidos.

No nó de início de um objeto, pode-se especificar suas características iniciais de apresentação. É possível, por exemplo, determinar suas dimensões, seu posicionamento e seu conteúdo.

Um nó espacial representa um evento espacial como o início de um movimento executado pelo objeto. Cada um dos conjuntos de nós é composto de subconjuntos de nós *OR* e *AND*. Um nó *OR* é ativado quando qualquer uma de suas arestas entrantes encontra-se ativada, enquanto um nó *AND* necessita que todas suas arestas entrantes estejam ativadas para que seja ativado.

Uma aresta de exibição representa a exibição de um objeto de mídia e deve conectar o nó de início ao respectivo nó de fim de um objeto de mídia, podendo ter zero ou mais nós espaciais entre eles.

Uma aresta de controle pode ser de sentido único (*1-CE*), ou duplo (*2-CE*). Uma aresta de controle de sentido único ativa seu nó alvo após um tempo especificado, enquanto que uma aresta de controle de sentido duplo ativa seus nós de origem e de destino simultaneamente. Ambas as arestas representam relações temporais entre eventos.

A Figura 2.11 ilustra a representação gráfica de nós e arestas em um grafo de sincronização.

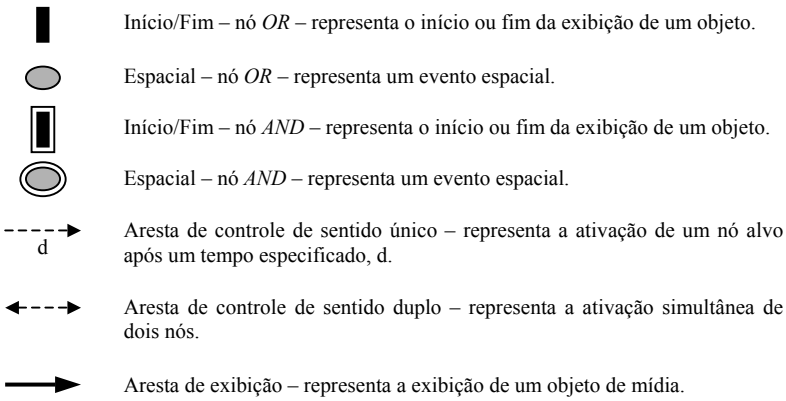


Figura 2.11 – Notação gráfica usada no SSTS.

A ativação dos eventos temporais e espaciais é controlada pelas arestas de controle. Abaixo são apresentadas as regras de ativação usadas em SSTS:

- a ação de um nó é iniciada se ele estiver ativado completamente;
- um nó de início sem arestas de controle entrantes é ativado no início da apresentação;
- um nó do tipo *AND* é ativado quando todas as suas arestas entrantes são ativadas;
- um nó do tipo *OR* é ativado quando qualquer uma de suas arestas entrantes é ativada;
- uma aresta de exibição só ativa o seu respectivo nó de fim ao término da exibição do objeto de mídia que ela representa;
- um nó com uma aresta de controle de sentido único só ativa seu nó alvo correspondente após terminado o tempo especificado, que pode ser igual a zero;
- dois nós conectados por uma mesma aresta de controle de sentido duplo só são ativados se todas suas arestas entrantes estiverem ativadas.

A especificação de eventos temporais em SSTS é bastante simples. Para se representar a apresentação de um objeto de mídia, basta que se conecte um nó de início a um nó de fim por uma aresta de exibição. O nó de início identifica o objeto de mídia associado e sua posição inicial, caso o nó esteja associado a um objeto de mídia visual. A Figura 2.12 apresenta alguns exemplos de sincronização entre eventos temporais. Em (a), o nó de início de P_β é completamente ativado τ_δ instantes de tempo após o término da apresentação de P_α. Em (b), a apresentação dos objetos P_α e P_β é iniciada simultaneamente, devido à ativação do nó de início de P_β pelo nó de início de P_α. Em (c), a apresentação dos objetos P_α e P_β é iniciada

simultaneamente, mas o seu término só ocorre quando os nós de fim de ambos os objetos estejam completamente ativados.

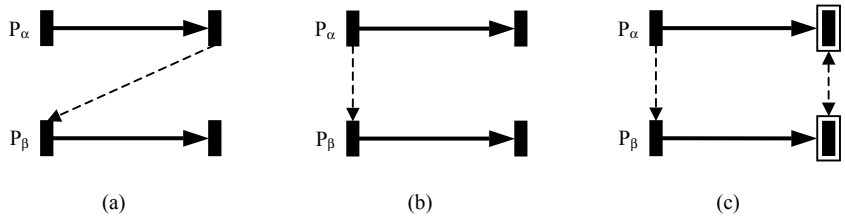


Figura 2.12 – Exemplos de especificação de sincronização de eventos temporais.

Em SSTS é possível especificar as relações temporais de alto nível, *before*, *meets*, *overlaps*, *during*, *starts*, *finishes* e *equals*, definidas por Allen [Allen83].

O SSTS permite a definição de mais eventos espaciais além da especificação estática do posicionamento de um objeto através de seu nó inicial. Os eventos espaciais básicos são:

- `AbsolutePath(time, pos)`. Move o objeto para a posição definida durante o intervalo de tempo definido;
- `RelativePath(time, dest_obj, dest_pt, src_pt, dist, angle)`. Move o objeto para a posição relativa ao objeto `dest_obj`.
- `CurveMotion(time, pos`$_1$`, pos`$_2$`, ..., pos`$_n$`)`. Move o objeto durante o intervalo determinado pela curva especificada pelos pontos `pos`$_1$, `pos`$_2$, ..., `pos`$_n$;
- `ShapeMotion(time, shape)`. Move o objeto juntamente com o objeto `shape` durante o intervalo determinado;
- `Visibility(mode)`. Exibe ou esconde o objeto;
- `Basis(obj)`. Põe o objeto sobre o objeto `obj`;
- `Scaling(time, x_pt, y_pt)`. Dimensiona o objeto segundo os valores percentuais de `x_pt` e `y_pt` durante o tempo determinado.

Como mencionado anteriormente, eventos espaciais são especificados entre os nós de início e fim de um objeto, sobre sua aresta de exibição. Vários eventos espaciais podem ser especificados em uma mesma aresta de exibição, não importando sua ordem, haja vista que eles somente são ativados por arestas de controle.

A principal característica de SSTS é permitir a especificação da sincronização de eventos temporais e espaciais de forma uniforme. A Figura 2.13 apresenta um exemplo de grafo de

sincronização onde são definidas relações entre eventos espaciais e temporais. Nele, os objetos *videoA* e *audioA* são iniciados simultaneamente. O término da exibição de *audioA* ativa a execução de um evento espacial em *videoA* (`RelativePath()`). Após esse movimento ser terminado e *d2* instantes de tempo depois, *audioB* é iniciado. Enquanto isso, *d1* instantes de tempo depois do início da exibição de *audioA*, a apresentação de *imageA* é iniciada, o que ativa o evento espacial `AbsolutePath()`. O término da exibição de *imageA* se dá simultaneamente ao final da apresentação de *audioB* e *videoA*.

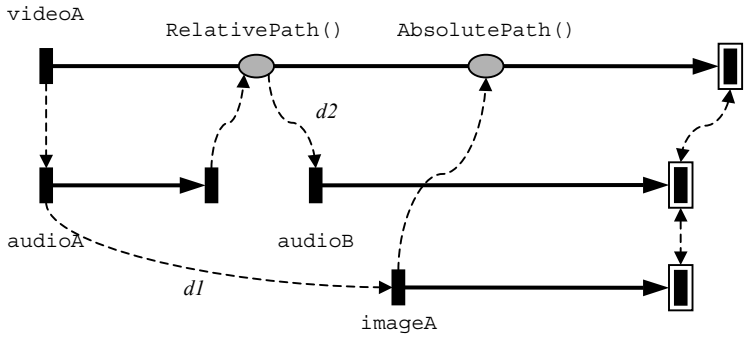


Figura 2.13 – Exemplo de especificação de sincronização de eventos espaço-temporais.

2.4 VAZIRGIANNIS

Vazirgiannis *et al.* [Vazirginannis00 e Theodoridis96] propõem um modelo integrado para a especificação da sincronização temporal e espacial de documentos hipermídia. Nesse modelo, a forma como os objetos de um documento são apresentados é determinada por um conjunto de **relações espaço-temporais**.

Uma apresentação é definida por um conjunto de composições, que agrupam os objetos de um documento. Relações espaço-temporais são definidas em composições e só podem relacionar os objetos de mídia contidos nela. Uma composição pode conter outras composições recursivamente. O modelo assume que cada objeto integrante de uma composição define suas características espaço-temporais, como largura, altura, duração, etc.

A definição de relações espaço-temporais é o ponto central do modelo. Cada relação é definida por uma relação temporal e uma relação espacial.

O modelo considera a apresentação de um objeto de mídia como um intervalo (assim como [Allen83]). O início e o fim (que pode ser natural ou forçado) da apresentação de um objeto são considerados eventos. O modelo também considera que a apresentação de um objeto pode ser interrompida e, posteriormente, continuada.

O modelo define um conceito importante de **instância temporal**, que corresponde a uma medida temporal, relativo ao início temporal da apresentação.

Vários operadores podem ser associados aos eventos definidos acima:

- considerando um objeto de mídia, *A*, *A*> representa o início da apresentação do objeto, *A*<, o seu fim natural, *A*!, o seu fim forçado, *A*||, sua interrupção e *A*|>, a retomada de sua apresentação;
- considerando dois objetos de mídia, *A* e *B*, a expressão *A*op1 *t B*op2 representa todas as relações temporais possíveis entre os dois objetos, onde op1 ∈ {>, <, ||, |>} e op2 ∈ {>, !, ||, |>}. Todas as relações temporais de Allen [Allen83] podem ser definidas através dessa expressão;
- $t_{A\text{op}}$ corresponde às instâncias temporais dos eventos *A*op, onde op ∈ {>, <, !, ||, |>};
- d_A é a duração temporal da apresentação do objeto de mídia *A*.

O modelo assume que objetos de mídia visual são representados espacialmente na forma de retângulos. A sincronização espacial entre objetos é definida através de relações espaciais que expressam relações topológicas, de direção e de distância. O modelo utiliza um conjunto de 169 (13^2) relações espaciais R_{i_j} (i = 1, ..., 13 e j = 1, ..., 13) que determinam todas as possíveis relações de direção entre dois objetos, incluindo informações topológicas [Papadias95 e Papadias97].

A definição da relação espacial entre dois objetos é completada com a especificação da distância entre eles, que deve ser determinada entre os vértices mais próximos dos objetos. Cada objeto possui quatro vértices, numerados de 1 a 4, em sentido horário, começando do vértice inferior esquerdo. A Figura 2.14 apresenta um exemplo de especificação de distância entre dois objetos.

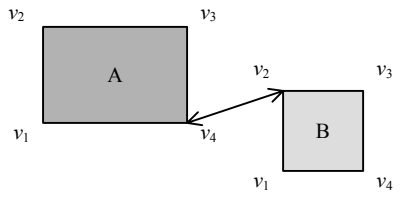


Figura 2.14 – Distância entre dois objetos segundo o modelo.

Uma relação espaço-temporal entre dois objetos de uma mesma composição é, então, definida por `ST_R(sp_rel, temp_rel)`, onde `sp_rel` é a relação espacial e `temp_rel` é a

relação temporal. A relação espacial, `sp_rel`, é definida pela tupla (R_{i_j}, v_i, v_j, x, y), onde R_{i_j} é uma das relações de direção, v_i e v_j representam os vértices mais próximos dos objetos relacionados, e x e y correspondem às distâncias entre esses vértices nos eixos horizontal e vertical.

[Vazirginannis97] apresenta um modelo para a definição de apresentações multimídia interativas (o modelo apresentado até aqui não leva interatividade em consideração). Como poderá ser observado adiante, esse modelo utiliza a construção de relações espaço-temporais apresentada anteriormente.

Esse modelo considera aspectos temporais de apresentações multimídia. Uma instância temporal corresponde a um ponto único no tempo de tamanho zero. Qualquer uma das relações *before* (<), *after* (>) e *equals* é válida na definição de relações temporais entre instâncias temporais. Essa é uma representação de tempo mais simples que a definida por [Allen83] que considera relações entre intervalos. No entanto, os modelos são equivalentes, já que um intervalo pode ser representado por uma diferença entre duas instâncias temporais.

Aspectos espaciais também são considerados na construção de apresentações multimídia. São eles: a representação do posicionamento de objetos e de relações espaciais entre eles; a animação de objetos com relação a um ou mais objetos; e eventos espaciais produzidos por ações relacionadas à animação de objetos ou a relações espaciais existentes entre eles. Eventos espaciais são produzidos quando dois objetos encontram-se justapostos ou quando um objeto encontra-se em movimento, por exemplo.

Um conceito relacionado à sincronização espacial considerado por esse modelo é o de **instância espacial**, representada pela tupla (x_1, y_1, x_2, y_2), correspondente à área do retângulo correspondente a um evento. Pode ser tanto usado para representar um clique, por exemplo, ou o retângulo correspondente a um objeto.

A interatividade do modelo é alcançada através da definição de **eventos** e **cenários multimídia**. O modelo utiliza a definição de que um evento é acionado pela ocorrência de uma ação e é associado a uma instância temporal e espacial.

O modelo considera os seguintes tipos de eventos:

- eventos gerados pela interação do usuário;

- eventos gerados internamente a objetos (“intraobjetos”), que podem estar relacionados à execução de operações sobre os objetos ou à mudança de estados dos objetos, que podem ser temporais ou espaciais;
- eventos gerados por dois ou mais objetos, que podem ser ativados por relações espaciais e/ou relações temporais entre os objetos;
- eventos gerados pela apresentação, relacionados com o seus estado; e
- eventos definidos pelo usuário, que podem ser definidos com relação ao conteúdo de objetos da apresentação.

Esses eventos são modelados em uma hierarquia que é disponibilizada para autores de apresentações multimídia.

Cada evento ocorre devido a uma **ação**, como citado anteriormente, causada por um **sujeito**, em um ponto determinado do tempo e/ou em uma posição, possuindo uma **assinatura espaço-temporal**, e pode afetar um ou mais objetos, chamados de **objetos** do evento.

Um cenário multimídia representa uma definição de alto nível da funcionalidade de uma apresentação multimídia, considerando suas características espaciais e temporais e as possíveis interações que podem existir entre o usuário, a aplicação e os objetos envolvidos. Cenários temporais são construídos através de um conjunto de tuplas (*scenario tuples*) [Vazirgiannis96b]. Cada tupla é definida pelos seguintes atributos:

- *start_time*. Identifica o evento que ativa a execução das ações descritas no atributo *action_expression*;
- *stop_time*. Determina o evento que encerra a execução da tupla;
- *action_expression*. Descreve uma lista de ações que serão executadas e seus domínios espacial e temporal;
- *content_condition*. Define uma expressão que indica uma condição correspondente ao conteúdo dos objetos incluídos nas tupla. Usado na definição de eventos pelo usuário;
- *synch_events*. Nomes dos eventos que identificam o início e o fim da apresentação da tupla, que podem ser usados na sincronização da apresentação;
- *constraints*. Definem uma série de restrições (espaciais, temporais, etc.) que devem ser respeitadas durante a execução da tupla;
- *is_active*. Determina se a tupla está sendo executada.

O modelo permite a criação de eventos compostos. Existem dois tipos de composição: algébrica e espaço-temporal. A composição de eventos usa alguns conceitos básicos:

- ponto de referência espaço-temporal, Θ, É início espaço-temporal de uma apresentação, que corresponde ao início da apresentação, associado ao canto superior esquerdo do dispositivo de exibição (início espacial);
- intervalo temporal, *t_int*, correspondente à distância temporal entre dois eventos, *(e_1, e_2)*; e
- eventos relativos *r_e = (e_1, t_int)* que ocorre um intervalo temporal após a ocorrência de outro evento.

As composições algébricas de eventos definidas pelo modelo são apresentadas a seguir.

Disjunção

e = OR (e_1, ..., e_n). Esse evento ocorre quando ao menos um dos eventos e_i, $1 \leq i \leq n$ ($i \in \aleph$).

Conjunção

e = ANY (k, e_1, ..., e_n). Esse evento ocorre quando ao menos *k* eventos e_i, $1 \leq i \leq n$ ($i \in \aleph$), ocorrem.

e = SEQ (e_1, ..., e_n). Esse evento ocorre quando a seqüência de eventos e_i, $1 \leq i \leq n$ ($i \in \aleph$), ocorre na ordem definida, não importando se outros eventos ocorram entre os eventos especificados.

e = TIMES (n, e_1). Esse evento ocorre quando o evento e_1 ocorre *n* vezes consecutivas.

Outras composições algébricas refletem restrições a serem aplicadas na definição de uma apresentação em intervalos temporais específicos:

Inclusão

e = IN (e_1, t_int), que ocorre quando o evento e_1 ocorre durante o intervalo temporal *t_int*.

Negação

e = NOT (e_1, t_int), que ocorre quando o evento e_1 **não** ocorre durante o intervalo temporal *t_int*.

Eventos estritamente consecutivos

e = S_CON (e_1, ..., e_n), que ocorre quando a seqüência de eventos definida ocorre na ordem determinada, sem que outro evento ocorra entre eles.

A composição espaço-temporal é feita através de uma série de operadores temporais e espaciais. Os operadores temporais representam as possíveis relações temporais entre eventos:

e = after (e_1, e_2), que ocorre quando os eventos e_1 e e_2 ocorrem e a assinatura temporal de e_1 é menor que a de e_2.

e = before (e_1, e_2), inversa à anterior.

e = simultaneously (e_1, e_2), que ocorre quando os eventos e_1 e e_2 ocorrem e as assinaturas temporais de ambos os eventos são a mesma.

As relações espaciais são definidas por uma das relações de direção possíveis entre dois objetos, definidas anteriormente [Papadias95 e Papadias97].

O modelo de composição espaço-temporal é completado com a definição de uma distância espaço-temporal, que corresponde à distância euclidiana entre as assinaturas espaço-temporais dos eventos envolvidos.

[Vodislav00], baseado nos trabalhos anteriores, apresentados em [Vazirgiannis96a, Vazirginannis00 e Vazirginannis97] e [Vodislav97], propõe a definição de animações interativas para documentos multimídia de forma declarativa. Ele define um modelo de animação 2½D composto de um modelo básico de animação e um modelo de cenário. O modelo básico de animação define as noções de objetos animados, composições de objetos e de transformações de animação. O modelo de cenário define um mecanismo que permite detectar eventos do usuário ou de objetos do documento, compor eventos. Cada cenário descreve a animação de objetos de forma declarativa.

O modelo básico de animação identifica três tipos de animação:

- *transformações espaciais*, que determinam mudanças de posicionamento (translação), de orientação (rotação) e de formato (dimensionamento) de objetos;
- *transformações gráficas*, que alteram o aspecto gráfico de objetos, como sua cor e estilo;
- *transformações de conteúdo*, que alteram o conteúdo de objetos ou habilitam o controle de sua apresentação;

O modelo permite a combinação de animações. É definida uma função que combina os efeitos das animações originais. Caso elas comecem e terminem ao mesmo tempo, a função final acumula os efeitos das animações básicas; caso contrário, devem ser definidas mais de uma função para que o efeito de animação final seja alcançado.

O modelo define dois tipos de comportamento para as animações, quando elas são aplicadas a um objeto que esteja sendo animado. Pode haver transformações de substituição (*replacement*), que se sobrepõem à animação ativa ou transformações cumulativas, que são aplicadas sobre o resultado da animação ativa.

Objetos animados podem ser simples ou compostos (de outro objetos). Um objeto animado é definido como uma tupla (*G*, c*ontents*, *name*, *Sp*, *Gr*, *vis*, *as*, *it*, *comp*), onde:

- *G* define a figura geométrica representada pelo objeto, determinando um conjunto de pontos;
- *contents* identifica o objeto de mídia que deverá preencher a área do objeto animado;
- *name* identifica o objeto;
- *Sp* define o estado espacial do objeto. Ele é composto da posição, do ângulo e da escala do objeto;
- *Gr* identifica o estado gráfico do objeto. Define uma tupla que contém atributos gráficos como a cor e o estilo do objeto;
- *CT* é uma tupla que descreve as transformações contínuas de translação, orientação, dimensionamento e mudança de cor do objeto. Cada transformação possui um estado que pode ser preparado (*idle*), ativo (*active*) ou suspenso (*suspended*);
- *vis* determina o estado de visibilidade do objeto, que pode ser falso (escondido) ou verdadeiro (mostrado);
- *as* indica o estado (ativo ou suspenso) da apresentação do objeto;
- *it* é a temporização interna do objeto;
- *comp* é o conjunto de objetos contidos, no caso de um objeto composto.

Cada objeto possui uma série de métodos que podem ser usados para modificar o seu comportamento de animação. São eles:

- *start_obj(pos, vis)*, *pause_obj()*, *resume_obj()*, *stop_obj()* que permitem, respectivamente, iniciar, interromper, reassumir e terminar a apresentação do objeto;
- *start(tr, law, ref)* que inicia uma transformação contínua;
- *pause(tr)*, *resume(tr)*, *stop(tr)* que alteram o estado de uma transformação;

- *translation(dx, dy)*, *rotation(a, C)*, *scaling(sx, sy, C)*, *color(r, g, b)*, etc., que aplicam transformações espaciais ou gráficas ao objeto;
- *change_contents(value)* que altera o conteúdo de um objeto de animação;
- *start_contents()*, *pause_contents()*, *resume_contents()* e *stop_contents()* que controlam a apresentação de objetos de mídia contínua, como vídeo e áudio;
- *hide()* e *show()* que controlam o estado de visualização do objeto;
- *raise()* e *back() que modificam a prioridade apresentação do objeto;*
- *center()*, *width()* e *height()* que retornam o valor do ponto central, da largura e da altura do objeto, respectivamente.

A composição de objetos introduz alguns métodos adicionais usados no controle das relações de inclusão entre os objetos. São eles: *parent()*, *add_member(obj)*, *remove_member(obj)* e *destroy()*.

A modelagem de cenários de animação é feita através da criação de regras do tipo evento-condição-ação. O evento descreve o momento em que a ação de animação ocorre. Eventos podem ser eventos de temporização, de usuário, "intraobjetos" ou "interobjetos". Adicionalmente, é definido o evento START, que representa o início da apresentação. Eventos podem ser compostos através dos operadores de composição apresentados anteriormente, como *SEQ*, *ANY* e *TIMES*, por exemplo.

A condição de uma regra é uma expressão lógica que envolve condições simples (sobre tempo, estados de objetos de animação, relações espaciais entre objetos). A ação é uma seqüência de ações simples, que correspondem a chamadas de métodos de objetos de animação (criação de objeto, controle do estado de transformações, operações de composição, etc., vistos anteriormente). Ações simples são separadas por retardos, que podem ter o valor 0 (zero), que determinam intervalos de tempo entre as suas execuções.

2.5 FOLHAS DE ESTILO

O uso de estilos na definição de documentos estruturados é bastante difundido na literatura. Várias linguagens de definição de folhas de estilos têm sido criadas nos últimos anos [CSS1, XSL, Marden98 e DSSSL].

Folhas de estilo são usadas para definir o formato de apresentação de um documento. Elas são definidas separadamente da estrutura de documentos e determinam diversas propriedades de apresentação como os tipos de fonte, cores e bordas a serem utilizados.

Adicionalmente, folhas de estilo podem definir o formato espacial de apresentação dos componentes de um documento. O W3C tem desenvolvido nos últimos anos algumas linguagens de definição de estilos para documentos XML [XML] (em especial, HTML [HTML4]). Uma delas, *Cascading Style Sheets* (CSS), é definida atualmente em três versões, CSS nível 1 (CSS1) [CSS1], CSS nível 2 (CSS2) [CSS2] e CSS nível 3 (CSS3) [CSS3], que é uma versão modularizada da linguagem que ainda se encontra em desenvolvimento. XSL (*eXtensible Style Sheets*) [XSL] é outra linguagem desenvolvida pelo W3C. Ela á baseada na linguagem DSSL (*Document Style Semantics and Specification Language*) [DSSSL].

Tanto CSS como XSL definem estilos para documentos XML. Além disso, ambas as linguagens possuem modelos para a definição da estrutura de apresentação dos objetos de documentos.

XSL utiliza um modelo de áreas (*area model*) para a definição da apresentação do conteúdo de documentos XML. São usadas áreas retangulares, que podem ser do tipo **bloco** ou ***inline***. O modelo permite uma definição hierárquica de áreas (organizada através de uma árvore de áreas), possibilitando que áreas possuam "áreas filhas". Cada área retangular é subdivida em três retângulos menores, como ilustra a Figura 2.15: um retângulo de conteúdo, onde pode ser apresentado algum conteúdo e/ou definidas outras áreas (filhas); e retângulos de preenchimento e de borda que regulam a justaposição de áreas em um mesmo nível ("irmãs").

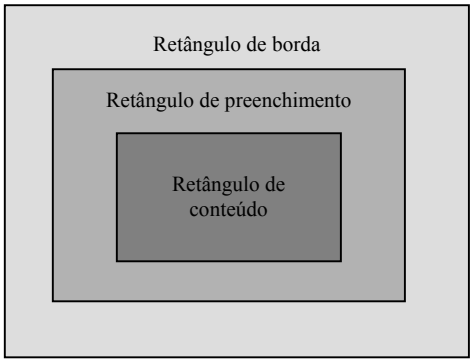


Figura 2.15 – Modelo de áreas usado por XSL.

As áreas definidas em uma folha de estilo XSL podem ser empilhadas. Esse empilhamento é feito através da definição de restrições de empilhamento. A Figura 2.16 apresenta um exemplo de empilhamento de blocos. É definido um bloco *P*, com dois blocos filhos, *A* e *B*. *B*,

por sua vez, possui, também, um bloco filho, *C*. Por simplicidade, considera-se que todas as áreas não possuem preenchimento ou borda (ver Figura 2.15). São definidas três restrições que especificam espaços entre os blocos. A primeira delas define que o bloco *A* deve ter um espaço de 3 pontos após a sua área. Antes do bloco *B* também é definida uma restrição que determina um espaço de 1 ponto. A última das restrições é definida entre o bloco *C* e bloco que o contém, *B*; é definido um espaço de 2 pontos antes de *B*.

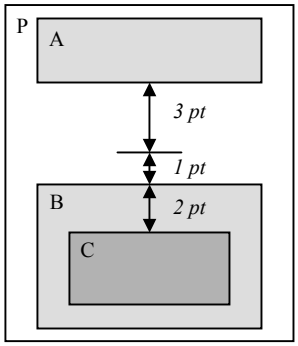


Figura 2.16 – Empilhamento de áreas do tipo bloco.

A Figura 2.17 apresenta os possíveis espaçamentos que podem ser usados na definição de restrições de empilhamento. Essa figura apresenta os retângulos que compõem uma área retangular (já ilustrados na Figura 2.15) e um novo retângulo, determinado pelos espaçamentos aplicados à área original.

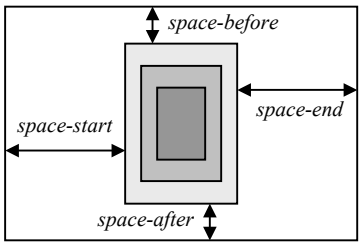


Figura 2.17 – Espaçamento utilizado por áreas.

Deve-se observar que as restrições de empilhamento podem ser definidas tanto entre áreas irmãs como entre uma área e sua área mãe.

O modelo usado por CSS define **caixas**, semelhantes às áreas, para a apresentação, através de um modelo de caixas (*box model*). Uma caixa possui uma estrutura de retângulos semelhante a definida pelo modelo de áreas (ver Figura 2.18).

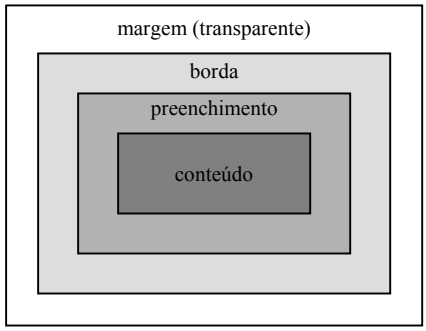


Figura 2.18 – Retângulos de uma caixa.

CSS também define um **modelo de formatação visual** [**CSS2**] para as caixas de uma estrutura de apresentação. Esse modelo considera a existência de um *viewport*, que é uma janela ou algum recurso usado para a apresentação de um documento. Ele determina que a apresentação deve se adaptar quando as dimensões do *viewport* usado forem alteradas. Alternativamente, o modelo permite que esquemas de rolagem sejam usados quando a apresentação se estender às dimensões do *viewport* utilizado.

O modelo de formatação define três esquemas de posicionamento para as caixas de uma estrutura. São eles:

- **fluxo normal** (*normal flow*) onde as caixas podem ser definidas como do tipo bloco ou *inline*. As caixas são espaçadas umas das outras através de suas margens, como acontece com as restrições de empilhamento do modelo de áreas de XSL.

  Nesse esquema, caixas podem ser **posicionadas relativamente** à caixa que as contém através das propriedades *bottom*, *left*, *top*, *right*, e que definem defasagens com relação à "caixa mãe", de forma semelhante à definição de regiões feita em SMIL (ver Seção 2.1).

- **flutuante**, onde uma caixa pode ser apresentada fora da estrutura de caixas da apresentação, flutuando sobre elas. Da mesma forma que uma caixa comum, uma caixa flutuante respeita as definições de margem especificadas.

- **posicionamento absoluto**. Uma caixa pode ser posicionada de forma absoluta com relação à caixa que a contém. Quando isso acontece, ela é excluída do fluxo normal da caixa mãe. Isso dá uma liberdade maior na definição de caixas em uma estrutura, não necessitando que elas sejam empilhadas, como acontece no fluxo normal.

  Naturalmente, caixas podem ser posicionadas de tal forma que sejam gerados conflitos na sua apresentação. O modelo prevê um atributo, de nome z-index, que controla a prioridade

de apresentação de caixas irmãs. Quanto maior o valor desse atributo, maior a prioridade de apresentação do objeto.

[Caloini96] apresenta outra aplicação de estilos na apresentação de documentos hipermídia. É apresentado um ambiente de autoria de documentos hipermídia, com um modelo de documentos que permite a composição de objetos de mídia para formar cenas de apresentação. São usados dois tipos básicos de estilo na apresentação de um documento, *slots* e *templates*. *Slots* definem o estilo de apresentação de objetos de mídia. *Templates* definem estilos de apresentação para cenas completas. Obviamente, quando *templates* forem utilizadas para a definição do estilo de apresentação de uma cena, o usuário deve indicar quais objetos devem ser associados a quais *slots*. Esse é um esquema bastante parecido com a construção de estruturas de apresentação.

2.6 SUMÁRIO

Esse capítulo apresentou vários trabalhos relacionados à sincronização espacial de documentos hipermídia. Na Seção 6.2, esses trabalhos serão comparados ao modelo definido por esta dissertação.

3 ESTRUTURAS DE APRESENTAÇÃO EM DOCUMENTOS HIPERMÍDIA

As características espaciais dos objetos de um documento durante sua apresentação podem ser definidas de formas diferentes. Pode-se defini-las diretamente, através de propriedades do próprio objeto (indicando onde ele deve ser posicionado, quais devem ser suas dimensões com relação ao dispositivo de exibição usado para a sua apresentação, etc.). Outra maneira de fazê-lo é através da associação de documentos a estruturas de apresentação, de forma que os objetos desses documentos herdem as características espaciais definidas pelos componentes das estruturas.

As características espaciais são em geral definidas através de valores absolutos, mas podem ser especificadas através de relações espaciais. Relações espaciais permitem que os valores das propriedades espaciais de objetos sejam definidos com base nos valores de outros objetos.

Este capítulo apresenta um modelo genérico para a especificação de estruturas de apresentação para documentos hipermídia. São definidas relações espaciais que podem ser usadas na definição dessas estruturas. Além disso, são definidas como estruturas de apresentação devem ser associadas aos objetos de um documento, para definir como eles devem ser apresentados.

Este capítulo encontra-se organizado como se segue. A Seção 3.1 apresenta como a estrutura de apresentação de um documento hipermídia pode ser especificada. A Seção 3.2 define como relações espaciais podem auxiliar o processo de autoria de estruturas de apresentação. A Seção 3.3 apresenta uma linguagem declarativa para a edição de estruturas de apresentação. A Seção 3.4 define como os componentes de uma estrutura de apresentação devem ser associados aos objetos de mídia de um documento. A Seção 3.5 apresenta como relações espaciais e espaço-temporais entre objetos de um documento hipermídia podem ser usadas

para facilitar a definição da sincronização espacial e temporal. Finalmente, a Seção 3.6 faz um breve sumário sobre o capítulo.

3.1 DEFINIÇÃO DE UMA ESTRUTURA DE APRESENTAÇÃO

Uma estrutura de apresentação é definida separadamente da estrutura lógica do documento, que pode referenciá-la nas definições espaciais de seus objetos. O modelo espacial aqui apresentado define três tipos de componentes (de apresentação) — dispositivos, janelas e regiões — usados na definição de estruturas de apresentação.

Uma estrutura de apresentação define os dispositivos de exibição a serem usados na apresentação de um documento. Entende-se por dispositivo de exibição qualquer conjunto de recursos de *hardware* ou *software* que possam ser usados para apresentar um objeto em sua mídia de representação. Por exemplo, um dispositivo de apresentação pode ser um terminal de vídeo, um canal de áudio ou uma janela em um ambiente de janelas de um sistema operacional.

Janelas subdividem uma apresentação em seus dispositivos. Uma janela pode conter várias regiões, mas não pode conter outras janelas. Cada janela deve identificar o dispositivo de apresentação a ser utilizado para sua apresentação e de suas regiões.

Regiões definem propriedades espaciais (largura, altura, cor de fundo, etc.) que determinam como os objetos de mídia de um documento devem ser apresentados. Quando uma estrutura de apresentação é associada a um documento, deve-se definir, entre outras coisas, que regiões serão usadas pelos objetos de mídia do documento, para determinar como eles deverão ser apresentados. O controlador/exibidor de um objeto de mídia é associado à janela que contiver a região utilizada.

Além de definir as propriedades de apresentação de objetos de mídia, uma região pode conter outras regiões recursivamente, permitindo uma composição hierárquica de regiões em estruturas de apresentação.

Cada janela de uma estrutura de apresentação define um escopo diferente em uma estrutura de apresentação, de forma que objetos apresentados em regiões de janelas diferentes de um documento não podem relacionar-se espacialmente (ver Seção 3.5). Isso se deve, principalmente, ao fato de que janelas podem ser apresentadas em dispositivos diferentes, não fazendo sentido relacionar espacialmente suas regiões com as de outras janelas.

Ao contrário das regiões, as janelas não carregam informações para a apresentação de objetos de mídia. Além disso, elas não precisam ser referenciadas pelos objetos de um documento, pois cada região da estrutura de apresentação referencia a janela que a contém. Dessa forma, os valores das propriedades espaciais de uma região são sempre relativos à janela que a contém.

Quando apresentadas, tanto janelas como regiões são representadas visualmente como áreas retangulares[6]. A Figura 3.1 ilustra um exemplo de estrutura de apresentação. Nela são especificadas diversas janelas e regiões. Cada uma das janelas da estrutura é associada a um dispositivo de apresentação.

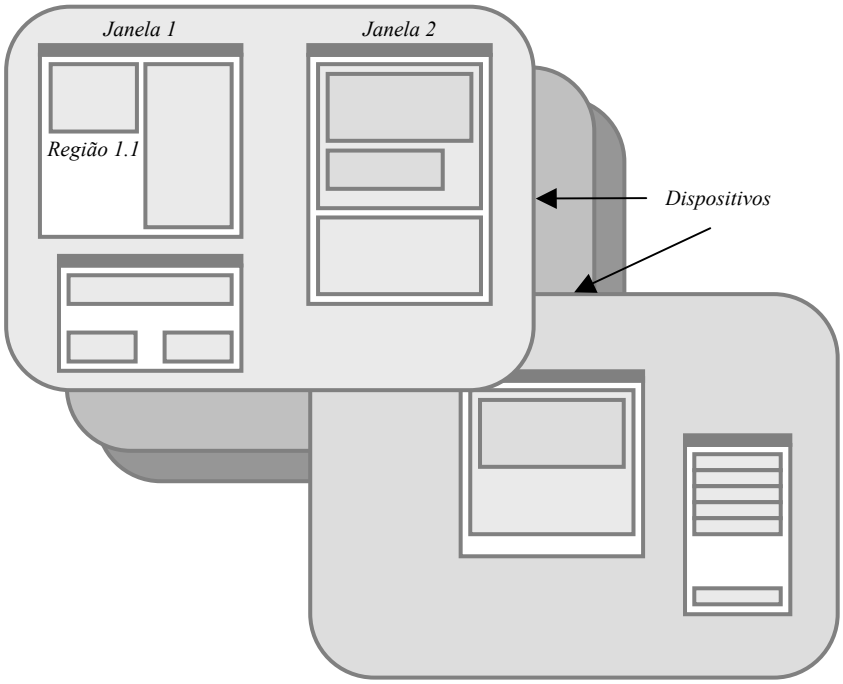


Figura 3.1 – Exemplo de estrutura de apresentação com vários dispositivos de exibição.

