3
Formatadores e Ferramentas de Exibição Hipermídia

Como já definido no Capítulo 1, o formatador é a entidade em um sistema hipermídia responsável por controlar a apresentação dos hiperdocumentos, baseando-se nas suas respectivas especificações e também, sempre que possível, na descrição do contexto de exibição (banda passante, recursos de E/S disponíveis no cliente, perfil e preferências do usuário etc.). Este capítulo destina-se a analisar as principais funcionalidades desejáveis de serem encontradas em um ambiente de execução hipermídia, partindo-se de um modelo de documentos orientado a eventos e adaptativo como o apresentado na Seção 2.3. É importante lembrar que, apesar deste trabalho tratar apenas da formatação no lado cliente, várias das funcionalidades discutidas neste capítulo podem ser desempenhadas também em servidores ou mesmo proxies hipermídia.

Além das características mencionadas na Seção 2.2 como sendo desejáveis de serem encontradas nos modelos de documentos hipermídia, as quais devem ser previstas e suportadas pelo formatador, é importante observar que existem também características de projeto e implementação interessantes de estarem presentes nos formatadores hipermídia.

Em primeiro lugar, é apropriado que haja uma independência entre os formatadores e as ferramentas de exibição (Seção 1.1), de tal modo que seja facilitada a incorporação no sistema de novas ferramentas, e também de ferramentas já existentes e disponíveis em outros sistemas. Tais ferramentas devem oferecer interfaces que permitam ao usuário interagir com os conteúdos dos objetos e ao mesmo tempo essas interações devem refletir no controle realizado pelo formatador. É também interessante que, para conteúdos dinâmicos (áudio e vídeo, por exemplo), o usuário possa modificar o ponto de exibição arbitrariamente e essa alteração refletir nos relacionamentos de sincronização do documento.

O segundo recurso desejável, cuja importância foi levantada nos Capítulos 1 e 2, é a existência de mecanismos capazes de adaptar as apresentações às restrições impostas pelo autor e ao contexto de exibição da aplicação hipermídia. Nesta tese, além dos ajustes elásticos das durações dos objetos, serão também tratados os aspectos de adaptação ao contexto de exibição em que o formatador executa (Seção 2.1.2), com ênfase na adaptação à plataforma. O modelo proposto (Seção 2.3.3) também contempla adaptações simplificadas dos elos e adaptações relacionadas ao interesse do usuário, apesar desses não serem o foco do trabalho.

A terceira funcionalidade de grande utilidade para um formatador hipermídia é a presença de um mecanismo de pré-busca dos conteúdos dos objetos e de antecipação às ações de exibição. Esse recurso possibilita melhorar a qualidade das apresentações e reduzir a carga de processamento exigida dos mecanismos de adaptação.

Por fim, é interessante que o formatador hipermídia explore, sempre que possível, facilidades de negociação e reserva de recursos que estejam disponíveis na infra-estrutura de execução, atuando como um usuário desses serviços com reserva, novamente visando melhorar a qualidade das apresentações e reduzir a carga de processamento exigida dos mecanismos de adaptação.

Este capítulo apresenta a estrutura básica para organização do formatador e acomodação dessas funcionalidades e também analisa detalhadamente a questão da integração do formatador com as ferramentas de exibição. As outras funcionalidades desejáveis (pré-busca, negociação de QoS e adaptação) serão discutidas no capítulo seguinte.

3.1
Estrutura básica de um formatador hipermídia

A Figura 16 apresenta os principais componentes que integram um formatador hipermídia ideal, destacando também alguns dos elementos externos a ele, com os quais é mantida uma maior interação. De fato, a figura oferece uma visão mais detalhada do ambiente de execução apresentado na Figura 1.

A primeira etapa na apresentação de um hiperdocumento é a chegada de uma solicitação para o início da sua exibição, na qual a descrição do documento (ou parte dele) é passada como parâmetro (seta 1, na Figura 16). Essa descrição

pode ser proveniente tanto do ambiente de autoria como do ambiente de armazenamento (Figura 1), no caso de um documento já previamente publicado no servidor. Vale ressaltar que o formatador pode receber o documento inteiro ou em partes, de maneira incremental.

