4
Pré-busca, Orquestração Intra-Mídia e Adaptação de Apresentações Hipermídia

Este capítulo analisa os requisitos para o projeto e implementação de mecanismos de pré-busca, orquestração de qualidade de serviço e adaptação das apresentações em formatadores hipermídia. Apesar desses mecanismos não serem essenciais, a presença deles no ambiente de execução contribui para uma maior proximidade entre os resultados finais das apresentações e as especificações dos autores.

Para cada mecanismo, a partir do levantamento de seus requisitos, o capítulo propõe uma arquitetura genérica que possa ser reusada no desenvolvimento de formatadores específicos. Adicionalmente, com a especificação de interfaces bem definidas para as estruturas genéricas, pretende-se facilitar a integração de módulos de pré-busca, orquestração e adaptação com outros componentes dos sistemas de apresentação hipermídia, inclusive permitindo que as estratégias desses mecanismos venham a ser incorporadas em implementações de formatadores já existentes.

4.1
Pré-busca em apresentações hipermídia

Para que o usuário perceba uma boa qualidade em uma apresentação hipermídia, diversos parâmetros precisam ser controlados pelo formatador. Dentre os parâmetros temporais destacam-se: a latência para exibição dos objetos e a taxa para apresentação dos objetos de mídia contínua. A *latência de exibição* corresponde ao intervalo de tempo entre o instante em que o conteúdo de um objeto de execução deve ser exibido e o instante em que o mesmo é efetivamente apresentado[13]. Efeitos desagradáveis que podem ser percebidos pelo usuário

[13] Note que o termo latência no contexto desta tese não se refere exclusivamente ao retardo de transferência na rede, mas pode englobar também atrasos no sistema operacional e na própria ferramenta de exibição.

quando ocorre uma latência elevada são a perda da interatividade e a perda da sincronização entre os objetos de execução, sem falar na própria impaciência causada pela espera.

As *taxas de apresentação* são características dos objetos de mídia contínua, tais como vídeo e áudio. Essas taxas normalmente envolvem uma quantidade de amostras por segundo e uma resolução para cada uma das amostras. Quando métodos de compressão são aplicados em conteúdos desse tipo, as taxas de busca podem variar ao longo do tempo. No entanto, para as ferramentas de exibição, as taxas de apresentação devem se manter em níveis constantes, pois variações podem resultar em desconforto, ou mesmo na total incompreensão da informação.

O limite máximo para a latência de exibição de um objeto varia em função do tipo de seu conteúdo (texto, imagem, áudio, vídeo etc.) e do tipo de aplicação hipermídia onde o objeto se encontra inserido (aplicação não interativa, aplicação interativa etc.). Na Web, por exemplo, algumas pesquisas constatam que uma espera de até cinco segundos para carregar uma nova página no *browser* é tida pelos usuários como satisfatória (Bhatti et al., 2000). No entanto, em aplicações que requeiram transmissão de voz, esse limite pode ter de ser reduzido para cerca de 150 milisegundos (ITU, 1996). O limite máximo para a latência de exibição vai depender também dos relacionamentos existentes com outros objetos (elos de sincronização, elos de restrição etc.). Flutuações na taxa de exibição (*jitter*) afetam apenas as mídias contínuas e devem ser sempre evitadas.

Técnicas de cache têm sido amplamente utilizadas nas mais diversas áreas da computação com o intuito de reduzir o tempo de resposta dos sistemas. Os mecanismos de cache procuram posicionar os dados em uma memória que ofereça ao sistema um tempo de acesso mais baixo que o experimentado quando os dados são trazidos de seu local original de armazenamento. A eficiência de um mecanismo de cache está diretamente relacionada à freqüência com que os dados são encontrados na memória cache, quando requisitados pelo sistema (*cache hit*). Áreas na memória cache (principalmente em RAM), funcionando como buffers, também podem ser úteis no auxílio a estratégias de compensação das variações estatísticas dos retardos, permitindo corrigir variações bruscas nas taxas de fornecimento dos dados.

Embora o uso da memória cache possibilite uma redução nos tempos de resposta do sistema, estudos mostram que as técnicas de cache em sistemas hipermídia, principalmente na Web, apresentam eficiência bem inferior (30-60%) à obtida com as técnicas de cache em hardware (90%) (Cohen & Kaplan, 2000; Feldmann et al., 1999; Khan & Tao, 2001a; Kroeger et al., 1997). Além dos vários aspectos relacionados às técnicas de geração de páginas dinâmicas que influenciam numa menor eficiência dos mecanismos de cache, existe também uma quantidade grande de dados que são requisitados pelo ambiente de execução uma única vez. A eficiência baixa, somada ao fato das apresentações hipermídia muitas vezes necessitarem que os objetos possuam uma latência de exibição baixa, mesmo na primeira vez em que são exibidos, e à questão dos próprios buffers para compensação da variação estatística causarem um atraso adicional na exibição dos objetos de mídia contínua, faz com que seja importante a existência de estratégias de pré-busca (*prefetch*) nos formatadores.

A principal responsabilidade do mecanismo de pré-busca é procurar garantir que o conteúdo de cada objeto (não necessariamente o objeto inteiro) esteja pronto para a ferramenta de exibição no momento em que a mesma precise exibi-lo, fazendo com que as latências de apresentação durante a execução dos documentos encontrem-se dentro dos limites aceitáveis. Como conseqüência da antecipação, para o caso de objetos de mídia contínua, o mecanismo de pré-busca também possibilita que durante a exibição não haja uma ausência de dados (*buffer underflow*) para serem consumidos pelas ferramentas de exibição.

O ponto de partida para o compilador de pré-busca é percorrer a cadeia temporal principal no plano de execução e estimar o tempo despendido na preparação de cada um dos objetos de execução. Para que os tempos estimados se aproximem da realidade, é necessário que o compilador de pré-busca obtenha informações a respeito dos objetos de execução. O compilador de pré-busca deve consultar o conteúdo de cada objeto para tentar descobrir o seu tamanho (ou duração). Além do tamanho, o compilador deve tentar encontrar no descritor de cada objeto o percentual mínimo do conteúdo total que precisa estar disponível antes do início da apresentação e a latência máxima permitida para a exibição do objeto.

Além das informações provenientes dos objetos de execução, o compilador de pré-busca também precisa dispor de pelo menos três informações adicionais do contexto de exibição, para estimar os tempos de busca. A primeira é o retardo de propagação dos dados do local em que estão armazenados até o buffer da ferramenta[14], a segunda informação é a taxa de transmissão disponível para transferência dos dados[15] e, por fim, a terceira informação é o próprio retardo imposto pela ferramenta de exibição para sua criação e iniciação, que muitas vezes não deve ser desprezado. Para as informações que não estiverem disponíveis nos objetos ou no contexto de exibição, o compilador deve utilizar heurísticas para a elas atribuir valores. A Figura 19 apresenta a fórmula geral para o cálculo do tempo de preparação de um objeto de execução.

```
T : tamanho do objeto (em bits)
pc: percentual (entre 0 e 1) do tamanho total que deve estar disponível
L : latência máxima para exibição do objeto (em segundos)
r : retardo de propagação imposto pela plataforma (em segundos)
tx: taxa de transmissão (em bits por segundo)
f : retardo imposto pela ferramenta (em segundos)

Tempo de preparação (s) = ((T * pc) / tx) + r + f - L
```

Figura 19 – Fórmula geral para o cálculo do tempo de preparação de um objeto de execução.

Além de calcular os tempos de preparação para cada um dos objetos de execução, a outra função do compilador de pré-busca é determinar a ordem das preparações, para com isso realizar a construção efetiva do plano de pré-busca. Esse procedimento deve ser feito com o uso de alguma estratégia de compilação de pré-busca. Na realidade, é a própria estratégia que deve coordenar o cálculo das estimativas de duração das preparações, estabelecendo inclusive as heurísticas para determinar os valores para os parâmetros citados anteriormente que eventualmente não estejam disponíveis.

Alguns aspectos importantes devem ser considerados na construção do plano de pré-busca, pois é importante que haja um planejamento na escolha dos

[14] Esse retardo pode englobar o próprio tempo para requisitar o objeto, tempo de *seek* no acesso a disco etc.

