Autoria e Formatação Estruturada de Documentos Hipermídia com Restrições Temporais

Luiz Fernando Gomes Soares
Depto. de Informática, PUC-Rio
R. Marquês de São Vicente 225
22453-900 - Rio de Janeiro, Brasil
lfgs@inf.puc-rio.br

Rogério Ferreira Rodrigues
Depto. de Informática, PUC-Rio
R. Marquês de São Vicente 225
22453-900 - Rio de Janeiro, Brasil
rogerio@telemidia.puc-rio.br

Resumo

Este artigo realiza uma análise crítica dos ambientes para especificação e formatação de documentos hipermídia com restrições temporais, apresentando ao longo de suas seções as soluções adotadas no ambiente de autoria e execução do sistema HyperProp.

Abstract

This paper presents a critical analysis of specification and presentation environments for hypermedia documents with time constraints. Throughout of its sessions, it is also presented the solutions implemented in the HyperProp authoring and execution environment.

Palavras-Chaves: Autoria, Estruturação de Documentos, Formatação Temporal, Hipermídia.

1. Introdução

Um sistema para apresentação de documentos multimídia com restrições temporais deve satisfazer no mínimo a três requisitos:

- dar suporte a diferentes segmentos de mídia;
- permitir representar explicitamente as relações temporais e espaciais previsíveis (a definição precisa de relações previsíveis é dada na Seção 2) entre os vários segmentos de mídia; e
- possuir um algoritmo de formatação (exibição) temporal e espacial versátil.

Um sistema para apresentação de documentos hipermídia deve possuir a mais, no mínimo a capacidade de representar relações entre segmentos de mídia que são disparadas sob o controle interativo do usuário (relação temporal imprevisível).

Adicionalmente, sistemas para apresentação de documentos multimídia/hipermídia podem introduzir várias facilidades desejáveis, tais como:

- suportar para cada segmento de mídia um conjunto extenso de capacidades, como: variabilidade de duração de apresentação, diferentes alternativas de exibição (p. ex., um texto ser sintetizado como áudio), mudanças de comportamento durante a exibição, etc;
- permitir a definição da qualidade de serviço (QoS) exigida pelo segmento de mídia (p. ex., variação de retardo intra-mídia (jitter), banda passante necessária, etc);
- permitir a definição do ambiente de apresentação, como retardo na rede, tipos de mídia que suporta, etc;
- permitir representar, explicitamente, relações temporais com tempos não determinísticos (p. ex.: exiba o áudio, que pode durar entre 20 e 30 seg., paralelo ao vídeo, que pode durar entre 25 e 28 seg., e 10 seg. depois, com uma tolerância de 2 seg., exiba o texto);

- permitir representar, explicitamente, relações espaciais entre segmentos de mídia, em um dado instante de tempo;
- permitir uma definição estruturada (hierárquica ou não) do documento;
- possuir um algoritmo de formatação temporal versátil, que leve em conta os não determinismos e permita a correção da apresentação no caso de ocorrência de eventos não previsíveis (p. ex.: atraso na rede, diretiva do usuário alterando a taxa de apresentação de um documento, etc.); e
- possuir um algoritmo de formatação que tire proveito das definições das QoS exigidas pelo segmento e das características do ambiente de exibição, por exemplo, realização de pré-busca, etc.

O problema da apresentação de documentos (a partir de agora leia-se sempre *documentos multimídia/hipermídia* quando sintetizado pelo termo documento) tem sido endereçado na literatura em três níveis distintos: armazenamento, especificação (autoria) e execução (formatação). Este artigo se detem na análise dos dois últimos. O nível de especificação define os requisitos das aplicações multimídia e as restrições associadas. A execução refere-se ao desenvolvimento de esquemas e protocolos para exibição, garantindo a sincronização inter e intra mídia, bem como a sincronização espacial.

Este artigo realiza uma análise crítica dos ambientes para especificação e formatação de documentos com restrições temporais, apresentando, em suas seções, as soluções adotadas no ambiente de autoria e execução do sistema HyperProp [Soares et al. 1995] e em sistemas relacionados. A sistemática apresentada, as discussões realizadas, bem como a exemplificação através da apresentação do HyperProp são as principais contribuições do trabalho. A Seção 2 dedica-se aos problemas de especificação de documentos, enquanto que a Seção 3 aborda os problemas de execução. A Seção 4 é reservada às conclusões.

2. Projeto de Documentos com Restrições Temporais: Abordagens e Problemas

A maioria dos sistemas atuais para apresentação de documentos estão no estágio de pré-formatação. Eles, praticamente, exigem que o autor crie os layouts temporais manualmente, posicionando os segmentos de mídia em uma linha de tempo do documento, constituindo-se em um processo tedioso e sujeito a erros. Outros métodos manuais incluem as linguagens de scripts temporais e árvores de relações temporais. Já vem de algum tempo a necessidade da geração automática dos layouts temporais, que só há alguns anos começou a ter soluções parciais em protótipos de laboratório.

Antes de se entrar na discussão desta seção, torna-se necessário definir alguns termos que serão utilizados no decorrer de todo o artigo. No contexto deste trabalho, um *evento* pode ser a exibição, *evento de exibição*, ou a seleção, *evento de seleção*, de um conjunto não vazio de unidades de informação, ou ainda a mudança de um atributo de um *objeto componente* (no texto, sinônimo de segmento de mídia) do documento, por exemplo seu estado, chamado *evento de atribuição*. A noção exata do que se constitui uma unidade de informação é parte da definição do tipo de mídia do objeto componente. Por exemplo, uma unidade de informação de um objeto vídeo pode ser um quadro, enquanto uma unidade de informação de um objeto texto pode ser um caracter, ou uma palavra. O início ou fim de um evento é instantâneo e é denominado *ponto de sincronização*.

Um ambiente de autoria deve oferecer ferramentas que permitam a criação dos documentos, através da edição de seus objetos componentes, isto é, de seus conteúdos, e da especificação de sua granularidade, que define quais conjuntos de suas unidades de informação podem ser usados na definição de eventos. Diz-se que a granularidade é

grossa, quando o menor conjunto é todo o segmento de mídia. A criação (definição) do conteúdo dos componentes e a definição de sua granularidade fogem do escopo deste trabalho. Para efeito de discussão no texto, são chamados *objetos de dados*, os objetos que contêm essas definições. Normalmente a criação de tais objetos é responsabilidade dos editores específicos da mídia. Um ambiente de autoria deve, no entanto, também permitir a descrição do comportamento esperado de cada componente, quando for exibido, e a especificação dos relacionamentos entre os componentes. Esses pontos são o assunto do restante desta seção.