Há uma diferença na definição das dimensões de janelas e regiões de uma estrutura de apresentação. Ao contrário das regiões, que definem áreas finitas para a apresentação de objetos, a área de apresentação de uma janela é infinita. As dimensões de uma janela definem apenas uma área visual (*viewport*) dentro de um dispositivo de apresentação. Implementações do modelo devem prover formas de navegação (barras de rolagem, por exemplo) para que as áreas definidas por janelas possam ser totalmente visitadas durante uma apresentação, no caso

[6] Um modelo genérico ideal permitiria que a representação visual de janelas e regiões fosse dependente de implementação. Janelas, assim como regiões, poderiam ser apresentadas nas mais diversas formas geométricas. No entanto, isso provavelmente não seria de grande utilidade. Tendo em vista isso e primando pela simplicidade, o modelo aqui proposto limita a representação espacial de componentes de apresentação à forma retangular.

de regiões internas a elas estarem posicionadas de forma que não possam ser visualizadas por inteiro.

Como forma de ilustrar essa característica, o exemplo da Figura 3.2 apresenta uma janela, que contém três regiões, associada a um dispositivo qualquer. Na situação (a) apenas uma das regiões é apresentada totalmente; as outras não podem ser apresentadas completamente. A situação (b) mostra o que ocorreria caso o conteúdo da janela "rolasse" para a esquerda, apresentando todos os componentes parcialmente. O posicionamento da janela não deve ser alterado durante o processo de rolagem.

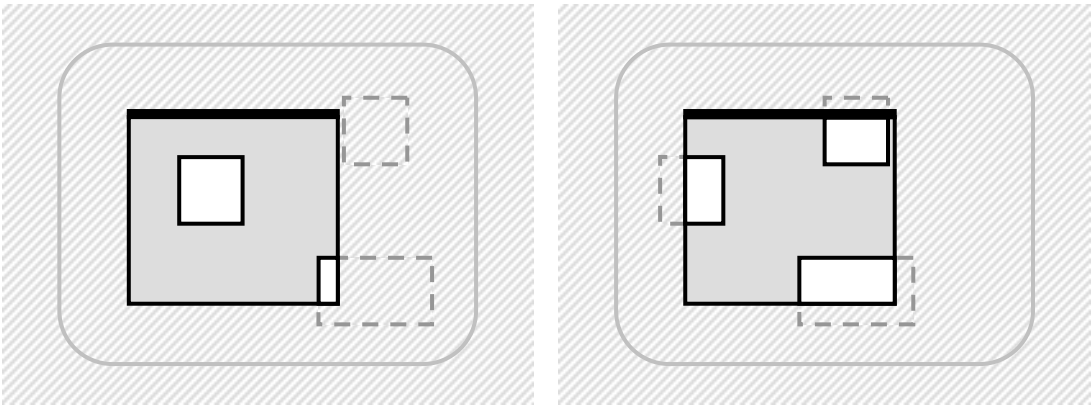


Figura 3.2 – Visualização de regiões internas a uma janela.

Como será visto adiante, tanto janelas como regiões possuem propriedades para a especificação de seu posicionamento e de suas dimensões absolutos. A definição exata das características espaciais desses componentes através de tais propriedades pode se tornar um tanto penosa. Como forma de evitar isso e facilitar a autoria de estruturas de apresentação, o modelo prevê a possibilidade de definição de **relações espaciais entre componentes de apresentação** (ver Seção 3.2), permitindo posicioná-los de forma relativa.

Deve-se salientar que uma estrutura de apresentação, ao ser associada a um documento, apenas determina a configuração espacial inicial dos objetos de mídia desse documento. Qualquer objeto pode ter tanto seu posicionamento, como qualquer uma de suas propriedades espaciais alterada durante a apresentação.

Detalhes a respeito da especificação de uma estrutura de apresentação, através de seus componentes, são apresentados nas subseções a seguir.

3.1.1 Janelas

Janelas subdividem a organização espacial da apresentação de um documento hipermídia. Elas agrupam regiões (ver seção 3.1.2) e podem conter a definição de relações espaciais envolvendo essas regiões. Cada janela deve identificar os dispositivos de exibição/apresentação que deve ser usados na sua apresentação.

Janelas restringem a definição de relações espaciais na especificação da composição espacial de um documento, não permitindo que regiões localizadas em janelas diferentes relacionem-se espacialmente (ver seção 3.2).

A especificação de uma janela contém a definição de uma série de propriedades que determinam:

- como ela deve ser identificada dentro da estrutura de apresentação e os dispositivos onde ela deve ser apresentada;
- seu posicionamento com relação ao dispositivo de apresentação ao qual ela se encontra associada;
- suas dimensões;
- suas características visuais; e
- como deve ser o seu comportamento durante a apresentação do documento.

A Tabela 3.1 lista as propriedades de uma janela.

Tabela 3.1 – Propriedades de uma janela.

backgroundColor[7]	Especifica a cor de fundo da janela.
bottom	Define a distância relativa (em alguma unidade ou percentual) do lado inferior da janela com relação ao mesmo lado da área determinada pelo dispositivo de apresentação ao qual ela se encontra associada (ver Figura 3.3).
close	Define em que situações uma janela deve ter sua exibição encerrada durante a apresentação do documento associado. Os valores possíveis para esse atributo são `whenNotActive` (valor *default*) e `never`. O primeiro define que a janela deixará de ser apresentada sempre que nenhuma de suas regiões estiver apresentando algum objeto de mídia. O segundo define que a janela somente poderá ser fechada a pedido do usuário, leitor do documento.
devices	Indica um conjunto de nomes, definindo os dispositivos de exibição onde a janela deve ser apresentada. Por exemplo, pode-se definir que uma janela seja apresentada simultaneamente nos dispositivos "*video1*" e "*video2*".

[7] O modelo não prevê a definição de figuras como pano de fundo de uma janela. No entanto, o mesmo efeito pode ser conseguido através da associação de um figura a uma região que cubra a área visível de uma janela.

	Essa informação indica ao formatador hipermídia como agrupar as janelas definidas em uma estrutura de apresentação nos dispositivos de exibição disponíveis. Essa indicação não precisa ser necessariamente respeitada pelo formatador, devido à possibilidade dos dispositivos indicados pela janela não estarem disponíveis para a apresentação. Independentemente dessa impossibilidade, o formatador é livre para alterar a associação de janelas a dispositivos, seja por indicação do usuário ou por especificação da plataforma. Caso essa propriedade não seja definida, o comportamento é definido pelo formatador. Por exemplo, pode-se definir que janelas que não definam algum dispositivo de apresentação devam ser exibidas em um dispositivo *default*.
height	Define a altura de uma janela. Seu valor pode ser absoluto — em alguma unidade — ou percentual, relativo à altura da área de apresentação determinada pelo dispositivo associado. O formatador deve tratar qualquer inconsistência na definição dessa propriedade, no caso dela exceder o permitido pelo dispositivo de apresentação.
id*	Define um identificador único para a janela na especificação da composição espacial de um documento.
left	Define a distância relativa (em alguma unidade ou percentual) do lado esquerdo da janela com relação ao mesmo lado da área determinada pelo dispositivo de apresentação a ela associado (ver Figura 3.3).
movable	Define se uma janela pode ter sua posição alterada durante sua apresentação. Pode receber os valores `true` (valor *default*) ou `false`. Esta propriedade impede que os valores das propriedades `bottom`, `left`, `top` e `right` sejam alterados.
open	Define em que condições uma janela deve ter sua exibição iniciada durante a apresentação do documento associado. Seus valores podem ser `always` ou `whenActive` (valor *default*). O primeiro determina que a janela deve ser apresentada sempre, logo que a apresentação do documento se inicie. O segundo define que a janela só deve ser apresentada quando algum objeto de mídia associado a uma das regiões internas a janela tiver sua exibição iniciada.
resizable	Define se as dimensões de uma janela podem ou não ser alteradas durante sua apresentação. Pode receber os valores `true` (valor *default*) ou `false`. Esta propriedade impede a alteração dos valores das propriedades `height` e `width`.
right	Define a distância (em alguma unidade ou percentual) do lado direito da janela com relação ao mesmo lado da área determinada pelo dispositivo de apresentação a ela associado (ver Figura 3.3).
top	Define a distância (em alguma unidade ou percentual) do lado superior da janela com relação ao mesmo lado da área determinada pelo dispositivo de apresentação a ela associado (ver Figura 3.3).
visible	Define se a janela deve estar visível ou não durante sua apresentação. Pode receber os valores `true` (valor *default*) ou `false`.
width	Define a largura de uma janela. Seu valor pode ser absoluto — em alguma unidade — ou percentual, relativo à largura da área de apresentação determinada pelo dispositivo associado. O formatador deve tratar qualquer inconsistência na definição dessa propriedade, no caso dela exceder o permitido pelo dispositivo de apresentação.
name	Define um nome para a janela, não necessariamente único em uma apresentação.
z-index	Define a ordem de prioridade da janela durante a apresentação do documento. Seu valor deve ser um número natural. Este atributo é usado quando duas janelas se sobrepõem durante a apresentação; quanto menor ele for, maior será a prioridade de apresentação da janela.

* Propriedades obrigatórias.

Dentre as propriedades listadas, apenas `id` é obrigatória na especificação de uma janela. Todas as outras são opcionais, no entanto, implementações desse modelo devem definir valores *default* para as propriedades que utilizam valores numéricos, `bottom`, `height`, `left`, `right`, `top`, `width` e `z-index`. Todas as outras propriedades já possuem valores *default* definidos.

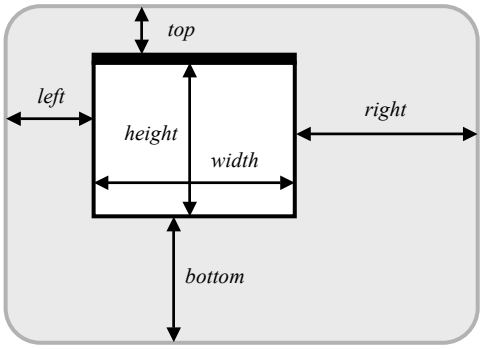


Figura 3.3 – Propriedades de posicionamento e dimensão de uma janela.

A Figura 3.4 apresenta um exemplo de definição de uma estrutura de apresentação composta por duas janelas, cujos identificadores únicos e o dispositivo onde devem ser apresentados encontram-se especificados. Além disso, elas definem algumas propriedades de posicionamento e dimensão, assim como suas prioridades de apresentação, no caso de sobreposição.

Layout

Janela 1

```
id: window1
devices: video
left: 50
top: 50
width: 250
height: 250
z-index: 0
```

Janela 2

```
id: window2
devices: video
left: 150
top: 100
right: 50
height: 250
z-index: 1
```

Figura 3.4 – Exemplo de definição de uma estrutura de apresentação (*layout*) com duas janelas.

É importante observar que as propriedades de posicionamento e dimensão de uma janela podem ser definidas de diversas formas. Por exemplo, na Figura 3.4, a *Janela 1* define as propriedades `left`, `top`, `width` e `height`, enquanto que a *Janela 2* especifica as propriedades `left`, `top`, `right` e `height`.

A Figura 3.5 apresenta uma representação gráfica do exemplo da Figura 3.4.

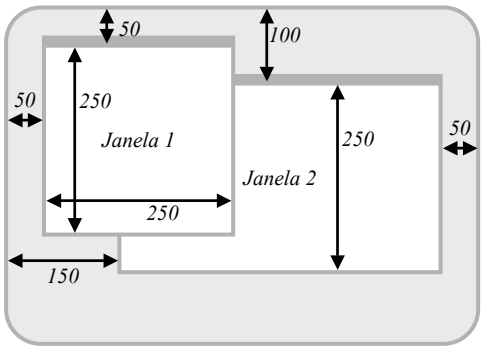


Figura 3.5 – Representação visual do exemplo definido na Figura 3.4.

A definição do posicionamento e das dimensões de uma janela através de suas propriedades deve ser feita segundo uma regra muito simples, somente duas propriedades devem ser definidas em cada eixo. Por exemplo, no eixo horizontal, as seguintes combinações são permitidas:

- `left` e `width`; ou
- `left` e `right`; ou
- `right` e `width`.

No eixo vertical ocorre algo semelhante. As combinações permitidas são as seguintes:

- `top` e `height`; ou
- `top` e `bottom`; ou
- `bottom` e `height`.

A definição de mais de duas propriedades para algum eixo deve ser considerada uma inconsistência na especificação de uma estrutura de apresentação.

Observa-se que a definição das propriedades de posicionamento e de dimensão pode gerar uma área de interseção entre as janelas. Esse conflito é contornado durante a apresentação usando-se o valor da propriedade `z-index` dos componentes envolvidos. No exemplo da Figura 3.5, a *Janela 1* tem `z-index` menor que o da *Janela 2* e, por isso, é apresentada sobre ela.

Como citado anteriormente, relações espaciais podem ser usadas para auxiliar a especificação do posicionamento e das dimensões de janelas (e regiões). Caso uma janela participe de uma relação espacial, algumas de suas propriedades não precisam ser definidas. No entanto, o modelo permite que uma janela tenha suas propriedades de posicionamento (e/ou de

dimensão) definidas mesmo que ela participe de alguma relação espacial. Nesse caso, os valores das propriedades devem ter precedência, em detrimento da relação espacial.

3.1.2 Regiões

Regiões são usadas para organizar a apresentação de um documento em cada uma de suas janelas. Como citado na Seção 3.1.1, uma janela pode conter uma ou mais regiões.

Um objeto de mídia de um documento hipermídia pode ser associado a alguma região da estrutura disponível para que possa ser apresentado apropriadamente. Regiões definem um conjunto de propriedades usadas pelos objetos de mídia para determinar como eles devem ser apresentados espacialmente.

Deve ser permitido que mais de um objeto de mídia seja associado a uma mesma região, podendo compartilhá-la em um mesmo instante de tempo. É responsabilidade do formatador hipermídia, usado na apresentação dos documentos, resolver conflitos quando eles ocorrerem.

O uso de regiões também fornece um mecanismo para a organização hierárquica de uma apresentação; uma região, além de poder ter um objeto de mídia associado, pode conter outras regiões. Assim, pode-se compor estruturas espaciais bastante complexas para a apresentação de documentos hipermídia. Esse modelo define uma única restrição com relação ao uso de regiões: uma região não pode estar contida diretamente em mais de um componente de apresentação.

Cada região define um conjunto de propriedades que, assim como as propriedades de uma janela, permitem especificar:

- sua identificação em uma estrutura de apresentação;
- seu posicionamento com relação ao componente de apresentação que a contém;
- suas dimensões;
- sua aparência visual; e
- como deve ser seu comportamento durante a apresentação do documento.

Regiões podem ser usadas para a apresentação de objetos de mídia auditiva. A apresentação de objetos de áudio é um caso especial. Eles, ao contrário de objetos representados em outras mídias, podem ter sua apresentação associada a duas regiões: uma delas contida em uma janela associada a um dispositivo que permite a exibição de objetos de mídia auditiva, onde seria apresentado o conteúdo do objeto; e uma outra, associada a uma janela apresentada em

um dispositivo de apresentação de objetos de mídia visual, onde poderiam ser apresentados os controles de apresentação do objeto (com funções para iniciar, interromper, ou terminar sua apresentação, por exemplo). A definição de propriedades de posicionamento e dimensão não é necessária para a primeira das regiões. Deve-se, somente, definir o volume de áudio a ser usado durante a exibição do objeto de áudio através da propriedade `soundLevel`, apresentada na Tabela 3.2.

A Tabela 3.2 traz uma lista das propriedades de uma região. Várias delas, já especificadas para janelas, foram apenas adaptadas.

Tabela 3.2 – Propriedades de uma região.

backgroundColor	Especifica a cor de fundo da região.
bottom	Define a distância relativa (percentual ou em alguma unidade) do lado inferior da região com relação ao mesmo lado da área visual determinada pelo componente que a contém (ver Figura 3.6).
fit[8]	Define o comportamento de uma região quando as dimensões do objeto de mídia associado não coincidirem com as da região. Pode ter os seguintes valores: • `fill`. Redimensiona o objeto de mídia para que suas dimensões coincidam com as da região; • `hidden`. Valor *default*. Se a altura/largura do objeto de mídia for menor que a altura/largura da região, o objeto é desenhado a partir do canto superior esquerdo da região e o espaço restante é preenchido com sua cor de fundo; se alguma das dimensões do objeto for maior que a da região, a diferença é simplesmente cortada, não sendo exibida; • `meet`. O objeto é desenhado a partir do canto superior esquerdo da região. Ele é redimensionado conservando a proporção entre suas dimensões até que alguma delas seja igual à correspondente da região. O espaço restante é preenchido com a cor de fundo; • `scroll`. Especifica que a região deve prover algum mecanismo de rolagem, caso o tamanho do objeto exceda o da região em alguma de suas dimensões. • `slice`. Comportamento semelhante ao definido pelo valor `meet`. A diferença é que o excedente do objeto de mídia é cortado e que nenhuma cor de fundo é mostrada.
height	Define a altura de uma região. Seu valor pode ser absoluto — em alguma unidade — ou percentual, relativo à altura da área visual determinada pelo componente que a contém. O formatador deve tratar qualquer inconsistência na definição dessa propriedade, no caso dela exceder o permitido pelo dispositivo de apresentação.
id*	Define um identificador único para a região na especificação da composição espacial de um documento.
left	Define a distância relativa (percentual ou em alguma unidade) do lado esquerdo da região com relação ao mesmo lado da área visual determinada pelo componente que a contém (ver Figura 3.6).
movable	Define se uma região pode ter sua posição alterada durante sua apresentação. Pode receber os valores `true` (valor *default*) ou `false`. Esta propriedade impede que os

[8] Exemplos de uso dessa propriedade podem ser encontrados nos módulos de *layout* da linguagem SMIL [SMIL2].

	valores das propriedades `bottom`, `left`, `top` e `right` sejam alterados.
`name`	Define um nome para a região. Não necessariamente é único em uma apresentação.
`resizable`	Define se as dimensões de uma região podem ou não ser alteradas durante sua apresentação. Pode receber os valores `true` (valor *default*) ou `false`. Esta propriedade impede a alteração dos valores das propriedades `height` e `width`.
`right`	Define a distância relativa (percentual ou em alguma unidade) do lado direito da região com relação ao mesmo lado da área visual determinada pelo componente que a contém (ver Figura 3.6).
`showBackground`	Especifica em que situações a cor de fundo da região deve ser apresentada quando não houver nenhum objeto de mídia associado. Pode assumir os valores `always` (valor *default*) ou `whenActive`. O primeiro especifica que a cor de fundo deve ser mostrada sempre; o segundo define que a cor de fundo não deve ser mostrada quando a região não estiver ativa, isto é, quando não tiver nenhum objeto de mídia associado. No caso da região conter outras, ela é considerada ativa sempre que pelo menos uma de suas regiões internas assim estiver.
`soundLevel`	Especifica o volume relativo com o qual o objeto de mídia auditiva associado deve ser apresentado. Pode receber valores percentuais, relativos ao volume especificado na plataforma de apresentação.
`top`	Define a distância relativa (percentual ou em alguma unidade) do lado superior da região com relação ao mesmo lado da área visual determinada pelo componente que a contém (ver Figura 3.6).
`visible`	Define se a região deve estar visível ou não durante sua apresentação. Pode receber os valores `true` (valor *default*) ou `false`.
`width`	Define a largura de uma região. Seu valor pode ser absoluto — em alguma unidade — ou percentual, relativo à largura da área visual determinada pelo componente que a contém. O formatador deve tratar qualquer inconsistência na definição dessa propriedade, no caso dela exceder o permitido pelo dispositivo de apresentação.
`z-index`	Define a ordem de prioridade da região durante a apresentação do documento. Seu valor deve ser um número natural. Este atributo é usado quando duas regiões se sobrepõem durante a apresentação; quanto menor ele for, maior será a prioridade de apresentação da região.

* Propriedade obrigatória.

Apenas a propriedade `id` é obrigatória na definição de uma região. Assim como na especificação de janelas, implementações do modelo devem definir valores *default* para as propriedades que utilizam valores numéricos. Todas as outras propriedades já possuem valores *default* definidos.

A regra de uso das propriedades de posicionamento (`bottom`, `left`, `right` e `top`) e de dimensão (`height` e `width`) em regiões é a mesma definida para janelas. As combinações possíveis são as mesmas mostradas na Seção 3.1.1. A Figura 3.6 mostra como essas propriedades são aplicadas na definição de uma região com relação ao componente que a contém.

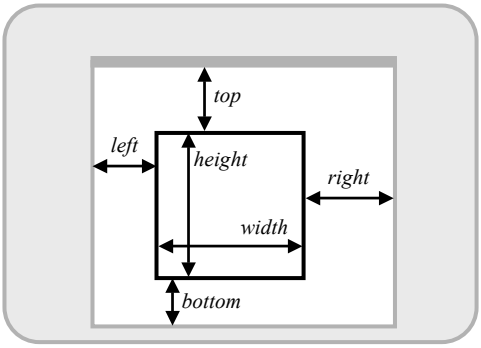


Figura 3.6 – Propriedades de posicionamento e dimensão de uma região.

A Figura 3.7 apresenta um exemplo de definição de uma estrutura de apresentação simples, composta de apenas uma janela e uma região. A Figura 3.8 apresenta a sua possível representação visual.

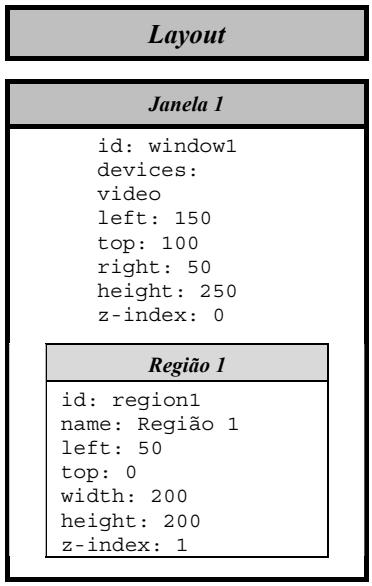


Figura 3.7 – Exemplo de definição de uma estrutura de apresentação (*layout*) com uma janela e uma região.

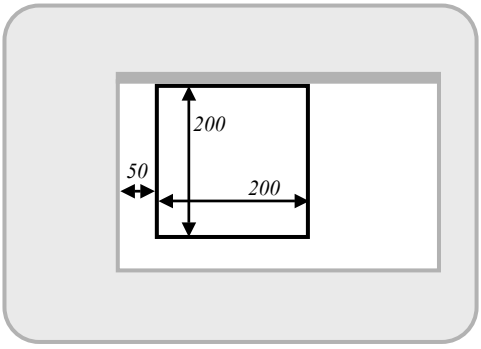


Figura 3.8 – Representação visual do exemplo definido na Figura 3.7.

Além da propriedade `soundLevel`, citada anteriormente, a definição de regiões inclui outras propriedades não existentes em janelas. São elas: `fit`, `name` e `showBackground` (ver Tabela 3.2).

Uma região pode conter outras regiões. A Figura 3.9 mostra um exemplo disso. Nela, é definida uma estrutura com uma só janela, *Janela 1*. Essa janela contém duas regiões, *Região 1* e *Região 2*. A primeira região, por sua vez, contém uma terceira, *Região 3*.

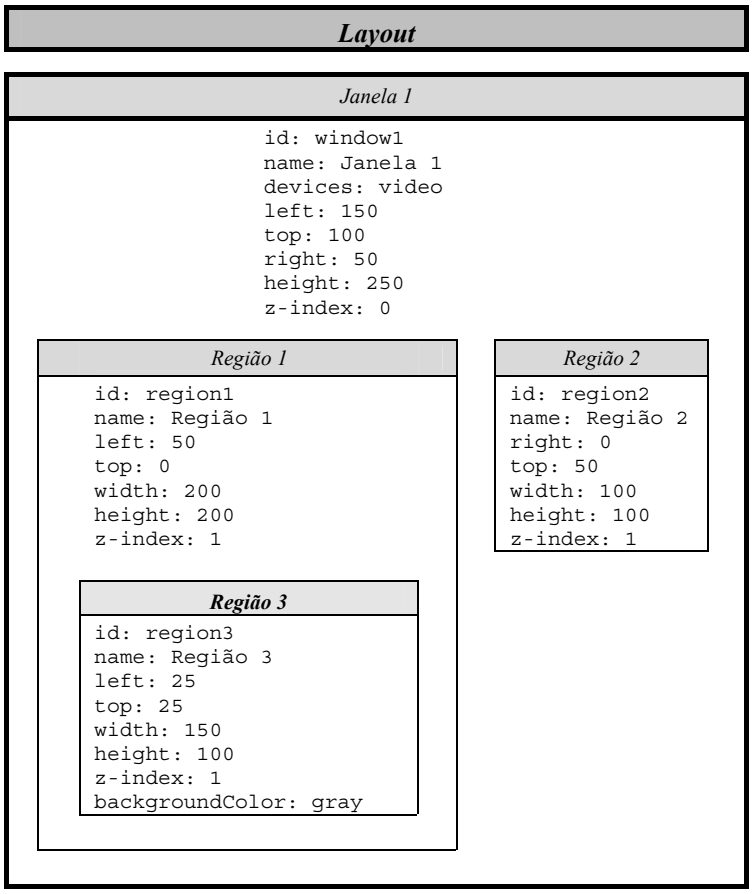


Figura 3.9 – Exemplo de definição de uma região com regiões internas.

A Figura 3.10 ilustra esse exemplo.

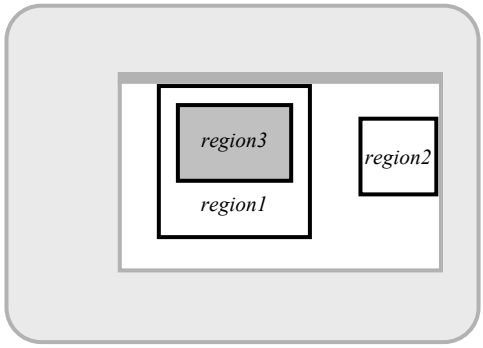


Figura 3.10 – Representação visual do exemplo definido na Figura 3.9.

Um fato que deve ser ressaltado na especificação de uma estrutura de apresentação composta de janelas e regiões é a existência de escopos distintos para a definição da prioridade de apresentação desses componentes, expressa através da propriedade `z-index`.

Cada componente contêiner[9] (janela ou região) define um novo escopo. A definição da estrutura de apresentação cria inicialmente um escopo, utilizado na definição de janelas. A resolução da prioridade de apresentação é feita para cada escopo quando necessário. A Figura 3.11 mostra a definição de uma estrutura de apresentação com algumas janelas e várias regiões. A Figura 3.12 ilustra esse exemplo, levando em consideração a resolução das prioridades de apresentação de cada um dos componentes.

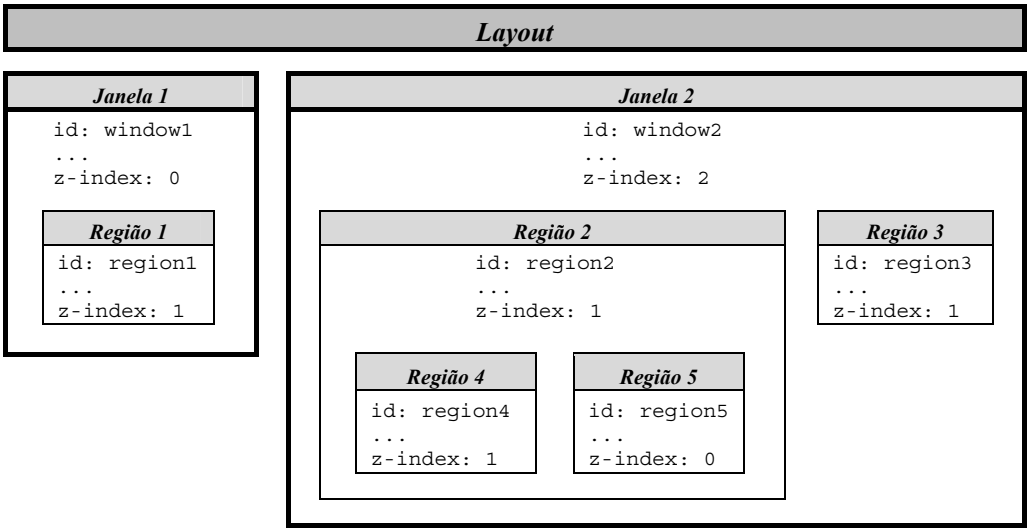


Figura 3.11 – Uso da propriedade `z-index` na definição de janelas e regiões.

Nesse exemplo são definidas duas janelas, *window1* e *window2*. A primeira delas contém uma região, *region1*; a segunda, duas regiões, *region2* e *region3*. A região *region2*, por sua vez, contém duas outras regiões, *region4* e *region5*.

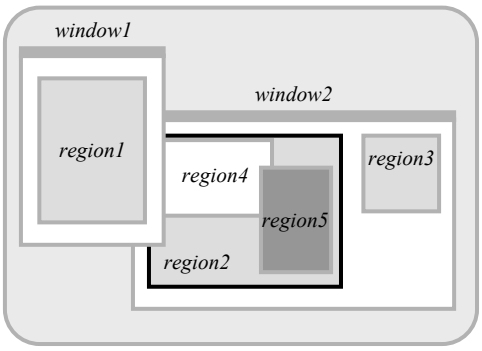


Figura 3.12 – Representação gráfica do exemplo mostrado na Figura 3.11.

A resolução da prioridade de apresentação é feita da seguinte forma:

[9] Um componente contêiner corresponde a uma janela ou região que contenha sub-regiões.

- no escopo definido pela apresentação, a janela *window1* tem precedência sobre a janela *window2*, devido ao valor de sua propriedade `z-index` ser menor. Observa-se na Figura 3.12 que todo o conteúdo da primeira janela é exibido com prioridade maior, sobrepondo-se à outra janela;
- no escopo definido pela janela *window1* não há necessidade de resolução, por haver apenas uma região definida;
- no escopo definido pela janela *window2* também não há necessidade de resolução, por não haver interseção entre as áreas das duas regiões definidas. Caso essa interseção existisse, devido ao fato de ambas as regiões possuírem o mesmo valor para a propriedade `z-index`, a resolução da prioridade ficaria a cargo da implementação do modelo. Por exemplo, poderia ser decidido dar maior prioridade para o componente definido primeiro;
- no escopo definido pela região *region2* a resolução é bastante simples. Por ter o valor da propriedade `z-index` menor que o da região *region4*, a região *region5* é apresentada com precedência maior no caso de conflito.

A definição de hierarquias de regiões, internas às janelas de uma estrutura de apresentação, serve apenas para auxiliar o processo de criação da estrutura de apresentação de documentos hipermídia. Como forma de simplificar e tornar uniforme o tratamento de relações espaciais definidas em uma estrutura de apresentação (ver Seção 3.2), as relações de inclusão (que podem ser consideradas relações espaciais) definidas entre regiões e outros componentes da estrutura, quando incorporadas a um modelo de documentos hipermídia, devem ser simplificadas, de forma que as regiões passem a estar contidas diretamente na janela onde são definidas. Obviamente, as relações de inclusão especificadas na definição da estrutura apresentação devem permanecer válidas de alguma forma. Isso pode ser feito através da alteração dos valores das propriedades `z-index` dos componentes envolvidos.

Por exemplo, considerando a estrutura definida na Figura 3.11 e ilustrada na Figura 3.12, a hierarquia de regiões definidas na janela *window2* seria convertida de forma que todas as regiões estivessem contidas diretamente na janela e suas propriedades `z-index` seriam alteradas, passando a ter os seguintes valores: *region2*, 2; *region3*, 2; *region4*, 1; e *region5*, 0.

O posicionamento e as dimensões de uma região podem ser definidos de forma relativa através de relações espaciais, não sendo necessário atribuir diretamente valores a quaisquer das propriedades de posicionamento/dimensão. A Seção 3.2, a seguir, apresenta como essas relações podem ser especificadas.

3.2 RELAÇÕES ESPACIAIS EM UMA ESTRUTURA DE APRESENTAÇÃO

Uma estrutura de apresentação é criada pela composição de janelas e regiões. Como visto nas Seções 3.1.1 e 3.1.2, tanto janelas, como regiões possuem várias propriedades que determinam como elas devem ser posicionadas e quais devem ser suas dimensões, no início de uma apresentação. Como também mencionado anteriormente, a definição dessas propriedades pode ser um tanto penosa. Dessa forma, para facilitar o processo de autoria de estruturas de apresentação, é sugerido aqui o uso de relações espaciais de alto nível na definição do posicionamento e/ou das dimensões dos componentes de uma estrutura de apresentação.

De uma maneira geral, uma relação espacial corresponde a uma relação entre propriedades espaciais dos componentes envolvidos. Uma relação entre propriedades é representada por uma equação, onde os valores das propriedades são associados para representar a situação desejada.

Algumas relações espaciais são sugeridas nas subseções a seguir. Elas são subdivididas em dois tipos:

- relações entre janelas ou regiões "irmãs". Só pode envolver janelas ou regiões contidas em um mesmo componente;
- relações entre regiões e o componente que as contém. Relacionam regiões "irmãs" com a janela ou região que as contém diretamente.

Essas relações são posteriormente definidas em uma linguagem para a definição de estruturas de apresentação, discutida na Seção 3.3.

3.2.1 Relações espaciais entre janelas ou regiões "irmãs"

Relações espaciais facilitam o posicionamento ou o dimensionamento de janelas e de regiões contidas em uma mesma janela ou região de uma estrutura de apresentação. Janelas podem ser posicionadas com relação a outras janelas ou ter suas dimensões determinadas pelas dimensões de outras janelas. O mesmo pode ser aplicado a regiões "irmãs" de uma estrutura de apresentação.

Cada um dos tipos de relações espaciais apresentados nesta seção considera uma equação do tipo:

$$A_1 \cdot a_1 = B_1 \cdot b_1 + B_2 \cdot b_2 + c \quad (1),$$

onde:

- A_1 é uma propriedade espacial de um objeto A;
- B_1 e B_2 são propriedades espaciais de um objeto B;
- a_1, b_1 e b_2 são fatores que são multiplicados aos valores das propriedades para representar a relação; e
- c é um valor constante.

Os tipos de relações sugeridos pelo modelo são os seguintes:

- relações de defasagem;
- relações de alinhamento;
- relações de espaçamento;
- relações de dimensionamento; e
- relações de centralização.

Com exceção das relações de centralização, as relações apresentadas são definidas considerando as propriedades espaciais `bottom`, `height`, `left`, `right`, `top` e `width` dos componentes envolvidos.

As relações de defasagem, de alinhamento e de espaçamento determinam equações entre os valores das bordas dos componentes envolvidos.

São quatro as relações de defasagem: defasagem à esquerda, à direita, em relação ao topo, e em relação à base. Todas elas representam diferenças de posicionamento entre dois componentes com relação à mesma borda. Elas são apresentadas na Tabela 3.3.

Tabela 3.3 – Relações de defasagem.

Defasagem à esquerda	Define que a borda da esquerda de um componente B deve estar defasada (de um valor definido) da mesma borda de um componente A. Especifica uma relação entre os valores das propriedades `left` de cada um dos componentes. Considerando que A_1 e B_1 se refiram às propriedades `left` dos componentes envolvidos na relação, ela pode ser definida da seguinte forma: $A_1 + x = B_1$ O elemento x corresponde à defasagem que deve ser aplicada ao componente B. Esta relação é apresentada na Figura 3.13 (a).
Defasagem à direita	Relação definida de forma idêntica à relação de defasagem à esquerda, substituindo-se a propriedade `left` por `right`. Esta relação é apresentada na Figura 3.13 (b).

Defasagem em relação ao topo	Relação definida de forma idêntica às anteriores, usando-se a propriedade `top`. É apresentada na Figura 3.13 (c).
Defasagem em relação à base	Relação definida de forma idêntica às anteriores, usando-se a propriedades `bottom`. É apresentada na Figura 3.13 (d).

Na Figura 3.13 cada uma das relações de defasagem é ilustrada.

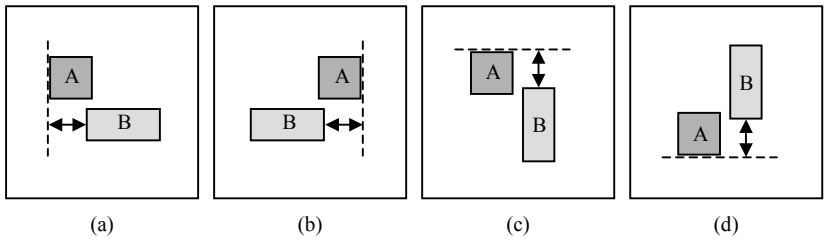


Figura 3.13 – Relações de defasagem.

As relações de alinhamento são casos especiais das relações de defasagem. Elas são relações de defasagem com valor de defasagem igual a zero. Assim como as relações de defasagem, existem quatro relações de alinhamento: alinhamento à esquerda, à direita, ao topo e à base. Todas elas são definidas como uma simplificação das relações de defasagem, atribuindo-se zero ao valor *x* das equações. As relações de alinhamento são ilustradas na Figura 3.14.

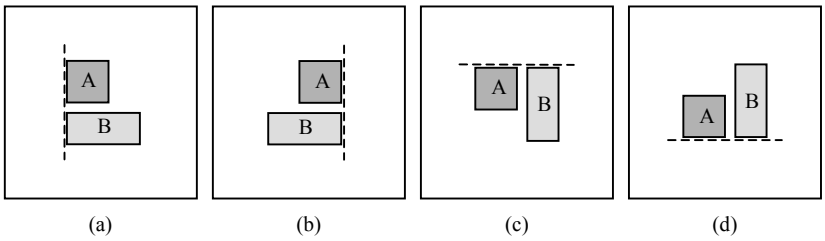


Figura 3.14 – Relações de alinhamento.

