UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Fabiano Fagundes

ESPECIFICAÇÃO DE UMA META-LINGUAGEM PARA SINCRONIZAÇÃO MULTIMÍDIA

José Mazzucco Júnior

(Orientador)

Florianópolis, julho de 2002

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Fabiano Fagundes

ESPECIFICAÇÃO DE UMA META-LINGUAGEM PARA SINCRONIZAÇÃO MULTIMÍDIA

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

José Mazzucco Júnior

(Orientador)

Florianópolis, julho de 2002

ESPECIFICAÇÃO DE UMA META-LINGUAGEM PARA SINCRONIZAÇÃO MULTIMÍDIA

Fabiano Fagundes

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Área de Concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

José Mazzucco Júnior, Dr.

_____

Fernando Ostuni Gauthier, Dr.

Banca Examinadora

_____

José Mazzucco Júnior, Dr.

_____

Elizabeth Specialski, Dra.

_____

João Bosco da Mota Alves, Dr.

AGRADECIMENTOS

Gostaria de agradecer às inúmeras pessoas que foram de extrema importância para que eu chegasse até aqui. Confesso que, devido a várias circunstâncias, a relação de pessoas a quem eu devo um sincero agradecimento extrapola tempo e distância. Corro o risco de deixar alguém de fora, mas quem esteve presente nesta etapa de minha vida sabe de sua importância.

Agradeço aos meus pais e irmãos, e também aos meus sobrinhos, que, ainda que não façam nem idéia do que acontece no desenvolvimento de um trabalho como este, estiveram presentes como uma parte segura da minha história, afinal, são minha família.

Aos meus amigos, aqueles que foram minha família em diversos momentos. Não sei se conseguirei lembrar de tantos, mas aqui vai meu agradecimento a grandes figuras da minha vida: Letícia, Duda, Ana Paula e Jussara, Marinês Pinho, Ana Elisa, Jaqueline, Mirla, Márcia, Sandra, Patrícia, Juliana, Marisa, Ernesto, Luis Carlos, Paulo e Simone (mais o Felipe). Obviamente, não posso deixar de citar os grandes amigos, Mário Oleskovicz, Luis Augusto (Garga), Luciano Senger e Rudinei Goulart, amigões de boas festas e aventuras.

Agradecimento especial e carinhoso a Parcilene e a Thereza, conhecedoras de todos os processos pelos quais passei, e também aos meus amigos e colegas de trabalho, que acompanharam e apoiaram minhas iniciativas: Carmen e Hugo, Paula e Josué, Delzimar e Pedro, Mário Sérgio, Andrés, Piveta, Deise, Augusto, Leal, Madson, Mádia, Irenides, Darlene. Aqui também entram os meus agradecimentos às ex-alunas e agora amigas Cristina e Lilissanne, companheiras de jornada.

Sem esquecer o apoio de alunos e ex-alunos que se fizeram presentes, especialmente aos amigos Carlos Henrique, Ricardo Marx, Jack, Wagner (Tigrão) e Anderson; e ao aluno, amigo e orientando (que muitas vezes assumiu papel de orientador) Fernando, peça-chave na conclusão deste trabalho. Compartilho esse carinho com os grupos de alunos que estiveram acompanhando e torcendo pelo sucesso dessa empreitada, especialmente os “branquinhos” e os hóspedes do “temático”, testemunhas de constantes “pseudos”-ataques criativos, e suas terríveis conseqüências.

Aos professores que, em diferentes momentos, deram sua contribuição: Raul, DeLucca, Marta, Isaias, Graça, entre outros. Agradecimento todo especial ao Mazzucco,

pela confiança desmedida que concedeu a mim e ao meu trabalho; a Beth, com suas tantas palavras amigas, mas nem por isso menos duras, que me indicavam o caminho quando este parecia perdido; e ao Bosco, em sua doideira animadora, incentivador da finalização do trabalho.

Devo estar deixando de falar de muita gente, ainda mais ao assumir o risco de citar nomes. Mas quem fez parte desse caminhada sabe que não me esquecerei jamais de cada instante vivido até aqui. Meus sinceros agradecimentos a todos, com a certeza de que, se a coisa apertar, teremos um F e um C gravados como um código representativo de que as preocupações não devem superar o prazer das vitórias.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

O desenvolvimento de aplicações que se utilizam de recursos multimídia envolvem a necessidade de trabalhar a correta ordenação no tempo e no espaço das mídias. Esta ordenação, conhecida como sincronização multimídia, é objeto de estudo especialmente no que se refere ao desenvolvimento de sistemas para a Internet. Neste contexto, foram definidos padrões, como o HyTime, e linguagens, como o HTML+TIME e o SMIL, que buscam atender aos requisitos de sincronização multimídia para um sistema. Especial atenção é dada ao módulo de sincronização do HyTime, padrão que, embora não seja utilizado *de facto*, oferece um grande número de características que devem ser consideradas na proposta de um modelo de especificação multimídia. Este trabalho visa buscar as características de cada padrão e linguagem, relacioná-los com as definições encontradas na literatura, e a partir disto definir uma meta-linguagem que se propõe atender aos requisitos de sincronização como especificados no estudos teóricos. Assim, tem-se, ao fim deste, a especificação de meta-linguagens que, vistas em conjunto, oferecem os elementos necessários para que se construa uma aplicação corretamente sincronizada.

ABSTRACT

The development of applications that use multimedia resources involves working the correct ordinance in the time and the space of the media object. This ordinance, known as multimedia synchronization, is very important for the development of the Internet systems. In this context, standards had been defined, as HyTime, and languages, as HTML+TIME and SMIL. Special attention is given to the HyTime synchronization module that is a standard that offers a great number of characteristics that must be considered in the proposal of a multimedia model. This work analyses the characteristics of each standard and language, to relate them with the definitions of the references, and to define a metalanguage that considers the requirements of synchronization as specified in the theoretical studies. Thus, this work presents the specification of metalanguages that offer the necessary elements to construct a synchronized application correctly.

1 Introdução

A crescente utilização de várias mídias em sistemas computacionais tem dado origem a inúmeras pesquisas que buscam responder às necessidades decorrentes dessa utilização. A característica principal de sistemas multimídia é a necessidade de compor dados de vários tipos e origem para apresentação, armazenamento e comunicação. Sistemas multimídia devem armazenar, recuperar e comunicar representações complexas de informação sob condições de grande heterogeneidade (LITTLE, 1990).

Um sistema que se utiliza de duas ou mais mídias (áudio, imagem, animação, vídeo, texto) é, de forma geral, chamado de sistema multimídia. Conceituando, um sistema multimídia é um sistema ou aplicação que suporta o processamento integrado de vários tipos de mídia, com ao menos uma mídia contínua (BLAKOWSKI, 1996). Mídia contínua diz respeito a mídias como áudio, vídeo, animações, como será descrito no próximo capítulo.

Quando um sistema utiliza-se somente de texto em linguagem natural, mas permite que o usuário utilize-se dele de uma forma não-linear (diferente de um documento tradicional de texto), ele é então um hipertexto. Um hipertexto é, genericamente, uma base de dados caracterizada por pequenos fragmentos de texto interconectados por ligações (CONKLIN, 1987).

Quando, no sistema multimídia, são permitidas ligações entre quaisquer mídias como em um hipertexto são permitidas ligações entre porções de texto, tem-se um sistema hipermídia. Da mesma forma que um sistema que contém somente texto é considerado um documento, tem-se, em um sistema hipermídia, um hiperdocumento.

A capacidade de produzir e visualizar um hiperdocumento envolve a necessidade de se criar mecanismos que permitam que as diferentes mídias sejam corretamente compostas, armazenadas e recuperadas, respeitando as características específicas de cada mídia e como elas são modificadas quando trabalhadas em conjunto. Também devem ser considerados os recursos de processamento, armazenamento e comunicação disponíveis.

O desenvolvimento de sistemas que utilizam várias mídias, que serão trabalhadas em conjunto, deve considerar a necessidade de obter-se uma correta ordenação dos componentes multimídia, ou seja, obter sincronismo multimídia. Com a utilização de diferentes plataformas, deve-se considerar ainda como permitir que o sistema produzido possa ser recuperado em cada uma dessas plataformas.

O padrão *Hypermedia/Time-Based Structuring Language* (HyTime) (ISO, 1992) foi desenvolvido pela *International Organization for Standardization* (HyTime) com o objetivo de possibilitar a construção de hiperdocumentos que sejam utilizados em diferentes plataformas. O padrão HyTime estabelece que a definição do conteúdo de um documento seja realizada de forma independente da definição de como ele será apresentado. Entre suas características, no que se refere ao tratamento das mídias, o padrão HyTime apresenta meios de especificar escalonamento e sincronização de eventos no espaço e no tempo. Um *Minimal Scheduling HyTime document* (MSHd) é um subconjunto do padrão HyTime que possui os construtores para a manipulação de escalonamento e sincronização de mídias. Este subconjunto trabalha com as formas arquiteturais de três dos seis módulos do padrão, e possui as facilidades necessárias para um tratamento de sincronização e escalonamento relacionados à utilização de eixos temporais. Este trabalho apresenta um MSHd que estende as facilidades deste subconjunto, fornecendo ao padrão HyTime novas formas para especificação de sincronização entre mídias.

A linguagem *Synchronized Multimedia Integration Language* (SMIL) foi projetada para permitir a inclusão de elementos de mídia em páginas da Internet, com possibilidade de escalonamento e sincronização das mídias. Desenvolvida por um grupo de trabalho da *World Wide Web Consortium* (W3C), a partir da *eXtensible Markup Language* (XML), possui um conjunto de elementos voltados especificamente para a tarefa de sincronizar uma apresentação multimídia.

Criada a partir de uma separação do grupo de trabalho responsável pelo SMIL, a linguagem HTML+TIME (*Timed Interactive Multimedia Extensions*) também disponibiliza um conjunto de elementos que permitem o trabalho com escalonamento e sincronização de mídias para a Internet. Trabalhando exclusivamente com o tratamento das mídias, visto que a questão de *layout* é deixada sob responsabilidade da parte HTML da linguagem, o HTML+TIME apresenta características que o diferem do SMIL em alguns pontos, no que se refere à sincronização multimídia.

O trabalho aqui proposto segue um estudo das características do padrão HyTime no que se refere à sincronização multimídia, especialmente pelo fato do mesmo ter sido desenvolvido com este propósito específico. O trabalho tem continuidade com o estudo e a comparação deste padrão com as características das linguagens de sincronização SMIL e HTML+TIME, bem como a observação das deficiências destas últimas, que não atendem a todos os requisitos de sincronização da forma como o HyTime possibilita.

O não atendimento de todos os requisitos de sincronização a que estas linguagens se propõem representa um grave problema principalmente ao se pensar em implementações futuras de navegadores ou *players* voltadas para a apresentação de elementos multimídia corretamente sincronizados. Este trabalho objetiva a investigação de um modelo para hiperdocumentos conformantes com o subconjunto *Minimal Scheduling HyTime Documents*, do HyTime; e a especificação de uma meta-linguagem a partir desse subconjunto que atenda justamente aos requisitos de sincronização como descritos na literatura.

No próximo capítulo é apresentada uma visão geral sobre os mecanismos de sincronização, entre os quais destacam-se os métodos para especificação de sincronização; o capítulo 3 apresenta os padrões SGML, HyTime e XML, para intercâmbio de hiperdocumentos. Neste capítulo é dada especial atenção ao subconjunto *Minimal Scheduling HyTime documents*, pois este oferece as formas básicas seguidas na estruturação da meta-linguagem aqui proposta; o capítulo 3 descreve ainda as linguagens HTML+TIME e SMIL, com suas características específicas de sincronização multimídia. No capítulo 4 são apresentados os DTDs propostos para a meta-linguagem desenvolvida a partir das características descritas no padrão HyTime para permitir o trabalho com sincronização de eventos. A monografia encerra com as Considerações Finais e a apresentação das Referências utilizadas neste trabalho.

2 Multimídia: sincronização e sua especificação

O trabalho aqui descrito envolve a especificação e apresentação de hiperdocumentos através dos elementos definidos pelo padrão HyTime, especialmente no que se refere ao tratamento de sincronização e escalonamento das mídias utilizadas nestes hiperdocumentos. Esta especificação é realizada através da verificação de quais elementos descritos no padrão podem ser trabalhados em conformidade com os requisitos relacionados à sincronização multimídia.

MEYER (1993) apresenta um modelo de referência de sincronização baseado em camadas. Este modelo será descrito na seção 2.4, com especial atenção para a camada de especificação, em cujo contexto as linguagens de especificação aqui trabalhadas e a meta-linguagem definida neste trabalho se inserem. Os padrões de desenvolvimento de páginas para a *World Wide Web* (Web) determinam a construção de um hiperdocumento através da especificação de sua estrutura. Dessa forma, é possível relacioná-lo à última camada do modelo de referência, relativa à especificação de sincronização. Essa camada é explorada através da descrição de quatro métodos para especificação de sincronização, os quais são apresentados com o propósito de possibilitar que se acrescentem características por eles identificadas às especificações de que serão realizadas.

As seções que seguem apresentam características de sistemas multimídia que se tornam relevantes para que se entenda a necessidade de sincronização. São apresentados alguns conceitos relativos a sincronização multimídia e alguns aspectos referentes a Qualidade de Serviço (QoS) em sincronização. Também é descrito o modelo de referência de sincronização com detalhamento dos métodos para especificação de sincronização, relativos à última camada do modelo de referência.

2.1 Sincronização Multimídia

A característica principal de sistemas multimídia é a necessidade de compor dados de vários tipos e origens para apresentação, armazenamento e comunicação. Sistemas multimídia devem armazenar, recuperar e comunicar representações complexas de informação sob condições de grande heterogeneidade (LITTLE, 1990).

Os dados que compõem um sistema multimídia possuem características específicas relacionadas com cada tipo de mídia. Assim, um objeto de texto não pode receber, do sistema, tratamento idêntico ao fornecido para um objeto de áudio.

LITTLE (1990) apresenta uma classificação desses dados, segundo suas características, em três tipos: estáticos, dinâmicos e mistos. Estáticos são aqueles que consistem de texto e/ou imagens dispostos em um espaço. Dinâmicos são aqueles compostos por áudio e/ou imagens de vídeo e/ou animações, de forma geral. Dados mistos são aqueles que têm elementos dos dois primeiros tipos em sua composição, unidos em uma única apresentação. Um dado composto de texto, imagem, áudio e animação, entre outros elementos que compõem os dados apresentados acima, serão referenciados aqui como objetos de mídia, ou simplesmente, objetos.

BLAKOWSKI (1996) descreve um conceito para sistemas multimídia que se baseia em três critérios que classificam um sistema como sendo, ou não, multimídia: o número de mídias, os tipos de mídias que são suportados e o grau de integração entre as mídias.

Pelo critério do número de mídias, um documento que contenha somente texto e figuras pode ser considerado um sistema multimídia, apesar de não ser essa a idéia que, em geral, tem-se de sistema multimídia. Usualmente considera-se um sistema multimídia aquele sistema que permite interação do usuário e utiliza mídias como som e vídeo. Necessita-se, então, agregar os outros dois critérios para que o conceito de sistema multimídia englobe esta consideração.

Para entender o critério que considera os tipos de mídias utilizados deve-se primeiro fazer a diferenciação entre mídias independentes de tempo (as chamadas mídias discretas) e as mídias dependentes de tempo (mídias contínuas). Um objeto de mídia discreta normalmente é apresentado utilizando somente uma unidade de

apresentação, enquanto objetos de mídia contínua utilizam-se de uma seqüência de unidades de apresentação. As unidades de apresentação citadas, para objetos de mídia contínua, são apresentadas em (BLAKOWSKI, 1996) como *Logical Data Units* (LDU's), ou seja, as seqüências de unidades de informação que constituem estes objetos, como os quadros (*frames*) de uma imagem de vídeo, por exemplo.

O critério que considera os tipos de mídia indica que um sistema multimídia deve suportar o processamento de mais de uma mídia com pelo menos uma mídia contínua. O critério do grau de integração representa o quanto diferentes tipos de mídia permanecem independentes, ainda que possam ser processadas em conjunto, como um só dado - que seria o documento multimídia como um todo.

A combinação desses critérios leva à definição de sistema multimídia como um sistema ou aplicação que suporta o processamento integrado de vários tipos de mídia, com ao menos uma mídia contínua (BLAKOWSKI, 1996). Para o trabalho com hiperdocumentos deve-se acrescentar a esse conceito a idéia de interatividade, ou seja, a possibilidade do usuário interagir com o sistema, através da ativação de áreas específicas em figuras, ou determinadas palavras ou expressões em um texto, a fim de acessar outras áreas de dados no sistema.

Com a utilização de mídia contínua na composição dos hiperdocumentos multimídia passa-se a necessitar de uma correta ordenação no tempo de cada um dos seus objetos componentes. A tarefa de coordenar esse tipo de apresentação é denominada sincronização multimídia (BUFORD, 1994).

Sincronização multimídia, numa visão geral, engloba relações de espaço, de tempo e de composição (ou configuracional) (VAZIRGIANNIS, 1993) (VAZIRGIANNIS, 1995). O relacionamento espacial dos objetos é genericamente expresso pela localização dos mesmos em um conjunto de coordenadas no espaço. O relacionamento de composição indica as conexões entre objetos inter-relacionados e o efeito que alterações em um objeto têm sobre o outro. Por fim, as relações de tempo são aquelas que definem as dependências temporais entre os objetos de mídia. É sob o enfoque das relações temporais que o termo sincronização multimídia será utilizado neste trabalho.

2.1.1 Sincronização intra e inter-objetos

A sincronização multimídia depende da especificação das relações temporais entre os objetos de mídias, que podem ser implícitas, como no caso da aquisição simultânea de voz e vídeo, ou podem ser formuladas explicitamente, como no caso de documentos multimídia compostos de texto e voz (STEINMETZ, 1995).

No caso das relações temporais implícitas, tem-se a sincronização intra-objeto, que se refere à relação de tempo entre várias unidades de apresentação de um objeto de mídia contínua. As relações formuladas explicitamente são conhecidas como sincronização inter-objetos, realizada entre diferentes objetos de mídia.

Em ambos os casos, as características de cada mídia e os relacionamentos entre elas devem ser estabelecidos a fim de prover uma apresentação corretamente sincronizada. A interatividade com o usuário também deve ser considerada e as relações temporais entre os objetos de mídia e as interações devem ser especificadas.

Os requisitos para que se tenha uma apresentação corretamente sincronizada compreendem, para sincronização intra-objeto, a precisão quanto aos atrasos na apresentação das LDU’s, evitando qualquer *jitter* na apresentação de LDU’s consecutivos; e para sincronização inter-objetos, a precisão na apresentação, em paralelo, dos objetos de mídia.

Atraso (*delay*) diz respeito ao tempo que o sistema deve esperar pela entrega uma seqüência de LDU's no local de destino, ou seja, onde esta seqüência será apresentada. Entre os atrasos que devem ser considerados estão os decorrentes da tarefa de processamento das LDU's, como compressão e criptografia.

Jitter diz respeito às falhas, como a inserção ou remoção de espaços na apresentação de uma seqüência de LDU's, decorrentes da variação no atraso da entrega de LDU's.

De forma geral, para sincronização multimídia, os requisitos para se obter uma apresentação sincronizada corretamente compreendem valores que determinam, para cada mídia, a qualidade de serviço esperada, como se apresenta a seguir.

2.2 Qualidade de Serviço

Dado que uma grande quantidade dos sistemas multimídia são sensíveis a atrasos (*delays*), os serviços relativos a esses sistemas (armazenamento, recuperação e comunicação) devem prover garantias de temporização (NAHRSTEDT, 1995).

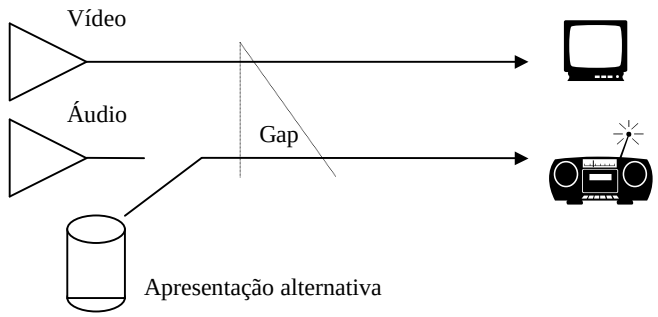


Figura 1: O problema do gap (BLAKOWSKI, 1996)

Determinadas situações são esperadas, principalmente no que se refere a *gaps*, que são falhas como, por exemplo, atraso, ou perda de quadros, na apresentação de seqüências de vídeo, enquanto o áudio prossegue normalmente, como representado na Figura 1.

Esses *gaps*, entretanto, podem ser tratados, e é esse tratamento que possibilita prover uma apresentação perfeitamente sincronizada. Alguns sistemas buscam resolver o problema com a apresentação de uma tela em branco ou em preto durante o *gap*, a utilização de apresentações alternativas, ou ainda a reapresentação de alguma seqüência já utilizada. Em situações como essa, ao buscar-se uma solução, deve-se levar em consideração o fator tempo. A utilização de uma ou outra solução deve verificar o tempo previsto para a falha, e o quanto de qualidade é esperado para o sistema.

O termo qualidade, mais especificamente Qualidade de Serviço (QoS), é bastante intuitivo para o usuário de aplicações multimídia: expressa basicamente quão bons estão os serviços fornecidos pelo sistema. Em geral, a partir do usuário da aplicação, QoS pode ser descrita em termos dos seguintes parâmetros (parâmetros de QoS do usuário):

- resolução: caracteriza a precisão do processo de digitalização de um *stream* e é descrita como uma função de bits por segmento e razão de amostragem;
- distorção: mede o nível de perda de informação de um *stream* e é função de erros de transmissão e perdas devido às estratégias de compressão, por exemplo;
- nível de sincronização: é uma medida de quão estável está a apresentação dos segmentos de um *stream* (sincronização intramídia) e entre *streams* relacionados (sincronização intermídia);
- interatividade: é uma medida de quanto um *stream* pode ser usado para trabalho cooperativo on-line.

Os requisitos relativos a qualidade de serviço, no que tange a requisitos de sincronização, podem ser expressos através de especificações QoS. Essas especificações dependem da mídia e da aplicação. As especificações QoS para um objeto de mídia contínua devem incluir a qualidade referente a precisão com que as relações temporais entre as LDU's desse objeto devem ser atendidas.

Quando há relação entre duas mídias, os parâmetros QoS devem definir o desvio aceitável entre seqüências de dados relacionadas, como, por exemplo, entre a apresentação da imagem de um locutor e o áudio relativo a essa locução. Os valores que esses parâmetros devem assumir varia de acordo com a tecnologia utilizada e com os meios de comunicação disponíveis, entre outros fatores. A definição desses valores deve levar em consideração esses fatores e a qualidade desejada para a apresentação.

Usualmente, a aplicação trabalha com opções finitas para o conjunto de parâmetros de QoS e interage com o usuário para escolhê-las. É comum que se trabalhe com descrições subjetivas como "qualidade de definição de áudio de CD" e "qualidade de definição de vídeo pobre", por exemplo.

A partir do ponto de vista do sistema, o conjunto de parâmetros de QoS é bem estabelecido (parâmetros de QoS do sistema), sendo composto por :

- atraso fim-a-fim: é o tempo transcorrido desde a captura ou ativação em uma base de dados de um *stream* até a sua apresentação;
- *jitter* fim-a-fim: é a variação do atraso fim-a-fim;

- confiabilidade - razão de pacotes perdidos (PER): é a porcentagem de pacotes descartados durante a transmissão;

- razão de bits errados (BER): é a porcentagem de bits errados devido ao processo de transmissão;

- vazão: a taxa máxima que alguma aplicação ou protocolo consegue manter entre dois pontos em uma rede.

De maneira geral, o atraso fim-a-fim afeta a interatividade da aplicação, o *jitter* fim-a-fim afeta o nível de sincronização e BER/PER têm impacto no grau de distorção. Quanto menores forem os valores desses parâmetros maiores serão os índices de qualidade de serviço da aplicação.

2.3 Modelo de referência

A representação de relações temporais entre objetos de mídia está baseada em modelos de tempo. Estes modelos tornam possíveis a identificação e a especificação de relações temporais entre diferentes mídias, e em particular as relevantes ao processo de sincronização multimídia.

MEYER(1996) apresenta um modelo de referência de quatro camadas, como apresentado na Figura 2, útil para entender os requisitos e os mecanismos que devem oferecer suporte a execução da sincronização. Cada camada implementa mecanismos de sincronização que são dotados de interfaces apropriadas para especificar e reforçar relacionamentos temporais. Cada interface pode ser usada pela aplicação diretamente ou pela camada superior. Camadas superiores apresentam abstrações de programação e de QoS mais elevadas.