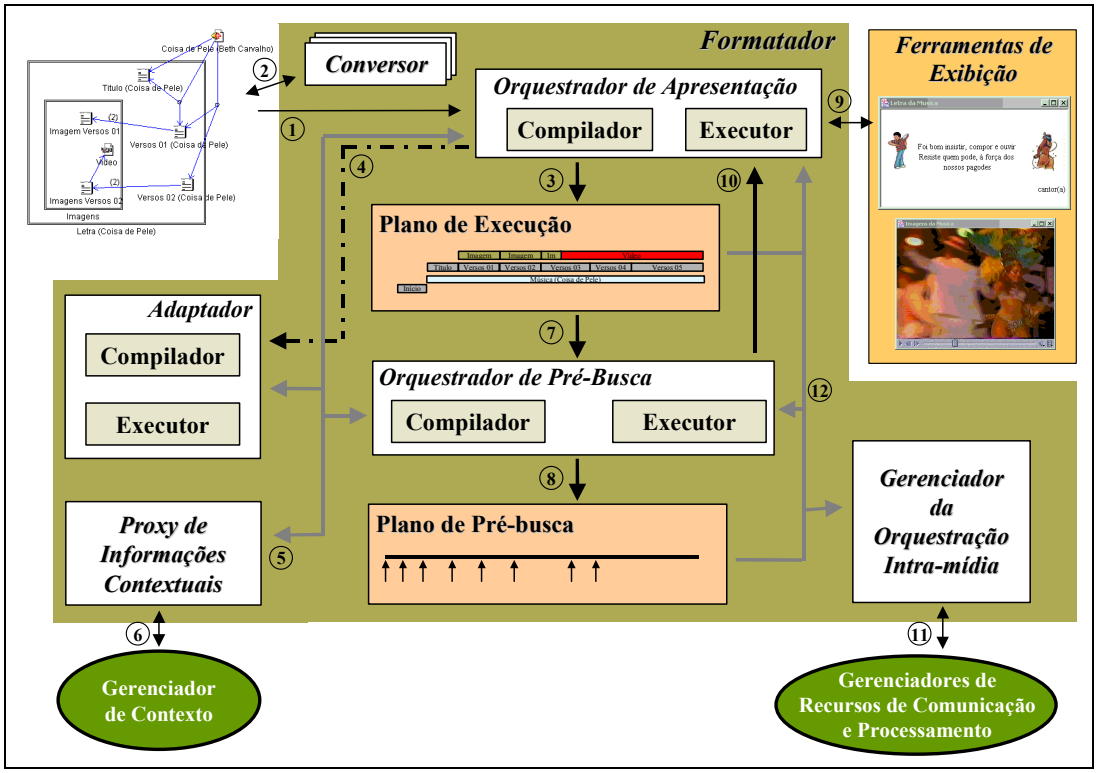

Figura 16 – Principais elementos que compõem um formatador hipermídia.

Na especificação do hiperdocumento serão encontradas, entre outras informações, a descrição dos nós que compõem a apresentação, a descrição dos relacionamentos entre esses nós (relacionamentos de estruturação, sincronização, alternativas etc.) e a descrição das características para a exibição de cada um dos nós. Se essa especificação não estiver de acordo com o modelo de execução esperado pelo formatador, ela deve ser antes entregue a um módulo *conversor* para transformá-la em uma descrição que siga o modelo de contêineres de objetos de execução e elos, descrito na Seção 2.3 (seta 2). É justamente essa etapa de conversão que permite que o ambiente de execução controle a exibição de documentos provenientes de diferentes modelos hipermídia. Evidentemente, algumas características do modelo de documentos de entrada podem ser perdidas, se as mesmas não puderem ser mapeadas no modelo de execução.

Para auxiliar na coordenação da apresentação, o formatador normalmente irá construir um *plano de execução* (ou plano de escalonamento), a partir da estrutura de dados de entrada (montada externamente ou pelo módulo conversor). Dentre as informações que devem estar contidas nesse plano, destacam-se as previsões das durações de ocorrência dos eventos não instantâneos, principalmente os eventos de exibição, e os instantes de tempo esperados para exibição dos vários objetos de execução. No entanto, é importante que o plano não seja um simples *timeline* dessas ações, mas sim que preserve as relações de dependência entre os objetos, pois isso irá favorecer sua manutenção, quando da aplicação das técnicas de adaptação durante a exibição do documento. Por exemplo, se as unidades de informação do conteúdo de um objeto não forem exibidas no instante previsto (ou com a duração prevista) por motivo de um congestionamento na rede, conhecendo as relações de dependência, o formatador pode tentar ajustar as durações dos eventos de exibição diretamente afetados, buscando assim recuperar a sincronização da exibição, sem a obrigação de refazer todo o plano de execução a partir da adaptação efetuada. Portanto, é importante que o plano de execução mantenha um vínculo com as informações descritas nos relacionamentos de causalidade e de restrição entre os objetos especificados no modelo de execução.

Também é importante que o plano de execução mantenha as informações referentes às regras de exibição e às alternativas para os parâmetros dinâmicos do contexto de exibição. Na verdade, o plano de execução complementa as informações contidas no modelo de execução, buscando favorecer os mecanismos de escalonamento e adaptação. A etapa em que o plano de execução é ao menos parcialmente construído, antes que os objetos comecem a ser exibidos, é denominada *fase de compilação* do documento, sendo a construção desse plano (seta 3, na figura) responsabilidade do submódulo *compilador*, no *orquestrador de apresentação*.

O *compilador de apresentação* espera como entrada contêineres do modelo de execução (Seção 2.3.4) e, assim como os conversores do ambiente de execução, também pode receber os dados de maneira progressiva.

Retornando à questão do cálculo das durações e tempos esperados, é importante salientar que podem existir objetos de execução para os quais o compilador de apresentação não consiga definir a priori os instantes de exibição

esperados, pois nem sempre a apresentação de um objeto é previsível. A idéia de previsibilidade de apresentação de um objeto está associada com a previsibilidade do evento de exibição correspondente, que por sua vez é sempre relativa a algum outro evento da apresentação ou ao início da apresentação como um todo.

Uma estratégia interessante para a construção do plano de execução é seguir a proposta do sistema Firefly (Buchanan & Zellweger, 1992) e estabelecer as cadeias temporais do documento. Nessa proposta, uma cadeia é denominada a principal e corresponde à seqüência de exibições previsíveis que começa com o próprio princípio da apresentação de um documento. Além disso, zero ou mais cadeias temporais auxiliares podem existir, cada uma sendo iniciada por uma transição imprevisível na máquina de estados de algum evento do documento e disparando uma seqüência de exibições previsíveis em relação a essa transição. Seguindo essa abordagem, o compilador de apresentação irá calcular, para cada cadeia temporal, os instantes de tempo esperados de uma maneira relativa ao início da cadeia. Note assim que o plano de execução é de fato uma estrutura de dados que reflete as diversas cadeias temporais.