[15] Note que a taxa de transmissão na pré-busca não precisa ter qualquer relação com a taxa de exibição do conteúdo.

tempos e na ordenação das requisições de pré-busca. Duas situações podem ocorrer se os instantes para iniciar a busca dos conteúdos não forem bem escolhidos (Jeong et al., 1997; Oren, 2000): a busca pode ser iniciada tardiamente e o objeto não estar preparado na hora da sua exibição; ou a busca pode iniciar cedo demais e o objeto ter de permanecer armazenado na memória cache durante muito tempo. No primeiro caso, pode ser introduzida uma latência maior que a permitida na exibição. Se houver pelo menos um relacionamento de sincronização temporal com paralelismo entre o objeto cuja preparação atrasou e um outro qualquer, pode ser que a sincronização seja perdida ou então que seja necessário retardar a exibição também do outro objeto. A antecedência exagerada na busca pode trazer duas conseqüências desfavoráveis. O objeto pode ter de ser descartado pela política de cache, gerando uma pré-busca inútil, ou pode obrigar que um outro objeto em uso, ou necessário em um futuro próximo, seja descartado, causando uma poluição na cache. A primeira situação (início tardio da busca) pode ser amenizada impondo um retardo maior para iniciar a exibição do documento como um todo. Isso aumenta o tempo de compilação e o tamanho necessário para a memória cache. A segunda situação pode ser evitada por meio de um controle mais apurado da alocação dos recursos na plataforma ou simplesmente incrementando o tamanho da memória cache.

Essas questões acrescentam dois parâmetros que devem ser considerados pelo compilador de pré-busca. O primeiro é o espaço disponível em memória (espaço na cache) para trazer os conteúdos dos objetos com antecedência. O segundo é a latência que será introduzida para iniciar a exibição do documento como um todo. O primeiro parâmetro é uma informação que deve ser obtida do gerente de contexto, pois está relacionado com a plataforma de exibição. Já o segundo parâmetro pode ser definido pela própria estratégia, ou estipulado pelo usuário do mecanismo de pré-busca.

Uma estratégia simples de compilação da pré-busca pode subtrair do instante esperado para exibição de cada objeto, o tempo estimado para sua preparação. Essa solução deve, no entanto, considerar as divisões da banda quando mais de um objeto tiver que ser buscado paralelamente, evitando causar tempos de preparação maiores que os previstos e, por conseqüência, latências inaceitáveis.

Para que a estratégia não precise se preocupar com divisões da banda passante disponível, a alternativa é serializar as buscas. Nesse caso, os objetos devem ser primeiramente ordenados em função dos instantes esperados para suas exibições. Os objetos que devem ser apresentados primeiro, devem ser preparados na frente. Quando mais de um objeto tiver que iniciar sua exibição ao mesmo tempo, o melhor é que os objetos com tempos de preparação maiores sejam deixados para depois (Jeong et al., 1997).

Além de serializar as buscas, o ideal é que a estratégia de compilação identifique, após o plano estabelecido, quais os objetos na cadeia temporal principal estão com uma latência de exibição ultrapassando o valor máximo permitido. Todas as violações devem ser somadas e o resultado utilizado para impor um retardo ao início da exibição do documento. Evidentemente, a estratégia deve observar se esse atraso inicial respeita a exigência de retardo do usuário e a capacidade da memória cache. Caso isso não seja possível, existem duas opções. A primeira é simplesmente iniciar a apresentação do documento respeitando as características do usuário e da plataforma e deixar para ajustar a apresentação durante a sua execução. A segunda opção é requisitar, ainda em compilação, os serviços do módulo de adaptação, para que o plano de execução do documento seja ajustado. No entanto, é importante que a estratégia considere que os próprios cálculos e ajustes do plano consomem o tempo da compilação.

Um outro aspecto que a estratégia de compilação da pré-busca deve levar em conta é a escolha do momento para iniciar a preparação dos objetos contidos em cadeias temporais auxiliares, pois é importante que, mesmo para os objetos com exibições imprevisíveis, sempre que possível a latência seja minimizada, como demonstram estudos anteriormente mencionados (Bhatti et al., 2000; Khan & Tao, 2001a). O ideal é que a estratégia consiga obter, ou diretamente do plano de execução, ou junto ao contexto de exibição, a probabilidade para que cada uma das cadeias auxiliares tenha sua execução iniciada. De posse das probabilidades e dos tempos estimados para preparação, o orquestrador de pré-busca pode ponderar esses valores e, em certas ocasiões, optar por também preparar objetos com exibições imprevisíveis de maior probabilidade.

Com o plano de pré-busca construído, o *executor do orquestrador de pré-busca* inicia sua tarefa que consiste em consultar o plano e disparar, nos instantes

programados, notificações ao executor para requisitar junto às ferramentas de exibição (ou seus adaptadores) o início da preparação de cada um dos objetos. Pode ser que, ao invés do conteúdo ser buscado pela ferramenta, exista no sistema um módulo único que coordene de maneira centralizada as buscas e o armazenamento dos dados.

Durante a fase de execução da pré-busca, o executor deve monitorar, junto ao plano de execução, qualquer alteração que ocorra nos instantes de tempo previstos para iniciar a exibição dos objetos de execução, incluindo a ocorrência de exibições inicialmente classificadas como imprevisíveis pelo compilador de apresentação. Quando necessários, os devidos ajustes devem ser feitos pelo executor no plano de pré-busca. Cabe ao executor também observar os tempos efetivamente despendidos nas preparações e compará-los com os tempos estimados, ajustando o plano de pré-busca quando for necessário. Se por acaso o executor de pré-busca não conseguir ajustar o plano e manter as restrições de latência impostas pelo documento, o orquestrador de pré-busca deve informar o fato ao orquestrador de apresentação, para que as adaptações sejam aplicadas sobre o plano de execução.

A próxima seção ilustra o framework proposto para implementação de mecanismos de pré-busca em formatadores hipermídia.

4.1.1 Arquitetura para implementação de mecanismos de pré-busca

A Figura 20 apresenta o diagrama de classes que visa facilitar o projeto e a implementação de mecanismos de pré-busca em apresentações hipermídia. A classe *Prefetcher* representa o orquestrador de pré-busca, correspondendo à mesma classe já comentada na Seção 3.2 (Figura 17). Essa classe funciona como a interface (fachada) do mecanismo oferecida aos demais componentes do formatador, ou a qualquer outro módulo que deseje utilizar os serviços de pré-busca de objetos hipermídia. O compilador (*PrefethCompiler*), o executor (*PrefetchExecutor*) e o plano de pré-busca (*PrefetchPlan*) constituem-se nos três principais componentes do orquestrador de pré-busca.

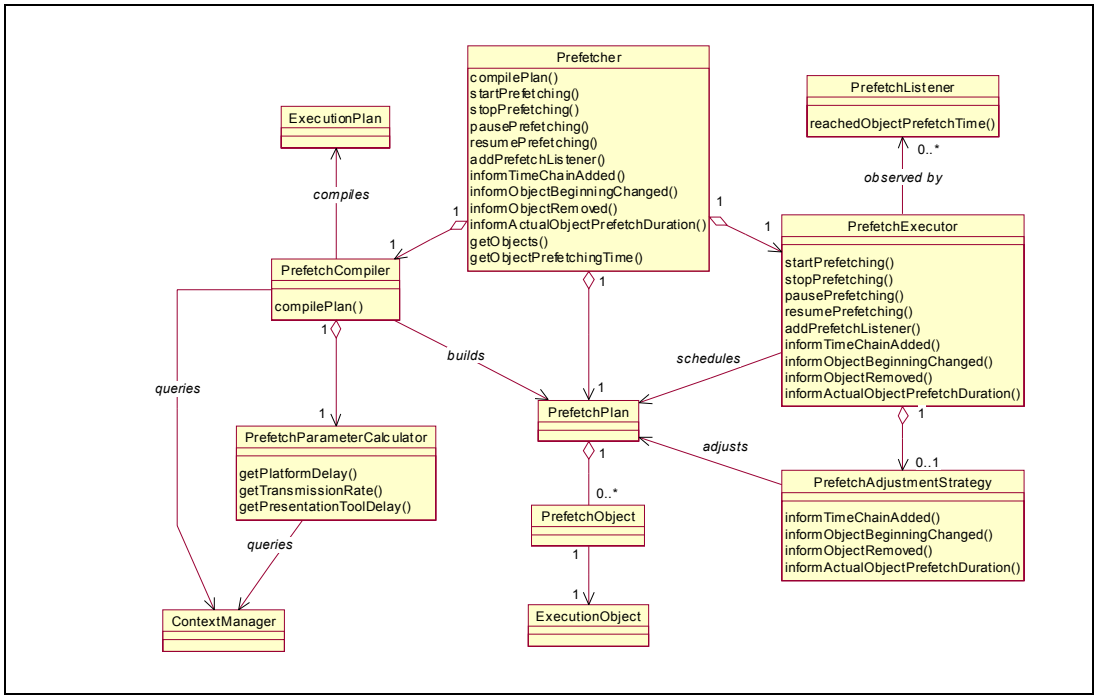

Figura 20 – Diagrama de classes para implementação de mecanismos de pré-busca em documentos hipermídia.