2.1. Especificação do Comportamento Temporal de Objetos Multimídia

Em um sistema de autoria deve ser possível especificar, para cada evento, a sua duração e a duração da preparação para sua ocorrência. Esta duração pode ser *previsível*, quando é possível determinar um intervalo finito para a ocorrência, ou não. Analogamente, um ponto de sincronização é dito previsível quando é possível determinar um intervalo finito para sua ocorrência. Dentro do intervalo, a duração de um evento pode assumir valores discretos ou contínuos, devendo ser possível a especificação do valor ótimo, isto é, aquele que apresenta a melhor qualidade de apresentação. Um evento cujo início do intervalo de duração é igual ao fim do intervalo é dito *não ajustável*. Outra capacidade a ser suportada é a possibilidade de adição de medidas objetivas de QoS para um segmento de mídia particular, dos custos para sua geração, seu transporte e exibição, de forma a permitir a seleção de uma melhor apresentação, em uma dada situação.

Todas as especificações definidas no parágrafo anterior devem ser, preferencialmente, realizadas independente do objeto de dados relativo ao componente do documento. No contexto deste trabalho, chama-se *descritor* o objeto que contém todas essas especificações.

A independência entre o descritor e o objeto de dados vai permitir um melhor reuso dos objetos do NCM. Por exemplo, a partir de descritores distintos, pode-se definir diferentes apresentações para um mesmo objeto de dados. Um segmento de dados de texto, por exemplo, pode ser exibido como texto, utilizando um descritor D1, como também pode ser sintetizado como áudio, através da associação de um outro descritor D2. Como descritores diferentes podem levar a durações de eventos diferentes, bem como QoS e requisitos de plataforma diferentes, essas definições não devem fazer parte do objeto de dados e sim do descritor. O agregado do objeto de dados e um descritor para a apresentação de um componente é denominado de *objeto de representação.*

Um descritor tem também como função definir as mudanças de comportamento que podem acontecer quando da ocorrência de eventos, durante a exibição de um objeto de representação. As mudanças de comportamento podem afetar apenas a disposição espacial do componente, por exemplo, zoom ou reposicionamento de uma janela de vídeo, aumento no volume de um áudio, etc. As mudanças podem, no entanto, afetar o comportamento temporal do documento, quando devem ser, então, reportadas ao formatador temporal, como será visto na Seção 3.

Cabe ainda ao descritor especificar o sincronismo espacial de uma apresentação. A noção de sincronização espacial depende da mídia e baseia-se em operadores que definem como combinar objetos para apresentação em um dispositivo de saída, em um dado instante de tempo. Por exemplo, uma sincronização espacial de objetos do tipo imagem, gráfico ou texto define como estes objetos devem ser dispostos para apresentação em uma janela da aplicação. No caso de objetos do tipo áudio, sincronização espacial envolve, por exemplo, mixagem no dispositivo de saída de som, com ajuste de ganho e tom.

A sincronização espacial pode, opcionalmente, ser definida nos relacionamentos responsáveis pela sincronização temporal. O relacionamento pode, por exemplo, não só especificar o momento que um dado evento deve ocorrer, mas também como deve ser sua exibição dentro do contexto de apresentação corrente. O caso mais geral, entretanto, é que o relacionamento seja apenas temporal e, quando da ocorrência do evento, seu descritor diga como será exibido. É assim que acontece nos sistemas HyperProp e Firefly [Buchanan & Zellweger 1992], onde as alterações de comportamento e o sincronismo espacial são manejados através de listas de operações associadas aos objetos de dados (no HyperProp essas listas fazem parte do objeto descritor).

Alguns sistemas permitem uma especificação conjunta do sincronismo espacial, definindo um outro objeto que pode ser compartilhado por vários objetos de representação. Neste caso, o descritor não realiza as funções de sincronização espacial. Tal é o caso do CMIF [van Rossum et al. 1993], onde um descritor de evento é atribuído a um canal, que é uma abstração de um conjunto de propriedades compartilhadas por outros eventos do mesmo tipo de mídia.

Todos os sub-sistemas de autoria discutidos neste trabalho, Firefly, CMIF, I-HTSPN [Willrich et al. 1996] e HyperProp tem o objeto descritor separado do objeto de dados e, também em todos, os objetos de dados fazem apenas referência ao seu conteúdo, que é mantido em outro objeto.

2.2 . Relacionamentos de Sincronização Temporal

Em todo este artigo, o termo sincronização temporal refere-se à sincronização inter-mídia (entre segmentos diferentes de mídia). A sincronização intra-mídia foge ao escopo do trabalho. Sistemas comerciais que endereçam a sincronização temporal ou são baseados em timeline (MAEstro, MacroMind Director, etc.) ou script (ToolBook, etc.).

A representação da sincronização utilizando timeline ou scripts apresenta uma série de problemas. Em ambas não é possível utilizar modularidade em uma apresentação. Quando se deseja reusar parte de uma apresentação, não é claro onde uma cópia deve começar ou terminar. Outra restrição é a dificuldade de se especificar requisitos temporais entre eventos cuja duração é variável, ou desconhecida, até o momento da execução. Assim como a programação, especificação por script é útil para apresentações pequenas, mas a manipulação e edição de grandes documentos torna-se difícil sem a definição de uma estruturação. Além disso, a especificação detalhada de atividades paralelas apresenta os mesmos problemas da maioria das linguagens de programação. A manutenção de uma apresentação em timeline é extremamente problemática. Uma mudança no posicionamento de um objeto pode representar o recálculo de toda a linha do tempo.

Sincronização baseada em restrições (constraint-based), ou em eventos, não possui as limitações descritas acima. Os sistemas citados na seção anterior (Firefly, I-HTSPN, CMIF e HyperProp) utilizam modelos baseados em restrições. Existem vários modelos para especificação temporal baseada em restrições. Este artigo aborda dois deles em mais detalhes, modelos baseados em extensões de redes de Petri e modelos orientados a objetos, isto é, modelos conceituais seguindo o paradigma de orientação a objetos.

Quatro são os atributos primários de uma relação temporal entre eventos [Buchanan & Zellweger 1993]. Granularidade, tipo de relação, flexibilidade e métricas de flexibilidade.

Granularidade especifica se as relações são entre pontos do tempo (um ponto de sincronização, previsível ou não, um tempo absoluto, um tempo relativo a um ponto de

sincronização, ou um evento de seleção), entre intervalos temporais (um evento), ou ambos.