Durante a fase de compilação da apresentação, se o modelo hipermídia permitir (ou seja, oferecer informações flexíveis), estratégias de adaptação, tais como ajuste de tempo dos objetos e escolha entre múltiplas opções de apresentação (por exemplo, opções de resolução, de dimensão, de idioma, tipos de mídia etc.), podem ser acionadas para encontrar a configuração que atenda as restrições do autor e ao mesmo tempo ofereça a melhor qualidade de exibição para o contexto em questão. Para tanto, o compilador de apresentação deve requisitar os serviços do módulo adaptador (seta 4), que por sua vez pode precisar consultar as informações do contexto de exibição (seta 5), antes de acionar seu submódulo *compilador*. O resultado da adaptação é então refletido no plano de execução. Para as alternativas que não possam ser resolvidas na fase de compilação, o compilador de apresentação deve deixar indicado no plano de execução a existência de um conjunto de alternativas e criar uma nova cadeia temporal auxiliar onde sejam colocados os objetos cujas exibições dependam dessa adaptação dinâmica.

Nesse momento, é importante comentar a existência no formatador do módulo denominado *proxy de informações contextuais*. A idéia dessa proposta é encapsular nesse elemento a responsabilidade de buscar as informações junto ao

gerenciador de contexto (seta 6). A presença do proxy oferece uma interface única para os demais componentes do formatador e desobriga que esses componentes conheçam o protocolo para consulta às informações contextuais, em especial as dinâmicas. Essa separação é interessante principalmente pela variedade de cenários e protocolos que podem vir a reger esse tipo de consulta, pois o proxy pode necessitar dialogar com mais de um gerenciador (informações do sistema operacional, informações da rede, informações do usuário etc.) e combinar o uso de diferentes protocolos e estruturas de dados. No entanto, nada impede que em cenários mais simples, o proxy seja, na realidade, o próprio gerente de contexto.

A existência de um mecanismo de pré-busca no formatador é importante para minimizar os retardos no início das exibições dos objetos e compensar as variações nas taxas de exibição dos objetos de mídia contínua. Torna-se então conveniente a construção de um plano de pré-busca a partir do plano de execução (setas 7 e 8), antecipando as ações especificadas no documento. Essa tarefa é realizada pelo módulo orquestrador de pré-busca, na Figura 16. Na construção do plano de pré-busca, também serão extremamente úteis informações que estejam disponíveis a respeito do contexto de exibição, tais como banda passante disponível e probabilidades de navegação (seta 5, na figura).

Uma vez iniciada a exibição dos objetos, o formatador entra na *fase de execução*, coordenada pelo submódulo *executor* do orquestrador de apresentação. O executor de apresentação é o elemento responsável por monitorar a apresentação do documento, instanciando e interagindo com as ferramentas responsáveis pela exibição dos objetos (seta 9, na Figura 16). Cabe ao executor escalonar as ações a fim de cumprir o plano de execução definido pelo compilador. Além disso, toda vez que um evento imprevisível ocorrer, como a interação do usuário, se houver uma cadeia temporal auxiliar iniciada por esse evento, essa cadeia deve ser unida à cadeia temporal principal, gerando as devidas correções no plano de execução. É também responsabilidade do executor de apresentação monitorar os instantes de tempo em que as ações efetivamente ocorrem e comparar com os tempos originalmente previstos. Sempre que o executor perceber diferenças, é desejável que o módulo de adaptação seja acionado a fim de realizar os ajustes possíveis de serem aplicados. O módulo de adaptação, por sua vez, pode precisar consultar as informações dinâmicas do

contexto de exibição (seta 5), antes de acionar seu submódulo *executor*, de forma análoga à fase de compilação.

Cabe aqui ressaltar a existência, no módulo de adaptação, de tanto um submódulo compilador como um executor. Normalmente, as estratégias de adaptação executadas no momento da compilação do documento podem demorar um tempo maior para gerar o seu resultado. Na realidade, várias das funções de compilação podem inclusive ser realizadas horas, ou mesmo dias, antes da apresentação. Pode ser inclusive interessante que algumas dessas tarefas sejam desempenhadas em tempo de autoria, numa fase denominada de *pré-compilação* (Rodrigues et al., 1997), oferecendo um recurso na ferramenta de edição que pode ser útil, por exemplo, para tornar mais ágil a identificação de apresentações inconsistentes (apresentações para as quais não é possível gerar um plano de execução). Por todas essas razões, os algoritmos de compilação, principalmente os algoritmos de adaptação, podem apresentar uma maior complexidade, buscando em contrapartida gerar um resultado mais próximo da qualidade ideal de exibição.

Após iniciada a exibição, os algoritmos de controle executados no formatador precisam oferecer resultados quase que imediatos. Nesse sentido, os algoritmos de ajuste são críticos, pois eles são requisitados não apenas para corrigir erros e atrasos nos sistemas de comunicação e processamento, como também lidar com os eventos imprevisíveis (principalmente interações do usuário) inerentes aos modelos hipermídia. A própria união de uma cadeia temporal auxiliar à cadeia principal irá resultar em novos cálculos e, provavelmente, na necessidade de novas adaptações. Por esses motivos, os algoritmos de compilação podem não se mostrar adequados para a fase de execução, exigindo novas técnicas e heurísticas para as estratégias de controle e ajuste.

O submódulo executor de adaptação deve também cuidar de monitorar os parâmetros dinâmicos do contexto de exibição, avaliando (ou reavaliando) as regras de exibição e as alternativas de objetos de execução, para que os resultados dessas avaliações sejam refletidas no plano de execução.