A classe *PrefetchCompiler* é, na verdade, uma estratégia (*design pattern Strategy* (Gamma et al., 1995)) que deve ser especializada em cada implementação particular de mecanismo de pré-busca. O compilador é responsável por efetuar a construção do plano de pré-busca a partir do plano de execução (*ExecutionPlan*) e dos dados provenientes do proxy de informações contextuais (*ContextManager*). Para tanto, ele deve utilizar os serviços de um elemento calculador (*PrefetchParameterCalculator*) que, para um determinado objeto de execução, informa os valores de cada um dos parâmetros da plataforma de exibição, relevantes na estimativa das durações de preparação (Figura 19). Implementações dessa classe de cálculo também devem prover heurísticas que definam valores para os vários parâmetros, quando não for possível obtê-los do gerenciador de contexto. A obtenção dos parâmetros relacionados aos objetos de execução e o cálculo efetivo do tempo de preparação devem ser realizados pela própria estratégia de compilação. Também cabe a essa estratégia observar se os limites de tamanho dos *buffers* e da latência para iniciar a exibição estão sendo cumpridos. Tanto o compilador (estratégia de compilação) como o calculador são pontos de flexibilização (*hotspots*) do framework, que devem ser preenchidos em função dos requisitos de formatação.

O executor de pré-busca (*PrefetchExecutor*) reúne as ações que devem ser desempenhadas após a construção, ao menos parcial, do plano de pré-busca. Sendo assim, a principal função do executor é escalonar o plano e notificar os objetos que tenham se registrado como observadores do mecanismo de pré-busca. Esse registro é feito através de uma chamada ao método *addPrefetchListener* do orquestrador, que por sua vez repassa a chamada ao executor. Na invocação do método, o objeto observador deve ser passado como parâmetro. Para isso, classes que desejem atuar como observadores do mecanismo devem ser do tipo *PrefetchListener* e implementar o método *reachedPrefetchTime* com o código para o tratamento das notificações. O método *reachedPrefetchTime* informa para os seus observadores o objeto de execução que teve o seu instante de preparação atingido.

O escalonamento do plano é iniciado com uma chamada ao método *startPrefetching* do orquestrador de pré-busca, que simplesmente repassa essa chamada para o executor. A qualquer momento, chamadas aos métodos *pausePrefetching* e *resumePrefetching* podem ser invocadas para, respectivamente, suspender ou retomar o escalonamento. Além disso, o método *stopPrefetching* permite que o processo de pré-busca seja encerrado.

Além de escalonar as ações de busca e preparação dos conteúdos, é desejável, porém não obrigatório, que o executor de pré-busca também possua uma estratégia de ajuste (*PrefetchAdjustStrategy*). A função dessa estratégia é estar atenta a modificações no plano de execução e distorções entre os tempos de preparação previstos e os tempos efetivamente gastos, para realizar correções no plano de pré-busca quando for necessário. A estratégia de ajuste pode ser informada, através de uma chamada ao seu método *informTimeChainAdded*, que uma cadeia temporal auxiliar foi associada à cadeia principal, como resultado da ocorrência de um evento imprevisível ou de uma adaptação dinâmica do formatador. Além desse tipo de modificação do plano de execução, a estratégia também pode ser notificada da alteração no instante previsto para iniciar a exibição de um objeto de execução, por meio de uma chamada ao método *informObjectBeginningChanged* e da remoção de um objeto do plano de execução, por intermédio de uma chamada ao método *informObjectRemoved*. Por fim, o método *setActualObjectPrefetchDuration* permite que a estratégia de ajuste

seja avisada do tempo efetivamente gasto na preparação de um determinado objeto de execução. Se no executor de pré-busca não estiver disponível a estratégia de ajuste, as chamadas a esses métodos devem ser simplesmente ignoradas. Do mesmo modo que no compilador, a estratégia de ajuste é um ponto de flexibilização na arquitetura proposta.

Como pode ser percebido, todos os métodos oferecidos na interface do orquestrador de pré-busca comentados até o momento terminam sempre por serem repassados aos componentes internos do mecanismo. Isso é feito com o intuito de evitar que usuários dos serviços de pré-busca precisem tomar conhecimento de todas as classes internas e de seus funcionamento, podendo interagir apenas com a classe *Prefetcher*.

O método *getExecutionObjects*, oferecido pelo orquestrador, retorna a lista dos objetos de execução inseridos no plano de pré-busca, já ordenados pelos respectivos instantes programados de preparação. Esses instantes podem ser obtidos através de invocações ao método *getObjectPrefetchingTime*, passando o objeto como parâmetro. Esses métodos podem ser úteis, principalmente, em plataformas que permitam reservar recursos antecipadamente, pois evitam que o processo de negociação da QoS para transmissão do conteúdo seja iniciado somente no instante que o escalonador (executor) notifica que a preparação deve ser começada. Os aspectos de negociação e manutenção da QoS nas transmissões dos conteúdos dos objetos são comentados na próxima seção.

4.2
Orquestração intra-mídia em apresentações hipermídia

O conceito de orquestração da qualidade de serviço está relacionado com a gerência integrada da alocação dos recursos e do escalonamento de suas utilizações, quando os mesmos encontram-se distribuídos por mais de um subsistema (Gomes, 1999). No controle de apresentações de documentos hipermídia, a questão da orquestração da qualidade de serviço aparece em dois níveis distintos: orquestração da QoS inter-mídia e orquestração da QoS intra-mídia.

A orquestração inter-mídia refere-se à manutenção dos relacionamentos de sincronização, tanto causais como de restrição, sendo responsabilidade do

orquestrador de apresentação, conforme já discutido no Capítulo 3. Também de acordo com o que já foi comentado, na tarefa de orquestração inter-mídia, os mecanismos de adaptação e de pré-busca atuam como importantes colaboradores.

O segundo nível de orquestração de qualidade de serviço, que é justamente o foco desta seção, é a orquestração intra-mídia. Essa orquestração está diretamente relacionada com a tarefa de buscar e exibir cada um dos objetos de execução, procurando fazer com que a qualidade de suas apresentações sejam mantidas. Para que isso seja possível, recursos devem ser reservados tanto nas estações cliente (formatador) e servidora de conteúdo, como também no provedor de comunicação entre essas estações. Caberá ao formatador negociar a QoS necessária junto aos provedores de comunicação e processamento, dando início ao processo de orquestração e reserva de recursos, tais como CPU e banda passante.

No formatador, o gerenciador intra-mídia é o módulo responsável por desempenhar esse papel de negociador. Na realidade, além de negociar a QoS para as exibições, o orquestrador deve também procurar monitorar se os acordos estabelecidos estão sendo respeitados, notificando os executores de apresentação e de pré-busca, sempre que não for possível manter a exibição de um objeto com a qualidade desejada.

Para cada conteúdo de objeto de execução a ser exibido, o gerenciador deve solicitar que um orquestrador intra-mídia tente distribuir a reserva de recursos desde o servidor onde o conteúdo encontra-se no momento gravado, passando pelo provedor de comunicação, até o *buffer* do formatador (ou da própria ferramenta de exibição), que servirá de memória cache para armazenar o conteúdo localmente. Além disso, se possível, a orquestração e reserva deve também procurar alocar recursos no sistema operacional da máquina onde executa a ferramenta de exibição, a fim de garantir que ela terá à sua disposição a capacidade de processamento necessária. Vale salientar que, nos casos em que a preparação não coloca previamente em cache todo o conteúdo do objeto, a reserva dos recursos no caminho até o local original de armazenamento das unidades de informação deve ser mantida durante a fase de apresentação do objeto de execução, até que todo o conteúdo do objeto seja obtido.

Note que, para completar o processo de reserva, pode ser necessário que se faça também uma negociação de alocação de recursos no sistema operacional da

máquina servidora de conteúdo, para que o processo servidor tenha garantidos, entre outros parâmetros, o retardo máximo e a banda passante no acesso aos dados armazenados. Como já colocado na Seção 3.1, a presença do gerenciador intra-mídia no formatador está condicionada à existência de algum suporte à reserva de recursos na rede ou nos sistemas operacionais.