Uma relação de espaçamento define o quanto dois objetos devem se manter distanciados. Ao contrário das relações de defasagem e de alinhamento, as relações de espaçamento envolvem bordas opostas dos componentes envolvidos. Há quatro tipos de relações de espaçamento: espaçamento à esquerda, à direita, em relação ao topo e em relação à base. A construção delas é semelhante a das relações de defasagem, como mostrado na Tabela 3.4.

Tabela 3.4 – Relações de espaçamento.

Espaçamento à esquerda	Define que a borda da direita de um componente *B* deve estar distanciada de um valor definido da borda esquerda de um componente *A*. Especifica uma relação entre os valores da propriedade `left` (A_1) de *A* e as propriedades `left` (B_1) e `width` (B_2) de *B*. Ela pode ser definida da seguinte forma:

	$$A_1 = B_1 + B_2 + x$$ O elemento *x* corresponde ao espaçamento que deve ser aplicado ao elemento *B*. Esta relação é apresentada na Figura 3.15 (a).
Espaçamento à direita	Define que a borda da esquerda de um componente *B* deve estar distanciada de um valor definido da borda direita de um componente *A*. É especificada de forma idêntica ao espaçamento à esquerda, trocando-se apenas as propriedades `left` dos componentes por `right`. Esta relação é apresentada na Figura 3.15 (b).
Espaçamento em relação ao topo	Define que a borda inferior de um componente *B* deve estar distanciada de um valor definido da borda superior de um componente *A*. É definida da mesma forma que o espaçamento à esquerda, usando-se as propriedades `top` e `height` ao invés de `left` e `width`. Esta relação é apresentada na Figura 3.15 (c).
Espaçamento em relação à base	Define que a borda superior de um componente *B* deve estar distanciada de um valor definido da borda inferior de um componente *A*. É definida da mesma forma que o espaçamento à direita, usando-se as propriedades `bottom` e `height` ao invés de `left` e `width`. Esse tipo de relação de espaçamento é apresentado na Figura 3.15 (d).

As relações de espaçamento são ilustradas na Figura 3.15.

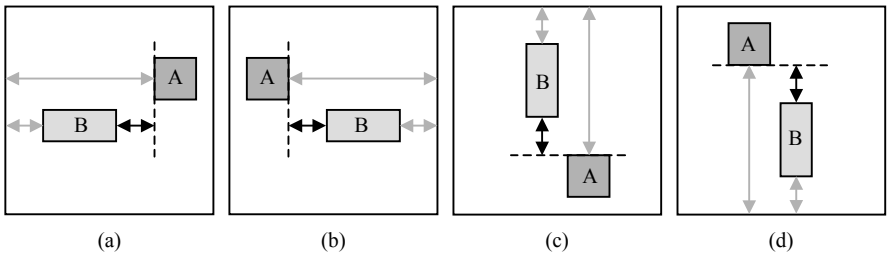


Figura 3.15 – Relações de espaçamento.

As relações de dimensionamento definem relações de proporção entre as propriedades de dimensão dos componentes envolvidos. Há apenas duas relações de dimensionamento: dimensionamento vertical e dimensionamento horizontal. Elas são apresentadas na Tabela 3.5.

Tabela 3.5 – Relações de dimensionamento.

Dimensionamento horizontal	Define que a largura do componente *B* deve ser proporcional a de *A*. Especifica a seguinte relação: $$B_1 = A_1 \cdot a_1,$$ onde A_1 e B_1 correspondem aos valores das propriedades `width` de *A* e *B* e a_1 define a proporção entre a largura de *B* com relação a *A*.
Dimensionamento vertical	Define que a altura de um componente *B* deve ser proporcional a de outro componente, *A*. Sua especificação é semelhante a do dimensionamento horizontal, substituindo-se a propriedade `width` por `height`.

As relações de centralização determinam que os objetos envolvidos devem ser mantidos centralizados com relação a um de seus dois eixos durante a apresentação. Há somente dois tipos de relações de centralização: centralização horizontal e centralização vertical. Elas são apresentadas na Tabela 3.6. Por simplicidade, a definição dessas relações considera a

existência de propriedades que definem as coordenadas do ponto central de uma janela ou região, não previstas pelo modelo.

Tabela 3.6 – Relações de centralização.

Centralização horizontal	Define que dois componentes, *A* e *B*, devem ser mantidos centralizados horizontalmente. Especifica a seguinte relação: $$A_1 = B_1 ,$$ onde A_1 e B_1 correspondem, respectivamente, aos valores das abscissas do ponto central dos componentes envolvidos. Esse tipo de relação é apresentado na Figura 3.16 (a).
Centralização vertical	Define que dois componentes, *A* e *B*, devem ser mantidos centralizados verticalmente. Especifica a mesma equação que a relação de centralização horizontal, referindo-se às ordenadas do ponto central dos componentes envolvidas. Esse tipo de relação é apresentado na Figura 3.16 (b).

A Figura 3.16 ilustra as relações de centralização.

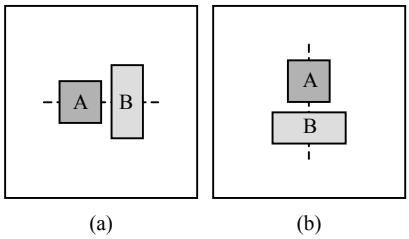


Figura 3.16 – Relações de centralização.

O posicionamento de uma região com relação à janela ou região que a contém também pode ser definido através de relações espaciais. Essas relações são apresentadas na próxima seção.

3.2.2 Relações espaciais entre regiões e os componentes que as contém

Também são sugeridas algumas relações espaciais para auxiliar o posicionamento de regiões internamente às janelas ou regiões que as contêm. Essas relações diferem um pouco das apresentadas na Seção 3.2.1, pois os valores das propriedades de uma região, quando contida em outro componente, referem-se ao espaço definido por ele.

As relações sugeridas resumem-se às relações de defasagem, de alinhamento, de dimensionamento e de centralização. Elas são apresentadas na Tabela 3.7.

Tabela 3.7 – Relações entre regiões e os componentes que as contém em uma estrutura de apresentação.

Defasagem à esquerda	Define que a borda da esquerda de uma região *B* deve estar defasada (de um valor definido) da mesma borda do componente que a contém, *A*. Especifica a seguinte equação: $$B_1 = x$$ onde B_1 corresponde ao valor da propriedade `left` de *B* e *x*, à defasagem que deve ser

	aplicada ao elemento *B*. Esta relação é apresentada na Figura 3.17 (b).
Defasagem à direita	É definida de forma idêntica à relação de defasagem à esquerda, usando-se a propriedade `right`, ao invés de `left`. Esta relação é apresentada na Figura 3.17 (b).
Defasagem em relação ao topo	Relação definida de forma idêntica às outras relações de defasagem, usando-se a propriedade `top`. É apresentada na Figura 3.17 (c).
Defasagem em relação à base	Relação definida de forma idêntica às outras relações de defasagem, usando-se a propriedade `bottom`. É apresentada na Figura 3.17 (d).
Centralização horizontal	Define que uma região *A* deve estar centralizada horizontalmente com relação ao componente que a contém, *B*. Da mesma forma que as relações de centralização apresentadas na Seção 3.2.1, utiliza-se do valor da abscissa do ponto central da região envolvida. Especifica a seguinte relação: $$A_1 = B_1 \cdot \frac{1}{2},$$ onde A_1 corresponde ao valor da abscissa do ponto central da região *A* e B_1, ao valor da altura do componente *B*. Esse tipo de relação é apresentado na Figura 3.19 (a).
Centralização vertical	É definida da mesma forma que a relação anterior. Relaciona o valor da ordenada do ponto central da região *A* com a largura do componente *B*, representado pela propriedade `height`. Esse tipo de relação é apresentado na Figura 3.19 (b).

As relações de alinhamento, como já observado, são casos particulares das relações de defasagem. Sua definição acompanha as modificações mostradas na Figura 3.18. As relações de dimensionamento são definidas como apresentado na Tabela 3.5, com a restrição de que as dimensões de uma região não podem extrapolar as dimensões da janela ou região que a contém.

As relações permitidas entre regiões e os componentes que as contém em uma estrutura de apresentação são ilustradas nas figuras a seguir.

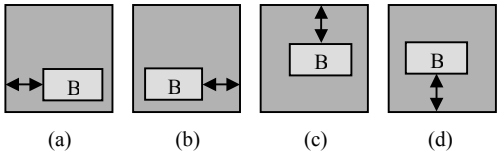


Figura 3.17 – Relações de defasagem entre regiões e o componente que as contêm.

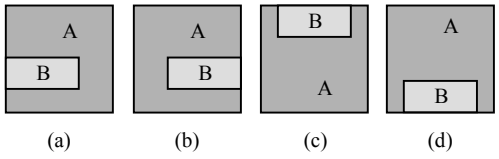


Figura 3.18 – Relações de alinhamento entre regiões e o componente que as contêm.

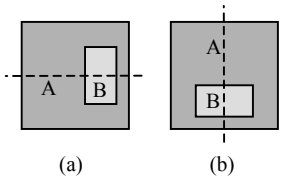


Figura 3.19 – Relações de centralização entre regiões e o componente que as contém.

Regiões não podem estar distanciadas dos componentes que as contém, por isso, relações de espaçamento não podem ser definidas.

A seção a seguir apresenta uma linguagem descritiva para a criação de estruturas de apresentação. Nela são apresentados elementos que permitem a definição de janelas, regiões e relações espaciais. Alguns exemplos de criação de relações espaciais são apresentados nessa seção.

3.3 UMA LINGUAGEM PARA A DEFINIÇÃO DE ESTRUTURAS DE APRESENTAÇÃO

Linguagens declarativas [SMIL1, SMIL2, Jourdan97, Vazirginannis00 e Antonacci00] têm sido bastante usadas para a definição da estrutura lógica de documentos hipermídia. Essas linguagens também definem formas de especificação da estrutura de apresentação de seus documentos.

Esta seção apresenta uma linguagem declarativa para a definição de estruturas de apresentação segundo o modelo espacial definido nesta dissertação. Essa linguagem é definida através da meta-linguagem XML [XML]. Ela pode ser facilmente aplicada às linguagens SMIL e NCL. Sua especificação é apresentada em [Moura01c].

A definição da composição espacial de um documento é feita através da construção de uma estrutura de apresentação composta por janelas e regiões, conforme apresentado. Na Figura 3.20 é mostrada a estrutura de um documento XML, supondo-se que o corpo do documento seja definido como conteúdo do elemento `body` e que o cabeçalho contenha as definições da estrutura de apresentação a ser usada.

```
<head>
   <layout>
      <!-- Definição da estrutura de apresentação do documento. -->
   </layout>
</head>
<body>
   <!-- Definição da estrutura lógica do documento e de seus componentes. -->
<body>
```

Figura 3.20 – Definição da estrutura de apresentação de um documento de forma descritiva.

A linguagem é apresentada nas próximas subseções. Nelas são apresentados os elementos que a compõem e são mostrados alguns exemplos de uso.

3.3.1 Elementos da linguagem

A linguagem de definição da composição espacial de um documento define um elemento para cada entidade do modelo espacial já descrito. São definidos os elementos `window`, `region` e `relation`, correspondentes às entidades *janela*, *região* e *relação espacial*, respectivamente.

Além desses elementos, são definidos os elementos `layout`, que demarca a definição da estrutura de apresentação, e `regPoint` (ver Seção 3.3.2), que define pontos de registro, semelhantes, mas mais poderosos, aos existentes na versão 2.0 da linguagem SMIL.

Os elementos da linguagem são apresentados nas subseções a seguir.

3.3.1.1 Elemento `layout`

O elemento `layout` é um elemento não-vazio que demarca a especificação da estrutura de apresentação de um documento. Ele possui os seguintes atributos:

- `id` que define um identificador para a estrutura de apresentação; e
- `type`, que define a linguagem utilizada na especificação da estrutura. Caso a estrutura seja definida de acordo com o modelo espacial aqui proposto, este atributo deve ter o valor "*ncm-basic-layout*".

Um documento só pode definir um elemento `layout`, indicando que somente uma estrutura de apresentação pode ser usada. Todos os componentes da estrutura de apresentação devem ser especificados internamente ao elemento `layout`. Eles são definidos a seguir.

3.3.1.2 Elemento `window`

O elemento `window` define uma janela de uma estrutura de apresentação. Ele é um elemento não-vazio que contém atributos correspondentes a cada uma das propriedades da entidade janela do modelo espacial (ver Seção 3.1.1 para uma descrição dessas propriedades).

Uma janela deve ser especificada obrigatoriamente dentro do elemento `layout`. Na Figura 3.21 é exemplificada a definição de uma janela através do elemento `window`.

```
<layout>
   <window id ="window1" devices="video" left="10" top="10" width="100" height="200">
      <!-- Definição da estrutura da janela. -->
   </window>
</layout>
```

Figura 3.21 – Definição de uma janela através do elemento `window`.

Nesse exemplo, é definida uma janela com o identificador *window1*. Essa janela e todas aquelas que possuírem o atributo `devices` com o valor *"video"* devem ser preferencialmente apresentadas no mesmo dispositivo. A janela contém também a definição de sua posição (atributos `left` e `top`) e tamanho (atributos `width` e `height`) iniciais.

Uma janela pode ter o valor de seus atributos de posição e/ou dimensionamento definidos por relações espaciais. Essas relações devem ser definidas dentro do elemento `layout`, no mesmo nível das definições de janelas, como ilustrado na Figura 3.22.

Detalhes a respeito da definição de relações espaciais serão apresentados posteriormente, na definição do elemento `relation`.

```
<layout>
   <window id ="window1" devices="video">
   </window>
   <window id ="window2" devices="video">
   </window>
   <!-- Definição de relações espaciais entre janelas. -->
</layout>
```

Figura 3.22 – Definição de duas janelas.

Uma janela pode conter a definição de várias regiões, mas não pode conter a definição de outras janelas. A definição de regiões é apresentada na seção a seguir.

3.3.1.3 Elemento `region`

O elemento `region` é usado na definição de regiões. Ele corresponde à entidade região do modelo espacial. Seus atributos são iguais às propriedades definidas para essa entidade (ver Seção 3.1.2).

Uma região deve ser definida obrigatoriamente dentro de uma janela ou de outra região. A Figura 3.23 define duas regiões dentro de uma janela.

```
<layout>
   <window id ="window1" width="100" height="100">
      <region id ="region1" left="10" top="10" width="50" height="50"/>
      <region id ="region2" left="10" top="70" width="70" height="25"/>
   </window>
</layout>
```

Figura 3.23 – Definição de uma janela contendo duas regiões.

Nota-se que a janela definida nesse exemplo não possui atributos de posicionamento. Seu posicionamento inicial deve ser definido pelo formatador. As duas regiões, ao contrário, têm seu posicionamento relativo (à janela) bem definido. A representação visual desse exemplo é mostrada na Figura 3.24.

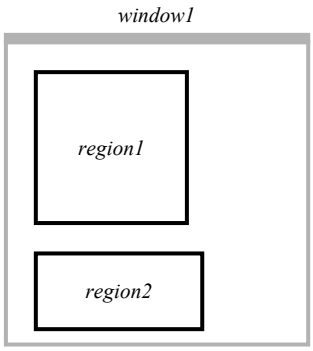


Figura 3.24 – Uma janela com duas regiões, conforme definido no exemplo da Figura 3.23.

Como dito na Seção 3.1.2, uma região pode conter outras regiões, o que proporciona um grau maior de estruturação à apresentação de um documento. Assim, além de existir o elemento `region` vazio, como mostrado na Figura 3.23, a linguagem também prevê um elemento não-vazio, de mesmo nome. Esse elemento só pode conter a definição de outras regiões e de relações espaciais entre elas.

Na Figura 3.25 é mostrada a definição de regiões hierárquicas. A estrutura de apresentação é constituída de uma janela. Nela são definidas duas regiões. A primeira delas, *region1*, é

composta de outras duas, *region11* e *region12*, ambas vazias. A segunda, *region2*, é uma região vazia. Nota-se que com exceção de *region1* todas as outras regiões são definidas através de um elemento `region` vazio.

```
<layout>
   <window id ="window1" width="400" height="250">
      <region id ="region1" left="20" top="20" width="200" height="200">
         <region id ="region11" left="20" top="20" width="150" height="80"/>
         <region id ="region12" left="10" top="120" width="100" height="50"/>
      </region>
      <region id ="region2" right="10" top="20" width="150" height="200"/>
   </window>
</layout>
```

Figura 3.25 – Definição de regiões hierárquicas.

O exemplo da Figura 3.25 é ilustrado na Figura 3.26.

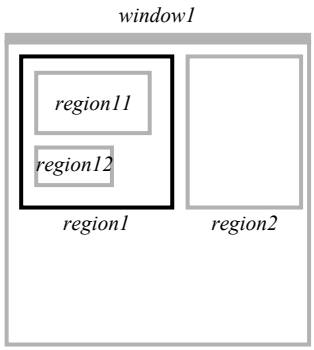


Figura 3.26 – Exemplo da Figura 3.25 representado graficamente.

Assim como ocorre com janelas de uma estrutura de apresentação, regiões podem ser relacionadas espacialmente. Relações espaciais entre regiões devem ser definidas através do elemento `relation` dentro do elemento (`window` ou `region`) que contém diretamente as regiões relacionadas. Também é permitido criar relações entre regiões e os componentes que as contêm diretamente, com o propósito de auxiliar o seu posicionamento. A definição desse tipo de relação também é feita dentro do componente que contém as regiões. Detalhes a respeito do estabelecimento de relações espaciais são apresentados na próxima seção.

3.3.1.4 Elemento `relation`

Uma relação espacial é definida através do elemento `relation`. Ele é um elemento não-vazio que deve conter referências aos componentes envolvidos e deve indicar o tipo de relação espacial a ser representado.

Como apresentado anteriormente, pode-se definir relações espaciais entre janelas, entre regiões, ou entre janelas e seus componentes. Dessa forma, o elemento `relation` pode estar contido nos elementos `layout`, `window` ou `region`.

Elementos `relation` devem ser definidos no mesmo nível dos componentes envolvidos. Relações entre janelas devem ser definidas dentro do elemento `layout` e relações entre regiões devem ser definidas dentro do elemento `window` ou `region` que as contiver. A única exceção se faz às relações entre regiões e o componente que as contém, onde a relação é definida dentro do componente contêiner, no mesmo nível das regiões. Essa forma de construção simplifica a definição de relações e facilita o trabalho de *parsers* da linguagem. Por exemplo, na Figura 3.25, caso deseje-se criar uma relação espacial envolvendo as regiões *region11* e *region12*, ela deve ser especificada dentro da região *region1*. Posteriormente serão mostrados alguns exemplos de definições de relações espaciais.

A Tabela 3.8 apresenta os atributos definidos para o elemento `relation`.

Tabela 3.8 – Propriedades do elemento `relation`.

`container`	Define que se trata de uma relação entre regiões e o componente que os contém. Não tem valor e não é definido por *default*.
`type`*	Define o tipo da relação espacial. Seu valor pode ser qualquer um dos relacionados abaixo (correspondentes aos tipos de relações espaciais definidos na Seção 3.2): • `identLeft`. Define uma relação de defasagem à esquerda; • `identRight`. Define uma relação de defasagem à direita; • `identTop`. Define uma relação de defasagem em relação ao topo; • `identBottom`. Define uma relação de defasagem em relação à base; • `spaceLeft`. Define uma relação de espaçamento à esquerda; • `spaceRight`. Define uma relação de espaçamento à direita; • `spaceTop`. Define uma relação de espaçamento em relação à base; • `spaceBottom`. Define uma relação de espaçamento em relação ao topo; • `centerHorizontal`. Define uma relação de centralização horizontal; • `centerVertical`. Define uma relação de centralização vertical; • `scaleHorizontal`. Define uma relação de dimensionamento horizontal; • `scaleVertical`. Define uma relação de dimensionamento vertical. Os componentes de uma relação são definidos através do elemento `component`, a ser apresentado na próxima subseção. As relações de alinhamento são definidas como relações de defasagem com valor zero. Esse valor é especificado através do atributo `value` dos componentes da relação.

* Propriedade obrigatória.

A definição de relações deve obedecer as restrições apresentadas na Seção 3.5. A próxima subseção explica como definir os componentes envolvidos em uma relação.

Componentes de uma relação

O elemento `component` é um elemento vazio que define um componente participante de uma relação espacial. Ele só pode ser definido dentro de um elemento `relation` e somente deve referenciar componentes que estejam definidos no mesmo nível da relação ou aquele que a contiver.

Cada elemento `component` define um componente da relação. As relações de defasagem, espaçamento e dimensionamento necessitam da definição de um **componente base**, com relação ao qual o posicionamento ou as dimensões dos outros devem ser determinadas. Esse componente, quando a relação envolver somente regiões "irmãs", deve ser aquele definido primeiro. Quando o elemento `relation` definir uma relação do componente contêiner (atributo `container`) e suas regiões, ele deverá ser considerado o componente base.

Os atributos do elemento `component` são relacionados na Tabela 3.9.

Tabela 3.9 – Propriedades do elemento `component`.

`id`*	Referência ao elemento que representa a entidade participante da relação. Esse identificador deve corresponder ao identificador de uma janela ou região que esteja definida no mesmo nível da relação ou ao do elemento "pai" da relação.
`value`	Especifica um valor associado à relação. Esse valor deve ser um número real ou uma porcentagem. Por *default*, esse valor é zero. Seu significado é dependente do tipo da relação: • no caso das relações de espaçamento (`spaceLeft`, `spaceRight`, `spaceTop` e `spaceBottom`), o valor desse atributo define o espaço que deve ser mantido entre o componente base e o outro integrante da relação; • no caso das relações de alinhamento (`identLeft`, `identRight`, `identTop` e `identBottom`) esse valor deve ser zero. Se a relação tiver algum valor diferente de zero associado, esse valor representará a defasagem entre o componente base e o outro integrante da relação; • relações de centralização desconsideram o valor desse atributo, caso seja configurado; • nas relações de dimensionamento, o valor é usado para dimensionar o tamanho dos componentes secundários com relação ao componente base.

* Propriedade obrigatória.

O exemplo da Figura 3.27 define uma relação espacial entre duas janelas. Nele, é definido um elemento `relation` dentro do elemento `layout`. O tipo da relação é especificado através do atributo `type`. Seu valor é `identLeft`. Os componentes envolvidos são identificados através dos atributos `id` dos elementos `component`.

```
<layout>
   <window id =“window1” width=“400” height=“250” left=“10” top=“10”>
   </window>
   <window id =“window2” width=“100” height=“100” top=“500”>
   </window>
   <relation type=“identLeft”>
      <component id=“window1”>
      <component id=“window2”>
   </relation>
</layout>
```

Figura 3.27 – Definição de uma relação de alinhamento entre duas janelas.

Na Figura 3.28 é mostrado um caso mais complexo, onde regiões em vários níveis da hierarquia são associadas através de relações espaciais diferentes. O exemplo define uma estrutura de apresentação composta de uma janela, *window1*. Ela contém duas regiões, *region1* e *region2*. A primeira contém duas sub-regiões, *region11* e *region12*, e a segunda, três, *region21*, *region22* e *region23*.

As regiões *region1* e *region2* são especificadas com um alinhamento à esquerda, dentro da definição da janela *window1*. A segunda relação espacial definida é uma relação de centralização horizontal entre duas sub-regiões de *region2*. A relação é especificada dentro dessa região.

```
<layout>
   <window id =“window1” width=“600” height=“250” left=“10” top=“10”>
      <region id =“region1” left=“20” top=“20” width=“200” height=“200”>
         <region id =“region11” left=“20” top=“20” width=“150” height=“80”/>
         <region id =“region12” left=“10” top=“120” width=“100” height=“50”/>
      </region>
      <region id =“region2” right=“10” top=“20” width=“150”
         <region id =“region21” left=“20” width=“150” height=“80”/>
         <region id =“region22” left=“110” width=“100” height=“50”/>
         <region id =“region23” left=“220” width=“100” height=“50”/>
         <relation type=“centerHorizontal”>
            <component id=“region21”>
            <component id=“region22”>
         </relation>
      </region>
      <relation type=“identLeft”>
         <component id=“region1”>
         <component id=“region2”>
      </relation>
   </window>
</layout>
```

Figura 3.28 – Definição de relações espaciais em níveis diferentes.

3.3.2 Pontos de registro

Janelas e regiões de uma estrutura de apresentação podem ter seu posicionamento definido através de suas propriedades de posicionamento ou por relações espaciais das quais elas participem. A linguagem define uma forma adicional de relacionamento que permite o posicionamento de janelas e regiões através da definição de pontos de registro. A definição de ponto de registro no escopo dessa linguagem estende a definição original usada pela linguagem SMIL (ver Seção 2.1). Aqui, esses pontos poderão ser usados tanto para o posicionamento de janelas e regiões como de objeto dentro de suas regiões.

Pontos de registro podem ser classificados como relações espaciais entre componentes contêiner (ou dispositivos) e janelas ou regiões. Eles são definidos através do elemento `regPoint`. Cada ponto de registro identifica um ponto, que pode ser usado por uma região ou janela para definir seu posicionamento com relação ao componente que a contém (no caso de janelas, a superfície determinada pelo dispositivo de apresentação). O elemento `regPoint` é um elemento vazio que deve ser definido dentro do elemento `layout` e pode

ser utilizado por qualquer um dos componentes de uma estrutura ou por objetos do documento. A Tabela 3.10 apresenta os atributos do elemento `regPoint`.

Tabela 3.10 – Propriedades do elemento `regPoint`.

`bottom`	Define a distância relativa do ponto de registro à borda inferior do componente contêiner associado. Seu valor pode ser percentual, com relação à altura do componente contêiner do componente que estiver usando o ponto de registro, ou ser especificado em alguma unidade.
`id`*	Especifica um identificador para o ponto de registro.
`left`	Define a distância relativa do ponto de registro à borda da esquerda do componente contêiner associado. Seu valor pode ser percentual, com relação à largura do componente contêiner do componente que estiver usando o ponto de registro, ou ser especificado em alguma unidade.
`regAlign`	Define o tipo de alinhamento a ser aplicado. O componente que utilizar o ponto de registro será alinhado de acordo com o tipo especificado. A propriedade pode ter um dos seguintes valores: • `topLeft` (valor *default*). Alinha o ponto superior esquerdo do componente ao ponto de registro especificado. • `topMid`. Alinha o ponto superior central do componente ao ponto de registro especificado. • `topRight`. Alinha o ponto superior direito do componente ao ponto de registro especificado. • `midRight`. Alinha o ponto central direito do componente ao ponto de registro especificado. • `bottomRight`. Alinha o ponto inferior direito do componente ao ponto de registro especificado. • `bottomMid`. Alinha o ponto inferior central do componente ao ponto de registro especificado. • `bottomLeft`. Alinha o ponto inferior esquerdo do componente ao ponto de registro especificado. • `midLeft`. Alinha o ponto central esquerdo do componente ao ponto de registro especificado. • `center`. Alinha o ponto central do componente ao ponto de registro especificado.
`right`	Define a distância relativa do ponto de registro à borda da direita do componente contêiner associado. Seu valor pode ser percentual, com relação à largura do componente contêiner do componente que estiver usando o ponto de registro, ou ser especificado em alguma unidade.
`top`	Define a distância relativa do ponto de registro à borda superior do componente contêiner associado. Seu valor pode ser percentual, com relação à altura do componente contêiner do componente que estiver usando o ponto de registro, ou ser especificado em alguma unidade.

* Propriedade obrigatória.

A linguagem já define alguns pontos *default*. São eles: `topLeft`, `topMid`, `topRight`, `midRight`, `bottomRight`, `bottomMid`, `bottomLeft`, `midLeft` e `center`. Cada um deles define um tipo predeterminado de posicionamento. A Figura 3.29 ilustra o posicionamento de uma região (área hachurada) em uma janela com o auxílio de um dos pontos de registro *default*. A região é posicionada da seguinte forma:

a) *topLeft*. A região é posicionada de forma que seu canto superior esquerdo seja posicionado sobre o canto superior esquerdo da janela.

b) *topMid*. A região é posicionada de forma que o ponto central de sua borda superior seja posicionado sobre o mesmo ponto da janela.

c) *topRight*. A região é posicionada de forma que seu canto superior direito seja posicionado sobre o canto superior direito da janela.

d) *midLeft*. A região é posicionada de forma que o ponto central de sua borda da esquerda seja posicionado sobre o mesmo ponto da janela.

e) *center*. A região é centralizada com relação à janela.

f) *midRight*. A região é posicionada de forma que o ponto central de sua borda da direita seja posicionado sobre o mesmo ponto da janela.

g) *bottomLeft*. A região é posicionada de forma que seu canto inferior esquerdo seja posicionado sobre o canto inferior esquerdo da janela.

h) *bottomMid*. A região é posicionada de forma que o ponto central de sua borda inferior seja posicionado sobre o mesmo ponto da janela.

i) *bottomRight*. A região é posicionada de forma que seu canto superior direito seja posicionado sobre o canto superior direito da janela.

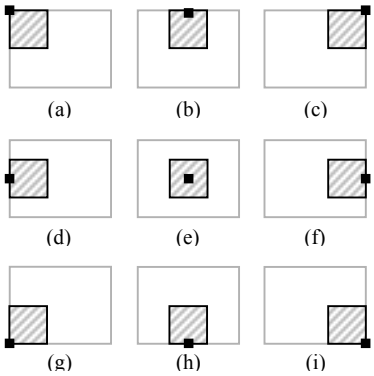


Figura 3.29 – Ilustração do posicionamento aplicado pelos pontos de registro *default*.

Para que os componentes de uma estrutura de apresentação possam usar pontos de registro para se posicionar, dois novos atributos devem ser definidos para os elementos `window` e `region`. Eles são apresentados na Tabela 3.11.

Tabela 3.11 – Propriedades adicionais dos elementos `window` e `region`.

`regPoint`	Especifica o ponto de registro usado pela região. Deve conter o identificador de um ponto de registro presente na definição da estrutura de apresentação ou de um dos pontos *default*.
`regAlign`	Usado para redefinir o tipo de alinhamento utilizado. Uma região pode usar um ponto de registro e sobrecarregar o seu algoritmo de alinhamento. Os valores permitidos para esta propriedade são os mesmos definidos para a propriedade de mesmo nome do elemento `regPoint`, mostrados na Figura 2.3.

A Figura 3.30 apresenta um exemplo de uso de um ponto de registro *default* no posicionamento de uma região em uma janela. A região region1 usa o ponto de registro *center*, ficando centralizada com relação à janela *window1*, como ilustra a Figura 3.31.

```
<layout>
   <window id =“window1” width=“300” height=“200” left=“10” top=“10”>
      <region id =“region1” regPoint=“center” width=“300” height=“200”>
   </window>
</layout>
```

Figura 3.30 – Posicionamento de uma região com o auxílio do ponto de registro *default* `center`.

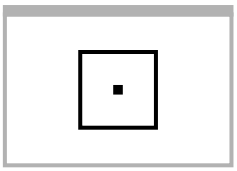


Figura 3.31 – Ilustração do exemplo da Figura 3.30.

A Figura 3.32 apresenta um exemplo de posicionamento de uma região com o auxílio de um ponto de registro. Nele, um ponto *p1* é definido na estrutura de apresentação do documento, que é usado pela região *region1* para posicionar-se sobre a janela *window1*. Esse exemplo é ilustrado na Figura 3.33.

```
<layout>
   <regPoint id=“p1” top=“50” left=“50” regAlign=“topLeft”/>
   <window id =“window1” width=“300” height=“200” left=“10” top=“10”>
      <region id =“region1” regPoint=“p1” width=“300” height=“200”>
   </window>
</layout>
```

Figura 3.32 – Posicionamento de uma região com o auxílio de um ponto de registro.

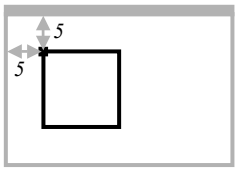


Figura 3.33 – Ilustração do exemplo da Figura 3.32.

Para que pontos de registro possam ser utilizados no posicionamento de objetos de mídia em suas regiões, os atributos mostrados na Tabela 3.11 devem ser acrescentados aos elementos que representarem esses objetos na linguagem de definição da estrutura lógica do documento.

A próxima seção apresenta como as regiões de uma estrutura de apresentação podem ser usadas pelos objetos de mídia de um documento na definição de suas características espaciais.

3.4 ASSOCIAÇÃO DE OBJETOS DE MÍDIA A REGIÕES

Como citado na Seção 3.1, os objetos de mídia de um documento hipermídia podem ser associados a regiões para que sejam apresentados de forma apropriada. Obviamente, caso um objeto especifique todas suas características espaciais (posicionamento, dimensionamento, cor, etc.) não se faz necessário o uso de regiões.

A forma de associação de um objeto de mídia a uma região é dependente do modelo de documentos hipermídia utilizado. No entanto, ela deve ser feita de forma que mais de um objeto possa estar associado a uma mesma região em um mesmo instante de tempo.

Uma região define um conjunto de propriedades que devem ser herdadas por um objeto de mídia de um documento quando este for associado a ela. Dessa forma, vários objetos que se refiram a uma mesma região espacial, apenas conterão propriedades com valores iniciais iguais aos definidos pela região, sendo-lhes permitido alterar esses valores independentemente.

As definições espaciais de uma estrutura de apresentação determinam somente a disposição inicial dos objetos que venham a utilizá-la. Qualquer uma das propriedades espaciais herdadas por um objeto de mídia pode ser modificada durante sua apresentação.

Da mesma forma que em uma estrutura de apresentação, o posicionamento e o dimensionamento de objetos de mídia de um documento pode ser determinado através de relações espaciais. O conceito de relações espaciais entre objetos estende a definição de

relações entre componentes de uma estrutura de apresentação. A próxima seção apresenta como podem ser criadas relações espaciais entre objetos de mídia de um documento.

3.5 RELAÇÕES ENTRE OBJETOS

Em determinadas situações, pode ser desejável que as características espaciais de objetos de um documento sejam mantidas durante a apresentação, prevenindo que ações do usuário ou outras definições existentes na estrutura lógica do documento possam alterar deliberadamente a sua configuração espacial. Por exemplo, pode ser desejável definir que dois objetos devam ser mantidos alinhados durante toda a apresentação. Faz-se necessária, portanto, a definição de relações espaciais entre objetos de um documento.

Relações espaciais representam restrições entre objetos, relacionando os valores de propriedades de objetos de um documento. Essas relações são definidas na própria estrutura de um documento hipermídia entre os objetos de mídia que devem ter sua apresentação controlada. Elas são semelhantes as relações espaciais entre componentes de uma estrutura de apresentação, definidas na Seção 3.2.

Além de tudo isso, relações espaciais simplificam a definição de posicionamento relativo entre objetos de um documento, facilitando o processo de autoria.

A subseção a seguir mostra a estrutura genérica de uma relação espacial e alguns tipos de relações sugeridos. A subseção seguinte apresenta algumas relações espaço-temporais, relações que envolvem tanto características espaciais dos objetos de um documento, como a sincronização temporal de sua apresentação.

3.5.1 Relações espaciais

As relações entre objetos sugeridas aqui são as mesmas apresentadas na Seção 3.2.1, levando-se em conta que os objetos de mídia de um documento são associados a regiões para serem exibidos. Relações de defasagem, de alinhamento, de espaçamento, de dimensionamento e de centralização são definidas da mesma forma. Logicamente, os objetos relacionados devem estar sendo apresentados em uma mesma janela.

Relações entre objetos, ao contrário das relações entre componentes de apresentação, devem ser mantidas durante a apresentação.

A restrição definida por uma relação espacial deve ser mantida válida desde que os objetos envolvidos estejam sendo apresentados. Por exemplo, caso seja criada uma relação de alinhamento à esquerda entre dois objetos *A* e *B*, no momento em ambos os objetos estiverem sendo apresentados, eles devem ter suas bordas da esquerda alinhadas.

A forma de manutenção de uma relação entre atributos pode ser feita de formas diferentes durante a apresentação dos objetos envolvidos, por isso, deve ser especificada explicitamente. Por exemplo, considere uma relação de alinhamento à esquerda entre dois objetos mostrada na Figura 3.34 (a). Pode haver três situações distintas para a manutenção dessa relação. A primeira delas define que independente do objeto que seja movido (ver Figura 3.34 (b) e (c)), a relação deve ser mantida, corrigindo o posicionamento do outro objeto envolvido. A segunda e a terceira situações são idênticas. Em ambas, é definido o sentido em que a relação deve ser mantida. Caso o sentido da relação seja definido como $A \rightarrow B$ e o objeto *A* seja movido, *B* deve ter sua posição corrigida para que a relação seja mantida, como mostrado na Figura 3.34 (b); no caso de *B* ter sua posição modificada, nada deve acontecer ao posicionamento de *A* (ver Figura 3.34 (e)). *A* é considerado o **componente de referência** para a manutenção da relação. No caso do sentido da relação ser especificado como $B \rightarrow A$, deve ocorrer o inverso, como ilustrado na Figura 3.34 (c) e na Figura 3.34 (d). *B* é considerado o componente de referência.

