6
Trabalhos Relacionados

Este capítulo busca oferecer uma visão das pesquisas relacionadas ao tema desta tese e compará-las com as proposições que foram colocadas ao longo deste trabalho. Com o objetivo de organizar as comparações, o capítulo as divide em quatro seções. A primeira seção comenta os trabalhos que tratam da questão da estruturação dos formatadores e orquestração das apresentações de uma maneira geral. As outras três seções tratam das características mais específicas do projeto de ambientes de execução: a segunda seção comenta os esforços para a integração de ferramentas de exibição em ambientes de apresentação hipermídia; a terceira seção apresenta uma comparação com os trabalhos que focam nos mecanismos de pré-busca para exibição de hiperdocumentos; e, por fim, a quarta e última seção do capítulo comenta a respeito dos trabalhos que oferecem suporte à adaptação das apresentações, com interesse maior no ajuste dos tempos elásticos.

6.1
Formatadores multimídia/hipermídia

Para praticamente todos os modelos, linguagens e sistemas citados no princípio do Capítulo 2 (Seção 2.1.1.1), existe pelo menos um protótipo de formatador implementado. No entanto, como são diversas propostas na literatura, esta seção restringiu os comentários e comparações apenas às propostas cujas idéias contribuíram diretamente para a constituição da arquitetura definida nesta tese.

O sistema Firefly (Buchanan & Zellweger, 1992; Buchanan & Zellweger, 1993a; Buchanan & Zellweger, 1993b) foi desenvolvido com o objetivo de fornecer facilidades para criação, edição, manutenção e apresentação de documentos multimídia/hipermídia. Pioneiro na definição do termo *formatador* como denominação ao controlador das apresentações, o sistema trouxe várias contribuições para esta tese.

O sistema é baseado em um modelo constituído por objetos de execução (*media items*), compostos por um conteúdo, um conjunto de procedimentos e uma lista de eventos. Os eventos podem ser síncronos ou assíncronos e são sempre instantâneos, correspondendo, na maioria das vezes, às transições do modelo de execução definido no Capítulo 2. Para cada par de eventos adjacentes, o sistema permite definir durações como funções de custo lineares do tipo *piece-wise*, com apenas dois segmentos, similar à função ilustrada na Figura 26c. Os relacionamentos no modelo são definidos como restrições temporais entre eventos, podendo ser especificados como equações ou inequações. Além disso, cada evento pode ter a ele associada uma lista de operações que permitem controlar os aspectos de apresentação do objeto de execução.

A arquitetura do formatador Firefly separa o controle da apresentação em duas fases, uma de compilação e outra de execução e estabelece a noção de cadeias temporais, sendo uma principal (*main schedule*) e as outras auxiliares (*auxiliary schedules*). Cada cadeia relaciona todos os eventos síncronos associados por alguma restrição temporal, ou pertencentes a um mesmo objeto de execução que tenha tido algum evento inserido na cadeia. Conforme já comentado, essa proposta exerceu influência sobre a definição da arquitetura de formatação e dos planos de execução descritos neste trabalho.

O compilador, com base nas restrições temporais e nas funções de custo das durações, calcula os tempos esperados para ocorrência de cada evento. Isso é feito para cada cadeia de maneira independente.

Uma vez iniciada a apresentação, o módulo de execução inicia um escalonador e um tratador de eventos. O escalonador monitora a cadeia principal e, com base em um relógio de execução, envia comandos para as ferramentas de exibição (*media managers*), quando os tempos programados pelo compilador são alcançados. O tratador de eventos observa as sinalizações dos eventos informados pelas ferramentas e, quando percebe que uma cadeia auxiliar deve ser iniciada, requisita ao escalonador a sua junção com a cadeia principal. O escalonador também realiza, baseado nas listas de operações, um controle de ativação e desativação dos eventos, mantendo o tratador informado para que não seja solicitada uma junção a partir de eventos que não estejam ativos no momento.