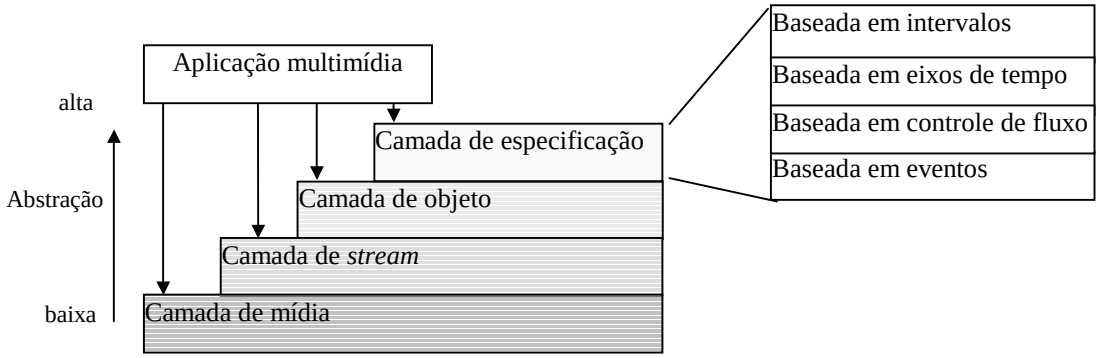


Figura 2: Modelo de referência para sincronização (MEYER, 1996) - modificado

2.3.1 Camada de mídia

Nesta camada, uma aplicação opera em uma única seqüência de mídia contínua, que é tratada como uma seqüência de LDU's. A abstração fornecida nesta camada é uma interface independente de dispositivo, com operações como *read(deviceHandle, LDU)* e *write(deviceHandle, LDU).*

2.3.2 Camada de *stream*

Esta camada trabalha tanto com uma seqüência de mídia contínua, como com grupos de seqüências. Em um grupo, todas as seqüências são apresentadas em paralelo utilizando mecanismos de sincronização *intra-stream*. A abstração oferecida nesta camada é a noção de *streams* com parâmetros de tempo relativos a QoS para sincronização *intra-stream* em um *stream* e sincronização *inter-stream* entre *streams* de um grupo (BLAKOWSKI, 1996). Exemplos de operações desta camada são: *start(stream), stop(stream), creat_group(list_of_streams), start (group)* e *stop(group).*

2.3.3 Camada de objeto

Esta camada trabalha com todos os tipos de mídias e oculta as diferenças entre mídias contínuas e discretas. A abstração aqui oferecida é a de uma apresentação completamente sincronizada (BLAKOWSKI, 1996). Esta camada recebe uma

especificação de sincronização como entrada e é responsável pelo correto escalonamento de toda a apresentação. As funções dessa camada são calcular e executar escalas de apresentação completas que incluem a apresentação de objetos de mídia não-contínuos e as chamadas para a camada de *stream*.

A camada de objeto é responsável por iniciar ações preparativas que são necessárias para permitir uma apresentação sincronizada. Esta camada não manipula as sincronizações *inter* e *intra-stream*; para isso ela se utiliza dos serviços da camada de *stream*. A tarefa da aplicação que utiliza a camada de objetos é fornecer uma especificação de sincronização (BLAKOWSKI, 1996).

2.3.4 Camada de especificação.

Esta é uma camada aberta e não oferece uma interface explícita. Esta camada contém aplicações e ferramentas que permitem criar especificações de sincronização, e é responsável também por mapear requisitos de QoS do nível do usuário para as qualidades oferecidas na interface da camada de objeto. Os métodos de especificação podem ser classificados nas seguintes categorias (STEINMETZ, 1995):

- especificações baseadas em intervalos, que permitem a especificação de relações temporais entre intervalos de tempo das apresentações de objetos de mídia;
- especificações baseadas em eixo, que relaciona eventos de apresentação a eixos que são compartilhados pelos objetos da apresentação;
- especificações baseadas no controle de fluxo, onde o fluxo das apresentações é sincronizado em pontos de sincronização fornecidos;
- especificações baseadas em eventos, onde os eventos na apresentação da mídia disparam ações da apresentação.

Estes métodos de especificação serão descritos na próxima seção, com especial atenção para o método baseado em eixos (utilizado pelo HyTime, e pelas linguagens HTML+TIME e SMIL) e para o método baseado em intervalos (utilizado pelas

linguagens HTML+TIME e SMIL), que serão mapeados para tornar possível sua especificação através de uma meta-linguagem.

2.4 Métodos de especificação de sincronização

Os métodos para especificação de sincronização são apresentados abaixo com o propósito de demonstrar algumas de suas características. Especial atenção é fornecida às especificações baseada em intervalos e baseadas em eixos. Esta última é utilizada pelo HyTime para sincronização dos objetos de mídia, e deseja-se permitir que o HyTime possibilite especificar sincronização através de intervalos.

2.4.1 Especificação baseada em intervalos

Na especificação de sincronização baseada em intervalo, a duração da apresentação de um objeto é chamada intervalo (LITTLE, 1993). Dois intervalos de tempo podem ser sincronizados em treze diferentes formas, as quais podem ser descritas por apenas sete, uma vez que seis dessas relações são o inverso de outras, como, por exemplo, *A before B* é a relação inversa de *B after A* (ALLEN, 1983). A Figura 2.3 apresenta essas relações.

WAHL (1994), com base nas treze relações temporais binárias básicas identificadas por ALLEN (1983) (Figura 3), propõe o modelo avançado de sincronização baseado em intervalos no qual 29 relações entre intervalos foram identificadas para apresentações multimídia. A partir dessas relações, foram definidos dez operadores para a especificação de relações entre os intervalos, apresentados na Figura 4 .

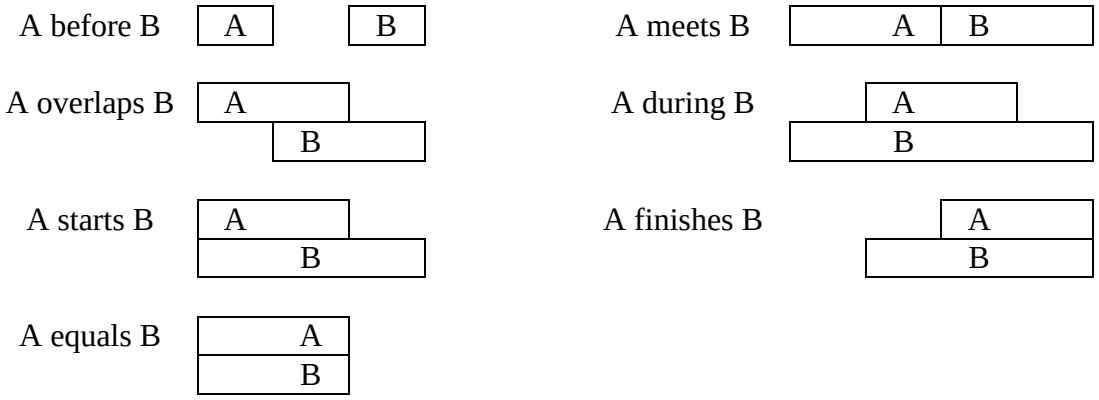


Figura 3: Relações temporais binárias (ALLEN, 1983)

Este modelo permite a definição da duração para objetos de mídia contínua e discreta. A vantagem deste modelo está em permitir a manipulação de LDU's , além de tornar possível a definição da duração de uma interação com o usuário (um dos pontos fortes deste modelo). Além disso, é possível especificar relações temporais não-determinísticas adicionais através da definição de intervalos por duração e por atraso. Uma outra característica é a possibilidade de se fazer disjunções de operadores. A disjunção de operadores pode ser usada para especificar relações como *não paralelo*, por exemplo. Este modelo é bastante flexível e permite a especificação de apresentações sujeitas a diversas variações em tempo de execução (STEINMETZ, 1995).

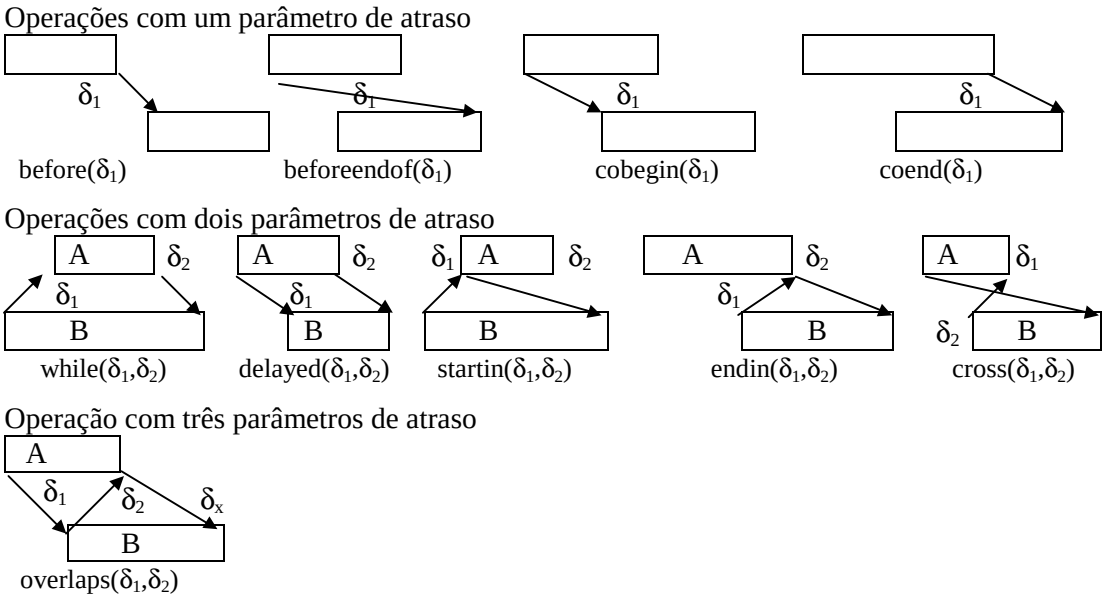


Figura 4: Operações do modelo avançado de sincronização baseado em intervalos (WAHL, 1994)

Por outro lado, este modelo não inclui especificações de desvio: apesar da especificação direta de relações de tempo entre objetos de mídia, ele não permite a especificação de relações temporais diretamente entre sub-unidades de objetos. Tais relações devem ser definidas indiretamente através de especificações de atrasos.

2.4.2 Especificação baseada em eixos

Neste modelo, os eventos de apresentação, como o início e o fim de uma apresentação, são mapeados para eixos que são compartilhados pelos objetos.

2.4.2.1 Sincronização baseada em um *timer* global

Através deste modelo, todos os objetos de uma só mídia são ligados a um eixo de tempo, que representa uma abstração do tempo real (STEINMETZ, 1995). Neste modelo, remover um objeto não altera a sincronização dos demais objetos.

A sincronização de objetos através da utilização de um eixo de tempo permite uma boa abstração da estrutura interna dos objetos de uma única mídia, e objetos multimídia aninhados. Por outro lado, a utilização de um *timer* global pode não ser suficiente para expressar as relações de sincronização entre diferentes seqüências de apresentação. Dependendo dessas seqüências, essa sincronização pode ser muito forte ou muito fraca. Uma solução possível pode ser a utilização de um parâmetro QoS para cada par de seqüências de mídias (BLAKOWSKI, 1996).

Na Figura 5 é representada a especificação de uma aplicação, com a observação de que não é possível manipular a duração de uma interação com o usuário. Uma vez que o escalonamento é feito em relação a um eixo de tempo global, e todas os objetos são localizados em determinadas posições deste eixo, o desconhecimento da duração de algum objeto (no caso, a interação com o usuário) impede a determinação do posicionamento dos objetos que o seguem.

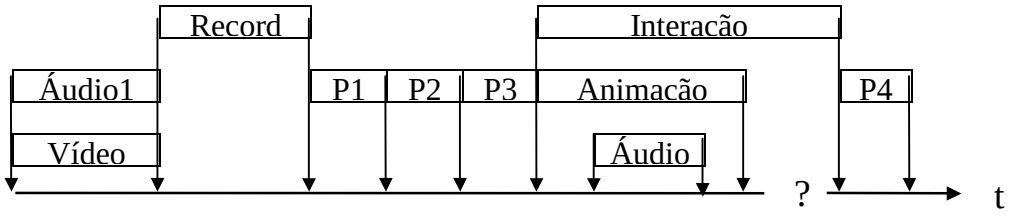


Figura 5: Especificação de sincronização baseada em um *timer* global (BLAKOWSKI, 1996)

2.4.2.2 Sincronização baseada em eixos virtuais

Este modelo, onde é possível especificar um sistema de coordenadas com unidades de medidas definidas pelo usuário, é utilizado pelo padrão HyTime (ISO, 1992). Toda a especificação de sincronização é realizada sobre esses eixos virtuais. O modelo permite que sejam definidos diferentes eixos, criando um espaço de coordenadas virtual (BLAKOWSKI, 1996). Com isso, pode-se utilizar eixos específicos para tratar a interação com o usuário, bem como um eixo de tempo tradicional (aqui entendido como eixo virtual), permitindo que o exemplo da Figura 5 seja modificado, para tratar a interação, como apresentado na Figura 6.

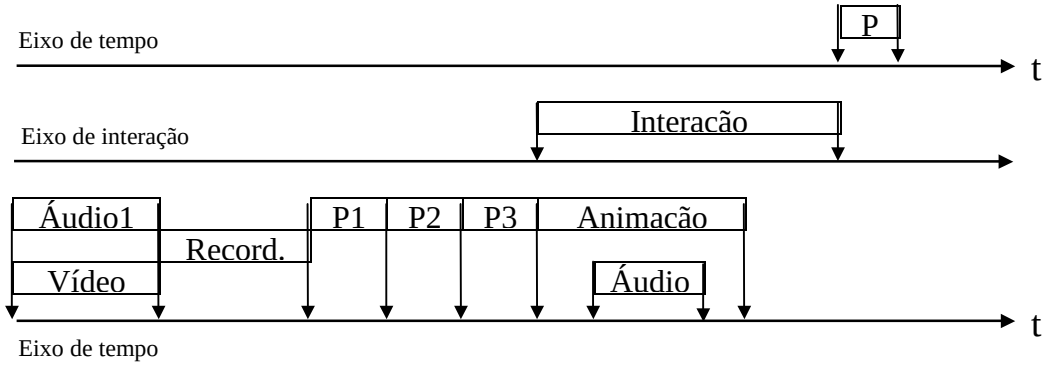


Figura 6: Especificação de sincronização baseada em eixos virtuais (BLAKOWSKI, 1996)

2.4.3 Especificação baseada em controle de fluxo

O fluxo dos caminhos concorrentes da apresentação são sincronizados em pontos pré-definidos, que podem seguir uma hierarquia básica, seguir pontos de referência ou ainda serem trabalhados através de uma Rede de Petri temporizada.

2.4.3.1 Especificação de hierarquia básica

Aqui as descrições são baseadas em duas operações de sincronização: sincronização serial e sincronização paralela de ações (STEINMETZ, 1995).

Nesta especificação, os objetos multimídia são trabalhados como uma árvore com nós que denotam a apresentação serial ou paralela das sub-árvores filhas.

Um ação pode ser atômica ou composta. Uma ação atômica manipula a apresentação de um objeto de mídia única, uma entrada do usuário ou um atraso. Ações compostas são a combinação de operadores de sincronização e ações atômicas.

As estruturas hierárquicas são fáceis de manipular. Essa estrutura, no entanto, faz com que cada ação seja sincronizada somente em seu início ou fim. Para obter sincronização em uma parte de uma animação, por exemplo, que não seja seu início ou fim, deve-se dividir a animação em partes que, essas sim, serão dispostas na árvore. Caso a animação tenha sido obtida como uma unidade indivisível, essa forma de sincronização não poderá ser obtida. Ou seja, não há suporte para abstração da estrutura interna dos objetos multimídia (STEINMETZ, 1995).

2.4.3.2 Pontos de referência

Neste modelo, objetos de mídia contínua são trabalhados como seqüências de LDU’s (BLAKOWSKI, 1996). O tempo inicial e final da apresentação de um objeto de mídia, bem como o tempo inicial das sub-unidades dos objetos de mídia contínua, são os chamados pontos de referência. A sincronização é obtida através da conexão dos pontos de referência dos objetos de mídia, como pode-se observar na Figura 7.

Esse modelo permite a sincronização em qualquer ponto, durante a apresentação de um objeto, da mesma forma que a sincronização baseada em eixo de tempo, com a vantagem de permitir a inserção de objetos com duração desconhecida, pois a condição

para que um objeto de mídia seja ativado é que o objeto anterior a ele “chegue” ao ponto de sincronização, independente da sua duração. Além de possibilitar também a integração com objetos de mídia discreta.

A detecção de inconsistências, como atrasos, neste modelo, requer a utilização de mecanismos, como, por exemplo, um *timer* global, sobre o qual os objetos podem ser escalonados. Dessa forma, a sincronização com *timer* global pode ser utilizado como um subconjunto de sincronização com pontos de referência.

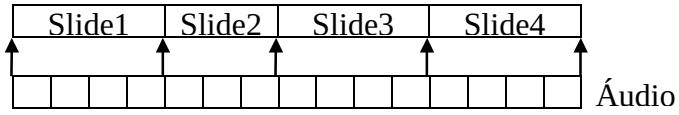


Figura 7: Sincronização através de pontos de referência
(BLAKOWSKI, 1996)

2.4.3.3 Redes de Petri Temporizadas

As Redes de Petri consistem de quatro elementos básicos: *tokens*, lugares, transições e arcos e um conjunto de regras que governam a operação da rede como um todo. A operação das Redes de Petri é baseada na visão de *tokens* movendo-se em uma rede abstrata (MURATA, 1989).

Tokens são entidades conceituais, que modelam os objetos que se movem por uma rede real. Lugares representam as localizações onde objetos esperam por processamento. Transições representam processos, eventos ou atividades. Cada transição em uma Rede de Petri tem um número de lugares de entrada e de saída; esses lugares podem representar uma pré-condição ou uma pós-condição de um evento, ou recursos requeridos e liberados por um evento. Arcos representam os caminhos dos objetos através do sistema, conectando lugares às transições e transições a lugares (a direção do caminho é indicada por uma seta no final do arco).

As regras de disparo de uma Rede de Petri especificam o comportamento das transições, isto é, as condições sob as quais processos ou eventos podem ocorrer. Três regras governam o disparo das transições (MURATA, 1989):

1. quando todos os lugares anteriores às transições são ocupados pelo menos por um *token*, a transição está habilitada;

2. uma vez habilitada, a transição dispara;

3. quando a transição disparar, exatamente um *token* é removido de cada lugar anterior e exatamente um *token* é colocado em cada lugar posterior às transições.

As Redes de Petri foram estendidas de várias formas, buscando permitir a especificação de requisitos relativos à estruturação da informação, intervalos de apresentação e sincronização entre cadeias, além da especificação de certos tipos de ligações temporais.

Estas variações foram utilizadas como ponto de partida para a especificação de apresentações multimídia e hiperdocumentos. Assim, foram propostos vários modelos, como o OCPN - *Object Composition Petri Net* (LITTLE 1990a), o TSPN - *Time Stream Petri Net* (DIAZ 93] e o HTSPN - *Hierarchical Time Stream Petri Net* (SÉNAC, 1995).

OCPN

LITTLE (1990) propõe o modelo OCPN (*Object Composition Petri Net*), para especificação de requisitos de sincronização em documentos multimídia. O modelo OCPN tem como objetivo especificar formalmente os requisitos de sincronização de documentos multimídia, permitindo a representação da duração de recursos como sendo lugares da rede. As regras de disparo foram adaptadas para permitir que lugares possuam estados (ativos ou inativos).

1. uma transição dispara quando cada um dos lugares de entrada contiverem um *token*;

2. após o disparo, os *tokens* serão removidos dos lugares de entrada e adicionados aos lugares de saída;

3. após receber um *token*, o lugar permanece no estado ativo pelo intervalo de tempo especificado. Durante esse intervalo, o *token* estará bloqueado até que o lugar se torne inativo, ou pela expiração da duração do intervalo, então o *token* será desbloqueado.

TSPN

O modelo OCPN, deve ser estendido para permitir a especificação das estratégias de sincronização entre as mídias (ou cadeias - explicado adiante). O modelo TSPN (*Time Stream Petri Net*) (DIAZ, 1993) realiza esta extensão, através da utilização de tipos de transições para cada estratégia. A sincronização pode ser obtida através de nove tipos de transições, segundo o modelo: "*strong_or*", "*weak_and*", "*or*", "*and*", "*strong_master*", "*weak_master*", "*or_master*", "*and_master*".

A estratégia *strong_or* especifica que a apresentação de uma das mídias (ou cadeias) é suficiente para que a apresentação seja considerada completa, desde que as demais mídias tenham sido apresentadas por um tempo mínimo especificado nos respectivos arcos. Neste modelo há essa especificação de um tempo mínimo (e também de um tempo máximo) pois ele trabalha com a especificação dos requisitos de sincronização em uma granularidade mais fina, permitindo que possam ser realizadas especificações entre as cadeias de LDU's. Os requisitos de sincronização são representados, então, através da especificação de uma tupla de valores para os arcos que vão dos lugares para as transições. Esta tupla $[\alpha,\eta,\beta]$ representa, respectivamente, o tempo mais adiantado de liberação do *token* para o disparo da transição, a duração ideal da apresentação da mídia, e o tempo mais atrasado de liberação do *token*.

A estratégia *weak_and* só é considerada completa quando as apresentações de todas as cadeias são realizadas; enquanto a *master* necessita que a somente a cadeia *master* esteja completamente apresentada para que o *token* seja liberado. A estratégia or indica que a primeira cadeia ou mídia apresentada dispare a transição, enquanto que a and exige que todas as cadeias tenham sido apresentadas ou tenham atingido seu tempo máximo, indicado no rótulo. Através destas estratégias, as redes tornam-se habilitadas a modelar os modelos de especificação baseados em intervalos, apresentados na seção 2.4.1.

HTSPN

O modelo HTSPN (*Hierarchical Time Stream Petri Net*) (SÉNAC, 1995) permite a modelagem formal, com extensões temporais, dos três principais conceitos do modelo Dexter (HALASZ, 1994): os componentes atômicos do modelo Dexter representam os dados codificados; as ligações permitem a definição das relações entre e no interior dos

componentes; e um componente composto fornece mecanismos de estruturação hierárquico. Estes conceitos são relevantes pela possibilidade de ser possível, através deles, especificar relações entre mídias, com a utilização ou não de *links* (interação com o usuário), em um hiperdocumento. Estes conceitos e sua utilização em Redes de Petri ficam claros quando WILLRICH (1997) descreve como o modelo HTSPN suporta cada um destes conceitos, respectivamente:

- um componente atômico é modelado por um arco com um intervalo de validade temporal (IVT) e um lugar do tipo atômico associado a um recurso. O IVT é uma tupla (α,η,β) onde α,η e β especificam, respectivamente, a duração mínima, ideal e máxima admissível do tratamento associado ao componente atômico para o lugar Vídeo1;

- uma ligação é modelada por um arco temporizado (L, t), onde L, na figura 8, é um lugar do tipo ligação, como exemplificado pelo lugar L;

- um componente composto é modelado por um lugar abstrato, chamado lugar composto, que representa uma sub-rede. O lugar C, na figura 8, é um exemplo de lugar composto.

A Figura 8 apresenta um exemplo de rede de Petri utilizada para especificação de sincronização multimídia. O exemplo é uma amostra de como o modelo HTSPN pode trabalhar com os conceitos do modelo Dexter.

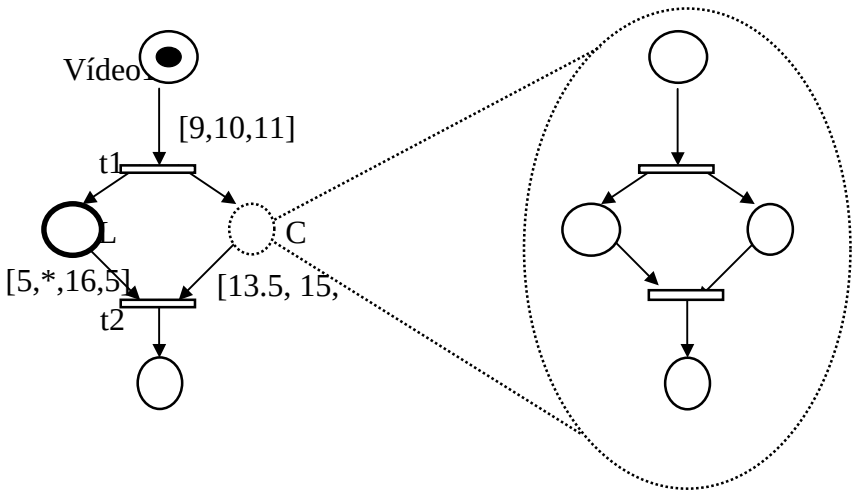


Figura 8 Modelagem dos componentes do modelo Dexter (WILLRICH, 1997) - resumido

A representação gráfica, juntamente com uma fácil modelagem de esquemas de sincronização e a possibilidade de analisar propriedades importantes fazem das Redes de Petri boas candidatas para modelagem de documentos multimídia e hipermídia (WILLRICH, 1997).

