UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Elisa Maria Pivetta Cantarelli

ANÁLISE E PROPOSTA DE EXTENSÃO DA LINGUAGEM SMIL 2.0 PARA INSERÇÃO DE APLICAÇÕES MULTIMÍDIA NA WEB

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

Prof. Roberto Willrich, Dr

Florianópolis, agosto de 2002

ANÁLISE E PROPOSTA DE EXTENSÃO DA LINGUAGEM SMIL 2.0 PARA INSERÇÃO DE APLICAÇÕES MULTIMÍDIA NA WEB

Elisa Maria Pivetta Cantarelli

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

Prof. Dr. Fernando Álvaro Ostuni Gauthier

Banca Examinadora

Prof. Dr. Roberto Willrich (Orientador)

Prof. Dr. Vitório Bruno Mazzola

Prof. Dr. Rosvelter João Coelho da Costa

Este trabalho é dedicado

Ao meu querido e amado esposo Luiz Antônio,
por todo o amor, carinho e incentivo.

Aos meus filhos, Marlon e Margel cujo amor,
carinho e compreensão sustentaram e estimularam
esta caminhada. Amo muito vocês.

E a Deus, nosso guia em todo o momento.

Agradecimentos

Ao professor, orientador Roberto Willrich, por acreditar em mim e pela atenção que dedicou a este trabalho.

Aos meus amigos e colegas, pela amizade, incentivo e apoio.

A URI – Universidade Regional Integrada - Campus de Frederico Westphalen, pelo auxílio prestado.

Aos colegas professores da URI, que de alguma forma contribuíram para a realização deste trabalho.

A minha família, pais, cunhados, sobrinhos, irmãs, e em especial meu marido e meus filhos, os quais não cansarei de agradecer.

SUMÁRIO

TABELAS

FIGURAS

SIMBOLOGIA

ADSL	Asymmetric Digital Subscriber Line
API	Application Programming Interface
CGI	Common Gateway Interface
CSS	Cascading Style Sheet – Folha de Estilo em Cascata
DOM	Document Object Model
DTD	Document Type Definition
FTTC	Fiber to The Curb
GriNS	Graphical iNterface to SMIL
HFC	Hybrid Fiber Coax
HTML	HyperText Markup Language
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ISO	International Standards Organization
MCU	Multicast Control Units
MHEG	Multimedia Hypermedia Expert Group
MPEG	Motion Picture Experts Group
MP3	MPEG 1 camada 3 ou Motion Picture Experts Group 1 camada 3
PDF	Portable Document Format
RSTP	Real Time Streaming Protocol
RTP	RTP Control Protocol
SAX	Simple API for XML
SDV	Switched Digital Video
SGML	Standard Generalized Markup Language
SMIL	Synchronized Multimedia Integration Language
SVG	Scalable Vector Graphics
TCP	Transmission Control Protocol
UML	Unified Modeling Language
URI	Universal Resource Identifier
URL	Universal Resource Locator
W3	World Wide Web
W3C	World Wide Web Consortium
WWW	World Wide Web
XHTML	XML com HTML 4.0
Xlink	XML Linking Language
XML	Extensible Markup Language
Xpointer	XML Pointer Language
XSL	XML Stylesheet Language

Resumo

SMIL (*Synchronized Multimedia Integration Language*) é a linguagem proposta pelo W3C (*World Wide Web Consortium*) para a inserção de cenários multimídia interativos em páginas Web, permitindo a orquestração de mídias estáticas e dinâmicas. A sincronização intramídia e intermídia exige que o sistema de comunicação atenda certos requisitos, como, largura de banda, capacidade de armazenamento, alta taxa de transferência, limitação e variação de atraso, sincronização espacial e temporal. A internet oferece um serviço do tipo melhor esforço, tornando muitas vezes as sincronizações previstas pelo autor da aplicação inviáveis de serem respeitadas. É necessário que os modelos de autoria proporcionem ao autor mecanismos para definir níveis de tolerância de sincronização e tratamentos de exceção, bem como a possibilidade de reutilização dos componentes, facilitando a autoria. Este trabalho descreve os requisitos para dispor documentos multimídia na Web e analisa se a linguagem SMIL 2.0 atende estes requisitos. Propõem-se também algumas extensões para a linguagem SMIL 2.0, de forma a se adequar melhor a estes requisitos e ao indeterminismo temporal do sistema de comunicação. SMIL é baseada na linguagem XML *(Extensible Markup Language)*, uma metalinguagem para a criação de novas linguagens de marcação. A extensão de SMIL é proporcionada usando namespace XML.

Palavras-chave: Multimídia, Internet, Sincronização, SMIL, XML.

Abstract

SMIL (Synchronized Multimedia Integration Language) is the language proposed by the W3C (World Wide Web Consortium) for the insertion of interactive multimedia scenarios in Web pages, allowing the orchestration of static and dynamic media. Inter-media and intra-media synchronization demands that the communication systems attend certain requirements, such as bandwidth, storing capacity, high transference tax, delay and jitter, spatial and temporal synchronization. Internet offers a kind of better effort service, becoming many times the synchronization previewed by the author of enviable application to be respected. It is necessary that, the models of authory provide to the author some mechanisms to define levels of tolerance of synchronization and treatment of exception, thus the possibility of reutilization of the components, making easier the authory. This work describes the request to make available multimedia in the Web and analyses if SMIL 2.0 language supports these requests. It also proposes some extensions to SMIL 2.0 language, to a better adequation to the request and to the temporal indeterminism of communication system. SMIL is based on the language XML (Extensible Markup Language) a metalanguage to the creation of new languages of marking. The extension of SMIL is provided using namespace XML.

Keywords: Multimedia, Internet, Synchronization, SMIL, XML.

1 Introdução

A Internet é uma rede mundial de comunicação através de computadores, que vem sendo apresentada como um esboço de uma nova mídia de alcance massivo, prometendo integrar todas as tradicionais formas de comunicação, correio, imprensa, telefone, televisão, num único suporte, uma espécie de "mega-mídia".

Dos recursos oferecidos pela rede, a *World Wide Web* é a que mais cresce, tanto em público como em alocação de recursos da rede. A tecnologia fundamental da Web é a linguagem HTML (*HyperText Markup Language*) (W3C, 2002a). Ela é basicamente um padrão para a apresentação de hipertexto, com recursos razoáveis de estruturação do texto e inclusão de imagens. HTML é uma tecnologia baseada em SGML (*Standard Generalized Markup Language)* que descreve documentos especificando rótulos (*tags*) e inter-relacionando suas estruturas. O SGML - ISO 8879 - tem com propósito atender as necessidades associadas à descrição de documentos, tornando possível à manipulação de documentos complexos e gerenciamento de repositórios com grande quantidade de informações. HTML foi definida pela W3C (*World Wide Web Consortium*), consórcio criado em outubro de 1994, tendo como finalidade conduzir a Web, promovendo sua evolução e assegurando sua interoperabilidade.

A linguagem HTML desempenha bem seu papel em difundir a utilização da Web. Todavia, por ser uma linguagem simples, não é adequada à definição de documentos hipermídia com relacionamentos de sincronização temporal e espacial. Além disso, algumas aplicações envolvem um grau elevado de complexidade de informações que requerem capacidade de estruturação e acesso. Em resposta às limitações da HTML surgem novas propostas, com diferentes abordagens.

Uma destas propostas é a linguagem SMIL (*Synchronized Multimedia Integration Language*) (W3C, 1998) - desenvolvida pelo W3C (*World Wide Web Consortium*) - uma linguagem para especificação de multimídia na Web. SMIL é baseada em XML (*eXtended Markup Language*) (W3C, 2000) e é muito similar a HTML. Com SMIL é possível posicionar os elementos de mídia onde quiser na tela, sincronizar estes elementos e exibir a mídia de acordo com as preferências do usuário.

A linguagem SMIL possui atualmente duas versões, SMIL 1.0 (W3C, 1998) e SMIL 2.0 (W3C, 2002b). SMIL Boston é o nome funcional, definido pelo W3C, para a versão SMIL 2.0. A Linguagem SMIL 2.0 é uma extensão da linguagem SMIL 1.0, acrescida de novas características, as quais acrescentam poder de expressão da linguagem.

A linguagem XML (W3C, 2000) é um avanço tecnológico que insere recursos da SGML para descrição de documentos, com características adicionais ao HTML. É uma metalinguagem para a criação de novas linguagens de marcação (markup). Na prática, a XML permite que seja definido um novo conjunto de *tags*, adequado à classe de documentos que se deseja representar.

Sendo baseada em XML, um fator significante ao usar a linguagem SMIL é sua possibilidade de extensão. Uma ferramenta que possibilita o autor a criar novos elementos ou atributos, para poder suprir as deficiências encontradas, é um fator positivo e constitui uma vantagem adicional ao ser comparada com linguagens que se limitam a elementos e comandos pré-definidos.

Por outro lado, quando o assunto é dispor multimídia na Web, o problema está diretamente relacionado ao comportamento da rede. Sendo a Internet uma rede temporalmente não determinista, onde não existe a possibilidade de garantir taxa de bits, atraso e variação de atraso na entrega da mídia, resultando assim uma espera não definida. Estes problemas deveriam ser contornados pela linguagem que projeta a mídia.

O objetivo deste estudo é levantar requisitos para uma linguagem de composição de documentos multimídia usando como suporte de comunicação, redes temporalmente não deterministas e verificar se a linguagem SMIL 2.0 atende estes requisitos. Esta dissertação também propõe algumas extensões para a linguagem SMIL 2.0 afim de torná-la mais adaptada a tratar dos problemas causados pelo indeterminismo temporal das redes de computadores, além de torná-la uma linguagem que possibilite a estruturação dos documentos multimídia interativos. Esta extensão é realizada utilizando XML, ou seja, utiliza a forma padronizada de definir extensões à linguagem SMIL.

Os procedimentos de extensão da linguagem SMIL usando XML não são claramente definidos pelo W3C, exigindo assim um estudo aprofundado da linguagem XML. Como resultado desta dissertação, pretende-se também definir, de forma clara e simples, os procedimentos de extensão da linguagem SMIL usando XML.

O desenvolvimento do trabalho está descrito em 7 capítulos. O capítulo 2 descreve conceitos, teorias e aplicações da multimídia, fazendo referência a hipertexto e hipermídia. Os capítulos 3 e 4 descrevem a estrutura, a sintaxe e a semântica das linguagens envolvidas para o uso de multimídia na web. Mais precisamente, o capítulo 3 trata da linguagem XML e o capítulo 4 da linguagem SMIL.

O capítulo 5 aborda *namespace*, o qual é empregado para definir extensões da linguagem SMIL. O capítulo 6 apresenta os requisitos de um modelo de autoria para a criação de documentos multimídia a serem disponibilizados em redes com um comportamento temporal não determinístico, como a Internet.

Em seguida, o capítulo 7 relata uma análise da linguagem SMIL em relação aos requisitos de um sistema multimídia, de maneira que as deficiências desta linguagem sejam verificadas. Além disso, este capítulo apresenta a extensão proposta para a linguagem SMIL, além de definir passo a passo os procedimentos de extensão de SMIL usando XML.

Finalmente, o capítulo 8 apresenta as conclusões deste trabalho e algumas propostas de trabalhos futuros.

2 Multimídia

O conceito de multimídia não é novo; áudio, som e imagens são meios naturais de comunicação entre as pessoas. Com o avanço tecnológico dos sistemas de informação, houve a integração destes tipos de mídia. Atualmente, usa-se a palavra multimídia para a tecnologia que permite ao computador trabalhar com múltiplas mídias. Tais mídias podem ser visuais, auditivas e textos.

Multimídia é a integração de textos, gráficos, imagens, vídeos, animações, sons, transmitidos pelo computador ou qualquer outro meio, onde a informação pode ser representada, armazenada, transmitida, e processada digitalmente (FLUCKIGER, 1995). Entretanto, uma definição mais significativa de sistema multimídia, é um sistema capaz de manipular ao menos um tipo de mídia discreta e um tipo de mídia contínua, as duas numa forma digital. Mídias discretas são mídias com dimensões unicamente espaciais, tais como textos, imagens, gráficos. Por outro lado, mídias contínuas ou dinâmicas são mídias com dimensões temporais, tais como sons, vídeos e animações (VAUGHAN, 1994).

Um computador multimídia, capaz de lidar com sons e imagens, representa uma forma de interação homem-computador diferente da tradicional, que era feita exclusivamente por meio de textos e números. A multimídia requer, especificamente, o computador como meio de apresentação, devido a suas características únicas, como acessibilidade não-linear, interatividade e integração entre aplicativos (PAULA, 2000).

Existem diferenças entre os recursos audiovisuais tradicionais e a multimídia computadorizada. Esta última baseia-se na facilidade de armazenar e recuperar rapidamente grandes quantidades de informações. Multimídias computadorizadas fornecem com facilidade a interatividade, isto é, o usuário pode controlar a apresentação, através de links ou com uma interface similar a um videocassete, como interromper, avançar, retornar (RATHBONE, 1995).

2.1 Aplicações da Multimídia

A potencialidade da multimídia faz parte da experiência humana de milhares de anos. A implementação das capacidades de multimídia em computadores é simplesmente o mais recente episódio de uma longa série de avanços.