Apesar da maioria das interações do orquestrador de apresentação com a gerência de contexto (proxy de informações contextuais) ser feita por intermédio do adaptador, algumas ações no controle da apresentação podem exigir uma comunicação direta entre esses elementos (seta 5, na figura), sem que seja

necessário ser envolvido o aspecto de adaptação. Um exemplo é a consulta, por parte do orquestrador de apresentação, às ferramentas de exibição disponíveis na plataforma para que possa ser iniciada a apresentação do conteúdo de um objeto. Os mecanismos de pré-busca também podem fazer uso das informações de contexto, independentemente do módulo de adaptação, conforme será detalhado na Seção 4.1.

Outra função que deve ser desempenhada na fase de execução é o escalonamento das ações de busca definidas no plano de pré-busca. Essa tarefa é responsabilidade do executor de pré-busca, que deve também, assim como o executor de apresentação, monitorar os tempos efetivamente gastos e compará-los com as previsões de tempo realizadas. Quando necessário, os devidos ajustes no plano, para garantir a sincronização da apresentação, devem ser efetuados. Se não for possível adaptar o plano de pré-busca, o orquestrador de pré-busca deve notificar o orquestrador de apresentação, para que as correções sejam feitas diretamente no plano de execução (seta 10). Da mesma forma, o executor de pré-busca deve estar atento a qualquer mudança no plano de execução que possa implicar em novos cálculos dos tempos de pré-busca.

É comum que a exibição dos objetos de execução, principalmente os baseados em mídias contínuas, exija uma alocação dos recursos da plataforma, tais como banda passante na rede, banda passante no acesso a disco, percentual de utilização da CPU etc. O papel do módulo *gerenciador da orquestração intra-mídia* (ou simplesmente *gerenciador intra-mídia*) no formatador é o de reunir as funções de negociação e monitoramento da qualidade de serviço junto aos provedores de comunicação e processamento (tipicamente rede e sistema operacional), para tentar garantir a busca e exibição dos conteúdos dentro dos limites estipulados pelo autor e aceitáveis pelo usuário (seta 11). Contudo, a presença desse módulo só é útil quando os provedores oferecerem suporte à reserva de recursos, que pode ser tanto imediata (Braden et al., 1997; Coulson & Blair, 1995), como antecipada (Berson et al., 1998; Degermark et al., 1995; Wolf et al., 1995).

A seta 12, na Figura 16, ilustra a interação do gerenciador da orquestração intra-mídia com os planos de pré-busca e execução e com os orquestradores de pré-busca e de apresentação. O gerenciador deve consultar os planos para decidir

os momentos de iniciar, para cada objeto, o processo de negociação da reserva dos recursos junto aos provedores, e monitorar o comportamento dos provedores para receber, caso ocorra, a sinalização de violações em parâmetros da QoS acordada. Quando não for possível manter essa qualidade para um determinado objeto de execução, a notificação de violação deve ser sinalizada para o orquestrador de pré-busca (ou diretamente para o orquestrador de apresentação) para que, idealmente, o problema seja corrigido através de ajustes nos planos.

Com exceção do orquestrador de apresentação, do plano de execução e da gerência de contexto (por mais simples que ela seja), todos os demais componentes são opcionais em um formatador. Obviamente, a presença dos módulos opcionais traz um melhor suporte ao controle das apresentações, proporcionando uma qualidade mais apurada nos documentos exibidos.

Complementando as tarefas de formatação no ambiente de execução, estão as ferramentas de exibição. As ferramentas são os elementos responsáveis por lidar diretamente com a apresentação dos conteúdos dos objetos de execução. Sendo assim, são elas que terão a função de exercer o controle sobre as máquinas de estados dos eventos, cuidando de gerar as transições para que os elos do documento possam ser avaliados e o plano de execução monitorado.

A próxima seção apresenta uma proposta de diagrama de classes para o desenvolvimento de formatadores hipermídia, seguindo a estrutura descrita nesta seção. As questões relacionadas ao projeto de ferramentas e à integração delas com o formatador são tratadas na seção subseqüente.

3.2 Diagrama de classes genérico para o desenvolvimento de formatadores hipermídia

A entidade principal para o desenvolvimento de um formatador hipermídia é a classe *Formatter*, apresentada na Figura 17. Como pode ser percebido, um objeto dessa classe possui, obrigatoriamente, um compilador (classe *Compiler*) e um executor (classe *Executor*), correspondendo respectivamente aos submódulos compilador e executor do orquestrador de apresentação. Além disso, o formatador também mantém um plano de execução (classe *ExecutionPlan*) e um gerente (ou proxy de gerência) de contexto (classe *ContextManager*). O formatador possui

ainda, opcionalmente, um adaptador de documentos (classe *DocumentAdapter*), um orquestrador de prefetch (classe *Prefetcher*) e um gerenciador da orquestração intra-mídia (classe *IntraMediaManager*), não havendo impacto no funcionamento se qualquer um desses componentes não estiver presente. Por fim, para cada modelo de documentos que o formatador souber tratar, diferente do modelo de execução, deve haver uma implementação de conversor de modelo (classe *ModelConverter*).

O formatador atua, ao mesmo tempo, como uma interface para os elementos externos ao ambiente de execução (qualquer aplicação que queira fazer uso dos serviços de formatação hipermídia) e como um mediador para os elementos internos ao ambiente de execução. Essa última característica permite que os módulos se comuniquem através de chamadas ao formatador. No entanto, como certos elementos devem manter uma interação maior, alguns relacionamentos são estabelecidos também entre as classes internas ao formatador, como são os casos dos relacionamentos entre o compilador e o executor com o plano de execução.