A Figura 21 utiliza a proposta do Modelo de Composição de Serviços (*SCM – Service Composition Model*), definido em (Colcher, 1999), para ilustrar os componentes e provedores envolvidos na transmissão e exibição do conteúdo de um objeto de execução em uma apresentação hipermídia.

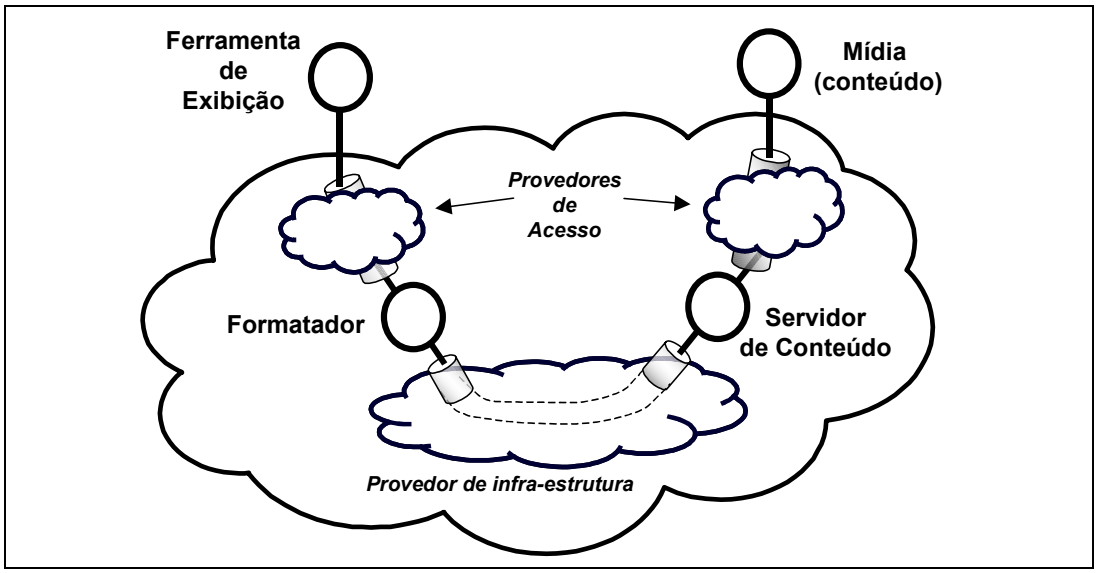

Figura 21 – Visão simplificada da exibição de conteúdos de documentos hipermídia representada através do modelo de composição de serviços[16].

O SCM define dois elementos básicos: os *componentes de usuário*, que correspondem às entidades que fazem uso direto dos serviços (na figura, representados como círculos), e os *provedores*, que são responsáveis pela provisão do serviço (representados como nuvens). Uma abstração especial do modelo, denominada *MediaPipe* (ilustrada como cilindros), permite capturar a idéia de uma associação entre componentes de usuário, sobre a qual especificações de QoS podem ser definidas e gerenciadas. Para que possa ser estabelecido um *MediaPipe* entre dois componentes, o provedor poderá ser formado por outros componentes de serviço (entidades de software ou hardware que implementem o serviço). Esses

[16] Para não sobrecarregar o desenho, foi ilustrada a associação apenas entre uma ferramenta e um conteúdo de objeto.

componentes, por sua vez, provavelmente utilizarão os serviços de um provedor de nível mais baixo de abstração, formando assim uma estruturação aninhada de provisão de serviços, que pode se repetir sucessivamente até que sejam alcançados os recursos físicos propriamente ditos (enlace de comunicação, CPU etc.). Os provedores do modelo podem ser classificados em *provedores de infra-estrutura* e *provedores de acesso*. O primeiro tipo permite o estabelecimento de *MediaPipes* entre componentes localizados em um mesmo nível de abstração, enquanto o segundo tipo possibilita o estabelecimento dessas associações entre um componente de usuário do serviço e um componente interno do provedor do serviço. Nada impede que os componentes de usuário sejam *componentes compostos*, constituídos de outros componentes de usuário e provedores interligando esses componentes.

Na Figura 21, o formatador, ou mesmo a ferramenta, deve conter um componente responsável por recuperar as unidades de informação do servidor de conteúdo e armazená-las em um *buffer* do formatador ou direto da ferramenta. Essa busca dos dados deve seguir uma implementação de protocolo que o servidor de conteúdo compreenda, tais como HTTP, RTP etc.

Quando o conteúdo de um objeto de execução encontrar-se armazenado na mesma máquina onde a ferramenta de exibição executa, o provedor de infra-estrutura deixa de existir no processo de busca do conteúdo e os papéis de servidor e formatador se fundem em um único componente, conforme ilustra a Figura 22. Nesse caso, embora desenhados como dois provedores de acesso, existirá apenas um provedor intermediando todas as comunicações, que será justamente o sistema operacional da máquina.

As duas figuras anteriores retratam o caminho percorrido pelos dados no serviço de apresentação das mídias. Atuando em um outro plano estão as orquestrações de QoS inter-mídia e intra-mídia, procurando garantir que o serviço de apresentação dos conteúdos encontre-se dentro dos limites desejados pelo autor e pelo usuário. Sendo assim, a orquestração de QoS pode ser considerada como um meta-serviço que age sobre o serviço de transmissão e exibição dos conteúdos. Meta-serviços são modelados pelo SCM utilizando os mesmos conceitos de componentes e provedores, em um plano separado. Através de interfaces *meta* nos componentes e provedores do plano de serviços (ou plano de dados), os

componentes do plano de meta-serviços (ou plano de controle) têm condições de manipular as entidades que implementam o serviço e suas associações (*MediaPipes*). Exemplo de aplicação dos conceitos de plano de meta-serviços e plano de serviços são encontrados, respectivamente, nas pilhas de controle e de dados das redes ATM. Outro exemplo são os protocolos de construção de tabelas de rotas (por exemplo: RIP, OSPF, BGP etc.) que atuam sobre o serviço de roteamento das informações (por exemplo: IP).

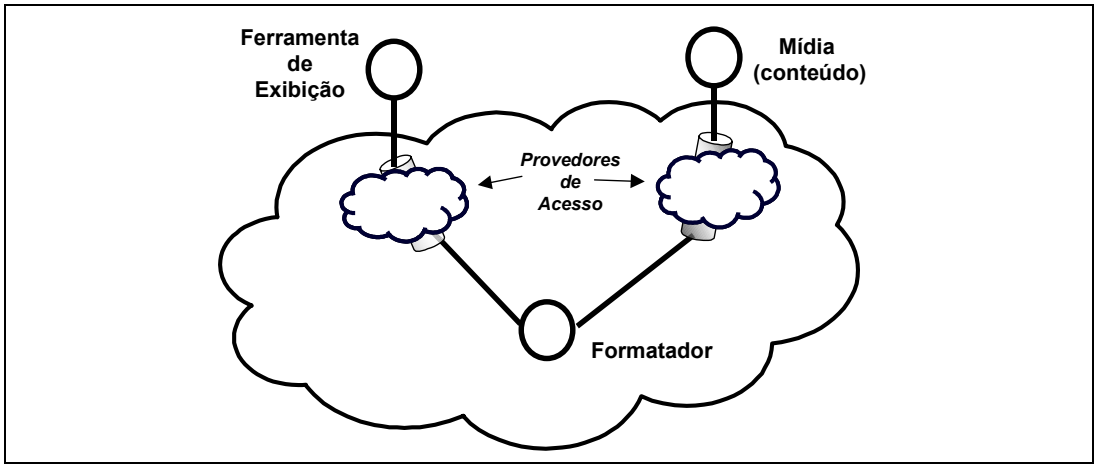

Figura 22 – Exibição de conteúdos armazenados localmente, eliminando a figura do provedor de infra-estrutura.

A Figura 23 ilustra os conceitos de serviço e meta-serviço no projeto de ambientes de execução hipermídia, estendendo e detalhando as informações anteriormente apresentadas na Figura 21 e na Figura 22. A figura é também uma nova visão para a Figura 16, descrita na Seção 3.1, mas com um destaque para o posicionamento das funções de formatação nos dois planos.