Atualmente, denota-se um vasto domínio de aplicações para multimídia. Publicações eletrônicas, jogos, aplicativos educacionais, aplicativos de produtividade pessoal, tele-ensino, e outros, são exemplos onde a multimídia é empregada como um brilhante recurso, pois humanos aprendem mais rapidamente quando vários dos seus sentidos são utilizados.

Nas comunicações *on-line,* o auge da multimídia não está longe. A tecnologia já existe, faz-se necessário implementá-la e aperfeiçoar as redes de comunicação. A corrida para implementar multimídia na Internet pode resultar em interfaces atraentes, porém, dadas às necessidades inatas do hardware de multimídia, muito mais têm que se ajeitar, isto é, acomodar quantidade de dados maiores, modens rápidos e largura de banda suficiente.

2.2 Multimídia na Web

A World Wide Web, um serviço de hipermídia da Internet, originalmente composta só por texto, tanto que o H de HTML significa *hypertext* e não *hypermedia.* Somente em 1993, foi que Marc Andreessen e Eric Bina adicionaram o tag <img>, consolidando imagens na Web.

A WWW é parte da Internet, ela representa os computadores (servidores) que oferecem acesso do usuário à informação e documentação baseadas em hipermídia. Na verdade, a Web permite que qualquer pessoa que tenha um computador e a conexão Internet apropriada seja um editor de multimídia (EAGAR, 1995).

A Internet, por sua vez, serve de suporte a um produto de multimídia na Web, os *sites* (sítios). Por *sites*, entendemos um título multimídia colocado em um servidor da WWW e visualizado remotamente por uma máquina cliente. A navegação nos *sites* é feita através de hiperligações (*hyperlinks*) entre as páginas (PAULA, 2000).

Porém, um problema em relação a Web decorre da limitação dos fluxos de dados nos canais utilizados para acesso à Internet. Essa limitação restringe o material multimídia. Técnicas de compressão são importantes; no entanto, o problema maior da multimídia na Web está relacionado com garantias temporais.

Outro problema é que mídias contínuas como vídeo e áudio são tratadas como mídias estáticas, isto é, cada arquivo é buscado do servidor e armazenado inteiramente no cliente local antes da apresentação.

Uma tecnologia recente utiliza a técnica de fluxo contínuo (*streaming),* a qual permite iniciar a exibição de som e imagem antes dos arquivos serem carregados completamente. Os dados são recebidos e armazenados em buffer, então são executados na máquina receptora enquanto mais dados são recebidos. Um exemplo de tecnologia de fluxo contínuo é dado pelo RealPlayer, da Real Netwoks (REAL, 2002).

O problema da tecnologia *streaming* é a manutenção do fluxo em tempo real. Variações de fluxo, devido a canais com pouca capacidade e qualidade, caracterizam ruídos e perda de quadro de vídeo.

Na verdade, a Internet ainda não suporta com êxito comunicações multimídia devido aos tipos de redes e protocolos. Quanto à rede, novas tecnologias baseadas por cabo e satélite, surgem para oferecer maior capacidade de fluxo. No que tange a protocolos, o HTTP *(Hypertext Transfer Protocol)* ou Protocolo de transferência de Hipertexto é baseado no TCP/IP *(TCP-Transmission Control Protocol/IP-Internet Protocol)*, o qual não é ideal para comunicações multimídia contínuas. RTP (*Real-Time*

Transport Protocol) (SCHULZRINNE, 1997) e o RTSP[1] (*Real Time Streaming Protocol*) (REAL, 2002), são protocolos que estão ganhando espaço, e fornecem suporte para aplicações com propriedades em tempo-real, tal como mídias streaming.

Um servidor de rede usa HyperText Transporte Protocolo (HTTP). Neste caso, os arquivos são carregados sem levar em conta a linha de tempo. O protocolo RTSP, por exemplo, permite ajustar o "derrame" de dados para tocar a mídia suavemente, mantendo o sincronismo. Comunicação assim não é possível pelo protocolo HTTP. A figura abaixo, fig 2.1, ilustra os dois protocolos – RTSP e HTTP.

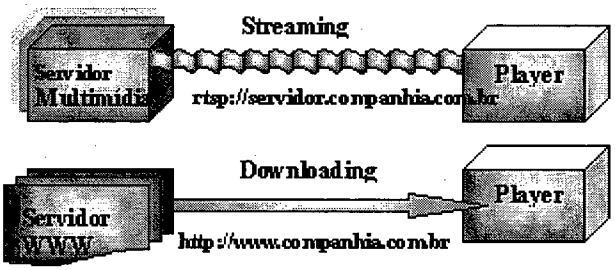


Figura 2-1 – Comparação entre servidor WEB e um servidor Multimídia

2.2.1 Hipertexto

A publicação de documentos na web é marcada pelos hipertextos. Hipertexto é um processo que permite embutir conexões especiais, chamadas *links* e que podem ser exibidos na tela do computador através de um *browser*[2]. A característica marcante destes sistemas é a apresentação da informação de forma não linear, deixando os usuários seguirem caminhos definidos através de uma grande coleção de informação textual (RANDALL, 1997).

Hipertexto refere-se especificamente a documentos baseados em computadores nos quais os leitores se movem de um local para outro dentro de um documento. As

[1] Proposta de padronização IETF Iinternet Engineering Task Force) para o controle de fluxo de mídia sobre redes IP, incluindo a Internet.
[2] São softwares para navegação na WWW.

informações não são acessadas de forma tradicional, isto é, do início ao fim; o usuário se move aleatoriamente dentro do documento. As palavras, expressões e ícones no documento tornam ligações que permitem saltar à vontade de um documento para outro, ou até mesmo para outro local no mesmo documento (EAGER, 1995).

Hipertexto tem sido definido como uma abordagem para o gerenciamento de informação no qual os dados são armazenados em uma rede de nodos conectados por ligações. Os nodos podem conter textos, gráficos, sons e vídeos, assim como, código fonte ou outras formas de dados.

Em uma rede hipertexto os nodos estão conectados uns aos outros através das ligações. O nodo origem é chamado de *referência* e o nodo destino é chamado de *referente*. São também freqüentemente chamados de *âncoras*. O conteúdo de cada nodo é exibido pela ativação das ligações, que podem ser bidirecionais, e, portanto, facilitar o processo da busca da informação.

O hipertexto apresenta vantagens sobre um texto normal. Uma destas vantagens está relacionada à facilidade de navegação em documentos muito grandes. Além de fornecer velocidade no uso, ajuda os leitores a explorar novas idéias e localizar fontes adicionais. Ainda, os usuários, exploram e tomam decisões sobre os tópicos que querem investigar (EAGER, 1995).

2.2.2 Hipermídia

Quando uma estrutura de informação de hipertexto oferece diferentes tipos de mídias, chamamos de *hipermídia*. Um documento hipermídia é uma fusão de documentos hipertexto e multimídia. O termo hipermídia é mais usado para descrever aplicações baseadas na Web. É um valioso instrumento que oferece oportunidades de interação, como movimento, ritmo, seqüência, ordenação, entre outros (DRISCOLL, 1998).

A hipermídia utiliza informações sob o controle de um computador, de maneira que um usuário possa *navegar* nela de maneira produtiva. A informação na hipermídia é amarrada com conexões, de modo que o usuário possa passar de uma informação para outra, de maneira não linear. Esta conexão é chamada de *hyperlink* (MARTIN, 1992).

A hipermídia permite reunir sob um mesmo suporte, os mais diversos tipos de material informativo: textos, imagens estáticas ou animadas, sons e vídeo. O formato hipermídia representa mais do que um somatório dos formatos tradicionais num mesmo meio. A sua especificidade está na forma de percorrer o conteúdo. Ao invés da leitura seqüencial e completa, privilegia mecanismos em rede, numa forma de malha com várias conexões entre vários blocos de informação. Um documento linear pode ser lido somente na ordem em que foi composto. A vantagem de um documento não linear é a capacidade de organizar objetos de diversas maneiras, dependendo das diferentes visões e demandas.

Portanto, o ambiente hipermídia oferece novas possibilidades de acesso a fontes de informações. O modelo hipermídia incentiva o autor a criar referências e modularizar suas idéias, embora ele seja obrigado a tomar decisões sobre qual a melhor maneira de particionar adequadamente as informações.

A interface com o usuário é uma das características marcantes em sistemas hipermídia, um ambiente gráfico proporciona facilidades de navegação. Outras vantagens estão citadas abaixo:

a) os segmentos podem ser estruturados de várias maneiras permitindo que o mesmo documento sirva para múltiplas funções;

b) uma vez que partes do mesmo documento podem ser referenciadas de vários lugares, as idéias podem ser expressas com pouca sobreposição ou duplicação;

c) vários autores podem cooperar na criação de um mesmo documento ou simplesmente adicionar e compartilhar comentários.

Por fim, provocam um ambiente de aprendizagem e de descoberta. A capacidade de acessar as informações rapidamente e relevantemente pela seleção da direção a ser tomada, parece ser uma característica essencial tanto do hipertexto quanto a hipermídia.

É desejável um formato padronizado para hipermídia genérica. Algumas tentativas de padronização estão em curso (PAULA, 2000):

- o formato MHEG, Multimedia, Hypermedia Expert Group, segue o modelo de referência OSI da ISO, segundo a arquitetura da camada de apresentação, tem a função de integrar e codificar as partes de multimídia para que se atinja o objetivo de, com "mínimo de recursos" de computação e independente da plataforma, trabalhar com qualquer aplicativo multimídia;

- o formato AAF, desenvolvido pela Microsoft, para autoria multimídia;

- o formato SMIL tem sido projetado de maneira a facilitar a autoria de apresentações multimídia dinâmicas e sincronizadas na Web - este formato é objeto de estudo deste trabalho.

2.3 Requisitos de Armazenamento e Largura de Banda

Um sistema multimídia é um sistema dotado de periféricos especiais de entrada e saída. Os requisitos de software e hardware dependem da aplicação, mas de modo geral as aplicações multimídia impõem requisitos na arquitetura do hardware, por exemplo, maior poder de processamento, equipamentos para digitalização, entre outros.

A capacidade de armazenamento é fator relevante num sistema multimídia. Outro ponto importante é a disponibilidade e a capacidade da rede quando se refere a sistemas distribuídos, pois sistemas multimídia requerem alta taxa de transferência de dados. Ambos, armazenamento e capacidade de rede estão descritos nas seções a seguir.

2.3.1 Armazenamento

O material produzido deve ser armazenado em dispositivos *off-line,* para liberar disco e dispor de cópias de segurança. O meio mais tradicional de armazenagem é o disquete, porém sua capacidade é limitada. Atualmente, dispomos de outros recursos, tais como, fitas magnéticas, CD-ROM, discos removíveis, entre outros. A tabela 2.4 mostra uma comparação entre dispositivos de armazenamento.

Tabela 2-1 - Comparação entre dispositivos de armazenamento

Meio	Capacidade	Tamanho	Velocidade de Acesso
Disquetes	Baixa	1,44 Mb	Média
Discos removíveis	Médio/Alta	Zip (100Mb) e Jaz (1Gb e 2 Gb)	Média/Alta
Fitas Magnéticas	Alta	200 a 800 Mb - DAT	Baixa (Seqüencial)
CD-ROM	Alta	650Mb p/dados – 74min.P/ áudio	Média/Alta

Fonte: WILLRICH, Roberto. Apostila Sistemas Multimídia Distribuídos, UFSC, março 2000

Atualmente, o CD-ROM, discos óticos, são os mais apropriados para armazenamento de multimídia.

Quanto a servidores multimídia, são sistemas computacionais que oferecem serviços multimídia para outros sistemas chamados de clientes. Um servidor deve ter alta capacidade de armazenamento e funcionalidade de captura de dados, e podem ter dedicação específica, por exemplo, um servidor de imagem e um servidor de vídeo (WILLRICH, 2000).

2.3.2 Largura de Banda

Em sistemas distribuídos, alguns requisitos de desempenho se fazem necessários para transmissão de multimídia, como velocidade, vazão, taxa de erro, atraso, variações de atrasos, capacidade de armazenamento e confiabilidade. A capacidade de transmissão

da rede é denominada de largura de banda (OTTE, 1995). Na verdade, largura de banda é um dos entraves associados aos requisitos de transmissão na web.

A largura de banda é determinada pelo meio de transmissão utilizado, isto é, protocolos, velocidade de comutação nos nós intermediários, distância entre os nós, meio físico (cabo coaxial, par trançado e fibra ótica). A fibra ótica é a que oferece maior largura de banda.

No domínio digital, a largura de banda é medida como taxa de bits. Taxa de bits é o número de dígitos binários (zeros e uns) que a rede é capaz de transportar por unidade de tempo.