Os tipos de relação podem ser: de ordem, de duração, contínua, de grupo, de iteração e condicional. Em mais detalhes e exemplificando, tipo de relação especifica se é uma relação de ordem (relação binária que determina a ordem de ocorrência de pontos ou intervalos de tempo em um documento), se de duração (que requer, por exemplo, que a duração de dois segmentos sejam idênticas — caso particular da relação de ordem), se contínua (quando o intervalo entre dois pontos de sincronização é determinado a partir de alguma informação especificada pelo usuário ou calculada usando parâmetros, como o tipo das mídias envolvidas e os níveis de percepção do ser humano, e este intervalo é mantido durante toda a apresentação da mídia — por exempo, é assumido que se o áudio e vídeo são sincronizados dentro de 150 ms, a diferença temporal entre os dois fluxos não deteriora a qualidade da apresentação), se de grupo (que permite agrupar pontos ou intervalos não relacionados para serem usados como uma única entidade em outra relação temporal — relação n:m), se de iteração (que permite especificar a exibição de um intervalo de tempo um número determinado de vezes), se condicional (que são dependentes se o documento ou sistema está ou não em um estado — utiliza eventos de mudança de atributo).

Flexibilidade especifica se a relação é obrigatória ou opcional, ou o intervalo de tempo que a relação temporal pode ser considerada satisfeita. Métricas de flexibilidade fornecem uma base objetiva de escolha entre as opções de flexibilidade.

Ainda outros atributos de uma relação temporal podem ser citados, como a especificação do comportamento espacial em um ponto ou intervalo, conforme discutido na Seção 2.1 como sendo uma das possíveis funções do descritor, cuja definição pode ser incorporada ao relacionamento temporal.

Em [Diaz & Sénac 1994], é definido um modelo de rede de Petri temporizada (TSPN - Time Stream Petri Nets) , onde um evento é representado por um lugar e pela associação de uma tupla [t_{min}, t_{ot}, t_{max}] ao arco que sai do lugar, onde a tupla [t_{min}, t_{max}] especifica o intervalo possível de duração do evento, e t_{ot} o tempo de duração com a melhor qualidade de apresentação. Na TSPN, retardos podem ser introduzidos através da adição de lugares com arcos de partida associados a um intervalo de possível ocorrência de retardo. Um hiper-elo (elo hipermídia) é representado por um novo tipo de lugar, tendo a tupla [t_{min}, t_{ot}, t_{max}] associada ao arco de saída do lugar o significado usual, representando o intervalo possível de disparo do hiper-elo (note que t_{max} pode ser ∞).

TSPN permite relações n:m. Por não endereçar problemas de estruturação da sincronização e também não permitir a especificação dos parâmetros de acesso à informação, nem características da apresentação audível e espacial, [Willrich 1996] propõe extensões a essas redes, as I-HTSPN, comentadas na próxima seção.

CMIF utiliza arcos de sincronização para a especificação da sincronização. Um arco de sincronização é uma relação entre dois eventos. As relações temporais são sempre 1:1 e especificam um retardo e um desvio permissível para o retardo. Os eventos no CMIF têm sempre duração previsível e determinada (não pode ser especificada por um intervalo) e referem-se à apresentação de todo o segmento de mídia (granularidade grossa). Um tipo especial de arco entre eventos, denominado continuous syc arc, determina que a variação

de retardo especificada deve ser mantida durante toda a apresentação concorrente dos eventos, realizando a relação do tipo contínua anteriormente mencionada.

Modelos orientados a objeto, tais como MHEG [MHEG 1995] e HyperProp, representam suas relações temporais nos elos. Nestes modelos, documentos são definidos com base nos conceitos usuais de nós e elos. *Nós* são fragmentos de informação e *elos* são usados para a interconexão de nós que mantêm alguma relação. O modelo distingue duas classes básicas de nós, chamados de nós de conteúdo e nós de composição, sendo estes últimos o ponto central do modelo. Intuitivamente, um *nó de conteúdo* contém dados cuja estrutura interna é dependende da aplicação (se constituem nos nós hipermídia tradicionais). A classe de nós de conteúdo pode ser especializada em outras classes (texto, gráfico, áudio, vídeo, etc.), conforme requerido pelas aplicações. Um *nó de composição* contém uma coleção de elos e nós, de conteúdo ou de composição, recursivamente. Nós podem ser interpretados tanto ao nível de objetos de dados, quanto ao nível de objetos de representação. Uma classe especial dos modelos é a classe descritor, mencionada na Seção 2.1.

Elos são definidos nos nós de composição, ao contrário de modelos, como os baseados em HTML, onde os elos são definidos nos nós de conteúdo. Um elo é caracterizado como uma relação *m:n* entre um conjunto de pontos terminais origem e destino. O conjunto de pontos terminais origem e destino especificam eventos associados aos nós. O elo possui ainda associado uma lista de *condições* e *ações*, especificando relações entre os eventos. Condições precisam ser satisfeitas em eventos de origem para que ações sejam aplicadas nos eventos de destino. Exemplos de condições são: evento *E* ocorreu, ou ocorrendo, etc. Exemplos de ações são: iniciar um evento, modificar a apresentação de um nó em exibição, etc. Se um elo possui ao menos um evento imprevisível associado a um de seus pontos de extremidade origem, ele é classificado como um *elo hipermídia (hiper-elo).* Caso contrário, o elo é classificado como *elo de sincronismo (sinc-elo).*

Modelos orientados a objeto são abstrações de mais alto nível que os anteriormente apresentados e permitirão, como se verá nas Seções 2.3 e 2.4, uma interface bem simples para a especificação total de um documento (sua estrutura, relação temporal e espacial entre seus componentes, etc.).

2.3. Composições na Estruturação de Documentos

A definição estruturada de documentos é desejável por trazer embutidos os conceitos de modularidade, encapsulamento e mecanismos de abstração. A estruturação lógica do documento é realizada pela introdução do conceito de composição, como um agrupamento de componentes do documento e de suas relações.