Como pode ser percebido, uma vez iniciada a apresentação do documento, o plano de execução torna-se um *timeline* das ações a serem desempenhadas. A arquitetura simples do formatador não dispõe de componentes para ajustar a exibição em função de variações na banda passante, por exemplo. O modelo de documentos também não prevê adaptações diferentes do ajuste elástico dos tempos, tais como seleção de alternativas. Módulos para pré-busca dos conteúdos também não são considerados como fazendo parte do formatador.

Madeus (Jourdan et al., 1998a; Jourdan et al., 1998b; Sabry-Ismail et al., 1999) é um sistema para autoria e apresentação de documentos multimídia com interatividade, baseado em um modelo de restrições temporais e espaciais entre os objetos de execução (*media objects*).

As restrições espaciais utilizam os operadores *alinhar*, *centralizar* e *deslocar* tanto em relação ao eixo horizontal como o eixo vertical, enquanto as restrições temporais baseiam-se nas relações de Allen (Allen, 1983). O modelo permite que os objetos sejam sincronizados simultaneamente no tempo e no espaço. O modelo também oferece hiper-elos para a especificação de relacionamentos de navegação tradicionais entre os objetos.

Todo objeto tem a ele associada uma duração flexível, que informa limites inferior e superior para a sua apresentação. O ambiente de execução do sistema Madeus possui um formatador (Sabry-Ismail et al., 1999) que calcula, em uma fase de compilação, o valor esperado para a duração de cada objeto, denominada *duração nominal*. O modelo define como *objetos incontroláveis* aqueles que não podem ter a sua duração esperada conhecida até que sua exibição termine. De maneira oposta, os objetos que possuem uma duração nominal calculada pelo compilador são classificados como *controláveis*.

A compilação e o controle da execução do documento são feitos a partir da construção de um grafo temporal, denominado HTSP (*Hypergraph of Simple Temporal Problems*), que funciona como o plano de execução do formatador Madeus (Jourdan et al., 1998a).

Na fase de execução, o formatador observa os tempos efetivos das durações e os compara com as durações programadas. Quando alguma variação que cause uma perda de sincronização na apresentação é identificada, o formatador aplica um algoritmo de ajuste. O executor Madeus também observa os instantes em que

os objetos incontroláveis terminam de ser exibidos para verificar se isso resulta na necessidade de algum ajuste das durações.

O formatador Madeus não aborda as questões relacionadas à orquestração da pré-busca. O mecanismo de adaptação do formatador permite recuperar o sincronismo da apresentação sempre que percebe alguma divergência com o plano programado, no entanto, esse ajuste é feito buscando restabelecer o sincronismo o mais rapidamente possível, sem que seja utilizada qualquer medida de qualidade. Outras técnicas de adaptação orientada ao contexto, como seleção de alternativas, não são comentadas nas referências citadas.

O formatador CMIFed, denominado *CMIFed player*, é o elemento responsável pelo controle da apresentação de documentos CMIF (*CWI Multimedia Interchange Format*) (van Rossum et al., 1993; Hardman et al., 1993b). As entidades presentes nesse modelo são praticamente as mesmas da linguagem SMIL (W3C, 1998; W3C, 2001a), e o seu formatador deu origem a um dos principais formatadores SMIL (ou *player* SMIL): o sistema GRiNS da Oratrix (Oratrix, 2002). Além do GRiNS, outro exemplo de formatador implementado para a linguagem SMIL versão 2.0 é a plataforma RealOne, da RealNetworks. Existe ainda um suporte oferecido pelo browser Internet Explorer 6.0 à proposta de padrão XHTML+SMIL Profile (W3C, 2001b), também baseada na linguagem SMIL. A comparação nesta seção será feita com relação às funcionalidades do formatador CMIFed.