Por outro lado, denomina-se *banda* a faixa de freqüência entre dois limites. No telefone, por exemplo, o limite de freqüência usado é de 3100 Hz. Portanto, a largura de banda do telefone é 3 100 Hz. Entretanto, denomina-se *banda larga* um canal de transmissão cuja largura de banda é maior que um canal de voz, permitindo transmissão de dados mais rápidos. Atualmente, várias tecnologias de banda larga são utilizadas para transmissão de dados. Segundo (CEREDA, 1997), "apostar em qual destas tecnologias será a vencedora não passa exatamente disso, uma aposta". Estas tecnologias utilizam diversos meios para combinar linhas analógicas e linhas digitais, tais como par trançado, cabo coaxial e fibra ótica. Entre elas, destacam-se a ISDN, ADSL, cabo e microondas.

- **ISDN** – Integrated Services Digital Network - Rede Digital de Serviços Integrados. Uma rede de transmissão de alta velocidade que permite tráfego de voz, imagem simultaneamente, compartilhando o mesmo meio de transmissão. Todos os sinais que trafegam na rede são digitais, assim não existem problemas de variações e ajustes necessários nos sinais analógicos. Na prática, redes ISDN utilizam a infra-estrutura das antigas redes telefônicas, porém um sinal digital em vez do sinal analógico é transmitido. Redes ISDN oferecem conectividade digital fim-a-fim, ou seja, toda a rede é digital, inclusive o acesso doméstico.

- **ADSL** (*Asymmetric Digital Subscriber Line*) – Linha Digital Assimétrica de Assinante - Pares trançados de fios de cobre de telefonia comum. Utiliza a fiação telefônica existente; entre a operadora e o usuário está o provedor de banda larga. Não utiliza discagem, deixando o telefone livre. Necessita um modem ADSL e placa de rede.

- **Sistema a Cabo** – Este sistema utiliza cabo coaxial, que transmite sinais múltiplos e freqüências extremamente altas. Este acesso é via operadora de TV.

- **Links de microondas (Rádio)** – Uma conexão por microondas para acessar a concessionária pública. Uma placa de rede conectada a um fio, que leva e traz os sinais da antena.

Outras tecnologias tais como, HFC (*Hybrid Fiber Coax*) Híbrido, Fibra e Coaxial, FTTC (*Fiber to The Curb*) Fibra até a Calçada – Também chamada por alguns autores de SDV (*Switched Digital Video*). A FTTC oferece, fibra da operadora até o domicílio ou grupo de domicílio. No último caso, distribuição final por pares trançados de fios de cobre (CEREDA, 1997).

Definida multimídia e como ela é empregada na Web, o próximo capítulo versará sobre uma linguagem de marcação, desenvolvida pelo W3C. Esta linguagem é chamada de XML (eXtensible Markup Language) e vem solucionar as limitações da HTML, quando o assunto é multimídia na WWW. A linguagem XML oferece uma abordagem para descrição, captura, processamento e publicação de informações na Web.

2.4 Conclusão

Este capítulo teve como objetivo conceituar multimídia, hipermídia e hipertexto, considerando a Internet o meio de comunicação usado para sua aplicação.

Foram apontados alguns problemas, decorrentes da limitação dos fluxos de dados nos canais utilizados para acesso à Internet. Esta limitação restringe o material multimídia.

Uma tecnologia recente utiliza a técnica de fluxo contínuo (streaming), a qual permite iniciar a exibição de som e imagem antes dos arquivos serem carregados completamente. O problema da tecnologia streaming é a manutenção do fluxo em tempo real. Variações de fluxo, devido a canais com pouca capacidade e qualidade, caracterizam ruídos e perdas.

Sistemas distribuídos requerem alta taxa de transferência de dados. Porém, a Internet não suporta com êxito comunicações multimídia devido ao tipo de rede e protocolos. A capacidade de armazenamento é também fator relevante.

O próximo capítulo apresentará a linguagem XML, uma linguagem de marcação, desenvolvida principalmente para solucionar as limitações da HTML. XML é uma metalinguagem, que define a linguagem SMIL.

3 Linguagem XML

A XML (eXtensible Markup Language) é uma nova linguagem de marcação, desenvolvida pelo W3C, principalmente para solucionar as limitações da HTML. A linguagem XML e a linguagem HTML são aplicações que seguem as regras da SGML (W3C, 1997). O exemplo abaixo é de um documento XML:

```
<documento>
 <título> meu primeiro documento xml </título>
 <texto_1> olá mundo! </texto_1>
 <texto_2> este é um documento em forma de marcação. </texto_2>
</documento>
```

É importante ressaltar que a XML não é uma versão estendida da HTML, mas sim uma versão simplificada da SGML. A XML pretende não só remover da Web a inflexibilidade da HTML, mas também a complexidade da SGML, com a qual é muito difícil de se trabalhar. A sintaxe da marcação da XML lembra a HTML (BRADLEY, 2000).

HTML é uma linguagem de marcação extremamente popular. Para acomodar essa popularidade, visto seu crescimento, foram inseridos muitos tags, além de suportar um conjunto de novas tecnologias como Java, JavaScrip, Flash, CGI, mídia streaming, e outros. Porém, nenhum conjunto de tags, não importa seu tamanho, poderá oferecer todas as tags que desejamos de forma conceptível. Aplicações de comércio eletrônico necessitam da tags para referências de produto, preço, endereços, etc., para controlar o fluxo de imagens e sons precisa-se de tags, a área de segurança necessita de tags para assinatura digital, a lista de aplicações é infinita. Para responder a todas estas necessidades a HTML foi tornando-se complexa. Se, por um lado, as aplicações necessitam de mais tags, por outro, autores e desenvolvedores desejam menos tags.

XML é uma tecnologia desenvolvida para suprir necessidades e contornar o problema da complexidade, ela não vem para substituir a HTML, mas inserir novas possibilidades. Não se faz necessário extinguir uma linguagem com tanto sucesso como a HTML, portando, já se trabalha em direção a uma fusão entre XML e HTML, a XHTML (MARCHAL, 2000).

Um documento XML como um HTML contém instruções especiais chamadas de tags. A estrutura da XML é flexível e extensível, não predefine tag nenhum, podemos criar linguagens de marcação personalizadas. O autor tem a liberdade de usar qualquer nome para implementar tags em seus dados. Para ilustrar, um pequeno comparativo entre um documento HTML, fig. 3.1 e um documento XML, fig. 3.2.

```
<html>
<head><title>XML</title></head>
    <body>
        <p><FONT face="arial black">XML</FONT></p>
        <p>Linguagem XML</p>
        <p>A XML (eXtensible Markup Language) é uma nova
    linguagem de marcação desenvolvida pelo W3C (World Wide Web
    Consortium) principalmente para solucionar as limitações da HTML. A
    linguagem XML oferece uma abordagem para descrição, captura,
    processamento e publicação de informações </p>
        <p>Para saber mais visite o site
            <Ahref="http//www.w3.org/XML">
            http//www.w3.org/XML</A></P>
    </body>
</html>
```

Figura 3-1 - Documento em HTML

```
<?xml version="1.0"?>
<!DOCTYPE memo SYSTEM "MEMO.DTD">
<memo>
<header>
<subject> Linguagem XML</subject>
</header>
<body>
<para>A XML (eXtensible Markup Language) é uma nova linguagem de marcação desenvolvida pelo W3C (World Wide Web Consortium) principalmente para solucionar as limitações da HTML. A linguagem XML oferece uma abordagem para descrição, captura, processamento e publicação de informações</para>
<para>Para saber mais visite o site <A href= "http//www.w3.org/XML"> http//www.w3.org/XML</A></para>
</body>
</memo>
```

Figura 3-2 - Documento em XML

A distinção ilustrada nesses dois documentos representa a essência da XML. A XML é inteligente para qualquer nível de complexidade. Um exemplo simples é dizer que uma marcação pode ser "<dog> Lassie </dog>".

3.1 Aplicações da XML

A XML não é apenas para publicação de *sites* na Web, existem outras aplicações também populares. Podemos classificar essas aplicações em dois grupos: o primeiro em aplicações que manipulam informações voltadas para o consumo humano e a segunda voltada para o consumo de software. O padrão XML é o mesmo para os dois grupos, porém atende a objetivos diferentes (MARCHAL, 2000).

Uma aplicação de XML é na publicação de documentos. XML é independente ao meio, isto é, ela se concentra na estrutura do documento e o torna independente do meio de distribuição. Por exemplo, documentos XML convertidos para PostScript, HTML,

PDF, XHTML e outros. A possibilidade de objetivar diferentes meios é importante, pois muitas publicações estão impressas. Além disso, a Web modifica rapidamente, o que é usual hoje poderá ser ultrapassado no ano seguinte (BOX, 2000).

Todavia, se a estrutura de um documento pode ser expressa em XML, a estrutura de um banco de dados também pode. A XML traz então uma nova aplicação, uma espécie de distribuição de dados para publicação.

Finalmente, uma aplicação da XML é designada para prover uma plataforma multimídia na web, a linguagem SMIL. Esta linguagem emprega XML para identificar e gerenciar a apresentação de arquivos de texto, imagens, som e vídeo.

3.2 Padrões Acompanhantes da XML

Por ser uma linguagem de marcação padrão, a XML oferece uma lista crescente com suporte de diversos padrões e produtos. Estes padrões que complementam a XML são denominados de *padrões acompanhantes da XML* (MARCHAL, 2000). Novos padrões são inseridos regularmente, alguns exemplos estão relacionados abaixo.

- **Folhas de estilo:** As folhas de estilo definem como um documento XML deve ser apresentado. A XML possui duas linguagens que trabalham com folha de estilo. A XSL (XML Stylesheet Language) e CSS (Cascading Style Sheet). A XSL é mais poderosa, no entanto, a CSS possui mais implementações.

- **XML Pointer Language (XPointer) e XML Linking Language (XLink):** define um padrão para representar os links entre os recursos. Além dos links simples, como a tag <A> da HTML, a XML possui mecanismos para ligar recursos múltiplos e diferentes. A XPointer descreve como endereçar um recurso, e a XLink descreve como associar dois ou mais recursos.

- **XML Namespace:** uma solução para ajudar a gerenciar a facilidade de extensão da XML.

- **DOM e SAX:** DOM (Document Object Model) e SAX (Simple API for XML) são APIs (API-Application Programming Interface) de acesso aos documentos XML. Elas permitem que aplicativos XML leiam documentos sem se preocuparem com a sintaxe.

3.3 Aplicativos para XML

Como qualquer linguagem, é necessário ambientes para elaboração, visualização e interação de documentos.Um navegador XML é o primeiro aplicativo a se desejar quando o assunto é Web. O Microsoft Internet Explorer tem compatibilidade para XML desde a versão 4.0. O Netscape Communicator, atualmente não possui suporte, exceto pelo Mozilla, porém ainda restrito. Entre outros, o mais interessante é o InDelv XML Browser (MARCHAL, 2000).

Por outro lado, existe uma quantidade muito grande de editores XML. Alguns exemplos: XML Notepad da Microsoft[3], o Adobe Framemaker, XML pro e outros.

Todo aplicativo XML é baseado em um Parser. Um Parser é uma ferramenta que tem por objetivo ocultar do desenvolvedor os detalhes da sintaxe XML. Existem vários disponíveis na Internet. Na versão 4.0 do Internet Explorer já estavam inclusos dois parsers XML.

O processador XSL é um aplicativo que permite que se produza a linguagem HTML clássica, com a vantagem de reter a XML internamente. Existem muitos à disposição; um exemplo é o LotusXSL.

---

[3] download gratuito na www.microsoft.com

3.4 Estrutura lógica e física de um documento XML

Um documento XML pode ser analisado de duas maneiras. Primeiro é visto sobre uma estrutura lógica, onde o documento é uma hierarquia de informações. Esta estrutura permite um documento a ser dividido em unidades e sub-unidades chamadas *elementos* (BRADLEY, 2000). Os dados dividem-se em forma de árvore e no alto esta o elemento *raiz.*

Lado a lado com a estrutura lógica, os documentos XML possuem uma estrutura física. Esta estrutura permite componentes do documento chamadas *entidades,* para ser nomeados e armazenados separadamente, de tal modo que esta informação também possa ser reusada num formato não XML, como por exemplo, uma imagem. Como acontece na estrutura lógica, a estrutura física de um documento XML também é hierárquica, significando que uma entidade pode conter referências a outras entidades, que por sua vez pode conter referência a outras (BRADLEY, 2000, MCGRATH, 1998).

3.5 Sintaxe XML

XML é considerada uma *metalinguagem,* isto porque é uma linguagem que descreve outras linguagens. Nela não existe uma lista pré-definida de elementos. XML é completamente livre para empregar elementos com nomes que são significativos para as aplicações (BOX, 2000).

XML foi concebida para descrever e manipular documentos estruturados, que são armazenados em formato eletrônico. Ela oferece uma estrutura em forma de árvore, é um mecanismo flexível, não dita como preencher esta árvore (BRADLEY, 2000).

Os arquivos XML consistem em caracteres conhecidos como *marcação,* juntamente com o conteúdo de informações, conhecido como *dados de caracteres.* Obviamente, é a marcação que distingue o documento XML do texto puro.

Um documento XML inclui tags de início, tags de término, comentários, entre outros. Este documento pode opcionalmente estar associado a um conjunto de regras que especificam a ordem e a ocorrência de marcação e de dados de caracteres que são permitidos. Estas regras encontram-se em um DTD (Document Type Definition) (MARCHAL, 2000).