Propriedades desejáveis da estruturação por composições são:

- Permitir o aninhamento das composições, isto é, uma composição conter outra composição.
- Permitir o agrupamento dos seus componentes e suas relações, independente dos tipos de eventos relacionados (seleção, apresentação ou atribuição).
- Permitir que uma composição possa ser usada como se fosse um novo tipo de segmento, em todos os sentidos, isto é:

  i. que possa ser exibida — é importante em uma apresentação exibir não apenas o conteúdo de seus dados, como sua estrutura (por exemplo, ao acessar uma composição representando o capítulo de um livro, pode-se querer não apenas a visualização do conteúdo de dados do capítulo, mas também de sua estruturação em seções).

ii. que possa auxiliar o leitor em uma navegação sobre o documento — o que vai implicar na necessidade da aplicação de algum algoritmo de filtragem, tal como olho de peixe estendido [Muchaluat et al. 1996], quando da exibição da estrutura, de forma a diminuir o problema de desorientação do leitor.
iii. que possam ser definidas relações entre composições

- Permitir que se tenha vários pontos de entrada em uma composição, isto é, que uma composição possa ter uma exibição diferente de seus componentes, dependendo do ponto de entrada. A duração de um nó de composição vai, assim, depender do ponto de entrada e não apenas da duração de seus componentes.
- Permitir a herança na definição das composições, no sentido de que relações possam ser definidas em uma composição *C*, referenciando a componentes recursivamente contidos em composições contidas em *C*. Este mecanismo é extremamente importante no reuso de composições. Como exemplo, tome uma composição representando o capítulo de um livro. Para um livro (outra composição L1) entregue a um leitor, poderia ser desejável a introdução de uma relação entre duas seções do capítulo, por exemplo para indicar um complemento de informações, que não seria necessário a um leitor de maior conhecimento, a quem pode ser entregue outro livro (outra composição L2), sem a relação. Note que L1 poderia ser definida simplesmente como uma composição contendo L2 e a relação introduzida, reusando toda a estruturação de L2.

O modelo orientado a objetos utilizado no sistema HyperProp permite a definição de composições com todas as características especificadas no parágrafo anterior.

O sistema Firefly não apresenta composições, que são endereçadas como um trabalho futuro. Tanto no CMIF quanto no I-HTSPN, as composições definidas não permitem herança de relacionamentos, nem a exibição da estrutura definida na composição.

No CMIF as composições são sempre hierárquicas e só permitem relacionamentos entre eventos de apresentação. Elas podem ser de dois tipos: paralelas, onde todos os componentes começam a ser exibidos juntos, ou seqüenciais, onde um componente é exibido após o outro. Os relacionamentos são implicitamente dados pela composição, que não permite nenhum outro tipo de relacionamento. Como trabalhos futuros, CMIF pretende colocar tempo máximo e mínimo nas restrições definidas por seus relacionamentos, o que não deve ser difícil para relacionamentos definidos nos seus arcos (vide Seção 2.2), mas deve complicar nos relacionamentos definidos em suas composições. Note que, ao contrário do HyperProp, no CMIF existem relacionamentos definidos nas composições e outros definidos nos arcos. Quando definidos em arcos, não permitem um reuso estruturado.

Note que em modelos orientados a objetos, como o utilizado no MHEG e HyperProp, é sempre possível definir uma subclasse composição-paralela e uma subclasse composição-seqüencial, a partir da classe composição, que terão as mesmas funcionalidades das composições definidas no CMIF.

As composições do I-HTSPN são um pouco mais gerais que as do CMIF, mas não tão gerais como as do HyperProp. Nas I-HTSPN, relacionamentos podem se referenciar a uma composição, mas não a componentes existentes dentro de uma composição. Composições são representadas por um tipo especial de lugar na rede. Um lugar de composição e sua subrede relacionada devem ser não apenas estruturalmente equivalentes (isto é devem ter um lugar de entrada e um lugar de saída), mas também temporalmente equivalentes. Desta forma, as composições I-HTSPN como as composições CMIF, só permitem um ponto de entrada.

Nas composições reside uma das diferenças mais fundamentais entre o sistema HyperProp e os sistemas CMIF e I-HTSPN. Esses últimos partem do paradigma que as composições são usadas apenas para a estruturação da apresentação dos documentos, ou que essa estruturação se confunde com a estruturação lógica do documento, o que nem sempre é verdade. Nestes modelos não é possível, por exemplo, definir uma composição *capítulo A*, que pode, sob a intervenção do leitor pelo disparo de um evento de seleção, acionar outra composição *capítulo B*, que não está contida em A. Para tanto, componentes dos *capítulos A e B* deveriam estar definidos em uma mesma composição, perdendo assim a estruturação lógica. No HyperProp as composições representam, de fato, uma estruturação lógica do documento.

2.4. Ambientes para Projeto de Documentos: Autoria no Sistema HyperProp

Quase todos os sistemas comentados no texto apresentam uma interface gráfica visando facilitar a autoria de documentos. A seguir é apresentada, resumidamente, o ambiente do sistema HyperProp e depois comentados, comparativamente, os outros sistemas.

O processo de autoria de documentos multimídia no HyperProp é apoiado em três diferentes visões, como apresentado na Figura 1.

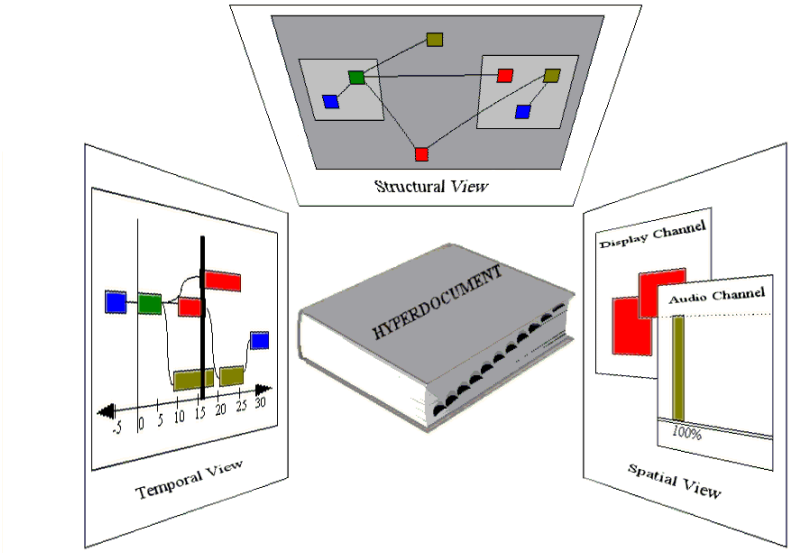

Figura 1 - Diferentes visões de um documento multimídia.

A primeira visão, Structural View, considera a edição da estrutura do documento, fornecendo recursos para editar fragmentos de informação (nós), seus relacionamentos, e o agrupamento desses nós em composições. Essa visão é apresentada no plano superior da Figura 1, onde fragmentos de informação são representados por retângulos, e o relacionamento de inclusão em composições é representado pela inclusão de um retângulo (nó) em outro retângulo (nó de composição). Algoritmos sofisticados de filtragem desenvolvidos evitam a desorientação do autor, facilidade em nenhum dos outros sistemas encontrada, onde a construção de documentos extensos pode causar uma desorientação total [Muchaluat et al. 1996].