2.4.4 Especificação baseada em eventos

Em RODRIGUES (1997) um evento é definido através da apresentação de um conjunto marcado de unidades de informação de um nó (evento de apresentação) ou pela sua seleção (evento de seleção) ou pela alteração de um atributo de um nó (evento de atribuição).

Um evento pode estar em um dos estados: *sleeping*, *preparing*, *prepared*, *occurring*, e *paused*, e cada evento tem um atributo associado no nó onde ele é definido, chamado, *ocurred*, que tem a função de contabilizar quantas vezes um evento transita dos estados *ocurring* para *prepared* durante a apresentação do documento (RODRIGUES, 1997).

A Figura 9 apresenta um exemplo de um evento de apresentação. Ela inicia no estado *sleeping*, passa para o estado *preparing*, que é onde são executados procedimentos sobre suas unidades de informação. Após esses procedimentos serem executados, o evento passa para o estado *prepared*. Quando do início da exibição das unidades de informação ele fica no estado *occurring*, que pode passar para o estado *paused* caso ocorra alguma suspensão temporária na exibição. Ao fim da exibição, o evento retorna ao estado *prepared*, quando o atributo *occurred* é incrementado.

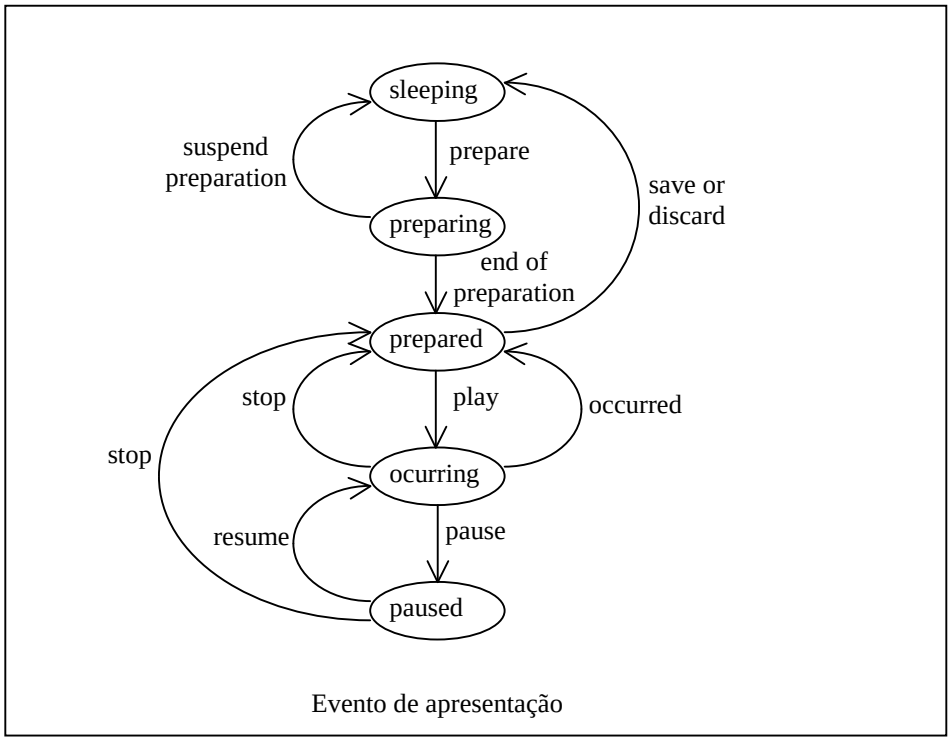


Figura 9 Máquina de estados de evento (RODRIGUES, 1997) - resumido

2.5 Considerações

Os modelos de especificação de sincronização descritos apresentam diferentes visões do trabalho de sincronização. A utilização de um destes modelos depende da estrutura da aplicação a que se destina a especificação.

De forma geral, os modelos são de fácil compreensão. Os modelos baseados em relações temporais oferecem condições de relacionar diferentes mídias através da especificação dos intervalos que devem existir entre elas (ou entre pontos delas, como seu início e seu término). Os modelos baseados em eixos de tempo apresentam a conveniência de poder localizar a mídia em um eixo temporal semelhante ao eixo de tempo real. A utilização de eixos virtuais mantém essa conveniência, com a vantagem de permitir a especificação de interações com o usuário. Esta é também uma das vantagens do modelo baseado em eventos, pois a interação pode ser um dos processos a serem executados para permitir a transição de um estado a outro da apresentação. A

utilização de controle de fluxo, mais especificamente Redes de Petri segundo o modelo HTSPN, apresenta a possibilidade de se valer das características deste modelo que já foram definidas para o trabalho com sistemas multimídia.

De forma geral, os métodos baseados em intervalos, em eixos de tempo, e em pontos de referência apresentam uma boa abstração do conteúdo das mídias, pois estas são especificadas como seqüências de dados, não ficando dispersas em nós, como no modelo de especificação hierárquica, ou baseado em Redes de Petri.

Estes modelos foram utilizados como base para uma extensão dos elementos definidos no HyTime para o trabalho com sincronização de mídias. Estes elementos e suas extensões são descritos no próximo capítulo.

3 Padrões de intercâmbio de hiperdocumentos

A expansão do universo de sistemas computacionais que se utilizam de dados multimídia, o crescimento dos meios de comunicação entre variadas plataformas e sistemas e o surgimento de novas tecnologias, entre outras variáveis, demandam o estabelecimento de padrões que permitam que os sistemas originados desse processo evolutivo possam realizar uma perfeita interação entre si.

Interação entre sistemas diz respeito à possibilidade de interoperabilidade, intercâmbio de dados multimídia e também reusabilidade de código entre aplicações. Para possibilitar que haja essa interação, alguns requisitos devem ser cumpridos (SOARES, 1993):

1. a arquitetura do sistema deve ser aberta quanto a dois aspectos distintos:

    - aberta quanto a interconexão: implicando que objetos intercambiáveis devam seguir um padrão internacional, permitindo a interoperabilidade entre sistemas;

    - aberta quanto a sua estrutura: implicando a necessidade de apresentação de interfaces bem definidas, correspondente aos diversos níveis de serviços - o que possibilitaria uma reusabilidade de código máxima entre as aplicações;

2. aplicações devem manipular informação multimídia de forma a garantir interatividade e intercâmbio em tempo real;

3. a arquitetura deve ser possível de ser utilizada em um ambiente distribuído.

Nesse sentido, têm sido desenvolvidos padrões, especialmente pela ISO, que procuram permitir o intercâmbio de informações entre sistemas desenvolvidos sob plataformas ou sistemas diferentes.

3.1 Standard Generalized Markup Language

O SGML (*Standard Generalized Markup Language*) é um padrão internacional desenvolvido para a representação e intercâmbio de documentos (ISO, 1986) e permite que documentos armazenados eletronicamente sejam definidos conforme seu conteúdo e sua estrutura (BROWN, 1996), independente de sua forma de apresentação.

De forma geral, SGML é uma linguagem para descrição da estrutura lógica de um documento utilizando-se de marcações (*markup*). Assim, um documento SGML é uma seqüência de caracteres, em formato legível, consistindo do texto do documento intercalado com comandos de marcação que identificam o início e o fim de cada item lógico (GOLDFARD, 1995). Item lógico, em um documento SGML, é reconhecido como um elemento SGML.

Cada documento SGML utiliza uma definição de tipo de documento (*Document Type Definition* - DTD) que declara que tipos de elementos podem existir no documento, que atributos cada um desses tipos de elementos pode ter, e como instâncias destes tipos de elementos podem ser relacionadas hierarquicamente. Um DTD é, genericamente, uma linguagem específica, que define uma classe de documentos de modo que várias instâncias de documentos podem compartilhar um DTD (RUTLEDGE, 1993). O SGML pode definir um conjunto ilimitado de DTDs, por exemplo, um para memorandos, um para livros, etc. (HERWIJNEN, 1994).

A estrutura que o SGML provê é uma linguagem de marcação genérica que define um modelo de documento hierárquico.

3.1.1 SGML *parser*

O SGML *parser* é um componente de um ambiente SGML que processa sintaticamente o texto de um documento SGML e fornece informações para o resto do ambiente (HERWIJNEN, 1994). O SGML *parser* assegura, através do reconhecimento e da validação das marcações, que um documento SGML tenha todas as suas marcações livres de erros, sendo assim consistente e possível de ser interpretado

corretamente (CLARK, 1996). A utilidade do *parser* está em auxiliar na prevenção de erros e da má utilização dos marcadores. De forma geral, o SGML *parser* verifica:

- se um DTD de um documento é conformante com o SGML;
- se a instância do documento é conformante ao DTD.

O *HyperText Markup Language* (HTML), utilizado como linguagem específica para a criação de páginas para a WWW (*World Wide Web*), é especificado com um DTD SGML (RUTLEDGE, 1996). O DTD HTML define um conjunto particular de *tags* de marcação que podem ser intercalados com o conteúdo de um documento texto para delimitar e estruturar seu conteúdo. HTML provê a possibilidade de inclusão de objetos de documento de mídia não-texto.

3.1.2 Document Type Definition

Um *Document Type Definition* (DTD) é uma linguagem específica que determina como poderá ser elaborado um documento associado a ele. A declaração de um DTD determina (BROWN, 1989):

- que tipos de elementos podem existir em um documento; por exemplo, um livro pode conter capítulos, seções, subseções, etc.;
- que atributos esses elementos podem ter; por exemplo, o número de versão e data da última revisão; e
- como as instâncias desses elementos estão hierarquicamente relacionadas; por exemplo: um livro contém capítulos, que contêm seções, que podem conter subseções, etc.

Por definição, o DTD descreve somente a estrutura lógica de um documento. O formato como o documento será apresentado é realizado através da aplicação que poderá ser definido através de *style sheets*, que armazenam informações relativas à forma como deverá ser realizada a apresentação.

```
1   <!-- DTD simplificado para uma dissertação                                    -->
2   <!ENTITY % frontpg “TITLE,AUTHOR,DIRECTOR” -- capa da dissertação            -->
3   <!ENTITY    MSHd    “Minimal Scheduling HyTime documents”                     >
4   <!ENTITY    dir     SYSTEM  “/home/CG/fagundes/index.sgm”                     >
5   <!--    ELEMENTOS           CONTEÚDO                                          -->
6   <!ELEMENT   DISSERT     (%frontpg;,INDEX,CHAP+,REFER)                         >
7   <!ELEMENT   (%frontpg;)     (#PCDATA)                                         >
8   <!ELEMENT   INDEX       (#PCDATA)                                         >
9   <!ELEMENT   CHAP        (TITLE,SECT+)                                     >
10  <!ELEMENT   SECT        (SUBT,PAR+)+                                      >
11  <!ELEMENT   SUBT        (#PCDATA)                                         >
12  <!ELEMENT   PAR         ((TEXT,BIBREF*)+)                                 >
13  <!ELEMENT   TEXT        (#PCDATA)                                         >
14  <!ELEMENT   BIBREF      EMPTY                                             >
15  <!ELEMENT   REFER       (REFITEM+)                                        >
16  <!ELEMENT   REFITEM     (AUTHOR*,TITLE,YEAR)                              >
17  <!ELEMENT   YEAR        (#PCDATA)                                         >
18  <!--    ATRIBUTOS   NOMES       VALOR                       DEFAULT     -->
19  <!ATTLIST DISSERT       STATUS      (CONFIDEN|PUBLIC)   PUBLIC      >
20  <!ATTLIST REFITEM       id          ID                  #IMPLIED    >
21  <!ATTLIST BIBREF        rid         IDREF               #IMPLIED    >
```

Figura 10: DTD para uma dissertação

A Figura 10 apresenta um DTD para uma dissertação. Os componentes de um DTD são elementos, atributos e entidades, descritos a seguir.

3.1.3 Elementos

Um elemento (element) SGML delimita uma parte de um documento como um único objeto do documento, podendo conter texto, outros elementos, ou ambos. Fazendo relação com uma publicação, os elementos representariam os capítulos, as notas de rodapé (DEROSE, 1994). Um elemento é delimitado por um *start-tag* e um *end-tag*. Um *start-tag* consiste de um delimitador de abertura de *start-tag* (<) seguido pelo identificador genérico (*tag-name*) que identifica o tipo de elemento como definido no DTD, mais o delimitador de fechamento de *tag* (>). O *end-tag* possui o mesmo identificador genérico da *start-tag*, sendo iniciado por um delimitador de abertura *de*

end-tag (</), como apresentado na Figura 11 que descreve um exemplo de *start-tag e end-tag*.

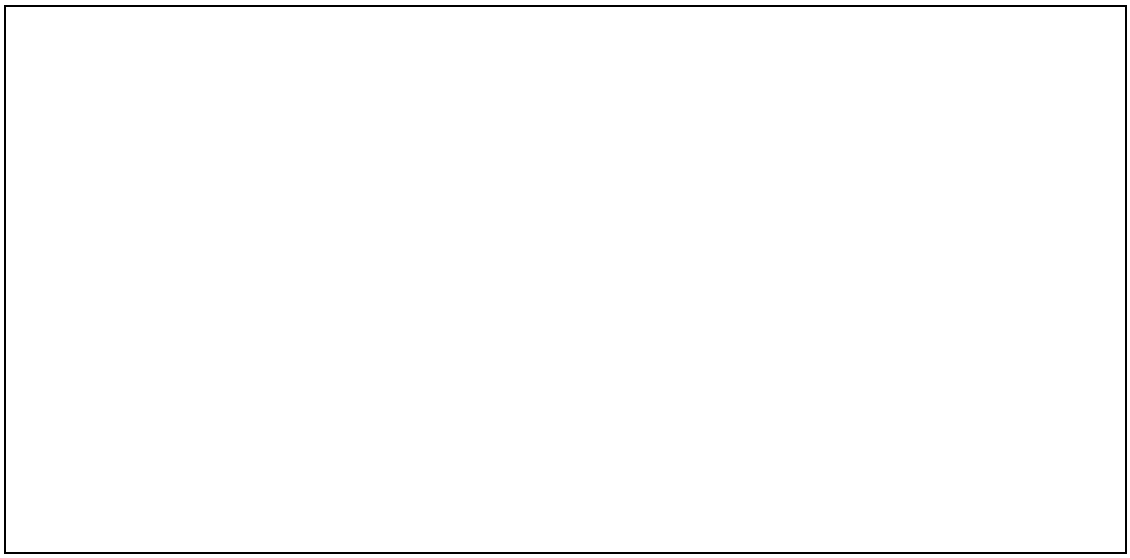


Figura 11: Anatomia de um elemento (HERWIJNEN, 1994)

O aninhamento de elementos SGML é realizado como apresentado na Figura 12, onde os elementos <SUBT> e <PAR> são aninhados no elemento <SECT>.

```
<CHAP>
  <TITLE>Documentos Estruturados</TITLE>
  <SECT>
    <SUBT>Considerações Iniciais</SUBT>
    <PAR>Este capítulo apresenta três padrões
internacionais...
    ... </PAR>
    <PAR> ... </PAR>
    ...
  </SECT>
</CHAP>
```

Figura 12: Exemplo de um documento SGML

3.1.4 Atributos

Um atributo (attribute) determina uma informação associada a um elemento, especificando uma propriedade como pertencente a ele. O atributo é posicionado nos *start-tags* de definição de um elemento. Um atributo consiste de um nome de atributo, um sinal de igualdade e um valor, como apresentado na Figura 13.

```
<title id = “tit1”> Título de nível 1 </title>
```

Figura 13: Exemplo de atributo (id)

A utilização de atributos como identificadores permite que elementos façam referências entre si. São atributos com essa possibilidade os “identificadores únicos” (IDs), que associam um nome a um elemento; e “referências a identificadores” (IDREF), que referenciam os elementos identificados com ID.

Os atributos pertencentes a um DTD são declarados em uma lista de declaração de atributos. Cada atributo tem um nome pelo qual ele é referenciado, um tipo que determina que valores ele pode ter, e um valor *default* para ser usado quando o atributo não for especificado (DEROSE, 1994).

3.1.5 Entidades

Uma entidade (entity) *é* um dado que será referenciado posteriormente (DEROSE, 1994), como todos os termos de uma abreviatura, por exemplo. Também podem ser referenciados através de entidades dados como imagens, texto, caracteres especiais, pertencentes, ou não, a um documento externo ao documento em questão.

Uma entidade é composta por um delimitador de abertura de referência a entidade (&), o nome da entidade e um delimitador de fechamento da referência a entidade (;).

A Figura 14 apresenta a definição de um texto para a abreviatura MSHd, e como utilizá-la em um documento.

```
Definição:
<!ENTITY     MSHd       “Minimal Scheduling HyTime documents”>

No documento:
“O subconjunto &MSHd; desenvolvido ...”

Interpretado como:
“O subconjunto Minimal Scheduling HyTime documents desenvolvido...”
```

Figura 14: Exemplo de definição e utilização de uma entidade

As entidades são declaradas através da apresentação do nome da entidade seguido do conteúdo do parâmetro literal (Figura 10, linha 3). No caso de definição de uma entidade externa, como um arquivo em diretórios, deve-se descrever, após o nome da entidade, a palavra-chave de identificador de sistema (SYSTEM) e o identificador de sistema (o caminho para o arquivo, por exemplo, como apresentado na Figura 10, linha 4)

Uma entidade externa registrada de uma forma padrão, que é citada como uma entidade pública, é especificada com a declaração, após o nome da entidade, da palavra-chave de identificador público (PUBLIC) e o identificador do proprietário do texto público (Figura 10, linha 2).

3.2 eXtensible Markup Language

O padrão XML descreve uma linguagem de marcação que simplifica a linguagem SGML, na qual é baseada (W3C, 1998a). De forma geral, como apresentado anteriormente, SGML é uma linguagem para descrição da estrutura lógica de um documento utilizando-se de marcações (HERWIJNEN, 1994). O XML, da mesma forma, permite que se defina a estrutura lógica para diferentes documentos. Uma das características do padrão XML é que a estrutura do documento, seu conteúdo e sua forma de apresentação são independentes, sendo que cada um destes pode ser definido

pelo usuário. Desta forma o mesmo oferece maior flexibilidade para desenvolver páginas e personalizá-las (W3C, 1998a).

A estrutura de um documento XML é definida através de um DTD em que são declarados quais tipos de elementos podem existir no documento, que atributos cada um desses tipos de elementos pode ter, e como instâncias destes tipos de elementos podem ser relacionadas hierarquicamente. A apresentação do conteúdo do documento pode ser definida utilizando o padrão XSL – *eXtensible Style Language* (W3C, 1998b). Este padrão define uma série de regras que, aplicadas ao documento XML, resultam no conteúdo deste com o estilo de apresentação aplicado e organizado como o documento XSL especifica.

Da mesma forma que cada instância da classe de documentos segue a mesma estrutura definida no DTD, o conteúdo de cada instância será apresentado conforme definido na especificação XSL (W3C, 1998b). Assim, cada mudança na definição do XSL implica em mudanças na apresentação de todos os documentos XML que a utilizam. É possível ainda criar múltiplas representações da mesma informação a partir de vários documentos XSL diferentes aplicados a um único documento, reforçando assim a característica de dissociar estrutura, conteúdo e apresentação do padrão XML.

Contendo apenas informações em forma de texto, os documentos XML são independentes de plataforma, assim uma ferramenta com potencialidade para trabalhar arquivos texto pode tanto ler quanto escrever documentos XML e apresentá-los em qualquer ambiente.

As linguagens HTML+TIME e SMIL, descritas nas próximas seções, são definidas como extensões da XML e utilizam-se de suas características, especialmente a definição através de uma DTD e a utilização de *stylesheets* para realizar a sua apresentação.

3.3 Hypermedia/Time-Based Structuring Language

O padrão HyTime (*Hypermedia/Time-Based Structuring Language*) é uma extensão do SGML que define como marcações e DTDs podem ser usados para

descrever a estrutura de documentos multimídia, com a utilização de *hyperlinks* e de escalonamento baseado em tempo (BUFORD, 1994).

O padrão HyTime não fornece uma representação para aspectos de um hiperdocumento que são específicos para um tipo de mídia particular ou interface com o usuário (NEWCOMB, 1991). De forma geral, HyTime especifica como certos conceitos considerados universais para todos documentos hipermídia podem ser representados utilizando SGML. Estes conceitos incluem a associação de objetos de documentos com *hyperlinks*, e o posicionamento e inter-relacionamento de objetos de documento de acordo com sistemas de coordenadas que podem representar espaço, tempo ou qualquer outra dimensão quantificável. HyTime, como SGML, fornece mecanismos para definir a representação de documentos hipermídia como arquivos texto que podem ser intercambiados e processados através de diferentes plataformas (RUTLEDGE, 1993).

3.3.1 Formas arquiteturais

O padrão HyTime é definido formalmente por um conjunto de regras, chamadas formas arquiteturais (FA - *architetural forms*), utilizadas para formalizar aspectos da estruturação de um documento. O padrão define um conjunto de formas arquiteturais para a definição de DTDs hipermídia. Cada forma arquitetural especifica como um tipo de elemento SGML pode ser definido em um DTD cujas instâncias contêm informações sobre certos conceitos HyTime.

Os aspectos de estruturação hipermídia incluem ligações multidirecionais e com múltiplas âncoras; os aspectos de estruturação multimídia incluem um mecanismo de localização de objetos flexível e poderoso, e o escalonamento de eventos no espaço e no tempo.

Uma instância de elemento é identificada como uma forma arquitetural HyTime através de um atributo chamado “HyTime”. O valor deste atributo é o nome da forma arquitetural que conforma com a instância de elemento. Essas formas arquiteturais constituem um meta-DTD que define como o DTD conformante com o HyTime pode ser construído. Uma aplicação que processa documentos HyTime pode ter um ou mais DTDs associados.

3.3.2 HyTime *engine*

O HyTime *engine* deve interagir com o SGML *parser* para reconhecer as formas arquiteturais e executar o processamento específico do HyTime, fornecendo informações para que a aplicação possa realizar a apresentação do hiperdocumento. Entre alguns dos processos que devem ser realizados estão a checagem de erros e a extração de atributos específicos do HyTime.

3.3.3 Módulos HyTime

O padrão HyTime é dividido em seis módulos: *Base Module*, *Location Address Module*, *Hyperlinks Module*, *Measurement Module*, *Scheduling Module* e *Rendition Module*, cuja interdependência é representada na Figura 15. Nem todos os módulos são necessários a todas as aplicações. Uma aplicação HyTime pode incorporar apenas os módulos que contenham as FAs necessárias aos seus hiperdocumentos e omitir o resto. Apenas o *Base Module* é obrigatório, os demais são opcionais.

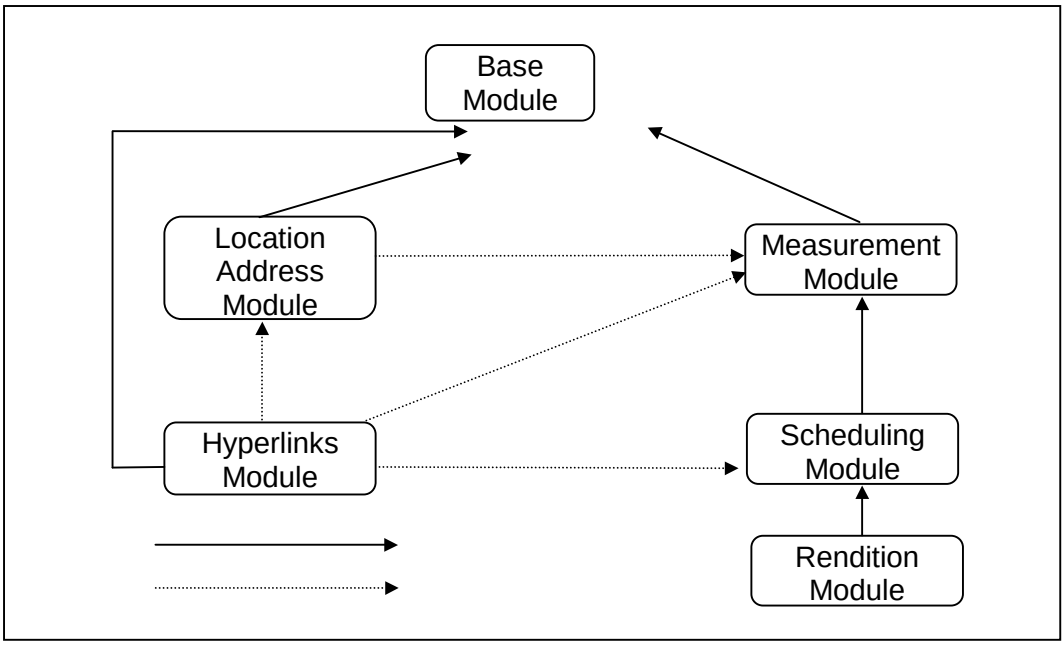


Figura 15: Interdependência dos módulos HyTime
(RUTLEDGE, 1993)

- *Base Module:* este módulo é obrigatório e é requisitado por todos os outros módulos. É formado por FAs de caráter independente, algumas delas obrigatórias e outras opcionais. As FAs obrigatórias fornecem suporte a gerência do hiperdocumento (utilizando SGML) e a identificação de propriedades HyTime enquanto as FAs opcionais fornecem facilidades para, por exemplo, a definição de valores de atributos e otimização de acesso direto a partes dos documentos (ISO, 1992).