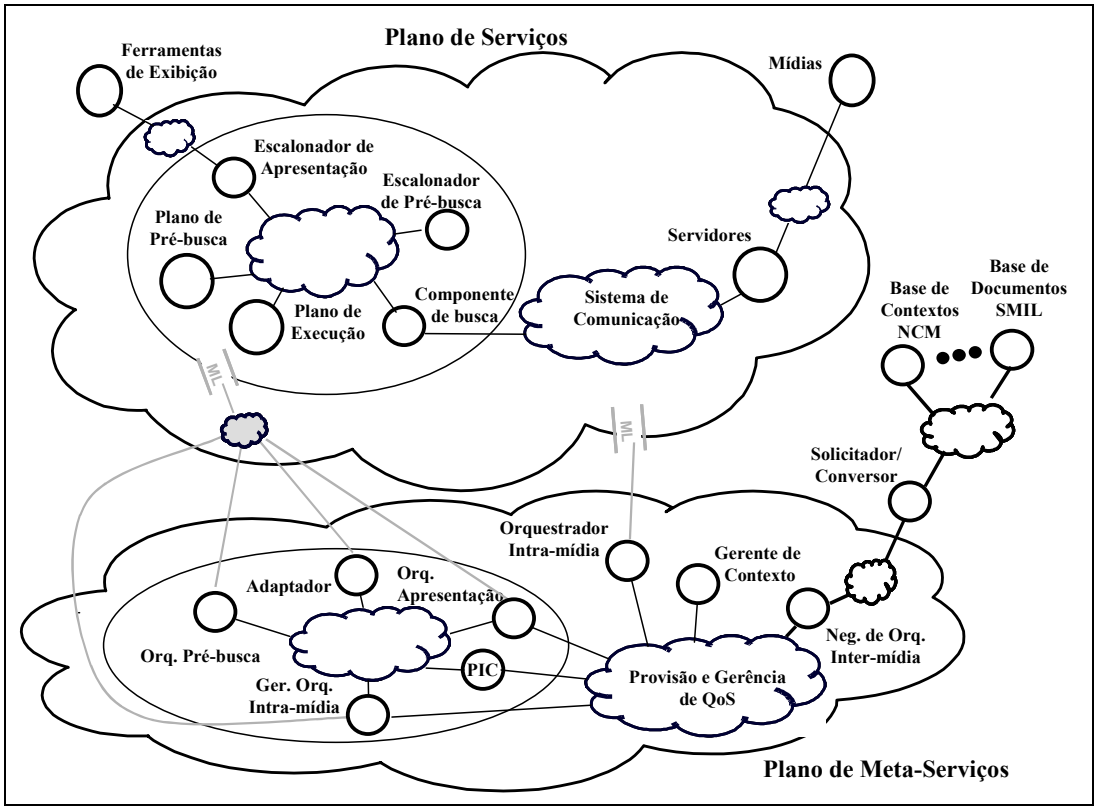

Figura 23 – Ilustração dos conceitos de serviço e meta-serviço nos ambientes de execução hipermídia.

A Figura 23 permite rever de uma forma sistemática os procedimentos desempenhados no controle da apresentação dos hiperdocumentos, enumerados a seguir. Como alguns dos procedimentos podem ocorrer de maneira concorrente, a ordem de apresentação não deve ser entendida como uma seqüência de execução.

procedimento 1: O *Solicitador* busca um documento (ou parte de um documento) e, caso seu modelo seja diferente do modelo de execução esperado pelo formatador, entrega o documento para ser traduzido por um conversor. Na figura, por simplificação, solicitador e conversor estão representados em um mesmo componente. Além da tradução, o solicitador pode também unir à descrição do documento de entrada, especificações oriundas de outros modelos, como discutido na Figura 3.

procedimento 2: De posse de uma especificação fiel ao modelo de execução, contendo as especificações de QoS intrínsecas ao documento (relacionamentos entre as mídias, parâmetros das características de exibição de cada objeto etc.), o solicitador passa a requisição da apresentação de um contêiner para o negociador de orquestração inter-

mídia. Além das especificações intrínsecas ao documento, restrições de QoS podem ser impostas pelo próprio usuário leitor, se o ambiente de navegação assim permitir. Além disso, o pedido deve também informar o ponto de entrada no contêiner (*evento de apresentação*) por onde a exibição deve ser iniciada.

procedimento 3: O *Negociador de Orquestração Inter-mídia* (o análogo de um *bandwidth broker intra-mídia,* só que para orquestração inter-mídia) tem como função dividir as responsabilidades de formatação entre os vários módulos de orquestração (inter-mídia) de apresentação que estiverem presentes no sistema (cliente, servidor, proxy etc.). Nessa etapa, o negociador pode, por exemplo, aplicar transformações sobre a especificação dos documentos, restringindo as possibilidades de formatação para cada adaptador do sistema. Como este trabalho trata apenas da formatação no lado cliente, essa etapa foi omitida na Seção 3.1, fazendo com que toda a responsabilidade de orquestração recaia sempre sobre o orquestrador de apresentação no cliente.

procedimento 4: O compilador do *Orquestrador de Apresentação* recebe a descrição do documento, juntamente com as informações adicionais do solicitador (ponto de entrada, restrições de QoS do usuário, hora prevista para iniciar a exibição etc.), e constrói o plano de execução. Para isso, o orquestrador consulta, com auxílio do *proxy de informações contextuais* (componente PIC), os parâmetros que definem o contexto de exibição, e requisita ao componente *Adaptador* que implemente os ajustes cabíveis (normalmente, estáticos), tais como seleção de alternativas e ajuste dos tempos elásticos. Também pode ser realizada nessa etapa a verificação da consistência temporal e espacial do documento. Se o mecanismo de pré-busca estiver presente no formatador, o compilador de apresentação aciona o compilador do orquestrador de pré-busca (procedimento 5), caso contrário, essa próxima etapa é ignorada.

procedimento 5: Considerando a existência de um mecanismo de pré-busca no formatador, o compilador de pré-busca aplica sua estratégia para construir o Plano de Pré-Busca, sendo para isso consultadas as características do Plano de Execução e do Contexto de Exibição.

procedimento 6: O controle retorna para o orquestrador de apresentação para encerrar a fase de compilação. Se a hora prevista para iniciar a exibição tiver sido informada pelo solicitador e mecanismos de reserva antecipada estiverem disponíveis no formatador, o compilador de apresentação aciona o gerenciador intra-mídia (procedimento 10), obviamente, se existir no formatador uma implementação desse componente. Caso a hora de início não esteja informada, o gerenciador intra-mídia só é acionado na fase de execução do documento (procedimento 7). De todo modo, o resultado final é a construção, por parte do orquestrador de apresentação, das diversas cadeias temporais do hiperdocumento e o retorno de um identificador que permite disparar a apresentação propriamente dita a partir do ponto de entrada originalmente informado pelo solicitador de exibição.

procedimento 7: O controle da apresentação propriamente dita do documento inicia sob a demanda do usuário. Isso implica iniciar a execução do componente Gerenciador Intra-Mídia no plano de meta-serviços (se isso ainda não tiver sido feito), do componente Escalonador (ou Executor) de Apresentação no plano de serviços e do componente Escalonador (ou Executor) de Pré-busca também no plano de serviços. Evidentemente, se os mecanismos de pré-busca e orquestração intra-mídia não estiverem disponíveis, a última iniciação não será feita.

procedimento 8: Uma vez iniciado, o *Escalonador de Apresentação* (plano de serviços) consulta o plano de execução e, sempre que necessário, instancia uma nova ferramenta de exibição. O escalonador também controla as transições nos estados dos eventos, avalia os elos e envia comandos de ação para as ferramentas sempre que a condição de um elo causal é satisfeita. Ainda como responsabilidade do serviço de execução da apresentação, encontram-se os envios de comandos para acelerar ou desacelerar a taxa de exibição das mídias contínuas, quando algum ajuste estiver programado no plano.

procedimento 9: O *Escalonador de Pré-busca*, no plano de serviços, ao ser iniciado, passa a observar o plano de pré-busca e, toda vez que

encontra uma nova busca programada, requisita ao *componente de busca* que a inicie.

procedimento 10: O *Gerenciador Intra-mídia*, no plano de meta-serviços, consulta o plano de pré-busca, ou mesmo de execução, para descobrir os momentos programados para iniciar as buscas. Para cada busca, o gerenciador solicita a um *Orquestrador Intra-mídia* que negocie a qualidade de serviço para a sua transferência. No processo de negociação, o orquestrador intra-mídia pode dividir as responsabilidades por mais de um orquestrador de QoS (sistema operacional, rede etc.). O *Orquestrador Intra-mídia* também permanece monitorando os orquestradores de QoS com os quais estabeleceu negociações, a fim de identificar qualquer violação (ou possibilidade de violação) no acordo firmado. Quando uma violação é notificada, primeiro o orquestrador tenta redistribuir as responsabilidades entre os orquestradores de QoS e, com isso, sintonizar a QoS intra-mídia. Se isso não for possível, o orquestrador intra-mídia reporta o problema para o gerenciador que, provavelmente, repassará a notificação ao orquestrador de pré-busca ou diretamente ao orquestrador de apresentação.

procedimento 11: O *Executor do Orquestrador de Pré-busca*, no plano de meta-serviços, permanece monitorando o contexto de exibição, o plano de execução, o plano de pré-busca e o gerenciador intra-mídia. Qualquer alteração ou violação informada, o orquestrador procura compensar direto no plano de pré-busca. Se isso não for possível, o orquestrador de pré-busca notifica o executor do orquestrador de apresentação a respeito do problema.

procedimento 12: O *Executor do Orquestrador de Apresentação*, no plano de meta-serviços, permanece monitorando o contexto de exibição, o plano de execução, o orquestrador de pré-busca e o gerenciador intra-mídia. Qualquer alteração ou violação informada, o executor procura ajustar o plano de execução. Quando necessário, o executor pode requisitar também os serviços do componente adaptador.