A segunda visão, Temporal View, é responsável pela especificação dos relacionamentos temporais entre os componentes de um documento multimídia, definindo suas posições relativas no tempo, como exibido no plano da esquerda da Figura 1. Nesse plano, nós são representados por retângulos, cujos comprimentos indicam suas durações esperadas de exibição e cujas posições relativas definem suas sincronizações no tempo. Uma diferença

importante da Temporal View para o timeline é que os tempos mostrados no diagrama não são diretamente especificados pelo autor. Eles são derivados das restrições temporais. Eles são, por conseguinte, apenas uma aproximação dos tempos reais da apresentação.

Finalmente, a terceira visão, Spatial View, permite a definição de relacionamentos espaciais entre componentes de um documento multimídia, estabelecendo suas características de apresentação em um determinado dispositivo, em um dado instante do tempo, como ilustrado no plano da direita da Figura 1. Nesse plano, os retângulos representam as características espaciais dos objetos em um dispositivo de exibição, especificando sua posição em um monitor de vídeo, seu volume em um dispositivo de áudio, etc.

As três visões estão relacionadas, de modo que um objeto em foco na Structural View é a base para a seqüência de sincronismo mostrada na Temporal View. Nessa última visão, para cada ponto selecionado no tempo, a Spatial View mostra como os componentes serão apresentados no espaço definido pelos dispositivos de saída. Elos e nós definidos na Temporal View são imediatamente incorporados na Structural View, e vice-versa.

O sistema CMIF também se apóia em três visões. A chamada Hierarchy View é usada para a visualização aninhada das composições CMIF. A chamada Channel View é semelhante à Temporal View do HyperProp. No entanto, como ela não tem relação com a Hierarchy View, não existe a noção de composição nos relacionamentos nela (Channel View) definidos. A terceira visão do documento é chamada Player. Player, de fato, é o próprio executor e não faz parte da autoria. O Player pode ser diparado de qualquer parte do documento. Todo posicionamento espacial é feito através dele. Quando o Player faz uma apresentação, os nós correspondentes na Channel view são salientados.

O sistema Firefly, como mencionado, não permite a definição de composições. Uma interface gráfica apresenta para cada segmento de mídia uma timeline (desenhada como uma linha vertical), baseada na duração ótima dos eventos, onde são especificados (um ponto marcado na linha de timeline) todos os pontos de sincronização relativos aos eventos de apresentação. Eventos cuja duração máxima é indeterminada são representados na timeline por uma linha pontilhada. Eventos de seleção são representados por um ponto colocado acima da timeline correspondente ao segmento de mídia onde foi definido. Os relacionamentos temporais entre eventos são representados por arcos rotulados ligando eventos, onde o rótulo especifica o tipo de relacionamento.

Nas I-HTPSN, um editor gráfico guia o autor na construção do documento. Um editor HTSPN permite a construção gráfica da rede de Petri, representando, como usual, lugares como círculos, transições como barras e arcos como setas interligando lugares e transições. Duas janelas, chamadas *Data Specification* e *Channel Specification*, permitem a especificação dos segmentos de mídia e das características de apresentação de cada segmento, assim como do espaço lógico onde apresentação será relizada.

Por ser uma abstração de mais alto nível que os grafos de dependências temporais e redes de Petri, os modelos orientados a objetos permitem uma interface com o usuário muito mais amigável. Ao contrário do que afirma Willrich [Willrich et al. 1996], não é irreal descrever documentos hipermídia usando esses modelos, como o apresentado no padrão MHEG. Basta que seja construída uma boa interface com o usuário. Se assim for feito, a expressividade muito maior de tais modelos pode ser passada de modo fácil e compreensível. Além de tudo, outras facilidades podem ser facilmente agregadas ao modelo, como controle de versões, facilidades para trabalho cooperativo, entre outras [Soares et al. 1995], que seriam de difícil incorporação nos outros modelos. Modelos

orientados a objetos, tais como o utilizado no HyperProp, podem também ser mais facilmente traduzidos para o padrão de intercâmbio de objetos multimídia/hipermídia definido no MHEG, e explorando toda sua potencialidade. Em outras palavras, advoga-se aqui a utilização de um modelo orientado a objetos como o resumidamente descrito para o sistema HyperProp, para a especificação do documento. Deste modelo, será extraída a estrutura que alimentará o formatador temporal e, aí sim, por estar mais perto da máquina, e não do usuário, um modelo de mais baixo nível para máquinas de estado paralelas pode ser o mais adequado, como discutido na próxima seção.

3. Formatação Temporal

Um formatador temporal automático usa a especificação temporal (consistindo dos objetos de representação, das relações temporais e da especificação da plataforma de exibição) para produzir o plano de execução temporal, que indica quando os eventos em um documento devem ocorrer. O plano serve como guia de apresentação ao formatador, permitindo ao mesmo antecipar operações futuras (p. ex., pré-busca), ou reagir em caso de alterações no comportamento esperado. Assim, as principais tarefas do formatador são a geração e manutenção de um plano de execução, o controle da apresentação de cada um dos componentes do documento e a alocação dos recursos para exibição das mídias.

Abstratamente, o encadeamento de pontos de sincronização é denominado uma cadeia temporal. Uma cadeia temporal é dita parcial quando o seu primeiro evento é imprevisível. Um plano de execução de um documento é construído pela concatenação de uma ou mais cadeias temporais parciais, onde todos os pontos de sincronização imprevisíveis são associados a um tempo de ocorrência, em tempo de execução.

