2
Modelos de Documentos Hipermídia com Recursos de Adaptação

A computação orientada a contexto (*Context-Aware Computing*) (Dey et al., 2001; Schilit et al., 1994) é um paradigma que se propõe a permitir que as aplicações possam descobrir e tirar proveito das informações que as circundam, tais como, perfil do usuário, localização, horário do dia, recursos próximos, infra-estrutura disponível, entre outros. Muitas das propostas nessa linha de pesquisa estão ligadas à computação móvel e à computação ubíqua; no entanto, várias das questões que devem ser tratadas relacionadas a essa área resolvem problemas que independem de características de mobilidade e ubiqüidade do sistema.

Conforme discutido no capítulo anterior, é desejável em sistemas hipermídia a existência de mecanismos que realizem a adaptação dos documentos, isto é, que apliquem algum tipo de transformação nos hiperdocumentos a fim de torná-los mais apropriados ao contexto em que estejam sendo visualizados. Dessa forma, os sistemas hipermídia que dispõem de técnicas de adaptação para a formatação dos hiperdocumentos podem ser considerados como exemplos de aplicações orientadas a contexto. Existe, contudo, uma grande quantidade de ajustes e transformações que podem ser entendidos como formas de adaptação de um hiperdocumento.

O objetivo deste capítulo é, inicialmente, discutir as principais possibilidades de adaptação de documentos hipermídia. Isso é feito basicamente respondendo a quatro perguntas: “o que adaptar?”, “em função do que adaptar?”, “onde adaptar?” e “quando adaptar?”. Essa análise permite estabelecer as principais características que devem permear o modelo hipermídia utilizado como base para a elaboração dos hiperdocumentos, assim como o modelo de informações contextuais. Além dos requisitos para favorecer a incorporação de mecanismos de adaptação, o capítulo também apresenta uma análise geral das características desejáveis de serem encontradas em um modelo de hiperdocumentos. Com base em todos esses estudos, o capítulo se encerra com a

PUC-Rio - Certificação Digital Nº 9916204/CA

descrição do modelo hipermídia de execução utilizado como ponto de partida para a construção da arquitetura de formatação apresentada no capítulo seguinte.

2.1
Adaptação em sistemas hipermídia

2.1.1
Documentos hipermídia

Para entender as possíveis formas de adaptação de documentos hipermídia, o ponto de partida é analisar o objeto da adaptação, ou seja, os próprios hiperdocumentos. Na realidade, mais que os documentos, é preciso compreender o modelo hipermídia por trás de suas descrições. Todo sistema hipermídia reside sobre pelo menos um modelo de documentos que estabelece um conjunto de entidades, a semântica dessas entidades, os seus inter-relacionamentos e as operações que podem ser aplicadas sobre as mesmas. É possível encontrar na literatura uma variedade de propostas de modelos de documentos multimídia/hipermídia. AHM (Hardman et al., 1993a), Dexter (Halasz & Schwartz, 1994), HyTime (ISO/IEC, 1992), I-HTSPN (Willrich et al., 1996), IMD (Vazirgiannis, 1999), Labyrinth (Díaz et al., 2001), Madeus (Jourdan et al., 1998a), MHEG (ISO/IEC, 1996), NCM (Soares et al., 1995; Soares et al., 2000), OCPN (Little & Ghafoor, 1990) e Z_YX (Boll & Klas, 1999) são alguns exemplos. As definições de alguns modelos podem também ser implicitamente encontradas na especificação de linguagens declarativas, como HTML (W3C, 1999c) e SMIL (W3C, 2001a). Além desses casos, certos sistemas como Chimera (Anderson, 1997), CMIFed (van Rossum et al., 1993), DHM (Grønbaek et al., 1997), Firefly (Buchanan & Zellweger, 1992), Hyper-G (Andrews et al., 1995) e Microcosm (Hall et al., 1996) também possuem definições de modelos multimídia/hipermídia associadas.

PUC-Rio - Certificação Digital Nº 9916204/CA

2.1.1.1 Adaptação dos nós

De um modo geral, duas entidades estão presentes em todos os modelos hipermídia: os nós e os relacionamentos entre os nós. Um nó hipermídia, também conhecido como objeto de dados ou recurso, tem como principais atributos sua identificação e o seu conteúdo. O principal tipo de nó encontrado em todos os modelos hipermídia é o nó de mídia, muitas vezes denominado nó de conteúdo ou objeto de mídia. Esses nós atômicos (ou simples) costumam ser especializados em subclasses, tais como texto, imagem, áudio e vídeo, em função do tipo de informação por ele contida. Normalmente, o valor do atributo conteúdo desses nós é expresso através de uma referência para os dados propriamente ditos (URI de um arquivo, por exemplo), mas nada impede que o conteúdo esteja inteiramente armazenado no próprio nó. Alguns modelos hipermídia utilizam o conceito de composição para a definição de um outro tipo de nó: o nó composto. Um nó de composição, também chamado de objeto de dados composto, permite agrupar diversos nós (simples ou compostos) recursivamente, oferecendo suporte para uma especificação estruturada e muitas vezes hierárquica dos documentos.

A adaptação dos objetos de mídia (objetos de dados simples) está diretamente relacionada com a adaptação do conteúdo do documento. A forma mais comum desse tipo de adaptação é a escolha de partes do documento baseando-se em diferentes alternativas de conteúdo contidas em arquivos, ou fontes de dados[3], distintos. No entanto, nada impede que as alternativas sejam estabelecidas automaticamente pelo sistema, extraídas de uma única versão do conteúdo; por exemplo, através de um mecanismo de descarte, ou inserção, automático de quadros que consiga alterar a taxa de exibição de um vídeo.

Em termos das opções de alternativas que podem ser estabelecidas para os conteúdos dos nós, é possível identificar duas principais categorias. A primeira é a adaptação pela simples variação da qualidade da informação, por exemplo, modificando a resolução em pixels ou as opções de cores de uma imagem. A segunda categoria de adaptação é aquela que modifica a própria quantidade (ou

[3] Exemplos de fontes de dados seriam câmeras de captura de vídeo.

PUC-Rio - Certificação Digital Nº 9916204/CA

tipo) de informação apresentada, tal como a alteração do texto de uma aula em função dos objetivos e pré-requisitos do aluno.

Algumas observações devem ser feitas com relação às definições dos dois parágrafos anteriores. Em primeiro lugar, o ajuste automático de um conteúdo (estabelecimento das alternativas pelo próprio sistema a partir de uma única versão) é mais simples de ser realizado quando a adaptação modifica apenas a qualidade da informação, ao passo que a seleção de alternativas a partir de diferentes versões pode ser facilmente utilizada em qualquer uma das opções. Outra consideração importante é que as adaptações tanto da quantidade como da qualidade da informação podem ser interessantes de serem aplicadas de modo combinado. Por exemplo, o vídeo de uma aula a ser exibida para um aluno pode ser escolhido dependendo do seu histórico (aula para iniciante, aula para aluno avançado etc.) e, posteriormente, a sua taxa de exibição escolhida em função da banda passante da conexão de rede utilizada pelo aluno.

A adaptação automática da qualidade dos conteúdos a partir de uma única versão é uma característica desejável em um sistema de apresentação hipermídia, pois traz como benefício, principalmente para os autores, a facilidade no reuso e manutenção dos objetos, evitando uma proliferação de versões desses objetos. Todavia, tal adequação automática traz consigo algumas dificuldades em termos de implementação. Por exemplo, para que a taxa de transmissão de um vídeo na rede efetivamente diminua, é preciso que o transmissor saiba efetuar uma manipulação de descarte do conteúdo e que o receptor consiga processar esse conteúdo modificado. Além disso, esse modo de adaptação limita o ajuste a uma taxa mínima, abaixo da qual não é possível compreender a informação (Boll et al., 1999a). Mais ainda, pode ser que o ambiente de execução do cliente não disponha de decodificadores ou dispositivos de saída apropriados para o tipo de mídia do conteúdo. Sendo assim, é útil que, além da adaptação automática a partir de uma única versão, também seja possível a adaptação dos documentos através de substituições dos conteúdos por outras alternativas, que podem, inclusive, ser de tipos de mídia diferentes. Por exemplo, em um dado contexto, pode ser apropriado substituir um nó de áudio por um nó de texto. Esse tipo de adaptação é também conhecido como adaptação *cross-media* (Boll et al., 1999a).

PUC-Rio - Certificação Digital Nº 9916204/CA

Apesar de existir a possibilidade de transformar o tipo de mídia de um conteúdo automaticamente, por exemplo, sintetizando um áudio a partir de um texto, a adaptação *cross-media* é mais usual de ser implementada utilizando a idéia de versões distintas previamente estabelecidas em conteúdos separados. Isso porque a transformação automática do tipo de mídia pode impor retardos proibitivos, principalmente para aplicações interativas.

Um caso particular de adaptação de conteúdo pela variação de qualidade que merece destaque é a modificação na duração dos objetos, também denominada *computação dos tempos elásticos* ou simplesmente *ajuste elástico* (Bachelet et al., 2000; Buchanan & Zellweger, 1993b; Kim & Song, 1995). Esse tipo de adaptação atua na apresentação dos objetos, aumentado (*stretching*) ou diminuindo (*shrinking*) suas durações em relação aos valores originais (ou ideais), com o objetivo de garantir a consistência temporal do documento, ou seja, respeitar as restrições temporais entre objetos de mídia impostas pelo autor. Para as mídias estáticas como texto ou imagem, esse tipo de ajuste é trivial de ser efetuado. No entanto, para as mídias contínuas, tais como vídeo e áudio, esse tipo de adaptação pode se tornar mais complexo de ser implementado. O ideal é que o sistema (ferramentas de exibição, ambiente de armazenamento etc.) possua algoritmos que modifiquem a duração do conteúdo dinamicamente, preservando ao máximo a informação original. Na impossibilidade de dispor desses algoritmos, duas soluções simples são: utilizar a idéia de seleção de alternativas (possuir o objeto em várias versões, com durações distintas) ou realizar um simples corte (truncamento) ou repetição (loop) de parte do conteúdo. A primeira solução, entretanto, traz dificuldades para a autoria, obrigando a geração do conteúdo em diversas versões, enquanto a segunda estratégia provavelmente resultará em uma má qualidade de apresentação.

É importante observar que os conceitos em que se baseia o ajuste elástico temporal podem ser transpostos para o arranjo espacial dos objetos, resultando em um ajuste elástico das dimensões dos nós cujos conteúdos são baseados em mídias visíveis. Na verdade, historicamente, a funcionalidade de ajuste espacial automático é anterior ao ajuste elástico temporal, sendo amplamente explorada, principalmente nos editores de texto.

PUC-Rio - Certificação Digital Nº 9916204/CA

Nem sempre o ajuste elástico está relacionado com a qualidade da mídia exibida, mas somente esse caso será tratado neste trabalho. Apenas como título de exemplo, suponha que para a manutenção do sincronismo temporal, fosse primeiro determinado o tempo disponível para exibição de cada objeto, que guiaria então a procura por uma alternativa que tratasse do assunto naquele tempo estabelecido.

A definição de composições em modelos hipermídia pode ser de utilidade para alguns dos tipos de adaptação comentados até aqui. Em primeiro lugar, se essas composições tiverem tratamento similar ao dos nós de conteúdo, ou seja, se elas também forem consideradas objetos de dados, a especificação de alternativas para os objetos de dados simples poderia ser facilmente estendida, permitindo, dessa forma, utilizar essa flexibilidade para, ao invés de realizar uma adaptação de mídias particulares, efetuar uma adaptação estrutural (ou mesmo no formato de apresentação) dos documentos. Um exemplo de substituição desse tipo seria a troca de um vídeo por uma seqüência de imagens juntas com um texto explicativo para acomodar a apresentação à banda passante disponível no acesso do cliente. Um segundo aspecto relaciona-se com o ajuste temporal. As composições podem permitir que os algoritmos de adaptação atuem de forma hierárquica, calculando as durações de maneira global em termos de objetos compostos e, independente e progressivamente, refinando o cálculo para os objetos contidos em cada composição.

Um aspecto importante nos modelos de documentos hipermídia é a maneira como os parâmetros de apresentação dos objetos de dados, tais como posição na tela, cor de fundo, ferramenta de exibição e duração são especificados. Em alguns modelos, essas informações são armazenadas em entidades específicas, separadas da estrutura lógica e do conteúdo do documento. Isso permite que um mesmo padrão de apresentação seja aplicado a vários objetos de dados, e que um mesmo objeto seja exibido com diferentes características, oferecendo uma interessante forma de reutilização e uma facilidade de manutenção das especificações de autoria. Neste trabalho, denominaremos *descritor* a entidade responsável por conter as informações para apresentação de um objeto de dados.

Permitir que o autor do documento especifique, ao invés de um único descritor, uma lista de opções, torna possível ao sistema hipermídia selecionar as

PUC-Rio - Certificação Digital Nº 9916204/CA

características de apresentação mais adequadas para cada objeto de dados a ser exibido. Com isso, os objetos do documento passam a poder ter não apenas os seus conteúdos, mas também os seus aspectos de exibição adaptáveis. Outra flexibilidade interessante é permitir que essas características de apresentação sejam escolhidas também em função da navegação do usuário e de suas preferências.

De um modo geral, todos os mecanismos de adaptação associados à qualidade da informação devem atuar cooperativamente com o controle de apresentação do sistema hipermídia. Dispor os ambientes de execução dos sistemas hipermídia com esses tipos de recursos de adaptabilidade permite que o autor crie um único documento que poderá ser exibido em uma diversidade de plataformas. Além disso, em plataformas onde não é possível negociar e reservar recursos, os mecanismos de adaptação da qualidade do conteúdo podem tornar o sistema capaz de reagir e adequar a apresentação aos recursos que estiverem sendo oferecidos em um dado instante. Mesmo em plataformas com provisão de QoS (reserva de recursos), existe sempre a possibilidade do provedor de serviços não conseguir manter a qualidade negociada, obrigando uma renegociação e um conseqüente ajuste da apresentação. Tudo isso evidencia a importância de um dos objetivos desta tese, qual seja, tratar dos mecanismos de adaptação ligados à variação na qualidade das informações exibidas.

A adaptação do tipo (ou quantidade) de informação apresentada é, sem dúvida, um requisito desejável em sistemas hipermídia, mas não faz parte do escopo deste trabalho. A referência (Brusilovsky, 1996) oferece um panorama amplo das pesquisas nessa área, e este capítulo, em particular, procura também contribuir com a análise das diversas possibilidades de adaptação para apresentações de hiperdocumentos.

2.1.1.2 Adaptação das relações

Conforme comentado no início da seção anterior, juntamente com os nós, os relacionamentos entre eles constituem-se num dos conceitos mais importantes presente em todos os modelos hipermídia. Esses relacionamentos são normalmente representados através de *elos* (*links*) ou de *composições*. Exemplos

PUC-Rio - Certificação Digital Nº 9916204/CA

do uso de elos podem ser encontrados nos hiper-elos da linguagem HTML (W3C, 1999c), nos elos espaciais e temporais do modelo NCM (Soares et al., 2000), nas transições das redes de petri dos modelos OCPN (Little & Ghafoor, 1990) e I-HTSPN (Willrich et al., 1996), e nos relacionamentos temporais no sistema Firefly (Buchanan & Zellweger, 1992). As composições paralela e seqüencial da linguagem SMIL (W3C, 2001a), os operadores temporais do modelo ZyX (Boll & Klas, 1999), bem como os templates da linguagem NCL (Muchaluat-Saade, 2003) são exemplos de composições que estabelecem semânticas de relacionamentos temporais entre os seus componentes.