Os aspectos da estruturação hipermídia, com a manipulação de ligações e âncoras, são trabalhados através das formas arquiteturais de dois módulos do padrão:

- *Location Address Module:* as FAs deste módulo estendem os identificadores únicos de endereçamento do SGML, permitindo a identificação de objetos que não podem ser endereçados somente com a utilização dos identificadores do SGML (BUFORD 1994). Também possibilita a localização de objetos que estão em documentos externos. Através das FAs aqui definidas é possível localizar padrões de caracteres, identificar porções de elementos, identificar elementos com determinadas propriedades e especificar um grupo de objetos como um único elemento (RUTLEDGE, 1993). A sua utilização em conjunto com o *Scheduling Module* permite endereçar porções em objetos externos a aplicação (ISO, 1992), criando endereçamento que pode ser especificado através de índices ao longo de determinadas dimensões;
- *Hyperlinks Module:* este módulo permite que sejam especificadas conexões (*hyperlinks*) entre objetos (ISO, 1992), independente da hierarquia do documento. A especificação de conexões (*link endpoints*) pode envolver o *Location Address*, *Measurement* e *Scheduling Module* (BUFORD, 1994).

A manipulação de objetos multimídia através de especificações HyTime é realizada através da utilização de formas arquiteturais de três dos módulos do padrão:

- *Measurement Module:* este módulo fornece mecanismos para a especificação da posição e dimensão de objetos, usando várias unidades de medida definidas pela aplicação (ISO, 1992). Permite que sejam definidas dimensões que utilizam unidades de medidas específicas, e que objetos sejam posicionados dentro dessas

dimensões em posições específicas que são determinadas por estas unidades de medida (RUTLEDGE, 1993);

- *Scheduling Module:* este módulo permite que eventos (ocorrências de objetos) sejam escalonados em eixos de coordenadas de espaço finitas (fcs - *finite coordinate spaces*) de tal forma que a posição desses objetos possa ser expressa em relação a esses eixos (ISO, 1992);

- *Rendition Module:* este módulo é dependente do *Scheduling Module* e estende as funções oferecidas por ele. As FAs desses módulos especificam como eventos em um *finite coordinate space* podem ser mapeados em outro (RUTLEDGE, 1993).

Dentre os inúmeros tipos de hiperdocumentos que podem ser formalizados através do HyTime, duas classes de documentos são definidas no padrão: a classe *Minimal Hyperlinking*, baseada no *hyperlinks module* e no *location address module*, que permite a definição de hiperdocumentos de estrutura elaborada; e a classe *Minimal Scheduling*, baseada no *scheduling module* e *measurement module*, cujas FAs permitem a formalização da estrutura de documentos multimídia. Deste modo, as classes *Minimal Hyperlinking* e *Minimal Scheduling* fornecem suporte, respectivamente, para a especificação da estrutura necessária para hiperdocumentos e elementos multimídia, e ilustram o fato do HyTime ser definido como *Hypermedia/Time-based Structuring Language.*

Os documentos HyTime especificam em suas declarações que módulos e opções são necessárias para o seu devido processamento. O padrão apresenta cinco possibilidades de utilização dos módulos de forma integrada para definir um subconjunto do HyTime que trabalhe determinadas classes de documentos:

- *Basic Hyperlinking HyTime document*: utiliza o *Base Module* e o *Location Address Module*, com FAs opcionais, e o *Hyperlinks Module*;

- *Basic Scheduling HyTime document*: todos os seis módulos do HyTime. Aos módulos *Base, Measurement, Location Address, Scheduling* e *Rendition* são acrescentadas FAs opcionais;

- *Minimal HyTime document*: utiliza somente o *Base Module*;

- *Minimal Hyperlinking HyTime document*: utiliza o *Base Module*, o *Location Address Module* com uma forma arquitetural opcional e o *Hyperlinks Module*;
- *Minimal Scheduling HyTime document*: utiliza o *Base Module*, acrescido da FA dcnatts, o *Measurement Module* acrescido da FA dimref, e o *Scheduling Module* acrescido de manyaxes.

Este último subconjunto é o objeto de investigação deste trabalho, sendo descrito na próxima seção através da apresentação das suas formas arquiteturais mais relevantes.

3.4 Minimal Scheduling HyTime Documents

Um *Minimal Scheduling HyTime document* (MSHd) é um subconjunto do padrão HyTime que possui os construtores necessários para trabalhar com escalonamento e sincronização de mídias. De acordo com a definição fornecida pelo padrão (ISO, 1992), um documento conformante com HyTime pertence ao subconjunto *Minimal Scheduling HyTime documents* se possuir as seguintes declarações:

```
<?HyTime VERSION “ISO/IEC 10744:1992” HYQCNT=32 >
<?HyTime MODULE    base       dcnatts >
<?HyTime MODULE    measure    dimref >
<?HyTime MODULE    sched      manyaxes=3 >
```

A primeira linha apresenta a versão do HyTime que será utilizada. As linhas seguintes apresentam os módulos que devem ser suportados. São declarados os módulos *Base Module, Measure Module e Scheduling Module*. Esses módulos, que permitem compor um documento do tipo MSHd, são descritos nas próximas seções.

Este subconjunto, por conter as formas arquiteturais destes módulos, fornece as estruturas necessárias para tornar possível a especificação de como as mídias devem relacionar-se com o tempo, e umas com as outras. A proposta deste trabalho é, através dos construtores ligados a este subconjunto, permitir que outras formas de relacionamento para sincronização entre as mídias possam ser especificadas. O resultado deste trabalho, descrito no capítulo 4, por utilizar-se das facilidades deste subconjunto, é considerado um MSHd.

3.4.1 Base Module

O *Base Module* trabalha com a definição das porções de um documento que serão processáveis pelo HyTime e com a definição de atributos comuns que podem ser utilizados em quaisquer outros elementos HyTime (ISO, 1992). São constantes do *Base Module* os construtores que serão utilizados posteriormente para definição de construtores em outros módulos.

Este trabalho fixa-se na descrição dos módulos *Measurement Module* e *Scheduling Module*, apresentados a seguir, que estão diretamente relacionados com a tarefa de escalonamento e sincronização de eventos.

3.4.2 Measurement Module

As formas arquiteturais deste módulo tornam possível a descrição de localizações e dimensões de objetos HyTime. A descrição de localização ou dimensão de um objeto é realizada de acordo com um posicionamento relativo do objeto em um espaço de coordenadas. Um espaço de coordenadas é um conjunto de eixos de coordenadas e um sistema de medidas nestes eixos.

O eixo de coordenadas é dividido em unidades chamadas *quantum*. Conceitualmente, um eixo é um conjunto ordenado de *quanta*. Os *quanta* são indivisíveis, sendo utilizados como conceito de um “ponto” em um eixo. A indivisibilidade do *quantum* tem o propósito de evitar que mudanças de plataforma dêem margem a inconsistências devido a mudanças de definições de ponto-flutuante (DEROSE, 1994).

Cada *quantum* pode ser especificado por um número ao longo do eixo e possui uma coordenada relativa a ele. Podem ser considerados *quantum* palavras, caracteres, nós em listas e árvores, etc. Dessa forma, podem ser considerados como eixos (ou parte de) uma frase, uma palavra, uma lista ou uma árvore. Para determinar uma área endereçável utiliza-se um espaço de números que podem especificar *quantum* no eixo. A área endereçável é determinada, então, por um *quantum-início* e um *quantum-fim*.

A especificação de um *quantum* específico em um eixo é realizada através de um marcador de eixo (*axis marker*). Se o marcador for positivo, o número é contado a partir

do início do espaço e se for negativo, conta-se a partir do fim do espaço de endereçamento.

A determinação da dimensão de um objeto é feita através da apresentação de dois desses marcadores: a posição (*quantum-inicial*), tamanho (número de *quanta*), e último *quantum*. Também pode-se determinar a dimensão de um objeto utilizando somente um marcador. Neste caso, a dimensão inicia a partir do fim da última dimensão especificada. Para especificação de dimensão há a FA dimspec, apresentada a seguir. Um marcador pode ser definido em termos de outro marcador em qualquer parte do documento.

O termo extensão do objeto diz respeito aos *quanta* ocupados por um objeto em múltiplos eixos, como, por exemplo, quando tem-se uma imagem que ocupará uma quantidade de *pixels* em uma tela. O espaço ocupado pela imagem nos eixos vertical e horizontal representa a extensão deste objeto. A especificação da extensão de um objeto é realizada através da FA extent, apresentada a seguir.

As seções seguintes descrevem algumas das formas arquiteturais do *Measurement Module*, exemplificando certos tipos de relacionamentos que há entre alguns deles.

Axis marker list

O conteúdo do marklist (*axis marker list*) é uma lista de valores que determinam uma seqüência de marcadores em um eixo. Cada marcador determina uma localização específica no eixo, podendo haver marklists aninhados. A Figura 16 mostra exemplos de marklist, com exemplo de aninhamento.

```
<marklist>1 2 3 4 -10</marklist>
<marklist>1 2<marklist>3 4 -10</marklist></marklist>
```

Figura 16: Exemplos de utilização do marklist

Dimension specification

O tipo de elemento dimspec (*dimension specification*) especifica uma dimensão como uma lista de dois axis markers. Contém um marklist com dois marcadores, como apresentado na Figura 17.

```
<dimspec><marklist>5 8</marklist></dimspec>
```

Figura 17: Exemplo de utilização do dimspec

A utilização de números negativos e positivos fornece significados especiais, como pode-se observar na Tabela 1. Um exemplo da utilização de números negativos e positivos é apresentado na Figura 18.

Marker1	Marker2	Interpretação
positivo	positivo	1: posição do *quantum* da esquerda 2: número de quanta
positivo	negativo	1: posição do *quantum* da esquerda 2: posição do *quantum* da direita medido do fim (da direita) do eixo
negativo	positivo	1: posição do *quantum* da esquerda contando do fim (da direita) do eixo 2: número de quanta
negativo	negativo	1: posição do *quantum* da esquerda contando do *quantum* da direita 2: posição do *quantum* da direita contando do fim (da direita) do eixo

Tabela 1: Semântica dos axis markers (DEROSE, 1994)

Esta é **uma explicação sobre o dimspec** architetural form

3	5
3	-3
-7	5
-5	-3

Figura 18: Utilização de marcadores negativos e positivos

Overrun

O atributo overrun está associado a elementos que definem espaços endereçáveis e permite que se especifique como deverão ser tratadas dimensões que ultrapassarem o espaço endereçável. Entre as possibilidades de tratamento estão o error (trata-se como um erro), o wrap (corta o primeiro e o último *quantum*, proporcionalmente), o trunc (corta o primeiro ou o último *quantum*) e o ignore (trata a dimensão como se ela não tivesse sido especificada) (ISO, 1992).

Dimension specification list

O tipo de elemento dimlist (*dimension specification list*) trabalha com uma seqüência de pares de marcadores ao longo do eixo

O conteúdo de um dimlist pode ser (DEROSE, 1994):

- elementos dimspec, representando dimensões individuais;
- pedaços de texto, representando marcadores;
- elementos marklist, cujos marcadores devem ser pares;
- elementos dimlist aninhados.

A Figura 19 apresenta exemplos de utilização de dimlist, em várias situações diferentes.

```
<dimlist>
    <dimspec>1 3</dimspec>
    <dimspec>4  3</dimspec>
    <dimspec>-9 2</dimspec>
</dimlist>

<dimlist>1 3 4 3 -9 2</dimlist>

<dimlist>1 3<dimspec>4 3</dimspec>-9 2</dimlist>

<dimlist>
    <marklist>1 3 4</marklist>
    <dimlist>3 -9 2</dimlist>
</dimlist>
```

Figura 19: Exemplo de utilização do dimlist

Scheduled extent

O tipo de elemento extent (*scheduled extent*) é utilizado com o Scheduling Module, especificando endereços de coordenadas em múltiplos eixos, com a definição de um dimlist por eixo. De forma geral, especifica uma seqüência de números que compõem as coordenadas de uma posição do evento em uma escala de eventos.

Scheduled extent list

O tipo de elemento extlist (*scheduled extent list*) tabela uma extensão como uma forma estendida do axis marker list, onde dimspecs são derivados dos axis markers, e os extents são derivados dos dimspecs.

Extent specification

O atributo exspec (*extent specification*) associa uma extensão escalonada em eixos de coordenadas com um evento. É utilizado em conjunto com o *Scheduling Module*.

Measurement domain definition

Um número utilizado como axis marker é especificado em termos de uma unidade específica de medida dentro de um domínio de medidas.

O padrão HyTime trabalha com unidades de medida padrão (*Standard Measurement Unit* - SMU) que define *quanta* genéricos. Esses *quanta* genéricos são utilizados para especificação de tamanhos e posições relativas que serão tratados de acordo com o sistema que está sendo utilizado para realizar a apresentação do documento.

O tipo de elemento measure (*measurement domain definition*) estabelece um conjunto de granularidade básica que será definida posteriormente em termos do SMU do domínio. O atributo SMU (*domain SMU*) especifica a unidade de medida padrão para o domínio. Com a utilização de um domínio virtual, isso deverá ser identificado através do texto "*Virtual Measurement Unit*".

Measurement Domain Unit rule

Uma unidade de domínio de medidas é um denominador comum das unidades de domínio utilizadas no HyTime utilizadas em um mesmo escopo.

O atributo docmdu (*document MDU*) especifica um MDU para cada domínio de medidas cujo escopo será o documento por inteiro (ISO, 1992).

Granule definition

O tipo de elemento granule (*granule definition*) identifica a granularidade como uma razão do domínio de medidas. Junto com esse elemento são utilizados os atributos gn (*granule name*) que nomeia a granularidade, e gd (*granule definition expression*) que especifica o numerador e o denominador para definir a razão de granularidade.

Schedule measurement

A lista de atributos schmeas (*schedule measurement*) define medidas para elementos que são escalas, ou serão posicionados em escalas, como uma escala de eventos. O atributo basegran (*base granule*) especifica a granularidade básica para cada eixo da escala.

Explicit dimension reference

O tipo de elemento dimref (*dimension reference*) permite que um componente de uma dimensão seja especificado em termos de outra.

O atributo elemref (*element reference*) identifica o elemento que está sendo referenciado (ISO, 1992). O atributo extref (*extent reference*) indica em que especificação de extensão de objeto está a dimensão especificada. O eixo onde está a dimensão é identificado através do atributo axisref (*axis referenced*).

O atributo selcomp (*selected component*) indica qual componente da dimensão será utilizado, podendo possuir os valores first (primeiro *quantum*), last (último *quantum*) e qcnt (quantidade de *quanta*). A Figura 20 demonstra como esses atributos podem ser utilizados.

```
<extlist id=act1>
  <dimspec id=dim1>1 45</dimspec> </extlist>

<extlist id=intermission1>
  <dimspec id=int1-dim>
        <dimref elecomp=dim1 selcomp=last>
        15      </dimspec> </extlist>

<extlist id=act2>
  <dimspec id=dim2>
        <dimref elecomp=int1-dim selcomp=last>
        45      </dimspec> </extlist>

<extlist id=intermission2>
  <dimspec id=int2-dim>
        <dimref elecomp=dim2 selcomp=last>
        15 </dimspec> </extlist>
```

Figura 20: Exemplo de utilização do dimref

A utilização de referências de dimensão recursivas possibilita a especificação de mecanismos de sincronização e relacionamentos entre objetos.

Marker function

A utilização de funções de marcação permite que sejam realizadas especificações de componentes de dimensões através de componentes de outras dimensões. De forma diferente do dimref, o tipo de elemento markfun (*marker function*) contém um elemento ou dado que será interpretado para representar uma marcação. Esses elementos ou dados podem ser um operador simples (HyOp) ou uma função (HyFunk).

HyOp: HyTime single operator marker function

O tipo de elemento HyOp tem como conteúdo uma lista de operandos de funções. Esses operandos podem ser originários de marklist, dimref, ou de outros markfun.

O tipo de operação que será realizada, ou seja, a função a ser executada, é identificada pelo atributo opname (*operator name*). Os valores que esse atributo pode assumir, ou seja, as funções que podem ser realizadas, são apresentadas na Tabela 2.

Operador	Operação
sum	adição
subt	subtração
mult	multiplicação
div	divisão
avg	média (arredondada)
max	máximo
min	mínimo
abs	valor absoluto
nabs	valor absoluto passado para negativo
incr	incrementa
decr	decrementa
rand	valor randômico
mod	resto da divisão

Tabela 2: Operadores do HyOp (ISO, 1992)

HyFunk element type

O conteúdo do tipo de elemento HyFunk é uma função montada através da linguagem definida para HyFunk. A função é uma expressão que se utiliza da notação oferecida pela linguagem HyFunk, e pode conter operandos ou argumentos cujos valores são passados através dos atributos arg do elemento HyFunk. O atributo usefn permite que seja atribuído um nome a essa função para ser utilizada posteriormente em outros elementos HyFunk.

3.4.3 Scheduling Module

O escalonamento de eventos tem o propósito de permitir que eventos possam ser sincronizados em um hiperdocumento. Para que eventos possam ser escalonados, é necessário que exista um espaço de coordenadas onde as dimensões e/ou as extensões dos eventos possam ser posicionadas.

As seções que seguem apresentam as formas arquiteturais relativas à tarefa de escalonar eventos em espaço de coordenadas.

Axis definition

O tipo de elemento axis (*axis definition*) define um domínio de medidas que pode ser visto como um eixo de coordenadas finito.

O atributo axismeas (*axis measurement domain*) é um SMU que identifica o domínio de medidas do eixo, enquanto o atributo axisdim (*axis dimension*) especifica a dimensão do eixo em MDUs (ISO, 1992).

Finite Coordinate Space

O espaço de coordenadas é definido através de um conjunto de eixos, especificado pelo tipo de elemento fcs (*finite coordinate space*). Cada fcs define o eixo de um espaço de coordenadas finito. O conteúdo de um elemento fcs é um ou mais elementos que utilizam a lista de atributos sched.

O atributo axisdefs (*axis definitions*) indica os eixos que serão utilizados para montar o espaço de coordenadas. O atributo fcsmdu (*fcs Measurement Domain Unit*) define um MDU para o fcs.

Event

Um evento representa a ocorrência de um objeto. O tipo de elemento event especifica essa ocorrência com a determinação do objeto que será considerado como um evento. A Tabela 3 apresenta alguns tipos de objetos que podem ser utilizados.

Notação	Objetos
Still	figuras simples: imagem JPEG, gráfico CAD, página de fax
Audio	áudio digital: áudio MPEG, CD, DAT
Video	vídeo digital: vídeo MPEG
Program	lógica de decisão compilada: C++
Script	lógica de decisão interpretada: linguagens de hipermídia interativa
Slide	sinais de controle de projetor de slides
Studio	sinais de controle de gravação
Device	sinais de controle de dispositivo analógico ou digital

Tabela 3: Tipos de objetos que podem ser utilizados (ISO, 1992)

O atributo exrecon (*extent reconciliation*) identifica uma estratégia de conciliação de extensão a ser seguida se o objeto não se enquadrar na extensão do evento. Estratégias de conciliação de extensão dizem respeito às técnicas que serão utilizadas para tratar objetos com extensão diferente daquela inicialmente prevista para ele, conciliando essas duas extensões. Este atributo será relacionado com o que estiver especificado para o elemento exrecon com o mesmo ID.

Event groups

Eventos contíguos podem ser agrupados de forma a possibilitar que sejam associadas propriedades a eles. A especificação de grupos de eventos é realizada através do tipo de elemento evtgrp (*event group*). Grupos de eventos também podem ser agrupados em outros evtgrps.

O atributo grpscope (*group scope*) especifica a extensão do grupo de evento, ou seja, a soma da extensão de todos os elementos presentes no grupo.

Event schedule

Uma seqüência de eventos, cada qual tendo uma dimensão nos eixos do espaço de coordenadas, é especificada através do tipo de elemento evsched (*event schedule*), como mostra a Figura 21. Para a especificação de vários eventos em uma mesma posição na escala, devem ser utilizados evscheds diferentes, um para cada evento.

```
<time id=time-axis>
<musicfcs>
        <evsched id=pop-concert>
                <event data = velocity-girl          exspec=act1>
                <event data = th-faith-healers       exspec=act2>
                <event data = MBV                    exspec=act3>
        </evsched>
</musicfcs>

<extlist id=act1> <dimspec>1 45</dimspec> </extlist>
<extlist id=act2> <dimspec>61 45</dimspec> </extlist>
<extlist id=act3> <dimspec>121 60</dimspec> </extlist>
```

Figura 21: Escalonamento de objetos de áudio (DEROSE, 1994)

Schedules

Uma escala (schedule) é uma seqüência de elementos na qual cada um deles possui uma dimensão em todos os eixos do espaço de coordenadas onde a escala ocorre (ISO, 1992).

O atributo axisord (*axis order*) é um conjunto ordenado de nomes de eixos de um fcs. O atributo apporder (*application ordering*) declara se a ordem de representação do elemento na escala é significante para a aplicação.

Extent reconciliation

O tipo de elemento exrecon (*extent reconciliation*) especifica técnicas para fazer com que um elemento enquadre-se na extensão definida para ele. Os atributos relacionados a este elemento definem quais das diferentes técnicas serão utilizadas para esse tratamento. Cada atributo faz referência a uma técnica diferente.

O atributo altrecon (*alternative reconciliation strategy*) identifica uma estratégia que pode ser utilizada para ajustar o objeto. Decidir se essa estratégia será seguida ou não é tarefa da aplicação, que para isso utilizará a informação fornecida pelo atributo altcrit (*alternative strategy criteria*). Uma alternativa é definir outro objeto para ser utilizado no lugar deste. Um outro objeto pode ser uma cópia do mesmo objeto, porém com configurações diferentes que permitam uma manipulação mais simplificada de parte do sistema. Como exemplo, pode-se disponibilizar um vídeo preto-e-branco caso o sistema não suporte apresentar um vídeo em cores. A referência a esse outro objeto é realizada através do atributo replace (*replacement object*).

O atributo align (*object alignment rule*) especifica como o objeto deve ser alinhado com a extensão para poder ser repetido, preenchido ou cortado. Pode ser definido o *quantum* central, o inicial ou o final para o alinhamento, como também pode ser utilizado um axis marker, que identificaria diretamente um *quantum*.

O atributo vamp (*vamping rule*) define que técnica de repetição deverá ser utilizada para utilizar toda a extensão disponível, caso o objeto seja menor. Pode-se repetir a apresentação do objeto até preencher toda a extensão disponível e então cortar o restante da última apresentação (*vampcrop*) ou realizar uma repetição menor até que o restante da extensão seja preenchida (*vampfill*).

O atributo filler (*malleable filler*) identifica um objeto que possa preencher a extensão restante.

Multiple axes

Para utilizar vários eixos, a opção manyaxes deve ser especificada. Com sua especificação na forma mais simples (Figura 22-a), um fcs pode ter qualquer número de eixos. Com a definição de um valor (Figura 22-b), limita-se o número de eixos a esse valor. O *Minimal Scheduling HyTime documents* trabalha com um número de eixos limitado a três.

```
<?HyTime        sched  manyaxes>          (a)

<?HyTime        sched   manyaxes = 3>      (b)
```

Figura 22 Especificação de múltiplos eixos

A disseminação da linguagem HTML, na verdade um DTD SGML, como instrumento de popularização da Internet através da WWW, apresenta a viabilidade de se construir mecanismos simples, como é o HTML, que permitam a divulgação de informações através das redes de computadores.

A utilização de meios mais complexos de informação, como áudio e vídeo, por exemplo, leva a necessidade de se criar novos mecanismos, que possibilitem que estes meios também sejam divulgados através das redes de computadores.

O padrão HyTime, extensão do SGML, visa suprir essa necessidade, apresentando facilidades para o tratamento de ligações de forma mais complexa e também oferecendo formas para que sejam realizados o escalonamento e a sincronização de mídias como áudio e vídeo.

O trabalho aqui apresentado sustenta-se na utilização dos mecanismos que o padrão HyTime oferece para escalonamento e sincronização de mídias buscando criar meios para anteriormente.

3.5 HTML+TIME

O padrão HTML+TIME (*Timed Interactive Multimedia Extensions*) foi desenvolvido por um grupo de especialistas, das empresas MICROSOFT, Compaq e Macromedia, pertencentes do comitê de desenvolvimento do SMIL, padrão de sincronização multimídia, recomendado pela W3C (W3C, 2000).