Esta seção apresenta, resumidamente, as principais características de um formatador temporal. Uma discussão mais detalhada pode ser encontrada em [Rodrigues et al. 1997], onde o formatador temporal do sistema HyperProp é apresentado em detalhes. Usando a mesma a taxonomia apresentada na referência citada (uma extensão à proposta de [Buchanan & Zellweger 1993], um fomatador temporal pode ser analisado quanto ao momento em que se constrói o plano de exibição (assunto da próxima seção); à flexibilidade usada na sua construção (se usa ou não a flexibilidade especificada para a duração de eventos); aos relacionamentos temporais que dá suporte (se apenas entre eventos previsíveis, ou também incluem eventos imprevisíveis); às mudanças comportamentais que admite, quer especificadas na fase de autoria (tais como o aumento do volume de um áudio em um dado instante, aceleração de um vídeo), bem como mudanças de comportamento provenientes de interação dinâmica com o usuário (no controle da taxa de apresentação, etc.); às inconsistências analisadas, temporais (detecção de falhas na especificação do autor) e espaciais (como por exemplo, a ausência de suporte à áudio num determinado equipamento); e a como reage à variações imprevisíveis (quando da alteração na taxa de exibição, correspondendo a atrasos ou adiantamentos na exibição dos objetos, causados pela transmissão do conteúdo das mídias em redes que apresentam retardos aleatórios, limitações dos dispositivos de exibição e armazenamento, e mesmo limitações do sistema operacional). Um formatador temporal deve ser também capaz de considerar as capacidades da estação e possivelmente da rede (especificação do ambiente).

3.1. Arquitetura dos Formatadores Temporais

Um formatador pode apresentar três fases: a pré-compilação, a compilação e a execução. O que as diferencia é o momento em que o plano de execução é construído. A Figura 2 [Rodrigues et al. 1997] apresenta a arquitetura geral de um formatador.

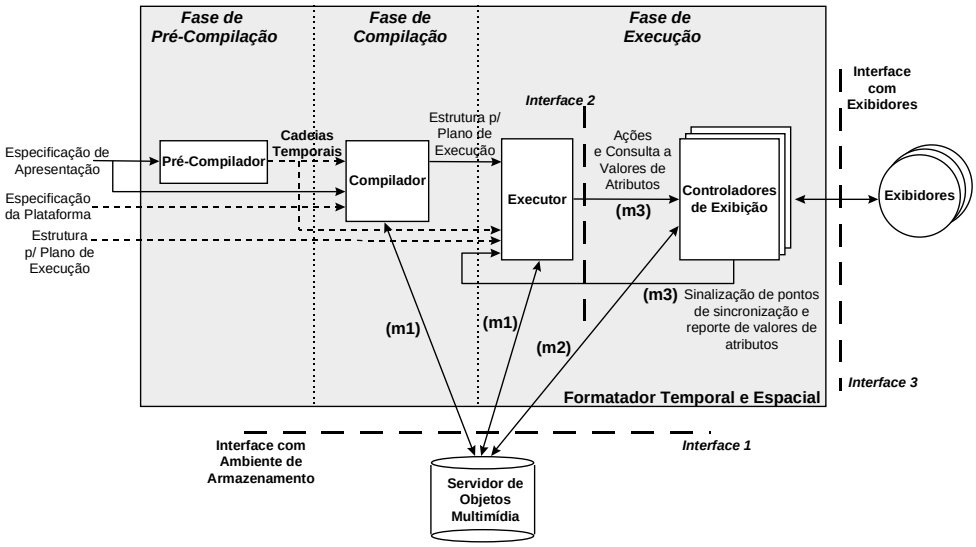

Figura 2 - Arquitetura genérica do ambiente de execução

A pré-compilação incremental, durante a criação da especificação temporal, pode ajudar ao autor a localizar erros, fornecendo uma realimentação imediata quando a edição provoca um descasamento temporal, espacial e de plataforma. O Pré-Compilador pode ou não alimentar o Compilador, ou o Executor, com as cadeias temporais parciais que cria.

O Compilador é responsável por receber toda a especificação de apresentação do documento e gerar como saída uma estrutura de dados através da qual o Executor pode gerar o plano de execução. É interessante observar que, no caso do documento possuir todo o seu comportamento de apresentação previsível, tanto o Pré-Compilador, como o Compilador, geram o mesmo resultado, pois haverá uma única cadeia temporal. Neste caso, o plano de exibição não precisa ser construído em tempo de exibição e pode ser gerado e passado pelo Compilador diretamente ao Executor.

O Executor é o elemento destinado às questões que só podem ser resolvidas no momento da apresentação do documento, como o tratamento de eventos imprevisíveis. Ao Executor cabe instanciar os Controladores de Exibição e reagir às sinalizações de ocorrência dos eventos, por eles informada.

Os Controladores de Exibição são responsáveis pela interface do sistema com os objetos em exibição. Deve existir uma instância de Controlador para cada objeto. O Controlador recebe dos objetos a sinalização de ocorrência de pontos de sincronização, tais como interação do usuário, ou início e fim de apresentação de um segmento de mídia, e as reporta ao Executor. Mudanças de comportamento realizadas pelo usuário em tempo de exibição, que afetam a exibição temporal, também são sinalizadas. Do Executor, por sua vez, os Controladores recebem comandos de apresentação, tais como preparar, iniciar, acelar, retardar, parar ou abortar a exibição do conteúdo de um objeto de representação.

O formatador Firefly, excetuando o fato de não possuir um pré-compilador, possui uma arquitetura idêntica à da Figura 2. Pretende-se, no entanto, ter, no futuro, uma compilação incremental nos moldes da atribuída ao Pré-Compilador. Seu Compilador gera cadeias temporais parciais em timeline, como estrutura de dados passada ao Executor.

O Player é o formatador temporal do CMIF. Ele possui um compilador capaz de gerar, a partir da especificação do documento, um grafo de dependências temporais como estrutura de dados passada ao executor.

O Toolkit-HTSPN tem um compilador, denominado Analyser, que realiza a análise temporal e detecta conflitos na utilização de recursos, realizando funções de um pré-compilador. Um outro módulo, chamado MHEG translator, gera a partir da especificação I-HTSPN uma representação MHEG. Cabe a uma máquina MHEG fazer a execução do documento. A próxima seção tece mais comentários sobre essa abordagem.

O formatador temporal HyperProp possui todos os componentes apresentados na Figura 2. Nessa figura, as linhas pontilhadas não existem no formatador HyperProp. O pré-compilador só atua no teste de consistência temporal de uma cadeia temporal parcial e detecta conflitos na utilização de recursos, mas não passa nenhuma informação ao compilador. Este último deve, a partir da especificação do documento, gerar uma estrutura de dados, baseada no modelo TSPN, para servir de entrada ao executor. [Rodrigues et al. 1997] apresenta uma discussão minuciosa do formator HyperProp.

3.2. Estrutura Temporal de Execução

Um possível modelo de dados para a especificação da apresentação do documento pode ser o mesmo utilizado para a estrutura de dados interna do formatador, que servirá de base para a geração dos planos de execução. No entanto, geralmente, essas estruturas são complexas para manipulação direta, principalmente quando os documentos possuem uma quantidade grande de componentes e relacionamentos, que escondem toda a estruturação dos documentos. Como conseqüência, uma linguagem de mais alto nível tem se mostrado mais adequada a autoria. A tradução da abstração de mais alto nível para a estrutura interna do formatador torna-se, assim, a ponte entre o ambiente de autoria e o formatador.