Existem vários tipos de relações que podem estar presentes em um modelo de documentos hipermídia (Soares et al., 2000): relações de referência, relações de contextualização, relações de sincronização, relações de derivação e relações entre tarefas. No entanto, a maioria dos trabalhos que abordam a questão da adaptação das relações (Brusilovsky, 1996) o fazem apenas para as relações de referência, também denominadas de relações de navegação. Na verdade, na referência citada, a adaptação das relações de contextualização, que são aquelas que especificam a estrutura hierárquica de organização do documento, por exemplo, a divisão de um livro em capítulos, dos capítulos em seções e assim por diante, também são mencionadas, porém englobadas como relações de navegação.

As técnicas de suporte à chamada navegação adaptativa procuram auxiliar os usuários a encontrar o caminho mais apropriado de interação no documento. Basicamente, essas técnicas se dividem em viagens guiadas, ordenação dos relacionamentos, ocultação adaptativa dos relacionamentos, anotação adaptativa dos relacionamentos e adaptação de mapas (Brusilovsky, 1996).

Não faz parte do foco desta tese abordar os aspectos de adaptação dos relacionamentos. Os estudos nesse tema sofrem uma grande influência da inteligência artificial e, na maioria das vezes, dependem de uma pesquisa em conjunto com a área de interação humano-computador. A referência (Brusilovsky, 1996) analisa diversos trabalhos que utilizam diferentes métodos para aplicação das cinco técnicas de adaptação citadas no parágrafo anterior. Além desse trabalho, as referências (Muchaluat-Saade, 1996; Muchaluat-Saade et al., 1998) mostram a utilização de visões de olho-de-peixe como técnicas de adaptação de mapas para auxiliar tanto na autoria como na navegação de documentos

PUC-Rio - Certificação Digital Nº 9916204/CA

hipermídia. Essas duas últimas referências também tratam do uso de trilhas para orientar os usuários em viagens guiadas.

Concluindo esta Seção 2.1.1, em termos de modelos de documentos hipermídia, é possível resumir como sendo duas as principais entidades (ou conceitos) que podem ser objetos de um mecanismo de adaptação: os nós (de mídia ou compostos) e os relacionamentos entre os nós (elos ou composições). Evidentemente, essas duas entidades podem ser ajustadas de uma maneira combinada. Esta tese, no entanto, tem como foco apenas a adaptação de nós ligada à qualidade da informação exibida.

2.1.2
Elementos que influenciam a adaptação dos hiperdocumentos

O objetivo desta seção é analisar as principais informações que, de alguma maneira, podem direcionar os mecanismos de adaptação de um sistema hipermídia na escolha da configuração mais adequada do documento. Em outras palavras, esta seção procura responder à pergunta: em função do que deve ser feita a adaptação dos hiperdocumentos? Na verdade, alguns exemplos da seção anterior já mencionaram certos aspectos ligados a essa questão.

Como ponto de partida, existem dois principais conjuntos de parâmetros externos aos documentos que podem exercer influência sobre suas adaptações: as características do usuário e as características da infra-estrutura (Boll et al., 1999b). Os valores desses conjuntos de parâmetros são justamente as informações que reunidas compõem o contexto de exibição de uma aplicação hipermídia, mencionado no capítulo anterior e no princípio deste capítulo. A adaptação às características do usuário, também classificada como adaptação ao interesse pessoal, pode levar em consideração o conhecimento do usuário a respeito dos assuntos tratados no conteúdo dos documentos, os objetivos e comportamento do usuário durante a sua navegação, a sua experiência prévia e as suas preferências. Já a adaptação à infra-estrutura concentra-se nos aspectos relacionados à plataforma de exibição, tais como a banda passante do sistema de comunicação, o poder computacional e memória disponíveis nos equipamentos participantes da transmissão e da exibição do documento, os dispositivos de E/S presentes na máquina onde o documento é exibido etc.

PUC-Rio - Certificação Digital Nº 9916204/CA

Manter os valores desses vários parâmetros exige a introdução do conceito de *gerência de contexto* nos sistemas hipermídia adaptativos. Devido ao dinamismo de algumas das informações, tais como banda passante e recursos de processamento e armazenamento disponíveis, é útil que haja agentes monitores distribuídos pelos ambientes, a fim de coletar os dados pertinentes para constituição do contexto, que pode ser mantido de uma maneira distribuída ou por uma unidade central. Com o objetivo de tornar a gerência de contexto um mecanismo independente e reutilizável por várias aplicações (não necessariamente apenas aplicações hipermídia) é importante que se estabeleça um protocolo aberto a ser utilizado na consulta às informações do contexto e também um formato aberto e extensível para descrição dos dados. Esforços em pesquisa têm sido feitos para tratar a questão da gerência de contexto em redes móveis (Chen & Kotz, 2000; Schilit et al., 1994) e para negociação de preferências e capacidades (Lemlouma & Layaïda, 2001). Uma tendência que se observa, justamente com o intuito de tornar o mecanismo o mais inter-operável possível, é a utilização de linguagens XML para descrição dos recursos (W3C, 1999a; W3C, 1999b) e de protocolos baseados em HTTP para consulta e negociação (Gudgin et al., 2001; Ohto & Hjelm, 1999). Esta tese não trata dos aspectos ligados à gerência de contexto, mas assume a sua existência para apoiar a formatação e, mais especificamente, a adaptação das apresentações dos hiperdocumentos. Como trabalho futuro, pretende-se integrar o formatador HyperProp, apresentado no Capítulo 5, a mecanismos externos de gerência de contexto.

De um modo geral, as características da infra-estrutura de exibição permitem guiar a adaptação dos documentos em termos de ajuste da qualidade dos conteúdos dos objetos apresentados. Em algumas situações, principalmente com relação às características dos dispositivos de E/S na plataforma de apresentação, as informações contextuais referentes à infra-estrutura podem também direcionar a escolha das características de apresentação (descritores) dos objetos e a escolha entre alternativas de tipos de mídia diferentes. Em contrapartida, os parâmetros relacionados ao usuário são mais propícios à adaptação da quantidade e tipo de informações contidas nos conteúdos dos documentos e à adaptação dos relacionamentos entre os objetos. Como este trabalho se propõe a focar nas questões de ajuste de qualidade, o mesmo irá centrar seu desenvolvimento na

PUC-Rio - Certificação Digital Nº 9916204/CA

adaptação dos documentos em função das características da infra-estrutura de exibição.

Para finalizar, é importante salientar que nem toda adaptação depende exclusivamente de informações externas aos documentos. O ajuste elástico das durações (ou mesmo das dimensões) dos objetos, comentado na Seção 2.1.1.1, é um exemplo de adaptação que é feito também com base nas especificações dos relacionamentos entre os nós.

2.1.3 Local e momento da adaptação

Para que um documento hipermídia seja adaptado em alguma das maneiras discutidas na Seção 2.1.1, é necessário que exista no sistema hipermídia um elemento capaz de analisar a descrição do documento em conjunto com as informações contextuais e, com esses dados, aplicar as estratégias de adaptação. Conforme comentado na Seção 1.1, em sistemas hipermídia distribuídos, as estratégias podem ser executadas tanto no ambiente de armazenamento (servidor) como no ambiente de execução (cliente), podendo inclusive estar presente nos dois ambientes paralelamente. Em alguns casos, pode ser também interessante a existência de um agente intermediário no sistema, atuando como um proxy, onde será incorporada também a funcionalidade de adaptação. A adaptação nesse elemento pode ser apropriada para reduzir o processamento em clientes e servidores e ao mesmo tempo reutilizar os resultados para mais de um cliente, oferecendo uma espécie de cache para os documentos já adaptados (Elson & Cerpa, 2001). Entretanto, é importante lembrar que, como mencionado na Seção 1.1, uma vez que as técnicas de adaptação relacionadas à qualidade da informação exibida devem estar próximas do usuário, é fundamental a presença dos mecanismos de adaptação no cliente hipermídia, mais especificamente no formatador.

Com relação ao momento em que a adaptação ocorre, é possível separar em duas opções: antes da apresentação do documento e durante a sua apresentação. A idéia principal é que as adaptações realizadas antes da exibição estão relacionadas às informações contextuais estáticas, sendo por isso essa opção também classificada como adaptação estática, enquanto a adaptação em tempo de execução

PUC-Rio - Certificação Digital Nº 9916204/CA

é também chamada de adaptação dinâmica, pois depende de parâmetros cujos valores se modificam ao longo da exibição do documento (Boll et al. 1999a; Boll et al., 1999b).

2.2
Requisitos para modelos de documentos hipermídia

Um modelo hipermídia deve, evidentemente, oferecer suporte à definição de documentos contendo objetos de diferentes tipos de mídia. Além disso, o modelo precisa possibilitar, ao menos, a especificação de relacionamentos entre os vários objetos, que permitam ao usuário, através de suas interações, percorrer o conteúdo de uma maneira não seqüencial.

Além dessas características obrigatórias, algumas outras são também desejáveis de serem encontradas em um modelo hipermídia. Em primeiro lugar, é interessante que o modelo permita que sejam especificadas, além das relações baseadas na interação do usuário, relacionamentos para sincronização tanto temporal como espacial dos objetos. Esses relacionamentos não devem ser limitados na quantidade de seus participantes, como acontece com os tradicionais elos de navegação da linguagem HTML, restrita a uma única origem e um único destino. Outro ponto desejável é que o modelo permita que os relacionamentos sejam definidos não apenas entre os objetos como um todo, mas entre partes internas de seus conteúdos, oferecendo assim uma granularidade mais fina para a sincronização dos nós.

A forma como as relações são definidas, principalmente as especificadas através de elos, traz conseqüências importantes para o modelo hipermídia. Documentos HTML, por exemplo, possuem os elos embutidos no conteúdo, o que gera limitações significativas (Anderson, 1997; Bouvin & Schade, 1999; Rodrigues et al., 2002). Sendo assim, é apropriado que os relacionamentos entre os nós possam estar separados do conteúdo dos objetos, trazendo, entre outros benefícios, uma possibilidade de reuso dos conteúdos sem uma herança obrigatória das relações.

Ainda com relação à questão dos relacionamentos de sincronização entre os objetos, é adequado que o modelo faça uma distinção entre dois tipos de relações: as relações de causalidade e as de restrição. As relações de causalidade envolvem

PUC-Rio - Certificação Digital Nº 9916204/CA

ações que devem ser executadas quando condições estabelecidas no relacionamento forem satisfeitas. Por exemplo, o início de um vídeo deve ocorrer quando uma imagem deixar de ser apresentada, ou o término de um áudio deve acontecer quando o usuário interagir com a aplicação. Como o próprio nome sugere, as relações de restrição estabelecem regras de sincronismo que o formatador hipermídia deve procurar respeitar. Por exemplo, um vídeo e um áudio, quando simultaneamente exibidos, devem terminar ao mesmo tempo. Como é possível perceber, os dois tipos de relações apresentam semânticas distintas e se complementam no oferecimento de um maior poder de expressão aos autores. Um exemplo da necessidade dos dois tipos de relação de sincronização está no uso comum que se faz das relações definidas por Allen (Allen, 1983) e ilustradas na Figura 2. Allen especificou treze possíveis relações que podem ser definidas entre intervalos e esse resultado tem sido utilizado como base na definição das relações de sincronismo temporal em modelos de documentos multimídia (Little & Ghafoor, 1990).

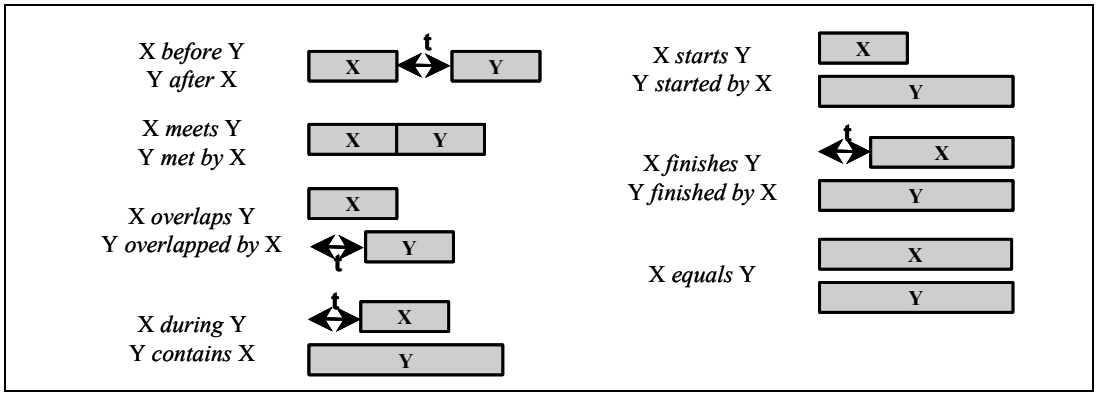


Figura 2 – Relações de Allen.

A questão da necessidade dos dois tipos de relação surge pelo fato das relações de Allen não expressarem precisamente a semântica desejada pelo autor para definir o sincronismo temporal entre dois intervalos de tempo, podendo gerar ambigüidades para o controle da apresentação (Duda & Keramane, 1995). Por exemplo, a relação *meets* apenas especifica que o final de um intervalo deve coincidir com o princípio de um outro, mas várias interpretações podem ser dadas. A relação pode ser considerada como sendo de restrição: o final da apresentação de *x* precisa coincidir com o início da apresentação de *y*. Contudo a relação poderia também ser vista com uma semântica de causalidade, permitindo inclusive

PUC-Rio - Certificação Digital Nº 9916204/CA

duas interpretações distintas: uma causalidade de iniciação (o final da exibição de *x* deve causar o início da exibição de *y*) ou uma causalidade de encerramento (o princípio da exibição de *y* deve ocasionar o término da apresentação de *x*). Sendo assim, é interessante que o modelo proveja meios para a especificação explícita dos dois tipos de relacionamentos de sincronização entre os objetos.

Outra característica em modelos de documentos hipermídia que merece destaque como sendo desejável é a existência de entidades que permitam agrupar os nós e seus relacionamentos, que possam ser reutilizadas em várias partes do documento, ou até mesmo em documentos distintos. A maneira usual de oferecer esse recurso é através da introdução nos modelos do conceito de nó de composição, já mencionado na Seção 2.1.1.1.

Conforme também discutido na Seção 2.1.1.1, uma outra característica desejável em modelos de documentos hipermídia é a separação dos parâmetros de exibição, a fim de torná-los independentes do objeto de dados. A idéia é que esses parâmetros possam ser associados aos nós conforme a conveniência. Essa separação permite, entre outras coisas, o reuso das mesmas características de apresentação em vários objetos e a modificação do estilo de apresentação sem que seja preciso alterar os objetos de dados. Esse tipo de separação pode servir também como uma facilidade para implementar adaptações de documentos pela simples modificação das características de exibição. Exemplos importantes dessa abordagem são as linguagens de estilo padronizadas para as páginas HTML (W3C, 1996; W3C, 1998a) e os *layouts* espaciais da linguagem SMIL (W3C, 2001a).