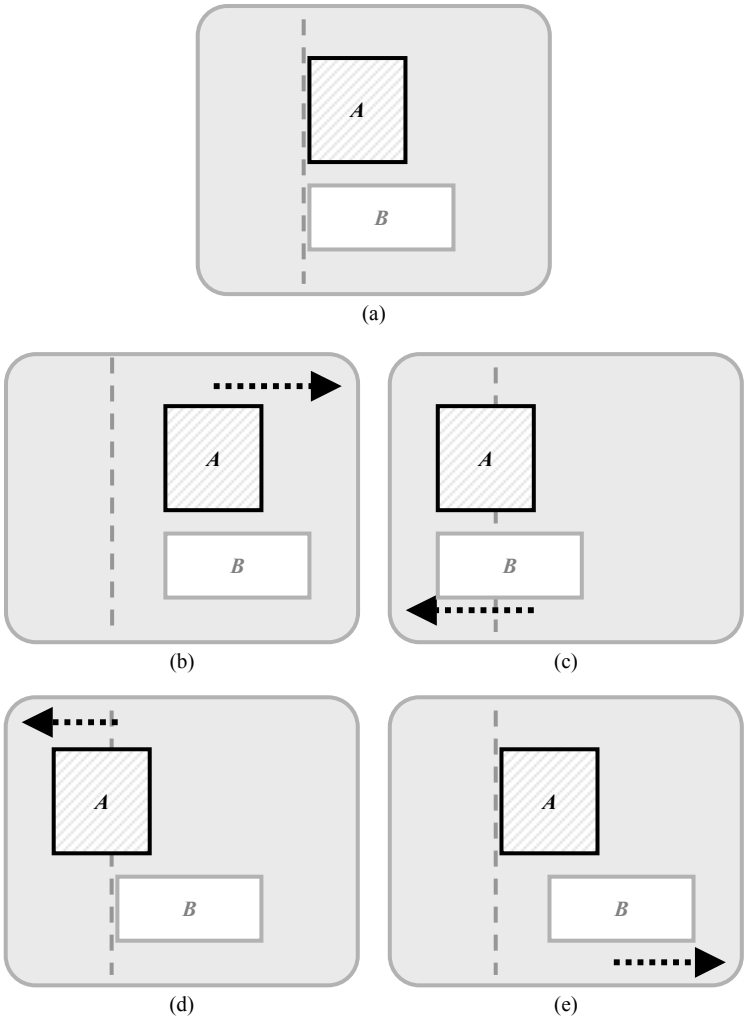


Figura 3.34 – Uso de componentes de referência em uma relação.

Para que relações espaciais desse tipo possam ser mantidas, a propriedade `movable` deve permitir que o posicionamento dos objetos envolvidos possa ser modificado. O mesmo ocorre para relações que envolvem propriedades de dimensionamento; seus objetos devem permitir a alterações dos valores das propriedades `width` e `height`, através da propriedade `resizable`.

Essa situação pode ser estendida para relações entre mais de dois objetos, onde o conceito de componente de referência é mantido. Pode-se definir, por exemplo, uma relação entre um número qualquer de objetos onde somente um deles é importante para a manutenção da relação. Ele é designado como componente de referência.

Finalmente, como uma relação pode envolver mais de uma propriedade de um mesmo objeto, é responsabilidade das implementações do modelo definir quais delas devem ter seus valores atualizados na necessidade de manutenção da relação.

A próxima seção apresenta um tipo de relação diferente. Ele associa a definição da sincronização espacial dos objetos de um documento com a sua sincronização temporal.

3.5.2 Relações espaço-temporais

O modelo espacial possibilita, além da definição de relações espaciais, a especificação de relações mais complexas, que envolvam atributos espaciais e temporais dos objetos de um documento. Essas relações são chamadas de relações espaço-temporais.

Exemplos de relações espaço-temporais são relações espaciais cuja validade é alterada de acordo com a ocorrência de eventos temporais (e/ou espaciais), durante a apresentação de um documento. Pode-se definir relações espaciais que só passem a valer quando algum evento (a seleção de um objeto, por exemplo) ocorrer e que possam ser alteradas com a ocorrência de outros eventos.

Relações espaço-temporais expressam relações de causalidade entre os objeto envolvidos, e não restrições como as relações apresentadas em 3.5.1.

A implementação de relações espaço-temporais é dependente do modelo de documentos utilizado. A Seção 4.3.3 sugere alguns tipos de relações espaço-temporais e apresenta alguns exemplos, implementados no modelo NCM.

3.6 SUMÁRIO

Neste capítulo foi apresentado um modelo genérico para a definição da sincronização espacial de documentos hipermídia. Esse modelo é composto da especificação de uma estrutura de apresentação, composta de uma hierarquia de janelas e regiões. As janelas são o elemento de nível mais alto da hierarquia. Elas devem ser associadas aos dispositivos de exibição disponíveis na plataforma de exibição.

Um documento pode referenciar uma estrutura de apresentação para sua exibição. Seus objetos de mídia podem ser associados às regiões dessa estrutura, herdando suas características espaciais, para que possam ser apresentados de forma apropriada.

Também foi apresentado como relações espaciais podem ser especificadas entre os objetos de mídia envolvidos em uma apresentação. Foram apresentadas algumas relações básicas que permitem alinhar objetos, estabelecer distâncias entre eles, centralizá-los e dimensioná-los, de acordo com as características espaciais de outros objetos.

4 SINCRONIZAÇÃO ESPACIAL NO MODELO DE CONTEXTOS ANINHADOS

No Capítulo 3 foi apresentado um modelo genérico para a definição da sincronização espacial da apresentação de documentos hipermídia. Neste capítulo, as entidades do modelo espacial são adaptadas e traduzidas em entidades e atributos de entidades do Modelo de Contextos Aninhados (NCM – *Nested Context Model*), um modelo conceitual para a definição de documentos hipermídia. Como poderá ser observado, várias entidades NCM tiveram que ser modificadas para que o modelo espacial pudesse ser completamente atendido.

Este capítulo encontra-se organizado com se segue. A Seção 4.1 apresenta o Modelo de Contextos Aninhados. A Seção 4.2 apresenta como as estruturas de apresentação propostas no modelo espacial podem ser definidas no NCM. A Seção 4.3 define como podem ser criadas relações espaciais e espaço-temporais entre objetos. Na Seção 4.4, é apresentado um breve sumário do capítulo.

4.1 O MODELO DE CONTEXTOS ANINHADOS

O Modelo de Contextos Aninhados é um modelo de documentos hipermídia baseado nos conceitos usuais de nós e elos. Nós são fragmentos de informação e elos são usados para a definição de relacionamentos entre os nós.

A seguir são apresentadas as entidades e atributos do modelo NCM que serão utilizadas na definição da sincronização espacial de documentos NCM. A referência [Soares00b] contém uma apresentação detalhada do modelo.

4.1.1 Nós e âncoras

Os principais atributos de um nó NCM são o seu conteúdo e um conjunto de descritores alternativos. O conteúdo de um nó é composto por uma coleção de unidades de informação. Uma unidade de informação (UI) é definida de acordo com a classe do nó.

O conjunto de objetos descritores alternativos contém um conjunto de descritores dos quais um deve ser (ou foi) selecionado (a escolha depende, por exemplo, de parâmetros definidos na plataforma de exibição onde o documento estiver sendo exibido), ou o valor nulo. O descritor selecionado de um nó determina como ele deve ser apresentado, no tempo e no espaço.

O modelo distingue duas classes básicas de nós, chamados de nós terminais (ou de conteúdo) e nós de composição.

Um nó terminal (ou nó de conteúdo) possui como atributos adicionais a especificação do tipo de conteúdo e uma lista ordenada de âncoras. O conteúdo de um nó define um conjunto de unidades de informação e é dependente da aplicação. O modelo permite que a classe de nós terminais seja especializada em outras classes (texto, áudio, imagem, etc.). A lista ordenada de âncoras é composta de âncoras. Uma âncora define uma região que identifica um subconjunto das unidades de informação do conteúdo do nó. Cada âncora da lista possui um nome (único na lista). Um tipo especial de âncora, existente em qualquer nó NCM, definido pelo símbolo especial λ, representa todo o conteúdo do nó.

Um nó de composição C é um nó cujo conteúdo é uma coleção C_L de nós (de conteúdo e de composição, recursivamente) que se constituem em suas unidades de informação.

Subclasses de composição definem semânticas para coleções específicas de nós. Uma importante subclasse de composição definida pelo modelo é a classe nó de contexto de usuário.

Um nó de contexto de usuário, ou simplesmente nó de contexto, é um nó de composição cujo conteúdo é composto de um conjunto de nós terminais ou de contexto, recursivamente. Os nós de contexto possuem como atributos adicionais uma lista ordenada de âncoras, um conjunto de elos e uma coleção de apresentação.

Assim como nos nós de conteúdo, a lista ordenada de âncoras de um nó de contexto é composta de âncoras. Entretanto, uma vez que os nós contidos em uma composição se constituem nas suas unidades de informação, a região de uma âncora de um nó de contexto C é um subconjunto dos nós contidos em C, ou o conteúdo de C como um todo (âncora λ).

Cada elo contido no conjunto de elos de um nó de contexto *C* define uma relação entre eventos de nós recursivamente contidos em *C*. Eventos serão definidos na Seção 4.1.2 e elos NCM (e nós cabeça de elo) serão definidos de forma detalhada na Seção 4.1.3.

Uma coleção de apresentação contém para cada nó contido em um nó de contexto *C*, um grupo de conjuntos de descritores, ou o valor nulo. Assim, tem-se, para cada nó de *C*, um conjunto de descritores selecionados, ou o valor nulo. No caso do nó contido em *C* ser um nó de contexto, o conjunto de descritores selecionados contém no máximo um objeto descritor. A noção de descritor, como já mencionado, será definida na Seção 4.1.4.

Nós de contexto servem, entre outras coisas, para definir uma estrutura hierárquica para documentos.

A seção a seguir apresenta a definição de evento.

4.1.2 **Evento**

Seguindo a definição de Pérez-Luque [Perez-Luque96], um evento é uma ocorrência no tempo que pode ser instantânea ou durar um período de tempo. No NCM, um evento é a exibição, evento de exibição, ou a seleção, evento de seleção, de uma âncora de um nó em uma dada perspectiva seguindo as diretrizes de um dado descritor. O NCM define também um terceiro tipo de evento, denominado evento de atribuição, que corresponde à mudança de valor de um atributo de um nó ou da condição de habilitação das mudanças de comportamento definidas no objeto descritor (como se verá na Seção 4.1.4).

Um evento pode estar em um dos seguintes estados: *dormindo*, *preparando*, *preparado*, *ocorrendo*, *suspenso* e *abortado*. A manutenção do estado dos eventos é responsabilidade da máquina de controle da apresentação dos documentos, denominada de formatador, detalhada em [Rodrigues97 e Bachelet00]. Todo evento possui um atributo denominado de *ocorrências*, que conta o número de vezes que ele muda do estado *ocorrendo* para o estado *preparado* durante a apresentação de um documento. Eventos de exibição também possuem um atributo denominado de *repetições*, que determina o número de vezes seguidas que ele deve ser exibido. Esse atributo pode conter um valor finito ou o valor *indefinido*, que levará a uma execução ininterrupta do evento, até que ela seja interrompida.

Eventos de exibição possuem também, além dos atributos *ocorrências* e *repetições*, um terceiro atributo denominado *duração*. Esse atributo guarda o tempo em que o evento deve permanecer no estado ocorrendo durante a apresentação do documento. Conforme será

apresentado na Seção 4.1.4, a duração de um evento será escolhida pelo *formatador* (máquina de controle da apresentação), baseando-se em parâmetros especificados no descritor, nas relações definidas para o documento e em outras informações, como a descrição da plataforma onde o documento será apresentado.

Intuitivamente, um evento inicia no estado *dormindo*. Ele passa para o estado *preparando* e nele permanece enquanto estiver sendo executado algum procedimento de *pre-fetch* de suas unidades de informação. Ao final do procedimento, o evento passa para o estado *preparado*, tendo seu atributo *repetições* iniciado, conforme será visto na Seção 4.1.3. 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 *suspenso* e nele permanece enquanto a situação durar. Ao final da apresentação, o evento retorna para o estado *preparado*, seu atributo *ocorrências* é incrementado de uma unidade e o atributo *repetições* é decrementado de uma unidade. Se, após ter sido 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 estado *abortado* e imediatamente depois volta ao estado *preparado*, 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, por exemplo, através de uma interação do usuário. Eventos de atribuição permanecem no estado *ocorrendo* enquanto a operação de atribuição durar. Evidentemente, eventos podem permanecer por um tempo infinitesimal no estado *ocorrendo*. A máquina de estados genérica dos eventos NCM é apresentada na Figura 4.1.

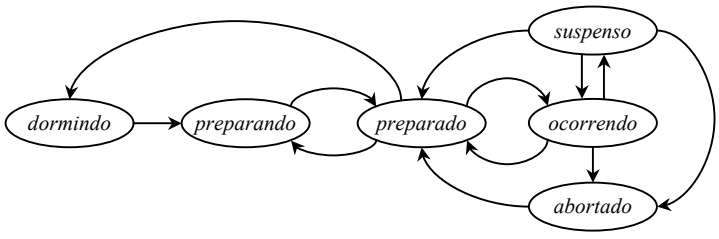


Figura 4.1 – Máquina de estados de eventos.

A próxima seção apresenta a definição da entidade elo do modelo NCM. Um elo, como mencionado anteriormente, representa relações entre eventos de nós que referencia. Estes nós são sempre objetos de representação. Um objeto de representação é criado a partir de um descritor e um nó associado. Mais precisamente, um objeto de representação é criado pelo

executor de apresentações, ou formatador NCM (o equivalente ao instanciador do Modelo Dexter [Halasz90]), a partir de um identificador de descritor e de um identificador de nó, objeto de dados. Os objetos de representação são responsáveis pela detecção e sinalização da ocorrência dos eventos. Detalhes a respeito da definição de objeto de dados e de representação podem ser encontrados em [Soares00b].

4.1.3 Elo e Ponto de Encontro

Um elo é uma entidade cujos atributos principais são o conjunto de pontos terminais de origem, o conjunto de pontos terminais de destino e o ponto de encontro. O conjunto de pontos terminais origem e destino definem eventos em objetos de representação e o ponto de encontro do elo define relações entre eventos. Múltiplos pontos terminais permitem definir relações m:n.

Os valores dos atributos conjunto de pontos terminais de origem e conjunto de pontos terminais de destino de um elo são conjuntos cujos elementos, chamados pontos terminais de um elo, são quádruplas da forma: $<(N_k, ..., N_2, N_1), \alpha, tipo, D>$ tal que:

1. N_{i+1} é um nó de contexto e N_i está contido em N_{i+1}, para todo $i \in (0,k)$, com k>0.
2. α é a identificação de uma âncora ou um atributo de N_1.
3. *tipo* especifica o tipo do evento (de exibição, seleção ou de atribuição) associado a α. No caso de evento de atribuição, o valor de α deve identificar o atributo ou uma âncora. No caso de evento de exibição ou seleção, α é sempre uma âncora.
4. *D* é um conjunto de objetos descritores alternativos.

Os relacionamentos entre eventos associados a um elo são expressos através do seu atributo ponto de encontro. A estrutura de um ponto de encontro varia de acordo com o tipo de elo. Existem dois tipos de elo: elo causal e elo de restrição.

Um objeto *ponto de encontro* de um *elo causal* define uma operação composta por uma condição e uma ação. Caso a condição seja satisfeita, a ação é executada. A condição de um elo causal diz respeito aos pontos terminais de origem do elo, enquanto que a ação deve ser executada sobre os pontos terminais de destino.

A condição de um ponto de encontro de elo causal avalia valores lógicos e pode ser binária simples ou composta. Uma condição binária simples (ou simplesmente condição 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 simples é satisfeita se tanto a condição prévia quanto a condição corrente forem satisfeitas. Tanto a condição prévia quanto a corrente podem receber o valor *VERDADE*, caso elas não sejam relevantes na avaliação da condição simples associada. Os operadores de comparação usados na avaliação das condições simples são: = (igualdade); ≠ (desigualdade); < (inferioridade); ≤ (igualdade ou inferioridade); > (superioridade); e ≥ (igualdade ou superioridade).

As comparações são realizadas com respeito aos estados de um evento (apenas as comparações = e ≠ são permitidas); a variável *ocorrências* ou a variável *repetições* associadas a um evento; e aos atributos de um nó, no caso de um evento de atribuição.

Qualquer expressão de condições baseada nos operadores lógicos ∧ (e), ∨ (ou) e ¬ (negação) definem uma condição composta. Além disso, a qualquer condição, simples ou composta, pode ser aplicado o operador modal ® (denominado retardo). O operador retardo aplicado a uma condição C é definido da forma C ® $[t_1, t_2]$, 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 ® $[t_1, t_2]$, é verdade no intervalo de tempo $[t+t_1, t+t_2]$.

A ação de um ponto de encontro do elo causal pode ser simples ou composta. Uma ação simples pode:

a) ser aplicada a pontos terminais que definem eventos de exibição em um dado objeto de representação:

1. *Prepara(E)*: se o evento E estiver no estado *dormindo*, ele passa para o estado *preparando*, caso contrário, nenhuma transição ocorre. Ao final do procedimento de preparação, o atributo *repetições* de E é iniciado com o valor especificado no descritor associado ao nó definido pelo evento. Caso não haja valor especificado, o atributo recebe o valor *default* 1.
2. *Inicia(E, n)*: estando o evento E nos estados *dormindo*, *preparando*, *preparado*, *ocorrendo* ou *suspenso*, ele passa para o estado *ocorrendo*. Se o estado inicial for *dormindo*, *preparando*, *preparado* ou *suspenso*, a apresentação do documento é iniciada a partir de seu começo (primeira unidade de informação). Se o estado inicial for o próprio estado *ocorrendo*, nada acontece. Essa ação possui ainda um parâmetro opcional para

especificar um número *n* de repetições para a apresentação do evento. Caso esse parâmetro não seja especificado, o atributo *repetições* de *E* é iniciado com o valor declarado no descritor associado ao nó âncora. No caso de *E* ser o evento de exibição da âncora λ de um nó de contexto (ou seja, todo o nó), deve-se ainda indicar se a exibição do evento será invisível ou visível.

3. *Termina(E)*: muda o estado do evento *E* de *ocorrendo* ou *suspenso* para *preparado*. Essa ação incrementa de uma unidade o valor do atributo *ocorrências* e decrementa de uma unidade o atributo *repetições* do evento. Se o estado inicial for qualquer outro, nada acontece. Se após decrementado o atributo *repetições* for maior que zero, uma nova exibição do evento é iniciada automaticamente.
4. *Suspende(E)*: muda o estado do evento *E* de *ocorrendo* para *suspenso*. Se o estado inicial for qualquer outro, nada acontece.
5. *Reassume(E)*: muda o estado do evento *E* de *suspenso* para *ocorrendo*, retomando a exibição a partir da última unidade de informação exibida antes da apresentação do evento ter sido interrompida. Se o estado inicial for qualquer outro, nada acontece.
6. *Aborta(E)*: muda o estado do evento *E* de *ocorrendo* ou *suspenso* para *abortado*, e imediatamente depois para *preparado*, sem incrementar o valor do atributo *ocorrências* e tornando zero o valor do atributo repetições do evento. Se o estado inicial for qualquer outro, nada acontece.

b) ser aplicada a pontos terminais que definem eventos de atribuição em um dado objeto de representação:

1. *Habilita(E)*: habilita a lista de operações associada ao evento *E*. Essas operações são definidas nos objetos descritores, conforme será visto na Seção 4.1.4. O atributo α do ponto terminal que define o evento deve identificar uma âncora.
2. *Inibe(E)*: inibe a lista de operações associada ao evento *E*. O atributo α do ponto terminal que define o evento deve identificar uma âncora.
3. *Ativa(E, c)*: ativa a lista de operações associada ao evento, passando como parâmetro a condição satisfeita que desencadeou a ação. O atributo α do ponto terminal que define o evento deve identificar uma âncora. As operações são executadas somente se estiverem habilitadas.

4. *Atribui valor absoluto(E, i)*: o atributo especificado por α no ponto terminal que define o evento *E* passa a conter o valor *i*.

5. *Atribui valor relativo(E, i)*: o atributo especificado por α no ponto terminal que define o evento *E* passa a conter o valor anterior somado a *i*.

Uma ação composta é formada por uma expressão de ações baseada nos operadores | (paralelo) e $\rightarrow$ (seqüencial), definindo a ordem de execução de cada elemento da ação. Além disso, toda ação, simples ou composta, possui um atributo opcional denominado *tempo de espera*. Esse atributo define um tempo que deve ser aguardado antes que a ação seja executada. Da mesma forma que a duração do evento (que será apresentada em detalhes na Seção 4.1.4), esse atributo é especificado através de uma função de custo.

Um elo causal pode ser de dois tipos: *hiper-elo* e *sinc-elo*. Um hiper-elo é um elo que possui pelo menos um evento de seleção associado a um dos elementos de seu conjunto de pontos terminais de origem. Um sinc-elo é um elo cujos pontos terminais de origem não estão associados a eventos de seleção.

Relações de causa e efeito (condições/ações) não são suficientes para determinar todas as relações existentes entre objetos de representação. Existem relações que, semanticamente, especificam restrições (*constraints*) entre os pontos terminais do elo, tais como: dois nós devem terminar sua exibição ao mesmo tempo, *se e somente se* os dois nós estiverem sendo exibidos.

Um *elo de restrição* é uma entidade cujo conjunto de pontos terminais de destino é nulo. O conjunto de pontos terminais de origem define eventos em objetos de representação e o ponto de encontro da restrição vai definir restrições entre esses eventos.

Um objeto ponto de encontro de um elo restrição contém uma única operação, denominada *simultânea*, composta por um conjunto de condições binárias simples. A semântica da operação é que todas as condições prévias que forem *VERDADE* em um dado instante de tempo, devem ter nas condições correntes correspondentes o mesmo valor (*VERDADE* ou *FALSO*), neste mesmo instante de tempo. Condições binárias simples dizem respeito aos pontos terminais de origem e têm as mesmas definições que nos elos causais.

4.1.4 Descritor

Um descritor especifica como um nó, objeto de dados, deve ser exibido, detalhando como ele deve ser iniciado, qual dispositivo de E/S deve ser utilizado e quais mudanças ocorrerão no seu comportamento durante a exibição. A associação de um nó a um objeto descritor, como mencionado na Seção 4.1.2, cria um objeto de representação.

Os atributos principais de um descritor são uma especificação de iniciação, uma especificação de término e uma coleção de descrições de eventos. Atributos podem ser adicionados a um descritor de acordo com a classe de nó a ele associado.

Uma especificação de iniciação contém as informações necessárias para iniciar a apresentação de uma entidade. Ela deve definir todos os parâmetros necessários para a criação de um objeto de representação a partir de um nó.

Uma especificação de término contém as informações necessárias para finalizar a apresentação de uma entidade. Uma especificação de término possui também uma lista ordenada de operações que devem ser executadas ao finalizar a exibição do nó.

A noção exata do que constitui uma especificação de iniciação e uma especificação de término depende da classe do nó (objeto de dados) ao qual o descritor será associado No caso mais simples, a especificação de início e término identificará um controlador, passando, a este, parâmetros para seu funcionamento correto.

A lista de operações contém uma seqüência ordenada de operações. Tal qual o objeto ponto de encontro de um elo causal, cada operação da seqüência é composta por uma condição e uma ação, que podem ser simples ou compostas. A satisfação da condição implica o disparo da ação associada. As condições da lista de operações são semelhantes às do objeto ponto de encontro de um elo causal, porém, o escopo das entradas está limitado aos eventos e atributos do nó associado ao descritor, ou aos atributos definidos no próprio descritor.

As ações de uma lista de operações, novamente, são dependentes da classe do nó ao qual o descritor será associado. As ações devem sempre corresponder a uma alteração de comportamento da exibição do objeto de representação, criado a partir do descritor. Por exemplo, no caso de áudio, uma ação de mudança de comportamento poderia ser “coloque o volume a X dB”.

A *descrição de um evento*, em um descritor, por sua vez, consiste da tupla <α, *tipo*, *dur*, *rep*, *oper*, *hab*>. O parâmetro α é um identificador de uma âncora pertencente à coleção de

âncoras do nó ao qual o descritor será associado para a formação do objeto de representação. *Tipo* especifica o tipo de evento associado à âncora (exibição, seleção) ou tem o valor especial ξ. *Dur* especifica uma função de custo para calcular a duração do evento, sendo válida apenas para eventos de exibição. No caso de um evento de seleção esse atributo recebe o valor nulo. *Rep* também só é válido para eventos de exibição e especifica o número de repetições do evento. Esse parâmetro inicia o atributo *repetições* do evento, caso o mesmo não seja iniciado por uma ação de um elo. *Oper* especifica um objeto lista de operações, que deve ser executada caso o evento associado à âncora ocorra e se o atributo *habilita* permitir. *Hab* inicia o atributo *habilita* do evento, que indica se a lista de operações pode ser executada ou não. Se o parâmetro *tipo* especificar um evento de exibição ou seleção, e o atributo *habilita* permitir, a lista de operações é avaliada sempre que houver uma mudança no estado do evento. Caso o parâmetro *tipo* receba o valor ξ, e o atributo *habilita* permitir, a lista de operações deve ser avaliada se o nó associado ao descritor receber uma mensagem de um elo especificando a âncora, resultado da ação *ativa* de um ponto de encontro.

Uma lista de operações pode ter restrições de sincronização temporal, como por exemplo, a ação X deve ser executada daqui a 5 segundos.

Mudanças de comportamento podem alterar a duração de um evento. Exemplos são: aumente a velocidade de exibição, dê uma pausa por 10 segundos, etc.

A duração de um evento de exibição (definida pelo atributo *Dur*) é especificada por uma função de custo, da mesma forma que o atributo *tempo de espera* de uma ação de um ponto de encontro ou descritor. A idéia é permitir a um autor especificar uma função "duração *versus* custo", onde os valores mínimo, ideal e máximo são definidos para a duração do evento (ou tempo de espera). A partir dessa função, o formatador pode inferir o intervalo de tempo que uma exibição de um evento (ou tempo de espera de uma ação), pode assumir, bem como a duração (ou tempo de espera) que levaria a uma melhor qualidade de exibição, dada por um ponto mínimo da função. A função também permite ao formatador determinar o custo correspondente a não se assumir a qualidade ideal, correspondendo a esticar ou encolher a duração do evento (ou do tempo de espera). A Figura 4.2 apresenta exemplos de funções custo.

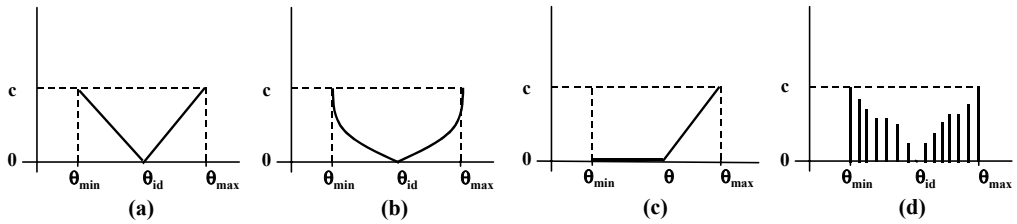


Figura 4.2 – Exemplos de funções de custo.

a) função de custo linear, com um valor ideal (θ_{id}); b) função de custo não linear com um valor ideal (θ_{id}); c) função de custo linear com múltiplos valores ideais ([θ_{min}, θ]); d) função de custo discreta com um valor ideal (θ_{id}).

O modelo permite que as durações sejam especificadas não apenas com valores reais, mas também com valores imprevisíveis. Eventos de exibição que são apresentados até que alguma ação externa os interrompa ou eventos que terminam sua exibição por conta própria, em um instante desconhecido *a priori*, são exemplos de eventos de exibição que possuem duração imprevisível.

4.2 DEFINIÇÃO DA APRESENTAÇÃO DE DOCUMENTOS NCM

A apresentação de um documento NCM é determinada pela associação de descritores aos nós do documento. Recordando, um descritor define a apresentação de um nó associado através de uma especificação de início, uma especificação de término e uma coleção de descrições de eventos (ver Seção 4.1.4).

O modelo espacial apresentado no Capítulo 3 pode ser utilizado na definição de descritores a serem utilizados em documentos NCM. A seção a seguir mostra como estruturas de apresentação podem ser definidas em objetos descritores.

4.2.1 Definição de janelas e regiões

A estrutura de apresentação de um documento hipermídia, segundo o modelo apresentado no Capítulo 3, é composta de um conjunto de janelas, associadas a dispositivos de apresentação, compostas de hierarquias de regiões. Para que um documento NCM possa usar uma estrutura de apresentação, seus nós devem ser associados a descritores onde os componentes de tal estrutura estejam especificados.

Cada descritor deve determinar em que região da estrutura de apresentação um nó associado deve ser apresentado. A especificação dessa região pode ser feita de duas maneiras:

- pode-se adicionar uma série de atributos (correspondentes às propriedades da entidade *região* do modelo espacial, apresentada na Seção 3.1.2) ao descritor que descrevam a região a ser usada na apresentação do objeto; ou
- pode-se adicionar um novo atributo, `region`, ao descritor, associando a região a ser usada na apresentação do objeto. Quando um nó vier a utilizar o descritor, esse deverá fazer uma cópia da região original. Isso previne que dois nós utilizem o mesmo objeto região para sua apresentação e interfiram na apresentação do outro objeto, como definido na Seção 3.4. A definição de regiões é feita através de uma nova entidade, *região*, adicionada ao modelo de documentos

Vale salientar que tanto nós de conteúdo como nós de contexto podem estar associados a objetos descritores. A apresentação de nós de conteúdo corresponde à exibição do objeto associado ao seu atributo *conteúdo*, enquanto que a apresentação de nós de contexto corresponde à exibição de sua estrutura.

A Figura 4.3 apresenta um exemplo de descritor construído com base no modelo espacial. Na especificação de inicialização, observa-se a existência das propriedades `left`, `top`, `width`, `height` e `z-index`, que identificam uma região de uma estrutura de apresentação. Os outros atributos espaciais foram omitidos.

```
Descriptor d1:
      Begin:
            ...
            left:    10
            top:     50
            width:   200
            height:  300
            z-index: 1
            ...
      End:
            ...
      Events:
            ...
```

Figura 4.3 – Exemplo de especificação de uma região em um descritor.

Segundo o modelo espacial, a definição de uma região não é suficiente para especificar a apresentação de um objeto. É preciso definir a janela em que a região deve estar contida. Como apresentado na Seção 3.1.2, quando agregada ao modelo de documentos, a definição de janelas e regiões é restringida, de forma que regiões devem estar contidas diretamente nas janelas que compõem a estrutura de apresentação do documento.

Assim como a definição de sua região, a especificação da janela de apresentação de um nó é pode ser feita de duas formas:

- através da definição de um conjunto de atributos, equivalentes às propriedades de uma janelas, definidas na Seção 3.1.1; ou
- através da adição de um novo atributo, `window`, ao descritor associado ao nó. Esse atributo deve conter uma referência a um objeto que represente a janela a ser utilizada, diferentemente do que ocorre na definição de regiões. Uma nova entidade, *janela*, é adicionada ao modelo de documentos para que a definição de janelas possa ser feita.

Essa nova entidade possui os mesmos atributos que a entidade *janela* do modelo espacial, apresentadas na Seção 3.1.1. Dentre elas, encontra-se a definição do dispositivo de exibição onde ela deve ser apresentada.

A criação de uma entidade para a especificação de janelas (ver Seção 3.1), especialmente no caso do modelo NCM, também serve a outro propósito: cada janela usada pelo documento pode ser diretamente associada a uma ferramenta de apresentação do tipo *janela*. Uma proposta de definição de ferramentas de exibição para documentos hipermídia pode ser encontrada em [Rodrigues01].

A Figura 4.4 apresenta um exemplo de estrutura de apresentação. Ela é composta de duas janelas, *window1* e *window2*. A primeira delas contém somente uma região, *region1*, e a segunda, duas, *region2* e *region3*. A região *region2*, por sua vez é composta de mais duas regiões, *region4* e *region5*.

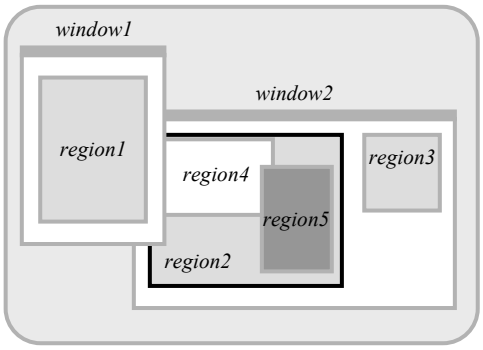


Figura 4.4 – Estrutura de apresentação de um documento.

Na Figura 4.5 são mostrados os descritores que devem ser usados para que a estrutura de apresentação da Figura 4.4 possa ser usada por documentos NCM, no posicionamento de seus

nós com relação aos dispositivos de exibição utilizados. Os descritores *d1*, *d2*, *d3*, *d4* e *d5* definem as regiões *region1*, *region2*, *region3*, *region4*, *region5*, respectivamente.

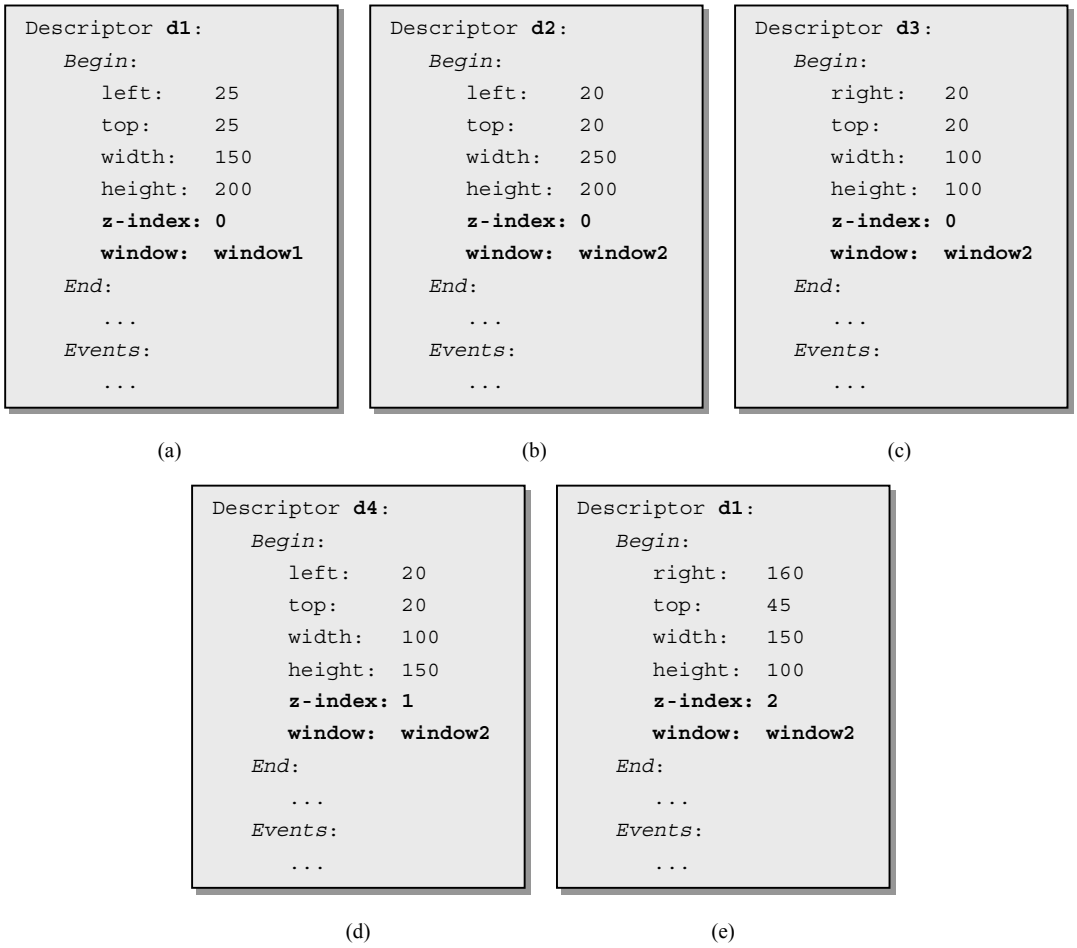


Figura 4.5 – Descritores NCM usados para especificar uma estrutura de *layout*.

Observa-se que o posicionamento de todas as regiões é feito de forma relativa às janelas que os contêm. Os atributos `z-index`, que definem a prioridade de apresentação de cada uma das regiões, também são definidos segundo o escopo de apresentação de cada uma das janelas.

Deve-se salientar que os valores das propriedades de janelas e regiões usadas por descritores NCM devem ter valores absolutos. Caso janelas e/ou regiões tenham sido definidas inicialmente através da linguagem declarativa e participem de relações espaciais, seus valores absolutos devem ser gerados automaticamente.

O posicionamento de objetos de um documento NCM pode ser definido de forma relativa ao de outros objetos. Isso é feito através da criação de relações entre os atributos espaciais desses objetos. Essas relações são definidas através de elos NCM, assunto da próxima seção.

A próxima seção apresenta como relações entre atributos de nós NCM podem ser definidos através de elos.

4.3 DEFINIÇÃO DE RELAÇÕES ENTRE ATRIBUTOS DE NÓS

Como dito na Seção 3.5.1, uma relação espacial entre objetos nada mais é do que uma relação entre atributos espaciais desses objetos. No NCM, essas relações são definidas relacionando-se atributos de nós objetos de representação de um documento.

De uma forma geral, essas relações são definidas envolvendo eventos de atribuição desses objetos. A definição anterior de evento de atribuição no modelo NCM teve que ser estendida para que possibilitasse a criação de tais elos. A subseção a seguir apresenta essa nova definição. Posteriormente, nas Seções 4.3.2 e 4.3.3, a criação de relações espaciais e espaço-temporais entre objetos NCM é apresentada. Ainda, a Seção 4.3.4 mostra como a animação da apresentação de um objeto NCM pode ser definida através de eventos de atribuição.