O padrão XML permite uma certa liberdade para a criação de tags, contudo quando estamos construindo um documento XML devemos seguir a sintaxe da linguagem XML, sob pena do processador XML não reconhecer o arquivo como um arquivo de programa XML. A um arquivo criado seguindo estas restrições é denominado de *bem formado.* Basicamente um documento bem formado é um arquivo que o processador XML pode criar com êxito, uma estrutura em árvore (BRADLEY, 2000).

Documentos bem formados não possuem DTD de modo que o processador XML não pode verificar sua estrutura. Ele apenas verifica se seguem as regras da sintaxe. Documentos válidos são mais restritos, além de seguir as regras da sintaxe, são compatíveis com uma estrutura específica, conforme escrita no DTD (MARCHAL, 2000).

Um documento XML bem formado pode ser classificado como *válido* se atender as restrições indicadas em um DTD associado. Um documento válido precisa ter uma declaração do tipo *Document Type* que determine qual conjunto de regras são válidos no documento em questão, qual conjunto de tags pode aparecer e como eles devem ser agrupados dentro do documento. Estas informações que são armazenadas num arquivo DTD podem ser declaradas no início de um arquivo XML ou em um arquivo DTD isolado. A sintaxe de uma DTD é diferente de uma sintaxe XML. DTD é um mecanismo para descrever cada objeto (elemento, atributo, etc.).

3.6 Elementos

Elemento é a mais comum forma de marcação. Delimitado pelos sinais de menor e maior, geralmente identificam a natureza do conteúdo que envolve. Um elemento é definido entre tags. Ele possui um nome e um conteúdo. Exemplo:

<from> Elisa Pivetta Cantarelli </from>

No exemplo, o elemento é a palavra *from* e seu conteúdo é Elisa Pivetta Cantarelli. Ao contrário da HTML, as tags de início e fim são obrigatórias.

O conceito de conteúdo em um elemento não se adapta a todos os elementos. Em HTML temos elementos sem conteúdo, por exemplo, *br,* que significa quebra de linha. Em XML ele é chamado de *elemento vazio*. Existe uma notação abreviada para os elementos vazios, a barra do tag de fim é inserida no final do tag de início. Para ilustrar:

<e-mail href=mailto:elisa@urifw.tche.br></e-mail>
<e-mail href=mailto:elisa@urifw.tche.br/>

Para criação de nomes dos elementos devem ser observadas algumas regras (BRAY, 1999), (MARCHAL, 2000):

a) devem começar com uma letra ou com o caractere underline (_), o restante do nome é constituído de letras, dígitos, caractere underline (_), ponto(.) ou o hífen (-) e não é permitido espaço no nome;

b) os nomes não podem começar com a string *xml;*

c) o sinal de dois-pontos (:) é primeiramente reservado para namespaces;

d) em XML letras maiúsculas e minúsculas são diferenciadas, por exemplo, <From>, <from> e <FROM> são diferentes.

Os atributos são informações anexadas ao elemento. Eles possuem nome e um valor. Os elementos podem ter um ou mais atributos. Todos os valores de atributos devem estar entre aspas. Por exemplo:

<fone preferencial= "sim" > 744-6270</fone>

3.7 Definição de Tipo de Documento – DTD

Um DTD é dividido em subconjuntos internos e externos. O subconjunto interno é inserido no próprio documento e está entre colchetes na declaração do tipo de documento. O externo aponta para uma entidade externa e é referenciado a partir da declaração de tipo de documento. É também necessário diferenciar *definição de tipo de documento* com *declaração de tipo do documento*. Esta última conecta um DTD a um documento. Por exemplo: <!DOCTYPE cadastro system "cadastro.dtd">

A instrução <!DOCTYPE> vincula o arquivo de documento ao arquivo DTD. A fig. 3.3 é documento XML do cadastro e a fig. 3.4, é o seu DTD.

```
<?xml version="1.0"encoding=ISO-8859-1" ?>
<!DOCTYPE cadastro system "cadastro.dtd">
<cadastro>
        <dados >
   < nome>Elisa Maria Pivetta Cantarelli</nome>
   <endereço>
     < rua>Rua Ceará</>
          <cidade>Frederico Westphalen/>
   <endereço/>
        <fone preferencial= "sim" > 744-6270</fone>
   <fone> 744-4040</fone>
   <e-mail href=/>
 </dados>
        <dados>
   <nome><pnome>Dr.Roberto</pnome><snome> Willrich</snome></nome>
   <fone>99999999999</fone>
   <e-mail href=mailto:willrich@inf.ufsc.br/>
 </dados>
</cadastro>
```

Figura 3-3 - Cadastro em XML

```
<?xml encoding=ISO-8859-1" ?>
<!ELEMENT cadastro (dados+)>
<!ELEMENT dados (nome,endereço*,fone*,e-mail*)>
<!ELEMENT nome (#PCDATA | pnome | snome)>
<!ELEMENT pnome  (#PCDATA)>
<!ELEMENT snome  (#PCDATA)>
<!ELEMENT endereço  (rua, cidade)>
<!ELEMENT rua (#PCDATA)>
<!ELEMENT cidade     (#PCDATA)>
<!ELEMENT fone  (#PCDATA)>
<!ATTLIST   fone  preferido (sim | não) "não">
<!ELEMENT e-mail  EMPTY>
```

Figura 3-4 - DTD do cadastro

Num DTD temos primeiramente a declaração do elemento seguido por seu modelo de conteúdo:

<!ELEMENT cadastro (dados+)>

Como nomes em XLM possuem regras, da mesma forma para nomes na DTD. Um nome pode começar por letra ou um conjunto limitado de sinais de pontuação ("_", ":"). Espaços não são permitidos e não podem começar com a string "xml" e como visto anteriormente o sinal de dois-pontos também tem função especial.

Os sinais de mais (+), asterisco(*) e o ponto de interrogação (?) são indicadores de ocorrência. Indicam se e como os elementos na lista de filhos podem se repetir:

a) um elemento sem nenhuma ocorrência deve aparecer somente uma vez no elemento que está sendo definido;

b) um elemento com sinal de mais, deve aparecer uma ou mais vezes no elemento que está sendo definido;

c) um elemento com sinal de asterisco pode aparecer zero ou mais vezes no elemento que está sendo definido;

d) um elemento com sinal de interrogação pode aparecer uma ou nenhuma vez no elemento que está sendo definido.

Os caracteres de vírgula (,) e barra vertical (|) são conectores, eles separam os filhos no modelo de conteúdo e indicam a ordem em que os filhos podem aparecer. Se aparecer a vírgula, indica que os dois elementos (o da direita e o da esquerda) devem aparecer na mesma ordem no documento. Se for a barra, indica que somente um deles (o da direita ou o da esquerda) deve aparecer no documento (MARCHAL, 2000).

3.7.1 Palavras-chave

Existem palavras-chave especiais dentro de uma DTD. Por exemplo:

- #PCDATA - indica que o elemento pode conter texto. Geralmente é usado para elementos terminais, isto é, os que não possuem filhos;

- EMPTY - indica um elemento vazio e é sempre terminal;

- ANY - significa que o elemento pode conter qualquer outro elemento declarado na DTD. Raramente é usado.

3.7.2 Atributos

Atributos também devem ser declarados na DTD. Declarações de listas de atributos identificam que elementos podem ter atributos, que atributos eles podem ter, que valores os atributos podem suportar e qual valor é o padrão. Eles são declarados na ATTLIST. Os atributos podem ser de vários tipos, abaixo estão descritos os mais usados.

a) CDATA são cadeias de caracteres, qualquer texto é permitido. Não confunda atributos CDATA com seções CDATA; eles não têm relação.

b) ID para identificador. Todos os valores usados para IDs em um documento devem ser diferentes. Os IDs identificam unicamente elementos individuais em um documento.

c) IDREF ou IDREFS, o valor de um atributo IDREF deve ser o valor de um único atributo ID em algum elemento no documento. O valor de um atributo IDREFS pode conter valores IDREF múltiplos separados por espaços em branco.

d) ENTITY ou ENTITIES, o valor de um atributo ENTITY deve ser o nome de uma única entidade. O valor de um atributo ENTITIES pode conter valores de entidades múltiplos separados por espaços em branco.

e) NMTOKEN ou NMTOKENS, atributos de símbolos de nome são uma forma restrita do atributo de cadeia de caracteres. Em geral, um atributo NMTOKEN deve consistir de uma única palavra, mas não há restrições adicionais para a palavra; não tem que estar associado com outro atributo ou declaração. O valor de um atributo NMTOKENS pode conter valores NMTOKEN múltiplos separados por espaços em branco.

Opcionalmente, a DTD pode especificar um valor padrão para o atributo. Este valor pode assumir um dos quatro valores abaixo:

- #REQUIRED, o atributo deve ter um valor explicitamente especificado em cada ocorrência do elemento no documento;

- #IMPLIED, o valor do atributo não é requerido, e nenhum valor padrão é fornecido; se um valor não é especificado, o processador XML deve proceder sem um;

- #FIXED seguido por um valor, significa que o valor do atributo deve ser o declarado na DTD;

- um valor literal significa que o atributo assumirá esse valor.

3.7.3 Criação e uso de DTDs

O projeto de um DTD é uma atividade criativa, porém, existem muitas DTDs para XML disponíveis. Sempre que possível, reutilize DTDs; a reutilização resulta em economia.

Uma solução mais fácil para criar DTDs é partir de um modelo de objeto.Cada vez mais modelos de objetos estão disponíveis em UML (Unified Modeling Language). Este modelo é geralmente usado para aplicações orientadas a objeto, como C++ ou Java, mas também pode ser usado com XML. O modelo de objeto oferece objetos prontos, que só precisa ser convertido para XML. Ele ainda identifica as propriedades e os relacionamentos entre os objetos.

Todavia, a criação de uma DTD sem um modelo de objeto resulta em mais trabalho. Faz-se necessário criar documentos de exemplos e analisá-los para saber como adaptar a DTD proposta. Uma variante é modificar uma DTD existente. Normalmente, a DTD básica não aceita todo o seu conteúdo ou é muito simples ou complexa para sua aplicação.

É interessante que, ao criar uma DTD, o modelo seja flexível o suficiente para acomodar extensões. Para evitar armadilhas, deve-se fornecer o máximo de informações de estrutura possível, porém não em demasia. A dificuldade é realmente chegar a este equilíbrio.

Existem algumas ferramentas que auxiliam na descrição de DTDs. Montar árvores num pedaço de papel parece fácil, mas depois de várias alterações começa a tornar-se obscuro. A ferramenta Near & Far, da Microstar (www.microstar.com) é um exemplo para modelar uma árvore com aspecto correto. Com a grande vantagem de que, ao salvar, ela converte a árvore para uma DTD, sendo assim, não é preciso lembrar da sintaxe.

3.8 Parsers

Todo documento XML é baseado em um parser. Parser é uma ferramenta de suma importância, pois tem a finalidade de ocultar do desenvolvedor os detalhes da sintaxe XML. Parser é um componente de software, de baixo nível, residente entre o aplicativo e os arquivos XML.

Geralmente, os parsers vêm com a ferramenta de desenvolvimento, mas existem vários disponíveis gratuitamente na Internet. A Microsoft, por exemplo, quando lançou o Internet Explorer 4.0 com suporte para XML, disponibilizou juntamente dois parsers XML.

Em compiladores, parser é o módulo que lê e interpreta a linguagem de programação. Não diferente, os parsers ou processadores XML, foram projetados para verificar e validar documentos. Desta forma, as aplicações devem ter conhecimento das regras de processamento definidas na especificação XML.

3.8.1 Parsers XML de validação e de não validação

Para processamento de documentos XML, duas classes de parsers são definidas:

- *parsers de não validação* - verifica se o documento é bem-formado, ou seja, se está de acordo com a normas da XML;

- *parsers de validação* - verifica se o documento está bem formado e se esta em conformidade com a especificação que define sua gramática.

Na verdade, os dois parsers impõem regras sintáticas, mas apenas o de validação, valida o documento comparando com a DTD. Estes dois parsers podem ser usados de modo complementar.

3.8.2 A aplicação e o Parser

A interface entre a aplicação e o parser pode ser realizada de duas maneiras:

- *baseada em objetos* - o parser cria uma árvore de objetos que contém todos os elementos do documento XML;

- *baseada em eventos* - o parser lê o arquivo e gera eventos à medida que encontra elementos, atributos ou texto.

Na prática, as duas maneiras citadas acima são úteis, mas atendem a objetivos diferentes. A baseada em objetos é ideal para aplicações que manipulam documentos XML, como, por exemplo, navegadores, processadores XSL, editores, etc.. Por outro lado, as baseadas em eventos servem para as aplicações que possuem sua estrutura de dados em formato não XML, como, por exemplo, aplicações que importam documentos XML para banco de dados. Neste caso, o formato da aplicação é o esquema do banco de dados, e não o esquema XML (MARCHAL, 2000).