Ainda de acordo com o que foi comentado na Seção 2.1, é desejável que o sistema hipermídia possua recursos para a adaptação dos documentos e de suas apresentações. Porém, para que a maioria das técnicas de adaptação possam ser aplicadas, é necessário que o modelo permita a especificação de certas informações de uma maneira flexível. Por exemplo, para que possa ser realizado o ajuste das durações dos objetos, as mesmas devem ser especificadas não mais de uma maneira rígida com um único valor, mas sim com uma faixa de valores aceitáveis. Mais interessante ainda se esses valores forem descritos através de uma função de custo, oferecendo assim uma métrica que permita à estratégia de adaptação decidir qual o melhor ajuste para uma dada apresentação e contexto de

PUC-Rio - Certificação Digital Nº 9916204/CA

exibição (Bachelet et al., 2000; Buchanan & Zellweger, 1993b; Kim & Song, 1995).

Ainda com relação à adaptação, para que o sistema hipermídia possa explorar a idéia da seleção de alternativas, é fundamental que algum tipo de meta-informação esteja associado a cada uma das várias possibilidades de conteúdo, de relacionamento ou mesmo de descrição das características de apresentação (Seção 2.1.1). Esses metadados podem ser explicitamente informados pelo autor ou automaticamente calculados pelo sistema, devendo estar disponíveis no momento da apresentação do documento, a fim de permitir que as técnicas de adaptação saibam como classificar e escolher entre as diferentes alternativas. Mais ainda, é necessária a existência de uma base de relacionamentos estabelecendo as associações entre as várias alternativas. Essas bases podem ser simples composições, tais como o *switch* das linguagens SMIL (W3C, 2001a) e NCL (Muchaluat-Saade, 2003), ou conjuntos de relações mais amplas e complexas, como as propostas em (Boll et al., 1999a).

Evidentemente, também é importante que exista um modelo para descrição das informações contextuais, pois, na verdade, conforme discutido na Seção 2.1.2, a adaptação deverá ser guiada pelos parâmetros do contexto em que a aplicação estiver inserida. No entanto, é importante observar que o modelo para estabelecimento das associações entre alternativas e o modelo para descrição do contexto não precisam fazer parte diretamente do modelo de hiperdocumentos. Esses modelos podem funcionar como camadas que venham a ser superpostas ao modelo de documento, oferecendo diferentes níveis de visões para as informações.

Os requisitos apresentados nesta seção buscam estabelecer as características desejáveis de serem encontradas nos modelos de dados que formam a base das informações passadas para o ambiente de execução hipermídia (Figura 1). Esse ambiente, por sua vez, deve ser projetado de modo a ser capaz de controlar as apresentações dos documentos com todos os recursos mencionados.

Internamente, o ambiente de execução (em especial, o formatador) deve estabelecer um modelo para construção de suas estruturas de dados, a fim de gerar os documentos formatados a partir das especificações disponíveis, conforme ilustrado na Figura 3. Esse modelo interno pode ser entendido como o modelo de execução dos hiperdocumentos e, conforme discutido e ilustrado na figura, em

PUC-Rio - Certificação Digital Nº 9916204/CA

alguns casos, será na verdade o resultado do processamento de informações provenientes de mais de um modelo. Dos vários modelos de entrada apresentados na figura que podem influenciar no modelo de execução, esta tese concentra-se nos aspectos relacionados ao modelo de documentos.

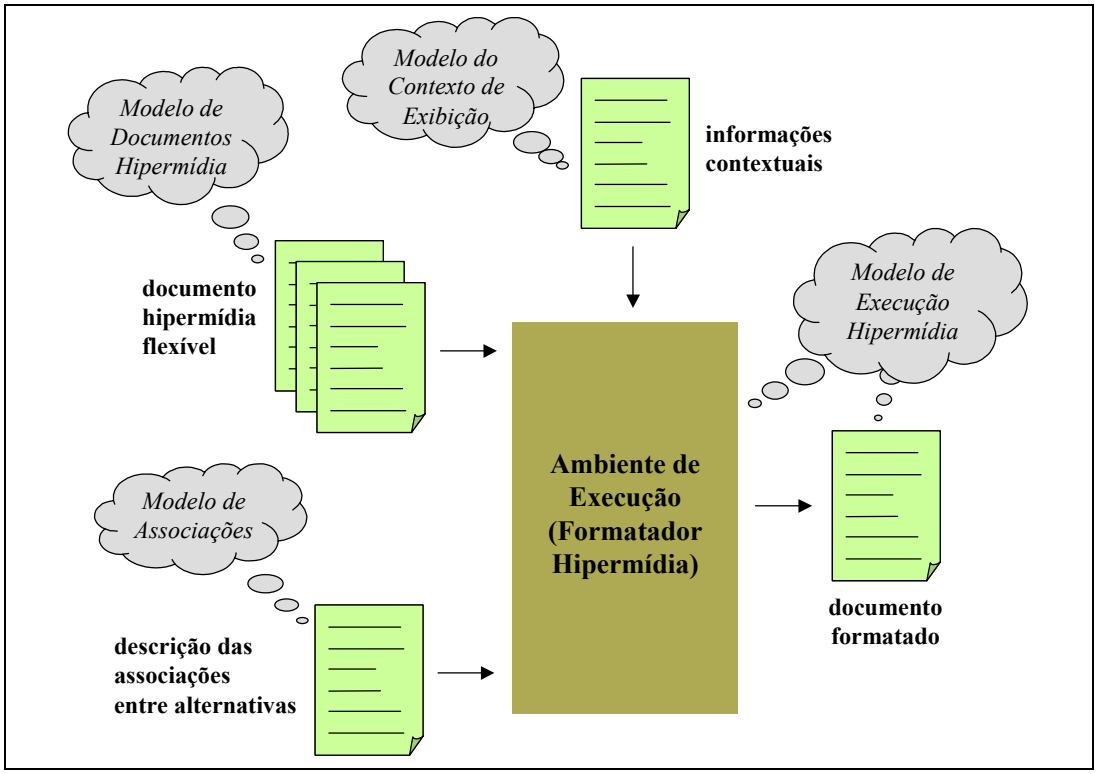


Figura 3 – Modelos das informações fornecidas a um formatador hipermídia para controle da execução dos hiperdocumentos.

O modelo de documentos hipermídia normalmente corresponde ao modelo utilizado para autoria[4]. Pela variedade de modelos desse tipo citados na Seção 2.1 e assumindo que seja desejável que um ambiente de execução possa controlar a apresentação de mais de um tipo de modelo de documento, torna-se necessário que, na maioria das vezes, a execução seja precedida por uma fase de transformação para conversão de formatos, conforme será melhor discutido no Capítulo 3 e exemplificado no Capítulo 5. Dois motivos justificam a necessidade dessa transformação. O primeiro é a necessidade de colocar certas abstrações para os autores, que podem ser eliminadas quando o documento se aproxima da

[4] Muitas vezes o modelo de autoria passado ao autor pode ter um nível de abstração maior, mas terá sempre por base um modelo de autoria tal qual os já exemplificados.

PUC-Rio - Certificação Digital Nº 9916204/CA

máquina que irá controlar a sua exibição. O segundo é uma simples questão subjetiva de preferência, refletida na variedade de propostas e padrões de modelos.

2.3
Modelo de documentos para controle da execução

As entidades apresentadas nesta seção formam a base do modelo de execução utilizado para a construção do framework de formatação de documentos hipermídia. Mais especificamente, esse modelo serviu também como referência para a implementação do formatador HyperProp, a ser apresentada no Capítulo 5. O modelo de execução a ser descrito baseia-se em conceitos do modelo NCM (Casanova et al., 1991; Soares et al., 1995; Souza, 1997; Soares et al., 2000; Soares et al., 2003), Modelo de Contextos Aninhados (*Nested Context Model*), mas sua aplicação é mais geral, podendo ser utilizado como modelo de execução para outros modelos de documentos hipermídia, conforme será também comentado no Capítulo 5, quando serão apresentados os conversores desenvolvidos como parte deste trabalho. Maiores informações a respeito do modelo NCM podem ser encontradas nas referências já citadas, em particular, no relatório técnico que descreve todas as entidades do modelo de maneira detalhada (Soares et al., 2003).

O modelo de execução proposto segue o paradigma de orientação a objetos e baseia a lógica e o sincronismo das apresentações em máquinas de estados de eventos, relacionamentos de causalidade entre eventos e também relacionamentos de restrição entre eventos. O modelo define uma estrutura básica com alguns pontos de flexibilização (*hotspots*), que devem ser preenchidos nas implementações particulares dos ambientes de execução. O objetivo principal é tornar a proposta genérica e capaz de atender a uma variedade de formatadores.

A descrição das classes do modelo e da maneira como essas classes se relacionam é feita com o auxílio da linguagem UML (Rumbaugh et al., 1999). Os diagramas são apresentados progressivamente, com o intuito de evitar uma sobrecarga visual nas figuras. Além disso, também com o objetivo de tornar os diagramas mais legíveis, apenas os principais atributos e métodos das classes foram destacados. A descrição completa das classes, atributos e métodos do modelo de execução pode ser encontrada em (Rodrigues, 2003).

PUC-Rio - Certificação Digital Nº 9916204/CA

2.3.1
Objetos de execução

O principal elemento no modelo é o *objeto de execução*, que representa uma instância de nó hipermídia a ser exibida. Sendo assim, os objetos de execução no formatador reúnem, para cada fragmento de informação (nó), todos os seus dados, inclusive as informações relativas às suas características de apresentação. Um objeto de execução (classe *ExecutionObject*) possui quatro atributos principais: identificador único, objeto de dados, descritor e lista de eventos, conforme ilustrado no diagrama da Figura 4.

O *identificador único* (classe *UID*), como o próprio nome sugere, permite distinguir univocamente o objeto de execução dentro do ambiente de formatação. Essa classe deve ser estendida na implementação de cada ambiente de execução particular, precisando apenas oferecer um método que permita comparar o identificador com um outro identificador do mesmo tipo. É responsabilidade da implementação do formatador garantir a unicidade dos identificadores.

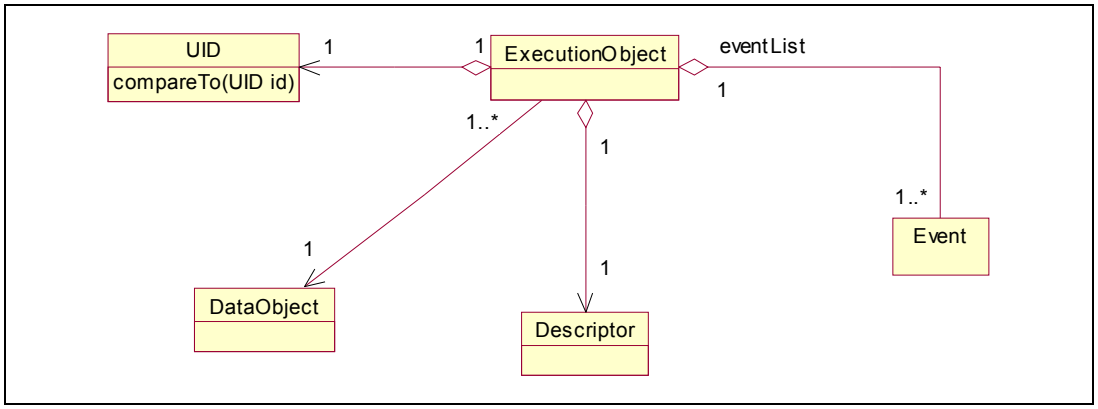


Figura 4 – Diagrama de classes para os objetos de execução.

O atributo *objeto de dados* (classe *DataObject*) guarda uma referência para o nó hipermídia (sem as informações de apresentação), enquanto o *descritor* (classe *Descriptor*) contém as informações referentes à exibição do objeto. Por fim, a *lista de eventos* (composta por objetos da classe *Event*) define as unidades básicas para o controle dos relacionamentos de sincronização temporal e espacial.

Cabe mencionar que um mesmo objeto de dados pode ser compartilhado por mais de um objeto de execução, oferecendo assim uma possibilidade de reuso de suas informações. Por outro lado, descritores e eventos pertencem a um único

PUC-Rio - Certificação Digital Nº 9916204/CA

objeto de execução. Existem, no entanto, algumas informações de apresentação contidas nos descritores que podem ser compartilhadas por mais de um descritor. As próximas subseções trazem informações mais detalhadas das classes para criação de objetos de dados, descritores e eventos.

2.3.1.1 Objetos de dados

Todo objeto de dados possui três atributos principais: um identificador único, um conteúdo e uma lista de âncoras, conforme ilustrado no diagrama de classes apresentado na Figura 5.

Da mesma forma que o objeto de execução, o objeto de dados possui um atributo da classe *UID* que identifica o objeto de dados de maneira única no ambiente de execução. Novamente, essa classe deve ser especializada em cada implementação para definir concretamente o identificador. É possível que a mesma especialização seja utilizada para identificar tanto os objetos de dados como os objetos de execução. No entanto, os espaços de nomes para os identificadores devem ser tratados de maneira independente pelo formatador.

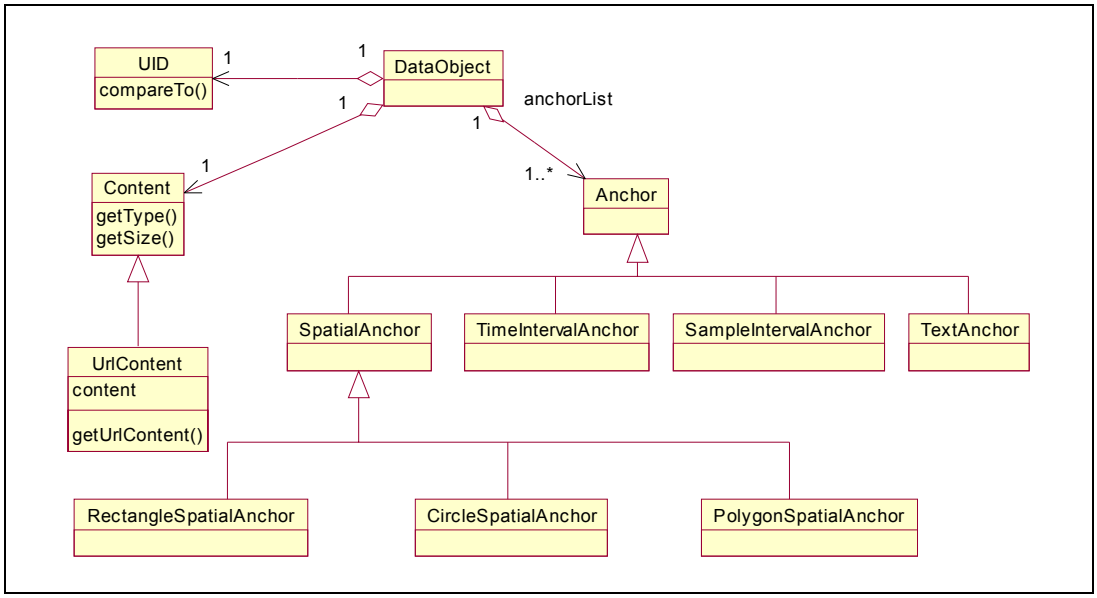


Figura 5 – Diagrama de classes para os objetos de dados.

O *conteúdo* de um objeto de dados identifica um conjunto de unidades de informação, enquanto as *âncoras* identificam subconjuntos de unidades de informação marcadas desse conteúdo. As noções exatas de conteúdo (unidades de

PUC-Rio - Certificação Digital Nº 9916204/CA

informação) e de âncora (unidades de informação marcadas) dependem da classe de nó hipermídia que venha a derivar o objeto de dados. Por exemplo, um vídeo pode ter como unidade de informação um quadro, enquanto um texto pode ter como unidade de informação um caractere ou uma palavra.