O HTML+TIME objetiva integrar áudio, vídeo, imagem e/ou texto em uma mesma aplicação, definindo como, quando e por quanto tempo cada uma destas mídias deve ficar ativa (MICROSOFT, 2002). Com a utilização desta tecnologia pode-se criar *sites* com conteúdo dinâmico melhorando a forma como ele é apresentado. Isto porque, em uma mesma aplicação, pode-se associar uma imagem a um texto definindo efeitos e regras de sincronização as suas apresentações (SCHMITZ, 1998).

Este padrão surgiu com a versão 1.0, foi aprimorada com novos elementos, atributos, métodos e eventos dando origem ao HTML+TIME 2.0. A limitação desta nova versão é sua compatibilidade com as versões antigas do Internet Explorer (na verdade esta é uma limitação do próprio padrão, visto que ele, somente pode ser interpretado pelo Internet Explorer), sendo suportada apenas pelas versões iguais ou superiores ao Internet Explorer 5.5. Outra característica importante é a semelhança com o SMIL visto que muito do que foi acrescentado foi baseado no próprio SMIL, tal como o elemento `t:ANIMATION` que define as formas de trabalhar com uma animação dentro de um documento (W3C, 2000).

Uma característica importante ao se trabalhar com uma linguagem de sincronização é o método de especificação que esta linguagem utiliza. Isto porque cada um destes métodos citados anteriormente define um conjunto de características que podem auxiliar no bom entendimento da linguagem. No caso do HTML+TIME não se utiliza apenas uma especificação. O modelo usado é formado pelas melhores características tanto da especificação baseada em eixos como da especificação baseadas em relações de intervalos (MICROSOFT, 2002), tais como:

- Especificação baseada em eixos: facilidade de visualizar e trabalhar com relações de tempo entre objetos de uma mesma mídia ou entre objetos de mídias diferentes. Isto porque, através desta especificação, pode-se definir o início e o

fim da apresentação de um objeto e sucessivamente o início e o fim dos objetos subseqüentes, como pode ser visualizado na Figura 23.

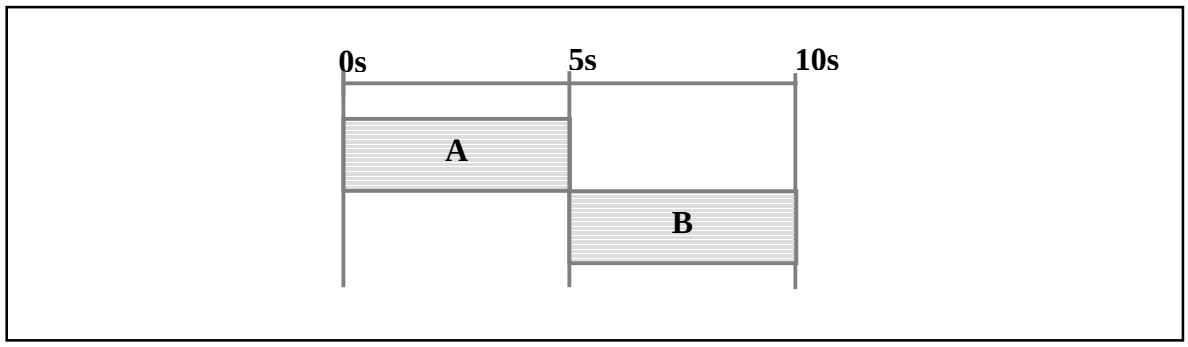


Figura 23: Utilização de eixo de tempo

A Figura 23 corresponde a uma apresentação composta por dois objetos, “A” e “B”. Cada um dos objetos será exibido de acordo com o que tiver sido definido pelos seus parâmetros: início, duração e fim. No caso, o objeto “A”, iniciará sua apresentação no tempo zero, ficará ativo durante cinco segundos e então será finalizado. Já o objeto “B” iniciará aos cinco segundos, tendo uma duração de cinco segundos e será finalizado aos 10 segundos.

- Especificação baseada em Relações Binárias: que favorece o tratamento de interações com o usuário, através das sete relações apresentadas anteriormente. Isto porque, através de uma destas relações (como o exemplo ilustrado pela Figura 24 abaixo: `A Meets B`), é possível condicionar a apresentação de um objeto a outro. Desta forma, pode-se definir que um objeto “B” somente será exibido após a interação do usuário, definido como o objeto “A”.

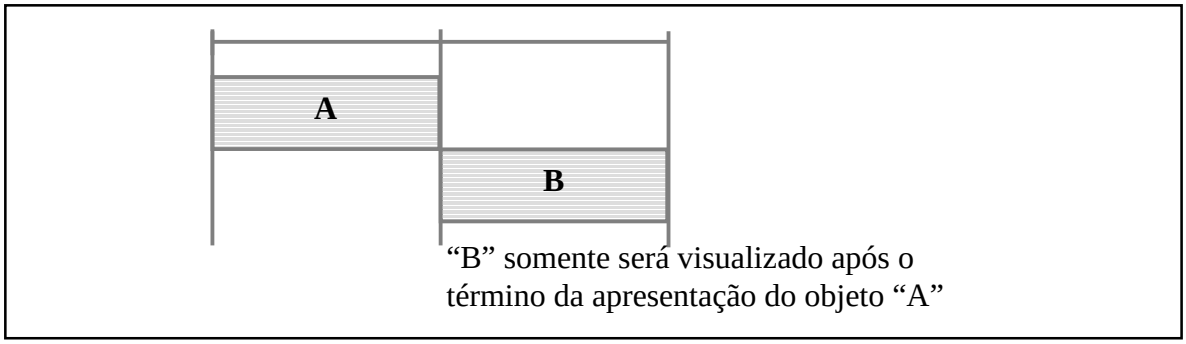


Figura 24: Relação baseada em intervalos

A Figura 24 define uma das sete relações binárias vistas na seção 2.4.1, mais especificamente a relação `A Meets B`, que indica que o objeto “B” somente poderá ser apresentado após a exibição do objeto “A”.

3.5.1 Sintaxe do HTML+TIME

O HTML+TIME é definido de acordo com uma DTD que define os elementos, atributos, propriedades, eventos e métodos que podem ser utilizados, bem como o resultado produzido pela utilização de um destes itens. Abaixo é mostrada a linha que realiza a ligação do documento HTML com esta DTD.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
```

Outra linha encontrada em documentos que trabalham com o padrão HTML+TIME corresponde a definição do espaço de nomes do HTML+TIME. Esta linha é apresentada abaixo.

```
<XML:NAMESPACE PREFIX="t"/>
```

Outra declaração necessária corresponde à definição da versão do HTML+TIME utilizado. Isto é definido através da criação de um estilo.

```
<STYLE>
        .time  {behavior:url(#default#time);}
</STYLE>
```

Um exemplo das funcionalidades descritas anteriormente pode ser visto na Figura 25.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
<html>
<head>
<title> Exemplo de Utilização </title>
<XML:NAMESPACE PREFIX="t"/>
<STYLE>
        .time  {behavior:url(#default#time);}
</STYLE>
</head>
<body>
        …
</body>
</html>
```

Figura 25: Estrutura de um documento com o HTML+TIME

Declarados os cabeçalhos que definem que o documento em questão irá utilizar o HTML+TIME para prover a sincronização dos objetos pertencentes a este documento, a etapa seguinte será definir o que deve ser apresentado e o que deverá ser sincronizado. O que for apenas apresentado usará somente *tags* HTML. Já o que for sincronizado usará estruturas implementadas pelo HTML+TIME no interior das *tags* HTML ou através de *scripts*, como poderá ser visto na Figura 26.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
<html>
<head>
<title> Exemplo de Utilização </title>
<XML:NAMESPACE PREFIX="t"/>
<STYLE>
        .time  {behavior:url(#default#time);}
</STYLE>
</head>
<body>
        <h1> Mensagem apenas apresentada.</h1>
        <h1 CLASS=”time” t:BEGIN=”5” t:DUR=”5”> Mensagem sincronizada. </h1>
</body>
</html>
```

Figura 26: Utilização do HTML+TIME

No exemplo acima a primeira mensagem, utilizando apenas *tags* HTML, não possui o objetivo de ser sincronizada e, portanto, simplesmente será apresentada logo que a página for iniciada, conforme definido para a *tag* HTML <h1>. Já a segunda mensagem será apresentada e sincronizada, conforme o que tiver sido definido pelas estruturas do HTML+TIME: t:BEGIN="5" e t:DUR="5". No caso, esta mensagem será apresentada cinco segundos após a inicialização da página (t:BEGIN="5") e ficará visível durante mais cinco segundos (t:DUR="5").

A seguir serão apresentados os principais elementos, métodos, atributos e propriedades utilizadas no HTML+TIME.

3.5.2 Elementos

O HTML+TIME possui uma série de elementos para trabalhar com as diferentes formas de mídia, tais como:

- t:AUDIO: Este elemento manipula um conjunto de métodos, atributos, propriedades e eventos que permitem o tratamento e a sincronização da mídia áudio;
- t:VIDEO: Trata da mídia vídeo e para tanto implementa recursos que permitam o tratamento de um vídeo bem como sua sincronização em relação as demais mídias;
- t:IMG: Este elemento visa oferecer os meios pelos quais imagens e textos podem ser apresentados de forma sincronizada;
- t:PAR: Define os meios para trabalhar com um conjunto de objetos pertencentes a uma apresentação de forma paralela;
- t:SEQ: Oferece um conjunto de recursos para sincronizar uma seqüência de objetos de uma apresentação.

Cada elemento representa um tipo de mídia e por isto possui um conjunto de características próprias. Assim uma imagem não deveria ser tratada da mesma forma que um áudio, visto que para a imagem, além de ter sua apresentação definida em relação a início, fim e duração, terá também definida sua posição no espaço.

Em HTML+TIME, a mesma forma usada para tratar um áudio será usada para tratar uma imagem, mesmo que alguma destas características não faça sentido para a mídia em questão. Outra consideração e que o HTML+TIME deixa a cargo do HTML a tarefa de dispor as mídias espacialmente.

3.5.3 Atributos e Propriedades

No HTML+TIME atributos e propriedades se diferenciam pela forma de sua utilização. Os atributos são usados juntamente com as *tags* HTML, enquanto que as propriedades são usadas através de *scripts*. Por padrão, o HTML+TIME adota letras maiúsculas para representar os atributos e letras minúsculas para escrever as propriedades.

Atributo BEGIN | Propriedade begin

	Sintaxe
Atributo:	< Element t:BEGIN = "valorTempo" >
Propriedade	obj.begin = "valorTempo"
Utilizado por:	Todos os elementos

Descrição: Esta propriedade/atributo determina o momento a partir do qual um objeto de uma apresentação será apresentado.

Exemplo de Utilização

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
<html>
<head>
<title> Exemplo de Utilização </title>
<XML:NAMESPACE PREFIX="t"/>
<STYLE>
        .time  {behavior:url(#default#time);}
</STYLE>
<script>
        function iniciar(objID)
        {
                objID.begin = "0"
        }
</script>
</head>
<body bgcolor="#CECCB9">
        <h3 ID="testarAtributo" CLASS="time" t:BEGIN="1">Apresentação de um texto
usando o atributo "t:BEGIN"</h3>
        <h4 ID="testarProp" CLASS="time" t:BEGIN="indefinite">Apresentação de um
texto através da propriedade "begin"</h4>
   <input type="button" name="btProp" value="Usar a propriedade"
onClick="iniciar(testarProp);">
</body></html>
```

Figura 27: Utilização do atributo | propriedade BEGIN

O exemplo da Figura 27 apresenta dois textos na tela de forma sincronizada, onde um deles (ID = "testarAtributo"), será apresentado um segundo após a página ser iniciada (t:BEGIN = "1"). Já o outro texto (ID = "testarProp"), ficará a espera de uma interação com o usuário. No momento em que o usuário clicar no botão, o evento "onClick" do botão será disparado e a função "iniciar" será chamada definindo que o objeto, no caso o segundo texto, seja imediatamente apresentado (objID.begin = "0").

Neste exemplo, caso o usuário não clicasse no botão, o segundo texto nunca seria visualizado. Isto porque, através da utilização do atributo t:BEGIN = "indefinite" definiu-se que o objeto associado não teria um tempo certo para sua apresentação, ficando a cargo da propriedade objID.begin = "0", ativada quando o evento "onClick" fosse chamado, definir a exibição deste objeto.

Atributo DUR | Propriedade dur

	Sintaxe
Atributo:	< Element t:DUR = "valorTempo" >
Propriedade	obj.dur = "valorTempo"
Utilizado por:	Todos os elementos

Descrição: Define quanto tempo durará a apresentação de um objeto.

Exemplo de Utilização

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
<html>
<head>
<title> Exemplo de Utilização </title>
<XML:NAMESPACE PREFIX="t"/>
<STYLE>
        .time  {behavior:url(#default#time);}
</STYLE>
<script>
        function iniciar(objID)
        {
                objID.dur="2"
        }
</script>
</head>
<body bgcolor="#CECCB9">
        <h3 ID="testarAtributo" CLASS="time" t:BEGIN="2" t:DUR="4">Apresentação do
primeiro texto.</h3>
        <h4 ID="testarProp" CLASS="time">Apresentação do segundo texto.</h4>
   <input type="button" name="btProp" value="Usar a Propriedade"
onClick="iniciar(testarProp);">
</body>
</html>
```

Figura 28: Utilização do atributo | propriedade DUR

Este exemplo, assim como o da Figura 27, possui a finalidade de apresentar dois textos na tela de forma sincronizada. Ao contrário do anterior, neste exemplo, a segunda mensagem é mostrada no momento em que a página for iniciada. Já o primeiro texto será visualizado dois segundos após, ficando ativo durante quatro segundos (t:DUR = "4") e desaparecendo em seguida. O segundo texto ficará sendo exibido até que o usuário interaja com a apresentação. Ao clicar no botão, o usuário dispara o evento

onClick. Este evento chama uma função denominada “iniciar” que definirá que o objeto será finalizado em dois segundos.

Atributo END | Propriedade end

	Sintaxe
Atributo:	< Element t:END = “valorTempo” >
Propriedade	obj.end = “valorTempo”
Utilizado por:	Todos os elementos

Descrição: Define o momento em que um objeto terá sua apresentação finalizada.

Exemplo de Utilização

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
<html>
    <head>
        <title> Exemplo de Utilização </title>
        <XML:NAMESPACE PREFIX="t"/>
        <STYLE>
                .time  {behavior:url(#default#time);}
        </STYLE>
        <script language="javaScript">
                function finalizar(obj)
                {   obj.end = "0"  }
        </script>
   </head>
   <body bgcolor="#CECCB9">
      <t:PAR ID="parent" CLASS="time" t:BEGIN=”0”>
        <img src="bolaAzul.gif" width="20" height="20" ID="BA" CLASS="time" t:DUR="2">
        <img src="quadradoVerde.gif" width="20" height="20" ID="QV" CLASS="time"
t:DUR="3">
        <img src="bolaVerde.gif" width="20" height="20" ID="BV" CLASS="time" t:DUR="3">
        <img src="quadradoAzul.gif" width="20" height="20" ID="QA" CLASS="time"
t:DUR="1">
      </t:PAR>
      <img src="todos.gif" width="100" height="70" ID="objTodos"  t:END = “30”
CLASS="time">
      <br><br><input type="button" name="btF" value="Fim"
onClick="finalizar(objTodos);"> Finalizar a apresentação da última imagem
   </body></html>
```

Figura 29: Utilização do atributo | propriedade END

O exemplo da Figura 29 utiliza o elemento “t:PAR” para apresentar, paralelamente e sincronizadamente, um conjunto de imagens. Todas as imagens definidas neste documento serão apresentadas logo que a página for iniciada, e de acordo com o tempo

definido pelo atributo t:DUR irão desaparecendo até restar a última imagem (ID= "objTodos"). Esta imagem é apresentada até que tenham se passados trinta segundos, sendo então finalizada (definido através do atributo t:END = "30"), ou então, esperará que o usuário, dentro destes trinta segundos, interaja com a aplicação e clique no botão. Ao clicar, a função "finalizar" definirá que o objeto (texto), através da propriedade "end", seja finalizada.

Atributo REPEATDUR | Propriedade repeatdur

	Sintaxe
Atributo:	< Element t:REPEATDUR = "valorTempo" >
Propriedade	obj.repeatdur = "valorTempo"
Utilizado por:	Todos os elementos

Descrição: Define um valor de tempo durante o qual os objetos pertencentes a uma aplicação serão apresentados repetidamente.

Exemplo de Utilização

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
<html>

<head>
<title> Exemplo de Utilização </title>
<XML:NAMESPACE PREFIX="t"/>
<STYLE>
        .time  {behavior:url(#default#time);}
</STYLE>
<body bgcolor="#CECCB9">
<table width="80%" border="0" cellspacing="0" cellpadding="0" align="center">
  <tr valign="middle" align="center">
  <t:SEQ ID="sequencia" class="time" t:REPEATDUR="10" t:BEGIN="1" t:DUR="4">
  <tr valign="middle" align="center">
    <td width="10%"> <img src="bolaAzul.gif" width="20" height="20" class="time"
t:DUR="1"></td>
    <td width="15%"> <img src="quadradoVerde.gif" width="20" height="20" class="time"
t:DUR="1"></td>
    <td width="9%"> <img src="bolaVerde.gif" width="20" height="20" class="time"
t:DUR="1"></td>
  </t:SEQ>
</table>
<br></body></html>
```

Figura 30: Utilização do atributo | propriedade REPEATDUR

Este exemplo utiliza o elemento “SEQ” para definir a apresentação de um conjunto de imagens de forma seqüencial e sincronizada. Cada imagem será exibida na ordem definida pelo documento HTML e encerrando o tempo de sua duração desaparecerá e dará lugar à próxima imagem. Pela definição normal de uma seqüência, terminada a apresentação da última imagem, a seqüência seria finalizada, no entanto, neste caso, através da utilização do atributo “t:REPEATDUR”, a seqüência será reiniciada até que seja esgotado o tempo definido por este atributo (10 segundos, de acordo com este exemplo).

Estas propriedades e atributos são definidos em relação ao tempo, ou seja: começam, terminam ou duram um certo tempo. Os formatos de tempo aceitos pelo HTML+TIME 1.0 são apresentados na Figura abaixo.

Formato	**Exemplos**	**Suportado por**
h:min:s.f	“1:03:10:375”: 1 hora, 3 minutos, 10,375 segundos; “10.5”: 10 segundo e meio	begin, dur, end, clipbegin, clipend, repeatdur.
Valor [h | min | s | ms]	“10h”: 10 horas “3.5s” or “3.5”: 3 segundos e meio	begin, dur, end, repeatdur
Onde: h: horas; min: minutos; s: segundos; f: frações de segundos; ms: milisegundos;		

Figura 31: Formatos de Tempo utilizados no HTML+TIME

Atributo REPEAT | Propriedade repeat

	Sintaxe
Atributo:	< Element t:REPEAT = “quantidade” >
Propriedade	obj.repeat = “quantidade”
Utilizado por:	Todos os elementos

Descrição: Utilizado para definir que a apresentação de um objeto será reapresentada um certo número de vezes. O valor definido por “quantidade” pode ser expresso pelo número de vezes que o objeto será reapresentado ou também pode receber o valor “indefinite” que indica que o objeto será reapresentado indefinidamente.

Exemplo de Utilização

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
<html>
<head>
<title> Exemplo de Utilização </title>
<XML:NAMESPACE PREFIX="t"/>
<STYLE>
        .time  {behavior:url(#default#time);}
</STYLE>
<body bgcolor="#CECCB9">
<table width="80%" border="0" cellspacing="0" cellpadding="0" align="center">
  <tr valign="middle" align="center">
  <t:SEQ ID="sequencia" class="time" t:REPEAT="5" t:BEGIN="1" t:DUR="4">
  <tr valign="middle" align="center">
    <td width="10%"> <img src="bolaAzul.gif" width="20" height="20" class="time"
t:DUR="1"></td>
    <td width="15%"> <img src="quadradoVerde.gif" width="20" height="20" class="time"
t:DUR="1"></td>
    <td width="9%"> <img src="bolaVerde.gif" width="20" height="20" class="time"
t:DUR="1"></td>
    <td width="11%"> <img src="quadradoAzul.gif" width="20" height="20" class="time"
t:DUR="1"></td>
  </tr>
  </t:SEQ>
</table>
<br>
</body>
</html>
```

Figura 32: Utilização do atributo | propriedade REPEAT

Este exemplo (Figura 32) funciona da mesma forma que o exemplo da Figura 30, porém utiliza o atributo "t:REPEAT" para fazer com que a seqüência seja repetida mais de uma vez. A diferença entre este atributo e o atributo anterior "t:REPEATDUR" é que enquanto este se baseia no tempo, ou seja, o objeto será repetido até que o tempo definido pelo atributo se esvaia, o "t:REPEAT" define que o objeto seja reapresentado uma certa quantidade de vezes, independentemente do tempo. Para que este atributo funcione corretamente é importante que o tempo de duração do objeto em questão seja definido, como pode ser visto no exemplo acima, através do atributo t:DUR = ″4″.

Atributo BEGINWITH | Propriedade beginWith

	Sintaxe
Atributo:	< Element t:BEGINWITH = "id">
Propriedade	obj.beginWith = "id"
Utilizado por:	Todos os elementos

Descrição: Utilizado para determinar que a apresentação de um objeto ocorrerá durante a apresentação de um outro. Caso não seja definido o início da apresentação deste objeto, através do atributo | propriedade "t:BEGIN", este objeto será visualizado ao mesmo tempo em que o objeto referenciado pelo "id". Caso seja (t:BEGIN = ″2″, por exemplo), sua exibição ocorrerá tantos segundos tiverem sido definido após o início do objeto referenciado pelo "id".

Exemplo de Utilização

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
<html>
<head>
<title> Exemplo de Utilização </title>
<XML:NAMESPACE PREFIX="t"/>
<STYLE>
        .time  {behavior:url(#default#time);}
</STYLE>
</head>
<body bgcolor="#CECCB9">
<table width="78%" border="0" cellspacing="0" cellpadding="0" align="center">
  <tr valign="middle" align="center">
    <td width="30%">
          <img src="bolaAzul.gif" width="30" height="30"  ID="bolaAzul" class="time"
t:BEGIN="2" t:DUR="5">
    </td>
    <td width="40%">
    </td>
    <td width="30%">
          <img src="bolaVerde.gif" width="30" height="30" class="time" ID= "bolaVerde"
t:BEGINWITH="bolaAzul" t:BEGIN="2" t:DUR="5">
        </td>
  </tr>
</table></body></html>
```

Figura 33: Utilização do atributo | propriedade BEGINWITH

Este exemplo utiliza o atributo t:BEGINWITH para definir que um objeto (ID=″bolaVerde″) deverá ser apresentado durante a exibição de um outro objeto

(ID=”bolaAzul”). No caso deste exemplo, ficou definido que o segundo objeto identificado por “bolaVerde” será visualizado dois segundos após o início do primeiro objeto identificado como “bolaAzul”.

Atributo BEGINAFTER | Propriedade beginAfter

	Sintaxe
Atributo:	< Element t:BEGINAFTER = “id” >
Propriedade	obj.beginAfter = “id”
Utilizado por:	Todos os elementos

Descrição: Determina que a apresentação de um objeto acontecerá somente após a apresentação de um outro. A utilização deste atributo | propriedade se torna imprescindível sempre que não se tiver certeza de quando a apresentação de um objeto anterior será finalizada. Este atributo / propriedade implementa a relação oposta de uma das sete relações binárias vista na seção 2.4.1, mas especificamente a relação A Before B. O valor “id” apresentado acima corresponde ao identificador do objeto de referência.

Exemplo de Utilização

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
<html>

<head>
<title> Exemplo de Utilização </title>
<XML:NAMESPACE PREFIX="t"/>
<STYLE>
                .time  {behavior:url(#default#time);}
</STYLE>
<body bgcolor="#CECCB9">
        <h1 ID="msg1" CLASS=time t:BEGIN="1" t:DUR="2">1ª Mensagem: Esta
mensagem será apresentada durante 2 segundos.</h1>
        <h2 CLASS=time t:BEGINAFTER="msg1" t:BEGIN="1" t:DUR="5">2ª Mensagem:
Esta será visualizada durante 5 segundos, porém sua apresentação somente ocorrerá quando a
1ª for finalizada.</h2>
</body>
</html>
```

Figura 34: Utilização do atributo | propriedade BEGINAFTER

O exemplo exibe duas mensagens na tela de forma sincronizada. Para tanto o mesmo utiliza o atributo t:BEGINAFTER para indicar que a segunda mensagem somente poderá ser visualizada após a apresentação da primeira.