Quando um usuário requisita a exibição de um documento CMIF, o formatador entra em uma fase de compilação, onde um grafo de dependências temporais é construído. No grafo, os nós representam transições em estados dos eventos de apresentação (pontos de sincronização) e as arestas são relações temporais entre os eventos. Inicialmente, o grafo é construído percorrendo a hierarquia do documento em profundidade, utilizando apenas as restrições temporais especificadas pelas composições paralelo e seqüencial oferecidas pelo modelo de documentos (van Rossum et al. 1993; W3C, 1998; W3C, 2001a). Em um passo seguinte, arestas representando os relacionamentos especificados por sync arcs (do modelo CMIF) são acrescidas. Quando um arco de sincronismo é definido utilizando uma marcação que não seja o início ou fim de um evento, um novo nó e novas arestas ligando a marcação a outras marcações do mesmo evento

são acrescidos ao grafo. Finalmente, dois nós especiais são definidos no plano de execução (grafo), o *start node* e o *end node*. O *start node* é conectado, através de uma aresta com duração nula, ao nó representando o início do primeiro evento a ser exibido no documento. Da mesma forma, uma aresta também com duração nula liga o nó que define o fim do último evento ao *end node*.

Construído o grafo, o formatador entra na fase de execução. Um relógio interno é disparado e a execução feita utilizando um algoritmo que percorre o grafo criado. Inicialmente, o *start node* é marcado. Quando um nó é marcado, todas as arestas que partem desse nó recebem o valor do relógio de execução no momento da marcação. Quando a duração especificada na aresta é completada, a aresta é marcada. Quando todas as arestas que chegam em um nó são marcadas, o nó é marcado. A execução termina quando o *end node* é marcado.

Ao simular uma execução e percorrer o grafo de dependências temporais, o sistema é capaz de descobrir erros na especificação temporal, através da detecção de caminhos fechados, e também conflitos por recursos. O grafo também permite que o formatador faça pré-busca do conteúdo dos objetos. O formatador CMIF realiza uma pré-busca simplificada, utilizando os momentos em que o formatador encontra-se ocioso. A utilização de heurísticas de busca para eventos gerados por pontos de sincronização imprevisíveis, bem como negociações de QoS para busca dos conteúdos não são abordados na implementação. O sistema também não aborda a questão dos ajustes das durações, trabalhando com tempos inflexíveis nas arestas dos arcos. Na versão mais recente da linguagem SMIL (versão 2.0), essa limitação foi eliminada com a introdução de um intervalo para especificação das durações. No entanto, da mesma forma que no sistema Madeus, essa flexibilidade não carrega consigo métricas para direcionar a qualidade do ajuste.

6.2
Integração entre formatadores e ferramentas de exibição

Os principais browsers WWW possuem uma API (*Application Program Interface*) para programação de plug-ins, originalmente proposta pela Netscape (Netscape, 1998), que permite que aplicações externas sejam incorporadas ao ambiente de navegação da Web, principalmente para que formatos não reconhecidos pelo browser sejam tratados em um ambiente único. Porém, plug-ins

não são ferramentas de exibição hipermídia e sim apenas ferramentas de exibição, pois a sua API não oferece suporte para definição de relacionamentos entre objetos apresentados em diferentes plug-ins, e até mesmo entre plug-ins e páginas HTML. Na realidade, existem alguns mecanismos, como LiveConnect e ActiveX, que permitem a integração de páginas HTML com plug-ins, e até mesmo com outras tecnologias como scripts e applets. No entanto, esses mecanismos são dependentes do browser utilizado (Netscape ou Internet Explorer) e baseados em códigos de programação não triviais que devem ser embutidos nos próprios objetos de mídia. Uma outra limitação da tecnologia é o fato do plug-in ter seu ciclo de vida amarrado ao ciclo de vida da página HTML que o contém.