O modelo define uma classe abstrata *Content* que deve ser estendida em função dos objetos exibidos. O modelo também define uma subclasse de conteúdo, denominada *UrlContent*, que identifica as unidades de informação através de uma URL (Berners-Lee et al., 1994b). Todavia, nada impede que outras subclasses de conteúdo sejam definidas em implementações particulares de formatador. Como exemplo, uma implementação de formatador pode definir uma subclasse de conteúdo que mantenha uma cópia das unidades de informação no próprio objeto de dados, permitindo que o objeto funcione como uma espécie de memória cache. Uma outra subclasse de conteúdo que pode ser útil é a que especifica uma série de outros objetos de dados, representando um nó de composição. A idéia de criar um objeto de execução associado a uma composição é permitir que, na apresentação do documento, a sua estruturação possa ser exibida como parte da informação a ser oferecida ao usuário. Isso porque pode ser útil para o usuário que, além das mídias, seja exibido um mapa do documento para auxiliá-lo em sua navegação, reduzindo, ou mesmo eliminando, o problema de desorientação (Muchaluat-Saade, 1996; Muchaluat-Saade et al., 1998).

Todo conteúdo deve também implementar métodos que descrevam os seus dados. O método *getType* deve retornar o tipo MIME (Freed & Borenstein, 1996) do conteúdo, enquanto o método *getSize* deve informar o tamanho do conteúdo em bytes. Ambos os métodos podem retornar um valor *não disponível*, quando não for possível determinar o valor de retorno. No caso da classe *UrlContent*, existe o método adicional *getContentUrl*, que retorna o endereço e o protocolo para respectiva localização e busca do conteúdo.

Similar ao conteúdo, o modelo define uma classe abstrata *Anchor*, cujas especializações devem ser feitas em função das unidades de informação do conteúdo passíveis de serem marcadas. As subclasses de âncora principais também são especificadas pelo modelo, conforme ilustra o diagrama de classes da Figura 5. A classe *TextAnchor* permite definir regiões em conteúdos do tipo texto, identificando a posição e a *string* da âncora. As classes *TimeIntervalAnchor* e

PUC-Rio - Certificação Digital Nº 9916204/CA

SampleIntervalAnchor permitem descrever regiões temporais. A primeira classe o faz pela especificação do instante de início e fim (em segundos), enquanto a segunda utiliza a posição das amostras. A classe *SpatialAnchor* e suas respectivas subclasses definem regiões espaciais em conteúdos como imagens e vídeos. Outras subclasses de âncora para, por exemplo, identificar objetos em vídeos e imagens podem vir a ser definidas em implementações específicas de formatadores.

Cabe salientar que combinações desses vários tipos de âncora mencionados no parágrafo anterior podem ser úteis, tais como a definição de uma área retangular em um vídeo durante um determinado intervalo de tempo. Contudo, combinações a partir das classes já definidas não requerem a criação de novas subclasses, pois, conforme será apresentado na definição dos elos (Seção 2.3.2), a possibilidade de definir relações entre duas ou mais âncoras (na realidade entre dois ou mais eventos) oferece, no modelo, poder de expressão equivalente.

Uma observação relevante com respeito à definição das subclasses de âncoras no modelo é que tanto as âncoras espaciais como as âncoras baseadas em amostras podem identificar suas fronteiras de maneira percentual, ao invés de discriminar os pixels ou a posição exata das amostras.

2.3.1.2 Descritores

O descritor reúne as informações referentes às características de exibição do objeto de execução. Todo descritor possui como principais atributos a identificação de uma *ferramenta de exibição* (classe *PresentationTool*) e de uma *região espacial de apresentação* (classe *LayoutRegion*). O modelo permite que uma mesma ferramenta controle a exibição do conteúdo de mais de um objeto de execução e, da mesma forma, que uma mesma região espacial seja compartilhada por mais de um objeto (mais de um descritor).

A escolha da *ferramenta de exibição* e sua instanciação são responsabilidades do formatador hipermídia, tendo como base a especificação do documento e a lista de ferramentas disponíveis discriminadas na plataforma de exibição (informação de infra-estrutura no contexto de exibição – Seção 2.1.2). Nada impede que seja utilizada, na apresentação do conteúdo de um objeto de

PUC-Rio - Certificação Digital Nº 9916204/CA

dados, uma ferramenta de um tipo de mídia completamente diferente do tipo de mídia do conteúdo. Por exemplo, um objeto de dados do tipo texto pode ser associado a um descritor que defina uma ferramenta de exibição sintetizadora de voz, e com isso ser apresentado como um áudio.

Conforme pode ser percebido na Figura 6, a *região espacial* (classe *LayoutRegion*) faz parte de uma hierarquia de classes mais ampla, que define a estrutura espacial da apresentação de um documento. Qualquer que seja o hiperdocumento a ser apresentado, deve haver no ambiente de execução uma estrutura espacial (instância da classe *Layout*) definida para exibição dos objetos de execução. Essa estrutura especifica um conjunto de janelas (atributo *windowSet*, da classe *LayoutWindow*) e uma janela *default* (atributo *defaultWindow*). O conjunto de janelas é derivado, quando especificado, da descrição do documento hipermídia de entrada (Figura 3), enquanto a janela *default* é criada pelo próprio formatador. Sua utilidade será descrita mais adiante. As definições apresentadas nesta seção são similares às encontradas na linguagem SMIL e na referência (Moura, 2001).

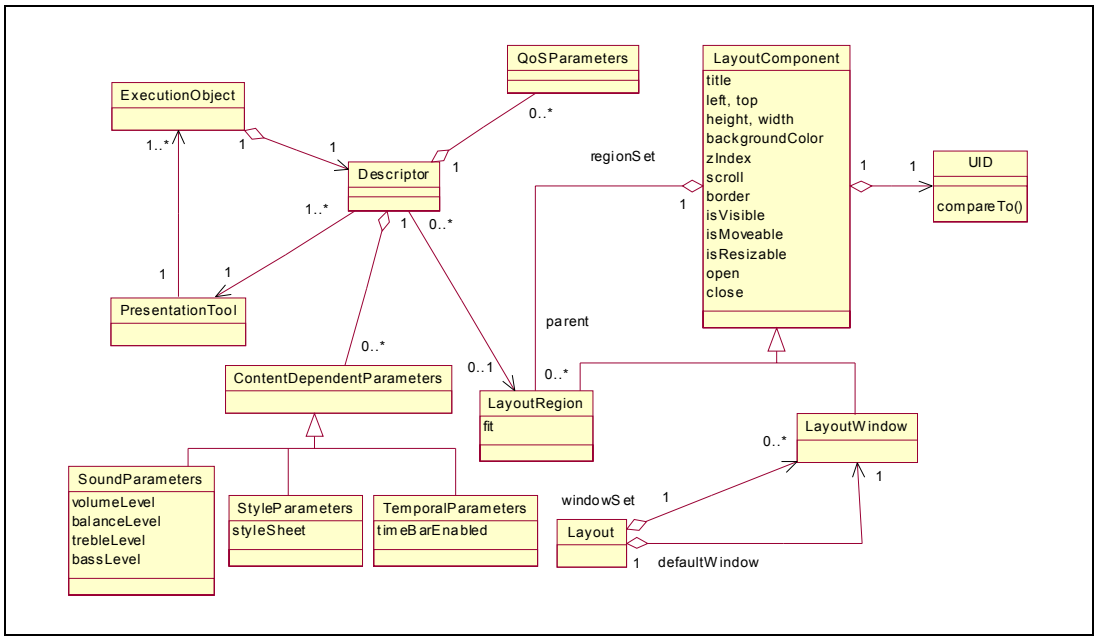


Figura 6 – Diagrama de classes para os descritores.

Janelas (classe *LayoutWindow*) e regiões herdam de uma superclasse comum, denominada *LayoutComponent*. Objetos dessa classe possuem um identificador único (classe *UID*, análoga à que foi definida para os objetos de execução e de dados) e um nome (atributo *name*). Além disso, a classe

PUC-Rio - Certificação Digital Nº 9916204/CA

LayoutComponent define atributos que permitem especificar largura (*width*) e altura (*height*) de uma área retangular[5] e coordenadas de posicionamento (atributos *top* e *left*) do componente. Qualquer uma dessas informações pode ser especificada tanto de maneira absoluta (em pixels, por exemplo) como de maneira proporcional.

A classe *LayoutComponent* possui também uma série de outros atributos referentes às propriedades de exibição dos objetos: *backgroundColor*, *zIndex*, *scroll*, *border*, *isVisible*, *isMoveable*, *isResizable*, *open* e *close*. O atributo *backgroundColor* serve para definir uma cor de fundo para o componente de layout. O atributo *zIndex* é utilizado para determinar a prioridade de superposição quando mais de um componente tiver que ser exibido em uma mesma área. Os componentes com valores de *zIndex* menores aparecem sobrepostos aos componentes com valores maiores. O atributo *scroll* define o comportamento da barra de rolagem na exibição do componente e pode receber um, dentre cinco valores: componente não possui barra de rolagem, componente possui barra de rolagem apenas vertical, componente possui barra de rolagem apenas horizontal, componente possui sempre barras de rolagem vertical e horizontal e, finalmente, o componente só possui barra de rolagem (tanto vertical quanto horizontal) quando for necessário. O atributo *border* define a espessura da borda para o componente de layout. O valor zero significa a ausência de borda para o componente. Os atributos, *isVisible*, *isMoveable* e *isResizable* informam, respectivamente, se o componente está ou não visível, se pode ser ou não modificado de posição e se pode ter ou não as suas dimensões alteradas. Os atributos *open* e *close* serão comentados mais adiante.

Além das informações referentes às características de exibição, todo componente de *layout* (janela ou região) possui um conjunto de regiões (atributo *regionSet*). Isso também implica que toda região deve estar contida em um componente de layout, identificado pelo atributo *parent* da classe *LayoutRegion*.

[5] Para tornar o modelo mais genérico, janelas, assim como regiões, poderiam ser especificadas nas mais diversas formas geométricas. No entanto, por entender que essa generalidade não seria de grande utilidade e primando pela simplicidade, o modelo aqui proposto, da mesma forma que em (Moura, 2001), limita a representação espacial de componentes de apresentação à forma retangular.

PUC-Rio - Certificação Digital Nº 9916204/CA

Nesse momento, cabe comentar a respeito dos atributos *open* e *close*, que determinam o comportamento de exibição de um componente de layout quando o mesmo não se encontra ativo. Um componente de layout está *ativo*, se existe algum objeto de execução a ele associado e sendo exibido, ou se alguma região diretamente ou recursivamente contida nele tem algum objeto de execução a ela associado e sendo exibido. Dessa forma, o atributo *open* determina se o componente deve ou não ser exibido mesmo que ainda não tenha se tornado ativo e o atributo *close* especifica se o componente deve continuar sendo exibido mesmo depois que deixa de estar ativo.

A classe *LayoutRegion*, conforme descrito, é uma especialização de *LayoutComponent* e é justamente o tipo de componente de layout ao qual os objetos de execução estarão associados. Essa classe possui um atributo adicional denominado *fit*. Esse atributo define o comportamento da região quando as suas dimensões não coincidirem com as do conteúdo do objeto de execução associado a ela. As possibilidades previstas pelo modelo são:

- redimensionar o objeto para que suas dimensões coincidam com as da região;
- desenhar o objeto a partir do canto superior esquerdo da região e preencher os espaços restantes com a cor de fundo da região, cortando as partes que ultrapassem as dimensões da região e aplicando a política de rolagem (*scroll*) definida;
- desenhar o objeto a partir do canto superior esquerdo da região e redimensioná-lo, conservando a proporção entre suas dimensões, até que todo o objeto esteja visível na região e o espaço restante esteja preenchido com a cor de fundo; ou
- desenhar o objeto a partir do canto superior esquerdo da região e redimensioná-lo, conservando a proporção entre suas dimensões, até que toda região esteja ocupada pelo objeto. Para a área do objeto cortada, deve ser aplicada a política de rolagem definida na superclasse *LayoutComponent.*

PUC-Rio - Certificação Digital Nº 9916204/CA

A classe *LayoutWindow* é uma especialização de *LayoutComponent* que não está contida em nenhum outro componente de layout e cujo conjunto de regiões deve possuir pelo menos uma região especificada.

Retornando à questão da janela *default* no layout do documento, essa janela é automaticamente criada pelo formatador e possui duas utilidades. Definir o espaço para exibir objetos de execução de documentos que não possuam um layout espacial especificado, assim como exibir os objetos de execução de documentos que possuam um layout espacial especificado, mas para os quais o identificador da região não possa ser encontrado nesse layout.

Uma vez que a região de layout está relacionada com as características espaciais visuais da exibição de um objeto de execução, esse atributo é opcional nos descritores. Objetos de áudio, por exemplo, podem dispensar a existência de uma região, apesar de ser perfeitamente possível que se deseje exibir uma representação visual da onda sonora.

Todo descritor possui, opcionalmente, além da ferramenta de exibição e da região espacial, um conjunto de parâmetros de qualidade de serviço (objetos da classe *QoSParameters*) que são deixados como pontos de flexibilização no modelo e devem ser especificados em implementações particulares de formatadores que saibam lidar com essas informações. Exemplos de atributos que poderiam ser definidos em especializações da classe *QoSParameters* seriam: latência máxima para o início da exibição do objeto, taxa de exibição desejável e o mínimo tolerável, percentual de erro aceitável na entrega dos dados etc.

O modelo define também uma classe abstrata *ContentDependentParameters*, que permite agrupar atributos relevantes para exibição de tipos particulares de conteúdo. Um descritor pode possuir um ou mais atributos dessa classe. Além disso, implementações de formatadores podem estender a hierarquia para oferecer suporte a outros atributos. O modelo define a subclasse *SoundParameters*, para reunir atributos relevantes para mídias sonoras (áudio e vídeo), *StyleParameters*, para objetos que utilizem folhas de estilo (W3C, 1996; W3C, 1998a) na formatação de seu conteúdo, e *TemporalParameters* para objetos que desejem possuir uma barra de controle visual do andamento da exibição.

PUC-Rio - Certificação Digital Nº 9916204/CA

Cabe ao formatador, a partir da descrição da plataforma de exibição (informação do contexto de exibição – Seção 2.1.2), realizar a associação entre os descritores e os dispositivos físicos (monitor, monitor de TV, placa de som etc.). Essa associação pode ser feita inclusive de modo distribuído, quando o ambiente de execução for composto por mais de uma máquina. Além disso, o formatador deve se preocupar em manter a consistência entre a ferramenta de exibição e o descritor dos objetos de execução. Exemplo de inconsistência seria a definição de um exibidor de imagem estática para um descritor de áudio.

2.3.1.3 Eventos

O evento é a unidade básica de sincronização definida no modelo de execução, estando contido na *lista de eventos* do objeto de execução. A definição de evento no modelo segue a proposta de Pérez-Luque (Pérez-Luque & Little, 1996), onde o mesmo é descrito como uma ocorrência no tempo que pode ser instantânea ou durar um período de tempo. Sendo assim, existe no modelo de execução dois tipos de eventos: aqueles que são *instantâneos* e aqueles que apresentam *duração não desprezível* (*não instantâneos*). Os eventos também podem ser classificados em relação à interação: eventos do tipo *interativos* possuem sua ocorrência obrigatoriamente relacionada com alguma ação do usuário, enquanto os eventos *não interativos* têm suas ocorrências desvinculadas da interatividade do usuário, dependendo apenas do controle do próprio ambiente de execução. Essas duas classificações dos tipos de evento podem ser combinadas. Por exemplo, eventos podem ser instantâneos/interativos, não instantâneos/interativos, instantâneos/não interativos e não instantâneos/não interativos. Os exemplos para essas possibilidades ficarão mais claros ao longo da seção.