Atributo BEGINEVENT | Propriedade beginEvent

	Sintaxe
Atributo:	< Element t:BEGINEVENT = "evento">
Propriedade	obj.beginEvent = "evento"
Utilizado por:	Todos os elementos

Descrição: Define que a apresentação do objeto em questão somente ocorrerá caso o evento esperado ocorra. O valor "evento" define qual o evento responsável por ativar o início da apresentação do objeto.

São exemplos de eventos:

- Evento onclick: este evento corresponde ao momento em que o objeto recebe um clique, como por exemplo, um clique sobre um botão de um formulário HTML.
- Evento onfocus: representa o momento em que o objeto recebeu o foco.
- Evento onblur: representa o momento em que o objeto perde o foco.

Atributo ENDEVENT | Propriedade endEvent

	Sintaxe
Atributo:	< Element t:ENDEVENT = "evento" >
Propriedade	obj.endEvent = "evento"
Utilizado por:	Todos os elementos

Descrição: Define que a apresentação do objeto em questão será finalizada caso o evento esperado ocorra.

Exemplo de utilização

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
<html>

<head>
<title> Exemplo de Utilização </title>
<XML:NAMESPACE PREFIX="t"/>
<STYLE>
        .time  {behavior:url(#default#time);}
</STYLE>
</head>
        <body bgcolor="#CECCB9">
<table width="78%" border="0" cellspacing="0" cellpadding="0" align="center">
  <tr valign="middle" align="center">
    <td width="27%"  class="time" t:ENDEVENT="botao.onclick" ><img src="bolaAzul.gif"
width="30" height="30">
    </td>
    <td width="28%">
      <input type="button" name="botao" id="botao" value="Fim | Início">
    </td>
    <td width="55%"  class="time" t:BEGINEVENT=" botao.onclick"><img
src="bolaVerde.gif" width="30" height="30" ></td>
  </tr>
</table>
<br>
</body>
</html>
```

Figura 35: Utilização dos atributos | propriedades BEGINEVENT e ENDEVENT

O exemplo da Figura 35 caracteriza-se pela ausência dos atributos / propriedades comuns tais como: t:BEGIN, t:END e t:DUR. O objetivo deste exemplo é exibir duas imagens, onde a primeira, logo que a página for iniciada, será visualizada, indefinidamente, até que o usuário interaja com a aplicação, onde o mesmo definirá que a primeira imagem tenha sua apresentação finalizada e a segunda imagem iniciada. Isto porque, os dois atributos / propriedades, responsáveis pela sincronização dos objetos desta apresentação (t:BEGINEVENT e t:ENDEVENT), estão associados ao evento onclick do botão. No momento em que o usuário clicar no botão o evento será ativado, a primeira imagem, através do atributo t:ENDEVENT, será encerrada e a segunda imagem, através do atributo t:BEGINEVENT, será iniciada.

Atributo TIMEACTION | Propriedade timeAction

	Sintaxe
Atributo:	< Element t:TIMEACTION = "ação" >
Propriedade	obj.timeAction = "ação"
Utilizado por:	Todos os elementos

Descrição: Determina a ação de um objeto (atribuindo ou retirando) enquanto persistir sua apresentação. O valor "ação" define o tipo de ação o qual o objeto será afetado durante a sua apresentação.

Exemplos de possíveis valores:

- display: Define que o objeto será apresentado enquanto perdurar sua apresentação desaparecendo logo em seguida. O interessante deste modo é que, após sua apresentação, o próximo objeto a ser apresentado inicia sua apresentação no mesmo local onde o mesmo foi apresentado, como se o objeto apresentado anteriormente tivesse deixado de existir.

- visibility: Este é o valor padrão e da mesma forma que o modo anterior, define que o objeto estará visível enquanto a apresentação perdurar, desaparecendo logo em seguida, porém ao contrário do modo apresentado anteriormente, o espaço onde este objeto foi apresentado continua reservado a ele, mesmo que sua apresentação já tenha terminado.

- style: Este modo define que enquanto o objeto estiver sendo apresentado será usado o estilo de apresentação definido e logo que a apresentação for finalizada o estilo de apresentação deixará de ser usado, porém ao contrário dos demais modos apresentados o objeto continuará visível.

Exemplo de Utilização

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
<HTML>
<HEAD>
<STYLE>
  .time  { behavior: url(#default#time);}
</STYLE>
</HEAD>
<BODY BGCOLOR="white">
<SPAN CLASS="time" STYLE="Color:Red; Font-Weight:bold;" t:BEGIN="0"
    t:DUR="5" t:TIMEACTION="style">
    <H3>Parágrafo 1</H3>
    <P>Este parágrafo utiliza o modo de ação "style". Note que durante 5 segundo o texto
estará em vermelho e negrito e assim que estes 5 segundo se esvair o texto tornará preto e
sem negrito.</P>
</SPAN>
<SPAN CLASS="time" STYLE="COLOR:Blue;" t:BEGIN="0" t:DUR="10"
    t:TIMEACTION="display">
    <H3>Parágrafo 2</H3>
    <P>Este parágrafo utiliza o modo "display". Depois de 10 segundos ele desaparecerá e o
próximo parágrafo se movimentará para o lugar onde este se encontrava.</P>
</SPAN>
<SPAN CLASS="time" STYLE="COLOR:Blue;" t:BEGIN="0" t:DUR="15"
    t:TIMEACTION="visibility">
    <H3>Parágrafo 3</H3>
    <P>Este parágrafo utiliza o modo "visibility". </P>
</SPAN>
</BODY>
</HTML>
```

Figura 36: Utilização do atributo | propriedade TIMEACTION

O exemplo exibe três mensagens sincronizadamente. Assim que a página for iniciada, as três mensagens serão apresentadas. A primeira mensagem, durante cinco segundos, será visualizada conforme o estilo previamente definido (vermelho e em negrito). Esgotado este período, a mensagem continuará visível, porém sem o estilo. A segunda mensagem ficará ativa durante dez segundos e após este período será finalizada e desaparecerá. Já a última, mesmo após o encerramento de sua apresentação, continuará visível.

Atributo PLAYER | Propriedade player

	Sintaxe
Atributo:	< Element t:PLAYER = "id" >
Propriedade	obj.player = "id"
Utilizado por:	

Descrição: Determina o tipo de *player* responsável por apresentar o objeto.

Atributo SRC | Propriedade src

	Sintaxe
Atributo:	< Element t:SRC = "local" >
Propriedade	obj.str = "local"
Utilizado por:	

Descrição: Determina o local onde o arquivo do objeto a ser apresentado se encontra.

Exemplos de utilização

```
<HTML>
<HEAD>
<TITLE>playerObject</TITLE>
<XML:NAMESPACE PREFIX="t"/>
<STYLE>
.time          { behavior: url(#default#time);}
</STYLE>
</HEAD>
<BODY bgcolor="#CECCB9">
<t:AUDIO ID="audio" CLASS="time" t:PLAYER="{22d6f312-b0f6-11d0-94ab-
0080c74c7e95}" t:SRC="D:\Músicas\Legião Urbana\eusei.mp3" t:BEGIN="5"
t:DUR="20" />
</BODY>
</HTML>
```

Figura 37: Utilização dos atributos | propriedades PLAYER e SRC

Este exemplo especifica o *player* a ser utilizado para tocar o áudio. Através do atributo t:PLAYER (no caso o *Windows Media Player*), define o local onde se encontra o áudio, através do atributo t:SRC. Este áudio será apresentado cinco segundos após a inicialização da página, durante vinte segundos, tendo então, sua apresentação finalizada.

3.5.4 Métodos

O HTML+TIME 1.0 implementa dois métodos: beginElement e endElement, que serão apresentados abaixo.

Método beginElement

	Sintaxe
Método	obj.beginElement()
Utilizado por:	Todos

Descrição: Inicia a apresentação de um objeto.

Método endElement

	Sintaxe
Método	obj.endElement()
Utilizado por:	Todos

Descrição: Finaliza a apresentação de um objeto.

Exemplo de Utilização

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
<html><head><title> Exemplo de Utilização </title>
<XML:NAMESPACE PREFIX="t"/>
<STYLE>
        .time  {behavior:url(#default#time);}
</STYLE>
</head>
<body bgcolor="#CECCB9">
<table width="80%" border="0" cellspacing="0" cellpadding="0" align="center">
  <tr valign="middle" align="center">
    <td colspan="5" height="20"><b><font color="#333333">Trabalhando
      com o Elemento "PAR"</font></b>  </tr>
  <t:PAR ID="paralelo" class="time" t:REPEAT="5" t:BEGIN="1" t:DUR="9">
  <tr valign="middle" align="center">
    <td width="10%"> <img src="bolaAzul.gif" width="20" height="20" class="time" t:DUR="2"></td>
    <td width="15%"> <img src="quadradoVerde.gif" width="20" height="20" class="time" t:DUR="3"></td>
    <td width="9%"> <img src="bolaVerde.gif" width="20" height="20"  class="time" t:DUR="3"></td>
    <td width="11%"> <img src="quadradoAzul.gif" width="20" height="20" class="time" t:DUR="1"></td>
</tr>
  </t:PAR>
  <tr valign="middle" align="center">
    <td colspan="4" height="50">
      <input type="button" name="btIniciar" value="Reiniciar PAR" onClick="paralelo.beginElement();">
      <input type="button" name="btIniciar" value="Finalizar PAR" onClick="paralelo.endElement();">
      <input type="button" name="btIniciar12" value="Reiniciar SEQ"
onClick="sequencia.beginElement();">
      <input type="button" name="btIniciar1" value="Finalizar SEQ" onClick="sequencia.endElement();">
    </td> </tr>
  <tr valign="middle" align="center">
    <td colspan="4" height="20"><b><font color="#333333">Trabalhando
      com o Elemento "SEQ" </font></b>  </tr>
  <t:SEQ ID="sequencia" class="time" t:REPEAT="5" t:BEGIN="1" t:DUR="4">
  <tr valign="middle" align="center">
    <td width="10%"> <img src="bolaAzul.gif" width="20" height="20" class="time" t:DUR="1"></td>
    <td width="15%"> <img src="quadradoVerde.gif" width="20" height="20" class="time" t:DUR="1"></td>
    <td width="9%"> <img src="bolaVerde.gif" width="20" height="20" class="time" t:DUR="1"></td>
    <td width="11%"> <img src="quadradoAzul.gif" width="20" height="20" class="time" t:DUR="1"></td>
  </tr>
  </t:SEQ>
</table>
</body>
</html>
```

Figura 38: Utilização dos métodos beginElement e endElement

Este exemplo utiliza diversos elementos, atributos e métodos apresentados até aqui. Possui o objetivo de demonstrar a utilização dos métodos beginElement e endElement e fazer uma comparação entre os elementos t:PAR e t:SEQ.

No momento em que a página for iniciada, tanto o objeto definido pelo elemento t:SEQ como o objeto definido por t:PAR terão suas apresentações iniciadas. O primeiro objeto (t:PAR) terá todas as imagens apresentadas ao mesmo tempo e de acordo com o período, definido para cada um, irão desaparecendo, até que a apresentação chegue ao fim, reiniciando logo em seguida, até que chegue ao fim do número de repetições definido através do atributo t:REPEAT.

Para o segundo objeto (t:SEQ), as imagens serão seqüencialmente apresentadas, sendo que ao final do período definido para a exibição de uma imagem, esta desaparece e dará lugar a imagem seguinte, sucessivamente até que chegue ao final da seqüência. A seqüência será reiniciada até que o número de repetições, definido pelo atributo t:REPEAT, seja atendida.

Este exemplo possui quatro botões, em que dois deles controlam a apresentação do objeto definido pelo elemento t:PAR e outros dois controlam o objeto definido pelo elemento t:SEQ Por exemplo, o usuário ao clicar no botão para finalizar o primeiro objeto (t:PAR), dispara o evento onclick do botão que ativa o método endElement associado ao objeto em questão. Da mesma forma ao clicar no botão para reiniciar este objeto, o método beginElement associado a este objeto é ativado, fazendo com que a apresentação do objeto seja reiniciada.

O HTML+TIME, permite que os objetos pertencentes a uma aplicação multimídia sejam corretamente sincronizados. Este padrão utiliza dois métodos de especificação (baseado em intervalos e baseado em eixos de tempo) trabalhando de acordo com os mesmos. Na seção seguinte será abordado o padrão SMIL que possui o mesmo objetivo: oferecer os meios para que uma aplicação multimídia seja corretamente apresentada.

3.6 Synchronized Multimedia Integration Language

O SMIL consiste no padrão adotado e desenvolvido pela W3C para o desenvolvimento de aplicações multimídia sincronizadas. Assim como o HTML+TIME, o SMIL possui o objetivo de integrar um conjunto de objetos (mídias) independentes em uma apresentação multimídia sincronizada (W3C, 2000) permitindo que seja

especificado o comportamento temporal da apresentação, a disposição espacial e as relações entre os diversos objetos.

Como o objetivo desta tecnologia é ser padrão com relação a sincronização multimídia, a sintaxe do SMIL foi definida através de uma DTD, possibilitando que qualquer outra tecnologia que possa trabalhar com o padrão XML possa usar o SMIL. Este padrão surgiu em sua versão 1.0, foi melhorado e dando origem a 2.0. Esta versão, em muitos pontos, serviu de base para o desenvolvimento do HTML+TIME 2.0. O SMIL, para ser apresentado, necessita de um *player* ou *browser* que o suporte.

Ao contrário do HTML+TIME, o SMIL trabalha apenas com o modelo de especificação baseada em eixos de tempo (W3C, 2000). Uma característica importante deste padrão, proveniente da utilização do XML para a definição de sua especificação é que, apesar da semelhança das *tags* usadas na criação de um documento SMIL com um em HTML, o SMIL não apresenta algumas das inconstâncias presentes na criação de documentos HTML, tal como: não ser *case-sensitive*, ou seja, o SMIL diferencia letras maiúsculas de minúsculas, sendo que, as *tags* que fazem parte de sua definição, devem ser escritas em minúsculo.

Um documento que usa o padrão SMIL possui a extensão ".smil" e será interpretado pelos *players* e *browser*s que consigam abrir e interpretar este tipo de arquivo. Os arquivos SMIL, por se tratar de um arquivo texto, podem ser editados por qualquer editor.

3.6.1 Sintaxe do SMIL

O SMIL, da mesma forma que o HTML+TIME (seção 3.5), está associado a um DTD responsável pela sua estruturação. Neste DTD, foram definidos os elementos e atributos que podem ser trabalhados por este padrão. Estes recursos serão especificados nas subseções seguintes.

3.6.2 Elementos

O SMIL possui um conjunto de elementos que podem ser manipulados na estruturação de uma apresentação que se deseja corretamente sincronizada. estes elementos são apresentados a seguir.

Elemento smil

Este elemento corresponde ao elemento raiz de um documento SMIL e todos os demais elementos serão definidos entre a *tag* de abertura (<smil>) e a *tag* de fechamento (</smil>) deste elemento.

Atributos	id
Conteúdo do elemento	head?, body?

Para o elemento smil é definido o atributo id, que especifica a unicidade do elemento dentro do documento. Já como conteúdo podem ser definidos os elementos: head e body, que são elementos opcionais em um documento SMIL. Caso estejam presentes, devem respeitar a ordem mostrada acima.

```
<smil id="1">
        <head>
                ...
        </head>
</smil>
```

```
<smil id="1">
        <body>
                ...
        </body>
</smil>
```

```
<smil id="1">
        <head>
                ...
        </head>
        <body>
                ...
        </body>
</smil>
```

Figura 40: Utilização do elemento smil

A Figura 40 demonstra as possíveis formas de utilização dos elementos head e body dentro de um documento SMIL.

Elemento head

O elemento head é o responsável por definir as informações do cabeçalho, levando em consideração apenas a sincronização espacial.

Atributos	id
Conteúdo do elemento	meta*, ((layout || switch), meta*)?

Os elementos definidos em seu conteúdo (layout, meta e switch), são os responsáveis por definir as regiões de apresentação (layout), as propriedades de um elemento (meta) e um conjunto de elementos como opções em que apenas um será usado (switch). Para que um documento SMIL esteja de acordo com a definição de sua DTD este elemento poderá apresentar, como seu conteúdo, zero ou mais elementos meta e apenas um elemento layout ou switch. Os elementos meta podem ser definidos antes ou depois de um elemento layout ou switch. O exemplo da Figura 41 demonstra estas relações.

<smil id=”1”> <head> <meta …/> <meta …/> <meta …/> </head> </smil>	<smil id=”1”> <head> <meta …/> <meta …/> <layout>...</layout> <meta …/> </head> </smil>	<smil id=”1”> <head> <meta …/> <meta …/> <layout>...</layout> </head> </smil> OU <smil id=”1”> <head> <layout>...</layout> <meta …/> <meta …/> </head> </smil>
<smil id=”1”> <head> <meta …/> <meta …/> <switch>...</switch> <meta …/> </head> </smil>	<smil id=”1”> <head> <meta …/> <meta …/> <switch>...</switch> </head> </smil> OU <smil id=”1”> <head> <switch>...</switch> <meta …/> <meta …/> </head> </smil>	<smil id=”1”> <head> <switch>...</switch> </head> </smil> OU <smil id=”1”> <head> <switch>...</switch> </head> </smil>

Figura 41: Utilização do elemento head

Elemento meta

É composto apenas por atributos (não tem conteúdo), possuindo a função de definir propriedades para um determinado elemento, tais como: autor, data de validade, lista de palavras-chaves.

Atributos (mais comuns)	Id, name, content
Conteúdo do elemento	vazio

O atributo id identifica o elemento como sendo único no documento. O atributo name define o nome da propriedade definida no elemento meta e o atributo content especifica um valor a esta propriedade. Uma possível propriedade que pode ser definida

através deste elemento é a definição do título da apresentação (name="title"), como pode ser visto no exemplo da Figura 42.

```
<smil id="1">
        <head>
                <meta name="title" content="Apresentação usando SMIL."/>
        </head>
</smil>
```

Figura 42: Utilização do elemento meta

Elemento layout

O elemento layout possui o objetivo de especificar o posicionamento dos objetos definidos no elemento body.

Atributos	id, type
Conteúdo do elemento	ANY (pode conter tanto caracteres como elementos)

Normalmente o conteúdo deste elemento é formado por: root-layout e region. A Figura 43 apresenta uma utilização do elemento layout.

```
<smil id="1">
        <head>
                <layout>
                        <region .../>
                </layout>
        </head>
</smil>
```

Figura 43: Utilização do elemento layout

Este exemplo define uma das possíveis seqüências para a definição do cabeçalho de um documento SMIL

Elemento switch

O elemento switch permite a definição de um conjunto de alternativas entre as quais somente uma será escolhida. Ao analisar o documento, a primeira das opões que satisfizer a condição será usada.

Atributos	id, title
Conteúdo do elemento (caso seja usado no interior de um elemento body)	(par | seq | (audio | video | text | img | animation | textstream | ref) | switch | a)*
Conteúdo do elemento (caso seja usado no interior de um elemento head)	layout*

O exemplo da Figura 44 apresenta uma possível forma de utilização do elemento switch.

```
<smil>
  <head>
        ...
   </head>
   <body>
     <switch>
        <opcao 1/>
        <opcao 2/>
     <switch>
   </body>
</smil>
```

Figura 44: Utilização do elemento switch

Elemento region

O elemento region permite a manipulação (da posição espacial, do tamanho e efeitos) dos objetos pertencentes a uma apresentação multimídia. O elemento region não possui conteúdo (elemento vazio), mas possui um conjunto de atributos que permitem fazer sua configuração.

Atributos (mais comuns)	background-color, fit, height, id, left, title, top, width
Conteúdo do elemento	Vazio

Elemento root-layout

O elemento root-layout define as propriedades de disposição do elemento raiz, especificando os parâmetros para a visualização da apresentação.

Atributos (mais comuns)	background-color, height, id, left, title, top, width
Conteúdo do elemento	Vazio

Atributos usados pelos elementos root-layout e region

Estes elementos são os responsáveis pela definição dos locais onde os objetos serão apresentados, e para que esta tarefa seja atendida, existe a necessidade que informações tais como: altura, largura, cor, entre outras; sejam definidas. A seguir estão relacionados os atributos responsáveis pela definição destas informações.

- **background-color:** especifica a cor de fundo da região definida.
- **height:** define a altura da região e pode ser expressa tanto em porcentagem (height="10%") como em *pixels* (height="50px" ou height="50").
- **width:** define o comprimento da região e pode ser expressa tanto em porcentagem (width="10%") como em *pixels* (width="50px" ou width="50").
- **left:** especifica a posição horizontal da região definida e pode ser expressa tanto em porcentagem (left="10%") como em *pixels* (left="50px" ou left="50"). Este atributo possui como valor padrão o valor zero.
- **top:** especifica a posição vertical da região definida e pode ser expressa tanto em porcentagem (left="10%") como em *pixels* (top="50px" ou top="50"). Este atributo possui como valor padrão o valor zero.
- **id:** atribui ao elemento um valor único.
- **title:** define um título para a região.

- **fit:** define o comportamento adotado caso a altura e a largura de um objeto extrapolar as dimensões definidas para uma região. Abaixo serão mostrados possíveis valores para este atributo:
    - **fill:** redimensiona o objeto automaticamente;
    - **meet:** redimensiona o objeto automaticamente porém, as proporções do objeto são mantidas.
    - **hidden:** o objeto será apresentado sem ser redimensionado. Este é o valor padrão para o atributo fit.

```
<smil id=”1”>
  <head>
    <layout>
      <root-layout id=”raiz” title=”Layout Raiz” width="400px" height="100px" background-
color="#CCCCCC" left=”10%” top=”50%” />
      <region id=”um” title=”Região” width="100px" height="20px" background-color =
"#CCCCCC" left=”100px” top=”50px” fit=”meet”/>
  </head>
</smil>
```

Figura 45: Utilização dos elementos root-layout e region

Este exemplo mostra a utilização dos elementos root-layout e region bem como a definição de alguns de seus atributos.

Elemento body

Todos os elementos responsáveis pela definição e sincronização entre os objetos de um documento SMIL fazem parte do conteúdo deste elemento.

Atributos	id
Conteúdo do elemento	(par | seq | (audio | video | text | img | animation | textstream | ref) | switch | a)*

Elementos par e seq

Estes elementos são os responsáveis por definir a sincronização de um conjunto de objetos. O elemento par especifica que cada objeto deste conjunto seja apresentado paralelamente entre si. Já o elemento seq define que os objetos pertencentes a este conjunto sejam apresentados seqüencialmente um ao outro.

Atributos (mais comuns)	id, begin, end, dur, repeat, region, endsync
Conteúdo do elemento	(par | seq | (audio | video | text | img | animation | textstream | ref) | switch | a)*

Cada elemento par ou seq pode conter os mesmos elementos que o elemento body e ambos possuem um conjunto de atributos que são usados para sincronização dos objetos, tais como:

- **id:** atribui ao elemento um valor único;
- **begin:** indica o instante em que um determinado objeto deve iniciar sua apresentação;
- **end:** define o instante em que a apresentação de um objeto será encerrada;
- **dur:** especifica a duração da apresentação de um objeto;
- **repeat:** define quantas vezes a apresentação de um objeto será repetida. O valor padrão para este atributo é um;
- **region:** especifica o local onde será apresentado o objeto. Na definição do cabeçalho, através do elemento head, podem ser definidas as regiões, através do elemento region, e cada um destas regiões possui um atributo id relacionado, o qual será definido um valor único, e é exatamente este valor usado para identificar o local onde será apresentado o objeto.

Através dos atributos descritos acima, um documento SMIL pode ser corretamente sincronizado. Isto porque, para cada objeto pertencente a uma apresentação multimídia, pode definir o início, duração e fim de sua exibição. Pode-se também definir sua disposição espacial e por quantas vezes será reapresentado.

Elementos multimídia: audio, video, text, img, animation, textstream, ref

Estes elementos são os responsáveis por definir a inclusão dos objetos multimídia em um documento SMIL.

Atributos (mais comuns)	id, begin, end, dur, repeat, region, src, alt
Conteúdo do elemento	anchor

Todos os elementos multimídia são compostos por um conjunto de atributos que especificam suas configurações e possuem o elemento anchor que permite que um objeto multimídia faça uma ligação com partes de um outro objeto.

Os atributos responsáveis por suas configurações são:

- **id, begin, end, dur, repeat, region:** estes atributos foram apresentados na especificação dos elementos: seq e par;
- **src:** define o local onde o arquivo multimídia se encontra;
- **alt:** especifica um texto para ser apresentado caso o objeto multimídia não possa ser apresentado.

Na seção seguinte serão apresentados diversos exemplos de utilização dos elementos e atributos vistos até aqui. Nestes exemplos será abordada tanto a sincronização entre os objetos como também sua sincronização espacial.

Exemplos de utilização dos Elementos

```
<smil>
  <head>
    <layout>
      <root-layout width="300" height="300" background-color="#CECCB9" />
      <region id="um" left="0" top="0" width="20" height="20" />
      <meta name="title" content="Apresentação - SMIL"/>
    </layout>
  </head>
  <body>
    <img src="bolaAzul.gif" region="um"/>
  </body>
</smil>
```

Figura 46: Utilização dos elementos de cabeçalho

O exemplo acima (Figura 46) apresenta apenas uma imagem, não levando em consideração questões de sincronização. O documento SMIL inicia com o elemento raiz <smil> composto por outros dois elementos, onde <head> é o responsável pela sincronização espacial, definindo informações acerca da apresentação (locais onde serão apresentados os objetos, título da apresentação, entre outras) e <body> responsável pela definição dos objetos a serem apresentados e sincronizados. O elemento <head>, por sua vez, possui o elemento <root-layout> que especifica a área da apresentação, o elemento <region> que define uma região o qual poderá ser visualizado um objeto multimídia e um elemento <meta> que define o título da apresentação. Já o elemento <body> possui apenas um elemento <img> que define a apresentação da imagem na região identificada por "um", através do atributo region="um".

Ao iniciar a apresentação, a imagem aparece e some logo em seguida. Isto porque, apesar de não ter sido definida a duração da apresentação, o atributo dur (responsável pela definição do tempo de duração da apresentação de um objeto) é um atributo implícito e possui um valor padrão e igual a um.