A existência de duas maneiras diferentes de realizar a interface entre a aplicação e o parser resulta em dois padrões diferentes. O padrão para a interface baseada em objetos é DOM (Document Object Model), publicado pelo W3C[4]. Para a baseada em eventos o padrão é SAX (Simple API), desenvolvido em colaboração pelos membros da lista de correspondência XML-DEV e editado por David Megginson[5].

3.9 Esquemas XML

O DTD, mesmo oferecendo recursos valiosos, possui algumas limitações herdadas da SGML:

[4] www.w3.org/TR/REC-DOM-Level-1
[5] www.megginson.com/SAX

a) DTD é baseado em uma sintaxe especial; não é possível processar DTDs em ferramentas XML, como editores ou navegadores;

b) não é possível incluir restrições de repetição; um elemento só pode aparecer zero, uma vez ou infinitamente, mas nunca restringir a um número x de vezes (por exemplo, aparecer 5 vezes);

c) os elementos são definidos como do tipo string, não aceitando, por exemplo, números, datas etc.

À medida que a XML cresce, alguns grupos criam propostas para contornar estas e outras restrições.

Desta forma, a recomendação *XML Schema Definition Language*[6] provê recursos para superar algumas limitações da DTD.

O próximo capítulo é um estudo da linguagem SMIL, a qual descende da linguagem XML. O capítulo tem por finalidade descrever a sintaxe e a semântica da linguagem SMIL 2.0.

3.10 Conclusão

Este capítulo fez uma apresentação da linguagem XML, descrevendo seu conceito, sua sintaxe e sua semântica.

XML é considerada uma metalinguagem, pois é uma linguagem que descreve outras linguagens. XML é usada para descrever e manipular documentos estruturados. Estes documentos não são limitados a Web, podendo incluir objetos em uma aplicação cliente/servidor.

---

[6] www.w3.org/XML/Schema

Um documento XML pode opcionalmente estar associado a um conjunto de regras que especificam a ordem e a ocorrência de marcação e de dados de caracteres que são permitidos. Estas regras encontram-se em um DTD (Document Type Definition). DTD é a linguagem de modelagem ou esquema original para a XML.

Uma aplicação da XML designada para prover uma plataforma multimídia na Web, é a linguagem SMIL. SMIL emprega XML para identificar e gerenciar a apresentação de arquivos texto, imagem, som e vídeo e será descrita no próximo capítulo.

4 Linguagem SMIL

SMIL (*Synchronized Multimedia Integration Language*) é uma linguagem declarativa para descrever apresentações multimídia na Web. Foi desenvolvida pelo W3C (*World Wide Web Consortium*) e lançada em 15 de Junho de 1998. SMIL permite integrar um conjunto de objetos multimídia independentes numa apresentação multimídia sincronizada. Os documentos em SMIL são documentos em XML (W3C, 1998).

Segundo o W3C (1999a), SMIL é a primeira linguagem que torna acessível ao mundo da multimídia sincronizada os benefícios da arquitetura da Web. Contém componentes, com os quais os usuários da Web estão familiarizados, como URLs (Universal Resource Locator) e os links HTML. Possui ainda uma sintaxe do tipo XML com colocação na página com base em CSS (*Cascading Style Sheet* – Folha de Estilo em Cascata).

A idéia básica é nomear componentes de mídia com URLs e programar a apresentação deles dentro de uma seqüência e/ou em paralelo. Uma apresentação SMIL típica tem as seguintes características:

a) os componentes têm tipos de mídia diferentes, como áudio, vídeo, imagem ou texto; o tempo de início e fim dos componentes é especificado e relativo a eventos.

b) a apresentação é composta de vários componentes que são acessíveis por URIs[7] (*Universal Resource* Identifier), ou seja, os arquivos são referenciados e não inclusos. Por exemplo, arquivos armazenados em um servidor da Web;

c) permite a interação do usuário, ou seja, o usuário possui controle da apresentação através de botões, tais como parar, avançar, retroceder e rebobinar;

---

[7] O termo URI é mais geral para recursos na Web, dos quais os URLs são um tipo especial. URI é um superconjunto do URLs. Este termo é usado exclusivamente em XML e padrões relacionados. Para todos os fins práticos um URI é um URL.

d) o usuário pode seguir *hyperlinks* embutidos na apresentação;

e) acesso aleatório, isto é, a apresentação pode ser iniciada em qualquer lugar;

f) controle da velocidade, por exemplo, apresentação em câmera lenta e outros.

Este trabalho alude à linguagem SMIL 2.0 ou também chamada de SMIL Boston, uma versão da linguagem SMIL 1.0. Porém, para simplificar, algumas vezes poderá estar referida simplesmente como SMIL. As definições e sintaxe aqui apresentadas são referentes às especificações do W3C de 05 de junho de 2001 (W3C, 2001a) e também da especificação de setembro de 2001(W3C, 2001b).

4.1 SMIL 2.0: uma extensão

SMIL 2.0 (W3C, 2002b) é uma extensão da linguagem SMIL 1.0, acrescida de novas características, as quais acrescentam poder de expressão da linguagem. SMIL 2.0 está organizada em um conjunto de módulos, permitindo que seus elementos e atributos sejam reutilizados em outras linguagens baseadas no padrão XML, em especial aquelas que necessitam representar relações de sincronização.

SMIL 1.0 estava centrada principalmente em animações lineares, não incluía suporte para a animação. A versão SMIL 1.0 define o tipo de documento e a semântica associada. Por outro lado, SMIL 2.0 define a funcionalidade da modularização. Esta versão da linguagem destaca e acrescenta importantes extensões, incluindo módulos reutilizáveis, animação genérica, interatividade aprimorada, tudo escrito em eXtensible Markup Language (XML).

O projeto padrão da linguagem SMIL 2.0 do W3C, permite integrar sistemas de televisão com características de interatividade da Web (MCCANNELL, 2000).

SMIL 2.0 é definida, como uma linguagem que permite integrar um conjunto de objetos multimídia independentes numa apresentação multimídia sincronizada. Com SMIL 2.0, um autor pode descrever o comportamento temporal da apresentação, associar hiperligações a objetos multimídia e descrever a disposição da apresentação na tela.

4.2 Módulos SMIL 2.0

Como a publicação de SMIL 1.0 cresceu o interesse pela integração de conceitos de SMIL com o HTML e outras linguagens de XML. Igualmente, o W3C HTML Working Group está especificando o XHTML (*Extensible Hypertext Markup Language*). A estratégia de integrar a funcionalidade com outras linguagens XML está baseado nos conceitos de modularização e perfil (*profile*). Com SMIL 2.0, por exemplo, é possível adicionar informação temporal a XHTML e a SVG (*Scalable Vector Graphics*), simplesmente importando o módulo de temporização de SMIL 2.0, em vez de construir modelos temporais e sintaxe a partir do zero.

Modularização é definida como sendo uma solução na qual a funcionalidade de uma linguagem é dividida em conjunto de elementos semanticamente relacionados. Por outro lado, perfil é a combinação destas características para resolver um problema particular. Seguem abaixo, algumas definições.

- **Elemento** - um elemento é uma representação de uma característica semântica. Um elemento tem uma representação em qualquer sintaxe.

- **Módulo** - um módulo é uma coleção de elementos semanticamente relacionados.

- **Família de módulo** - uma família de módulo é uma coleção de módulos semanticamente relacionados. Cada elemento está em uma e só uma família de módulo. Módulos em uma família de módulo geralmente são ordenados para aumentar a funcionalidade

- **Perfil:** um perfil (*profile*) é uma coleção particular de módulos a um domínio de aplicação ou linguagem. Por exemplo, o perfil de SMIL corresponde à coleção de módulos que compõem a linguagem SMIL. Em geral, um perfil incluiria só um módulo de uma família de módulo particular.

- **Família de perfil:** uma família de perfil é uma coleção de perfis que parte de um conjunto comum de módulos. Esses módulos estão definidos como obrigatórios para um perfil que deseja fazer parte daquela família de perfil. Exemplos são a família de XHTML e a família de SMIL.

A funcionalidade de SMIL é particionada em módulos baseados nas seguintes exigências:

a) assegurar que a semântica de um módulo mantenha compatibilidade com a semântica de SMIL;

b) módulos serem isomórficos com outros módulos (isomorfismo ajudará a compartilhar módulos);

c) prover funcionalidade de multimídias à linguagem XHTML[8];

d) adotar novas recomendações do W3C quando apropriadas e não entrar em conflito com outras exigências;

e) integrar suporte ao modelo de objeto de documento (DOM);

f) uma coleção de módulos poder ser "recombinada".

Para atender as exigências acima, foram criados nove módulos, que estão relacionados abaixo:

[8] XHTML é a reformulação de HTML 4.0 em XML

- Módulo de Animação
- Módulo de Controle de Conteúdo
- Módulo de Layout
- Módulo de Ligação
- Módulo de Objeto de Mídia
- Módulo de Meta Informação
- Módulo de Estrutura
- Módulo Temporal e de Sincronização
- Módulo de Efeito de Transição

Cada um destes módulos define suas propriedades, seus elementos e atributos. Cada módulo pode ser dividido em níveis (zero, um e dois). Porém, quando um perfil incluir um módulo de um nível mais alto, devem ser inclusos os módulos dos níveis mais baixos. Alguns elementos ou atributos estão rotulados com perfil específico. Isto significa que esses elementos ou atributos são opcionais ao perfil. Os módulos são complementares.

4.2.1 Módulo de Animação

Este módulo foi construído na funcionalidade da primeira versão de SMIL. Porém, algumas mudanças e extensões foram incluídas para suportar interatividade.

Este módulo incorpora animação sobre uma linha temporal (*timeline*) e um mecanismo para compor os efeitos de animações múltiplas. Inclui um conjunto de elementos de animação básica que podem ser aplicados a qualquer linguagem baseada em XML.

O módulo de animação é provido de uma estrutura que incorpora animação sobre uma linha temporal, um mecanismo para compor os efeitos de animações múltiplas. Quando este módulo é usado, ele adiciona os elementos abaixo para um modelo que

contenha os elementos, par, *seq, e excl.* Também acrescenta estes elementos no elemento *body* do módulo de estrutura.

Este módulo possui dois níveis de funcionalidade: o nível zero define a semântica para os elementos *animate, set, animateMotion, e animateColor.* Neste nível temos os atributos: *targetElement, attributeName, from, to, by, values, accumulate, additive, calcMode, path* e *origin.* No entanto, no nível um não existe um elemento especificado e os *atributos declarados são: keyTimes e keySplines.*

4.2.2 Módulo de Controle de Conteúdo

O módulo controle de conteúdo provê uma estrutura baseada em um conjunto de atributos, para controle do conteúdo. Este módulo define os elementos *switch*, *prefetch* e *uGroup*. Quando este módulo é usado, ele adiciona o elemento *switch* aos elementos *par, seq, e excl.* Também acrescenta este elemento ao elemento *body* do módulo de estrutura e no módulo de ligação. Além disso, acrescenta este elemento ao elemento *head* do módulo de estrutura.

Para o nível zero somente o elemento *switch* é requerido. Neste nível, os atributos podem ser: *SkipConten, systemBitrate, systemCaptions, systemLanguage, systemOverdubOrSsubtitle, systemRequired, systemScreenSize, systeScreenDepth, systemAudioDesc, systemOperatingSystem, systemCPU* e *systemComponent.*

No que se refere ao nível dois deste módulo, os elementos são *prefetch, uGroup e userAttributes,* todos com perfil específico. Os atributos podem ser *uState e uGroup.*

4.2.3 Módulo de Layout

O módulo de layout dispõe de uma estrutura para o espaço de componentes visuais. O módulo de layout define semântica para os elemento *layout* , *root-layout*, e *region*. Quando este módulo for usado, acrescenta o elemento de *layout* ao elemento

head do módulo de estrutura. Também acrescenta este elemento ao elemento *switch* no módulo de controle de conteúdo.

Para o nível zero os elementos *layout, region e root-layout* são referenciados. *Os atributos para este nível são type, backgroundColor, fit, left, right, top, bottom, height, width e z-index.* No que tange ao nível um o elemento *viewport* com seu atributo *soundLevel* são apresentados. Para o nível dois aparece o elemento *regPoint* com os atributos *regPoint* e *regAlign.*

4.2.4 Módulo de Ligação

O módulo de ligação fornece uma estrutura de documentos relacionados. Ele define a semântica para elemento *a* e para o elemento *área* (substituiu o elemento *anchor* da versão SMIL 1.0).

Quando este módulo é usado, acrescenta os elementos *a* e *área* aos elementos *par, seq, e excl* do módulo temporização e sincronização. Também acrescenta estes elementos ao elemento *body* no módulo de Estrutura.

Para o nível zero nenhum elemento é referenciado, porém possui os atributos sourceLevel, *destinationLevel, sourcePlaystate, destinationPlaystate, show, accesskey, tabindex, target, external* e *actuate* relacionados.No que se refere ao nível um, aparecem os elementos *a* e *area* com os atributos *href, nohref, shape e coords*

4.2.5 Módulo Objeto de Mídia

O módulo objeto de mídia provê um bloco para declaração de mídia. Ele define a semântica para os elementos *ref, animation, audio, img, video, text*, and *textstream.*

Quando este módulo é usado, os elementos *ref, animation, audio, img, video, text*, and *textstream* são adicionados ao conteúdo dos elementos *par, seq*, de excl do módulo

temporização e sincronização. Também acrescenta estes elementos ao elemento *body* do módulo de estrutura. Também acrescenta estes elementos ao conteúdo dos elementos do módulo de ligação.