Na Figura 7 é possível observar o diagrama de classes para as entidades do modelo relacionadas à definição dos eventos. O modelo estabelece uma interface *EventInterface*, a partir da qual derivam os quatro tipos de evento mencionados no parágrafo anterior: *NonInstantaneousEvent* e *InstantaneousEvent* para, respectivamente, os tipos de evento não instantâneo e instantâneo, e *NonInteractiveEvent* e *InteractiveEvent* para modelar, respectivamente, os eventos

PUC-Rio - Certificação Digital Nº 9916204/CA

do tipo não interativo e do tipo interativo. O modelo também define uma classe *Event*, que implementa a interface *EventInterface*, a partir da qual derivam todas as subclasses de evento do modelo, assim como novas subclasses que venham a ser criadas.

Independente do tipo, existem duas subclasses de evento definidas no modelo: eventos baseados em âncora (*AnchorEvent*) e eventos baseados em atributo (*AttributeEvent*). *Eventos baseados em âncora* possuem um atributo que identifica uma âncora do objeto de dados (fragmento do conteúdo do objeto de dados), enquanto *eventos baseados em atributo* possuem um atributo que identifica um outro atributo do objeto de execução. Entende-se por atributo de um objeto de execução qualquer atributo diretamente definido no objeto ou que possa ser alcançado pela navegação em profundidade a partir de seus atributos. Por exemplo, tanto o atributo *dataObject* como o atributo *descriptor.layoutRegion.top* podem ser definidos em um evento baseado em atributo.

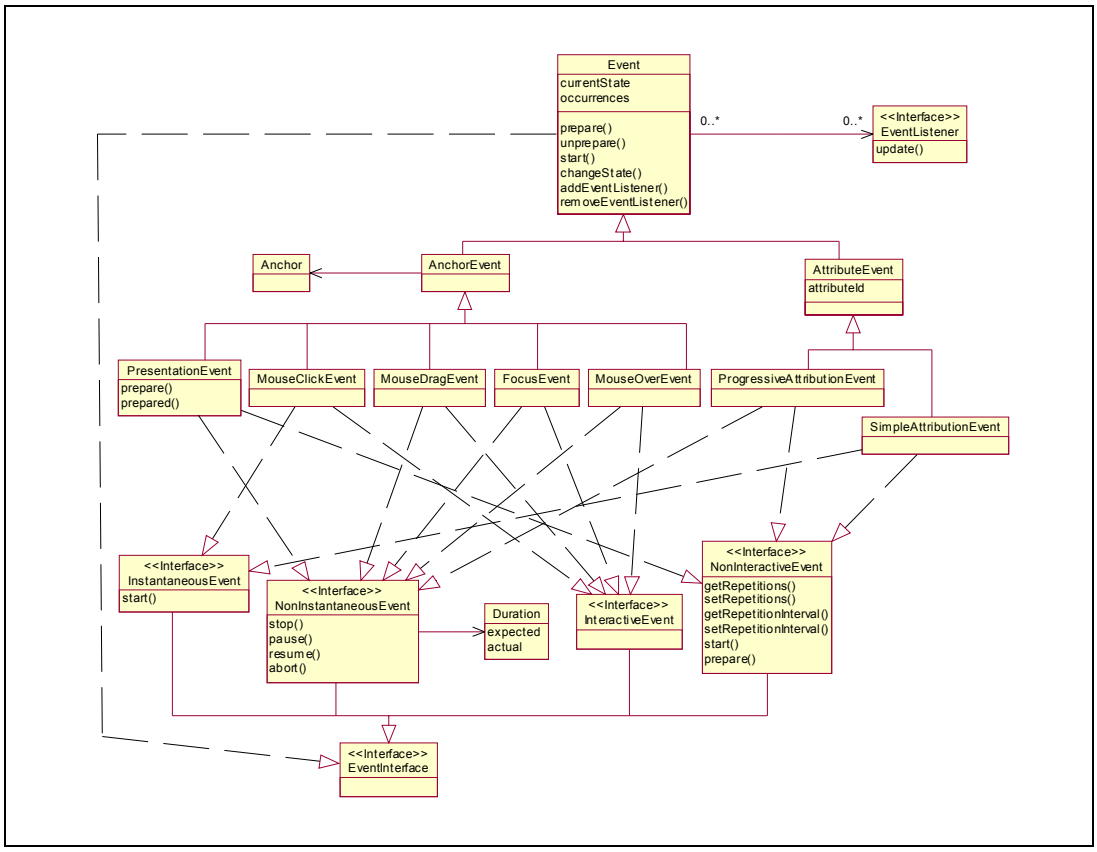


Figura 7 – Diagrama de classes para os eventos.

Em uma implementação de formatador hipermídia, devem ser definidas especializações das subclasses evento baseado em âncora e evento baseado em

PUC-Rio - Certificação Digital Nº 9916204/CA

atributo, que especifiquem semânticas para os eventos na apresentação do documento. Algumas especializações já são definidas no próprio modelo, pois serão necessárias em qualquer que seja o sistema hipermídia. A primeira especialização é a classe *evento de exibição* (ou *evento de apresentação*) – classe *PresentationEvent*, subclasse de evento baseado em âncora, cujo objetivo é permitir especificar unidades básicas de sincronização associadas com a apresentação de fragmentos dos conteúdos dos objetos de dados, ou seja, com a apresentação dos subconjuntos de unidades de informação marcadas. A outra especialização que deve existir em qualquer implementação de formatador é a classe *evento de seleção* (*MouseClickEvent*), também subclasse de evento baseado em âncora. Essa subclasse permite estabelecer sincronizações que dependam da interação (clique, mais especificamente) do usuário sobre as âncoras dos objetos de dados.

Além dessas duas principais classes, o modelo define outras especializações que podem ser úteis em apresentações hipermídia. *Evento de arraste* (classe *MouseDragEvent*), *evento de foco* (classe *FocusEvent*), *evento de mouse sobre a âncora* (classe *MouseOverEvent*) são subclasses de evento baseado em âncora. Para a classe de eventos baseados em atributos, o modelo define duas subclasses: *atribuição instantânea simples* (*SimpleAttribution*), para permitir sincronizações baseadas em valores dos atributos (ou mesmo mudança desses valores), e *atribuição progressiva* (*ProgressiveAttribution*), para, por exemplo, definir animações nas apresentações (Moura, 2001). Adicionalmente, novas classes podem ser criadas em implementações particulares de formatadores, conforme as necessidades dos documentos e de suas apresentações.

Qualquer que seja a especialização de evento, deve estar associada à respectiva classe uma combinação dos tipos (ou interfaces) definidos no início desta seção. Por exemplo, eventos de exibição, além de serem baseados em âncora, são também eventos do tipo não instantâneo e não interativo. Eles são não instantâneos porque sempre existe uma duração associada com a exibição da âncora (mesmo que essa duração seja imperceptível para o usuário – um quadro de vídeo, por exemplo). Além disso, eles são não interativos porque a exibição

PUC-Rio - Certificação Digital Nº 9916204/CA

efetiva independe da intervenção do usuário[6]. Os eventos baseados em ações do usuário com o mouse são sempre interativos. Alguns podem ter duração (*arraste*, *foco*, *mouse-over*) e outros podem ser instantâneos (*clique*).

Existe sempre uma máquina de estados associada ao evento, para qualquer que seja a sua classe. Os estados e transições de estados, juntamente com os valores de atributos dos eventos e do objeto de execução, formam a base para a definição dos relacionamentos de sincronização, que serão detalhados na Seção 2.3.2. Além disso, toda ação executada durante a apresentação do documento é feita sobre algum evento e, normalmente, reflete em uma transição na máquina de estados desse evento.

As máquinas de estados dos eventos variam em função da classe de evento, mas existem três estados que toda máquina obrigatoriamente deve possuir: *dormindo*, *preparado* e *ocorrendo*. Intuitivamente, o evento encontra-se no estado *ocorrendo* quando sua semântica na apresentação do hiperdocumento efetivamente acontece. Por exemplo, um evento de exibição entra no estado *ocorrendo* quando o subconjunto de unidades de informação marcadas, definidas pela âncora do evento, são apresentadas pela ferramenta de exibição. Os eventos de atribuição permanecem no estado *ocorrendo* enquanto a operação de atribuição durar. Os eventos instantâneos permanecem no estado *ocorrendo* por um tempo infinitesimal (por exemplo, o instante do clique do usuário ou o instante da atribuição). Diz-se que *um evento ocorre toda vez que entra em seu estado ocorrendo* e denomina-se *ocorrência do evento* a fase em que o evento encontra-se no estado *ocorrendo*.

Um evento no estado *preparado* encontra-se habilitado a mudar instantaneamente para o estado *ocorrendo*. Por outro lado, o evento no estado *dormindo* está impossibilitado de passar para o estado *ocorrendo* sem que antes seja realizado um procedimento de preparação das suas unidades de informação ou atributos, efetuada por uma chamada ao método *prepare* da classe *Event*. O evento também pode ser colocado de volta no estado *dormindo*, através de uma

[6] O conceito de não interatividade do evento está associado com o elemento que faz com que o evento efetivamente mude de estado durante a execução do documento. Um vídeo pode ter a sua apresentação condicionada à seleção de um botão por parte do usuário. No entanto, o evento de exibição do vídeo permanece como não interativo, pois a sua exibição efetiva depende do controle da ferramenta de exibição, e não do usuário.

PUC-Rio - Certificação Digital Nº 9916204/CA

invocação ao método *unprepare*, também definido na classe *Event*. O método *start* coloca o evento no estado *ocorrendo*, desde que ele se encontre no estado *preparado*. O comportamento padrão estabelece ainda que, quando o evento deixa o seu estado *ocorrendo*, ele retorna para o estado *preparado*. O evento também oferece um método *setState*, que pode ser útil em extensões da classe *Event* e do funcionamento da máquina de estados.

Baseado na máquina de estados, todo evento possui dois atributos, denominados *estado corrente* e *ocorrências*, cujos valores são utilizados pelo formatador para coordenar a sincronização das apresentações. Como o próprio nome sugere, o atributo *estado corrente* de um evento guarda o seu estado na máquina em um dado momento. Já o atributo *ocorrências* conta o número de vezes que o evento deixa o estado *ocorrendo* e retorna para o estado *preparado*. Eventos do tipo não interativos (interface *NonInteractiveEvent*) também devem manter dois atributos adicionais: *repetições* e *tempo de espera entre repetições*. *Repetições* determina o número de vezes seguidas que ainda restam para que o evento ocorra. Esse atributo pode conter um valor finito ou o valor *indefinido*, que levará a uma execução cíclica do evento, até que a mesma seja interrompida. Toda vez que um evento não interativo retorna para o estado *preparado*, o seu atributo *repetições* deve ser consultado. Se o valor for maior que zero, uma nova chamada ao método *start* deve ser realizada para que o evento ocorra novamente. O atributo *tempo de espera entre repetições* define um intervalo de pausa entre as ocorrências sucessivas. A interface *NonInteractiveEvent* também define duas novas opções para os métodos *prepare* e *start*, que recebem como parâmetros os valores para iniciar os dois novos atributos (*repetições* e *tempo de espera entre repetições*) a serem mantidos.

Os eventos do tipo não instantâneos possuem a noção de duração associada. A duração é tratada no modelo como uma classe que possui dois atributos: *duração esperada* (*expectedDuration*) e *duração efetiva* (*actualDuration*). O primeiro atributo (*duração esperada*) guarda o tempo previsto pelo formatador para a próxima ocorrência do evento. A *duração efetiva*, por sua vez, guarda o tempo efetivo da última ocorrência do evento. A duração esperada de um evento pode ser especificada como indeterminada, quando o formatador não tiver como prever o tempo para que o evento permaneça no estado *ocorrendo*.

PUC-Rio - Certificação Digital Nº 9916204/CA

Como uma classe do modelo, a duração pode ser estendida para representar descrições temporais mais complexas. Por exemplo, uma subclasse de duração pode ser especificada permitindo que os valores esperados sejam definidos como intervalos. Outras subclasses podem possibilitar descrever as durações com funções de custo (Bachelet et al., 2000; Buchanan & Zellweger, 1993b; Kim & Song, 1995), oferecendo assim métricas para o ajuste das durações. Exemplos dessas subclasses de durações mais flexíveis serão tratados no Capítulo 4, quando será abordada a adaptação dos tempos elásticos dos documentos.

Os eventos não instantâneos também possuem dois novos estados definidos em suas máquinas de estados: *suspenso* e *abortado*. Um evento é colocado no estado *suspenso*, quando uma chamada a seu método *pause* é efetuada. O resultado dessa chamada é a suspensão temporária da ocorrência do evento. O método *resume* permite colocar o evento novamente no estado *ocorrendo*, a partir do ponto em que a pausa foi realizada. A ocorrência de um evento pode também ser abruptamente interrompida por uma invocação ao método *abort*. O resultado dessa chamada é a transição do estado corrente do evento para o estado *abortado* e, logo em seguida, para *preparado*. Nesse caso, o atributo *ocorrências* não é incrementado e, sendo um evento do tipo não interativo, o atributo *repetições* é colocado com valor igual a zero. O método *stop* dos eventos não interativos geram a transição imediata do estado *ocorrendo* para o estado *preparado*, interrompendo a ocorrência no ponto em que estiver, porém respeitando o comportamento dos atributos *ocorrências* e *repetições*.

O método *start* dos eventos instantâneos modifica ligeiramente o método *start* da superclasse *Event*, colocando o evento imediatamente de volta no estado *preparado*.

A Figura 8 ilustra as máquinas de estados para os eventos definidos no modelo. Conforme pode ser observado, os eventos de seleção e atribuição simples compartilham uma mesma máquina mais simples. Uma máquina um pouco mais complexa é compartilhada pelos eventos de foco, arraste, mouse sobre âncora e atribuição progressiva. Uma terceira máquina, bastante similar à anterior, porém com um estado adicional de preparação (estado *preparando*) é definida para o evento de apresentação. Isso também torna necessária a redefinição do método *prepare* na classe *PresentationEvent*, que coloca o evento no estado *preparando* e

PUC-Rio - Certificação Digital Nº 9916204/CA

não mais diretamente no estado *preparado*. Esse novo estado exige também a criação de um método *prepared*, para colocar o evento no estado *preparado*. A idéia é que a preparação para exibição de um objeto requer um procedimento de busca das unidades de informação de seu conteúdo que não deve ser desprezado. Normalmente, todos os demais eventos terão suas preparações dependentes da preparação de algum evento de apresentação do documento.