A estrutura interna de sincronização do formatador pode se utilizar de timelines, apresentando basicamente os mesmos problemas mencionados na Seção 2.2. Firefly tenta contornar o problema através da concatenação de cadeias temporais parciais para dar suporte à manipulação de comportamentos imprevisíveis. O escalonador (compilador) constrói a cadeia temporal principal do início da apresentação e, se necessário, várias outras cadeias auxiliares (cadeias temporais parciais), quando um usuário pede ao sistema a apresentação de um documento. O executor utiliza essas cadeias para montar o plano de execução. Como cada cadeia usa um timeline, ajuste que porventura o executor queira fazer, devido a um reporte dos controladores, pode ter uma grande repercussão no restante da cadeia. Embora localizado em uma cadeia, o problema de alteração mencionado na Seção 2.2 permanece e se agrava na medida que a cadeia temporal parcial cresce. Hoje Firefly só permite ajustes em tempo de compilação, mas no futuro isso pode ser um problema.

Como visto, estruturas de sincronização baseada em restrições (ou eventos), não possuem as limitações descritas acima. No entanto, causam um maior overhead de mensagem e programação, uma vez que é necessário que o editor da mídia reporte os eventos. Um preço que pode ser muito pequeno para a flexibilidade adquirida.

A utilização de modelos orientados a objetos não é apropriada nesse nível de abstração. Caso utilizados como a estrutura interna, o executor, na geração do plano de execução, certamente a converterá para outra estrutura que melhor espelhe a máquina de estados do ambiente de execução, como uma estrutura em timeline ou mesmo em rede de Petri.

A noção de estado global permite uma simulação natural do comportamento dinâmico do sistema modelado com uma TSPN. Este fato e toda a sua facilidade de expressão de relacionamentos temporais (não de composição de estruturas lógicas, que não é mais necessário neste nível de abstração) caracteriza a TSPN como uma forte candidata para a

estrutura interna de dados de um formatador. Este fato levou o formatador HyperProp a adotá-la. No HyperProp é realizada uma conversão do modelo de especificação para o modelo utilizando TSPN, que é então usada na geração do plano de execução.

Ao contrário, [Willrich 1996] especifica um documento em I-HTSPN e depois traduz essa especificação para o modelo orientado a objetos do padrão MHEG. Uma máquina MHEG será então responsável pela geração do plano de controle. Provavelmente esta máquina terá, a partir da especificação em MHEG, de gerar uma outra estrutura de dados, talvez mesmo, baseada em redes de Petri, voltando a uma estrutura próxima à primeira.

Métodos de verificação foram desenvolvidos para checar a consistência temporal de uma TSPN. Ora, a utilização de uma TSPN como estrutura de dados interna do executor permitirá não apenas o teste da consistência quando da geração da especificação, mas também da consistência durante a execução, uma vez que o plano de execução pode sofrer mudanças ao longo do tempo, devido a eventos imprevisíveis e diretivas do usuário que alterem o fluxo normal da apresentação.

CMIF utiliza grafos de dependências temporais para a especificação da sincronização. No grafo, os nós representam pontos de sincronização e as arestas especificam um retardo e um desvio permissível para aquele retardo. O desvio especificado é usado pelo formatador (player) para compensar os retardos específicos da plataforma. Os grafos utilizados pelo CMIF têm características próximas a de uma rede de Petri.

3.3. Algoritmos de Elasticidade

Recentemente, as noções de esticar e encolher durações têm sido exploradas para obtenção de planos de execução. Conceitualmente, um formatador temporal procede da seguinte forma. Primeiro, ele tenta posicionar os segmentos da mídia, sem considerar a flexibilidade de suas durações e relações temporais. Se o plano não ficar consistente e apresentável na plataforma de exibição, o formatador usa a flexibilidade para reformulá-lo.

A determinação de que algoritmo utilizar para a escolha da duração, aqui chamado algoritmo de eslasticidade, ainda é um problema em aberto. Alguns algoritmos interessantes incluem: ordenação topológica, caminho mais curto entre pares, programação linear, programação dinâmica, hierarquias de restrição e algoritmos incrementais.

No Firefly, o escalonador resolve o problema de atribuição de tempo usando a técnica de programação linear. No entanto, a solução é limitada, uma vez que só é computada uma solução (de menor custo) para um conjunto de restrições. Para encontrar uma outra solução, que poderia existir, deve-se alterar custos de elasticidade e resolver o problema novamente. Como é o compilador, e não o executor, que realiza o algoritmo de elasticidade, não há como corrigir assincronismos devido a eventos imprevisíveis.

O paradigma de autoria com tempo elástico foi também implementado como parte do ambiente Isis de autoria [Kim & Song 1995]. A interface gráfica do Isis permite ao autor manipular diretamente as caixas de tempo (representação de nós semelhantes as da Time View do sistema HyperProp). O sistema calcula e exibe, na correspondente visão temporal do Isis, a solução ótima. Durante o cálculo, a consistência temporal também é testada. No algoritmo utilizado, a solução ótima é encontrada além de um domínio de soluções, dando ao autor a capacidade de refinar, interativamente, o projeto do documento. Se desejado, o autor pode escolher uma alternativa, dentro do domínio de soluções, para o comprimento do documento, fazendo com que o sistema compute a solução revisada. O sistema usa o algoritmo *pairs shortest paths*. O algoritmo também permite ao autor reini-

ciar, a partir de um ponto, o comprimento de uma história que já foi gerada, e aplicar o algoritmo a partir de então. Isso torna possível a execução do algoritmo à medida que o plano de execução é alterado por eventos imprevisíveis, tais como uma interação do usuário que mande alterar a velocidade de exibição de um dado objeto. O algoritmo minimiza o custo total de alterar as durações e também se preocupa em compartilhar o custo da forma mais justa possível. No entanto, pode ser melhor concentrar o custo de alteração em poucos objetos. A descrição feita do algoritmo não relata esta possibilidade.

Uma vez ocorrido um evento imprevisível que altera o plano de execução, várias alternativas podem ser tomadas: bloquear a espera do fluxo em atraso; ou não bloquear e tomar a ação corretiva na transição para o próximo objeto a ser exibido, quando então pode-se abortar o fim do fluxo mais lento, ou esperar pelo término deste fluxo para depois começar a apresentação do próximo objeto; ou não bloquear e, quando o assincronismo tornar-se maior que um certo limite, acelerar o fluxo mais atrasado.