Para o nível zero deste módulo temos os elementos, *ref, img, text, audio, video, animation, textstream.* Seus atributos são *abstract, alt, author, clipBegin, clipEnd, copyrigth, longdesc, src e type.*

Para o nível um os elementos possuem perfil específico. São eles *brush* e *param.* Seus atributos, *stripRepeat, readIndex e shape* também com perfil específico e ainda, os atributos *erase, name, value, valuetype, type* e *color*

Adicionalmente, como funcionalidade de mídia *streaming,* temos o elemento *rtpmap* com os atributos *payload, encoding, port, transport* e *rtpformat.*

4.2.6 Módulo Metainformação

O módulo metainformação provê um bloco para descrever um documento, para informar o usuário ou ajudar na automatização. Este módulo define semântica para o elemento *meta* e o *metadata.* Como só possui o módulo de nível zero, os elementos são os dois citados e seus atributos são *content* e *name.*

Quando este módulo é usado, é acrescentado o elemento meta ao *head.*

4.2.7 Módulo de Estrutura

O módulo de estrutura provê um nível para estruturar um documento SMIL. Este módulo define a semântica do nível zero para os elementos *smil, head,* e *body.* Este módulo é uma parte obrigatória em qualquer perfil da família SMIL.

Quando este módulo é usado, são acrescentados os atributos *id, class, xml:lang, title, xmlns* e *profile,*

4.2.8 Módulo Temporal e de Sincronização

O módulo temporal e de sincronização provê um bloco para descrever estrutura de tempo, propriedades de controle de temporização, e relações temporais entre elementos. Este módulo define a semântica para os elementos *par, seq* e *excl.* Além disso, este módulo define a semântica dos atributos *begin, dur, end, repeatCount, repeatDur,*entre outros. Este módulo é obrigatório em qualquer perfil que incorpora módulos de SMIL.

Os atributos de temporização são usados por todos os elementos dos módulos de objeto de mídia, de ligação, de controle conteúdo e no módulo de sincronização e temporização. Na realidade, só se aplica quando esses módulos forem parte do perfil. Como na integração com módulos não SMIL, os elementos deste módulo podem aparecer como atributos em vez de elementos.

Para o módulo de nível zero temos os elementos par e seq com os atributos *begin, end, endsync, dur, repeatCount, repeatDur, timeAction* e *timeContainer.* No nível um os elementos são: *excl* e *priorityClass*, com os atributos *begin, end, restart, restartDefault e fill.* Para o nível dois somente atributos são especificados: *begin, end, syncMaster* e *syncTolerance, syncToleranceDefault, syncBehavior* e *syncBehaviorDefault.*

Além destes atributos, é possível manipular o objeto de mídia através do tempo usando o atributo *speed.* Ainda, para isto, temos os atributos *accelerate, decelerate* e *autoReverse.*

4.2.9 Módulo de Efeito de transição

O módulo efeito de transição define uma taxonomia de efeitos de transição como também semântica e sintaxe por integrar efeitos em documentos de XML.

Para este módulo temos dois níveis: o nível zero apresenta o elemento *transition*, com os atributos *transition, type, subtype, starPercent, endePercent, direction, horzRepeat, vertRepeat, borderWidth, color, multiElement* e *childrenClip.* Para o nível

um temos os elementos *transition* e *transitionFilter,* ambos com o atributo *percentDone.*

4.3 Estrutura básica de um documento SMIL

Como visto, a linguagem SMIL 2.0 é dividida em módulos. Porém, esta seção descreverá a estrutura da linguagem de forma geral, usando elementos e atributos básicos.

Um documento SMIL é seqüencial, ou seja, uma composição definida em série. SMIL é muito similar a HTML e isso faz com que seja fácil de ler e de entender. Porém, existem diferenças básicas entre SMIL e HTML (COURTAUD, 1999).

a) SMIL é "case-sensitive", isto é, difere caracteres maiúsculos dos minúsculos e todos os tags precisam ser escritos em minúsculos;

b) SMIL é baseado em XML. Os tags precisam ser fechados </>.

O documento abaixo, fig. 4.1, é um código fonte SMIL. É apenas um esqueleto uma vez que não foi especificado nenhum elemento de mídia.

```
<smil>
 <head>
    <meta name="copyright" content="Nome" />
    <layout>
     <!-- tags de layout  -->
    </layout>
 </head>
 <body>
    <!-- tags de sincronização e de mídia -->
 </body>
</smil>
```

Figura 4-1 - Esqueleto de um código fonte SMIL

Um código fonte começa com <smil> e termina com </smil>. Os documentos SMIL têm duas partes: cabeçalho (*head*) e corpo (*body*), ambos precisam estar inclusos

dentro do tag <smil>. O elemento *body* contém informações associadas ao comportamento temporal e relacional do documento. Este elemento pode possuir, por exemplo, os seguintes dependentes: *a, animation, audio, img, par, ref, seq, switch, text, textstream e video.*

4.3.1 Elemento head

O elemento *head* pode conter os seguintes dependentes: *layout, meta, switch.* O elemento *head* pode conter qualquer número de elementos meta; pode, igualmente conter um elemento *layout* ou um elemento *switch.*

Tudo que diz respeito ao módulo layout, incluindo as definições de janela (dimensão e cor de fundo) é disposto entre os tags <layout> e </layout> no cabeçalho.

Para ajustar a largura e a altura da janela em que a apresentação será exibida, observe o código abaixo, fig. 4.2. Ele criará uma janela com as dimensões de 300x200 pixeis.

```
<smil>
 <head>
   <layout>
    <root-layout width="300" height="200"
          background-color="white" />
   </layout>
 </head>
 <body>
 </body>
</smil>
```

Figura 4-2 - Exemplo para definir uma janela

Como já visto anteriormente, no módulo layout, podemos encontrar elementos como *layout, root-layout, region, viewport e regPoint.*

Para inserir um elemento de mídia, precisamos especificar a região onde ele será exibido. Para tal, usamos o tag *<region>*. Este controla a posição, tamanho e escala dos

objetos multimídia. Precisamos também dar uma identidade (*id*) que identifique a região. O cabeçalho, compreendido entre os tags <head> e </head> - ficaria assim:

```
<head>
   <layout>
      <root-layout width="300" height="200"
         background-color="white" />
      <region id="ufsc_icon" left="75" top="50"
         width="32" height="32" />
   </layout>
</head>
```

Figura 4-3 - Exemplo do tag <region>

Além do atributo *id* outros atributos são necessários, veja alguns abaixo:

- **left** - define a distância em porcentagem ou pixels do elemento definido ao lado esquerdo da janela principal;
- **top** - define a distância em porcentagem ou pixels do elemento definido à parte superior da janela principal;
- **height** - define a distância em porcentagem ou pixels do elemento definido;
- **width** - define a largura em porcentagem ou pixels do elemento definido;
- **backgroundColor** - define a cor de fundo da região. O padrão é transparente;
- **title** - permite descrever o significado do elemento que foi definido.

4.3.2 Elemento body

No elemento *body,* ou seja, no corpo do documento é que inserimos os vários elementos de mídia.

Para adicionar uma mídia, deve-se incluir no elemento *body* um *tag* que descreve o tipo de mídia e a localização.

Os tipos de mídia podem ser áudio, vídeo, imagem, texto, texto streaming, etc.. A tabela a abaixo, tabela 4.1, faz uma relação entre elementos de mídia, fornecendo um exemplo de como descrevê-los dentro do elemento *body*.

Tabela 4-1 - Tipos de mídia

Elementos	Usado para	Exemplo
animation	Animação	<animation src="animação.swf" .../>
audio	Áudio	<audio src="rtsp://www.w3c.org/ SYMM/ arquivo.au" .../>
img	Imagem	
ref	Qualquer tipo de mídia não coberto por outro atributo	<ref src=" qualquermídia.rp" ... />
text	Mídia estática do tipo texto	<text src="arquivo.html" ... />
textstream	Textos do tipo streaming	<textstream src="arquivo.rt " ... />
video	Vídeo ou outro que apresenta movimento	<video src="rtsp://www.w3c.org/SYMM/ arquivo.rm" .../>

Fonte: Real – http://www.service.rel.com/help/library/smil.htm

O *player* não deduz o tipo exato do nome do elemento de objeto multimídia. Ele utiliza, por exemplo, o atributo *type*, ou a informação de tipo comunicada pelo servidor ou pelo sistema operacional. Os nomes servem apenas para a legibilidade. O atributo src é um URI que localiza o objeto.

Por exemplo, para inserir o ícone da UFSC na região chamada "ufsc_icon", deve-se usar o elemento *<img>*. O atributo *alt* é utilizado para especificar um texto alternativo, caso não seja possível apresentar a imagem. Exemplo, fig. 4.4.

O posicionamento de um elemento de mídia dentro de uma janela está baseado em dois conceitos. O posicionamento absoluto, o qual posiciona o elemento a partir do canto esquerdo superior da janela e o posicionamento relativo, pelo qual pode-se definir a posição dos elementos de mídia relativamente às dimensões da janela. Por exemplo, se quiser exibir o ícone a 30% da margem esquerda e a 50% da margem superior, modifique o código fonte e substitua os atributos "left" e "top", conforme fig. 4.5.

```
<smil>
  <head>
    <layout>
      <root-layout width="300" height="200"
        background-color="white" />
      <region id="ufsc_icon" left="75" top= "50"
        width="32" height="32" />
    </layout>
  </head>
  <body>
    <img src="ufsc32x32.gif" alt="UFSC"
      region="ufsc_icon" />
  </body>
</smil>
```

Figura 4-4 - Inserindo uma imagem

```
<smil>
 <head>
  <layout>
   <root-layout width="300" height="200"
     background-color="white" />
   <region id="ufsc_icon" left="30%" top="50%"
     width="32" height="32" />
  </layout>
 </head>
 <body>
  <img src="ufsc32x32.gif" alt="UFSC"
     region="ufsc_icon" />
 </body>
</smil>
```

Figura 4-5 - Definindo uma posição para a imagem

Além de posicionarmos um objeto de mídia, a linguagem SMIL permite temporizar e sincronizar estes objetos, adicionando parâmetros de tempo aos elementos. Por exemplo, pode-se rodar mídias uma após a outra ou rodar diversas mídias em paralelo. Como já visto, é o módulo de temporização e sincronização que é responsável pelos elementos e atributos que permitem aplicarmos sincronismo e tempo a um objeto.

4.3.3 Duração dos Elementos

Os elementos de mídia contínua possuem um valor de tempo intrínseco, ao contrário dos de mídia discreta, os quais não possuem um valor de tempo definido. Porém, a validação temporal de ambos os elementos, discretos ou contínuos, pode ser descritos através de atributos que a linguagem SMIL oferece.

Elementos têm uma duração. Esta duração da apresentação de um elemento pode ser definida através do atributo *dur*. No modelo abaixo, fig. 4.6, foi adicionado o atributo *dur*, ajustando-o para 10 segundos. Isto fará com que o ícone da UFSC seja visível por apenas 10 segundos.

```
<smil>
  <head>
    <layout>
      <root-layout width="300" height="200"
          background-color="white" />
      <region id="ufsc_icon" left="75" top="50"
          width="32" height="32" />
    </layout>
  </head>
  <body>
    <img src="ufsc32x32.gif" alt="UFSC"
        region="ufsc_icon" dur="10s" />
  </body>
</smil>
```

Figura 4-6 - Definindo um tempo de duração para apresentação

A duração de um objeto multimídia pode ser intrínseca, isto é, deduzida do conteúdo. A duração de objetos discretos, como texto ou uma imagem, é zero. Por outro lado, uma duração é explícita quando o valor é pré-estabelecido. Por exemplo, a duração de um objeto pode ser aumentada pela repetição do conteúdo e o número de exibições é determinado pelos atributos *repeatCount* e *repeatDur*.

Ex$_1$: <video src="histufsc.mpv" dur="25s" repeatCount="3" />

Neste exemplo, o vídeo será executado três vezes, levando 75 segundos para o término de sua execução.

Ex_2: <video src="histufsc.mpv" dur="25s" repeatDur="50s" />

No segundo exemplo, a execução do vídeo será repetida até completar os 50 segundos. Supondo que se deseja esperar 5 segundos antes de apresentar o ícone da UFSC, para isto usa-se o atributo *begin*, conforme fig. 4.7.

```
<smil>
  <head>
    <layout>
      <root-layout width="300" height="200"
        background-color="white" />
      <region id="ufsc_icon" left="75" top="50"
        width="32" height="32" />
    </layout>
  </head>
  <body>
    <img src="ufsc32x32.gif" alt="UFSC"
      region="ufsc_icon" dur="6s" begin="5s" />
  </body>
</smil>
```

Figura 4-7 - Definindo um tempo para iniciar a apresentação

Alguns valores podem ser especificados para iniciarem ou finalizarem a execução de uma apresentação, em resposta a um evento disparado pelo usuário. Para isso, atributos recebem valores. Por exemplo:

- **atributo *begin* com o valor *mensagem.click***: - a um clique do usuário, é apresentado um evento como resposta;

- **atributo *end* com o valor *click***: - quando o usuário clicar na apresentação, termina a apresentação;

- **atributo *end* com o valor *click e next.click:*** - ao clicar, o usuário tem a opção de encerrar ou ir para a próxima apresentação;

- **atributo *end* com o valor *mutebutton.click:*** - ao clicar em determinada apresentação que contenha este valor, encerrará a apresentação correspondente.