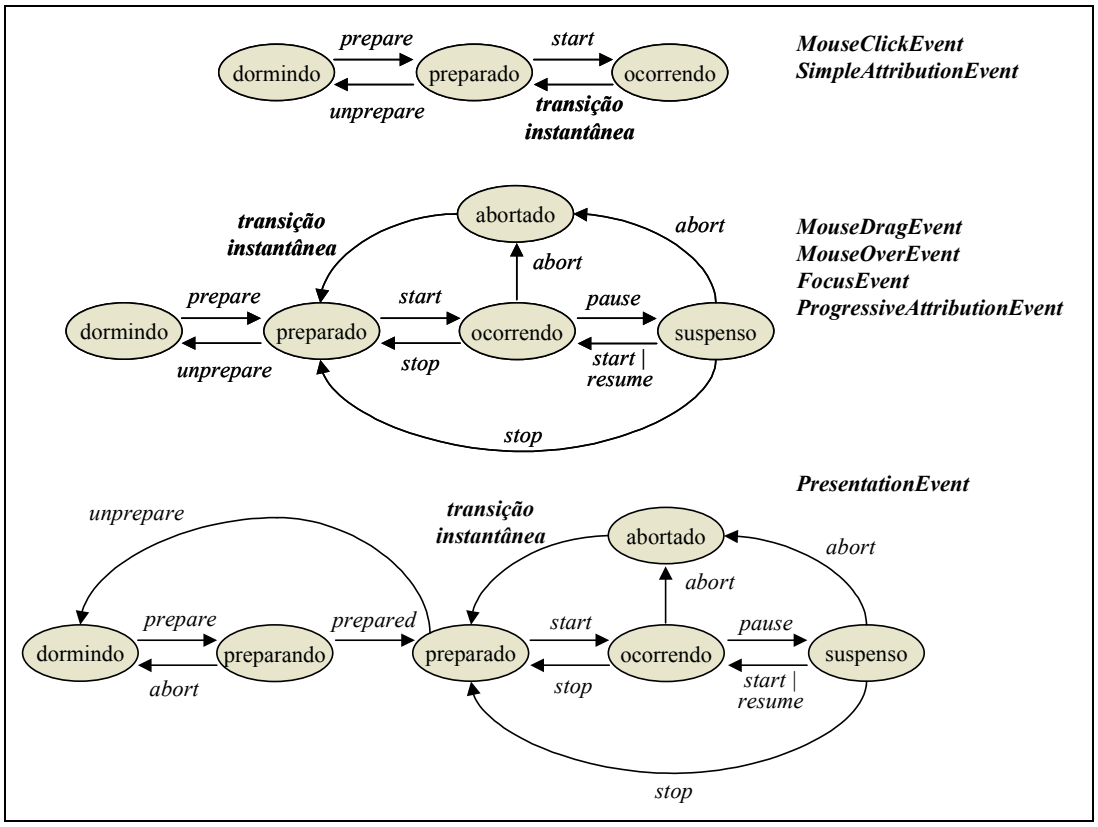


Figura 8 – Máquinas de estados dos eventos[7].

Os eventos são capazes de manter uma lista de objetos que os observam. Esses objetos são notificados toda vez que ocorre uma transição na máquina de estados do evento. Essa sinalização informa sempre o novo estado e o estado anterior na máquina e é feita após os atributos estado corrente, ocorrências e, quando existir, repetições, serem atualizados. Esse padrão de comportamento é inspirado no *design pattern* denominado *Observer* (Gamma et al., 1995) e é ilustrado na Figura 7. A interface *EventListener* define o tipo que deve ser implementado pelas classes dos objetos que se interessem por observar os eventos.

[7] Os nomes das transições estão em inglês na figura para refletir os nomes dos métodos definidos nas classes e interfaces.

PUC-Rio - Certificação Digital Nº 9916204/CA

O método *transitionUpdate* é invocado no objeto observador sempre que o estado corrente do evento tiver seu valor alterado. O método *addEventListener* e *removeEventListener* permitem, respectivamente, que um objeto se inscreva ou se retire do cadastro de observadores do evento[8].

De acordo com o que foi definido na Seção 2.3.1, todo objeto de execução possui uma lista de eventos. Nessa lista, devem encontrar-se apenas os eventos do objeto de execução que participam em algum relacionamento no documento. Toda lista deve possuir ao menos um evento de exibição (evento da classe *PresentationEvent*), o qual identificará o conjunto completo de unidades de informação do conteúdo do objeto de dados que serão apresentadas quando o objeto de execução for exibido. Normalmente, esse evento referenciará o conteúdo inteiro do objeto, mas nada impede que se deseje efetuar uma apresentação apenas parcial do conteúdo. Nesse caso, o evento de exibição estará associado a uma âncora interna do objeto de dados.

2.3.2 Elos

O *elo* é a entidade do modelo que possibilita a definição dos relacionamentos entre os componentes de um documento, mais especificamente entre os eventos definidos nos objetos de execução. Existem dois tipos de relacionamento entre os eventos de um documento que podem ser especificados no modelo, dando origem a duas especializações da classe elo. O primeiro tipo corresponde a relacionamentos causais entre os eventos, onde uma asserção lógica no elo especifica uma condição sobre determinados eventos, que quando satisfeita, implica o disparo de ações sobre outros eventos. Essa forma de relacionamento é definida em objetos da classe *elo causal*. O outro tipo de relacionamento especifica restrições sobre os eventos, que devem ser satisfeitas durante a apresentação do documento. Esse segundo tipo de relacionamento é especificado utilizando objetos da classe *elo de restrição*.

[8] Apesar desse mecanismo de observação e notificação estabelecer uma modelagem de comportamento, que poderia ser definida no formatador, optou-se por inseri-la no modelo de execução, entendendo que sua utilidade para reuso nas implementações é significativa.

PUC-Rio - Certificação Digital Nº 9916204/CA

A Figura 9 apresenta o diagrama de classes simplificado para os dois tipos de elo do modelo. Todo elo possui a expressão do relacionamento contida em um *ponto de encontro* (classe *MeetingPoint*), que é especializado em função da subclasse de elo.

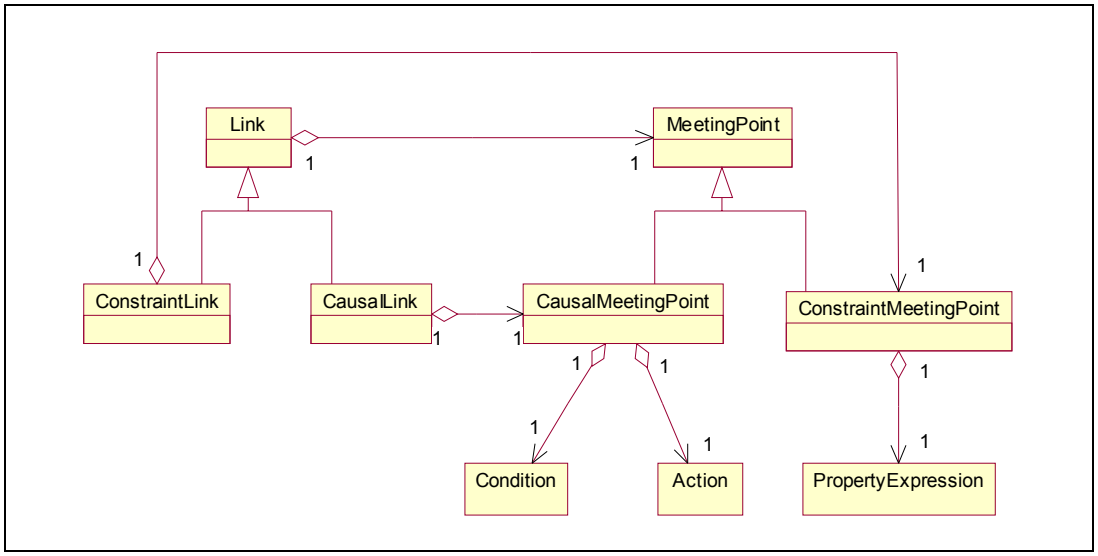


Figura 9 – Diagrama de classes simplificado para os elos.

2.3.2.1 Elos causais

Um elo causal possui como principal atributo um *ponto de encontro causal* (classe *CausalMeetingPoint*). Esse ponto de encontro expressa uma regra de causalidade, sendo constituído de uma condição (classe *Condition*) e uma ação (classe *Action*). Conforme mencionado anteriormente, a satisfação da condição implica no disparo da ação a ele associada. Tanto as condições como as ações de pontos de encontro causais são associados a eventos dos objetos de execução. A Figura 10 apresenta o diagrama detalhado para as classes relacionadas à definição de elos causais.

A condição de um ponto de encontro, quando avaliada, retorna um valor booleano e pode ser uma condição simples ou composta. Toda *condição simples* (classe *TransitionCondition*), também chamada de *condição de transição*, possui como atributos um evento *E*, um *estado anterior* (atributo *previousState*) e um *estado corrente* (atributo *newState*). Esse tipo de condição avalia uma transição na máquina de estados do evento *E*, retornando o valor verdade sempre que o estado

PUC-Rio - Certificação Digital Nº 9916204/CA

de *E* mudar do valor especificado pelo atributo estado anterior para o valor especificado pelo atributo estado corrente.

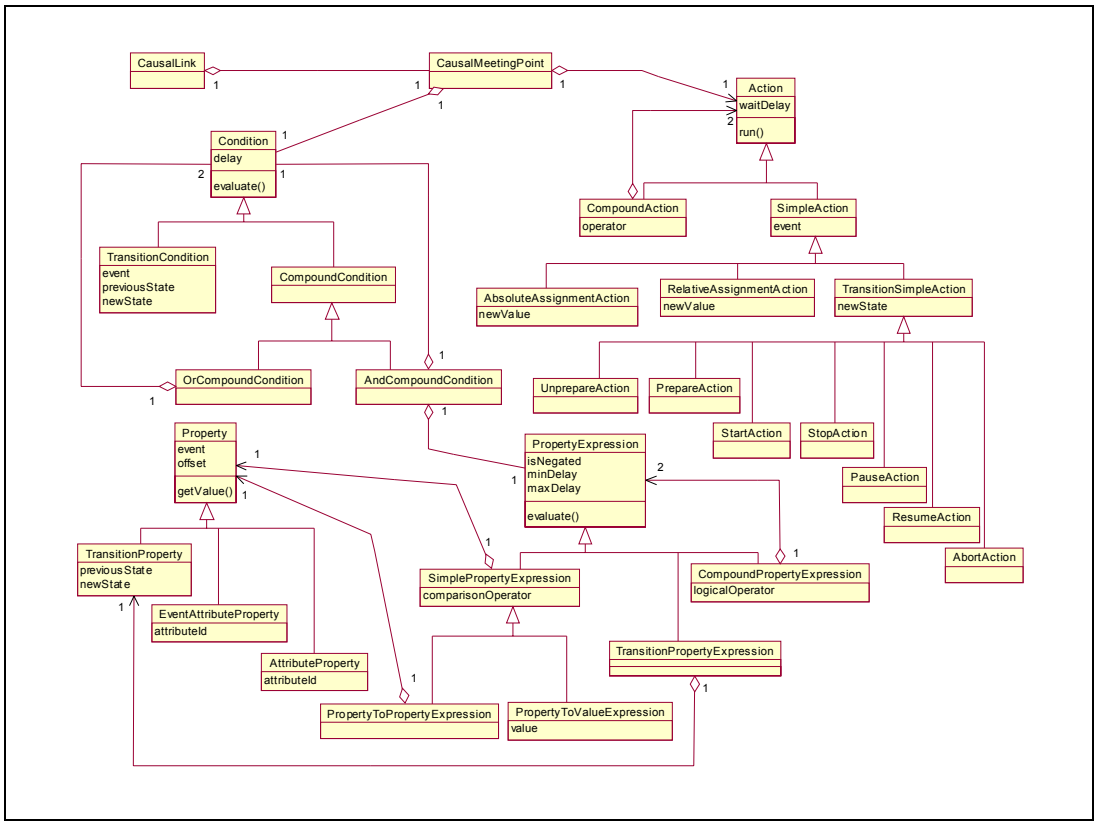


Figura 10 – Diagrama de classes para o elo causal.

Para a definição de uma condição composta (classe *CompoundCondition*), é necessário, inicialmente, definir o conceito de *propriedade* (classe *Property*).

Uma *propriedade* possui um atributo que identifica um evento *E*, podendo ser especializada em uma das três subclasses: propriedade de transição (classe *TransitionProperty*), propriedade de atributo de evento ou propriedade de atributo genérico. A *propriedade de transição* possui, além do evento, dois atributos adicionais que identificam os estados da transição (*estado anterior* e *novo estado*). A diferença da propriedade de transição para a condição de transição é que, quando avaliada, ao invés de retornar um valor booleano, a propriedade retorna o instante em que a transição ocorreu. A *propriedade de atributo de evento* (classe *EventAttributeProperty*) possui, além do evento *E*, o *identificador* de um atributo de *E* (por exemplo, *estado corrente*, *ocorrências*, *repetições*, *duração esperada* etc.). Quando avaliada, a propriedade retorna o valor corrente desse atributo. Por fim, a *propriedade de atributo genérico* (classe *AttributeProperty*) é uma

PUC-Rio - Certificação Digital Nº 9916204/CA

especialização de propriedade, onde o evento *E* deve ser obrigatoriamente um evento baseado em atributo. Essa subclasse de propriedade é similar à anterior, retornando, sempre que avaliada, o valor do atributo referenciado por *E*. Toda propriedade possui um atributo *offset*, que é somado ao seu valor quando a propriedade é consultada. Evidentemente, essa soma pode não fazer sentido para alguns atributos, como por exemplo, o estado corrente de um evento. Nesses casos, o valor do atributo *offset* deve ser ignorado pelo formatador.

A propriedade de atributo de evento poderia ser especificada utilizando a propriedade de atributo genérico. No entanto, isso obrigaria aumentar o número de eventos na lista de eventos do objeto de execução, causando uma proliferação de máquinas de estados no ambiente de execução. Por exemplo, para tratar o número de ocorrências de um evento *A*, seria necessário criar um outro evento baseado em atributo que referenciasse o atributo *ocorrências* de *A*. Por esse motivo, foi definida uma classe de propriedade separada para tratar os atributos dos eventos.

Uma *expressão de propriedades simples* (classe *SimplePropertyExpression*) permite comparar duas propriedades ou uma propriedade e um valor. Quando avaliada, essa expressão retorna verdade se a comparação por ela definida for satisfeita. Caso contrário, a expressão retorna falso. Dessa forma, toda expressão de propriedades simples possui um atributo que identifica uma propriedade e um outro atributo que identifica um operador de comparação (igual, diferente, menor ou igual, maior ou igual, menor, ou maior). As expressões que comparam duas propriedades (classe *PropertyToPropertyExpression*) possuem um terceiro atributo que identifica uma segunda propriedade, enquanto as expressões que comparam a propriedade a um valor (classe *PropertyToValueExpression*) possuem um terceiro atributo que armazena o valor a ser comparado. O tipo desse valor deve ser igual ao tipo do valor retornado pela propriedade. Da mesma forma, na comparação entre duas propriedades, as duas devem retornar valores do mesmo tipo.

Dependendo do tipo do atributo (genérico ou do evento) identificado pela propriedade, alguns operadores de comparação podem não fazer sentido na expressão de propriedades. Por exemplo, para o atributo estado de um evento só é possível utilizar as comparações de igualdade e de diferença.

PUC-Rio - Certificação Digital Nº 9916204/CA

Outra subclasse de expressão de propriedades é a *expressão de propriedades composta* (classe *CompoundPropertyExpression*). Essa última consiste em uma expressão lógica binária, baseada nos operadores "*E*" e "*OU*", envolvendo duas expressões de propriedades (superclasse *PropertyExpression*).