4.3.1 Redefinição do Evento de Atribuição

Os atributos de um objeto de representação têm seus valores iniciados na definição de especificações de início de objetos descritores, utilizados para a sua criação. Naturalmente, o valor desses atributos pode ser alterado durante a apresentação do objeto.

Uma atribuição corresponde à mudança do valor de um atributo de um objeto de representação. Essa mudança possui uma **duração**, durante a qual o valor do atributo deve variar uniformemente entre um **valor inicial** e um **valor final**.

Quaisquer uns dos valores iniciais e finais podem ser definidos em função do valor corrente do atributo. Isso é feito usando-se o símbolo ∅, que é substituído pelo valor corrente do atributo.

A duração de uma atribuição é determinada por uma função de custo (ver Seção 4.1.4) e pode ser adaptada pelo formatador. Ele controla a velocidade de modificação do valor de atributo entre o valor inicial e o valor final especificados. Quando a duração não é especificada, é assumido o valor 0 (zero).

O valor final de um atributo pode ser definido como função dos valores de outros atributos. Quando isso ocorre, a duração da atribuição não precisa ser especificada. Ela é definida implicitamente pela duração da variação dos atributos do domínio da função.

Quando o valor inicial de uma atribuição não é definido (valor *nulo*), ele deve assumir o valor final (que deve ser sempre especificado).

Uma atribuição pode ser usada, por exemplo, para determinar uma variação do valor da largura de um objeto de um valor inicial de 10 *pixels* a um valor final de 100 *pixels*, em um período de tempo determinado por uma função de custo qualquer. A variação da largura do objeto pode ser ilustrada na Figura 4.6. O valor da propriedade varia uniformemente entre os valores definidos, em um período de tempo determinado pelo formatador com base na função de custo especificada. Essa duração determina a velocidade com que o valor do atributo varia. Na Figura 4.6 são ilustradas duas possíveis durações para a atribuição. Caso a duração não tivesse sido especificada, a atribuição seria instantânea e o a largura do objeto assumiria o valor final imediatamente.

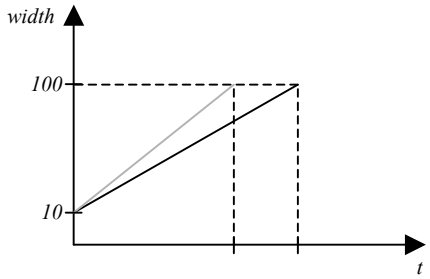


Figura 4.6 – Exemplo de variação da largura de um objeto.

Através de atribuições, pode-se animar a apresentação de um objeto, modificando o seu posicionamento ao longo de sua apresentação. Considerando que a posição de um objeto é determinada por um par ordenado (*x*, *y*), que identifica o canto superior esquerdo do retângulo que o representa, pode-se definir as seguintes atribuições:

a) Atribuição de *x*

- valor inicial: 50;
- valor final: 100.
- duração: *t*.

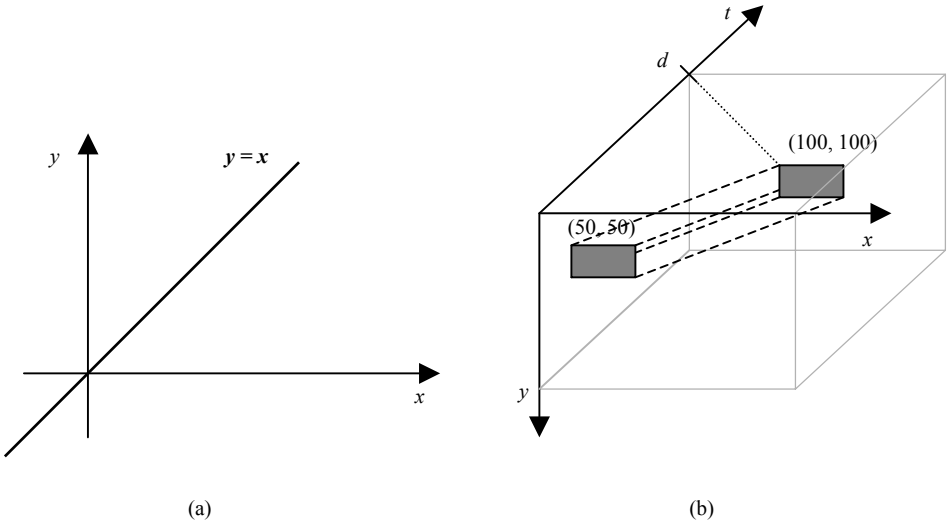
b) Atribuição de *y*

- valor final: *f(x)* = x.

O valor de *x* varia de 50 a 100 no intervalo de tempo determinado pela função de custo *t*. O valor de *y* é determinado pela função *f(x)* definida pelo valor final de sua atribuição. Como pode ser observado, tanto o valor inicial como a duração da atribuição de *y* não foram

definidos. Dessa forma, o valor inicial de *y* é determinado pela função definida pelo valor final e a duração da atribuição é dependente da duração da atribuição de *x*, determinada por *t*.

A animação, resultado das atribuições é ilustrada na Figura 4.7 (b). Observa-se que, segundo a função *f(x)* usada na atribuição de *y* ((a)), o valor de *y* permanece igual ao de *x* durante toda a atribuição. As animações devem iniciar simultaneamente para que o efeito total da animação seja atingido. Caso, por exemplo, a atribuição de *y* seja acionada durante a ocorrência da atribuição de *x*, o valor de *y* mudará instantaneamente de seu valor corrente para o valor corrente de *x* e continuará o progresso determinado pelas atribuições até o valor final ser atingido.



(a) (b)

Figura 4.7 – Exemplo de animação de um objeto durante sua apresentação.

Como apresentado na Seção 4.1.2, o modelo NCM define três tipos de eventos: evento de apresentação, de seleção e de atribuição. O evento de atribuição, em especial, é utilizado para representar a atribuição de valores estáticos a atributos dos objetos de apresentação ou a alteração no estado de habilitação da lista de operações de eventos de exibição e de seleção de objetos de representação.

Com o objetivo de comportar a especificação de relações entre atributos, seja de um mesmo objeto de representação ou de objetos diferentes, o evento de atribuição teve sua definição alterada. Deve-se salientar, no entanto, que o seu uso na alteração do estado de habilitação das listas de operações associadas a eventos de exibição e seleção não foi modificado.

Um evento de atribuição representa a atribuição de um conjunto de valores a um atributo. Ele pode estar nos estados *dormindo*, *preparando*, *preparado*, *ocorrendo*, *suspenso* ou *abortado*. Assim como o evento de exibição, ele possui um atributo de nome *ocorrências*, que conta

quantas vezes há uma mudança do estado ocorrendo para o estado preparado, e um atributo de nome *repetições*, que determina o número de vezes seguidas que ele deve ocorrer. A definição de ambos os atributos é feita da mesma forma que nos eventos de exibição (ver Seção 4.1.2). A duração de uma atribuição é determinada pelo atributo *duração*. Ele define o tempo em que o evento deve permanecer no estado ocorrendo durante a apresentação do documento. Da mesma forma que nos eventos de exibição, a duração de um evento é escolhida pelo formatador. Caso a duração de um evento de atribuição não seja definida, ela deve assumir um valor instantâneo.

Um evento de atribuição, seja qual for o atributo cujo valor esteja sendo modificado, é associado implicitamente ao evento de exibição da âncora λ do nó (objeto de representação). O evento inicia no estado *dormindo*, permanece no estado *preparando* enquanto o conteúdo do nó está sendo recuperado e passa ao estado *preparado* quando a recuperação é concluída. Nesse momento, ele tem seus atributos *repetições* e *duração* iniciados. Assim que algum valor é atribuído ao atributo, o evento passa ao estado *ocorrendo*. Durante a sua ocorrência, o evento pode ser temporariamente suspenso, indo para o estado *suspenso.* Ao final de sua ocorrência, determinada pelo atributo duração, o evento passa para o estado *preparado*. Nesse instante, seu atributo *ocorrências* é incrementado e seu atributo *repetições* é decrementado. Caso o atributo *repetições* possua um valor maior que zero, a atribuição é reiniciada automaticamente. A ocorrência do evento pode ser abortada abruptamente. Nesse caso, ele passa pelo estado *abortado* e vai para o estado *preparado*, terminando automaticamente a atribuição; o número de ocorrências do evento recebe o valor zero e o número de repetições não é decrementado. A máquina de estados do novo evento de atribuição é apresentada na Figura 4.8. Ela é igual à máquina apresentada na Figura 4.1.

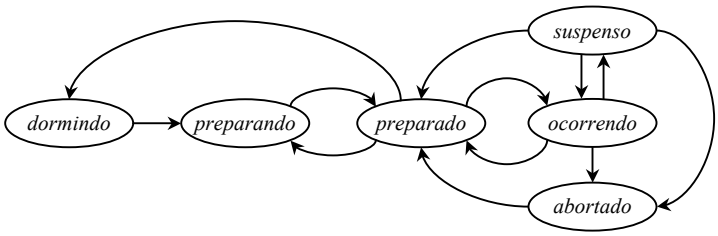


Figura 4.8 – Máquina de estados do evento de atribuição.

A descrição de um evento de atribuição consiste da tupla <α, *tipo*, *dur*, *rep*, *oper*, *hab*>. O parâmetro α deve identificar uma âncora do objeto de dados, ou um atributo do descritor, ou um atributo do objeto de dados associado. *Tipo* deve conter o valor atribuição ou o valor

especial ξ (usado na ativação da lista de operações do evento, conforme mostrado na Seção 4.1.2). *Dur* especifica uma função de custo que determina a duração do evento. *Rep* define o número de repetições do evento. Ele inicia o atributo *repetições* do evento, caso o mesmo não seja iniciado por uma ação de um elo. Caso α se referencie a um atributo, *hab* indica se o atributo associado ao evento pode ou não receber algum valor, proveniente da ação de um elo causal ou de uma operação de alguma lista de operações; caso α seja um o identificador de uma âncora, *hab* indica se a lista de operações associada ao evento está habilitada ou não.

A atribuição de valores a atributos deve ser feita através de ações de elos causais ou de operações de uma lista de operações especificada em algum descritor. O conjunto de ações que podem ser aplicados a pontos terminais que definem eventos de atribuição foi redefinido, conforme descrito abaixo:

- as ações *Habilita(E)*, *Inibe(E)* e *Ativa(E, c)* continuam a ser usadas, respectivamente, para habilitar, inibir e ativar a lista de operações associada a um evento *E*, desde que ele não referencie um atributo. Nesse caso, apenas as ações *Habilita(E)* e *Inibe(E)* podem ser usadas, servindo, respectivamente, para habilitar e inibir a atribuição de valores ao atributo associado ao evento;
- *Atribui valor (E, valor, d, r)*, onde *valor* define que o conjunto de valores do atributo especificado pelo evento *E*. O parâmetro *valor* é composto pelo par *(v_i, v_f), onde* v_i define o valor inicial e v_f, o valor final da atribuição. O parâmetro *d* define uma função de custo que determina a duração da variação. Quando a duração recebe o valor *nulo*, a variação do valor inicial para o valor final dá-se instantaneamente. Finalmente, o parâmetro *r* define um número de repetições da variação (seu valor *default* é 1). A execução dessa ação determina uma mudança do estado do evento de *preparado* para *ocorrendo*.

  O valor final de uma atribuição pode ser definido em função de atributos de outros objetos do documento. Quando a atribuição é definida como a ação de um elo causal, os atributos devem fazer parte do conjunto de pontos terminais de origem; quando a atribuição é definida como ação de uma das operações da lista de operações da descrição de um evento em um descritor, os atributos devem estar definidos no descritor ou no objeto de dados associado.

  Também é permitido definir valores iniciais e finais relativos ao valor atual do atributo. Para isso, deve-se usar a o valor $\varnothing$, que identifica o valor do atributo no instante da execução da ação. Esse símbolo é substituído pelo valor corrente do atributo no instante da

execução da ação de atribuição. Exemplos de uso desse símbolo serão vistos na Seção 4.3.4. A ação *Atribui valor* substitui as ações *Atribui valor absoluto* e *Atribui valor relativo*, da versão anterior do NCM.

A atribuição de um valor ao atributo de um objeto, através da ação *Atribui valor*, só ocorre quando este estiver sendo apresentado, isto é, quando o evento de exibição de sua âncora λ estiver no estado *ocorrendo*. Dessa forma, caso a ação *Atribui valor* seja executada sobre algum atributo de um objeto que não esteja sendo apresentado, nada acontece.

Finalmente, o valor de um atributo pode ser obtido através da operação *valor*. Por exemplo, o valor do atributo *attr* pode ser obtido através da expressão *valor*(*attr*). Essa operação é utilizada na atribuição relativa, quando o valor de atributo depende do valor de outro atributo. Exemplos de uso serão vistos nas definições de relações espaciais na Seção 4.3.2.

- são acrescentadas três ações novas, *Suspende(E)*, *Reassume(E)*, *Aborta(E)* e *Termina(E)*, semelhantes às definidas para eventos de exibição, conforme descrito na Seção 4.1.3. Elas controlam o estado do evento *E* e os valores de seus atributos. *Suspende* altera o estado do evento para *suspenso*, caso ele se encontre no estado *ocorrendo*; caso contrário nada acontece. *Reassume* retorna do estado *suspenso* para o estado *ocorrendo*. *Aborta* modifica o estado do evento para *preparado* (passando pelo estado *abortado*), caso esteja no estado *ocorrendo* ou *suspenso*, incrementa o valor do atributo *ocorrências* e torna zero o valor do atributo repetições do evento. *Termina* altera o estado do evento para *preparado*, caso esteja no estado *ocorrendo* ou *suspenso*, incrementa o número de ocorrências do evento e decrementa o seu número de repetições. Ao término de uma atribuição, através da ação *Termina*, o atributo recebe o valor final, especificado na ação de atribuição. No caso da ação *Aborta*, esse valor é um valor intermediário entre o valor inicial e final especificados, da duração da atribuição, antes de ser abortada.

4.3.1.1 Atribuições simultâneas

Cada atribuição especifica um intervalo de tempo, durante o qual o valor de um atributo é modificado. Um documento pode conter várias atribuições de valores a um mesmo atributo. Os intervalos determinados por cada uma dessas atribuições podem ter interseções, resultando em conflitos na alteração do valor do atributo.

A solução adotada[10] pelo modelo para resolver conflitos entre atribuições simultâneas define que a execução de uma atribuição se sobrepõe à atribuição corrente. Essa solução foi escolhida por sua simplicidade.

A Figura 4.9 ilustra um exemplo de resolução de conflitos. Nele, uma atribuição, que determinaria a variação do valor de um atributo qualquer de 20 a 100 no tempo determinado pelo intervalo [0, t_2], é interrompida por uma nova atribuição, que determina uma variação do valor do mesmo atributo de 50 a 0 no tempo determinado pelo intervalo [t_1, t_3]. Deve-se observar que a primeira atribuição é interrompida abruptamente, sendo o valor do atributo substituído pelo valor inicial da segunda atribuição.

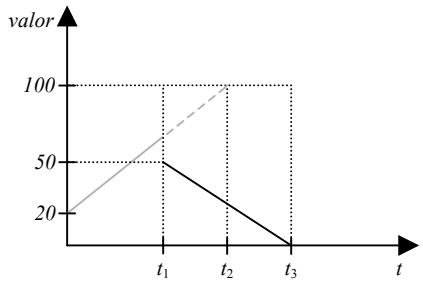


Figura 4.9 – Resolução de conflitos na ocorrência de atribuições simultâneas.

As próximas seções apresentam a especificação de relações entre atributos de objetos segundo a definição de evento de atribuição.

4.3.2 Relações espaciais

A definição genérica de relação espacial feita na Seção 3.5.1 pode ser aplicada diretamente na construção de relações espaciais entre nós, objetos de representação, de um documento NCM.

Como mencionado, uma relação espacial representa uma restrição entre os valores de propriedades dos objetos envolvidos. O modelo NCM permite a definição de restrições entre objetos através de elos de restrição (ver Seção 4.1.3).

Entretanto, o elo de restrição NCM não se adequa à definição de relações espaciais. Devido a sua ação "simultânea", ele é mais apropriado à criação de restrições temporais.Um exemplo de seu uso é a definição de uma restrição entre dois objetos, determinando que apresentação de ambos seja iniciada simultaneamente.

---

[10] Outras soluções podem definir que uma atribuição em execução não deva ser sobreposta. Cada nova atribuição deve ser enfileirada e executada posteriormente, segundo algum critério.

A restrição definida por uma relação espacial determina que a expressão envolvendo os atributos dos objetos envolvidos deve ser válida no momento em que eles estiverem sendo apresentados. Isso demanda a correção desses valores sempre que algum deles for modificado. Por exemplo, uma relação de centralização determina que os objetos envolvidos devem estar centralizados quanto estiverem sendo apresentados. Sempre que o posicionamento de um objeto for modificado, os outros objetos que estiverem sendo apresentados precisam ter seu posicionamento modificado para que a expressão de centralização continue válida.

Restrições espaciais podem ser criadas através de elos causais, envolvendo eventos de atribuição. Como mencionado na Seção 4.3.1, a atribuição de valores a um atributo ocorre somente se o objeto que o contém estiver sendo apresentado.

Esta seção apresenta diversos **elos espaciais** usados na construção das relações espaciais sugeridas na Seção 3.5.1. Como será exemplificado, no final da seção, relações espaciais podem envolver atributos completamente diferentes. Este trabalho, entretanto, atém-se à definição de alguns tipos básicos de relação.

Os elos apresentados nesta seção consideram uma definição comum dos objetos envolvidos. São sempre considerados dois nós (objetos de representação), *A* e *B*, contidos (direta ou indiretamente) em um mesmo nó de contexto, *C*. A partir desses nós, podem ser definidos os seguintes pontos terminais:

- A_{B} = <(*C*, …, *A*), *bottom*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *bottom* do objeto *A*;
- A_{PB} = <(*C*, …, *A*), *previousBottom*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *previousBottom* do objeto *A*;
- A_{H} = <(*C*, …, *A*), *height*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *height* do objeto *A*;
- A_{PH} = <(*C*, …, *A*), *previousHeight*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *previousHeight* do objeto *A*;
- A_{L} = <(*C*, …, *A*), *left*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *left* do objeto *A*;
- A_{PL} = <(*C*, …, *A*), *previousLeft*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *previousLeft* do objeto *A*;

- A_R = <(*C*, …, *A*), *right*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *right* do objeto *A*;
- A_{PR} = <(*C*, …, *A*), *previousRight*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *previousRight* do objeto *A*;
- A_T = <(*C*, …, *A*), *top*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *top* do objeto *A*;
- A_{PT} = <(*C*, …, *A*), *previousTop*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *previousTop* do objeto *A*;
- A_W = <(*C*, …, *A*), *width*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *width* do objeto *A*;
- A_{PW} = <(*C*, …, *A*), *previousWidth*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *previousWidth* do objeto *A*;
- B_B = <(*C*, …, *B*), *bottom*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *bottom* do objeto *B*;
- B_{PB} = <(*C*, …, *B*), *previousBottom*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *previousBottom* do objeto *B*;
- B_H = <(*C*, …, *B*), *height*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *height* do objeto *B*;
- B_{PH} = <(*C*, …, *B*), *previousHeight*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *previousHeight* do objeto *B*;
- B_L = <(*C*, …, *B*), *left*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *left* do objeto *B*;
- B_{PL} = <(*C*, …, *B*), *previousLeft*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *previousLeft* do objeto *B*;
- B_R = <(*C*, …, *B*), *right*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *right* do objeto *B*;
- B_{PR} = <(*C*, …, *B*), *previousRight*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *previousRight* do objeto *B*;

- B_T = <(*C*, …, *B*), *top*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *top* do objeto *B*;
- B_{PT} = <(*C*, …, *B*), *previousTop*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *previousTop* do objeto *B*;
- B_W = <(*C*, …, *B*), *width*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *width* do objeto *B*; e
- B_{PW} = <(*C*, …, *B*), *previouswidth*, *atribuição*, ∅>, que representa a atribuição de valores ao atributo *previouswidth* do objeto *B*.

Todos os pontos terminais são definidos através de quatro propriedades: a primeira identifica uma seqüência de nós, onde o último deles é o objeto a ser considerado; a segunda referencia um atributo do objeto; a terceira, o tipo do evento associado; e a quarta, um conjunto de descritores vazio.

Deve-se observar que os pontos terminais que referenciam eventos de atribuição foram definidos aos pares. Para cada atributo espacial (*left*, *right*, etc.), foi definido outro atributo que guarda o seu valor anterior (*previousLeft*, *previousRight*, etc.). Esses atributos serão muito úteis na definição das relações espaciais a seguir, pois muitas delas são mantidas levando-se em consideração a diferença entre o valor atual e o anterior dos atributos espaciais.

Todas as relações apresentadas a seguir consideram que os eventos de atribuição do objetos envolvidos encontram-se habilitados (atributo *hab*, ver Seção 4.3.1) e que esses objetos estejam sendo exibidos na mesma janela.[11]

As subseções a seguir apresentam a construção de relações espaciais através de elos NCM.

4.3.2.1 Relações de defasagem

A primeira classe de relações a ser apresentada é a das relações de defasagem. Como apresentado na Seção 3.2.1, uma relação de defasagem entre dois objetos *A* e *B* é definida de forma que uma das bordas (esquerda, direita, superior ou inferior) do objeto *B* esteja distanciada de um valor qualquer da mesma borda do objeto *A*.

---

[11] Deve-se lembrar que, segundo o modelo espacial apresentado no Capítulo 3, somente objetos associados a regiões que estejam sendo apresentadas em uma mesma janela podem ser relacionados espacialmente. Objetos de janelas diferentes não podem ter qualquer relação espacial.

O valor da defasagem de *B* com relação a *A* deve ser definido inicialmente pelos descritores utilizados na apresentação dos objetos. Por exemplo, caso seja definido que o objeto *A* deva estar defasado à esquerda de *B* de 50 unidades, essa diferença deve ser refletida no valor do atributo *left* de *A*, de forma que ele seja 50 unidades maior que o valor do atributo *left* de *B*. Por exemplo, caso o valor inicial do atributo *left* de *B* seja 100, o valor do atributo *left* de A deve ter o valor inicial de 150.

A Tabela 4.1 apresenta a especificação de elos espaciais para a representação de relações de defasagem entre os objetos *A* e *B*, onde *B* encontra-se defasado de *A*. Cada relação definida na tabela é composta por dois elos, um para cada sentido da relação.

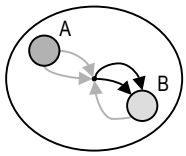
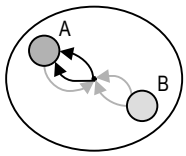
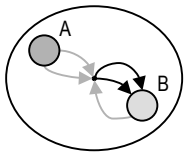
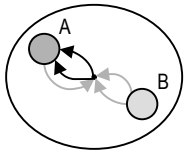
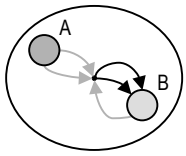
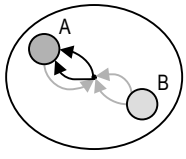
Os elos são utilizados para a manutenção da relação, corrigindo os valores das propriedades dos objetos envolvidos. Ao corrigir o valor de um atributo, o elo também atualiza o seu valor anterior. Por exemplo, a relação de defasagem à esquerda, no sentido *A* → *B*, define um elo para corrigir o valor do atributo *left* do objeto B, caso o posicionamento do objeto A seja alterado. Antes disso, o valor do atributo *previousLeft* é corrigido. O uso de ambos os elos define uma relação bilateral, onde ambos os objetos são componentes de referência (ver Seção 3.5.1), enquanto que o uso de somente um elo especifica uma relação unilateral, onde o objeto origem do elo é o componente de referência.

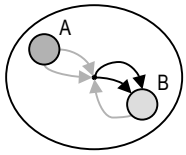
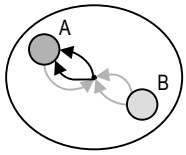
Deve-se observar que em todas as ações de atribuição especificadas, o valor inicial, a duração da variação e o número de repetições recebem o valor nulo. Dessa forma, os valores dos atributos são atualizados instantaneamente, recebendo o valor final especificado apenas uma vez.

Os elos que definem as outras relações de defasagem são especificados da mesma forma que os da relação de defasagem à esquerda.

Por fim, cada definição de elo é acompanhada de uma figura. As setas de cor cinza representam os pontos terminais de origem; as de cor preta, os pontos terminais de destino; e o ponto, o objeto ponto de encontro.

Tabela 4.1 – Relações de defasagem definidas através de elos NCM.

Defasagem à esquerda	**Elo (A → B)** Conjunto dos pontos terminais de origem = $\{A_L, A_{PL}, B_L\}$ Conjunto dos pontos terminais de destino = $\{B_L, B_{PL}\}$ Ponto de encontro: **Condição: (<*estado(A_L) = preparado, estado(A_L) = ocorrendo*>** **Ação: *Atribui valor(B_{PL},(nulo, valor(B_L)), nulo, nulo) →*** ***→ Atribui valor(B_L,(nulo, valor(B_L) + (valor(A_L) – valor(A_{PL}))), nulo, nulo)***  **Elo (B → A)** Conjunto dos pontos terminais de origem = $\{B_L, B_{PL}, A_L\}$ Conjunto dos pontos terminais de destino = $\{A_L, A_{PL}\}$ Ponto de encontro: **Condição: (<*estado(B_L) = preparado, estado(B_L) = ocorrendo*>** **Ação: *Atribui valor(A_{PL},(nulo, valor(A_L)), nulo, nulo) →*** ***→ Atribui valor(A_L,(nulo, valor(A_L) + (valor(B_L) – valor(B_{PL}))), nulo, nulo)*** 
Defasagem à direita	**Elo (A → B)** Conjunto dos pontos terminais de origem = $\{A_R, A_{PR}, B_R\}$ Conjunto dos pontos terminais de destino = $\{B_R, B_{PR}\}$ Ponto de encontro: **Condição: (<*estado(A_R) = preparado, estado(A_R) = ocorrendo*>** **Ação: *Atribui valor(B_{PR},(nulo, valor(B_R)), nulo, nulo) →*** ***→ Atribui valor(B_R,(nulo, valor(B_R) + (valor(A_R) – valor(A_{PR}))), nulo, nulo)***  **Elo (B → A)** Conjunto dos pontos terminais de origem = $\{B_R, B_{PR}, A_R\}$ Conjunto dos pontos terminais de destino = $\{A_R, A_{PR}\}$ Ponto de encontro: **Condição: (<*estado(B_R) = preparado, estado(B_R) = ocorrendo*>** **Ação: *Atribui valor(A_{PR},(nulo, valor(A_R)), nulo, nulo) →*** ***→ Atribui valor(A_R,(nulo, valor(A_R) + (valor(B_R) – valor(B_{PR}))), nulo, nulo)*** 
Defasagem em relação ao topo	**Elo (A → B)** Conjunto dos pontos terminais de origem = $\{A_T, A_{PT}, B_T\}$ Conjunto dos pontos terminais de destino = $\{B_T, B_{PT}\}$ Ponto de encontro: **Condição: (<*estado(A_T) = preparado, estado(A_T) = ocorrendo*>** **Ação: *Atribui valor(B_{PT},(nulo, valor(B_T)), nulo, nulo) →*** ***→ Atribui valor(B_T,(nulo, valor(B_T) + (valor(A_T) – valor(A_{PT}))), nulo, nulo)***  **Elo (B → A)** Conjunto dos pontos terminais de origem = $\{B_T, B_{PT}, A_T\}$ Conjunto dos pontos terminais de destino = $\{A_T, A_{PT}\}$ Ponto de encontro: **Condição: (<*estado(B_T) = preparado, estado(B_T) = ocorrendo*>** **Ação: *Atribui valor(A_{PT},(nulo, valor(A_T)), nulo, nulo) →*** ***→ Atribui valor(A_T,(nulo, valor(A_T) + (valor(B_T) – valor(B_{PT}))), nulo, nulo)*** 

Defasagem em relação à base	**Elo (A → B)** Conjunto dos pontos terminais de origem = $\{A_B, A_{PB}, B_B\}$ Conjunto dos pontos terminais de destino = $\{B_B, B_{PB}\}$ Ponto de encontro: **Condição: (<*estado(A_B) = preparado, estado(A_B) = ocorrendo*>** **Ação: *Atribui valor(B_{PB},(nulo, valor(B_B)), nulo, nulo) →*** ***→ Atribui valor(B_B,(nulo, valor(B_B) + (valor(A_B) – valor(A_{PB}))), nulo, nulo)***  **Elo (B → A)** Conjunto dos pontos terminais de origem = $\{B_B, B_{PB}, A_B\}$ Conjunto dos pontos terminais de destino = $\{A_B, A_{PB}\}$ Ponto de encontro: **Condição: (<*estado(B_B) = preparado, estado(B_B) = ocorrendo*>** **Ação: *Atribui valor(A_{PB},(nulo, valor(A_B)), nulo, nulo) →*** ***→ Atribui valor(A_B,(nulo, valor(A_B) + (valor(B_B) – valor(B_{PB}))), nulo, nulo)*** 

Como mencionado anteriormente, as correções definidas pelos elos espaciais só ocorrem quando o objeto destino do elo estiver sendo apresentado, ou seja, quando o evento de apresentação da âncora λ do nó estiver no estado *ocorrendo*. Por exemplo, no caso uma relação de defasagem à esquerda entre os objetos *A* e *B*, a ação do elo **A → B** só ocorre se o nó *B* estiver sendo apresentado. O mesmo ocorre no sentido inverso.

Relações de alinhamento, como mencionado anteriormente, são casos particulares de relações de defasagem. Sua especificação, através de elos NCM, é semelhante a das relações de defasagem, bastando que os valores dos atributos usados para a definição da defasagem tenham o mesmo valor inicial. Como mencionado anteriormente, a construção dos elos é independente dos valores dos atributos envolvidos.

4.3.2.2 Relações de espaçamento

As relações de espaçamento também podem ser especificadas através de elos NCM. Uma relação de espaçamento entre dois objetos *A* e *B* é definida de forma que uma das bordas (esquerda, direita, superior ou inferior) do objeto *A* esteja distanciada de um valor qualquer da borda oposta do objeto *B*.

Assim como ocorre nas relações de defasagem, o valor do espaçamento de *B* com relação a *A* é definido inicialmente pelos descritores utilizados na apresentação dos objetos. Por exemplo, pode-se definir que o objeto *B* deva estar espaçado à esquerda de *d* unidades do objeto *A*, conforme ilustrado na Figura 4.10. Essa diferença deve ser refletida nos valores dos atributos de posicionamento dos objetos. Pode-se definir o valor do atributo *right* de *B* seja igual à

soma dos valores do atributos *right* e *width* de *A* e do valor do espaçamento (a Figura 3.6, apresentada na Seção 3.1.2 ilustra a propriedades de posicionamento de uma região).

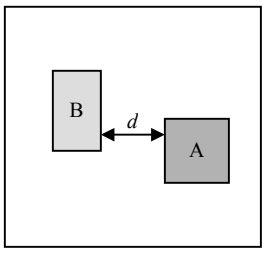


Figura 4.10 – Exemplo de relação de espaçamento ente dois objetos.

A definição dos elos que compõem uma relação de espaçamento difere sensivelmente dos outras relações. Além dos elos definidos para cada um dos sentidos da relação (idênticos aos apresentados anteriormente), como ocorre com as relações de defasagem, é definido um terceiro, obrigatório, não importando se a relação for unilateral ou bilateral.

Esse elo é usado para manter a relação no caso de alteração de uma dimensão (largura ou altura, de acordo com o tipo da relação) de um dos objetos envolvidos. Isso se deve ao fato de que, por *default*, a alteração do valor da largura (ou altura) de um objeto implica na mudança do valor de sua propriedade *right* (ou *bottom*); os valores das propriedades *left* e *top* permanecem os mesmos (ver Figura 4.11).

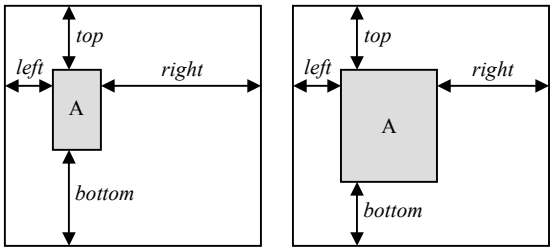
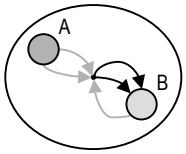
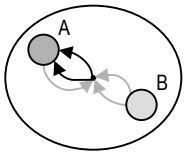
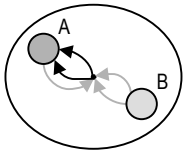


Figura 4.11 – Efeito da alteração das dimensões de um objeto.

O objeto alvo da correção é sempre aquele que estiver à direita ou abaixo durante a relação. Por exemplo, no caso da relação de espaçamento à esquerda (ver Figura 4.10 e Tabela 4.2), o elo altera o posicionamento do objeto *A*, que está à direita, sempre que a largura do objeto *B* é modificada, mantendo o espaçamento entre os objetos.

A definição das relações de espaçamento através de elos NCM é apresentada na Tabela 4.2.

Tabela 4.2 – Relações de espaçamento definidas através de elos NCM.

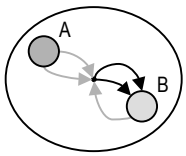
Espaçamento à esquerda	**Elo (A → B)** Conjunto dos pontos terminais de origem = $\{A_L, A_{PL}, B_L\}$ Conjunto dos pontos terminais de destino = $\{B_L, B_{PL}\}$ Ponto de encontro: **Condição: (<*estado(A_L) = preparado, estado(A_L) = ocorrendo*>** **Ação: *Atribui valor(B_{PL},(nulo, valor(B_L)), nulo, nulo) →*** ***→ Atribui valor(B_L,(nulo, valor(B_L) + (valor(A_L) – valor(A_{PL}))), nulo, nulo)***  **Elo (B → A)** Conjunto dos pontos terminais de origem = $\{B_L, B_{PL}, A_L\}$ Conjunto dos pontos terminais de destino = $\{A_L, A_{PL}\}$ Ponto de encontro: **Condição: (<*estado(B_L) = preparado, estado(B_L) = ocorrendo*>** **Ação: *Atribui valor(A_{PL},(nulo, valor(A_L)), nulo, nulo) →*** ***→ Atribui valor(A_L,(nulo, valor(A_L) + (valor(B_L) – valor(B_{PL}))), nulo, nulo)***  **Elo "obrigatório"** Conjunto dos pontos terminais de origem = $\{B_W, B_{PW}, A_L\}$ Conjunto dos pontos terminais de destino = $\{A_L, A_{PL}\}$ Ponto de encontro: **Condição: (<*estado(B_W) = preparado, estado(B_W) = ocorrendo*>** **Ação: *Atribui valor(A_{PL},(nulo, valor(A_L)), nulo, nulo) →*** ***→ Atribui valor(A_L,(nulo, valor(A_L) + (valor(B_W) – valor(B_{PW}))), nulo, nulo)*** 

Espaçamento à direita

Elo (A → B)

Conjunto dos pontos terminais de origem = $\{A_L, A_{PL}, B_L\}$

Conjunto dos pontos terminais de destino = $\{B_L, B_{PL}\}$



Ponto de encontro:

Condição: (<*estado(A_L) = preparado, estado(A_L) = ocorrendo*>

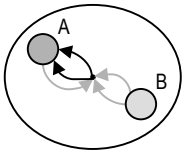
Ação: *Atribui valor(B_{PL},(nulo, valor(B_L)), nulo, nulo)* →

→ *Atribui valor(B_L,(nulo, valor(B_L) + (valor(A_L) – valor(A_{PL}))), nulo, nulo)*

Elo (B → A)

Conjunto dos pontos terminais de origem = $\{B_L, B_{PL}, A_L\}$

Conjunto dos pontos terminais de destino = $\{A_L, A_{PL}\}$



Ponto de encontro:

Condição: (<*estado(B_L) = preparado, estado(B_L) = ocorrendo*>

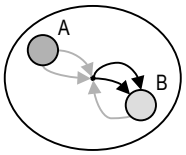
Ação: *Atribui valor(A_{PL},(nulo, valor(A_L)), nulo, nulo)* →

→ *Atribui valor(A_L,(nulo, valor(A_L) + (valor(B_L) – valor(B_{PL}))), nulo, nulo)*

Elo "obrigatório"

Conjunto dos pontos terminais de origem = $\{A_W, A_{PW}, B_L\}$

Conjunto dos pontos terminais de destino = $\{B_L, B_{PL}\}$



Ponto de encontro:

Condição: (<*estado(A_W) = preparado, estado(A_W) = ocorrendo*>

Ação: *Atribui valor(B_{PL},(nulo, valor(B_L)), nulo, nulo)* →

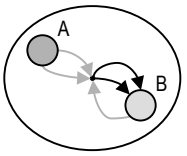
→ *Atribui valor(B_L,(nulo, valor(B_L) + (valor(A_W) – valor(A_{PW}))), nulo, nulo)*

Espaçamento em relação ao topo

Elo (A → B)

Conjunto dos pontos terminais de origem = $\{A_T, A_{PT}, B_T\}$

Conjunto dos pontos terminais de destino = $\{B_T, B_{PT}\}$



Ponto de encontro:

Condição: (<*estado(A_T) = preparado, estado(A_T) = ocorrendo*>

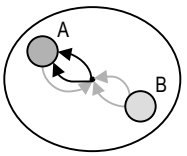
Ação: *Atribui valor(B_{PT},(nulo, valor(B_T)), nulo, nulo)* →

→ *Atribui valor(B_T,(nulo, valor(B_T) + (valor(A_T) – valor(A_{PT}))), nulo, nulo)*

Elo (B → A)

Conjunto dos pontos terminais de origem = $\{B_T, B_{PT}, A_T\}$

Conjunto dos pontos terminais de destino = $\{A_T, A_{PT}\}$



Ponto de encontro:

Condição: (<*estado(B_T) = preparado, estado(B_T) = ocorrendo*>

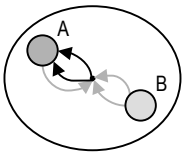
Ação: *Atribui valor(A_{PT},(nulo, valor(A_T)), nulo, nulo)* →

→ *Atribui valor(A_T,(nulo, valor(A_T) + (valor(B_T) – valor(B_{PT}))), nulo, nulo)*

Elo "obrigatório"

Conjunto dos pontos terminais de origem = $\{B_H, B_{PH}, A_T\}$

Conjunto dos pontos terminais de destino = $\{A_T, A_{PT}\}$

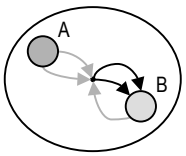
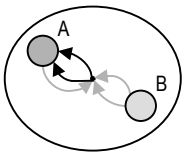
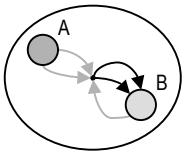


Ponto de encontro:

Condição: (<*estado(B_H) = preparado, estado(B_H) = ocorrendo*>

Ação: *Atribui valor(A_{PT},(nulo, valor(A_T)), nulo, nulo)* →

→ *Atribui valor(A_T,(nulo, valor(A_T) + (valor(B_H) – valor(B_{PH}))), nulo, nulo)*

Espaçamento em relação à base	**Elo (A → B)** Conjunto dos pontos terminais de origem = $\{A_T, A_{PT}, B_T\}$ Conjunto dos pontos terminais de destino = $\{B_T, B_{PT}\}$ Ponto de encontro: **Condição: (<*estado(A_T) = preparado, estado(A_T) = ocorrendo*>** **Ação: *Atribui valor(B_{PT},(nulo, valor(B_T)), nulo, nulo) →*** ***→ Atribui valor(B_T,(nulo, valor(B_T) + (valor(A_T) – valor(A_{PT}))), nulo, nulo)***  **Elo (B → A)** Conjunto dos pontos terminais de origem = $\{B_T, B_{PT}, A_T\}$ Conjunto dos pontos terminais de destino = $\{A_T, A_{PT}\}$ Ponto de encontro: **Condição: (<*estado(B_T) = preparado, estado(B_T) = ocorrendo*>** **Ação: *Atribui valor(A_{PT},(nulo, valor(A_T)), nulo, nulo) →*** ***→ Atribui valor(A_T,(nulo, valor(A_T) + (valor(B_T) – valor(B_{PT}))), nulo, nulo)***  **Elo "obrigatório"** Conjunto dos pontos terminais de origem = $\{A_H, A_{PH}, B_T\}$ Conjunto dos pontos terminais de destino = $\{B_T, B_{PT}\}$ Ponto de encontro: **Condição: (<*estado(A_H) = preparado, estado(A_H) = ocorrendo*>** **Ação: *Atribui valor(B_{PT},(nulo, valor(B_T)), nulo, nulo) →*** ***→ Atribui valor(B_T,(nulo, valor(B_T) + (valor(A_H) – valor(A_{PH}))), nulo, nulo)*** 

Assim como nas relações de defasagem, as ações de atribuição definidas pelos elos das relações de espaçamento só ocorrem se os objetos alvo estiverem sendo apresentados.