A seção 4.4 descreve a validade temporal e a sincronização das mídias. Nesta seção outros elementos e atributos estão referenciados para um melhor controle na apresentação das mídias.

4.3.4 Os elementos par, seq e excl

SMIL 2.0 especifica três elementos que podem controlar a apresentação de mídias. Estes elementos são *par, seq e excl.* Destes três, somente *excl* é um elemento exclusivo da versão SMIL 2.0. Ambos, par e seq, já faziam parte da versão SMIL 1.0.

Quando necessário apresentar uma seqüência de mídia, isto é, uma mídia após a outra, faz-se uso do tag *<seq>*. No exemplo, fig. 4.8, o ícone da UFSC aparece durante 10 segundos, aguarda 1 segundo e depois o ícone da informática aparece por mais 10 segundos.

```
<smil>
  <head>
    <layout>
      <root-layout width="300" height="200"
        background-color="white" />
      <region id="ufsc_icon" left="75" top="50"
        width="32" height="32" />
      <region id="info_icon" left="150" top="50"
        width="100" height="30" />
    </layout>
  </head>
  <body>
    <seq>
      <img src="ufsc32x32.gif"
        alt="UFSC"
        region="ufsc_icon" dur="10s" />
      <img src="info.gif"
        alt="Informática"
        region="info_icon" dur="10s" begin="1s" />
    </seq>
  </body>
</smil>
```

Figura 4-8 - Apresentação seqüencial

Além do tag <seq> existe o tag <par>, no qual diversos elementos de mídia podem ser apresentados ao mesmo tempo, isto é, em paralelo. Na fig. 4.9, é ilustrado como o tag <par> é usado.

```
<smil>
  <head>
    <layout>
      <root-layout width="300" height="200"
          background-color="white" />
      <region id="ufsc_icon" left="75" top="50"
          width="32" height="32" />
      <region id="info_icon" left="150" top="50"
          width="100" height="30" />
    </layout>
  </head>
  <body>
    <par>
      <img src="ufsc32x32.gif"
                    alt="UFSC"
          region="ufsc_icon" dur="10s" />
      <img src="info.gif"
          alt="Informática"
          region="info_icon" dur="10s" />
    </par>
  </body>
</smil>
```

Figura 4-9 - Apresentação em paralelo

O exemplo abaixo, fig. 4.10, usa o tag *<par>* para apresentar diferentes tipos de mídia, texto, vídeo e áudio, em paralelo.

```
    <par>
      <text src="Historia_UFSC.html" region="superior" dur="25s" />
      <video src="histufsc.mpv" region="esquerda" alt="História da UFSC" />
      <audio src="histufsc.aiff" begin="0,5s" />
    </par>
```

Figura 4-10 - Apresentação de diferentes tipos de mídia em paralelo

É possível combinar os tags *<seq>* e *<par>*. Pode-se criar composições paralelas apresentadas em seqüências. Por exemplo, apresentar em paralelo um texto, um vídeo e um áudio e na seqüência apresentar uma outra composição em paralelo e, assim,

consecutivamente. Observe o exemplo abaixo, o qual o video1 toca primeiro. Após o video1, é executado em paralelo, isto é, juntos video2 e video3. E por fim, é executado o vídeo 4.

Exemplo 1:

```
<seq>
   video1
     <par>
        video2
        video3
     </par>
   video4
</seq>
```

No exemplo 2, o video1, video2 e o video4 iniciam ao mesmo tempo. Porém, somente quando o video2 termina, é que o vídeo3 inicia.

Exemplo 2:

```
<par>
   video1
     <seq>
        video2
        video3
     </seq>
   video4
</par>
```

Para ilustrar estes agrupamentos, a fig. 4.11 corresponde ao exemplo 1 e a fig. 4.12 correspondendo ao exemplo 2.

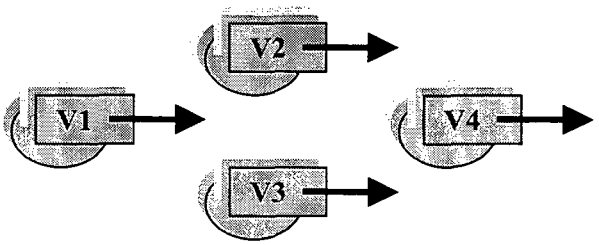


Figura 4-11 - Combinação dos elementos seq e par

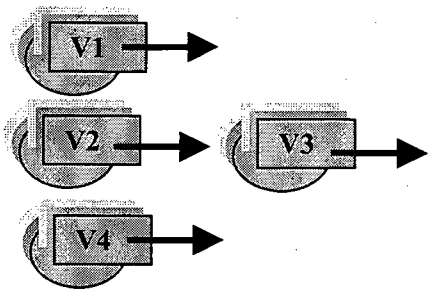


Figura 4-12 - Combinação dos elementos par e seq

Adicionalmente, as composições paralelas possuem o atributo *endsync*, o qual determina o término da composição em relação ao término de um de seus componentes. O valor desse atributo pode identificar um dos componentes, indicando que todos os demais serão terminados quando o componente selecionado terminar. O atributo pode, ainda assumir os valores primeiro (*first*), elemento referenciado (*id-ref*) ou último (*last*).

SMIL na versão 2.0 define um elemento novo que é o excl. Sua semântica está fundada no elemento *par*, mas com um diferencial, o qual somente um elemento filho pode ser executado em um determinado momento. Se um elemento começa a tocar enquanto outro estiver sendo executado, o elemento que estava sendo executado ou é pausado ou é parado. Na fig. 4.13, um exemplo do elemento excl.

```
<par>
    <video id="video" .../>
      <excl>
          <par begin="englishBtn.activateEvent" >
              <audio begin="video.begin" src="english.au" />
          </par>
          <par begin="frenchBtn.activateEvent" >
              <audio begin="video.begin" src="french.au" />
          </par>
          <par begin="swahiliBtn.activateEvent" >
              <audio begin="video.begin" src="swahili.au" />
          </par>
      </excl>
</par>
```

Figura 4-13 - Exemplo do elemento excl

Os três elementos de <par> são filhos do *excl*, e somente um pode tocar de cada vez. O áudio de cada <par> é definido para começar quando o vídeo começar. Cada áudio só pode ser ativado quando o recipiente de tempo "pai", o elemento <par>, for ativado.

Elementos são agrupados em categorias, e seus comportamentos podem ser controlados (pausa ou interrupção) usando o elemento *priorityClass*. O elemento *priorityClass*, adiciona prioridade para controlar o comportamento dos elementos filhos.

4.3.5 Os elementos de Ligação

Dentro de um documento SMIL existem elementos que servem de elos de navegação, os *links*. Esta ligação se dá através dos elementos *a* e *area*. O tag <a> da linguagem SMIL é similar ao <a> em HTML. Você especifica um URL (destino) sempre que o link for ativado por uma interação do usuário. O elemento *a* inclui os atributos listados abaixo.

a) **sourceLevel** - este atributo é relativo ao volume do áudio na apresentação que contém um link. A aplicação de controle de volume de sourceLevel é cumulativa com qualquer mídia que tem controle volume específico, na apresentação que contém a ligação. O valor de padrão é 100%.

b) **destinationLevel** - este atributo é relativo ao volume do áudio na apresentação de um recurso remoto quando contém um link. Este atributo pode introduzir valores não negativos. Também, como sourceLevel, é acumulativa com qualquer mídia que tem controle de volume específico.O valor de padrão é 100%.

c) **sourcePlaystate** - este atributo controla o comportamento temporal da apresentação que contém uma ligação, quando a ligação é percorrida. Pode ter os seguintes valores: play, pause e stop.

d) **destinationPlaystate** - este atributo controla o comportamento temporal de um recurso externo (tipicamente identificado pelo atributo *href*) quando a ligação é percorrida. Só aplica quando este recurso for um objeto de mídia contínuo. Pode ter os seguintes valores: play e pause. O valor padrão é play.

e) **alt** - quando uma mídia em particular não pode ser exibida, este atributo especifica uma mídia alternativa. É recomendado que todas as mídias contenham um atributo *alt* breve, pequeno. Ferramentas de autoria deveriam assegurar que nenhum elemento poderia ser introduzido em um documento SMIL sem este atributo. O valor deste atributo é uma string CDATA.

f) **show** - o atributo *show* controla o comportamento do objeto de origem quando o elo é percorrido. Se a apresentação destino acontecer em outro contexto, ou seja, sem afetar a apresentação de origem, o valor para o atributo show é *new*. Se a apresentação destino substituir a de origem, o valor para o atributo show é *replace*. Existe, ainda, para o atributo show, o valor *pause*, o qual define uma pausa na apresentação. O valor padrão para o atributo show é *replace*.

g) **accesskey** - este atributo nomeia uma tecla do teclado cuja ativação pelo usuário ativa uma ligação. Tem o mesmo significado e mesmo nome em HTML.

h) **tabindex** - este atributo provê a mesma funcionalidade como o atributo tabindex de HTML. Especifica a posição do elemento na ordem indicada para o documento atual. A ordem indicada define a ordem na qual os elementos receberão foco.

i) **target** - este atributo define outro meio de exibição no qual a ligação deveria ser iniciada (por exemplo, uma região SMIL, um frame HTML ou outra janela).

j) **external** - este atributo define se o destino da ligação deveria ser aberto pela aplicação atual ou alguma aplicação externa.

k) **Actuate** - o atributo actuate determina se a ligação é ativada ou não por algum evento.

O exemplo abaixo, fig. 4.14, mostra o uso dos atributos *target* e *accesskey*. A parte superior da apresentação exibe uma imagem. A partir de um clique do mouse na imagem, a apresentação é jogada para parte inferior. A mesma coisa acontece se o usuário apertar a tecla "a".

```
<smil xmlns="http://www.w3.org/2001/SMIL20/PR/Language">
  <head>
    <layout>
      <region id="origem"     height="%50"/>
      <region id="destino"    top  ="%50"/>
    </layout>
  </head>
  <body>
    <a href="encaixeSMIL.smil" target="destino" accesskey="a">
      <img region="origem" src="origem.jpg" dur="indefinite"/>
    </a>
  </body>
</smil>
```

Figura 4-14 - Exemplo dos atributos target e accesskey

O elemento *area* substituiu o elemento *anchor* da versão SMIL 1.0. Sua semântica é semelhante ao elemento área do HTML. Este elemento permite decompor um objeto multimídia em sub-partes espaciais ou temporais. As sub-partes espaciais utilizam o atributo *coords* e as temporais utilizam os atributos *begin* e *end*.

A versão 2.0 de SMIL inclui um novo atributo como parte do elemento área, o atributo *fragment*. Este atributo remete a uma parte interna de um objeto de mídia. O valor do atributo *fragment* deve ser reconhecido pelo processo que administra a mídia como uma porção capaz de ativar o objeto. Se o objeto de mídia referenciado for um arquivo HTML, então o valor do atributo *fragment* é uma âncora dentro do arquivo HTML. Se o objeto de mídia referenciado for um arquivo XML, então o valor do atributo *fragment* é um identificador de fragmento. A porção de mídia é identificada usando o sinal sustenido (#) em um URI. Por exemplo, fig. 4.15, o código de SMIL

mostra uma parte de um menu. Clicando em um item deste menu, o link ativa outro lugar dentro da apresentação.

```
<smil xmlns="http://www.w3.org/2001/SMIL20/PR/Language">
      ...
  <ref src="menu.html" region="menubar">
    <area fragment="menuitem1" href="#selecao1"/>
  </ref>
      ....
```

Figura 4-15 - Atributo fragment

O atributo *fragment* só é aplicado na primeira mídia embutida, ou seja, somente é executado no primeiro nível de profundidade

4.3.6 Elementos de Controle de Conteúdo

Uma das maiores idéias propostas com SMIL é adaptar sua apresentação às capacidades do sistema do usuário final. O elemento *switch* provê meios para definir comportamentos alternativos na apresentação de um documento. São especificados elementos alternativos, um de cada condição pode ser escolhido.

Algumas condições possíveis com o uso de *switch,* tais como escolha de um idioma, a dimensão da tela, a profundidade de tela, legenda ou dublagem, recursos com taxa de bits diferentes, o tipo de mídia que o *viewer* pode exibir para a apresentação. Como exemplo: se o modem do usuário for de 14000 bps, seria melhor fazê-lo "baixar" imagens mais "leves" (COURTAUD, 1999). Outro exemplo seria traduzir partes da apresentação de acordo com o idioma do usuário. Abaixo, a sintaxe para o elemento *<switch>*.

```
<switch>
 <!-- child1 testAttributes1 -->
 <!-- child2 testAttributes2 -->
 <!-- child3 testAttributes3 -->
</switch>
```

A regra é: o primeiro dos filhos (*children*) do tag <*switch*> cujos atributos de teste forem avaliados como verdadeiros (true) é executado. Um tag sem atributos de teste é avaliado como verdadeiro.