Expressões de propriedades possuem um atributo denominado *isNegated*, que determina o sinal da expressão. Esse atributo especifica se, após avaliada, o valor retornado pela expressão deve ser ou não invertido (negado). Além disso, toda expressão de propriedades possui dois atributos, *minDelay* e *maxDelay*, onde *minDelay* deve ser menor ou igual a *maxDelay* e ambos os valores devem ser não negativos. Esses atributos definem um intervalo de deslocamento para a avaliação da expressão. Por exemplo, dado que uma expressão de propriedades *P* é verdadeira em um instante *t*, uma outra expressão *P'*, definida com atributos *minDelay*="t_1" e *maxDelay*="t_2" e a mesma expressão de *P*, é verdadeira no intervalo dado por [$t+t_1$, $t+t_2$]. O sinal de negação da expressão deve ser aplicado somente após essa avaliação.

Além das expressões de propriedades simples e compostas, o modelo define uma terceira subclasse de expressão de propriedades, denominada *expressão de propriedade de transição*, ou simplesmente *expressão de transição* (classe *TransitionPropertyExpression*). Essa expressão contém uma *propriedade de transição* e restringe o atributo *maxDelay* como devendo possuir um valor maior que o valor do atributo *minDelay*. A expressão de transição permite definir expressões de propriedades que se tornam verdadeiras durante um intervalo após a transição na máquina de estados de um determinado evento. Esse tipo de expressão permite que as ações dos elos causais estejam condicionadas a intervalos onde não é garantido haver um estado de evento para ser testado. Por exemplo, uma expressão de transição associada à mudança do estado *ocorrendo* para o estado *preparado* de um evento de apresentação *E*, com atributos *minDelay*="*10s*" e *maxDelay*="*25s*", será considerada verdadeira somente 10 segundos após o término da ocorrência de *E* e permanecerá verdadeira durante os 15 segundos seguintes.

Uma *condição composta* consiste de uma expressão lógica binária, também baseada nos operadores "*E*" e "*OU*". A condição composta baseada no operador "*OU*" (classe *OrCompoundCondition*) envolve duas condições (simples ou

PUC-Rio - Certificação Digital Nº 9916204/CA

composta), enquanto a condição composta baseada no operador "*E*" (classe *AndCompoundCondition*) envolve uma condição (simples ou composta) e uma expressão de propriedades. As definições apresentadas visam, ao mesmo tempo, garantir que toda condição de um elo causal só possa ser satisfeita em um instante de tempo infinitesimal e impedir a especificação de uma condição composta "*E*", contendo em cada lado da comparação condições que somente sejam satisfeitas em instantes de tempo infinitesimais.

Opcionalmente, toda condição possui um atributo denominado *retardo* (atributo *delay*). O atributo retardo é definido por um valor *t*. Quando definido em uma condição *C*, a mesma é satisfeita em um instante de tempo *t'*, se a condição *C* estiver satisfeita em um instante de tempo dado por (*t'* - *t*).

Semelhante às condições, as ações de um ponto de encontro do elo causal podem ser simples ou compostas. As ações oferecem um método *run* para que sejam executadas e retornam um valor booleano, informando se foi possível ou não realizar a sua execução.

Toda ação simples possui um atributo que referencia um evento (atributo *event*). Essas ações podem ser de transição (classe *TransitionAction*), de atribuição absoluta (classe *AbsoluteAssignmentAction*) ou de atribuição relativa (classe *RelativeAssignmentAction*). Outras classes de ação podem ser criadas em função da implementação do formatador.

Uma *ação simples de transição* possui como atributo o novo estado que o evento deve assumir (atributo *newState*) em sua máquina de estados. Dependendo do estado corrente do evento e das regras definidas pelo formatador, essa transição pode ser ou não realizada. Subclasses para realizar as transições nas máquinas de estados apresentadas na seção anterior são também definidas no modelo.

As *ações de atribuição absoluta* e *relativa* só se aplicam a eventos baseados em atributo. Essas ações possuem como atributo o valor que deve ser atribuído (absoluta) ou somado (relativa) ao atributo identificado pelo evento.

Uma ação composta é formada por uma expressão de ações baseada nos operadores *paralelo* e *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* (atributo *waitDelay*). Esse atributo define um tempo que deve ser aguardado antes que a ação seja executada. Da

PUC-Rio - Certificação Digital Nº 9916204/CA

mesma forma que a duração do evento, esse atributo é especificado através de um objeto da classe duração (classe *Duration*), que pode inclusive ser flexível, cabendo ao formatador escolher um valor para o tempo de espera que garanta ao documento a melhor qualidade de apresentação possível.

De acordo com o que foi apresentado, os elos causais especificam condições sobre eventos (normalmente transições nesses estados) que, quando satisfeitas, devem disparar ações sobre outros (às vezes até os mesmos) eventos especificados no documento. São justamente essas ações que irão ocasionar novas transições nas máquinas de estados dos eventos. A seguir são ilustrados como alguns exemplos de relacionamentos hipermídia seriam representados internamente no ambiente de execução.

Exemplo 1

Suponha uma apresentação com um objeto de execução V_1 contendo um vídeo e um outro objeto A_1 contendo um áudio, onde se deseja que o áudio seja apresentado apenas durante um determinado trecho do vídeo, identificado por um evento de exibição ev_1. Considere que ea_1 é o evento de exibição que corresponde à apresentação de todo o objeto A_1 e que a máquina de estados para os eventos de exibição é a mesma ilustrada na Figura 8. A Tabela 1 apresenta a representação gráfica do relacionamento e os elos causais que seriam criados no formatador.

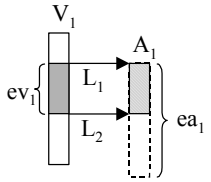
	Elo	Expressão do Relacionamento
	L_1	ev_1(preparado, ocorrendo) => start(ea_1)
	L_2	ev_1(ocorrendo, preparado) => stop(ea_1)

Tabela 1 – Sincronização temporal através de elos causais.

Exemplo 2

Esse exemplo ilustra como composições com semântica de relacionamentos de sincronização podem ser mapeadas em elos causais. Seja uma composição seqüencial com dois vídeos, V_1 e V_2, que devem ser exibidos nessa ordem. Os eventos ev_1 e ev_2 correspondem à apresentação de todo o conteúdo de cada um dos objetos, respectivamente. A Tabela 2 apresenta a representação gráfica da composição e como o relacionamento entre seus componentes seria expresso por uma causalidade entre eventos.

PUC-Rio - Certificação Digital Nº 9916204/CA

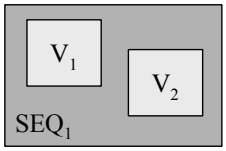
	Composição	Expressão do Relacionamento
	SEQ_1	ev_1(ocorrendo, preparado) => start(ev_2)

Tabela 2 – Sincronização temporal através de elos causais, a partir de uma especificação de sincronização utilizando composições.

Exemplo 3

Sejam dois objetos de execução: uma imagem I_1 do mapa do Brasil e um texto T_1 com informações sobre o estado do Rio de Janeiro. Deseja-se que, quando o usuário selecionar com o mouse o estado do Rio de Janeiro em I_1, seja apresentado T_1 ao lado da imagem, com largura de 150 pixels e altura de 200 pixels. Esse relacionamento pode ser especificado entre um evento de seleção ei_1, definido em I_1 sobre a região do Rio de Janeiro, e o evento de apresentação et_1, correspondendo à exibição de T_1. As características espaciais da apresentação estariam definidas na região apontada pelo descritor de T_1. A Tabela 3 ilustra o relacionamento descrito e a especificação do elo.

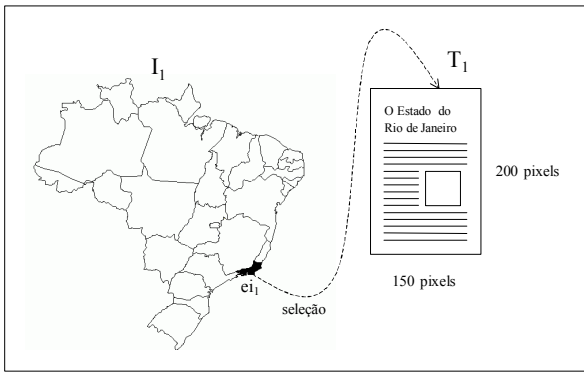
	Elo	Expressão do Relacionamento
	L_1	ei_1(preparado, ocorrendo) => start(et_1)

Tabela 3 – Sincronização espacial e temporal através de elo causal.

Conforme discutido na Seção 2.2, relações de causa e efeito (condições/ações) não são suficientes para determinar todas as relações existentes entre componentes de um documento hipermídia. Existem relações que, semanticamente, especificam restrições (*constraints*) entre eventos, tais como: dois objetos devem terminar suas exibições ao mesmo tempo. A definição desses tipos de relacionamentos é capturada pela classe elo de restrição.

PUC-Rio - Certificação Digital Nº 9916204/CA

2.3.2.2 Elos de restrição

Um *elo de restrição* é um elo cujo ponto de encontro, ao invés de definir uma regra causal, especifica restrições entre eventos. Mais especificamente, o ponto de encontro possui uma expressão de propriedades (classe *PropertyExpression*). A expressão de propriedades tem a mesma definição dada na seção anterior, quando da especificação de condições compostas do ponto de encontro de um elo causal. No entanto, por não fazer sentido considerar algumas das expressões nos elos de restrição, o modelo estabelece algumas imposições. Em primeiro lugar, não é permitido o uso de *expressões de transição* nos elos de restrição. Além disso, o atributo *isNegated* da expressão deve ser sempre *falso* e os atributos *minDelay* e *maxDelay* devem ser sempre iguais a zero. Caberá a cada implementação particular de formatador aumentar ainda mais as observações restritivas para aquelas expressões de propriedades que a máquina não souber, ou não fizer sentido, monitorar.

Os elos de restrição estipulam regras que o formatador deve procurar garantir que sejam respeitadas durante a execução do documento. Como exemplo de ponto de encontro de restrição, suponha uma propriedade de transição *P*, especificando a transição "*preparado->ocorrendo*", representando o instante de início da ocorrência de um evento de exibição *A*, e uma outra propriedade *Q*, especificando a transição "*ocorrendo->preparado*", representando o instante de término da ocorrência de um outro evento de exibição *B*. Se uma expressão de propriedades S_1 definir que "$P = Q$", isso implica que a relação *meet* de Allen (Figura 2) deve ser satisfeita na execução do documento como uma relação restritiva. Ou seja, se e somente se os dois eventos de exibição ocorrerem na apresentação, o final de *A* deve coincidir com o início de *B*. Caso um dos dois eventos não ocorra, o modelo considera que a restrição foi satisfeita.

A impossibilidade de respeitar as regras de restrição significa que existe uma inconsistência na apresentação especificada pelo autor. Essa inconsistência pode ser causada pelas próprias relações definidas internamente no documento, ou pelo cenário imposto pelo contexto de exibição, externo ao documento. As referências (Felix et al., 2002; Santos et al., 1998a; Santos et al., 1998b) discutem aspectos de verificação de consistência em apresentações hipermídia.

PUC-Rio - Certificação Digital Nº 9916204/CA

2.3.3
Suporte à adaptação no modelo de execução

O modelo de execução proposto dispõe de alguns recursos para oferecer suporte à adaptação das apresentações. O primeiro tipo de suporte reside na maneira flexível que as durações são definidas, tendo sido comentado anteriormente. O modelo permite que os tempos de um modo geral (duração dos eventos de apresentação, intervalo entre repetições, tempo de espera para executar ações etc.) sejam especificados através de intervalos e até mesmo métricas (funções de custo), possibilitando a aplicação de algoritmos de ajuste dos tempos elásticos, como será melhor explicado no Capítulo 4.

O segundo suporte que o modelo oferece para a adaptação dos hiperdocumentos são as regras de exibição (classe *PresentationRule*). Apesar de omitida na Seção 2.3.1.2, a *regra de exibição* é um atributo opcional dos descritores e estabelece uma condição para que os objetos de execução efetivamente participem da exibição do documento.

Além dos descritores, os elos também possuem como atributo opcional uma regra de exibição que, quando não satisfeita, deve gerar a remoção do relacionamento do documento. Nada impede que uma mesma regra seja referenciada por mais de um descritor e mais de um elo. A Figura 11 ilustra o diagrama de classes para as regras de exibição.

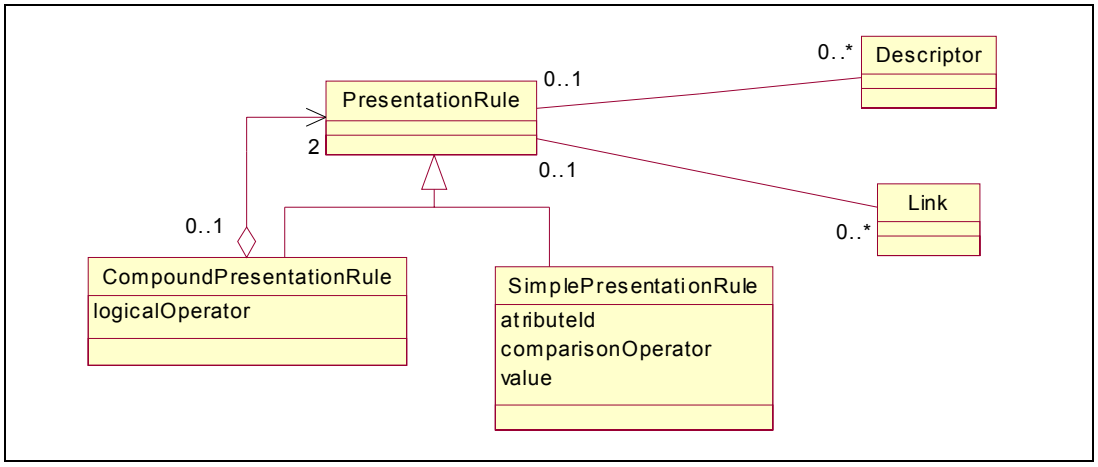


Figura 11 – Diagrama de classes para as regras de exibição.

Baseado nas informações contextuais, o formatador hipermídia deve avaliar cada uma das regras e com isso determinar se as entidades devem ou não estar

PUC-Rio - Certificação Digital Nº 9916204/CA

presentes na execução do documento. Diz-se que um objeto de execução ou elo está *desabilitado*, quando a regra de exibição a ele associada é avaliada como falsa. Caso contrário a entidade é considerada *habilitada*. Cabe a cada implementação particular de formatador decidir o momento em que as regras serão avaliadas, definindo assim se a adaptação seguirá uma estratégia estática ou dinâmica (Seção 2.1.3). É inclusive possível que as duas estratégias sejam combinadas em função dos parâmetros que estejam sendo testados. Nada impede também que, em estratégias de adaptação dinâmica, algumas regras sejam avaliadas mais de uma vez, podendo alterar o estado de habilitação das entidades durante a exibição dos documentos.

As regras de exibição são sempre baseadas em parâmetros do contexto de exibição. Em implementações de formatadores que não disponham de um gerenciador de contexto, ou que o gerenciador não saiba informar o valor do parâmetro sendo testado, as regras devem ser simplesmente ignoradas e todas as entidades devem ser consideradas habilitadas na execução do documento. Da mesma forma, objetos de execução e elos que não tenham uma regra de exibição a eles associada são sempre considerados habilitados.