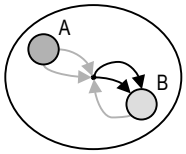
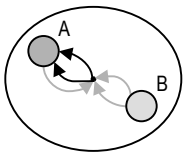
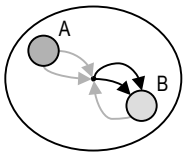
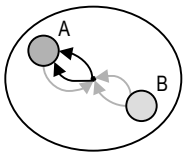
4.3.2.3 Relações de centralização

Relações de centralização também podem ser definidas através de elos NCM. Existem dois tipos de relações de centralização, que determinam que os componentes envolvidos devem estar centralizados horizontalmente ou verticalmente.

Assim como ocorre com as relações de defasagem e espaçamento, a definição das relações de centralização é feita através da definição dos valores iniciais dos atributos de posicionamento dos objetos envolvidos, de forma que eles se encontrem centralizados inicialmente, e da criação de elos para a manutenção da relação em ambos os sentidos da relação.

A Tabela 4.3 apresenta a especificação de elos espaciais para representar relações de centralização entre os objetos *A* e *B*. Eles são os mesmos usados pelas relações de defasagem. Levando-se em conta que os objetos encontram-se centralizados inicialmente, apenas a mesma correção de posicionamento é necessária.

Tabela 4.3 – Relações de centralização definidas através de elos NCM.

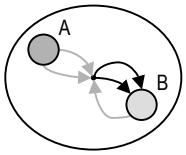
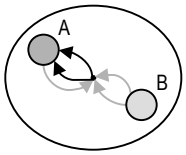
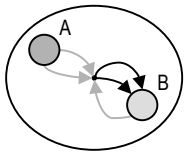
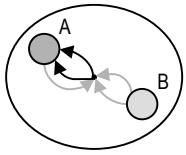
Centralização horizontal	**Elo (A → B)** Conjunto dos pontos terminais de origem = $\{A_T, A_{PT}, B_T\}$ Conjunto dos pontos terminais de destino = $\{B_T, B_{PT}\}$ Ponto de encontro: **Condição: (<*estado(A_T) = preparado, estado(A_T) = ocorrendo*>** **Ação: *Atribui valor(B_{PT},(nulo, valor(B_T)), nulo, nulo) →*** ***→ Atribui valor(B_T,(nulo, valor(B_T) + (valor(A_T) – valor(A_{PT}))), nulo, nulo)***  **Elo (B → A)** Conjunto dos pontos terminais de origem = $\{B_T, B_{PT}, A_T\}$ Conjunto dos pontos terminais de destino = $\{A_T, A_{PT}\}$ Ponto de encontro: **Condição: (<*estado(B_T) = preparado, estado(B_T) = ocorrendo*>** **Ação: *Atribui valor(A_{PT},(nulo, valor(A_T)), nulo, nulo) →*** ***→ Atribui valor(A_T,(nulo, valor(A_T) + (valor(B_T) – valor(B_{PT}))), nulo, nulo)*** 
Centralização vertical	**Elo (A → B)** Conjunto dos pontos terminais de origem = $\{A_L, A_{PL}, B_L\}$ Conjunto dos pontos terminais de destino = $\{B_L, B_{PL}\}$ Ponto de encontro: **Condição: (<*estado(A_L) = preparado, estado(A_L) = ocorrendo*>** **Ação: *Atribui valor(B_{PL},(nulo, valor(B_L)), nulo, nulo) →*** ***→ Atribui valor(B_L,(nulo, valor(B_L) + (valor(A_L) – valor(A_{PL}))), nulo, nulo)***  **Elo (B → A)** Conjunto dos pontos terminais de origem = $\{B_L, B_{PL}, A_L\}$ Conjunto dos pontos terminais de destino = $\{A_L, A_{PL}\}$ Ponto de encontro: **Condição: (<*estado(B_L) = preparado, estado(B_L) = ocorrendo*>** **Ação: *Atribui valor(A_{PL},(nulo, valor(A_L)), nulo, nulo) →*** ***→ Atribui valor(A_L,(nulo, valor(A_L) + (valor(B_L) – valor(B_{PL}))), nulo, nulo)*** 

4.3.2.4 Relações de dimensionamento

Por fim, relações de dimensionamento também podem ser definidas através de elos NCM. Uma relação de dimensionamento entre dois objetos *A* e *B* define uma proporção entre uma de suas dimensões. Existem dois tipos de dimensionamento: horizontal e vertical.

A proporção do dimensionamento de *B* com relação a *A* deve ser definida pelos valores dos atributos de dimensionamento de ambos os objetos. Além disso, assim como na definição dos outros tipos de relação, devem ser criados elos para que a proporção entre os objetos seja mantida. A definição das relações de dimensionamento através de elos NCM é apresentada na Tabela 4.4.

Tabela 4.4 – Relações de dimensionamento definidas através de elos NCM.

Dimensionamento horizontal	**Elo (A → B)** Conjunto dos pontos terminais de origem = $\{A_W, A_{PW}, B_W\}$ Conjunto dos pontos terminais de destino = $\{B_W, B_{PW}\}$ Ponto de encontro: **Condição: (*<estado(A_W) = preparado, estado(A_W) = ocorrendo>*** **Ação: *Atribui valor(B_{PW},(nulo, valor(B_W)), nulo, nulo) →*** ***→ Atribui valor(B_W,(nulo, valor(B_W) · (valor(B_{PW}) / valor(A_{PW}))), nulo, nulo)***  **Elo (B → A)** Conjunto dos pontos terminais de origem = $\{B_W, B_{PW}, A_W\}$ Conjunto dos pontos terminais de destino = $\{A_W, A_{PW}\}$ Ponto de encontro: **Condição: (*<estado(B_W) = preparado, estado(B_W) = ocorrendo>*** **Ação: *Atribui valor(A_{PW},(nulo, valor(A_W)), nulo, nulo) →*** ***→ Atribui valor(A_W,(nulo, valor(B_W) · (valor(A_{PW}) / valor(B_{PW}))), nulo, nulo)*** 
Dimensionamento vertical	**Elo (A → B)** Conjunto dos pontos terminais de origem = $\{A_H, A_{PH}, B_H\}$ Conjunto dos pontos terminais de destino = $\{B_H, B_{PH}\}$ Ponto de encontro: **Condição: (*<estado(A_H) = preparado, estado(A_H) = ocorrendo>*** **Ação: *Atribui valor(B_{PH},(nulo, valor(B_H)), nulo, nulo) →*** ***→ Atribui valor(B_H,(nulo, valor(B_H) · (valor(B_{PH}) / valor(A_{PH}))), nulo, nulo)***  **Elo (B → A)** Conjunto dos pontos terminais de origem = $\{B_H, B_{PH}, A_H\}$ Conjunto dos pontos terminais de destino = $\{A_H, A_{PH}\}$ Ponto de encontro: **Condição: (*<estado(B_H) = preparado, estado(B_H) = ocorrendo>*** **Ação: *Atribui valor(A_{PH},(nulo, valor(A_H)), nulo, nulo) →*** ***→ Atribui valor(A_H,(nulo, valor(B_H) · (valor(A_{PH}) / valor(B_{PH}))), nulo, nulo)*** 

Como mencionado no início desta seção, pode-se estabelecer outros tipos de relações espaciais. Melhor ainda, pode-se estabelecer qualquer tipo de relação entre os atributos de nós de um documento. Por exemplo, relações de dimensionamento entre atributos de dimensões diferentes de dois objetos poderiam ser facilmente definidas de forma semelhante às relações de dimensionamento apresentadas, apenas substituindo-se os eventos envolvidos.

Um bom exemplo seria a definição de uma relação entre um atributo de posicionamento de um nó (uma imagem estática que representasse um botão) com o volume (atributo

`soundLevel`) de um nó de áudio. O primeiro objeto poderia servir de controlador do volume de apresentação do segundo.

A próxima seção apresenta a definição de relações entre objetos usando tanto atributos espaciais como temporais. Assim como as relações definidas aqui, essas relações espaço-temporais também utilizam elos causais e eventos de atribuição para a sua especificação.

4.3.3 Relações espaço-temporais

Relações espaço-temporais (Seção 3.5.2) podem ser usadas na definição da apresentação de documentos hipermídia.

No NCM, relações espaço-temporais são definidas através da construção de elos causais entre objetos de um documento. Esses elos podem envolver todos os tipos de eventos previstos no modelo, em especial o evento de atribuição na definição dos valores de atributos espaciais dos objetos envolvidos.

A forma de construção de relações espaço-temporais no NCM é descrita a seguir. São definidos três tipos de relações espaço-temporais. Cada um deles contém alguns exemplos de implementação.

a) Relação espacial cuja validade depende de outros objetos

O autor de um documento pode especificar relações espaciais que, ao contrário das relações espaciais comuns, apresentadas na Seção 3.5.1, não devam permanecer válidas durante toda a apresentação dos objetos envolvidos. A validade de uma relação espacial pode ser condicionada à ocorrência de algum evento durante a apresentação do documento ou ao valor de uma propriedade de algum objeto.

Pode-se definir relações espaciais que só passem a valer quando algum evento ocorrer, e que posam ser alteradas devido à ocorrência de outros eventos. Exemplos de eventos são a seleção de algum objeto pelo usuário ou o início/fim da apresentação de um objeto.

O início ou o fim da validade de uma relação também pode ser condicionado ao valor de alguma propriedade do documento, como o número de ocorrências da apresentação de algum objeto. Obviamente, eventos podem ser combinados para indicar o início e o fim da validade de uma relação. Um caso especial desse tipo de relação é uma que define uma relação espacial entre dois objetos que deve ser mantida válida se, e somente se, um terceiro objeto estiver sendo apresentado.

Na Tabela 4.5 são apresentados alguns exemplos de relações espaço-temporais entre nós NCM. Em todos eles, são definidas relações espaciais cuja validade é determinada por eventos ou atributos existentes no documento. Todos os exemplos levam em consideração o documento definido na Figura 4.12. Nele é definido um nó de contexto *C*, composto de outros três nós, *A*, *B* e *D*, que podem ser tanto de conteúdo como de contexto.

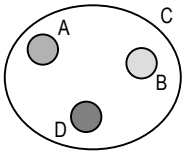


Figura 4.12 – Exemplo de documento NCM.

Assim como na Seção 4.3.2, cabe definir alguns pontos terminais que serão utilizados nos exemplos a serem apresentados adiante. Eles são relacionados abaixo:

- A_E = <*(C, A), λ, exibição*>;
- A_T = <*(C, A), top, atribuição*>;
- A_W = <*(C, A), width, atribuição*>;
- A_D = <*(C, A), distance, atribuição*>;
- B_E = <*(C, B), λ, exibição*>;
- B_T = <*(C, B), top, atribuição*>;
- B_W = <*(C, B), width, atribuição*>;
- B_D = <*(C, B), distance, atribuição*>; e
- D_E = <*(C, B), λ, exibição*>.

Todos os pontos terminais definidos são indiferentes a respeito do tipo de nó envolvido, podendo ser usados tanto para nós de conteúdo como de contexto.

Tabela 4.5 – Exemplos de relações espaço-temporais no NCM (a).

Exemplo 1: Os nós *A* e *B* devem permanecer alinhados ao topo enquanto *D* estiver sendo apresentado.

Assim como feito para as relações espaciais apresentadas na Seção 4.3.2, a definição dessa relação é feita através de dois elos, uma para cada sentido da relação, que devem manter os valores do alinhamento definidos nos descritores usados pelos objetos *A* e *B*.

Além desses elos, são definidos mais dois, que habilitam e inibem a atribuição de valores aos atributos *top* dos objetos *A* e *B* de acordo com o estado do evento de exibição da âncora λ do objeto *D*. Os elos são apresentados abaixo:

Elo (A → B)

Conjunto dos pontos terminais de origem = $\{A_T, A_{PT}, B_T\}$

Conjunto dos pontos terminais de destino = $\{B_T, B_{PT}\}$

Ponto de encontro:

Condição: $(<estado(A_T) = preparado, estado(A_T) = ocorrendo>$

Ação: $Atribui\ valor(B_{PT},(\ nulo,\ valor(B_T)),\ nulo,\ nulo) \rightarrow$

$\rightarrow Atribui\ valor(B_T,(\ nulo,\ valor(B_T) + (valor(A_T) - valor(A_{PT}))),\ nulo,\ nulo)$

Elo (B → A)

Conjunto dos pontos terminais de origem = $\{B_T, B_{PT}, A_T\}$

Conjunto dos pontos terminais de destino = $\{A_T, A_{PT}\}$

Ponto de encontro:

Condição: $(<estado(B_T) = preparado, estado(B_T) = ocorrendo>$

Ação: $Atribui\ valor(A_{PT},(\ nulo,\ valor(A_T)),\ nulo,\ nulo) \rightarrow$

$\rightarrow Atribui\ valor(A_T,(\ nulo,\ valor(A_T) + (valor(B_T) - valor(B_{PT}))),\ nulo,\ nulo)$

Elo de habilitação:

Conjunto dos pontos terminais de origem = $\{D_E\}$

Conjunto dos pontos terminais de destino = $\{A_T, B_T\}$

Ponto de encontro:

Condição: $<estado(D_E) = preparado, estado(D_E) = ocorrendo>$

Ação: $Habilita(A_T)\ |\ Habilita(B_T)$

Elo de inibição:

Conjunto dos pontos terminais de origem = $\{D_E\}$

Conjunto dos pontos terminais de destino = $\{B_T\}$

Ponto de encontro:

Condição: $<estado(D_E) \neq preparando, estado(D_E) = preparado>$

Ação: $Inibe(A_T)\ |\ Inibe(B_T)$

Exemplo 2: Os nós *A* e *B* devem permanecer com suas larguras iguais a partir do instante em que o nó *B* terminar sua exibição.

A relação de dimensionamento é definida do modo usual, através de dois elos responsáveis pela manutenção dos valores das larguras dos objetos envolvidos (ver Tabela 4.4).

Além dos elos correspondentes à relação espacial, é definido um terceiro, correspondente à habilitação (através da ação *Habilita*) dos eventos de atribuição envolvidos na construção dos elos espaciais. Os elos são apresentados abaixo.

Elo (A → B)

Conjunto dos pontos terminais de origem = $\{A_W, A_{PW}, B_W\}$

Conjunto dos pontos terminais de destino = $\{B_W, B_{PW}\}$

Ponto de encontro:

Condição: (<*estado(A_W) = preparado, estado(A_W) = ocorrendo*>

Ação: *Atribui valor(B_{PW},(nulo, valor(B_W)), nulo, nulo)* →

→ *Atribui valor(B_W,(nulo, valor(B_W) · (valor(B_{PW}) / valor(A_{PW}))), nulo, nulo)*

Elo (B → A)

Conjunto dos pontos terminais de origem = $\{B_W, B_{PW}, A_W\}$

Conjunto dos pontos terminais de destino = $\{A_W, A_{PW}\}$

Ponto de encontro:

Condição: (<*estado(B_W) = preparado, estado(B_W) = ocorrendo*>

Ação: *Atribui valor(A_{PW},(nulo, valor(A_W)), nulo, nulo)* →

→ *Atribui valor(A_W,(nulo, valor(B_W) · (valor(A_{PW}) / valor(B_{PW}))), nulo, nulo)*

Elo de habilitação dos eventos de atribuição:

Conjunto dos pontos terminais de origem = $\{D_E\}$

Conjunto dos pontos terminais de destino = $\{A_W, B_W\}$

Ponto de encontro:

Condição: <*estado(D_E) = ocorrendo, estado(D_E) = preparado*>

Ação: *Habilita(A_W) | Habilita(B_W)*

Exemplo 3: O nó *B* deve permanecer à esquerda do nó *A* até que o número de ocorrências do evento de apresentação de *D* atinja o valor 10.

A relação de espaçamento (ver Tabela 4.2), é definida por dois elos, um para cada sentido da relação, de forma que o espaçamento entre os objetos, definido pelos valores de suas propriedades de posicionamento nos descritores usados para a sua apresentação, seja mantido. Além disso, como visto anteriormente, é necessário um terceiro elo, que corrige o posicionamento de *A* sempre que a largura de *B* é modificada.

Além desses elos, é definido um outro, que avalia o número de ocorrências da exibição de *D* e quando este valor atinge um máximo de 10, inibe simultaneamente os eventos de atribuição dos nós *A* e *B*, através da ação *Inibe*. Obviamente, é considerado que os eventos de atribuição encontram-se habilitados inicialmente.

Os elos são definidos da seguinte forma:

Elo (A → B)

Conjunto dos pontos terminais de origem = $\{A_L, A_{PL}, B_L\}$

Conjunto dos pontos terminais de destino = $\{B_L, B_{PL}\}$

Ponto de encontro:

Condição: (<*estado(A_L) = preparado, estado(A_L) = ocorrendo*>

Ação: *Atribui valor(B_{PL},(nulo, valor(B_L)), nulo, nulo)* →

→ ***Atribui valor(B_L,(nulo, valor(B_L) + (valor(A_L) – valor(A_{PL}))), nulo, nulo)***

Elo (B → A)

Conjunto dos pontos terminais de origem = {B_L, B_{PL}, A_L}

Conjunto dos pontos terminais de destino = {A_L, A_{PL}}

Ponto de encontro:

Condição: (<*estado(B_L) = preparado, estado(B_L) = ocorrendo*>

Ação: *Atribui valor(A_{PL},(nulo, valor(A_L)), nulo, nulo)* →

→ ***Atribui valor(A_L,(nulo, valor(A_L) + (valor(B_L) – valor(B_{PL}))), nulo, nulo)***

Elo "obrigatório"

Conjunto dos pontos terminais de origem = {B_W, B_{PW}, A_L}

Conjunto dos pontos terminais de destino = {A_L, A_{PL}}

Ponto de encontro:

Condição: (<*estado(B_W) = preparado, estado(B_W) = ocorrendo*>

Ação: *Atribui valor(A_{PL},(nulo, valor(A_L)), nulo, nulo)* →

→ ***Atribui valor(A_L,(nulo, valor(A_L) + (valor(B_W) – valor(B_{PW}))), nulo, nulo)***

Elo de inibição dos eventos de atribuição:

Conjunto dos pontos terminais de origem = {D_E}

Conjunto dos pontos terminais de destino = {A_W, B_W}

Ponto de encontro:

Condição: <*VERDADE, ocorrências(D_E) =10*>

Ação: *Inibe(A_L) | Inibe (B_L)*

b) Configuração espacial de um ou mais objetos dependente de eventos ou propriedades de outros objetos

Nesse tipo de relação, os valores de atributos espaciais de objetos do documento são alterados em resposta a eventos ocorridos durante a sua apresentação ou à alteração dos valores de outros atributos. Esse tipo de relação é um pouco diferente do apresentado em a). Aqui não são especificadas relações espaciais, embora possam ser definidas relações onde o valor de algum atributo espacial de um objeto seja derivado do valor de uma atributo de outro objeto, representando um alinhamento instantâneo entre eles, por exemplo.

A Tabela 4.6 apresenta um exemplo desse tipo de relação espaço-temporal. Ele leva em consideração o documento exemplo da Figura 4.12, os pontos terminais definidos em a) e o ponto terminal A_L = <*(C, …, A), left, atribuição*>, referente ao atributo *left* do objeto *A*. Ainda, considera-se que os eventos de atribuição usados no exemplo encontram-se habilitados.

Tabela 4.6 – Exemplo de relação espaço-temporal no NCM (b).

Exemplo 1:	O nó *A* deve ser colocado na posição (15, 200) e *D* deve ser alinhado à direita de *A* assim que a exibição de *B* termine. É criado apenas um elo que avalia o estado do evento de exibição do nó *B*. Caso este nó tenha encerrado sua apresentação, o nó *A* é posicionado na posição determinada e *D* é alinhado à sua direita. A ação do elo é uma ação composta, onde as atribuições são feitas simultaneamente. O elo é apresentado a seguir:
Elo:	
Conjunto dos pontos terminais de origem = $\{A_L, A_W, B_E, D_W\}$	
Conjunto dos pontos terminais de destino = $\{A_L, A_T, D_L\}$	
Ponto de encontro:	
Condição: $\langle estado(B_E) = ocorrendo, estado(B_E) = preparado \rangle$	
Ação: $(Atribui\ valor(A_L, (nulo, 15), nulo, nulo) \mid Atribui\ valor(A_T, (nulo, 200), nulo, nulo)) \rightarrow Atribui\ valor(D_L, (nulo, valor(A_L) + valor(A_W) - valor(D_W)), nulo, nulo)$	

c) Ocorrência de algum evento ou valor de alguma propriedade do documento depende da configuração espacial de um ou mais objetos

Esse tipo de relação é o inverso do apresentado em b). Aqui, o estado de eventos e os valores de atributos de um documento podem ser alterados com base no posicionamento e nos valores das dimensões de um ou mais objetos.

A Tabela 4.7 apresenta exemplos desse tipo de relação espaço-temporal, através de elos NCM. Neles, o documento exemplo da Figura 4.12 é usado, assim como os eventos definidos em a) e b).

Tabela 4.7 – Exemplos de relações espaço-temporais no NCM (c).

Exemplo 1:	A exibição do nó *D* é iniciada quando *A* estiver na posição (50,125). Além disso, deve ser especificado que a exibição de *D* deve ser repetida por duas vezes. É definido o elo causal abaixo, que avalia a posição do nó *A* e, caso ele seja a desejada, inicia a exibição de *D*, através da ação *Inicia*.
Elo:	
Conjunto dos pontos terminais de origem = $\{A_L, A_T\}$	
Conjunto dos pontos terminais de destino = $\{D_E\}$	
Ponto de encontro:	
Condição: $\langle VERDADE, valor(A_L) = 50 \rangle \wedge \langle VERDADE, valor(A_T) = 125 \rangle$	
Ação: $Inicia(D_E, 2)$	
Exemplo 2:	Caso os nós *A* e *B* estejam centralizados horizontalmente, a exibição do nó *D* é concluída. É definido um único elo causal que verifica se os valores dos atributos *left* e *width* dos objetos *A* e *B* possuem valores que determinam que eles estejam centralizados. Em caso positivo, a exibição do nó *D* é concluída, através da ação *Termina*.
Elo:	
Conjunto dos pontos terminais de origem = $\{A_L, A_W, B_L, B_W\}$	

Conjunto dos pontos terminais de destino = $\{D_E\}$
Ponto de encontro:
Condição*: <VERDADE, valor(A_L) + valor(A_W)/2= valor(B_L) + valor(B_W)/2>*
Ação: *Termina(D_E)*

Os exemplos apresentados nesta seção mostraram algumas relações espaço-temporais que podem ser definidas em documentos NCM. Outros exemplos mais complexos podem ser gerados combinando-se vários eventos e atributos na definição das condições de elos espaço-temporais.

Na seção seguinte é apresentado o uso de eventos de atribuição na definição de animações de nós NCM durante sua exibição.

4.3.4 Animações

Um evento de atribuição, como apresentado na Seção 4.3.1, permite a atribuição de uma tupla composta de um valor inicial e de um valor final a um atributo. A duração dessa variação é determinada por uma função de custo, também especificada em uma ação de atribuição.

Essa forma de atribuição de valores permite a definição de mudanças de comportamento de apresentação de nós de um documento e em particular, a criação de animações, através da alteração dos valores dos atributos espaciais de nós no decorrer de sua apresentação.

Através de uma ou mais atribuições, pode-se:

- determinar uma variação uniforme de um atributo espacial com uma duração definida;
- pode-se acumular valores intermediários, entre repetições de atribuições, assim como em SMIL (ver Seção 2.1);
- determinar o "congelamento" do valor de um atributo, assim como em SMIL;

A seguir são apresentados vários exemplos de definição de animações de atributos de objetos de representação. Todos eles usam o documento da Figura 4.12 como base. Como poderá ser observado, todos os exemplos definem eventos de atribuição associados a atributos dos objetos de representação envolvidos na animação.

O Exemplo 1 apresenta um caso particular de animação, onde o valor de um atributo deve permanecer constante durante um tempo definido. Esse tipo de animação é semelhante ao definido pelo elemento `set` do conjunto de módulos de animação da linguagem SMIL, versão 2.0 (ver Seção 2.1).

Exemplo 1: O nó *A* deve permanecer na posição (50,100) durante um tempo determinado.

São definidos dois pontos terminais que referenciam os atributos *top* e *left* do objeto de representação:

$$E_T = <(C, A), top, atribuição, \varnothing> \text{ e } E_L = <(C, A), left, atribuição, \varnothing>.$$

Considerando que a atribuição de valores aos atributos *top* e *left* esteja habilitada, para que a animação ocorra, a seguinte ação composta deve ser executada:

$$(Atribui\ valor(E_T, (50, 50), 0, nulo) \mid Atribui\ valor(E_L, (100, 100), 0, nulo)) \rightarrow$$

$$\rightarrow (Inibe(E_T) \mid Inibe(E_L)) \rightarrow ((Habilita(E_T) \mid Habilita\ (E_L)) \circledR\ t)$$

Essa ação define uma atribuição simultânea de valores aos atributos *top* e *left*, seguida da inibição de atribuições de valores a esses atributos e da habilitação de atribuições aos mesmos atributos, após um retardo especificado por *t*.

Os dois exemplos seguintes apresentam formas simples de animação. O Exemplo 2 define um movimento uniforme de um nó, especificando seus valores inicial e final. O Exemplo 3 apresenta a definição de uma animação relativa.

Exemplo 2: O nó *A* deve se mover durante um período determinado de forma que o valor de sua propriedade *top* varie uniformemente, com relação ao tempo, de 10 a 100 *pixels*.

É definido o ponto terminal $E = <(C, A), top, atribuição, \varnothing>$ que ser refere ao atributo *top* do objeto. Para que a animação ocorra, deve ser executado o método *Atribui valor(E, (10, 100), d, nulo)*, onde *d* corresponde à duração da animação.

A Figura 4.13 apresenta como seria a variação do valor do atributo *top* com relação ao tempo, caso a duração escolhida pelo formatador fosse de 5 segundos.

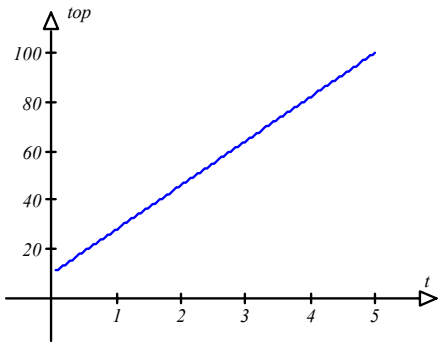


Figura 4.13 – Variação do valor do atributo *top* no Exemplo 2.

Exemplo 3: O nó *A* deve ser movido da posição (50,50) para a posição (100,100) em um tempo determinado.

São definidos dois pontos terminais que referenciam os atributos *top* e *left* do objeto de representação:

$$E_T = <(C, A), top, atribuição, \varnothing> \text{ e } E_L = <(C, A), left, atribuição, \varnothing>$$

Para que a animação ocorra, é executada a seguinte ação composta:

$$Atribui\ valor(E_T, (50, 100), d, nulo) \mid Atribui\ valor(E_L, (nulo, top), nulo , nulo)$$

Essa ação determina que o valor do atributo *top* deve variar uniformemente entre 50 e 100 durante o tempo *d*. O valor do atributo *left* é determinado pelo valor de *top*. Essa animação corresponde à apresentada na Figura 4.7, na página 91.

Outra função (uma função quadrática, por exemplo) poderia ter sido usada para determinar o valor do atributo *top*, definindo uma animação diferente do posicionamento do objeto.

O Exemplo 4 apresenta um tipo especial de animação onde o valor de um objeto deve ser mantido constante até que alguma ação externa ocorra. Este "congelamento" ocorre da mesma forma que na definição de animações em SMIL (ver Seção 2.1), quando o atributo `fill` definido em SMIL recebe o valor *freeze*.

Exemplo 4: O valor da largura do nó *A* deve permanecer com valor igual a 100 por tempo indeterminado.

O seguinte ponto terminal é criado para controlar a atribuição de valores ao atributo *width*:

$$E_W = <(C, A), width, atribuição, \varnothing>$$

A seguinte ação composta deve ser executada (ela considera que a alteração de valor do atributo encontra-se habilitada):

$$Atribui\ valor(E_W,\ (nulo,\ 100),\ 0,\ nulo) \rightarrow Inibe(E_T)$$

Essa ação determina a alteração instantânea do valor do atributo seguida da inibição de mudança do valor do atributo. Assim o valor do atributo permanece "congelado" até que atribuições passem a ser permitidas novamente, por exemplo, através de outra atribuição.

Ações de atribuição permitem que animações sejam definidas de forma cumulativa ou não, como em SMIL (ver Seção 2.1). Os dois próximos exemplos mostram como ambos os tipos de animação podem ser definidos.

O Exemplo 5 apresenta a animação não-cumulativa de um atributo. No Exemplo 6 a mesma animação é feita de forma cumulativa. A definição de animações cumulativas é a mesma empregada em SMIL (ver Seção 2.1).

Exemplo 5: O nó *A* deve ter o valor de sua largura variando uniformemente de menos cem unidades do seu valor corrente durante 10 segundos. Isso deve se repetir por 3 vezes.

Define-se o seguinte ponto terminal, para controlar a atribuição de valores ao atributo *width* do objeto:

$$E_W = \langle (C, \ldots, A),\ width,\ atribuição,\ \varnothing \rangle$$

Para que a animação ocorra, é executa a seguinte ação:

$$Atribui\ valor(E_W, (\varnothing,\ \varnothing - 100,\ nulo),\ d,\ 3),$$

onde:

- o valor inicial corresponde ao valor corrente do atributo;
- o valor final corresponde ao valor corrente do atributo diminuído de 100 unidades;
- a duração do evento define uma função cujo valor ótimo é de 10 segundos; e
- o número de repetições do evento é igual a 3.

O símbolo ∅ é substituído pelo valor corrente do atributo quando a ação de atribuição é executada. Considerando que o valor do atributo *width* antes da aplicação da atribuição seja de 300 unidades, o valor inicial é substituído por 300 e o valor final, por 200. Assim, o que ocorre é uma variação do valor de 300 até 200 na duração determinada, três vezes

seguidas. Como pode ser observado, o valor do atributo não é acumulado entre as repetições da animação.

A variação do valor do atributo é apresentada na Figura 4.14, considerando um valor corrente de 300 unidades.

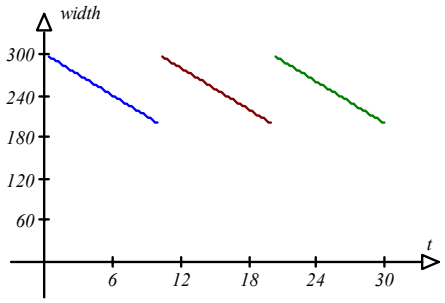


Figura 4.14 – Exemplo de animação não-cumulativa do atributo *width*.

Exemplo 6: O nó *A* deve ser animado de forma idêntica ao definido no Exemplo 5. A única diferença é que o valor final de cada repetição deve corresponder ao valor inicial da repetição seguinte, resultando em uma animação cumulativa.

A ação de atribuição é definida de forma diferente do exemplo anterior. Deve-se executar a ação composta:

$$Atribui\ valor(E_W, (\varnothing, \varnothing - 100, nulo), d, nulo) \rightarrow$$

$$Atribui\ valor(E_W, (\varnothing, \varnothing - 100, nulo), d, nulo) \rightarrow$$

$$Atribui\ valor(E_W, (\varnothing, \varnothing - 100, nulo), d, nulo)$$

A execução seqüencial resulta no acúmulo dos valores finais de cada uma das repetições para a repetição seguinte, pois, a cada execução, o valor corrente do atributo é substituído.

A variação do valor do atributo é apresentada na Figura 4.15, considerando um valor inicial de 300 *pixels*. Nota-se que a animação é definida de forma que o valor final de cada interação corresponde ao valor inicia da próxima, de forma que a animação ocorra de forma cumulativa.

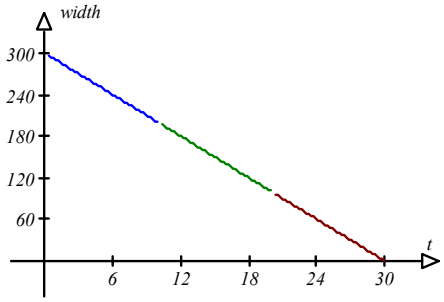


Figura 4.15 – Exemplo de animação cumulativa do atributo *width*.

Através de exemplos, foram apresentadas formas simples de animação que podem ser criadas através de eventos de atribuição. Pode-se especificar animações mais complexas através da composição das diversas situações apresentadas. Ainda, pode-se criar seqüências de animações com a atribuição de diversas funções em seqüência para um mesmo evento de atribuição ou animações de mais de um objeto através do uso associado de animações e relações espaciais. Outros atributos, como a cor de um objeto, também podem ser animados através de eventos de atribuição.

4.4 SUMÁRIO

Neste capítulo foi apresentada a aplicação do modelo espacial definido no Capítulo 3 ao Modelo de Contextos Aninhados (NCM). Nele foi descrito como estruturas de apresentação, relações espaciais e relações espaço-temporais podem ser definidas através de entidades NCM.

A aplicação do modelo espacial ao modelo NCM incorreu em algumas modificações nas entidades deste modelo. Em particular, o evento de atribuição passou a ter uma duração, recebendo mais alguns atributos e tendo sua máquina de estados alterada. Ainda, novos atributos, correspondentes às propriedades das entidades do modelo espacial, tiveram que ser adicionadas à entidade descritor.

Finalmente, com a modificação do evento de atribuição do modelo NCM, foi possível também definir melhor as mudanças de comportamento de objetos de mídia. Pôde-se, então, criar animações dos valores dos atributos de objetos, o que não era previsto no modelo espacial apresentado no capítulo anterior.

5 O EDITOR E *BROWSER* DE SINCRONISMO DO SISTEMA HYPERPROP

O sistema HyperProp [Soares00a, Soares99 e Rodrigues98] é composto de uma ambiente de autoria de documentos hipermídia definidos no modelo de contextos aninhados.

O ambiente de autoria do sistema HyperProp é composto de um *browser* de estrutura, que permite a definição da estrutura lógica dos documentos, e de um editor e *browser* de sincronismo (EBS) que permite definir a sincronização da apresentação do documento através de duas **visões**, espacial e temporal. O processo de autoria de um documento é feito de forma integrada em ambos os *browsers*.