Bouvin e Schade (Bouvin & Schade, 1999) propõem extensões à API de plug-ins com o principal objetivo de permitir a criação de elos entre objetos e sub-regiões desses objetos, independente da ferramenta (plug-in) que os exibe. Uma vez que não era possível alterar a API utilizada pelos browsers existentes, os autores implementaram um sistema, denominado Mimicry, utilizando applets Java para emular os relacionamentos. Um proxy específico do sistema, chamado *DHMProxy*, encarrega-se de substituir as marcações de plug-ins das páginas (*EMBED* e *OBJECT*) por referências aos applets do sistema Mimicry. No entanto, os elos que o sistema permite criar são sempre elos de interatividade, não havendo suporte para a definição de relacionamentos de sincronização temporal entre os objetos. Além disso, os autores não abordam os aspectos relacionados ao desenvolvimento das ferramentas e suas integrações com o formatador hipermídia, como foi feito com o framework apresentado no Capítulo 3 desta tese.

O modelo de execução do sistema Madeus (Jourdan et al., 1998a) define os objetos de execução como sendo a própria ferramenta de exibição, adotando um mecanismo de plugin para permitir a integração com objetos que devam ser apresentados por aplicações externas. No entanto, a interface para comunicação é uma simplificação da interface padrão para os objetos de execução que o ambiente de execução sabe controlar. Essa simplificação obriga que os objetos de execução controlados por plugins sejam tratados como objetos com durações imprevisíveis (*objetos incontroláveis*), dificultando a manutenção das relações de restrição temporal que o Madeus utiliza.

Diferente dos trabalhos anteriormente mencionados, esta tese propôs um modelo de interface, genérico e adaptável, para integração de ferramentas de exibição e formatadores hipermídia, que permite tratar igualmente todos os objetos sendo exibidos, independente da aplicação exibidora que os controla. O framework de integração das ferramentas permite que novas ferramentas de exibição venham a ser incorporadas, e outras tantas existentes possam ser adaptadas, funcionando de maneira integrada entre si e com o formatador hipermídia. A construção de adaptadores para ferramentas que utilizam o JMF foi apenas um exemplo de como o framework apresentado pode ser aplicado. No momento, encontram-se, também em desenvolvimento, especializações para integrar o formatador HyperProp a exibidores QuickTime e RealOne.

6.3
Mecanismos de pré-busca

Jeong et al. (Jeong et al., 1997) desenvolveram um mecanismo para efetuar o pré-escalonamento de apresentações multimídia. O módulo de pré-busca, denominado EPS (*Event Pre-Scheduler*), estima os tempos máximos para recuperação de cada um dos objetos por inteiro e constrói um plano de pré-busca para que todos os objetos estejam prontos antes de suas exibições. O algoritmo para construção do plano de pré-busca retarda o início da apresentação do documento como um todo, para evitar que durante a exibição ocorram "*gaps*" e perdas de sincronização. A estrutura de dados do plano de execução, utilizada como entrada para a construção do plano de pré-busca, baseia-se na divisão dos objetos de mídia em segmentos mínimos que satisfazem, entre si, a condição de igualdade (*equals*) de Allen (Allen, 1983). Durante a execução, existe um monitor que compara o tempo de pré-busca de cada objeto com o tempo previsto no plano. Sempre que esse tempo ultrapassa o limite previsto, um procedimento para recalcular as durações de apresentação dos objetos é disparado a fim de manter a sincronização dos segmentos. Quando necessário, o algoritmo sacrifica as mídias estáticas, reduzindo suas durações. O aspecto interessante desse algoritmo é que ele serve também para corrigir atrasos que tenham sido causados não apenas pelo sistema operacional, mas pelo próprio processamento do mecanismo de pré-busca.

O EPS é um trabalho interessante porque apresenta, de uma maneira prática, vários dos requisitos que devem ser considerados na elaboração de uma estratégia de pré-busca. No entanto, a proposta considera apenas objetos com exibições previsíveis e todos os conteúdos devem estar armazenados localmente (em disco) e de maneira contígua, não sendo abordados aspectos referentes a transmissões na rede. Além disso, o mecanismo funciona sempre carregando o conteúdo inteiro dos objetos em memória antes das exibições, não existindo a possibilidade da ferramenta de exibição consumir os dados em paralelo com a busca do conteúdo. A proposta também limita o reajuste, em tempo de execução, apenas do plano de execução do documento, não sendo previsto pela estratégia a possibilidade de realizar correções no próprio plano de pré-busca.