```
<smil>
 <head>
  <layout>
   <root-layout width="300" height="300" background-color="#CECCB9" />
   <region id="um" left="0" top="0" width="20" height="20" />
   <region id="dois" left="50" top="0" width="20" height="20" />
   <region id="tres" left="100" top="0" width="20" height="20" />
   <region id="quatro" left="150" top="0" width="20" height="20" />
   <region id="cinco" left="200" top="0" width="100" height="70" />
  </layout>
 </head>
 <body>
  <seq>
   <img src="bolaAzul.gif" region="um" begin="1s" dur="2s" />
   <img src="quadradoVerde.gif" region="dois" begin="0s" dur="2s" />
   <audio src="audio1.mp3" region="tres" begin="0s" dur="2s" />
   <img src="quadradoAzul.gif" region="quatro" begin="1s" dur="2s" />
   <img src="todos.gif" region="cinco" begin="2s" dur="2s"/>
  </seq>
 </body>
</smil>
```

Figura 47: Utilização do elemento seq e dos atributos begin e dur

Este exemplo (Figura 47) possui o objetivo de apresentar um conjunto de objetos de forma sincronizada. Este conjunto de objetos é relacionado através do elemento <seq>, os quais serão apresentados seqüencialmente. Na ordem como foram definidos, cada objeto será apresentado levando em consideração o instante definido para o início de sua apresentação (através do atributo begin), durante um certo intervalo de tempo (definido pelo atributo dur) e em uma determinada região (definido pelo atributo region). Ao final do período estabelecido para a apresentação deste objeto, o mesmo desaparecerá e será a vez do próximo objeto da lista ser apresentado.

```
<smil>
 <head>
  <layout>
   <root-layout width="400" height="300" background-color="#CECCB9" />
   <region id="um" left="0" top="0" width="20" height="20" />
   <region id="dois" left="50" top="0" width="20" height="20" />
   <region id="tres" left="100" top="0" width="20" height="20" />
   <region id="quatro" left="150" top="0" width="20" height="20" />
   <region id="cinco" left="200" top="0" width="100" height="70" />
  </layout>
 </head>
 <body>
  <par end="10s">
   <img src="bolaAzul.gif" region="um" begin="1s" dur="5s" />
   <img src="quadradoVerde.gif" region="dois" begin="2s" dur="3s" />
   <audio src="audio1.mp3" begin="0s" dur="15s"/>
   <img src="quadradoAzul.gif" region="quatro" begin="1s" dur="4s" />
   <img src="todos.gif" region="cinco" begin="4s" dur="5s" end="5"/>
  </par>
 </body>
</smil>
```

Figura 48: Utilização do elemento par e do atributo end

Este exemplo, ao contrário do exemplo anterior, apresenta os objetos paralelamente. Independentemente da ordem em que foram definidos, todos os objetos iniciarão sua apresentação, obedecendo ao que tiver sido definido em seus atributos de sincronização. Passados dez segundos, mesmo que os objetos não tenham sido encerrados, a aplicação será finalizada. Isto por causa do atributo end definido no elemento par (<par end="10s">).

```
<smil>
 <head>
  <meta name="title" content="Apresentação - SMIL" />
  <layout>
   <root-layout width="300" height="300" background-color="#CECCB9" />
   <region id="um" left="130" top="130" width="20" height="20"/>
  </layout>
 </head>
 <body>
  <par>
   <img src="bolaAzul.gif" region="um" begin="0s" dur="5s" repeat="2"/>
   <audio src="audio1.mp3" begin="0s" dur="10s" />
  </par>
 </body>
</smil>
```

Figura 49: Utilização do elemento par e do atributo repeat

Este exemplo (Figura 49) apresenta um áudio e uma imagem paralelamente. A imagem, através do atributo repeat, será repetida duas vezes, enquanto que a apresentação do áudio será repetida uma única vez.

```
<smil>
 <head>
  <meta name="title" content="Apresentação - SMIL" />
  <layout>
   <root-layout width="300" height="300" background-color="#CECCB9" />
   <region id="um" left="130" top="130" width="20" height="20"/>
  </layout>
 </head>
 <body>
  <par>
   <img src="bolaAzul.gif" id="img" region="um" begin="0s" dur="4s"/>
   <audio src="audio1.mp3" id="audio" begin="id(img)(2s)" dur="10s" />
  </par>
 </body>
</smil>
```

Figura 50: Utilização do elemento par e do atributo begin

O exemplo define a apresentação sincronizada de uma imagem e um áudio. Logo que a apresentação for iniciada, a imagem será apresentada. O áudio somente iniciará sua apresentação dois segundos após o início da apresentação da imagem. Em (W3C, 2000) é definido que o modelo de especificação adotado pelo SMIL corresponde ao baseado em eixos de tempos. Mas neste exemplo a apresentação de um objeto é condicionada a apresentação de um outro, o que indica uma especificação baseada em relação de intervalos (apresentado na seção 2.4.1). No caso deste exemplo, o áudio inicia sua apresentação durante a apresentação da imagem, a imagem encerra sua apresentação e o áudio continua até que também seja finalizado, o que representa a relação A Overlaps B.

O SMIL, da mesma forma que o HTML+TIME, permite que objetos pertencentes a uma apresentação sejam sincronizados. Este padrão, de acordo com sua especificação, trabalha com o método de sincronização baseados em eixos. No entanto, como foi apresentado na Figura 40, pode-se também trabalhar com sincronização baseada em intervalos. Ao contrário do HML + TIME, o SMIL, através de seu DTD, implementa tanto os elementos e atributos responsáveis pela sincronização como os responsáveis pela apresentação. Na próxima seção serão abordados os materiais e métodos usados para que este trabalho fosse definido.

As funcionalidades oferecidas pelo HyTime para a especificação de escalonamento e sincronização dos objetos de mídia que compõem um hiperdocumento estão descritas neste capítulo. As formas arquiteturais aqui descritas possibilitam a construção de DTDs com enfoque principal nas atividades de escalonamento e sincronização de objetos multimídia; documentos que necessitem desse enfoque podem ser produzidos com a utilização destes modelos.

Os padrões de sincronização multimídia apresentados possuem algumas características que os aproximam e outras que os diferenciam. Uma característica que os aproximam corresponde a forma como são definidos: os dois utilizam o padrão XML, para especificar os seus elementos e atributos de sincronização (HTML+TIME e SMIL) e apresentação (SMIL).

Outra característica que os aproxima corresponde ao surgimento de suas novas versões. Ambos estão na versão 2.0 e incorporaram algumas propriedades um do outro. No caso do HTML+TIME 2.0, foram acrescentados inúmeros elementos existentes no SMIL 1.0, e um exemplo corresponde à presença do elemento animation, ausente na sua versão anterior, e que está disponível no SMIL desde sua primeira especificação. A recíproca também é válida, e um exemplo é a presença do atributo timeAction, presente na versão 1.0 do HTML+TIME e agora acrescentada ao SMIL.

Tanto o SMIL como o HTML+TIME se adequam ao modelo de sincronização baseado em eixos de tempo. Isto porque oferecem recursos que permitam determinar o início, duração e fim da apresentação de um objeto. No entanto, os mesmos, não cumprem inteiramente aos conceitos existentes no modelo baseado em relação de intervalos, como poderá ser visto nas seções 3.5 e 3.6.

Uma característica importante e que na teoria diferencia estes dois padrões corresponde ao modelo de sincronização utilizado. O SMIL, de acordo com sua referência (W3C, 2000), trabalha com o modelo de sincronização baseado em eixos de tempo, porém na prática, ele permite que, através do atributo begin, a apresentação de um objeto seja condicionada a um outro (Figura 40), o que indica um modelo baseado em intervalos. Já o HTML+TIME, define a utilização de dois modelos: baseado em intervalos e em eixos de tempo.

Outra diferença entre estes padrões corresponde a forma como são divididas as responsabilidades: apresentação visual e sincronização. O SMIL, define tanto os elementos e atributos responsáveis pela sincronização como os responsáveis pela apresentação visual dos objetos. Já o HTML+TIME divide as tarefas, deixando a apresentação dos objetos a cargo do HTML e a sincronização por conta do HTML+TIME.

Uma consideração importante corresponde a forma como estes padrões trabalham com a sincronização espacial dos objetos. O SMIL trabalha com a idéia de regiões para fazer a sincronização espacial, enquanto que o HTML+TIME deixa esta tarefa por conta das *tags* HTML.

O trabalho aqui desenvolvido baseou-se principalmente nas especificações oficiais dos padrões e linguagens utilizados para análise. Assim, trabalhou-se sobre a especificação do padrão HyTime, conforme estabelecido pela ISO, e das linguagens SMIL e HTML+TIME, conforme disponibilizado pela W3C.

A análise foi realizada ainda através da verificação das funcionalidades destes padrões e linguagens através de *parsers* e validadores, alguns destes já integrantes de softwares e ferramentas disponíveis na Web:

- Internet Explorer 6.0 – para a validação e apresentação de aplicações baseadas em HTML+TIME e SMIL;
- RealPlayer Plus – para a validação e apresentação de aplicações baseadas em SMIL;
- HTML Validator – para a validação de páginas baseadas em HTML;
- XMLWriter – validação de páginas baseadas em XML, que foram trabalhadas como um subconjunto SGML, atuando assim como parser SGML.

A partir da especificação, validação e testes sobre aplicações desenvolvidas nestas linguagens pode-se chegar a delimitação de elementos e atributos necessários para uma meta-linguagem que se proponha a atender os requisitos de sincronização multimídia.

O próximo capítulo descreve as meta-linguagens desenvolvidas utilizando-se das facilidades aqui apresentadas e procurando estendê-las.

4 A Meta-linguagem Proposta

A seguir são apresentados os construtores, modelados como elementos XML, e seguindo a sintaxe de SGML e HyTime, que fazem parte da meta-linguagem proposta para a classe de hiperdocumentos que trabalha com escalonamento e sincronização de mídias.

Esta classe de hiperdocumentos utiliza formas arquiteturais do *Base Module*, do *Measurement Module* e do *Scheduling Module*, podendo ser classificada, então, como um *Minimal Scheduling HyTime document*. De acordo com a proposta de modificação do padrão HyTime, o *Base Module* passa a agregar os construtores anteriormente definidos para o *Measurement Module*, que deixa de existir.

O meta-DTD proposto inicialmente desenvolveu-se como três meta-DTDs, que contém construtores para o trato de especificação de sincronização hipermídia segundo os métodos de especificação de sincronização do modelo de referência de BLAKOWSKI (1996): baseado em intervalos, baseado em controle de fluxo e baseado em eventos. Cada meta-DTD é estruturado para um método específico. Assim, tem-se respectivamente os meta-DTDs *Interval-based Scheduling Language* (ISched), *Petri-Net Scheduling Language* (PNSched) e *Event-based Scheduling Language* (ESched).

O método baseado em eixos de tempo não é descrito pois este é o método básico de especificação de sincronização do padrão HyTime, que se utiliza de eixos virtuais para o escalonamento de eventos em posições espaciais (eixos horizontal e vertical) e no tempo.

Os construtores definidos para os demais métodos de especificação de sincronização utilizam-se das formas arquiteturais do HyTime oferecidas para a especificação baseada em eixos virtuais. Estas formas arquiteturais são modificadas para permitir a especificação dos demais métodos.

De forma geral, os meta-DTDs aqui apresentados atuam como uma linguagem que permite a especificação de um hiperdocumento que integre diferentes mídias, com sincronização entre elas.

4.1 Sincronização baseada em intervalos

A especificação de sincronização através da definição de intervalos é realizada através do meta-DTD Isched, aqui proposto, que se utiliza, basicamente, das formas arquiteturais `extlist` e `dimref` do padrão HyTime.

Para a especificação de sincronização com base em intervalos, de acordo com o modelo apresentado na seção 2.4, as formas arquiteturais do HyTime `extlist` e `dimref` são suficientes. Assim, a definição dos elementos `location` e `sync`, apresentados na Figura 51, é realizada sob estes dois construtores, respectivamente.

```
<!element location (sync|#pcdata) >
<!attlist location
  HyTime    name      #fixed    extlist
  id        idref     #required      >

<!element sync empty >
<!attlist sync
  HyTime    name      #fixed    dimref
  elemcomp  cdata     #required
  selcomp   (first|last|qcnt)   qcnt
  flip      (flip|noflip)       noflip  >
```

Figura 51: Especificações de `location` e `sync` com `extlist` e `dimref`

O elemento `location` utiliza três conjuntos de pares de marcadores para definir estes intervalos nos eixos de escalonamento. Assim, como exemplifica a Figura 52, uma determinada mídia pode ser definida para ocupar a posição 10, com uma extensão 20, no eixo x; a posição 5, com uma extensão 15, no eixo y, e a posição 1, com extensão 100, em um eixo de tempo. Estes valores são virtuais, sendo codificados para valores reais em tempo de execução pela aplicação responsável pela apresentação.

```
<location id = evt1> 10 20 5 15 1 100 </location>
```

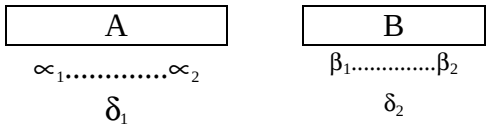
Figura 52: Exemplo da utilização do elemento `location` para escalonamento de um evento

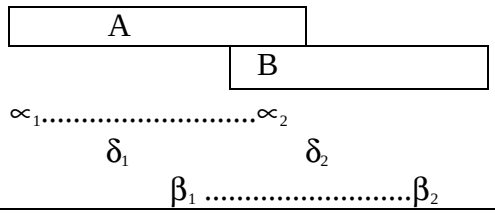
A especificação das relações entre as mídias, ou seja, a descrição de existência de intervalos entre as apresentações das mídias, pode ser realizada através do atributo `elemcomp` e `selcomp` do elemento `sync`. O atributo `elemcomp` indica que mídia (identificada pelo atributo `id` do elemento `location`) será referenciada. O atributo `selcomp` indica qual a forma de relacionamento que há entre as apresentações das duas mídias, através da definição, para este atributo, de algum desses valores:

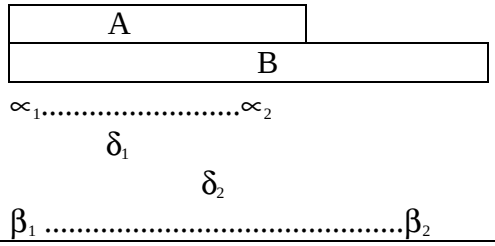
- *first*, que indica o primeiro marcador da dimensão;
- *last*, que aponta o último marcador; e
- *qcnt*, que seleciona o tamanho da dimensão.

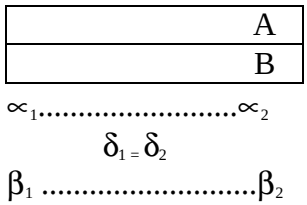
Através dos elementos `location` e `sync` pode-se construir todas as especificações de sincronização baseadas em intervalos descritas em (ALLEN, 1983). Seguindo o conceito de eixos virtuais que o HyTime utiliza para escalonamento de eventos, os elementos citados necessitam que sejam definidas as posições que estes ocuparão nos eixos. Assim, para cada evento é feita a especificação de sua posição inicial e de sua duração.

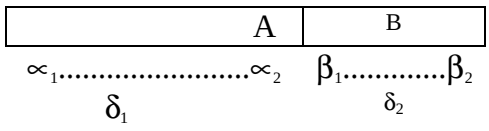
A Tabela 4 apresenta a utilização de `location` e `sync` com a apresentação de instâncias de documento que especificam as treze relações temporais baseada em intervalos.

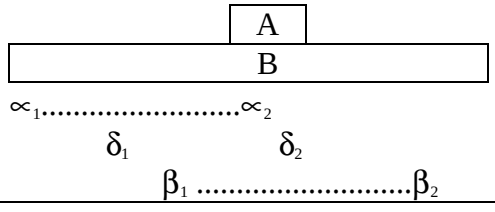
A *before* B		`<location id = evtA>`
		α_1 δ_1
		`</location>`
		`<location id = evtB>`
		β_1 δ_2
		`</location>`

A *overlaps* B		`<location id = evtA>`
		α_1 δ_1
		`</location>`
		`<location id = evtB>`
		β_1 δ_2
		`</location>`

A *starts* B		`<location id = evtA>`
		α_1 δ_1
		`</location>`
		`<location id = evtB>`
		`<sync elecomp = evtA selcomp = first>` δ_2
		`</location>`

A *equals* B		`<location id = evtA>`
		α_1 δ_1
		`</location>`
		`<location id = evtB>`
		`<sync elecomp = selcomp = first>`
		`<sync elecomp = evtA selcomp = qcnt>`
		`</location>`

A *meets* B		`<location id = evtA>`
		α_b δ_1
		`</location>`
		`<location id = evtB>`
		`<sync elecomp = evtA selcomp = last>` δ_2
		`</location>`

A *during* B		`<location id = evtA>`
		α_1 δ_1
		`</location>`
		`<location id = evtB>`
		β_b δ_2
		`</location>`

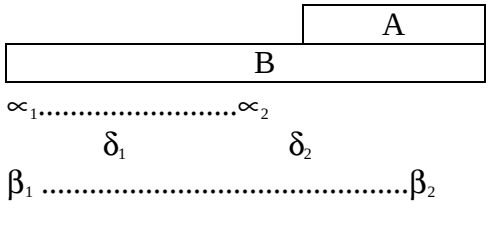
A *finishes* B		`<location id = evt1>`
		α_1 δ_1
		`</location>`
		`<location id = evt2>`
		β_b
		`<sync elecomp = evt1 selcomp = last>`
		`</location>`

Tabela 4: As treze relações temporais de sincronização baseadas em intervalos especificadas com `location` e `sync`

A especificação de elementos pode incluir a presença de parâmetros de atraso, utilizados na definição de operações de sincronização que estabelecem relações entre o instante inicial e o instante final dos eventos, como são as operações do modelo avançado de sincronização baseado em intervalos, descritas em (WAHL, 1994) e apresentadas na seção 2.4. Neste sentido, apresenta-se a definição de um novo elemento, `opersync` (Figura 53), de modo a estender os elementos `location` e `sync`.

Para o elemento `opersync` são especificados os atributos `oper`, `delta1`, `delta2`, `delta3` e `ext`. O atributo `oper` indica a operação de sincronização desejada, e `delta1`, `delta2`, `delta3` indicam o(s) intervalo(s) para a realização das operações de sincronização.

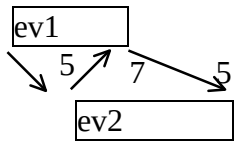
Os valores para `delta2` e `delta3` serão necessários somente quando `oper` indicar operação com dois ou três parâmetros, respectivamente. Os valores desses atributos serão interpretados para definição das dimensões e do posicionamento dos eventos em uma escala de eventos. Foram realizadas alterações também nos elementos `location` e `sync`, anteriormente apresentados, de forma a suportar a definição de `opersync`.

```
<!element location (sync|#pcdata)>
<!attlist location
  HyTime    name      #fixed    extlist
  id        id        #required      >

<!element sync     (opersync)>
<!attlist sync
  HyTime    name      #fixed    dimref
  elemcomp  cdata     #required      >

<!element opersync empty>
<!attlist opersync
  oper      (before|beforeendof| cobegin|coend|
             while|delayed|startin|endin|cross|
             overlaps)              #required
  delta1    cdata                  #required
  delta2    cdata                  #implied
  delta3    cdata                  #implied  >
```

Figura 53: Especificação das operações do modelo avançado de sincronização `location, sync e opersync`

Com estes elementos pode ser realizada a especificação de relações temporais baseadas em intervalos segundo o modelo avançado descrito por (WAHL, 1994).

Uma instância de documento, que exemplifica a relação *overlaps* entre dois eventos, é apresentada na Figura 54.




```
<location id = evt1> 10 20 </location>
<location id = evt2>
    <sync elemcomp = evt1>
            <opersync oper = overlaps
                      delta1 = 5
                      delta2 = 7
                      delta3 = 5        >
    </sync>
     </location>
```

Figura 54: Exemplo da especificação da relação *overlaps*

4.2 Sincronização baseada em eventos

Quando, entre os eventos de mídia que estão se relacionando, existe algum cuja duração é desconhecida (como a interação com o usuário, por exemplo), o relacionamento não pode ser especificado com os construtores apresentados na seção 3.4, na forma como eles são descritos. Faz-se necessário que se utilize alguma forma de permitir que um evento sinalize a sua conclusão para possibilitar a ativação de outro que depende dele.

Esse método de especificação de sincronização, baseado em eventos, pode ser utilizado com a adição de alguns construtores como o elemento `action`, ou da possibilidade de assumirem-se valores desconhecidos (através de um marcador X, por exemplo) para a duração dos eventos.

O elemento `action`, utilizado junto com a especificação dos eventos, permite que o término do evento, ou a ação do usuário, exerça uma ação sobre outro evento. Para isso, a definição de evento (`slidevnt`) é determinada, no DTD, como apresenta a Figura 55, com a possibilidade de serem especificadas, além da localização, as ações que este evento pode exercer sobre outro.

```
<!element slidevnt  (image|animatn|audio, location,
action*) >
<!attlist slidevnt
     HyTime    name      #fixed         event
     Id        id        #implied       >
```

Figura 6.5: Definição, no DTD, de um evento

A especificação de eventos deste tipo, então, passa a ser feita com o posicionamento do evento no eixo temporal, através do elemento `location`. Os demais elementos, `sync` e `opersync`, continuam sendo utilizados, com o acréscimo do elemento `action`, como apresentado na Figura 56.

```
<!element location   (sync|#pcdata)>
<!attlist location
     HyTime    name      #fixed          extlist
     idevt idref #required            >

<!element sync  (opersync) >
     HyTime    name      #fixed    dimref
     elecomp   cdata #required  >

<!element opersync empty >
<!attlist opersync
     oper      < - - operações com um parâmetro - - >
               (before|beforeendof|cobegin|coend|
               <- - operações com dois parâmetros - - >
               while|delayed|startin|endin|cross|
               <- - operações com três parâmetros - - >
               overlaps)  >


<!element action     empty >
<!attlist action
     <- - o atributo when indica se a ação deverá ser efetuada
     quando o evento iniciar ou quando ele terminar - ->
```

```
when       (begin|end)     #required

           <- - o atributo delta indica que a ação deverá ser
      efetuada delta tempo "após" o início ou fim do evento, de
      acordo com o que o atributo when indicar. Caso o valor de
      delta não seja fornecido, a aplicação entenderá como delta
      = zero - ->

delta      cdata      #implied

           <- - como ações possíveis temos somente a ativação
      (init) ou o encerramento (stop) (através do atributo what)
      do evento referenciado através do atributo who - ->


what       (init|stop)     #required

      <- - o atributo who indica qual evento será "atingido"
      pela ação - - >

who        idref      #implied>
```

Figura 56: Especificação de operações com suporte a ações (eventos)

Assume-se que a utilização de um marcador diferente de um número (especificamente, X), denotará desconhecimento da duração do evento, como explicita a Figura 57. A utilização de X deve ser interpretada pelo tradutor.

```
<slidevnt id = "evt1">

     <audio  id       = "aud1"
             filename = "aud1"
             format   = "au">

     <location> 1 x </location>

     <action when = "end"
             delta = 10
             what  = "init"
             who   = "evt2">

</slidevnt>
```

Figura 57: Especificação de um evento de duração desconhecida

Como pode-se observar, há um outro evento, `evt2`, que depende do término do evento `evt1` para poder iniciar, após um *delay* de 10 unidades de tempo. Isso é

indicado através dos atributos `when` (início ou fim do evento em questão), `what` (que ação realizar sobre o outro evento: iniciá-lo ou encerrá-lo), `who` (quem é o outro evento, sobre o qual a ação será exercida) e `delta` (que define um *delay* para a realização da ação).

Assim, tem-se um evento, `evt2`, cujo início é desconhecido pois depende de um evento de duração desconhecida. A especificação deste evento pode ser realizada como a Figura 58 apresenta. Entende-se que a não especificação de um valor para seu início está diretamente ligada à necessidade de se ter especificado um outro evento que atuará sobre ele.