O *browser* de sincronismo, em particular, também é usado para acompanhar a apresentação de documentos, fornecendo visões dos cenários temporal e espacial da exibição dos objetos que os compõem.

Este capítulo encontra-se organizado como se segue. A Seção 5.1 apresenta o *browser* de estrutura do sistema HyperProp. A Seção 5.2 apresenta o *browser* de sincronismo. Por último, a Seção 5.3 faz um breve resumo do capítulo.

5.1 O *BROWSER* DE ESTRUTURA

O *browser* de estrutura [Soares00a, Pinto00 e Saade96] é usado para definir a estrutura lógica de um documento NCM. Ele permite basicamente criar documentos, adicionar nós (de contexto ou de conteúdo) e criar elos entre os nós.

Depois de criado um nó, é possível editar suas âncoras e definir o conjunto de descritores disponíveis para a sua apresentação. A edição dos elos compõe a definição de seus pontos terminais de origem e de destino e de seu ponto de encontro.

A Figura 5.1 apresenta o *browser* de estrutura. Nele é exibida a estrutura de um documento NCM. O documento é um nó de contexto, composto de cinco composições (*comp1*, *comp2*, *comp3*, *comp4* e *comp5*) e de nós terminais (*B*, *D* e *H*). A Figura 5.2 apresenta o mesmo documento, com todos as composições expandidas.

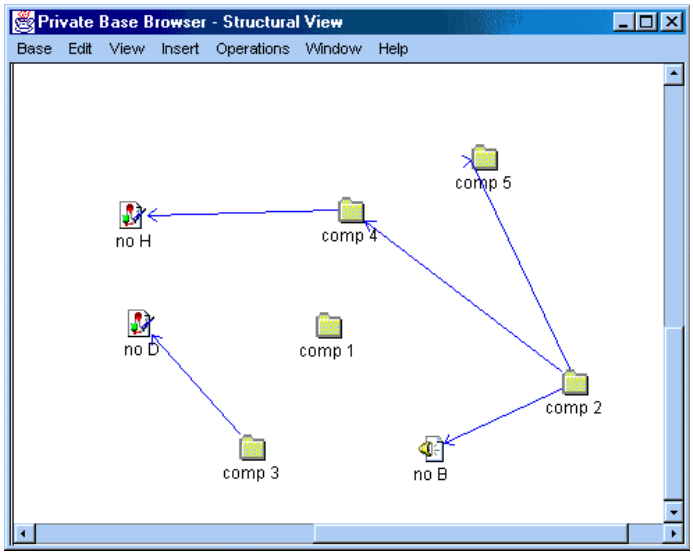


Figura 5.1 – Um documento no *browser* de estrutura do sistema HyperProp.

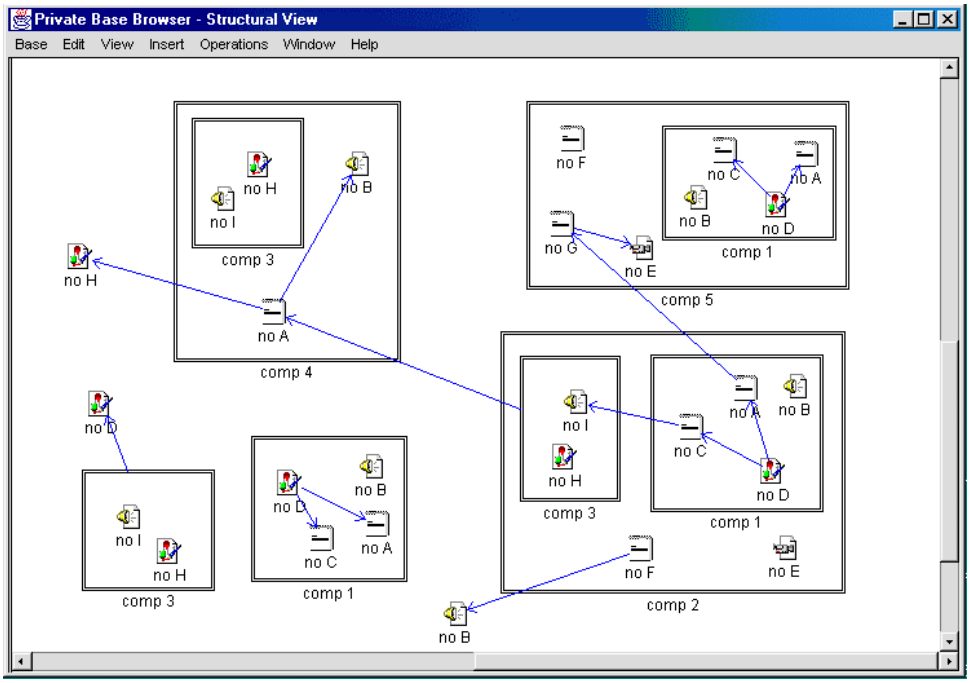


Figura 5.2 – Todos os níveis de um documento no *browser* de estrutura do sistema HyperProp.

Uma característica interessante desse *browser* é a existência de um mecanismo de filtragem, conhecido como “olho de peixe” [Saade96 e Pinto00], que facilita o processo de autoria e

navegação de um documento hipermídia, filtrando a exibição do documento, de acordo com o nó selecionado (com o foco).

Mais informações a respeito do implementação atual do *browser* de estrutura do sistema HyperProp podem ser encontradas em [Pinto00].

O processo de autoria de um documento NCM é completado através da edição das relações temporais e espaciais entre os nós que o compõem. Essas relações são criadas no *browser* de sincronismo, apresentado na próxima seção.

5.2 O EBS

O *browser* de sincronismo é uma das ferramentas do sistema HyperProp usadas no processo de autoria de documentos NCM. Primeiramente apresentado em [Costa96], ele permite definir tanto a sincronização temporal como a espacial de um documento, através de duas visões (espacial e temporal) distintas, mas diretamente relacionadas. Aqui é apresentada a versão 2.0 do Editor e *Browser* de Sincronismo.

As subseções a seguir apresentam como a sincronização de um documento pode ser definida através das visões temporal e espacial da implementação atual do *browser* de sincronismo.

5.2.1 A Visão Temporal

A visão temporal do *browser* de sincronismo permite a definição da sincronização temporal de um documento hipermídia. Ela apresenta a visualização do progresso da apresentação de um documento através de uma linha de tempo (*time line*). A Figura 5.3 apresenta a visão temporal do *browser* de sincronismo.

A visão temporal possui um eixo de tempo de tempo móvel, definido por uma régua, que permite visualizar todos os componentes de um documento hipermídia que estejam relacionados temporalmente. Esse eixo de tempo divide a apresentação em diversos canais. Cada canal apresenta os nós da visão temporal associados a um mesmo dispositivo de apresentação. Na Figura 5.4, por exemplo, é apresentado o nó *text1* associado ao dispositivo *video*.

A visão temporal possui uma região especial, chamada de região de limbo, onde são posicionados os nós cujo início da apresentação não pode ser determinado. Além disso, é disponibilizada uma barra de tempo móvel, que pode ser posicionada em qualquer instante de

tempo, definindo quais nós devem ser apresentados na visão espacial do *browser* (ver Seção 5.2.2). Na Figura 5.3, essa barra encontra-se na posição de 3,9 s.

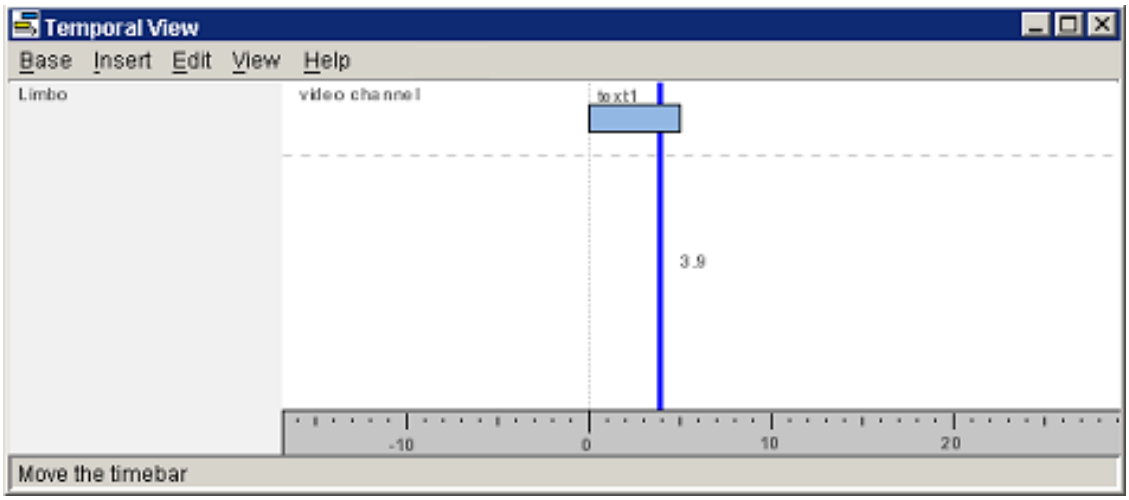


Figura 5.3 – A visão temporal do *browser* de sincronismo do sistema HyperProp.

Os objetos apresentados pela visão temporal são objetos de representação exibidos na forma de retângulos, cujo comprimento é determinado pela duração do evento de apresentação da âncora λ do nó. O posicionamento dos nós é determinado pelos elos de sincronização, levando-se em consideração o posicionamento do **objeto base do sincronismo**. Esse nó é posicionado na origem o eixo de tempo. Por exemplo, na Figura 5.5, o nó *text1* é o objeto base do sincronismo.

Os elos de sincronização (ou *sinc-elos*) definidos entre o nós também são apresentados. Um elo de sincronização é um elo cujos pontos terminais de origem não estão associados a eventos de seleção (ver Seção 4.1.3). Os demais elos são apresentados apenas no *browser* de estrutura.

A inserção de nós na visão temporal do *browser* de sincronismo é feita de forma recursiva. Quando um nó é adicionado à visão temporal ou o objeto base do sincronismo é selecionado, são incluídos os nós ligados diretamente ao nó, de acordo com as seguintes regras:

- se um nó incluído na visão for destino de um elo de sincronização 1:n, o nó de origem desse elo é incluído na visão na posição determinada pelo elo;
- se um nó incluído na visão for origem de um elo de sincronização, todo os nós de destino de seus elos de sincronização cujas condições forem satisfeitas, também são incluídos na visão e posicionados segundo esses elos.

A Figura 5.4 apresenta um exemplo, onde o nó *text1*é definido como base do sincronismo. Os nós *video1*, *text2* e *audio1* são incluído automaticamente, de acordo com as regras acima. Pode-se observar que o posicionamento desses nós é determinado segundo o posicionamento do nó base e as regras definidas pelos elos de sincronização.

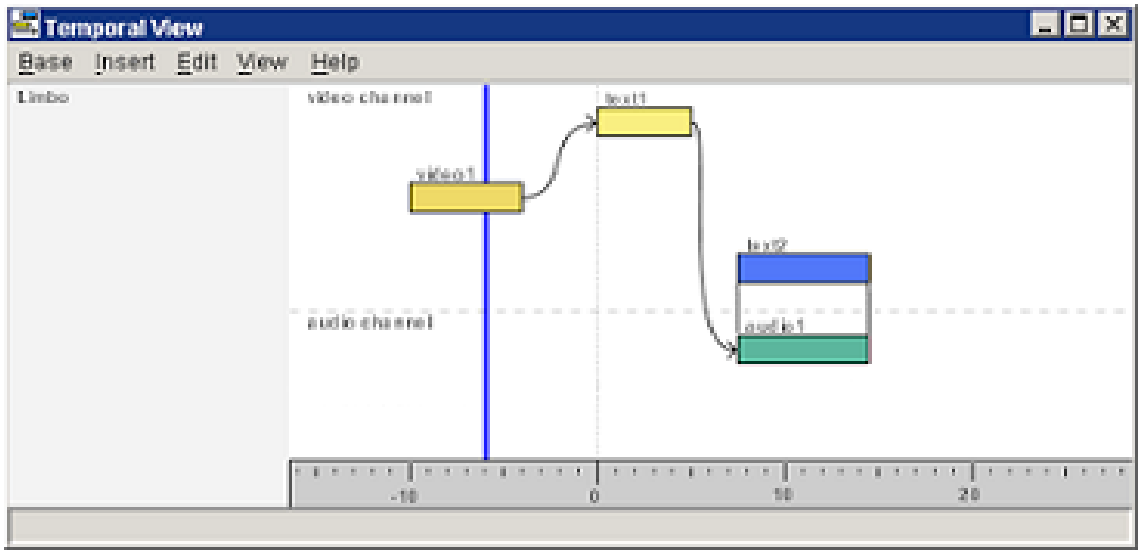


Figura 5.4 – Nós apresentados na visão temporal do *browser* de sincronismo.

Também, pode-se incluir um nó diretamente na visão temporal. Isso é feito de duas maneiras diferentes. Pode-se adicionar um nó e posicioná-lo na origem do eixo de tempo. Nesse caso, é criado um elo de restrição entre o nó adicionado e o nó base do sincronismo, de forma que os dois sejam apresentados simultaneamente. Essa opção é ilustrada na Figura 5.5, onde o nó *image1*, associado ao dispositivo *handheld* é adicionado.

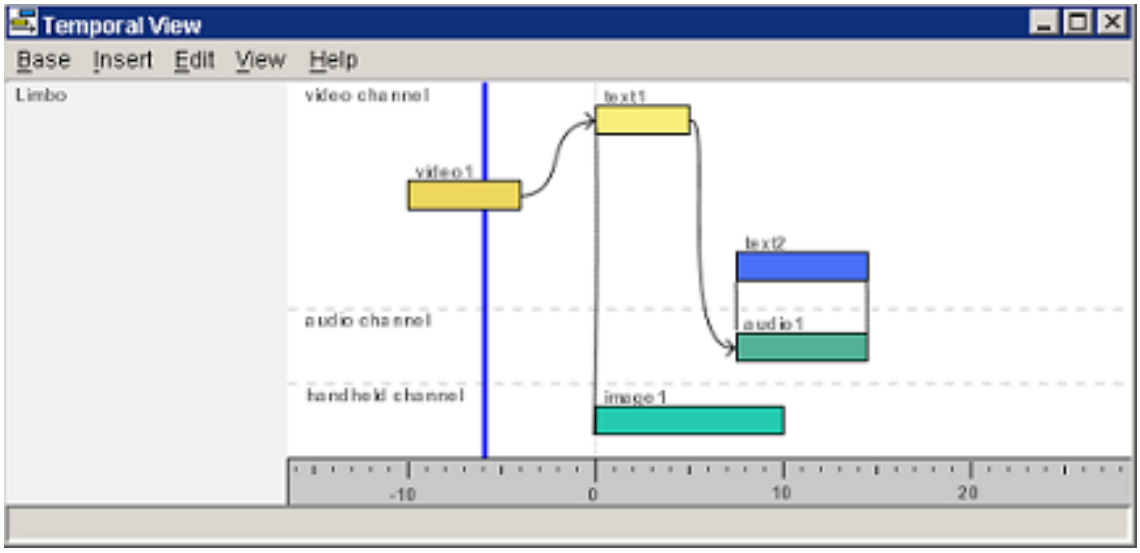


Figura 5.5 – Exemplo de inclusão de um nó na visão temporal.

Um nó também pode ser adicionado à região de limbo da visão temporal, como ilustrado na Figura 5.6. Quando isso ocorre, define-se que o início da apresentação do nó não pode ser determinado com antecedência. Nós posicionados na região de limbo podem ser posteriormente relacionados a outros nós da visão temporal, como será visto na próxima seção.

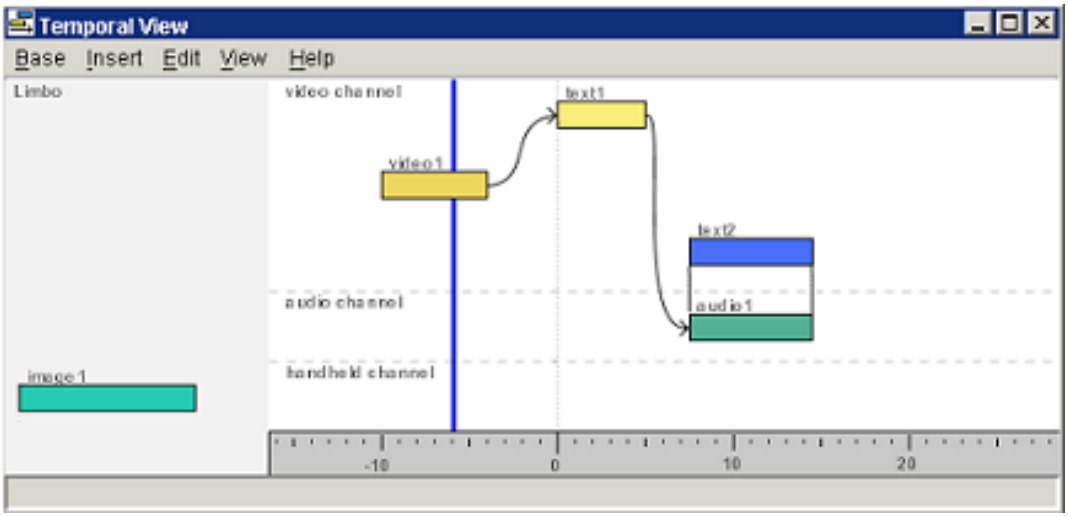


Figura 5.6 – Exemplo de inclusão de um nó na região de limbo da visão temporal.

Na visão temporal é possível criar elos temporais entre eventos de objetos de um documento de forma a definir o encadeamento temporal da apresentação. Pode-se, também, definir algumas relações temporais de alto nível [Allen83].

A visão temporal possui um conjunto de *templates* para a criação de elos de sincronização, permitindo, inclusive, a adição de relações temporais de alto nível, como *starts*, *finishes* e *equals* e *meets*.

Em linhas gerais, a implementação atual da visão espacial do EBS corresponde à apresentada em [Costa96]. Extensões serão propostas na Seção 6.3.

Detalhes sobre a hierarquia de classes utilizadas na implementação da versão 2.0 da visão temporal do EBS podem ser encontradas em [Moura01b]

5.2.2 A Visão Espacial

A visão espacial permite definir a sincronização espacial de um documento NCM e acompanhá-la durante uma apresentação. Nela é possível:

- criar estruturas de apresentação compostas de janelas e regiões. Isso é feito através de uma ferramenta de edição de estruturas de apresentação. Essas estruturas podem ser usadas na definição dos descritores de nós NCM na visão espacial do *browser*; e
- definir relações espaciais entre os objetos de um documento, usando-se um conjunto básicos de relações espaciais, sugeridas na Seção 3.5.1.

O editor de estruturas de apresentação permite criar estruturas de apresentação segundo o modelo apresentado no Capítulo 3. Pode-ser criar estruturas compostas de janelas e hierarquias de regiões e definir relações espaciais entre elas.

A Figura 5.7 apresenta o editor de estruturas de apresentação do *browser* de sincronismo do sistema Hyperprop. Nela é apresentado um exemplo de estrutura de apresentação composta por duas janelas, ambas associadas ao dispositivo *video1*. A primeira das regiões contém somente uma região, enquanto a segunda, outras três.

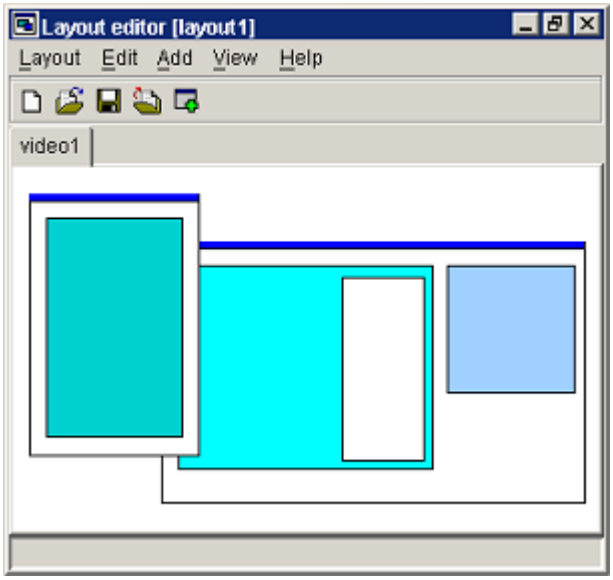


Figura 5.7 – O editor de estruturas de apresentação.

A criação de janelas e regiões é feita através da definição de suas características espaciais. A Figura 5.8 apresenta as caixa de diálogo usadas na criação e edição das propriedades de janelas e regiões de uma estrutura de apresentação.

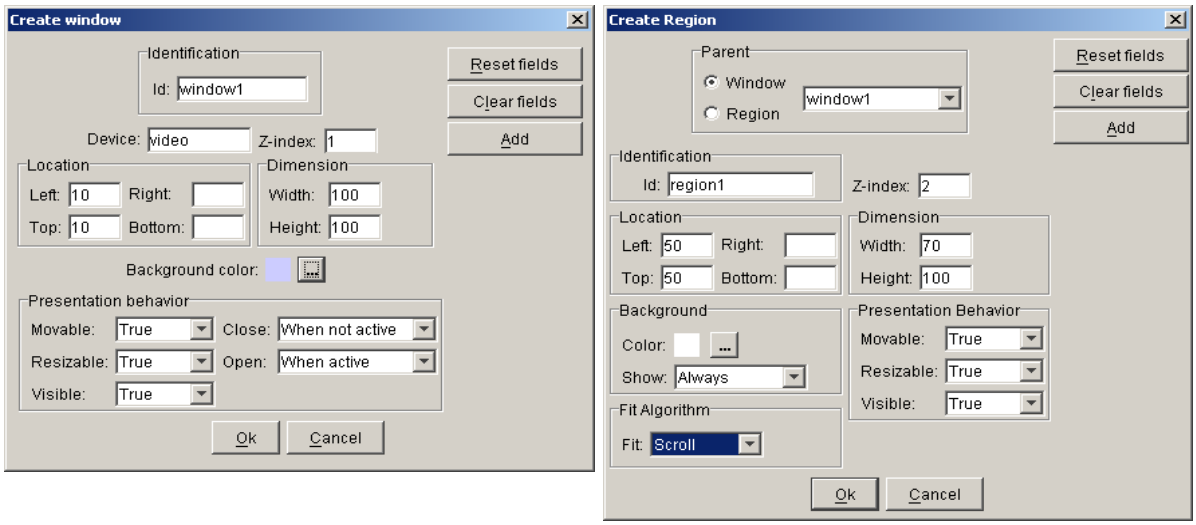


Figura 5.8 – Edição das propriedades de janelas e regiões de uma estrutura de apresentação.

Depois de criadas janelas e regiões, é possível definir relações espaciais entre elas. Isso é feito através de um *wizard*, onde o usuário define o tipo de relação (entre janelas, entre regiões ou entre regiões e o componente que os contém) a ser criada, os componentes da estrutura que farão parte da relação, o tipo da relação e, opcionalmente, valores a serem associados a cada um dos componentes. A Figura 5.9 apresenta a seqüência de janelas do *wizard* usado na criação de relações espaciais.

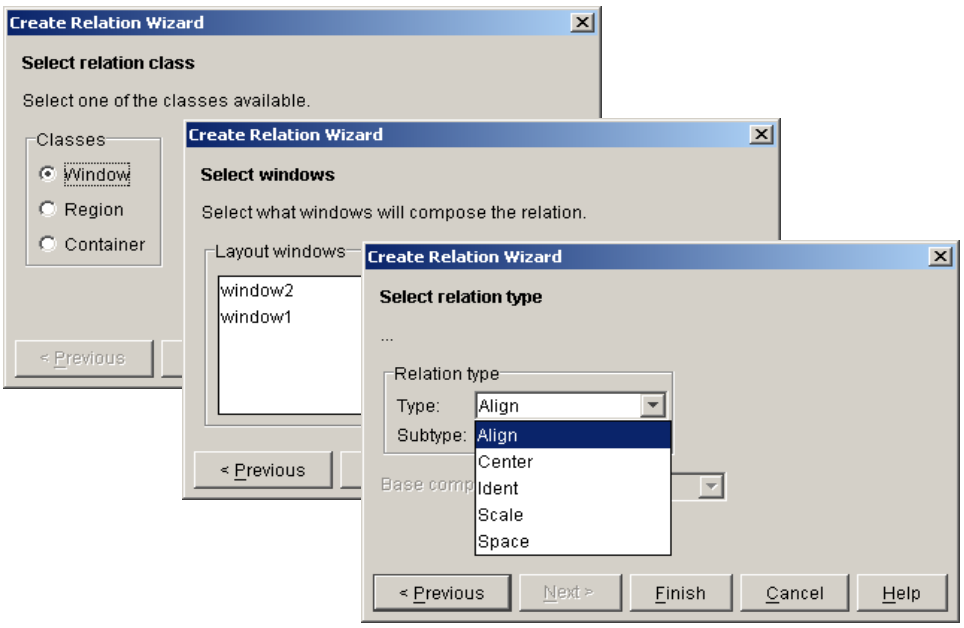


Figura 5.9 – *Wizard* para a criação de relações espaciais em uma estrutura de apresentação.

Os componentes de uma estrutura de apresentação são implementados segundo um conjunto de classes que definem janelas (*HNCMLayoutWindow*), regiões (*HNCMLayoutRegion*) e relações espaciais (*HNCMLayoutRelation*). O editor é implementado segundo o padrão arquitetural Model-View-Controller [Buschmann97]. Cada objeto (janela ou região) é associado a um objeto que representa sua representação visual (visão), apresentada na interface gráfica (representada por um retângulo, como visto na Figura 5.7), e a um controlador que coordena a alteração dos valores das propriedade do objeto.

As relações espaciais definidas no editor são mantidas, enquanto a estrutura de apresentação estiver sendo editada. Mais detalhes a respeito do editor de estruturas de apresentação podem ser encontrados em [Moura01b e Moura01a].

As estruturas de apresentação definidas no editor de estruturas de apresentação podem ser usadas na definição da estrutura de apresentação usada por um documento. Isso é feito através da especificação das propriedades dos descritores utilizados pelo nós durante sua apresentação, como visto na Seção 4.2.1. Cada descritor deve identificar tanto a janela como a região a ser utilizada para a apresentação do objeto associado. A Figura 5.10 apresenta a definição das propriedades espaciais (janela e região) de um descritor de nós de texto.

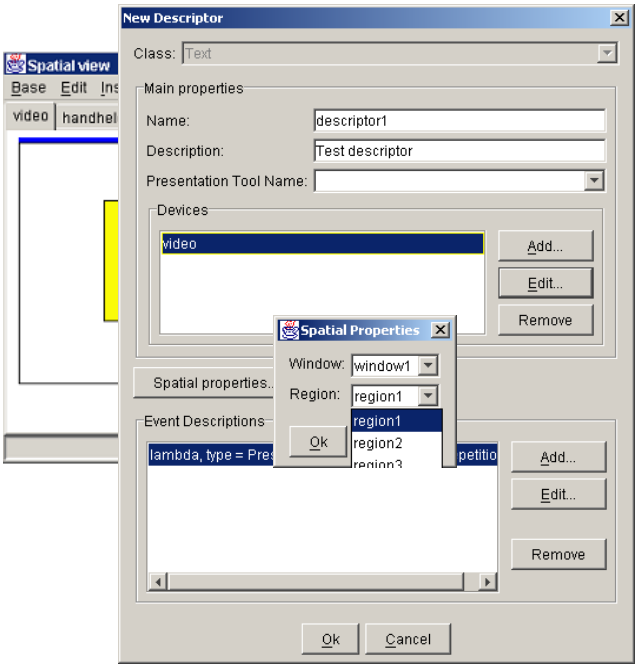


Figura 5.10 – Edição das propriedades espaciais de um descritor.

Na visão espacial também é possível definir relações espaciais entre os objetos de um documento. A visão espacial dispõe de um conjunto básico de relações espaciais. Elas são implementadas através dos elos espaciais apresentadas na Seção 4.3.2. Ao usuário basta selecionar os componentes e definir os valores correspondentes à relação. Os elos são criados automaticamente, assim como são definidos os valores das propriedades espaciais dos objetos envolvidos.

A Figura 5.11 ilustra a criação de relações espaciais entre objetos. Nela é apresentada a criação de uma relação (bilateral) de alinhamento à esquerda entre dois objetos (a). Eles são selecionados (b) e é aplicada a relação (c). O posicionamento de ambos é corrigido e é criado o elo espacial (assim como explicado em 4.3.2.1). A partir desse momento, quando os dois objeto estiverem sendo apresentados, a relação é mantida, corrigido-se o posicionamento dos objetos de forma que eles permaneçam alinhados (d). É importante observar que os objetos podem se movimentar livremente na vertical.

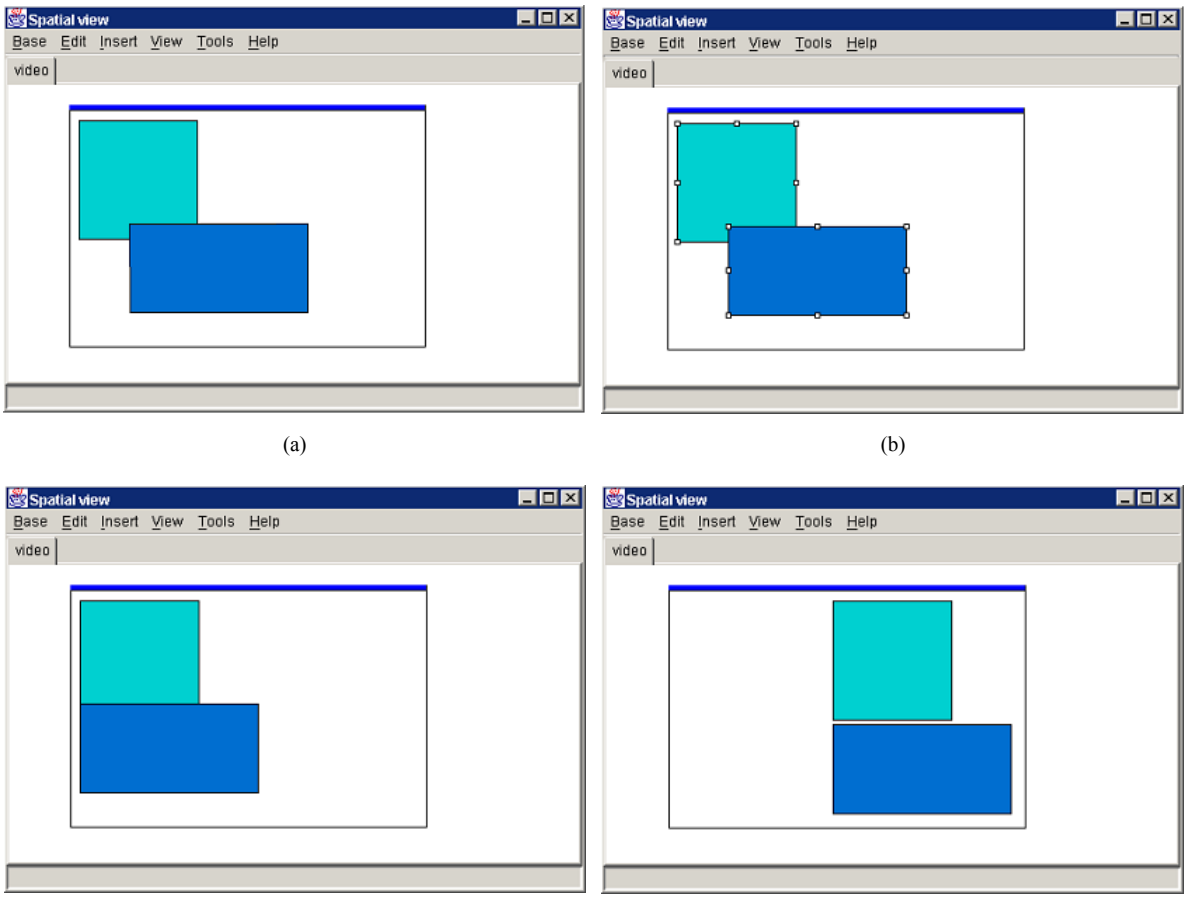


Figura 5.11 – Criação de uma relação espacial entre objetos na visão espacial.

Também é permitido criar novas relações espaciais e adicioná-las ao conjunto de relações do *browser*.

A visão espacial pode ser usada em conjunto com a visão temporal do *browser*. Como mencionado na Seção 5.2.1, a visão temporal possui uma barra de tempo móvel, cujo posicionamento identifica os nós que devem ser apresentados na visão espacial. A Figura 5.12 apresenta um exemplo de uso da barra de tempo móvel na visão temporal. Nela, a barra encontra-se posicionada no instante 4s, quando estão sendo apresentados os nós *text1* e *image1*. A Figura 5.13 apresenta a visão espacial, onde os mesmos nós são apresentados com suas características espaciais. Cada uma das abas da janela representa um dos dispositivos de apresentação usados pelo documento naquele instante. Os nós são representados pelas regiões onde são apresentados, contidas em suas janelas respectivas, com as características espaciais dos objetos no instante selecionado na visão temporal do *browser*.

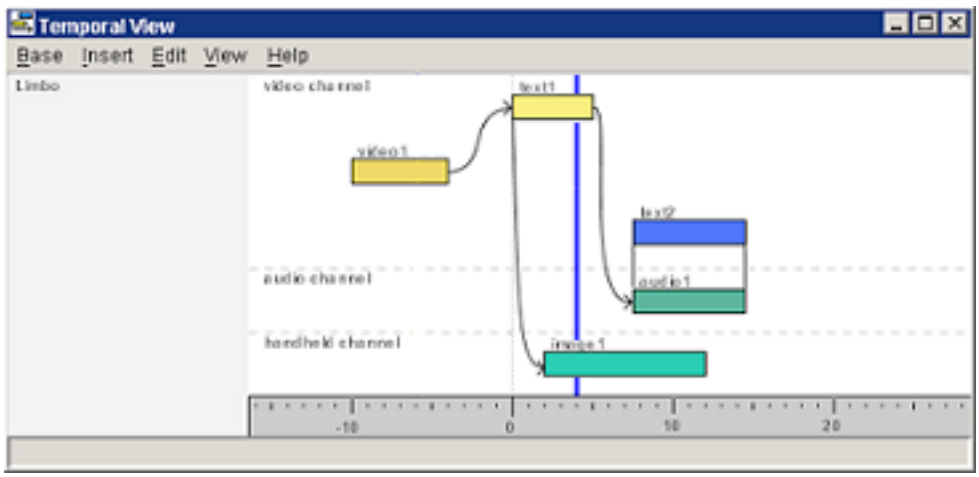


Figura 5.12 – Uso da barra de tempo móvel na visão temporal do *browser* de sincronismo.

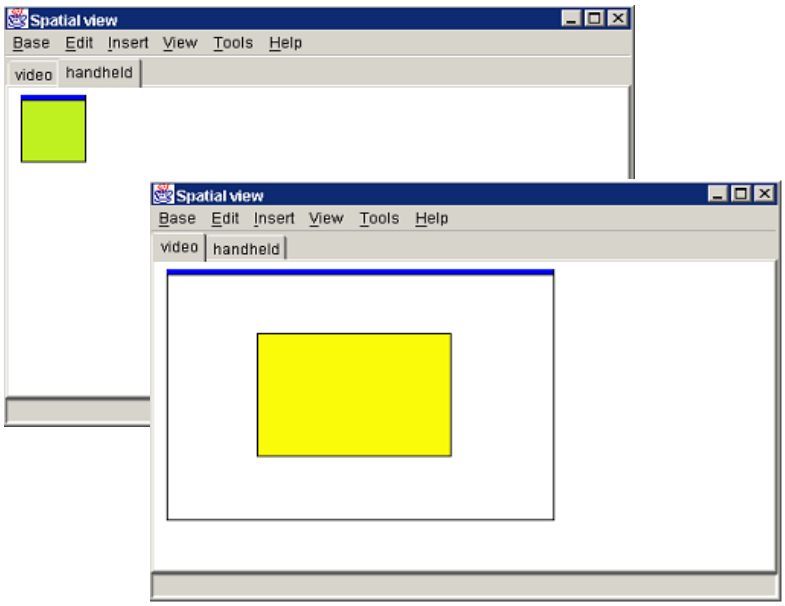


Figura 5.13 – A visão espacial do *browser* de sincronismo do sistema HyperProp.

Informações detalhadas a respeito da modelagem e implementação do *browser* de sincronismo do sistema HyperProp podem ser encontradas em [Moura01b].

5.3 SUMÁRIO

Esse capítulo fez uma apresentação breve do *browser* de sincronismo do sistema HyperProp. Esse *browser* é dividido em duas visões: temporal e espacial. A visão temporal permite definir a sincronização temporal de um documento NCM, através da criação de elos de sincronização entre os objetos que o compõem. A visão espacial permite criar estruturas de apresentação, e

usá-las em documentos NCM, além de permitir a criação de relações espaciais entre seus objetos.

6 CONCLUSÕES

O trabalho aqui realizado procurou definir um modelo para a sincronização espacial de documentos hipermídia e mostrar sua funcionalidade através de sua aplicação em um modelo de documentos já existente. Este capítulo apresenta as principais contribuições do trabalho realizado, fazendo algumas comparações com os trabalhos que o influenciaram.

Este capítulo encontra-se organizado como se segue. A Seção 6.1 apresenta as principais contribuições da dissertação. A Seção 6.2 apresenta uma comparação do trabalho realizado nesta dissertação com os trabalhos relacionados, apresentados no Capítulo 2. Finalmente, a Seção 6.3 apresenta alguns trabalhos futuros.