Khan e Tao (Khan & Tao, 2001a; Khan & Tao, 2001b) estabelecem um modelo que permite determinar a melhor seqüência de pré-busca para páginas compostas na Web. Páginas compostas são aquelas que incluem, além do tradicional texto HTML, referências a outros objetos como imagens, applets, animações, vídeo, áudio etc. Essas páginas compostas são denominadas composições hipermídia. Os autores definem uma classificação em três perfis para os conteúdos dos elementos de uma composição, considerando os aspectos de transmissão na rede: objetos de taxa constante (mídia CBR: áudio, vídeo etc.), objetos de uma única rajada (mídia impulsiva: página HTML, imagens estáticas, applets JAVA etc.) e objetos de múltiplas rajadas (mídia impulsiva em série: documentos PDF de várias páginas, animações, apresentações em slides etc.). Um modelo matemático permite calcular, para cada composição hipermídia, a quantidade mínima de dados que deve ser trazida antes de sua exibição. Juntamente com a informação de probabilidade de navegação em cada elo, o trabalho demonstra qual é a melhor escolha para a construção do plano de escalonamento, ou seja, qual a melhor ordem de busca para as páginas compostas. Um algoritmo divide a largura de banda da rede para que as requisições de pré-busca convivam com as requisições dos conteúdos sendo efetivamente apresentados, sem que o desempenho dessas últimas requisições seja prejudicado. O trabalho, no entanto, não contempla, a possibilidade de existirem relacionamentos de sincronização temporal entre as mídias, na construção do plano de pré-busca.

É importante observar que tanto o EPS como a proposta de Khan e Tao poderiam ser implementados como instanciações do framework de pré-busca definido nesta tese. As propostas poderiam estar presentes principalmente nas implementações das estratégias de construção dos planos de pré-busca e de estimativa do tempo de busca.

Em (Revel et al., 1996) foi desenvolvida a implementação de um módulo de pré-busca, denominado *Prefetcher*, para copiar antecipadamente, do disco para a memória principal, o conteúdo das aplicações multimídia, em especial vídeos MPEG. O *Prefetcher* é implementado como um módulo integrado ao sistema operacional, oferecendo uma interface para que as aplicações multimídia possam descrever o padrão de acesso aos dados (tamanho dos quadros, taxa em que os quadros serão consumidos, padrão de repetição dos quadros I, P e B etc.) e o instante de tempo em que os dados serão necessários. A proposta permite que as aplicações multimídia, após agendarem a apresentação, utilizem chamadas tradicionais de entrada e saída para ter acesso aos dados. Internamente, o *Prefetcher* cuida de enviar chamadas assíncronas ao sistema operacional, que resultam numa pré-busca dos dados pelo próprio sistema operacional. Contudo, a interface oferecida é bastante limitada, não prevendo chamadas para aplicações que possuam uma quantidade mais variada de tipos de conteúdo e que permitam exibições imprevisíveis. Além disso, o estudo da extensão da arquitetura para ambientes distribuídos em rede é deixado como trabalho futuro.

Diferente da proposta desta tese, nenhum dos trabalhos comentados nesta seção aborda a questão de integrar o mecanismo de pré-busca a funções de provisão e gerência de QoS. Um outro ponto identificado é, com exceção do EPS, as outras duas propostas não mencionam a possibilidade de que os resultados das pré-buscas realimentem os planos de execução dos documentos.

6.4 Mecanismos de adaptação

Propostas para computação dos tempos elásticos em apresentações multimídia/hipermídia têm sido endereçadas na literatura (Buchanan & Zellweger, 1993a; Kim & Song, 1995; Sabry-Ismail et al., 1999).