HyperProp não implementa ainda algoritmos de elasticidade, embora o modelo usado, tanto para autoria quanto para execução, preveja seu uso no futuro.

4. Conclusões

O ambiente para autoria e formatação temporal do sistema HyperProp vem sendo desenvolvido no Laboratório TeleMídia da PUC-Rio há três anos, sendo que o modelo para especificação de documentos foi proposto e vem sofrendo refinamentos desde 1991.

A implementação atual foi realizada sobre a plataforma UNIX usando a linguagem C++. Para implementar a visão estrutural, foram utilizadas rotinas do editor de grafos compostos D-ABDUCTOR, da FUJITSU Laboratories LTD, que fornece o layout automático de grafos e recursos de animação. As visões temporal e espacial foram implementadas utilizando um conjunto de ferramentas portáteis para a construção de interfaces, chamadas IUP/LED e CD. A utilização de ferramentas distintas para a construção das interfaces foi considerada, inicialmente, irrelevante para o trabalho. Um dos próximos passos é homogeneizar a interface do editor gráfico integrado, utilizando uma ferramenta portátil. O ambiente de autoria integrado se encontra operacional desde final de 1995. O formatador temporal encontra-se na sua primeira versão, como relatado ao longo do texto.

A experiência adquirida no projeto nos leva a tirar várias conclusões, entre elas as que passamos a citar. Primeira, que não só é possível, mas extremamente mais flexível trabalhar com modelos de mais alto nível, como os de restrições temporais orientados a objetos, para a especificação de documentos em subsistemas de autoria. As vantagens cremos que foram salientadas no texto. Segundo, que partir de uma estruturação lógica do documento, e não só de uma estruturação para apresentação, é mais natural e facilita o trabalho do autor. Assim, modelos com composição permitindo todos os tipos de relacionamentos entre seus componentes passam a ser muito importantes. Terceiro, que é possível construir interfaces facilmente utilizáveis, mesmo usando modelos com tanta expressividade, que possibilitem a construção de documentos extensos, sem que o autor se perca no emaranhado de seus componentes. Para tanto, mecanismos eficientes de filtragem e animação foram desenvolvidos no projeto do exibidor da visão estrutural. Quarto, que a pré-compilação é um estágio extremamente útil para auxiliar o autor na construção de documentos consistentes. Quinto, que modelos de mais baixo nível, ou seja, mais próximos da execução da máquina, devem ser usados nas estruturas internas do formatador, o que facilita em muito alterações do plano de execução em tempo de exibição.

Como trabalhos futuros, pretende-se integrar o controlador ao subsistema de armazenamento de dados, de forma que possam ser realizadas a negociação de parâmetros de QoS no transporte dos objetos multimídia para exibição. Isto permitirá a realização de algoritmos de pré-busca e, assim, um melhor controle de exibição. Almeja-se, também, usar a flexibilidade especificada para a duração de eventos e permitir mudanças comportamentais provenientes de interação dinâmica com o usuário, que não são realizadas na implementação atual. Pretende-se, ainda, acoplar algoritmos de elasticidade ao executor, de forma a permitir a reação a comportamentos imprevisíveis, tais como a variação na taxa de exibição, devido a atrasos ou adiantamentos na exibição dos objetos, causados pela transmissão do conteúdo das mídias em redes que apresentam retardos aleatórios, a limitações dos dispositivos de exibição e armazenamento, etc.

Agradecimentos

Este trabalho foi desenvolvido com o apoio do CNPq, através do projeto ProTeM II - HyperProp, e com o apoio da Embratel, através do projeto RAVel. Os autores gostariam de agradecer a Guido Souza, Fábio Costa e Débora Muchaluat pelas valiosas discussões sobre o tema e pelas implementações que permitiram o teste das idéias apresentadas.

Referências

[Buchanan & Zellweger 1992] Buchanan, M.C.; Zellweger, P.T. “Specifying Temporal Behavior in Hypermedia Documents”, *Proceedings of European Conference on Hypertext, ECHT’92*. Milano. Dezembro 1992.

[Buchanan & Zellweger 1993] Buchanan, M.C.; Zellweger, P.T. “Automatic Temporal Layout Mechanisms”. *Proceedings of ACM Multimedia’93*, Anaheim, California. 1993. pp. 341-350

[Diaz & Sénac 1994] Diaz, M.; Sénac, P. “Time Stream Petri Nets, a Model for Timed Multimedia Information”, *Proc. Of the 15th Int. Conf. On Application and Theory of Petri Nets*, Zaragoza, 1994. pp. 219-238

[Kim & Song 1995] Kim, M.Y.; Song J. “Multimedia Documents with Elastic Time”. *Proccedings of ACM Multimedia’95*, San Francisco, California. Novembro 1995.

[MHEG 1995] Multimedia and Hypermedia Information Coding Expert Group (MHEG). *ISO/IEC DIS 13522-1* - Coded Representation of Multimedia and Hypermedia Information (MHEG), Part I: MHEG Object Representation, Base Notation (ASN.1). Setembro 1995.

[Muchaluat et al. 1996] Muchaluat, D.; Soares, L.F.G.; Casanova M.A. “Editores Gráficos para Estruturas com Composições”. *Anais do Simpósio Brasileiro de Engenharia de Software*, São Carlos. Outubro de 1996.

[Rodrigues et al. 1997] Rodrigues, R.F.; Soares, L.F.G.; Souza, G.L. “O Ambiente de Execução do Sistema HyperProp para Apresentação de Documentos Multimídia/Hipermídia”. *III Workshop sobre Sistemas Multimídia e Hipermídia*, São Carlos. Maio 1997.

[Soares et al. 1995] Soares, L.F.G.; Casanova, M.A.; Rodriguez, N.L.R. "Nested Composite Nodes and Version Control in an Open Hypermedia System". *International Journal on Information Systems; Special issue on Multimedia Information Systems*. September 1995.

[van Rossum et al. 1993] van Rossum, G.; Jansen J.; Mullender K.S.; Bulterman C.A. “CMIFed: A Presentation Environment for Portable Hypermedia Documents”. *Proceedings of ACM Multimedia’93*, Anaheim, California. 1993.

[Willrich et al. 1996] Willrich, R.; Sénac, P.; Saqui-Sannes, P.; Diaz, M. “Towards Hypermedia Documents Design”. *Anais do XIV SBRC*, Fortaleza. Maio 1996. pp. 473-491.