A próxima seção define a interface para utilização dos serviços de orquestração intra-mídia no formatador.

4.2.1
Arquitetura para implementação da orquestração intra-mídia

A modelagem proposta nesta seção faz uso do framework de orquestração de recursos (ou framework de QoS) proposto em (Gomes 1999; Gomes et al., 2001). As classes preenchidas com a cor cinza, na Figura 24, destacam os componentes do framework de QoS que influenciaram e foram diretamente utilizados na proposta de suporte à implementação de mecanismos de orquestração intra-mídia em formatadores hipermídia. De acordo com o diagrama de classes da figura, o gerenciador (classe *IntraMediaManager*) é a fachada do mecanismo e é definido como sendo composto por um orquestrador intra-mídia (classe *IntraMediaOrchestrator*).

O gerenciador intra-mídia oferece em sua interface cinco principais métodos. O método *prepareOrchestration* recebe o plano (de busca ou de execução, podendo inclusive monitorar ambos) a ser monitorado e, opcionalmente, a hora prevista para a chamada ao seu método *startOrchestration*. Esse último parâmetro é útil apenas em cenários com suporte à reserva antecipada, pois permite que o gerenciador tenha conhecimento dos instantes programados para iniciar as buscas. O método *startOrchestration* deve ser chamado para acionar a orquestração intra-mídia; sua chamada pressupõe que o plano passado na preparação começou a ser executado. O método *stopOrchestration* encerra o processo de orquestração intra-mídia e limpa as estruturas de dados do gerenciador. Por fim, os métodos *addIntraMediaListener* e *removeIntraMediaListener* permitem, respectivamente, acrescentar ou remover um objeto da lista de observadores do mecanismo de gerência da orquestração intra-mídia (objetos da classe *IntraMediaOrchestrationListener*). Esses observadores serão notificados da impossibilidade de estabelecer um novo contrato (chamada ao método *notifyAdmissionFail*) ou violações em contratos já estabelecidos (chamada ao método *notifyTuningFail*). As chamadas a esses métodos informam o objeto de execução com o qual o problema foi identificado.

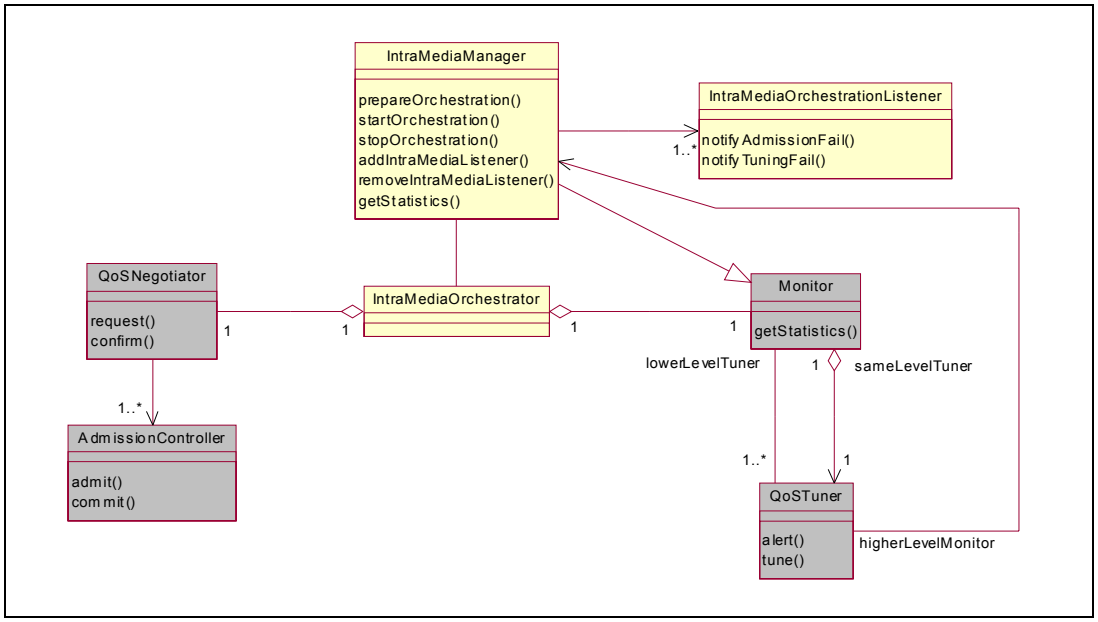

Figura 24 – Diagrama de classes para implementação de mecanismos de orquestração intra-mídia.

A partir de uma análise dos planos, para cada objeto de execução, o gerenciador deve iniciar, junto ao orquestrador, um processo de negociação de reserva de recursos. Essa negociação resultará em uma chamada ao método *request* do negociador do orquestrador (classe *QoSNegotiator*). Cabe ao negociador separar as responsabilidades entre os vários controladores de admissão (classe *AdmissionController*) dos provedores que estejam envolvidos na provisão do serviço e repassar aos mesmos um pedido de admissão. A partir daí, o processo assume o comportamento recorrente definido pelo framework de orquestração de recursos (Gomes, 1999; Gomes et al., 2001). O negociador de QoS fica aguardando que todos os controladores de admissão retornem se foi possível ou não estabelecer a reserva. Em caso afirmativo, o negociador passa então a esperar um pedido de confirmação da reserva (chamada ao método *confirm*) que é então repassada aos controladores de admissão. Se algum dos controladores de admissão não conseguir reservar os recursos solicitados, o negociador pode tentar uma nova distribuição das responsabilidades, até que, sendo impossível concluir a negociação com sucesso, o gerenciador é informado pelo orquestrador e notifica seus observadores (objetos da classe *IntraMediaOrchestrationListener*).

O acionamento da orquestração intra-mídia faz com que seja também iniciado o monitor intra-mídia (classe *Monitor*) do orquestrador intra-mídia. Esse

monitor permanece observando os sintonizadores dos provedores de serviço (objetos da classe *QoSTuner*) e é notificado de qualquer violação nos contratos de QoS já firmados. Essa notificação se dá através de uma chamada ao método *getStatistics* do monitor intra-mídia. Isso faz com que o monitor acione o sintonizador de QoS intra-mídia (chamada ao método *alert* do objeto da classe *QoSTuner* do orquestrador intra-mídia) que deve tentar uma nova distribuição das responsabilidades de qualidade de serviço entre os provedores. Se isso não for possível, o sintonizador intra-mídia notifica o gerenciador da violação. Note que o gerenciador da orquestração intra-mídia no formatador atua como monitor do seu orquestrador intra-mídia.

Quando informado da violação, o gerenciador pode realizar uma nova tentativa de negociação sem que seja preciso notificar os orquestradores de pré-busca e de apresentação. Caso não tenha sucesso, o orquestrador de pré-busca, ou o orquestrador de apresentação, deve ser informado, para que as correções sejam efetuadas diretamente no respectivo plano, pois, nesse caso, para corrigir o erro na orquestração intra-mídia, a única solução é sintonizar a orquestração inter-mídia.

4.3 Mecanismos de adaptação

Com o intuito de obter uma qualidade próxima da ideal durante a execução de um documento, as técnicas de pré-busca procuram antecipar as ações e previamente preparar o sistema e os dados dos conteúdos dos objetos. Nesse sentido, o principal objetivo da orquestração de pré-busca é ter os eventos de exibição já preparados, para que, quando necessária, a transição para o estado *ocorrendo* seja sempre imediata, ou dentro de limites aceitáveis.