O algoritmo de ajuste do formatador Firefly (Buchanan & Zellweger, 1993a) realiza o cálculo das durações ótimas através de programação linear e fazendo uso do método simplex. No modelo de documentos utilizado pelo sistema, as durações são definidas como funções de custo lineares, similares à exemplificada na Figura 26c.

Abordagem semelhante é utilizada pelo formatador do sistema Isis (Kim & Song, 1995). No entanto, o cálculo de otimização das durações acrescenta a noção de justiça na distribuição dos ajustes pelos objetos de mídia, procurando soluções que minimizem o quanto cada objeto tem sua duração modificada, em troca de uma maior quantidade de objetos tendo seus tempos de apresentação ajustados. O mecanismo de ajuste também permite que o usuário imponha uma restrição de duração global para uma cadeia temporal e é capaz de retornar não apenas a solução ótima como também os limites de tempo mínimo e máximo possíveis para a duração da apresentação da cadeia.

Tanto o formatador Firefly como o formatador Isis realizam o ajuste dos tempos elásticos apenas na fase de compilação do documento. Além disso, a utilização do método simplex de maneira eficiente para o cálculo da solução, normalmente requer o uso de bibliotecas que necessitam de uma grande quantidade de memória e disco, o que dificulta sua integração à aplicação. Outro ponto a ser considerado é que pode ser que o aumento do número de objetos que têm suas durações modificadas não seja uma boa estratégia, podendo ser mais interessante em termos de esforço computacional, manter os ajustes em uma quantidade mínima de elementos.

Diferente das abordagens anteriores, a ferramenta de ajuste utilizada pelo formatador HyperProp não é baseada em programação linear, mas sim na utilização de algoritmos para distribuir, com um custo mínimo, tensões e fluxos em grafos de tensões (Bachelet et al., 2000). Essa abordagem faz com que as estruturas de dados e bibliotecas necessárias para o cálculo sejam bem inferiores às normalmente exigidas pela programação linear, com o custo de uma eficiência aproximadamente duas vezes pior (Bachelet et al., 2000).

Apesar do pior desempenho, a utilização de grafos de tensão para modelar o problema de ajuste dos tempos elástico trouxe uma nova visão e novas perspectivas de algoritmos para solução do problema. Os métodos *out-of-kilter*

(Fulkerson, 1961) e *dual cost scaling* (Ahuja et al., 2000) são atualmente suportados pela ferramenta utilizada no formatador HyperProp. Além disso, o uso de novos métodos para os documentos com topologia do tipo série-paralelo (Duffin, 1965) estão sendo investigados. Acredita-se que essa abordagem viabilize a aplicação dos ajustes durante a execução dos documentos. Para isso, é preciso que a estratégia de ajuste seja capaz de identificar nas cadeias temporais estruturas que respeitem essa topologia.

O formatador do sistema Madeus (Sabry-Ismail et al., 1999) descreve uma solução para ajuste em tempo de execução (*on-the-fly*) Uma vez que as durações dos objetos são flexíveis, mas não dispõem de métricas de custo de alteração, a estratégia de ajuste utiliza uma abordagem diferente. Quando o formatador percebe uma violação nos tempos programados, o algoritmo de correção é acionado para tentar encontrar uma nova configuração que reestabeleça os tempos originalmente programados o mais rapidamente possível. A idéia da estratégia é minimizar o período que o documento se encontra fora do sincronismo idealmente imposto pelo autor, respeitando os limites para duração de cada objeto.

A característica de adaptação através de seleção de alternativas é possível de ser encontrada nos formatadores SMIL, explorando o uso do elemento *switch* da linguagem (W3C, 1998; W3C, 2001a), no formatador CMIFed, explorando o conceito de canais e a possibilidade de torná-los habilitados ou não (Hardman et al., 1993a; Hardman et al., 1993b; van Rossum et al., 1993) e na aplicação Cardio-OP (Klas et al., 1999), explorando os elementos de adaptação *switch* e *query* do modelo de documentos Z_yX (Boll & Klas, 1999).