6.1 CONTRIBUIÇÕES DA DISSERTAÇÃO

As contribuições da dissertação podem ser divididas em três partes:

- a definição de um modelo para a sincronização espacial de documentos hipermídia;
- a aplicação desse modelo ao modelo NCM; e
- a implementação de um ambiente de edição e apresentação da sincronização espacial de documentos NCM.

Elas são apresentadas a seguir.

Definição do modelo espacial

O Capítulo 3 apresentou um modelo que permite a definição da sincronização espacial de documentos hipermídia. Ele é baseado na construção de estruturas de apresentação.

Uma das vantagens da existência de estruturas de apresentação é a definição do formato espacial de um documento independentemente de sua estrutura lógica (vantagem

compartilhada por outros modelos, como será mostrado na Seção 6.2.2). Outra vantagem do modelo é a divisão de apresentações em dispositivos de apresentação diferentes.

As estruturas de apresentação definidas pelo modelo podem conter relações espaciais entre janelas e regiões. O uso de relações espaciais facilita a definição das características espaciais de janelas e regiões, possibilitando a definição dos valores de suas propriedades de posicionamento e dimensão de forma relativa. Aliado ao modelo, foi sugerido um conjunto básico de relações espaciais.

Um subproduto do modelo foi a definição de uma linguagem declarativa (ver Seção 3.3) para definição de estruturas de apresentação. Essa linguagem herda todas as definições feitas pelo modelo espacial.

Além do modelo espacial, o Capítulo 3 definiu em linhas gerais como uma estrutura de apresentação pode ser utilizada por um modelo de documentos hipermídia. Foi definido como deve ser feita a associação dos objetos de mídia de um documento as regiões de uma estrutura de apresentação. Além disso, foram sugeridas algumas relações espaciais e espaço-temporais que podem ser especificadas entre os objetos de um documento. Em particular, foi apresentado como os atributos de objetos de mídia de um documento devem ser associados para que relações espaciais sejam definidas.

Aplicação do modelo espacial ao modelo NCM

O modelo espacial definido nessa dissertação pode ser aplicado a qualquer modelo de documentos hipermídia. No Capítulo 4 foi apresentada a aplicação desse modelo espacial ao modelo de contextos aninhados (NCM).

A definição de estruturas de apresentação em documentos NCM implicou no refinamento e na modificação de algumas de suas entidades. A entidade descritor foi acrescida de atributos para a definição da janela e da região a serem usadas na apresentação de objetos. Além disso, foram adicionadas duas novas entidades ao modelo NCM, *região* e *janela*, correspondentes às entidades do modelo espacial.

Foram definidas relações espaciais entre objetos NCM através de elos. Além disso, foi apresentado como animações podem ser adicionadas a documentos NCM (ver Seção 4.3.4). Para que relações e animações pudessem ser definidas, o evento de atribuição, já existente no modelo, precisou ser modificado drasticamente (ver Seção 4.3.1). Uma atribuição passou a ser

definida como a variação do valor de uma atributo de um *valor inicial* a um *valor final* durante um tempo definido por uma função de custo (*duração*).

Implementação do ambiente de autoria

O Capítulo 5 apresentou o Editor e *Browser* de Sincronismo do sistema HyperProp (EBS). O EBS é um ambiente composto de uma visão temporal e de uma visão espacial.

A próxima seção apresenta uma comparação do trabalho realizado nesta dissertação com os trabalhos apresentados no Capítulo 2.

6.2 COMPARAÇÃO COM TRABALHOS RELACIONADOS

Nas próximas subseções, a definição do modelo espacial e sua aplicação ao modelo NCM são analisadas, comparando-as aos trabalhos apresentados no Capítulo 2. A comparação feita é baseada nos aspectos relacionados à definição de sincronização espacial, apresentados na Seção 1.1.

6.2.1 Dispositivos de apresentação

O modelo espacial permite o uso de vários dispositivos na apresentação dos objetos de um documento. Como visto na Seção 3.1.1, a entidade *janela* do modelo espacial possui uma propriedade, de nome *devices*. Essa propriedade permite especificar os dispositivos de apresentação a serem usados na apresentação de uma janela e, por conseguinte, dos objetos que referenciem suas regiões. Entretanto, como explicado, o formatador de um documento é livre para modificar (ou ignorar) tal definição, segundo as restrições da plataforma de exibição ou configurações do usuário, leitor do documento.

Nenhum dos trabalhos apresentados no Capítulo 2 possui característica semelhante. Todos eles consideram a apresentação de todos os objetos de mídia visual (vídeo, texto, imagem estática) de um documento em um mesmo dispositivo de exibição. Obviamente, em todos eles, considera-se que os objetos de mídia auditiva são exibidos em um canal de áudio apropriado.

6.2.2 Posicionamento e dimensões de objetos de mídia

O modelo espacial define o uso de estruturas de apresentação, compostas de janelas e regiões (ver Seção 3.1). Cada região define um conjunto de propriedades que determinam, entre outras coisas, sua posição (*bottom*, *left*, *right* e *top*) e suas dimensões (*width* e *height*). Cada objeto de um documento hipermídia deve indicar a região (ou *slot*, como definido na Seção 1.1.2) a ser usada para sua apresentação. O objeto herda as características da região associada, inclusive suas propriedades de posição e dimensionamento. Também é permitida a definição de relações entre os componentes de uma estrutura de apresentação, facilitando a definição do posicionamento e das dimensões desses componentes.

A definição do modelo espacial foi influenciada pela linguagem SMIL [SMIL2] e pelo modelo Madeus [Villard00a]. SMIL define estruturas de apresentação baseadas em janelas e regiões. Madeus possui um modelo espacial composto por regiões, permitindo a definição de relações espaciais entre elas. A definição de relações espaciais será abordada em mais detalhes na Seção 6.2.5.

No SSTS [Nang97], o posicionamento e as dimensões dos objetos de uma apresentação são definidos nos nó de início correspondentes no grafo de sincronização da apresentação (ver Seção 2.3).

O modelo desenvolvido por Vazirgiannis [Vazirginannis00 e Theodoridis96], considera que cada objeto de uma apresentação multimídia possui suas características espaciais, como altura, largura, posição e cor, definidas implicitamente (ver Seção 2.4).

As linguagens de definição de folhas de estilo, apresentadas na Seção 2.5, especificam a apresentação de documentos XML através de hierarquias de áreas, que são associadas aos elementos de um documento através de estilos.

De uma maneira geral, a grande vantagem da definição de estruturas de apresentação, de que goza o modelo espacial, é a centralização da especificação das propriedades de apresentação dos objetos de um documento. O mesmo ocorre em SMIL e no modelo Madeus.

A definição de relações espaciais na estrutura de apresentação também facilita o processo de autoria, não precisando o autor definir um a um os valores das propriedades dos componentes de uma estrutura de apresentação.

Os pontos de registro, presentes na linguagem definida pelo modelo espacial, são usados tanto para posicionar componentes de apresentação, como para posicionar um objeto em uma região. O ponto de registro estende a entidade de mesmo nome da linguagem SMIL (definida nos módulos de *layout)* usada somente para posicionar um objeto em sua região associada (ver Seção 2.1).

Uma extensão ao modelo espacial, quando aplicado ao modelo NCM, permite a definição de animações (Seção 4.3.4) de objetos durante sua apresentação, através da definição de atribuições (Seção 4.3.1) com uma duração associada. Os atributos espaciais podem ter seu valor alterado segundo uma função, variando de um *valor inicial* a um *valor final*.

Todos os a trabalhos apresentados, com a exceção das linguagens de folhas de estilo (como será visto adiante), permitem a definição de algum tipo de animação. Em todos eles, assim como no modelo espacial, uma animação corresponde à variação do valor de algum atributo de um objeto.

A linguagem SMIL (ver Seção 2.1) [SMIL2] possui um conjunto de módulos para a definição de animações. Nele, são definidos elementos (`animate`, `animateMotion`, `animateColor`, `set`) que permitem definir a variação do valor de atributos com durações definidas. Atributos dos elementos, como `from`, `to`, `by`, `values`, `path` e `keyTimes`, possibilitam uma definição de alto nível do comportamento do valor de um atributo em uma animação. Além disso, é possível definir repetições de animações e "congelar" o valor de um atributo após uma animação.

Madeus [Tran_Thuong01] possui um modelo de animação baseado nos modelos de animação de SMIL e uma extensão, definida em SVG [Bowler01]. Ele estende a definição de SMIL, permitindo que animações sejam definidas como relação a um intervalo abstrato (*[0,1]*) e possa ser usado para animar a apresentação de qualquer objeto de um documento Madeus, adaptando-se ao intervalo associado ao objeto. (ver Seção 2.2)

SSTS [Nang97] permite a definição de eventos espaciais em seus grafos de sincronização. Cada evento espacial é representado por um nó que é colocado sobre a aresta de apresentação de um objeto. Um evento espacial define uma mudança nas características espaciais do objeto, permitindo movimentá-lo, ou alterar suas dimensões. (ver Seção 2.3)

Uma extensão [Vodislav00] ao trabalho de Vazirgiannis, apresentada na Seção 2.4, define um modelo de animação que permite a criação de animações interativas de objetos. Ele permite

aplicar transformações (espaciais, gráficas ou de conteúdo) aos objetos de uma apresentação multimídia. São definidos vários métodos que permitem, por exemplo, mudar o posicionamento, a orientação e a cor de um objeto durante sua apresentação. Uma característica única a esse modelo, é que um objeto é representado por um polígono qualquer.

Linguagens de definição folhas de estilo (ver Seção 2.5), individualmente, não possibilitam a criação de animações; no máximo, permitem a definição de estilos diferentes, onde as características de partes de um documento XML (como o posição de um bloco, sua cor e o tamanho de fonte usado) são definidas de forma diferentes. Linguagens de *script*, como JavaScript [ECMAScript] e VBScript, podem ser usadas em conjunto com estilos para definir mudanças na apresentação (animações) de um documento de forma interativa.

No NCM, animações são criadas através de atribuições de valores a atributos de nós. Como apresentado na Seção 4.3.1, uma atribuição é definida por um *valor inicial*, um *valor final* e uma duração, determinada por uma função de custo. Ainda, o *valor final* de uma atribuição pode ser definido em função dos valores de outros atributos, que podem ser do mesmo objeto ou de outros.

A definição de uma animação em NCM é bastante flexível. Os tipos de animação apresentados nos trabalhos relacionados podem ser definidos através de elos NCM determinando atribuições. Entretanto, devido à flexibilidade presente na definição de atribuições, a criação de animações pode se tornar um pouco complexa com relação aos outros modelos. Eles possuem um conjunto bem definido de animações, determinados por elementos ou métodos específicos. Nada impede, no entanto, que ambientes de autoria forneçam animações pré-fabricadas, facilitando a definição de animações em documentos NCM (ver Seção 6.3).

6.2.3 Redimensionamento de objetos de mídia

O modelo espacial utiliza o mesmo mecanismo definido por SMIL e usado parcialmente pelo modelo Madeus. Através da propriedade `fit` da entidade região, um objeto pode ter suas dimensões adaptadas (ver Seção 3.1.2). Os outros trabalhos não tratam deste aspecto.

6.2.4 Prioridade de apresentação

O modelo espacial define a propriedade `z-index`, que permite especificar a prioridade de apresentação de cada um das janelas e regiões de uma estrutura de apresentação. Isso define

como os objetos, que usem as regiões de uma estrutura, devem ser sobrepostos durante a apresentação. Essa é a mesma abordagem usada pela linguagem SMIL (ver Seção 2.1), pelo modelo Madeus (ver Seção 2.2) e semelhante à usada pela linguagem CSS (ver Seção 2.5).

Tanto SSTS [Nang97] como o modelo de animação de Vazirgiannis e Vodislav [Vodislav00] possuem métodos que permitem alterar a prioridade de apresentação de um objeto. No SSTS, existe o método `Basis(obj)`, que permite posicionar um objeto sobre outro (`obj`). No modelo de animação de Vazirgiannis, os métodos *raise()* e *back()*, permitem aumentar ou diminuir a prioridade de apresentação de um objeto.

O uso de uma propriedade ou de um método na definição da prioridade de apresentação de um objeto é indiferente, visto que o valor da propriedade deve ser alterado através de uma chamada de método ou ação (no caso do modelo NCM).

6.2.5 Relações entre objetos

Dentre os trabalhos apresentados, somente o modelo Madeus (ver Seção 2.2), o método SSTS (ver Seção 2.3) e o modelo proposto por Vazirgiannis (ver Seção 2.4), permitem a definição de relações entre os objetos de um documento ou apresentação.

O modelo Madeus [Villard00a, Layaïda97] define um conjunto básico de relações espaciais de alto nível [Carcone97b] que permitem relacionar espacialmente os componentes de uma composição. As relações espaciais sugeridas do modelo espacial apresentado nessa dissertação estendem esse conjunto (ver Seção 3.5.1). O modelo não faz qualquer restrição à criação de novas relações.

Uma vantagem do uso de relações espaciais no modelo NCM, com relação à linguagem Madeus, é que os objetos relacionados não precisam estar contidos diretamente na mesma composição.

Tanto SSTS [Nang97] como o modelo apresentado por Vazirgiannis [Vazirginannis00 e Theodoridis96] baseiam-se na definição de relações espaço-temporais. A vantagem principal de ambos os trabalhos é a representação uniforme de eventos espaciais e temporais.

O método SSTS mescla a definição de eventos temporais, que se resumem ao início ou o fim da apresentação de um nó, à ativação de eventos espaciais que representam mudanças nas propriedades de um ou mais objetos. Um objeto pode ter definido um movimento através de um caminho predefinido ou seu posicionamento exato.

Vazirgiannis define suas apresentações através de cenários, que mesclam a uso de relações espaciais (escolhidas a partir de um conjunto de 169 relações básicas) e relações temporais. Cada relação espacial define como dois objetos devem estar relacionados espacialmente. As relações temporais definem como a apresentação dos objetos é encadeada. Isso é feito usando-se eventos que permitem identificar o início, o fim, a interrupção e o reinício da apresentação de um objeto e a ativação de ações que iniciam, abortam, interrompem ou reiniciam a exibição de objetos. Essas relações (causais) são agrupadas em cenários que podem ser associados a eventos da apresentação ou do usuário para compor uma apresentação multimídia interativa.

Madeus [Villard00a, Villard98] também permite a definição de relações espaço-temporais. Ele define relações espaciais baseadas no tempo, onde a validade de relações tem uma duração predefinida, e relações temporais baseadas no espaço, onde ações, como o início ou término da apresentação de um objeto, podem ser executadas com base no posicionamento de um ou mais objetos. A definição de relações temporais com base no espaço se assemelha à definição de relações espaço-temporais dos outros trabalhos. Elas incluem-se nas categorias de relações espaço-temporais apresentadas na Seção 3.5.2. Exemplos de definição de tais relações através de elos NCM são apresentados na Seção 4.3.3.

A vantagem da definição de relações entre objetos no modelo NCM sobre os outros trabalhos é sua flexibilidade. Pode-se relacionar tantos objetos quanto se deseje e executar tipos diferentes de ações sobre esses objetos. Obviamente, isso só é possível pelo caráter baixo nível do modelo. O ambiente de autoria, como será destacado na Seção 6.3 deve prover um conjunto restrito de relações, como os outros trabalhos apresentados.

6.2.6 Transições

O trabalho aqui realizado não define efeitos de transição na apresentação de objetos. Somente a linguagem SMIL, em seus módulos de efeitos de transição, permite a criação de transições em uma apresentação.

6.3 TRABALHOS FUTUROS

Alguns aspectos do trabalho realizado nessa dissertação podem ser ainda explorados em trabalhos futuros. Eles voltam-se basicamente à aplicação do modelo espacial ao modelo NCM e algumas extensões possíveis.

Os trabalhos futuros podem ser resumidos em:

- definição de um modelo de efeitos de transição para o modelo NCM. Esse modelo pode-se aproveitar da taxonomia de efeitos de transição definida nos módulos de efeitos de transição de SMIL [SMIL2]. A implementação de efeitos de transição no modelo pode ser feita através da adição de um atributo à ação *Inicia* que indica qual o efeito de transição a ser aplicado no início da apresentação de um nó;
- revisão e definição de extensões à linguagem NCL (*Nested Context Language*) [Antonacci00] — a linguagem descritiva usada para a definição de documentos NCM. As alterações das entidades do modelo NCM apresentadas nesta dissertação devem ser refletidas na linguagem. Além disso, a linguagem deve ser estendida de forma a prover meios de definição de animações e efeitos de transição de forma declarativa. Os módulos de animação e efeitos de transição da linguagem SMIL podem ser utilizados com esse objetivo;
- reformulação do elo de restrição do modelo NCM, de forma que seja possível criar restrições entre atributos de objetos, facilitando a modelagem de relações espaciais;
- criação de um conjunto de relações espaço-temporais e animações, como ocorre em vários dos trabalhos relacionados, o que facilitaria a definição da sincronização espacial e temporal de documentos NCM. Esse conjunto deverá ser disponibilizado no Editor e *Browser* de Sincronismo, como já ocorre com as relações espaciais apresentadas no Capítulo 3. Outra extensão ao *EBS* seria a possibilidade de criação de efeitos de transição na apresentação de documentos NCM. Obviamente, essa extensão depende da adição de um mecanismo de definição de efeitos de transição ao modelo;
- extensão da visão temporal do EBS. Ela deve permitir que os nós de contexto possam ser expandidos, possibilitando uma granularidade maior da edição e visualização da sincronização temporal de um documento NCM. A organização visual dos objetos de uma composição deve respeitar a classe de nó de contexto utilizada. Caso sejam apresentados

nós de contexto seqüenciais ou paralelos, seus nós internos devem estar organizados de acordo com a semântica do nó de contexto;

- geração de estruturas de apresentação na forma declarativa a partir das estruturas de apresentação definidas no editor de *layout* da visão espacial do EBS;
- integração do processo de edição de documento nos *browsers* do sistema Hyperprop.

BIBLIOGRAFIA

[Allen83] ALLEN, James F. Maintaining Knowledge about Temporal Intervals. **Communications of the ACM**, v. 26, n. 11, nov. 1983, p. 832-843.

[Antonacci00] ANTONACCI, Meire Juliana. NCL: Uma Linguagem Declarativa para Especificação de Documentos Hipermídia com Sincronização Temporal e Espacial. **Dissertação de Mestrado**, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, abr. 2000.

[Bachelet00] BACHELET, Bruno; MAHEY, Philippe; RODRIGUES, Rogério Ferreira; SOARES, Luiz Fernando Gomes. Elastic Time Computation for Hypermedia Documents. **VI Simpósio Brasileiro de Sistemas Multimídia e Hipermídia (SBMídia2000)**, jun. 2000, p. 47-62.

[Bertino00] BERTINO, Elisa, FERRARI, Elena, STOLF, Marco. MPGS: An Interactive Tool for the Specification and Generation of Multimedia Presentations. **IEEE Transactions on Knowledge and Data Engineering**, v. 12, n. 1, p. 102-125, jan./fev., 2000.

[Bowler01] Scalable Vector Graphics (SVG) 1.0 Specification. **Proposta de Recomendação do World Wide Web Consortium (W3C)**, 19 jul. 2001.
Disponível em http://www.w3.org/TR/2001/PR-SVG-20010719.

[Buchanan93] BUCHANAN, M. Cecelia e ZELLWEGER, Polle T. Automatic Temporal Layout Mechanisms. **Proceedings of ACM Multimedia 93**, jun. 1993, p. 341-350.

[Buschmann97] BUSCHMANN, F.; MEUNIER, R.; ROHNERT, H.; SOMMERLAD, P.; e STAL, M. **Pattern-Oriented Software Architecture — A System of Patterns**, John Wiley & Sons, 1997.

[Caloini96] CALOINI, Andrea, TANAKA, Eiichiro. Extending Styles to Hypermedia Documents. **Proceedings of the 1996 International Conference on Multimedia Computing and Systems (ICMCS '96)**, p. 417-424.

[Carcone97a] CARCONE, Laurent; JOURDAN, Muriel; ROISIN, Cécile. Présentation de documents multimédia basée sur les contraintes. **Workshop on Electronic Page Models**, Lampe, França, set. 1997.

[Carcone97b] CARCONE, Laurent. Formatage spatial dans un environnement d'édition/présentation de documents multimédia. **Mémoire d'ingénieur**,

Cnam, dez. 1997.

[Costa96] COSTA, Fábio. Um Editor Gráfico para Definição e Exibição do Sincronismo de Documentos Multimídia/Hipermídia. **Dissertação de Mestrado**, Departamento de Informática da Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, ago. 1996.

[Cruz97a] CRUZ, Isabel F., LUCAS, Wendy T. A Visual Approach to Multimedia Querying and Presentation. **Proceedings of the ACM Multimedia 97**, Seattle, Washington, EUA, p. 109-119.

[Cruz97b] CRUZ, Isabel F., LUCAS, Wendy T. Delaunay^{MM}: a Visual Framework for Multimedia Presentation. **Proceedings of the 1997 IEEE Symposium on Visual Languages**, Ilha de Capri, Itália, p. 216-223, 23-26 set., 1997.

[CSS1] Cascading Style Sheets, level 1. **Recomendação do World Wide Web Consortium (W3C)**, 17 dez. 1996, revisado em 11 jan. 1999.
Disponível em http://www.w3.org/TR/1999/REC-CSS1-19990111.

[CSS2] Cascading Style Sheets, level 2 – CSS2 Specification. **Recomendação do World Wide Web Consortium (W3C)**, 12 mai. 1998.

Disponível em http://www.w3.org/TR/1998/REC-CSS2-19980512.

[CSS3] Introduction to CSS3. **Trabalho em andamento do World Wide Web Consortium (W3C)**, 6 abr. 2001.
Disponível em http://www.w3.org/TR/2001/WD-css3-roadmap-20010406.

[DSSSL] Document Style Semantics and Specification Language (DSSSL). **Information processing – Text and office systems, ISO/IEC 10679:1996**, Genebra, ISO/IEC, 1996.

[ECMAScript] ECMAScript Language Specification, Standard ECMA-262, **Standardizing Information and Communication Systems**. 3ª edição, dez. 1999.
Disponível em ftp://ftp.ecma.ch/ecma-st/Ecma-262.pdf.

[Eun93] EUN, Seong Bae, NO, Eun Suk, KIM, Hyung Chul, YOON, Hyunsoo, MAENG, Seung Ryoul. Specification of Multimedia Composition and A Visual Programming Environment. **Proceedings of the ACM Multimedia '93**, p. 167-174.

[Foley] FOLEY, James D.; VAN DAM, Andries; FEINER, Steven K.; HUGHES, John F.; e PHILLIPS, Richard L. **Computer Graphics: Principles and Practice**, Segunda Edição, Addison-Wesley, p. 488-491.

[Gamma95] GAMMA, E; HELM, R.; JOHNSON, R.; e VLISSIDES, J. **Design Patterns — Elements of Reusable Object-Oriented Software**, Addison-Wesley Inc., 1995.

[Halasz90] HALASZ, F.G., SCHWARTZ, M. The Dexter Hypertext Reference Model. **NIST Hypertext Standardization Workshop**, Gaithersburg. jan. 1990.

[Hardman98] HARDMAN, Lynda. Modelling and Authoring Hypermedia Documents. **Tese de Doutorado**, Universidade de Amsterdam, ISBN: 90-74795-93-5, 1998, 247pp.

[HTML4] HTML 4.01 Specification. **Recomendação do World Wide Web Consortium (W3C)**, 24 dez. 1999.
Disponível em http://www.w3.org/TR/1999/REC-html401-19991224.

[Jourdan97] JOURDAN, Muriel, LAYAÏDA, Nabil, SABRY-ISMAIL, Loay, TARDIF, Laurent. MADEUS – An Authoring Environment for Multimedia Documents. **Proceedings of the 1997 IEEE International Conference on Multimedia Computing and Systems (ICMCS'97)**, Ottawa, Canadá, jun. 1997, p. 644-645.

[Jourdan98a] JOURDAN, Muriel, ROISIN, Cécile, TARDIF, Laurent. Constraints Techniques for Authoring Multimedia Documents. **Proceedings of the Workshop on Constraints for artistic applications (ECAI 98)**, Brighton, Inglaterra, ago. 1998.

[Jourdan98b] JOURDAN, Muriel, LAYAÏDA, Nabil, ROISIN, Cécile, SABRY-ISMAIL, Loay, TARDIF, Laurent. Madeus, an Authoring Environment for Interactive Multimedia Documents. **Proceedings of the ACM Multimedia'98**, pp. 267-272, ACM, Bristol (UK), set. 1998.

[Jourdan99a] JOURDAN, Muriel, ROISIN, Cécile, TARDIF, Laurent., VILLARD, Lionel. Authoring SMIL documents by direct manipulations during presentation. **World Wide Web**, Balzer Science Publishers, v. 2, n. 4, dez. 1999.

[Jourdan99b] JOURDAN, Muriel, TARDIF, Laurent, VILLARD, Lionel. SMILY, a SMIL authoring enviroment. **Proceedings of the ACM Multimedia '99,** (Part 2), Orlando, Flórida, EUA, 1998, p. 198.

[Jourdan99c] JOURDAN, Muriel, ROISIN, Cécile, TARDIF, Laurent. A Scalable Toolkit for Designing Multimedia Authoring Environments, **Special Number, 'Multimedia Authoring and Presentation: Strategies, Tools, and Experiences' of Multimedia Tools and Applications Journal**, Kluwer Academic Publishers, 1999.

[Layaïda97] LAYAÏDA, Nabil. Madeus: système d'édition et de présentation de documents structurés multimédia. **Tese de Doutorado**, Université

documents structurés multimédia. **Tese de Doutorado**, Université Joseph Fourier, Grenoble, France, jun. 1997.

[Marden98] MARDEN Jr., Philip M., MUNSON, Ethan V. PSL: An Alternate Approach to Style Sheets on the Web. **Proceedings of WebNet 98, World Conference of the WWW, Internet and Intranet**, Orlando, FL, EUA, nov. 1998.

[Moura01a] MOURA, Marcel Stanley Albuquerque de. Um Modelo de *Layout* Genérico para Documentos Hipermìdia e Implementação de Um Editor de *Layout* para o Sistema HyperProp, **Projeto Final de Programação**, mar. 2001.

[Moura01b] MOURA, Marcel Stanley Albuquerque de. O *Browser* e Editor de Sincronismo do Sistema HyperProp, versão 2.0, 2001.

[Moura01c] MOURA, Marcel Stanley Albuquerque de. Uma Linguagem Genérica para a Definição de Estrutruras de Apresentação de Documentos Hipermídia, 2001.

[Munson98] MUNSON, Ethan V., PFEIFFER, Mark. A Representation of Media for Multimedia Authoring and Browsing Systems. **Proceedings of the AAAI 98 Workshop on Representations for Multi-Modal Human-Computer Interaction**, Madison, WI, EUA, jul. 1998.

[Nang97] NANG, J. e KANG, S. A New Multimedia Synchronization Specification Method for Temporal and Spatial Events. **Proceedings of the 1997 International Conference on Multimedia Computing and Systems (ICMCS '97)**, Ottawa, Canadá, jun. 1997, p. 236-243.

[OCPN] LITTLE, T. e GHAFOOR, A. Synchronization and Storage Models for Multimedia Objects. **IEEE Communications Magazine**, v. 8, n. 3, abr. 1990, p. 413-427.

[Papadias95] PAPADIAS, Dimitris, THEODORIDIS, Yannis, SELLIS, Timos, EGENHOFER, Max J. Topological Relations in the World of Minimum Bounding Rectangles: A Study with R-trees. **Proceedings of the ACM SIGMOD International Conference on Management of Data**, 1995

[Papadias97] PAPADIAS, Dimitris, THEODORIDIS, Yannis. Spatial Relations, Minimum Bounding Rectangles, and Spatial Data Structures, **International Journal of Geographic Information Systems**, v. 11 (2), 1997.

[Perez-Luque96] PÉREZ-LUQUE, M. e LITTLE, T. A Temporal Reference Framework for Multimedia Synchronization. **IEEE Journal on Selected Areas in Communications (Special Issue: Synchronization Issues in Multimedia Communication)**, v. 14, n. 1, jan. 1996, p. 36-51.

[Pinto00] PINTO, Luís Arthur. Autoria Gráfica de Estruturas de Documentos Hipermídia no Sistema HyperProp. **Dissertação de Mestrado**, Departamento de Informática da Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, ago. 2000.

[Rodrigues00] RODRIGUES, Leandro Marques. Integração de Documentos SMIL ao Sistema HyperProp e Desenvolvimento de Ferramentas para Exibição de Objetos com Relacionamentos de Sincronização. **Dissertação de Mestrado**, Departamento de Informática da Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, nov. 2000.

[Rodrigues97] RODRIGUES, Rogério Ferreira. Formatação Espacial e Temporal no Sistema HyperProp. **Dissertação de Mestrado**, Departamento de Informática da Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, mai. 1997.

[Rodrigues98] RODRIGUES, Rogério Ferreira, SAADE, Débora Cristina Muchaluat, SOARES, Luiz Fernando Gomes. Composite Nodes, Contextual Links and Graphical Structural Views on the WWW. **Journal of the Brazilian Computer Society, Special Issue on World-Wide Web**, v. 5, n. 2, Sociedade Brasileira de Computação**,** nov. 1998, p. 31-44.

[Rodrigues99a] RODRIGUES, Leandro Marques, RODRIGUES, Rogério Ferreira, SAADE, Débora Cristina Muchaluat, SOARES, Luiz Fernando Gomes. Acrescendo Facilidades NCM a Documentos SMIL. **Anais do Simpósio Brasileiro de Sistemas Multimídia e Hipermídia (SBMídia'99)**, Goiânia, Goiás, Junho de 1999.

[Rodrigues99b] RODRIGUES, Leandro Marques, RODRIGUES, Rogério Ferreira, SAADE, Débora Cristina Muchaluat, SOARES, Luiz Fernando Gomes. Improving SMIL Documents with NCM Facilities. **Proceedings of the Multimedia Modeling Conference'99**, Ottawa, Canadá, out. 1999.

[Rodrigues01] RODRIGUES, Leandro Marques, RODRIGUES, Rogério Ferreira e SOARES, Luiz Fernando Gomes. A Framework for Event-Driven Hypermedia Presentation Systems, **Multimedia Modeling Conference – MMM'2001**, Amsterdam, nov. 2001. (aceito para publicação)

[Roisin99] ROISIN, Cécile, TRAN_THUONG, Tien e VILLARD, Lionel. Integration pf structured video in a multimedia authoring system. **Proceedings of the Eurographics Multimedia ’99 Workshop**, Springer Computer Science, set. 1999, p. 327-337.

[Rutledge99] RUTLEDGE, Lloyd, HARDMAN, Lynda, BULTERMAN, Dick C. A. **Proceedings of the ACM Multimedia ’99 (Part 2)**, Orlando, Flórida, EUA, p. 200.

[Saade96] SAADE, Débora Cristina Muchaluat. Browsers e Trilhas para Documentos Hipermídia Baseados em Modelos com Composições

Documentos Hipermídia Baseados em Modelos com Composições Aninhadas. **Dissertação de Mestrado**, Departamento de Informática da Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, mar. 1996.

[Shih97] SHIH, Timothy K., CHANG, Anthony Y. Toward a Generic Spatial/Temporal Computation Model for Multimedia Presentations, **Proceedings of the 1997 International Conference on Multimedia Computing and Systems (ICMCS '97)**, p. 228-235.

[Shih98] SHIH, Timothy K., CHANG, Anthony Y., LIN, Hwei-Jen, YEN, Shwu-Huey, CHIU, Chuang-Feng. Interval for Spatio-Temporal Composition of Distributed Multimedia Objects. **Proceedings of the IEEE International Conference on Parallel and Distributed Systems**, 14-16 dez. 1998, Taiwan, pp. 308-315.

[Shim00] SHIM, Choon-Bo, CHANG, Jae-Woo. Spatio-temporal Representation and Retrieval Using Moving Object's Trajectories. **Proceedings of the ACM Multimedia Workshop**, Marina Del Rey-CA, EUA, p. 209-212, 2000.

[SMIL1] Synchronized Multimedia Integration Language (SMIL) 1.0 Specification. **Recomendação do World Wide Web Consortium (W3C),** 15 jun. 1999.
Disponível em http://www.w3.org/TR/1998/REC-smil.

[SMIL2] Synchronized Multimedia Integration Language (SMIL) 2.0 Specification. **Trabalho em andamento no World Wide Web Consortium (W3C)**, 01 mar. 2001.
Disponível em http://www.w3.org/TR/2001/WD-smil20-20010301.

[Soares00a] SOARES, Luiz Fernando Gomes, RODRIGUES, Rogério Ferreira, MUCHALUAT-SAADE, Débora Cristina. Authoring and Formatting Hypermedia Documents in the HyperProp System. **ACM Multimedia Systems Journal**, v. 8, n. 2, Springer-Verlag, mar. 2000, p. 118-134.

[Soares00b] SOARES, Luiz Fernando Gomes et al. Modelo de Contextos Aninhados Versão 2.4. **Relatório Técnico do Laboratório de Redes de Computadores e Sistemas Multimídia**, Pontifícia Universidade Católica do Rio de Janeiro, 2000.

[Soares99] SOARES, Luiz Fernando Gomes, SOUZA FILHO, Guido Lemos, RODRIGUES, Rogério Ferreira, MUCHALUAT-SAADE, Débora Cristina. Versioning Support in the HyperProp System, **Journal of Multimedia Tools and Applications**, v. 8, n. 3, mai. 1999.

[Souza97] SOUZA FILHO, Guido Lemos. Sincronismo na Modelagem e Execução de Apresentações de Documentos Multimídia. **Tese de Doutorado**, do Departamento de Informática da Pontifícia Universidade

Católica do Rio de Janeiro, Rio de Janeiro, set. 1997.

[Tardif00] TARDIF, Laurent, BES, Frédéric, ROISIN, Cécile. Constraints for multimedia documents. **Proceedings of the International Conference on Practical Applications of Constraint Techniques and Logic Programming**, Manchester, Inglaterra, abr. 2000.

[Theodoridis96] THEODORIDIS, Yannis, VAZIRGIANNIS, Michalis, SELLIS, Timos. Spatio-Temporal Indexing for Large Multimedia Applications. **Proceedings of the 3rd IEEE International Conference on Multimedia Computing and Systems (ICMCS'96)**.

[Tran_Thuong01] TRAN_THUONG, Tien e ROISIN, Cécile. A Sub-Element Based Model for Animation and Structured Video Presentation in Multimedia Document. Artigo submetido ao **ACM Symposium on Document Engineering 2001 (SDE'01),** Atlanta, Georgia, USA, 9-10 nov. 2001.

[Vazirgiannis96a] VAZIRGIANNIS, Michalis, THEODORIDIS, Yannis, SELLIS, Timos. Spatio-Temporal Composition in Multimedia Applications. **Proceedings of the International Workshop on Multimedia Software Development (IEEE-ICSE '96)**, Berlin, 1996.

[Vazirgiannis96b] VAZIRGIANNIS, Michalis, SELLIS, Timos. Event and Action Representation and Composition for Multimedia Application Scenario Modelling, **Proceedings of the ERCIM Workshop on Interactive Distributed Multimedia Systems and Services**, Berlin, mar. 1996.

[Vazirginannis00] VAZIRGIANNIS, Michalis et al. Interactive Multimedia Documents: a Modeling, Authoring and Rendering approach. **Multimedia Tools & Applications Journal (Kluwer Academic Publishers)**, 2000.

[Vazirginannis97] VAZIRGIANNIS, Michalis, BOLL, Suzanne. Events In Interactive Multimedia Applications: Modeling And Implementation Design. **Proceedings of the 1997 IEEE International Conference on Multimedia Computing and Systems (ICMCS'97),** Ottawa, Canadá, jun.1997, p. 244-251.

[Villard00a] VILLARD, Lionel, ROISIN, Cécile, e LAYAÏDA, Nabil. A XML-based multimedia document processing model for content adaptation. **Digital Documents and Electronic Publishing (DDEP'00)**, LNCS, set. 2000.

[Villard00b] VILLARD, Lionel. Madeus Model DTD, 2000.
Disponível em http://www.inrialpes.fr/opera/madeusmodel.dtd.

[Villard98] VILLARD, Lionel. Spécification de relations spatio-temporelles dans un document multimédia. **Relatório de DEA em Informática**, Universidade Joseph Fourier (Grenoble 1), Grenoble, França, jun. 1998.

[Vodislav00] VODISLAV, Dan, VAZIRGIANNIS, M. Structured Interactive Animation for Multimedia Documents, **Proceedings of the 16th IEEE International Symposium on Visual Languages (VL'00)**.

[Vodislav97] VODISLAV, Dan. A Visual Programming Model for User Interface Animation, **Proceedings of the 13th IEEE International Symposium on Visual Languages (VL'97)**, Capri, Itália, set. 1997.

[XML] Extensible Markup Language (XML) 1.0. **Recomendação do World Wide Web Consortium (W3C)**, 10 fev. 1998.
Disponível em http://www.w3.org/TR/1998/REC-xml-19980210.

[XSL] Extensible Stylesheet Language (XSL), Version 1.0. **Trabalho em andamento do World Wide Web Consortium (W3C)**, 27 mar. 2000.
Disponível em http://www.w3.org/TR/2000/WD-xsl-20000327.

[XSLT] XSL Transformations (XSLT), Version 1.0. **Recomendação do World Wide Web Consortium (W3C)**, 16 nov. 1999.
Disponível em http://www.w3.org/TR/1999/REC-xslt-19991116.