A orquestração intra-mídia aparece como uma segunda funcionalidade, atuando na gerência dos recursos necessários para manter a exibição, ou mesmo a preparação, do conteúdo de cada um dos objetos de execução do documento. Mais uma vez, o objetivo é manter a qualidade da execução dos documentos em níveis aceitáveis, diminuindo a probabilidade de que as especificações do autor não venham a ser respeitadas.

Apesar de serem recursos importantes, apenas os mecanismos de pré-busca e orquestração intra-mídia não são suficientes para garantir a boa qualidade das

apresentações. Sempre pode haver situações que impeçam continuar a exibição do documento de modo satisfatório. No caso da pré-busca, a memória cache pode não ter espaço suficiente para armazenar todos os dados necessários. No caso da orquestração intra-mídia, a rede e o sistema operacional (provedores de serviços de comunicação e processamento) podem não conseguir oferecer, ou garantir, a qualidade de serviço solicitada. Pode ser, inclusive, que algum provedor não ofereça suporte para que a QoS seja negociada, restando apenas como opção um serviço de melhor esforço. Em todos esses casos, para que o formatador recupere a sincronização da apresentação, é necessário que sejam acionadas estratégias de adaptação para, por exemplo, diminuir a duração (ou a taxa de exibição) de alguns objetos, substituir objetos por outros de menores exigências de QoS, ou mesmo descartar alguns objetos.

Conforme já discutido na Seção 2.1, são muitas as motivações para a existência de mecanismos de adaptação em apresentações hipermídia. A adaptação pode ser interessante, não apenas como um mecanismo reativo a situações que possam causar impacto na qualidade da apresentação, como um mecanismo que, antes de iniciar a apresentação, procura adequar o documento às informações do contexto de exibição e às especificações do autor.

De acordo com os conceitos apresentados na Seção 4.2, os mecanismos de adaptação, de um modo geral, podem ser vistos como parte integrante do processo de orquestração inter-mídia da apresentação. Quando utilizada como um mecanismo de reação e ajuste a violações no plano de execução originalmente previsto, a adaptação atua como um auxiliador à estratégia de sintonização, enquanto que, quando aplicada na fase de compilação, a adaptação apóia a estratégia de negociação.

O restante desta seção aborda, de maneira mais detalhada, a questão da adaptação através do ajuste elástico das durações dos objetos. Os aspectos relacionados à seleção de alternativas não serão tratados, pois foram discutidos no Capítulo 2 (Seções 2.1 e 2.3.3). Contudo, no final, a seção trata da interface do adaptador para atuar integrado aos demais componentes do formatador, oferecendo serviços de ajuste dos tempos elásticos e de seleção de alternativas.

4.3.1
Ajuste dos tempos elásticos

Mecanismos de ajuste dos tempos elásticos (*elastic time computation*), termo formalmente definido em (Kim & Song, 1995), procuram adaptar as apresentações dos objetos a fim de garantir a consistência temporal dos documentos. O objetivo é alcançado através de diminuições (*shrinking*) ou aumentos (*stretching*) nas durações ideais dos eventos de apresentação. Evidentemente, para que se possa aplicar esse tipo de mecanismo de adaptação, é preciso que o modelo de execução ofereça flexibilidade na especificação das durações dos eventos.

A especificação da duração, como uma classe extensível no modelo de execução (Seção 2.3.1.3), permite à arquitetura de formatação explorar a definição de durações flexíveis para as ocorrências dos eventos não instantâneos (em particular os de apresentação) e, com isso, aplicar os mecanismos de ajuste elástico. O diagrama de classes da Figura 25 ilustra as extensões já definidas no modelo de execução para a classe *Duration*.

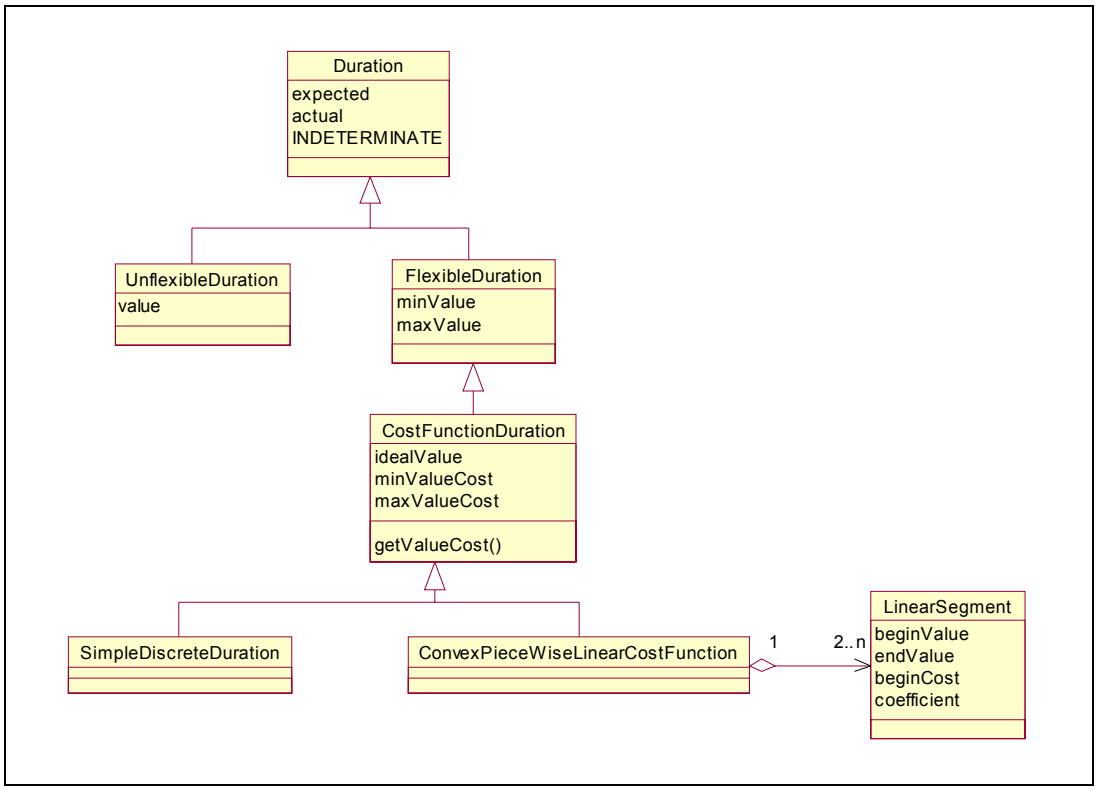

Figura 25 – Classes de duração no modelo de execução.

De acordo com a figura, o modelo define uma duração como podendo ser de dois tipos: inflexível (classe *UnflexibleDuration*), que não permite que ajustes sejam feitos sobre o valor esperado, ou flexível (classe *FlexibleDuration*), que especifica uma faixa de possíveis valores para escolha do valor esperado. São justamente as durações derivadas dessa classe (ou de subclasses dela) que devem ser utilizadas pelos mecanismos de ajuste dos tempos elásticos.

Uma subclasse especial de *FlexibleDuration* é definida para permitir a especificação de durações flexíveis como funções de custo (classe *CostFunctionDuration*). Uma função de custo possui, além dos limites para escolha do valor esperado, a especificação de pelo menos um valor ideal e medidas de custos, que permitem inferir a degradação da qualidade quando se escolhe um valor esperado diferente do ideal. A Figura 26a ilustra graficamente a descrição de uma instância de *CostFunctionDuration* com valores mínimo, ideal e máximo respectivamente iguais a θ_{min}, θ_{id} e θ_{max}. No exemplo, o custo do valor mínimo é dado por c_1 e o custo do valor máximo é dado por c_2. Na realidade, uma função desse tipo define dois custos de degradação: um para durações abaixo da ideal e maiores ou iguais à duração mínima, e outro para durações acima da ideal e menores ou iguais à duração máxima.

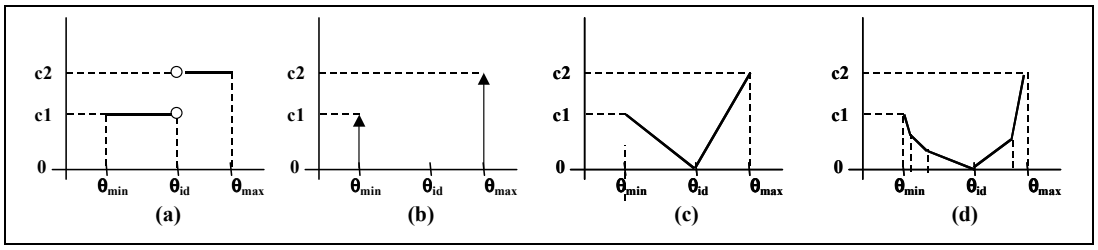

Figura 26 – Exemplos de funções de custo. a) Função de custo simples. b) Função de custo simples, porém discreta. c) Função de custo linear do tipo *piece-wise* com dois segmentos. d) Função de custo linear do tipo *piece-wise* com múltiplos segmentos.