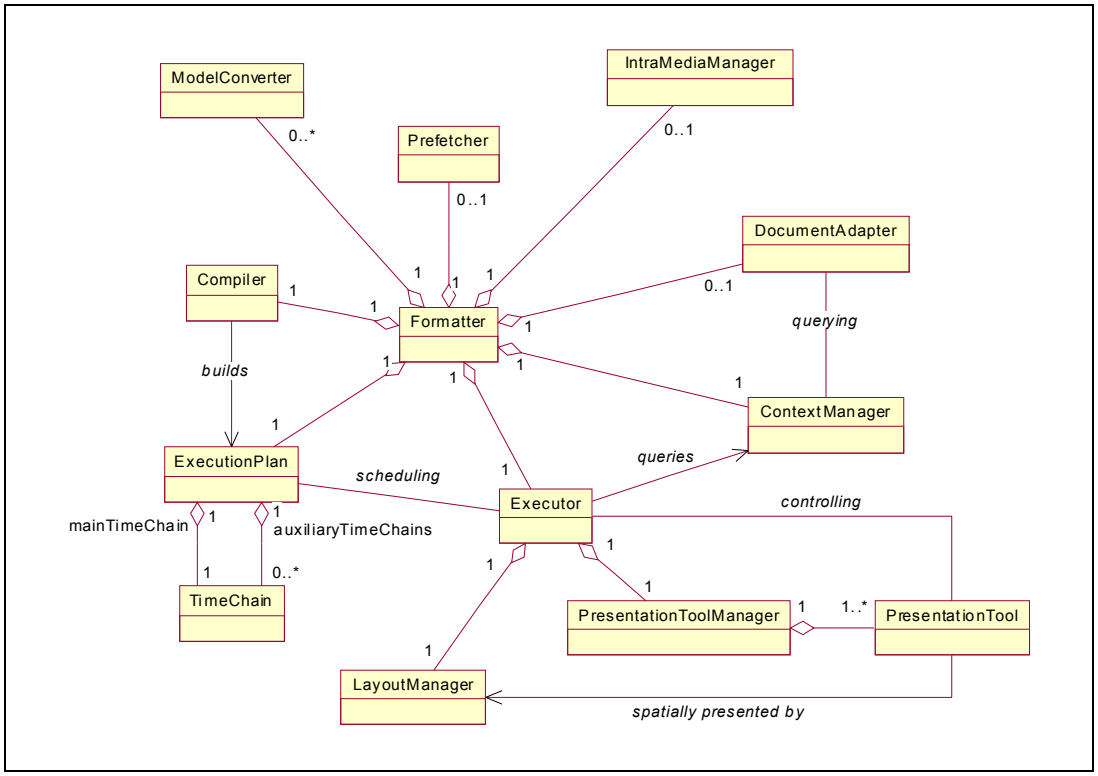

Figura 17 – Diagrama de classes para o desenvolvimento de formatadores[9].

[9] Nos relacionamentos do diagrama, onde não foi colocada a multiplicidade, é porque ela é igual a um.

O plano de execução é, na verdade, composto por uma *cadeia temporal principal* (atributo *mainTimeChain*), que guarda as transições de eventos relativas ao início da apresentação do documento, e um conjunto de *cadeias temporais auxiliares* (atributo *auxiliaryTimeChains*), que são iniciadas por ocorrências imprevisíveis, mas que uma vez iniciadas geram uma seqüência de transições previsíveis. As cadeia temporais são todas modeladas pela classe *TimeChain*.

O executor mantém com o gerenciador de contexto uma relação direta de consulta às informações, principalmente da plataforma. As informações provenientes do contexto são úteis principalmente para os dois componentes do executor destacados na Figura 17: gerenciador de layout (classe *LayoutManager*) e gerenciador de ferramentas (classe *PresentationToolManager*).

O *gerenciador de layout* é responsável por mapear as janelas e regiões definidas no documento, em componentes gráficos disponíveis na plataforma onde o documento é exibido. É o gerenciador de layout que deve manter a instância da classe *Layout* (Figura 6), controlando, inclusive, a criação da janela de exibição *default* do formatador. A partir das janelas e regiões criadas, o gerenciador de layout controla as características espaciais das ferramentas de exibição e, por conseqüência, dos objetos de execução. É também esse gerenciador o elemento responsável por manter a relação entre os dispositivos de saída da plataforma e os objetos de execução.

O *gerenciador de ferramentas* deve possuir uma lista das ferramentas de exibição disponíveis na plataforma e controlar suas instanciações, utilizando as informações provenientes dos conteúdos e descritores dos objetos de execução. Uma vez instanciadas, as ferramentas passam a interagir diretamente com o submódulo executor do formatador.

Por lidarem diretamente com a informação apresentada ao usuário, as ferramentas de exibição são componentes fundamentais no ambiente de execução. Mais importante ainda é a interface de comunicação entre essas ferramentas e o formatador hipermídia, pois a avaliação dos relacionamentos (elos) e execução das ações depende dessa troca de mensagens. Dessa forma, a próxima seção discute as questões referentes ao projeto e integração de ferramentas de exibição em ambientes de execução hipermídia.

3.3
Integração entre ferramentas de exibição e formatadores hipermídia

As ferramentas de exibição são responsáveis pela apresentação de cada um dos objetos de execução que compõem o documento. Sendo assim, qualquer ferramenta precisa entender ao menos algumas das entidades definidas no modelo de execução. Mais especificamente, as ferramentas devem saber como interpretar as informações contidas nos objetos de dados, nos eventos que devem ser por ela monitorados e nos descritores.