```
<slidevnt id = “evt2”>

     <audio id       = “aud2”
            filename = “aud2”
            format   = “au”     >

     <location>
          x 20
     </location>

</slidevnt>
```

Figura 58: Especificação de um evento dependente de outro

São descritas abaixo duas instâncias de documentos que, através da utilização de elementos diferentes (na primeira, `opersync`; na segunda, `action`) obtém o mesmo efeito sobre os eventos

```
<slidevnt id = “evt1”>
     <audio id = ”aud1” filename = “aud1” format = “au”>
     <location> 1 x </location>
</slidevnt>

<slidevnt id = “evt2”>
     <audio id = “aud2” filename = “aud2” format = “au”>
     <location>
          <sync elemcomp = “evt1”  oper = “before”
           delta1 = 10> 20
     </location>
```

Pode-se considerar que a presença de **x** indica desconhecimento da extensão do objeto

```
<slidevnt id = “evt1”>
     <audio id = ”aud1” filename = “aud1” format = “au”>
     <location> 1 x </location>
          <action when = “begin” delta = 10 what = “init”
                  who = “evt2”>
</slidevnt>

<slidevnt id = “evt2”>
     <audio id = “aud2” filename = “aud2” format = “au”>
     <location>
          x 20
     </location>
</slidevnt>
```

Figura 59: Instâncias de documentos com utilização de `opersync` e `action`

4.3 Sincronização baseada em controle de fluxo (Redes de Petri)

Dos modelos de sincronização baseados em controle de fluxo, optou-se por verificar a possibilidade de mapear os conceitos de Redes de Petri para HyTime, objetivando um futuro mapeamento com os modelos OCPN, TSPN, e HTSPN (LITTLE 1990) (SÉNAC 1995) (DIAZ 1993) (Wilrich 1996).

Assim, foi desenvolvido o meta-DTD *Petri-Net Scheduling Language* (PNSched) que, para a modelagem de Redes de Petri através dos construtores HyTime, utilizou-se dos elementos que permitem a especificação de lugares e transições. Os arcos são representados através de *links* entre os lugares e transições. A apresentação de um exemplo para o modelo de rede OCPN, descrito abaixo, torna mais clara essa idéia.

4.3.1 OCPN

Baseado no construtor `m-nlnk`, descrito na seção 3.4, foi definido o construtor `transition`, apresentado na Figura 60, que permite a especificação das transições de uma Rede de Petri. Um elemento definido como `transition`, com respectivos "*source*" e "*target*" definidos, ao ser acessado por todos os *links* "*source*" (ou seja, os lugares indicados como *source* tiverem sido executados), ativará todos os *links* "*target*" (ou seja, indicará o início da execução dos lugares indicados como *target*).

```
<!element transition  (nameloc+)  >

<!attlist transition
     HyTime    name      #fixed    "ilink"
     id        id        #implied
     anchrole  cdata     #fixed    "source #AGG target #AGG"
     linkends  idrefs    #required
     aggtrav   names     mem  >
```

Figura 60: Definição do elemento `transition`

Para definição do conteúdo dos lugares, ou seja, das mídias que serão apresentadas, tem-se o construtor `place`, apresentado na Figura 61, conformante com a forma arquitetural `event`, que permite a especificação de um evento de mídia, que pode ser do tipo áudio, animação ou imagem.

```
<!element place        (image|animatn|audio) >

<!attlist place
        HyTime        name          #fixed          event
        id            id            #implied                >
```

Figura 61: Definição do elemento `place`

Para permitir a especificação de uma Rede de Petri no formato OCPN, o construtor `place` passa a permitir a definição de uma duração, através do elemento `time`. Este elemento recebe um valor que indicará o tempo de execução do lugar. A Figura 62 apresenta a definição deste elementos.

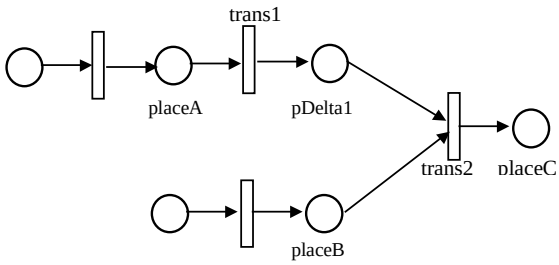
```
<!element place     ((image|animatn|audio),time?)>

<!attlist place
     HyTime    name      #fixed          event
     id        id        #implied        >

<!element time      (%marker|#PCDATA) >
```

Figura 62: Elementos `place` (modificado para OCPN) e `time`

A Figura 63 apresenta a definição de um elemento para especificação de uma Rede de Petri.. Assim, a Figura 64 apresenta uma Rede OCPN e sua representação através destes elementos.

```
<!element net  - -  (transition, place) >
<!attlist net
     id   id   #required >
```

Figura 63: Definição do elemento `net`




```
<net id = redeOCPN>
<transition id = trans1 linkends =”source1 target1” aggtrav=agg>
        <!- - âncoras origem da transição - ->
        <nameloc id = source1  aggloc = agglink>
                <nmlist nametype = element> placeA</nmlist>
        </nameloc>
        <!- - âncoras destino da transição - ->
        <nameloc id = target1 aggloc=agglink>
                <nmlist nametype = element> pDelta1 </nmlist>
        </nameloc>
</transition>

<transition id = trans2 linkends =”source2 target2” aggtrav=agg>
        <!- - âncoras origem da transição - ->
        <nameloc id = source2  aggloc = agglink>
                <nmlist nametype = element>pDelta1 placeB </nmlist>
        </nameloc>
        <!- - âncoras destino da transição - ->
        <nameloc id = target2 aggloc=agglink>
                <nmlist nametype = element> placeC </nmlist>
        </nameloc>
</transition>


<!- - descrição dos lugares - ->
<place id = placeA>
        <audio id = aud1 filename = aud1 format = au>
</place>

<place id = placeB>
        <audio id = aud2 filename = aud2 format = au>
</place>

<place id = pDelta1>
        <time> 10  </time>
</place>

</net>
```

Figura 64: Especificação de uma Rede OCPN

4.3.2 TSPN

A especificação HyTime de uma Rede TSPN pode ser realizada através da modificação dos elementos `transition` e `place`, para que estes suportem a nova modelagem de rede.

O elemento `transition` passa a contar com um atributo, `sync`, que indicará o tipo de estratégia de sincronização requerida (strong_or, weak_and, etc.). As transições indicam que tipo de sincronização deve ser realizada entre os lugares a elas relacionados (como lugares de entrada). Para isso, deve-se especificar um tipo para a transição entre os nove estratégias definidas para os modelos TSPN e HTSPN, através do atributo sync. Como tipo *default*, é determinado o tipo "normal", pois nem todas as transições estão envolvidas em relacionamentos entre mídias.

O elemento `place` passa a receber um `label` no lugar do elemento time. A alocação de um evento no lugar também é opcional, o que possibilita que sejam definidos lugares que atuarão somente como *timers* que realizarão a tarefa de sincronização entre os eventos.

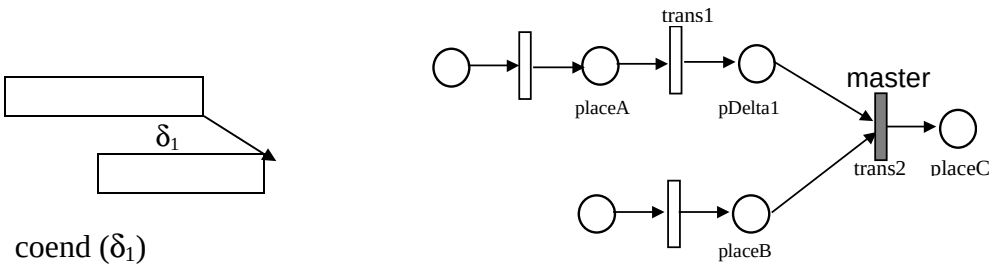
O lugar pode ser identificado como sendo do tipo "normal" ou "*master*", para efeito de tratamento das estratégias de sincronização que serão indicadas pelas transições. É indicado o valor "normal" como default, pois grande parte dos lugares não estão envolvidos em estratégias de sincronização, e mesmos estes não estarão relacionados, necessariamente, com estratégias do tipo "*master*".

O `label` é um elemento que contém três marcadores, sendo o primeiro para indicar o valor de α (tempo mais adiantado de liberação do *token*), o segundo indica o valor de η (duração ideal de apresentação) e o último para o valor de β (o tempo mais atrasado para liberação do *token*). Essas modificações são apresentadas na Figura 65.

```
<!element transition  (nameloc+) >
<!attlist transition
     HyTime     name       #fixed     “ilink”
     id         id         #implied
     anchrole   cdata      #fixed     “source #AGG target #AGG”
     linkends   idrefs     #required
     sync       (strong_or|weak_and|or|and|strong_master|
                 weak_master|or_master|and_master)      #implied
     aggtrav    names mem   >

<!element place - -    ((image|animatn|audio),label)>
<!attlist place
     HyTime     name            #fixed          event
     id         id              #implied
     type       (normal|master) normal               >

<!element label o o  (%marker|#PCDATA) >
```

Figura 65: Elementos `transition`, `place` e `label` para TSPN

O exemplo da Figura 64, especificado agora com um tipo para as transições, pode ser visualizado como a modelagem da relação temporal *coend*. Esta relação é modelada através da utilização da estratégia *strong_or*, que fará com que a finalização da apresentação de algum dos lugares force a finalização dos lugares a ele sincronizados. Esta modificação é apresentada na Figura 66.



coend (δ_1)

```
<net id = net1>
<transition id = trans1 linkends =”source1 target1” aggtrav=agg>
        <!- - âncoras origem da transição - ->
        <nameloc id = source1  aggloc = agglink>
                <nmlist nametype = element> placeA</nmlist>
        </nameloc>
        <!- - âncoras destino da transição - ->
        <nameloc id = target1 aggloc=agglink>
                <nmlist nametype = element> pDelta1 </nmlist>
        </nameloc>
</transition>

<transition id = trans2 linkends =”source2 target2” aggtrav=agg_sync  = master >
        <!- - âncoras origem da transição - ->
        <nameloc id = source2  aggloc = agglink>
                <nmlist nametype = element>pDelta1 placeB </nmlist>
        </nameloc>
        <!- - âncoras destino da transição - ->
        <nameloc id = target2 aggloc=agglink>
                <nmlist nametype = element> placeC </nmlist>
        </nameloc>
</transition>


<!- - descrição dos lugares - ->
<place id = placeA>
        <audio id = aud1 filename = aud1 format = au>
</place>

<place id = placeB>
        <audio id = aud2 filename = aud2 format = au>
</place>

<place id = pDelta1 type = master >
        <label> x δ1 y </label>
</place>

</net>
```

Figura 66: Especificação da relação temporal *coend*

4.3.3 HTSPN

Retornando aos três conceitos do modelo Dexter que são perfeitamente suportados pelos modelo HTSPN, são descritas a seguir as formas como a meta-linguagem aqui definida consegue corresponder-se com este modelo. Para esta correspondência são utilizados os construtores descritos nas seções anteriores, com algumas modificações.

1. No modelo HTSPN um componente atômico é modelado por um arco com um IVT e um lugar do tipo atômico associado a um recurso.

O elemento `place`, como apresentado acima, pode ser associado a um recurso. As ligações entre os lugares e as transições, ou seja, o arco, são realizadas através da utilização de uma forma arquitetural do HyTime: o `ilink`, sobre o qual o elemento `place` baseia-se e com o qual conforma. A solução encontrada está em fornecer o IVT junto com o lugar onde o recurso está alocado. Para isso, o elemento `place` modificaria-se conforme a Figura 67, para adequar-se a necessidade de um IVT.

```
<!element place     ((image|animatn|audio)?,IVT?) >

<!attlist place
 HyTime         name       #fixed          event
 id             id         #implied        >

<!element IVT       o o  (marklist|#PCDATA) >
```

Figura 67: Definição do elemento `place` com a incorporação do IVT

Dessa forma, o elemento `IVT` pode ser fornecido como uma tupla contendo os três valores necessários para a definição do intervalo de validade temporal.

2. No modelo HTSPN uma ligação é modelada por um arco temporizado (L, t), onde L é um lugar do tipo ligação.

O fato do elemento `transition` ser definido com a utilização da forma arquitetural `nameloc` leva à possibilidade de se referenciar, além dos elementos `place`, outros elementos quaisquer definidos para a meta-linguagem. Isso é visível

através da definição do valor `element` para o atributo `nametype`, como apresentado na Figura 56, que permite que, onde estão definidos os elementos `placeA`, por exemplo, podem ser definidos outros elementos, como `link1`, conformante com a elementos correspondentes com a definição de ligações.

3. No modelo HTSPN um componente composto é modelado por um lugar abstrato, chamado lugar composto, que representa uma sub-rede.

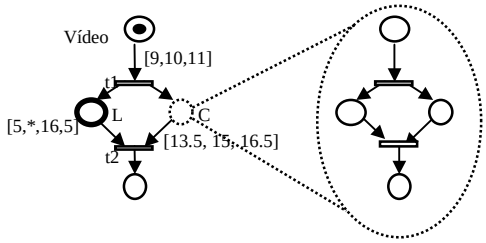
De forma análoga ao item anterior, a solução está na possibilidade de definir novas redes, indicadas pelos seus `ids`, na posição onde são definidos os lugares como `placeA` e `placeB`, por exemplo. Estas novas redes serão entendidas como subredes da rede principal.

A utilização da especificação de redes como sub-redes envolve a necessidade de indicar, para a rede que é definida com este propósito, intervalos de tempo como IVT. Essa necessidade é verificada através do lugar C da Figura 69. Para isso ser possível, o elemento net é modificado para aceitar especificação de IVT (Figura 68).

```
<!element net  ((transition, place+), IVT?)  >
<!attlist net
        id      id                      #required
        type    (master| normal)        normal>
```

Figura 68: Elemento net modificado

Deve-se considerar que a sub-rede especificada pode ser utilizada como elemento master para uma estratégia de sincronização. Para isso, é adicionado ao elemento net o atributo *type*, que pode assumir os valores *master* (para estratégias de sincronização) ou normal (valor default). A Figura 69 apresenta a utilização destes elementos.




```
<net id = net2>
<transition id = t1 linkends =”source1 target1” aggtrav=agg>
        <!- - âncoras origem da transição - ->
        <nameloc id = source1  aggloc = agglink>
                <nmlist nametype = element> video</nmlist>
        </nameloc>
        <!- - âncoras destino da transição - ->
        <nameloc id = target1 aggloc=agglink>
                <nmlist nametype = element> L C </nmlist>
        </nameloc>
</transition>

<transition id = t2 linkends =”source2 target2” aggtrav=agg>
        <!- - âncoras origem da transição - ->
        <nameloc id = source2  aggloc = agglink>
                <nmlist nametype = element> L C </nmlist>
        </nameloc>
        <!- - âncoras destino da transição - ->
        <nameloc id = target2 aggloc=agglink>
                <nmlist nametype = element> next </nmlist>
        </nameloc>
</transition>


<!- - descrição dos lugares - ->
<place id = video>
        <audio id = vid1 filename = vid1 format = avi>
        <IVT> 90 100 110 </IVT>
</place>

<place id = L>
        <hyperlink HyTime = clink linkend = otherframe> anchor text </hyperlink>
        <IVT> 50 10 165 </IVT>
</place>

<net id = C>
        <transition id ...
        <transition id...
        <place...
        <IVT>135 150 165</IVT>
</net>

</net>
```

Figura 69: Especificação de uma rede modelo HTSPN através de elementos HyTime

5 Considerações Finais

O desenvolvimento de novas tecnologias, especialmente no que se refere a utilização de várias mídias nos sistemas multimídia, desde as tarefas de armazenamento, comunicação e transmissão, e apresentação, leva a necessidade de oferecer recursos para garantir a correta apresentação destes sistemas, de acordo com o que for inicialmente especificado por seus desenvolvedores. As novas necessidades dos sistemas de computação, em questões de armazenamento, comunicação e apresentação de diversos formatos de dados, tornaram as tecnologias existentes insuficientes para que, sistemas multimídia, fossem corretamente sincronizados e apresentados. Esta insuficiência possibilitou diversos estudos acerca do tema, o que viabilizou a criação de linguagens que comportassem estas necessidades.

Assim, tem-se na literatura a apresentação de modelos de especificação de sincronização, especialmente o modelo de referência de BLAKOWSKI que define quatro formas de especificação: baseada em eixos de tempo, baseada em intervalos, baseada em controle de fluxo e baseada em eventos.

Apesar da existência formal deste modelo de referência, os padrões e linguagens criados para realizar tais especificações não os atendem em sua totalidade. O padrão HyTime baseia-se, em seu módulo de escalonamento, na especificação baseada em eixo de tempo, reais e virtuais. As linguagens HTML+TIME e SMIL trabalham basicamente a especificação baseada em eixos de tempo e, ainda que não apresentem formalmente tal característica, possuem elementos que permitem que se realize, em parte, a especificação baseada em intervalos.

A partir do trabalho com o padrão SGML, especialmente a partir da definição do XML como um padrão que permite a construção de linguagens passíveis de serem “estendidas”, tem-se elementos para a criação de uma linguagem de sincronização que atenda aos requisitos de sincronização como definidos no modelo de referência de BLAKOWSKI. Baseados nestes requisitos, e nas características das linguagens

estudadas, pode-se definir elementos que, posteriormente ampliados em suas funcionalidades, atendam às necessidades de sincronização multimídia bem como a outros requisitos de um hiperdocumento, como a navegabilidade, por exemplo.

Os meta-DTDs aqui apresentados oferecem três diferentes métodos de trabalhar com especificação de sincronização entre mídias com a utilização das facilidades HyTime. Foram definidos construtores que permitem que cada um dos métodos descritos em (BLAKOWSKI, 1996) possa ser utilizado em um documento especificado através do padrão HyTime. Tem-se, então, a possibilidade de trabalhar com relações temporais, com relações entre eventos, e com modelos baseados em Redes de Petri, mais precisamente OCPN, TSPN e HTSPN.

Com isso, estende-se as facilidades HyTime de tratamento de escalonamento e sincronização de eventos, que limita-se, no padrão, ao trabalho com eixos virtuais. Passa-se a poder trabalhar com especificações mais complexas, que consideram a viabilidade de trabalhar com eventos de duração desconhecida, dos quais pode-se destacar a interação com o usuário. Para tanto, observou-se muitas das características das linguagens SMIL e HTML+TIME.

Com os resultados apresentados, pode-se verificar a possibilidade de se desenvolver alguns trabalhos futuros:

- Verificação da viabilidade de se utilizar *stylesheets* desenvolvidos em XSL para realizar a conversão de uma apresentação especificada nesta meta-linguagem em um formato seguindo o SMIL ou o HTML+TIME, que já possuem *players* específicos;
- A implementação de um *player* para esta meta-linguagem;
- A comprovação, através de experimentos com diferentes projetos de rede, da manutenção dos requisitos de QoS através de especificações baseadas nesta meta-linguagem.

A meta-linguagem, como definida, oferece subsídios para que, em conjunto com outras áreas, seja possível desenvolver elementos que realizem a manutenção da qualidade dos serviços multimídia disponibilizados. Pode-se, por exemplo, utilizar os valores especificados para cada elemento, durante o desenvolvimento de uma aplicação,

para através de técnicas de Inteligência Artificial realizar processos de tomada de decisão antes mesmo que eventuais problemas de comunicação venham a ser observados e confirmados.

Tem-se, então, com esta meta-linguagem, a possibilidade de estender as características do HTML+TIME e do SMIL, aproveitando várias das características do HyTime que não foram exploradas em outras aplicações. Da mesma forma, dada a característica de ser baseada em SGML, criada a partir de elementos XML, esta meta-linguagem também pode ser estendida, atendendo a outros requisitos de desenvolvimento de aplicações multimídia que não sejam relacionados unicamente a sincronização.

6 Referências Bibliográficas

ALLEN, J. F. Maintaining knowledge about temporal intervals. **Communications of the ACM**, 1983.

BLAKOWSKI, G., STEINMETZ, R. A media synchronization survey: reference model, specification and case studies. **IEEE JSAC**, v. 4, n.1, pp. 5-35, jan. 1996.

BROWN, H. Standards for structured documents. **Computer**, p. 505-514, jun. 1989.

BUFORD, J. F. K. et al. **Multimedia Systems**. ACM Press, 1994.

CLARK, J. **An SGML parser**. Disponível em http://www.jclark.com/sp.html, 1996.

DEROSE, S. J., DURAN, D. G. **Making Hypermedia Work**: A Users's Guide to HyTime. Kluwer Academic Publishers, Massachusetts, 1994.

GOLDFARD, Charles F.. **The SGML Handbook**. Oxford University Press, Oxford, 1995.

van HERWIJNEN, E. **Practical SGML**. Kluwer Academic Publishers, 2. ed., 1994

ISO/IEC 8879 Information Processing — Text and Office Systems — Standard Generalized Markup Language. Out. 1986.

ISO/IEC88744 Hypermedia/Time-Based Structuring Language-HyTime, 1992.

LITTLE, T. D. C., GHAFOOR, A. Synchronization and storage models for multimedia objects. IEEE JSAC, v. 8, pp. 413-426, nov. 1990.

LITTLE, T. D. C., GHAFOOR, A. Interval-Based Conceptual Models for Time-Dependent Multimedia Data. IEEE Transactions on Knowledge and Data Engineering (Special Issue: Multimedia Information Systems), vol. 5 n. 4, pp. 551-563, agosto 1993.

MICROSOFT. **Introduction to HTML+TIME**. Disponível em <http://msdn.MICROSOFT.com/library/default.asp?url=/workshop/author/behaviors/time.asp>, 2002. Acesso em 25/06/2002.

MURATA, T. Petri nets: properties, analysis and applications; **Proceedings of IEEE**, vol.77, no.4, pp.541--580, 1989.

NAHRSTEDT, K., SMITH, J. M. The QOS Broker. **IEEE Multimedia**, v.2, n.1, p. 53-67, 1995.

NEWCOMB, S.; KIPP, N.; NEWCOMB, V. The "HyTime" Hypermedia/Time-Based Document Structuring Language. **Communications of the ACM**, v. 34,. n. 11, pp. 67-83, 1991.

RODRIGUES, Rogério Ferreira. **Formatação Temporal e Espacial no Sistema Hyperprop**, Dissertação (Mestrado em Informática) 1997. Departamento de Informática da Universidade PUC-Rio. PUC-Rio, Rio de Janeiro. 116p.

RUTLEDGE, Lloyd. **A HyTime Engine for Hypermedia Document Presentation**. Dissertação de Mestrado, Universidade de Massachusetts Lowell, 1993.

RUTLEDGE, L., BUFORD, J. F., RUTLEDGE, J. L. Presenting HyTime Documents with HTML. **EDMEDIA'96**

SCHMITZ, Patrick; YU, Jin; SANTANGELI, Peter. **Timed Interactive Multimedia Extensions for HTML**. Disponível em <http://www.w3.org/TR/NOTE-HTMLplusTIME>, 1998. Acesso em 25/06/2002.

SOARES, L. F. G., CASANOVA, M. A., COLCHER, S. An architecture for hypermedia systems using MHEG standard objects interchange. **Proceedings of the Workshop on Hypermedia and Hypertexts Standards**. Amsterdan, abril 1993.

STEINMETZ, R., NAHRSTEDT, K. Resource Management in Networked Multimedia Systems. **IEEE Computer**, maio 1995.

VAZIRGIANNIS, M., MOURLAS, C. An object-oriented model for interactive multimedia presentations. **The Computer Journal**, v. 36, n. 1, pp. 78-86, 1993.

VAZIRGIANNIS, M., MOURLAS, C. Integrated Multimedia Object and Application Modelling Based on Events and Scenarios. **Proceedings of the Int. Workshop on Multi-Media Database Management Systems**, agosto 1995, pp. 48-55.

W3C – World Wide Web Consortium. **eXtensible Markup Language (XML).** Disponível em <http://www.w3.org/XML/>, 1998. Acesso em 25/06/2002.

W3C – World Wide Web Consortium. **eXtensible StyleSheet Language (XSL).** Disponível em <http://www.w3.org/Style/XSL/>, 1998. Acesso em 25/06/2002.

W3C – World Wide Web Consortium. **Synchronized Multimedia Integration Language 1.0 (SMIL).** Disponível em <http://www.w3.org/AudioVideo>, 2000. Acesso em 25/06/2002.

WAHL, T., ROTHERMEL. K. Representing Time in Multimedia-Systems. In Proc. **IEEE International Conference on Multimedia Computing and Systems**, Boston, Maio 1994.