Com relação à consistência do documento, sempre que um objeto de execução deve ser desabilitado em uma apresentação, alguns cuidados devem ser tomados com os relacionamentos. Em primeiro lugar, ações de elos causais que se apliquem a algum evento contido no objeto devem ser ignoradas. Implementações de formatadores podem inclusive desabilitar os elos cujas ações estejam definidas apenas sobre eventos contidos em objetos desabilitados. O segundo ponto diz respeito às expressões de propriedades, tanto nos elos de restrição como nas condições dos elos causais. Expressões de propriedades simples que possuam uma propriedade cujo evento esteja contido em um objeto desabilitado devem ser consideradas verdadeiras quando avaliadas.

As regras de exibição podem ser simples ou compostas. Uma regra de exibição simples (classe *SimplePresentationRule*) é análoga à expressão que compara uma propriedade a um valor (Figura 10) e possui três atributos: o identificador de um atributo do contexto de exibição, um operador de comparação e um valor. A regra de exibição composta (classe *CompoundPresentationRule*), por sua vez, é uma expressão lógica binária envolvendo duas regras de exibição

PUC-Rio - Certificação Digital Nº 9916204/CA

(simples ou composta) relacionadas através do operador "*E*" ou do operador "*OU*".

O terceiro e último recurso que o modelo oferece para a adaptação das apresentações são as alternativas de exibição. A *alternativa* é definida como uma especialização de objeto de execução (classe *ExecutionObjectAlternatives*) cujos atributos (identificador, objeto de dados, descritor e lista de eventos) não estão ainda todos definidos. Adicionalmente, toda alternativa de exibição possui um segundo identificador único, um conjunto de pontos de interface, um objeto de execução *default* opcional, uma lista ordenada de regras de exibição e, para cada regra da lista, um objeto de execução associado.

Na execução do documento, o formatador deve seguir a ordem da lista e avaliar as regras para cada alternativa. Para o primeiro objeto de execução que tiver a regra associada satisfeita, o formatador deve utilizar todos os seus atributos (identificador, objeto de dados etc.) para preencher os atributos não resolvidos da alternativa. Diz-se que o objeto de execução escolhido é o *objeto selecionado*. Nesse momento, é como se a alternativa deixasse de existir e fosse substituída pelo objeto de execução escolhido. Caso a lista inteira seja percorrida sem que nenhuma regra venha a ser satisfeita, o *objeto de execução default* da alternativa deve ser selecionado. Não havendo um objeto *default*, nenhum dos objetos de execução devem ser considerados na apresentação do documento. O modelo permite que alternativas sejam definidas de modo aninhado (alternativa contida em outra alternativa). A Figura 12 ilustra as classes relacionadas à definição de alternativas no modelo.

Como pode ser percebido na figura, a alternativa possui também um identificador adicional (atributo *uid*). Esse atributo permite identificar a alternativa no ambiente de execução e deve ser diferente dos identificadores de todos os outros objetos de execução presentes no formatador. Além disso, a alternativa também herda o identificador único da classe *ExecutionObject*. Esse identificador irá possui o valor do identificador do objeto de execução que vier a ser selecionado.

Os *pontos de interface* das alternativas (classe *InterfacePoint*), de acordo com o diagrama da figura, são instâncias de uma subclasse de evento. Similar à própria alternativa, o ponto de interface é um evento que ainda não foi preenchido

PUC-Rio - Certificação Digital Nº 9916204/CA

com os seus atributos. Todo ponto de interface contém uma lista de mapeamentos (atributo *mapList*), onde é definido, para cada objeto de execução *OE* contido na lista das alternativas, uma associação com algum evento do conjunto de eventos de *OE*. Quando um objeto de execução é selecionado, os mapeamentos dos pontos de interface para eventos do objeto selecionado são também escolhidos para compor a lista de eventos da alternativa.

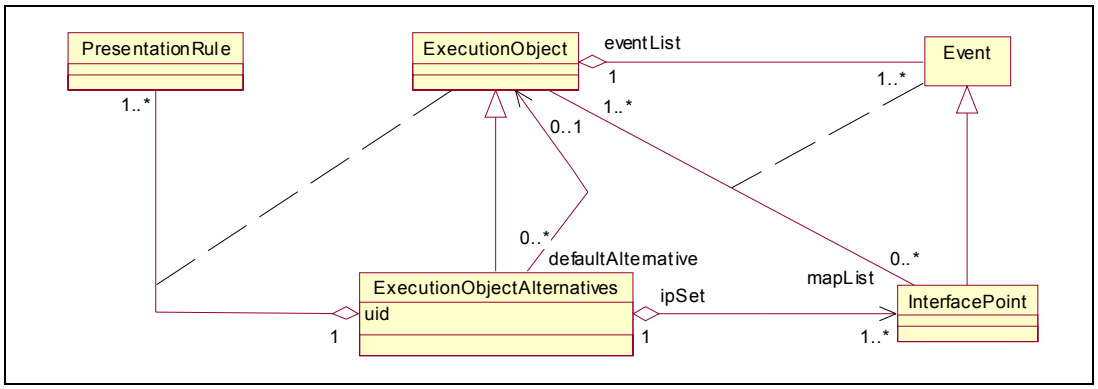


Figura 12 – Diagrama de classes para as alternativas de objetos de execução.

A definição de ponto de interface serve para que relacionamentos (elos) no modelo possam ser especificados não apenas entre objetos de execução tradicionais, como também entre alternativas de objetos, como ilustra a Figura 13. O modelo permite ainda que um mesmo objeto de execução esteja contido em mais de uma alternativa. Além disso, nada impede que alguns elos sejam definidos no modelo diretamente conectados a um objeto de execução que esteja inserido em uma alternativa. Quando o objeto não for selecionado, esses relacionamentos serão provavelmente ignorados na execução do documento.

PUC-Rio - Certificação Digital Nº 9916204/CA

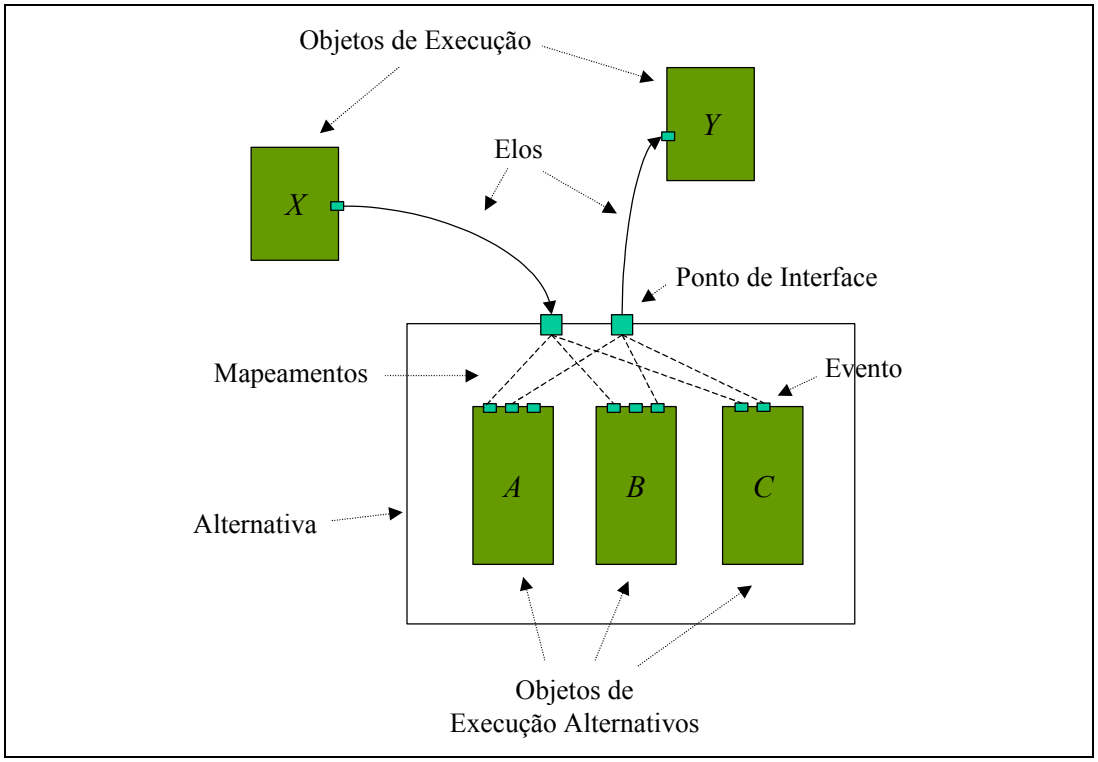


Figura 13 – Alternativas de objetos, pontos de interface e elos.

Da maneira como são definidas no modelo as alternativas de objetos, os pontos de interface e as extremidades dos elos, é possível implementar nos formatadores mecanismos de adaptação das relações (2.1.1.2). Para isso, basta colocar em uma mesma alternativa cópias de um mesmo objeto de execução com diferentes elos associados aos seus eventos. A única restrição é que cada cópia de objeto seja tratada como um objeto de execução diferente, definindo um novo identificador único. A Figura 14 oferece um exemplo de uso das alternativas para adaptar as relações. No exemplo, o elo que exibe o objeto de execução *Z* só estará habilitado no documento se a alternativa *A'*, cópia de *A*, for o objeto de execução selecionado.

PUC-Rio - Certificação Digital Nº 9916204/CA

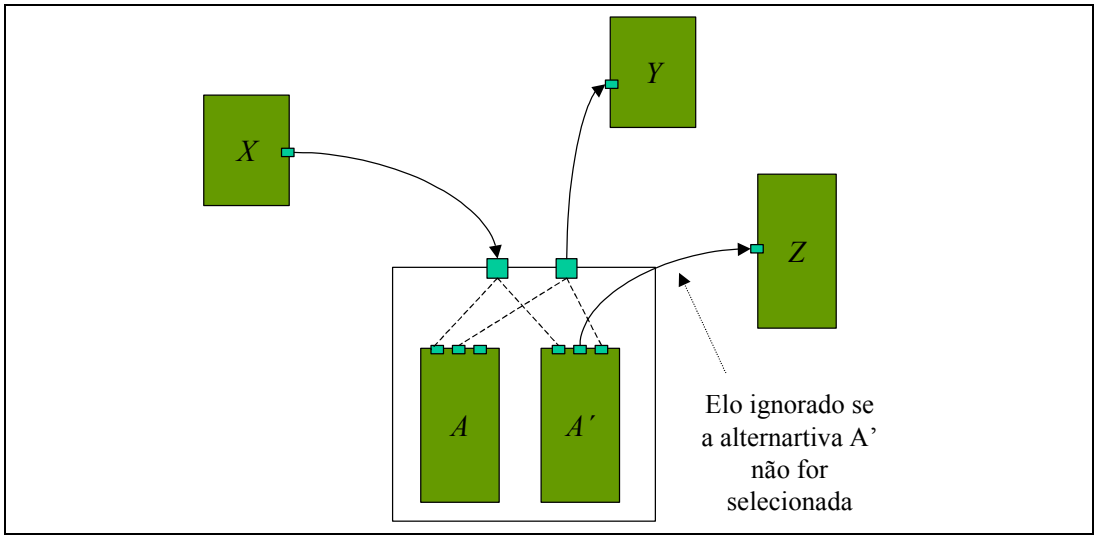


Figura 14 – Suporte à adaptação de relações no modelo.

Assim como para as regras de exibição, é critério do formatador e de suas políticas de adaptação definir o momento em que deve ser feita a avaliação de cada uma das alternativas.

2.3.4
Agrupamentos de objetos de execução e relacionamentos

O modelo define uma classe *Container*, que reúne um conjunto de objetos de execução (atributo *executionObjects*), um conjunto de elos causais (atributo *causalLinks*) e um conjunto de elos de restrição (atributo *constraintLinks*), conforme ilustrado na Figura 15.

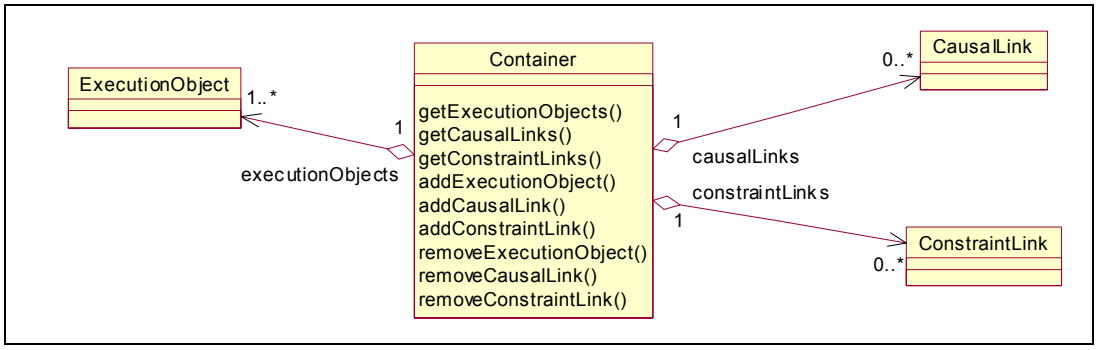


Figura 15 – Diagrama de classes para o contêiner.

Além dos atributos, o contêiner oferece um conjunto de métodos para navegação em sua estrutura, para inserção e para remoção de elementos (objetos de execução e elos). Os elos causais e de restrição são mantidos separadamente no

PUC-Rio - Certificação Digital Nº 9916204/CA

contêiner, para favorecer implementações de funções de formatação que necessitem manipular os diferentes tipos de relacionamentos de maneira seletiva.

O formatador hipermídia irá lidar com instâncias da classe *Container* para constituir o plano de controle da apresentação dos documentos. Os contêineres podem ser criados externamente ao ambiente de execução, por um ambiente de autoria que trabalhe com uma abstração de modelo idêntica ao modelo de execução, e passados diretamente como parâmetros para o formatador. A outra opção, já comentada na Figura 3, é ter o contêiner instanciado no formatador, a partir de um processo de conversão que traduz um outro modelo de documentos hipermídia no modelo proposto nesta seção.

Um documento no formatador pode ser resultado de um único contêiner ou de uma coleção deles. Quando originado de mais de um contêiner, nada impede que os mesmos sejam entregues (pelo ambiente externo ou pelo conversor) paulatinamente, permitindo que o plano de execução de um documento seja construído aos poucos. Esse recurso permite ao formatador tratar documentos de outros modelos que não apresentem claramente uma delimitação de início e fim da apresentação, como é o caso dos documentos na Web. Páginas HTML, ao possuírem os elos embutidos nos seus conteúdos e permitirem que esses elos referenciem qualquer outro recurso na rede, geram teias de relacionamentos que podem se expandir por uma quantidade imensa de nós. Mesmo em modelos (ou linguagens) que exploram o conceito de composição e definem as fronteiras para os documentos, a divisão em contêineres, e de forma progressiva, também pode ser útil, pois para os documentos mais extensos pode não ser viável construir a estrutura de execução completa antes de iniciar a exibição dos objetos.

O conceito de *contêiner* é análogo ao conceito homônimo definido no padrão MHEG (ISO/IEC, 1996). No entanto, no padrão MHEG essa entidade foi definida para transportar conjuntos de nós e elos de servidores para clientes hipermídia, enquanto no modelo de execução, esse conceito é definido na interface (API – *Application Program Interface*) para interação com os formatadores. As duas abstrações podem até conviver em uma implementação de cliente MHEG que siga o modelo de execução proposto nesta seção e a arquitetura de execução que será apresentada nos próximos dois capítulos.

PUC-Rio - Certificação Digital Nº 9916204/CA