A principal informação contida no objeto de dados (Seção 2.3.1.1) que a ferramenta deve saber tratar é o conteúdo do nó hipermídia. Na realidade, esse entendimento passa por uma cooperação com o gerenciador de ferramentas do formatador, que, teoricamente, só irá lhe entregar objetos cujo conteúdo possam ser decodificados e exibidos.

Juntamente com o conteúdo, os eventos são elementos que também precisam ser muito bem compreendidos pela ferramenta. Para os eventos baseados em âncora, a ferramenta deve identificar a sua porção no conteúdo e controlar diretamente as suas máquinas de estados. De modo similar, os eventos baseados em atributos devem resultar no monitoramento dos valores dos atributos e também no controle de suas máquinas de estados. As ferramentas precisam também conhecer a semântica das classes de eventos definidas no formatador (*exibição*, *seleção*, *arraste* etc.) para tratar as ocorrências quando as mídias são efetivamente apresentadas.

Diferente dos objetos de dados e dos eventos, nem todas as informações contidas nos descritores precisam ser compreendidas pela ferramenta de exibição. A responsabilidade da interpretação da parte espacial definida na região e na janela pode ser dividida com o gerenciador de layout, cabendo à ferramenta utilizar o identificador da região para requisitar serviços de exibição ao gerenciador. Nesse relacionamento, é também importante que a ferramenta consiga monitorar os valores de certos atributos espaciais para que consiga controlar a máquina de estados dos eventos de atribuição.

Os parâmetros de qualidade de serviço definidos nos descritores também não precisam ser entendidos pela ferramenta, mas sim pelas estratégias de pré-busca, de adaptação e pelo orquestrador intra-mídia. Evidentemente, em

implementações que integrem o orquestrador à ferramenta, indiretamente os parâmetros de QoS terminarão por ser compreendidos também pela própria ferramenta.

Outro atributo do descritor que não necessita de ser tratado pela ferramenta é a regra de exibição. O tratamento das regras deve ser feito pelo mecanismo de adaptação do documento, antes que a ferramenta seja acionada para exibir o objeto de execução.

Restam como dados do descritor a serem interpretados exclusivamente pelas ferramentas, os atributos específicos de cada conteúdo, tais como folha de estilo principalmente em objetos do tipo texto, altura do volume para áudio e vídeo etc.

É fundamental em um ambiente de execução hipermídia que as ferramentas estejam bem integradas ao formatador, para não somente exibir objetos de diferentes tipos de mídia, mas também oferecer o suporte para que os relacionamentos de sincronização especificados pelo autor do hiperdocumento sejam obedecidos.

Conforme já comentado e ilustrado na Figura 1 e na Figura 16, uma maneira modular de desenvolver o ambiente de execução é separar o formatador das ferramentas. Dessa maneira, as ferramentas podem se tornar isoladas da complexidade das funcionalidades de formatação comentadas nas duas seções anteriores (compilação, escalonamento da execução, adaptação, pré-busca, negociação e manutenção de QoS etc.) e concentrar seus esforços na decodificação e exibição do conteúdo e no controle das máquinas de estados dos eventos. Mais ainda, através de uma interface bem especificada e padronizada para troca de mensagens entre as ferramentas e o formatador, é possível alcançar uma independência para os projetistas desses elementos, ao mesmo tempo garantindo uma interoperabilidade. Essa separação permite reusar implementações de ferramentas em diferentes sistemas hipermídia e até mesmo integrar ferramentas de implementadores distintos em um mesmo ambiente de execução. Outro benefício dessa separação é o oferecimento de um suporte para a implementação de relacionamentos de sincronização temporal e espacial, entre objetos exibidos em diferentes ferramentas de exibição.

Apesar da uniformização da interface trazer os benefícios comentados no parágrafo anterior, a proposta de integração obriga que as ferramentas conheçam o

modelo de execução utilizado pelo formatador. No entanto, existe uma variedade de exibidores de mídia com interfaces e modelos próprios, tais como os exibidores JMF (Sun, 1999), exibidores QuickTime[10], exibidores da RealNetworks[11] etc., que não seguem o padrão de funcionamento esperado pelo formatador, mas que são interessantes de serem integrados ao ambiente de execução. Para esses casos, se for possível, a alternativa é desenvolver, como parte da ferramenta de exibição, um módulo capaz de efetuar a ponte entre as duas interfaces, conforme ilustrado na Figura 18.