A Figura 26b ilustra um exemplo de instância da classe *SimpleDiscreteDuration*. A diferença é que, nesse caso, apesar de flexível, a duração esperada precisa ser escolhida como sendo uma das três opções (mínima, ideal ou máxima), representando uma função de custo discreta.

O modelo define ainda uma classe *ConvexPieceWiseLinearCostFunction*, para representar funções de custo convexas que sejam compostas de múltiplos segmentos lineares. Cada segmento (classe *LinearSegment*) possui uma duração

inicial e uma duração final, o custo da duração inicial e o coeficiente angular do segmento. Os gráficos da Figura 26c e da Figura 26d são exemplos de funções de custo lineares, *piece-wise* e convexas. Novos tipos de durações, e principalmente de funções de custo, podem ser facilmente incorporados ao formatador através de extensões das classes apresentadas na Figura 25.

Em certos casos, é útil que alguns dos atributos das durações (por exemplo, duração ideal, os limites para a duração etc.) sejam especificados como valores indeterminados. Um nó cujo conteúdo é apresentado até que alguma ação externa interrompa a sua exibição ou um nó que termina por si próprio a sua apresentação são exemplos de objetos de execução, cujos eventos de apresentação possuem durações imprevisíveis. Sendo assim, a classe *Duration* define uma constante *INDETERMINATE*, com um valor negativo, que serve para especificar atributos da duração como sendo indeterminados.

O objetivo de um mecanismo de ajuste dos tempos elásticos é procurar calcular as durações esperadas para os eventos de apresentação dos objetos, respeitando os valores válidos para a duração de cada evento e, ao mesmo tempo, os relacionamentos impostos pelo autor. Contudo, em algumas situações, o mecanismo de ajuste pode encontrar mais de uma solução para o problema colocado. Se as durações forem especificadas como funções de custo, o algoritmo de ajuste pode tentar aplicar uma função objetiva, que calcule as durações esperadas para cada objeto, de tal maneira que o custo total dos ajustes seja minimizado (Bachelet et al., 2000; Buchanan & Zellweger, 1993b; Kim & Song, 1995). No entanto, esse pode não ser o único parâmetro para escolha da melhor configuração das durações para o documento. Outros critérios que o mecanismo de ajuste pode levar em conta são:

- Minimizar o número de objetos com eventos de exibição com duração esperada diferente da duração ideal. A idéia desse critério é diminuir o número de objetos que tenham as suas durações modificadas, assumindo que modificar a duração de um objeto pode impor um custo computacional elevado na plataforma exibição.
- Minimizar a soma dos desvios em relação às durações ideais. Quando houver mais de uma solução para o critério anterior, esse critério pode servir como um complemento. No entanto, se aplicado como primeiro

critério, ele tende a ser contrário ao critério anterior, pois a idéia é distribuir ao máximo as alterações, buscando torná-las o menos perceptível possível.

O mecanismo de ajuste pode também se guiar por restrições externas ao documento, como por exemplo, encontrar o menor custo de ajuste a partir de uma duração total de apresentação que venha a ser definida. Uma aplicação prática desse tipo de restrição seria na programação de TV, onde os tempos de alguns programas ou comerciais podem precisar ser ajustados dinamicamente, devido a atrasos imprevisíveis na grade. Outra restrição externa que pode vir a ser imposta é o número máximo de objetos que podem ter suas durações recalculadas.

É útil também que uma estratégia de ajuste saiba, não apenas encontrar uma solução ótima, como também responder qual seria a menor e a maior duração válida para a exibição do documento, ou de um de seus trechos. Melhor ainda se a estratégia for capaz de manter essas informações atualizadas, conforme transcorre a apresentação do documento, pois pode auxiliar quando o ajuste precisar ser feito em tempo de execução (Seção 3.1).

4.3.2 Arquitetura para implementação de mecanismos de adaptação

O módulo adaptador no formatador (classe *DocumentAdapter*) responde por três funções básicas: avaliar as regras de exibição (Seção 2.3.3), definir o objeto de execução a ser selecionado em um conjunto de alternativas (Seção 2.3.3) e ajustar as durações dos eventos não instantâneos (Seção 2.3.1.3) contidos em uma cadeia temporal que seja passada como parâmetro.

As duas primeiras funções podem ser desempenhadas pela própria implementação do módulo de adaptação, utilizando as informações coletadas do proxy de informações contextuais (classe *ContextManager*). O ajuste dos tempos elásticos, devido a sua maior complexidade e variedade de possibilidades para realizar a tarefa, é modelado como um ponto flexível, através de uma estratégia genérica. O adaptador pode, inclusive, possuir estratégias distintas para o ajuste em tempo de compilação e o ajuste em tempo de execução. O método de ajuste dos tempos (*adjustTimeChain*) recebe como parâmetro, além da cadeia temporal,

um atributo que informa se o cálculo deve ser realizado com a estratégia de compilação (atributo *compileTimeStrategy*) ou com a estratégia de execução (atributo *executionTimeStrategy*) do adaptador. Além desses atributos, opcionalmente, o método de ajuste pode ser informado sobre a restrição de duração imposta pelo autor (ou pelo usuário) e o número máximo de objetos que podem ter suas durações modificadas. O método *adjustTimeChain* retorna um valor booleano dizendo se foi possível, ou não, encontrar uma configuração consistente.

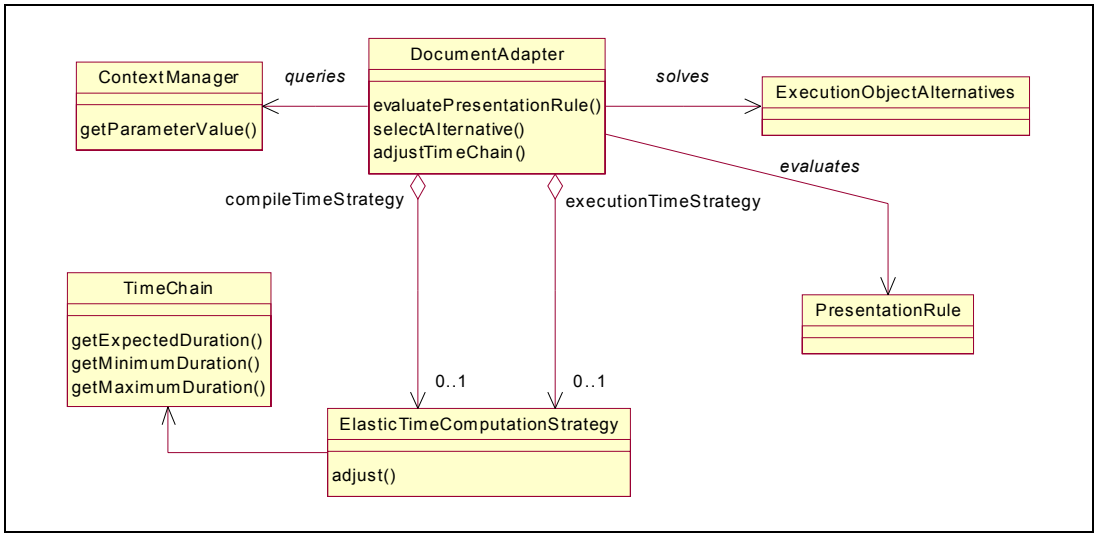

Figura 27 – Diagrama de classes para implementação de mecanismos de adaptação.

Uma vez executada e conseguindo encontrar uma solução para a cadeia temporal, a estratégia de ajuste deve gravar na cadeia a duração programada para sua exibição completa. Se possível, a estratégia deve especificar também os limites mínimo e máximo para a duração da cadeia. Esses valores podem, posteriormente, ser consultados através de chamadas aos métodos *getExpectedDuration*, *getMinimumDuration* e *getMaximumDuration* da própria cadeia temporal. Da mesma forma, os eventos não instantâneos no modelo possuem dois atributos adicionais que podem guardar a duração mínima e máxima para sua ocorrência que não infringe a consistência temporal do documento. Esses valores, se possível, são calculados pelas estratégias de ajuste e podem se modificar ao longo da exibição do documento.