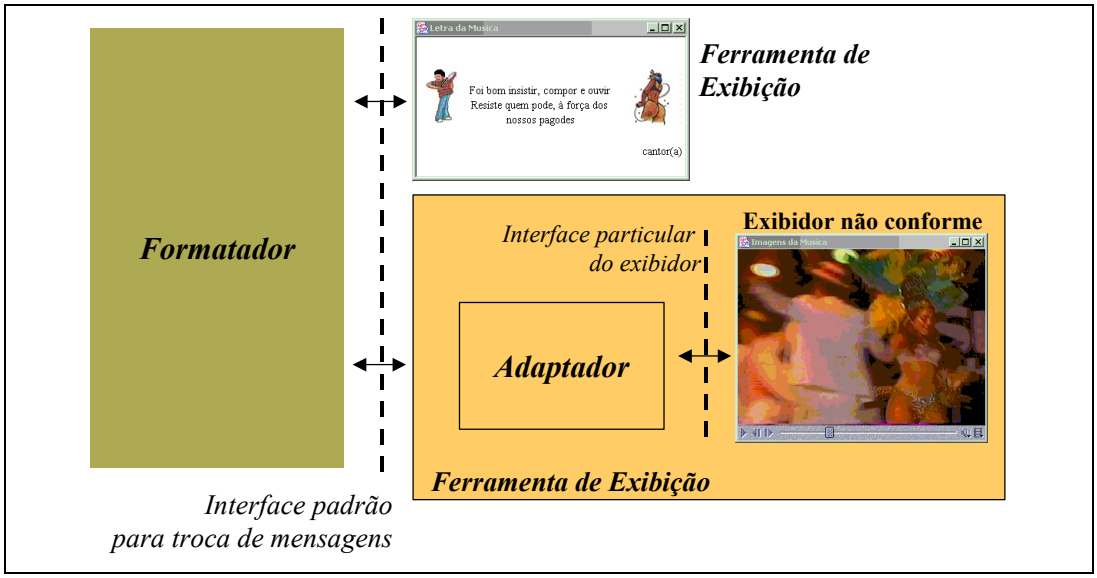

Figura 18 – Interface para integração de ferramentas de exibição e formatadores, prevendo a utilização de adaptadores para integração com exibidores cuja interface difere da esperada pelo formatador.

A próxima subseção define, de maneira mais detalhada, a interface para integração das ferramentas de exibição a formatadores hipermídia. A interface proposta permite não apenas que ferramentas diversas interajam com o formatador, como também cooperem entre si na apresentação de um hiperdocumento.

[10] http://www.apple.com/quicktime/

[11] http://www.realnetworks.com/

3.3.1
Interface entre formatadores e ferramentas de exibição

De um modo geral, a interface entre ferramentas de exibição e formatadores deve padronizar a maneira como as ferramentas notificam o formatador de certas ocorrências durante a apresentação dos objetos, como por exemplo, que a exibição de uma determinada região do conteúdo (*âncora*) de um objeto se encerrou, que uma âncora foi selecionada pelo usuário, que a posição do objeto na tela se modificou etc.

Baseando-se no modelo de execução proposto na Seção 2.3, o envio de mensagens das ferramentas para o formatador é feito pelo próprio controle da máquina de estados dos eventos. Sempre que a ferramenta modifica o estado de um evento, todos os elementos que estiverem registrados como observadores do evento serão notificados. Cabe ao executor do orquestrador de apresentação registrar-se como observador dos eventos (implementar a interface *EventListener* – Seção 2.3.1.3) e, eventualmente, ao compilador do orquestrador de apresentação colocar também as condições de transição como observadores dos eventos.

No sentido inverso, isto é, do formatador para as ferramentas, a interface deve especificar como solicitar que as ações sejam executadas sobre os eventos não interativos, tais como, iniciar a exibição de um conteúdo, suspender a exibição, alterar o valor do atributo taxa de apresentação, modificar o valor do atributo volume do som etc. Essas ações irão por sua vez refletir em novas transições nas máquinas de estados dos eventos.

Toda ferramenta de exibição deve ser projetada como uma subclasse de *PresentationTool* (Figura 6 e Figura 17) e implementar os métodos definidos na interface dessa superclasse. Esses métodos permitem justamente requisitar a execução das ações sobre os eventos controlados pela ferramenta. A idéia é que toda vez que uma ação de um elo causal tiver que ser executada, a requisição seja feita à ferramenta de exibição, para que a ferramenta implemente a semântica associada ao evento e gere a transição na sua máquina de estados. A Tabela 4 descreve a interface genérica que deve ser oferecida por toda ferramenta de exibição.

Método	Dados de Entrada	Descrição
initialize	Objeto de execução	Coloca o objeto de execução, sob o controle da ferramenta e inicia as estruturas de dados, principalmente as máquinas de estados correspondentes aos eventos do objeto.
addExecutionObject	Objeto de execução	Coloca mais um objeto de execução sob o controle da ferramenta.
removeExecutionObject	Objeto de execução	Remove um objeto de execução do controle da ferramenta.
prepare	Objeto de execução, evento	Inicia a preparação do evento passado como argumento. Esse método pode gerar também a preparação de todos os eventos que estejam relacionados com o evento passado como parâmetro.
unprepare	Objeto de execução	Coloca todos os eventos do objeto de execução no estado dormindo. Se algum evento estiver ocorrendo, sua ocorrência deve ser antes abortada.
start	Objeto de execução, evento não interativo	Inicia a ocorrência do evento passado como parâmetro. O evento deve pertencer ao objeto de execução também passado como parâmetro. Se o evento não estiver preparado, a preparação deve ser automaticamente iniciada. Se o evento estiver sendo preparado, a ocorrência deve ser iniciada logo em seguida. Se o evento estiver em qualquer outro estado diferente de *dormindo*, *preparando* e *preparado*, a ação deve ser ignorada.
stop	Objeto de execução, evento não instantâneo	Encerra a ocorrência do evento passado como parâmetro. O evento deve pertencer ao objeto de execução também passado como parâmetro. Se o evento estiver em qualquer estado diferente de *ocorrendo* e *suspenso*, a ação deve ser ignorada. Se o evento for nulo a ação deve ser aplicada a todos os eventos não instantâneos do objeto de execução.
pause	Objeto de execução, evento não instantâneo	Suspende a ocorrência do evento não instantâneo. O evento deve pertencer ao objeto de execução passado como parâmetro. Se o evento estiver em qualquer estado diferente de *ocorrendo*, a ação deve ser ignorada. Se o evento for nulo a ação deve ser aplicada a todos os eventos não instantâneos do objeto de execução.
resume	Objeto de execução, evento não instantâneo	Retoma a ocorrência do evento não instantâneo. O evento deve pertencer ao objeto de execução passado como parâmetro. Se o evento estiver em qualquer estado diferente de *suspenso*, a ação deve ser ignorada. Se o evento for nulo a ação deve ser aplicada a todos os eventos não instantâneos do objeto de execução.
abort	Objeto de execução, evento não instantâneo	Aborta a ocorrência do evento não instantâneo. O evento deve pertencer ao objeto de execução passado como parâmetro. Se o evento estiver em estado diferente de *ocorrendo* e *suspenso*, a ação deve ser ignorada. Se o evento for nulo a ação deve ser aplicada a todos os eventos não instantâneos do obj de execução.
changeEventState	Objeto de execução, evento, novo estado	Se possível, coloca o evento no novo estado passado como argumento. O evento deve pertencer ao objeto de execução também passado como parâmetro[12].
setAttributeValue	Objeto de execução, evento de atribuição, valor, atribuição relativa	Atribui o valor ao atributo identificado pelo evento de atribuição. O evento deve pertencer ao objeto de execução passado como parâmetro. O parâmetro atribuição relativa informa se o valor deve ser atribuído de maneira absoluta ou relativa.
getAttributeValue	Objeto de execução, evento de atribuição	Retorna o valor do atributo identificado pelo evento de atribuição. O evento deve pertencer ao objeto de execução.
setTimeControlVisible	Objeto de execução, booleano	Habilita ou desabilita uma barra para controle temporal da exibição do objeto de execução.

Tabela 4 – Interface que deve ser oferecida por toda ferramenta de exibição.

[12] Esse método objetiva permitir que as ferramentas lidem com extensões das máquinas de estados dos eventos, sem que seja obrigatória a criação de novos métodos na interface.

O método *initialize* recebe, como parâmetro, o objeto de execução a ser exibido (objeto de dados, descritor e eventos). Cabe à ferramenta extrair dos eventos os atributos e as regiões correspondentes no conteúdo do objeto (as âncoras), para efetuar o controle das respectivas máquinas de estados. A interface permite que uma mesma ferramenta controle a exibição de mais de um objeto de execução. Por esse motivo todos os métodos esperam um parâmetro que identifica o objeto de execução na ferramenta. Novos objetos são incluídos ou retirados de uma ferramenta com chamadas, respectivamente, aos métodos *addExecutionObject* e *removeExecutionObject*.

Os métodos *prepare* e *unprepare* atuam na preparação dos eventos. Normalmente, o método *prepare* será chamado com o evento de apresentação que identifica o conteúdo inteiro do objeto de execução. Geralmente, a preparação do evento mais abrangente resultará também na preparação dos demais eventos do objeto. A preparação dos eventos de exibição envolve a busca das unidades de informação do conteúdo (ao menos parcialmente) e seu armazenamento em um *buffer* da ferramenta. O método *unprepare* permite requisitar a liberação desses espaços alocados quando não houver mais interesse por parte do formatador em manter os eventos preparados.

O método *start* inicia a ocorrência do evento não interativo passado como parâmetro. Normalmente, esse evento será o evento de exibição anteriormente preparado. Quando for realizada uma chamada para o método *start* de um evento que esteja ainda sendo preparado (encontrar-se no estado *preparando*), a ferramenta deve aguardar o término da preparação e iniciar a ocorrência do evento logo em seguida.

Os métodos *stop*, *pause*, *resume* e *abort* recebem, todos, um objeto de execução e um evento não instantâneo como parâmetros e devem ser aplicados conforme as descrições dadas na terceira coluna da Tabela 4. Todos esses métodos podem também ser chamados recebendo apenas um objeto de execução como argumento. Nesse caso, as ações desempenhadas devem ser feitas sobre todos os eventos não instantâneos do objeto de execução que estiverem no estado adequado.

Os métodos *setAttributeValue* e *getAttributeValue* permitem que o formatador interaja com as ferramentas de exibição, modificando e consultando os

valores dos atributos dos eventos e dos objetos de execução. Supondo que a ferramenta controle a apresentação de uma mídia contínua, uma importante utilidade do método *setAttributeValue* é permitir que o formatador aumente ou diminua a taxa de apresentação do conteúdo, para tentar garantir que as relações de restrição especificadas pelo autor sejam respeitadas.

O método *changeState* na interface procura oferecer flexibilidade para que novas máquinas de estados sejam definidas sem que isso cause impacto na estrutura proposta. A interface também não restringe as classes dos eventos, possibilitando que novas extensões sejam definidas. Obviamente, a implementação da ferramenta que for efetivamente controlar a exibição do objeto, dos novos tipos de eventos e das novas máquinas de estados deverá conhecer os detalhes das extensões.

O método *setTimeControlVisible* é útil para a apresentação de objetos de execução, cujos eventos de exibição associados tenham durações bem definidas. Isso será mais comum para os objetos associados a vídeos e áudios. O método espera um valor booleano, que quando verdade, exibe um componente de interface gráfica para o controle temporal da apresentação. Normalmente, esse controle irá possuir botões e barras de progressão que permitem realizar ações sobre a exibição do objeto, tais como iniciar, encerrar, suspender, reassumir, modificar o ponto da exibição etc.

Antes de discorrer sobre os aspectos da implementação de ferramentas de exibição e de um exemplo de formatador hipermídia que segue a estruturação apresentada neste capítulo, o próximo capítulo discute os aspectos relacionados às funcionalidades essenciais para que se busque apresentações de hiperdocumentos com qualidade.