O elemento switch é o único elemento que pode ser definido tanto no cabeçalho como no corpo de um documento SMIL. Seus atributos são:

- **systemBitrate -** especifica a taxa de transferência disponível na rede;

- **systemCaptions** - "on | off"- permite aos autores fornecer legenda;

- **systemLanguage -** especifica o idioma desejado;

- **system-overdub-or-caption** - "caption | overdub" - seleciona entre a dublagem e legenda. Também podem ser usados os atributos *system-caption* e *systemOverdubOrSubtitle;*

- **systemRequired -** um espaço de designação XML para versões futuras;

- **systemScreenSize -** tamanho da tela= screen-height x screen-width;

- **systemScreenDepth** - dá a profundidade da paleta de cores que o *player* é capaz de exibir - valores característicos são:1 | 4 | 8 | 24 | 32;

- **systemAudioDesc -** "on | off"- seleciona entre ter áudio ou não; pretende-se com isto proporcionar para os autores a habilidade para suportar descrições auditivas da mesma forma que systemCaptions provê legendas de texto;

- **systemCPU** - este atributo especifica a CPU - uma implementação tem que permitir ao usuário a possibilidade de poder ligar a este atributo um valor não conhecido; valores que são suportados: alpha, arm, arm32, hppa1.1, m68k, mips, ppc, rs6000, vax, x86, unknown;

▪ **systemComponent** - cada URI identifica um componente do sistema, por exemplo, características ou componentes do *player*, número de canais de áudio, HW mpeg decodificador, etc.; o URI é dependente de implementação, para cada implementação é publicada uma lista de componente URIs que podem ser usados para identificar a presença de componentes dependentes da implementação;

▪ **systemOperatingSystem** - este atributo especifica o sistema operacional - uma implementação tem que permitir ao usuário a possibilidade de poder atribuir a este atributo um valor não conhecido; valores que são suportados: aix, beos, bsdi, dgux, freebsd, hpux, irix, linux, macos, ncr, nec, netbsd, nextstep, nto, openbsd, openvms, os2, osf, palmos, qnx, sinix, rhapsody, sco, solaris, sonly, sunos, unixware, win16, win32, win9x, winnt, wince, unknown;

▪ **systemOverdubOrSubtitle** - "overdub | subtitle" - *overdub* substitui um sinal de voz por outro, *subtitle* exibe um texto em um idioma diferente daquele que está sendo usado pelo áudio.

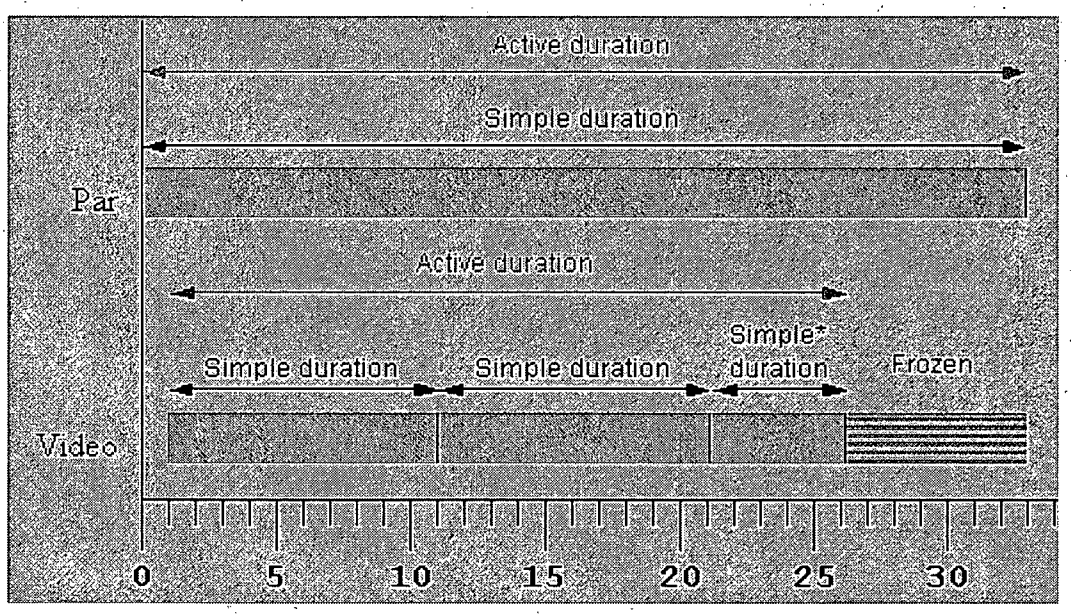
4.4 Validade Temporal e de Sincronismo de SMIL 2.0

Um diferencial que a versão SMIL 2.0 apresenta em relação à versão SMIL 1.0 é um melhor controle na validade temporal e na sincronização das mídias. O módulo de tempo e sincronização da linguagem SMIL 2.0 define elementos e atributos que coordenam e sincronizam a apresentação de mídias. Esta seção dará ênfase a alguns elementos deste módulo, pois os mesmos são objetos de estudo deste trabalho.

O módulo de sincronização provê atributos que podem ser usados para especificar o comportamento de um elemento no tempo. Elementos têm um início e uma duração simples. Um início pode ser especificado de vários modos, por exemplo, um elemento pode começar em um determinado momento, ou baseado em outro elemento, ou quando algum evento acontece (como um click de mouse). A duração simples define a duração de apresentação básica de um elemento. Podem ser definidos elementos para repetir a

duração simples, várias vezes ou em uma quantidade de tempo. Ao terminar a apresentação o elemento poderá ser removido ou congelado (W3C, 2001a).

A fig. 4.16 (W3C, 2001a) ilustra o suporte básico de um elemento <vídeo> repetindo dentro um elemento <par>. O vídeo inicia 1s após o elemento <par> e tem duração de 10s, repete 2,5 vezes completando sua apresentação após 25s. No tempo restante o vídeo permanece congelado. A fig 4.16 Ilustra também, parte do código, onde isto é definido. O valor *simple** indica que a duração simples é parcial, isto é, termina mais cedo.




```
<par begin="0s" dur="33s">
   <video begin="1s" dur="10s" repeatCount="2.5" fill="freeze"
         ...
   />
</par>
```

Figura 4-16 - Validação temporal das mídias

A duração simples e qualquer efeito de repetição são combinados para definir a duração ativa. Quando um elemento não especificar uma duração simples, é definida uma duração simples implícita para o elemento. Quando a duração ativa de um elemento terminar, o elemento pode ser removido da apresentação ou congelado.

O módulo de sincronização define como são interpretados os elementos e atributos num gráfico de tempo. O gráfico de tempo é uma estrutura dinâmica. Em um ambiente ideal, executaria a apresentação conforme especificada. Porém, limitações, como demora da rede, podem influenciar na execução da mídia.SMIL 2.0 inclui alguns atributos que permitem o autor controlar o comportamento de sincronização em tempo de apresentação.

Vários são os atributos que fazem parte do módulo de sincronização. Porém, nem todos serão descritos. Esta seção dará enfoque aos atributos que fazem parte do objeto de estudo deste trabalho. Inicialmente, serão abordados os atributos *dur, min* e *max.*

4.4.1 O atributo dur

O atributo *dur* especifica o tempo de duração para um elemento. Este valor deve ser maior do que zero. Se o elemento não tem um valor válido, a duração definida é a duração implícita do elemento. O valor do atributo pode ser:

a) um valor definido em valores de relógio, ou seja, em quantidade de tempo;

b) *media* - especifica a duração simples, como a duração de mídia intrínseca - isto só é válido para elementos que definem mídia;

c) *indefinite* - especifica a duração simples como indefinido.

Nas mídias estáticas, discretas, a duração implícita é definida para ser zero. Se for especificado um valor para *dur* mais curto que a duração implícita do elemento, a duração implícita será cortada. Se for especificada uma duração simples mais longa que

a duração implícita, é estendida a duração implícita do elemento à duração simples especificada. Por exemplo:

a) para um elemento de mídia discreto, as mídias serão mostradas durante o tempo especificado;

b) para um elemento de mídia contínuo, o estado final da mídia será mostrado, do fim da duração da mídia intrínseca até o fim da duração simples especificada;

c) para os elementos seq, par ou excl, se o comportamento é "freeze", os elementos filhos serão congelados até o fim da duração simples.

O exemplo abaixo, fig. 4.17, mostra o atributo dur. Neste caso o áudio iniciará em 5s depois do elemento <par> com duração simples de 4s.

```
<par>
  <audio src="musica.au" begin="5s" dur="4s" />
</par>
```

Figura 4-17 - Atributo dur - I

Elementos também podem ser especificados para começar com respeito a um evento. No exemplo seguinte, fig. 4.18, o elemento imagem aparece quando o usuário clica no elemento "show". Após o evento, o tempo de duração da imagem é de 3.5 segundos.

```
<smil ...>
  ...
    <text id="show" ... />
    <img begin="show.activateEvent" dur="3.5s" ... />
  ...
</smil ...>
```

Figura 4-18 - Atributo dur - II

SMIL 2.0 provê um controle adicional na duração ativa dos elementos. O atributo *end* permite controlar a duração ativa, limitando-a. No exemplo, fig. 4.19, a duração ativa terminará em 10 segundos, e cortará a duração simples definida para ser 20 segundos.

```
<par>
    <audio src="musica.au" dur="20s" end="10s" ... />
</par>
```

Figura 4-19 - Atributo dur e end

Existe a possibilidade de controlar o tempo para que duas mídias, que estejam inseridas num tag <par>, possam começar em tempos diferentes. No exemplo, figura 4.20, foi utilizado, além do atributo *dur*, os atributos *clipBegin* e *clipEnd*. Neste caso, a primeira mídia inicia imediatamente, mas realmente começa tocar aos 30.4 segundos em relação à linha de tempo, tocando exatamente durante 30 segundos.

```
<par>
 <audio src="song1.rm" clipBegin="30.4s" dur="30s"/>
 <audio src="song2.rm" begin="28s" clipBegin="2.4s"clipEnd="13.7s"/>
</par>
```

Figura 4-20 - Atributos clipBegin e clipEnd

O tempo da segunda mídia será iniciado aos 28 segundos. Isso significa que sobrepõe a primeira mídia por 2 segundos. Começa aos 2,4 segundos na linha de tempo e terminará aos 13,7 segundos na linha de tempo, tocando assim, durante 11.3 segundos. O tempo total para este grupo é de 39.3 segundos. Sendo, 30 segundos para o primeiro clipe, mais 11,3 segundos para o segundo clipe, menos os 2 segundos sobrepostos. A fig 4.21 ilustra as relações das linhas de tempo do clipe para a linha de apresentação global.

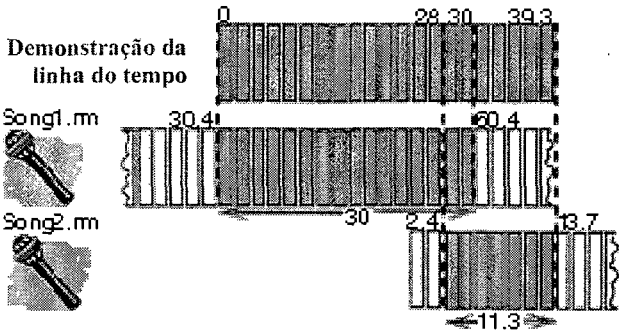


Figura 4-21 - Linha temporal das mídias

Mesmo com a possibilidade de controlar quando um elemento começa a tocar em relação a outro ou em relação à linha de tempo, o atributo *dur* possui pouca flexibilidade para especificar a validade temporal dos objetos, pois ele apenas atribui uma duração exata para apresentação de mídia.

A versão SMIL 2.0 introduziu novos atributos para o controle da duração de um elemento, os atributos *min* e *max*. Estes atributos proporcionam uma maneira de controlar o mínimo e o máximo da duração ativa de um elemento.

4.4.2 Os atributos min e Max

O atributo *min* especifica o valor mínimo de duração ativa de um elemento e pode ser especificado por um valor de tempo, que deve ser maior ou igual a zero. Também poderá ser atribuído o valor *media,* o qual especifica o valor mínimo da duração ativa como a duração da mídia intrínseca. Se houver algum erro na sintaxe o argumento para *min* será ignorado. O valor padrão para *min* é zero.

O atributo *max* especifica o valor máximo de duração ativa de um elemento e pode ser especificado por um valor de tempo, que deve ser maior que zero. Também poderá ser atribuído o valor *media,* o qual especifica o valor máximo da duração ativa como a duração da mídia intrínseca. Se for atribuído o valor *indefinite* para o atributo *max,* significa que o valor máximo está indefinido, não limitando o tempo do elemento. O valor padrão é *indefinite.*

Se forem especificados atributos *min* e *max,* então o valor do atributo *max* deve ser maior ou igual o valor do atributo *min.* Se esta exigência não for cumprida, ambos atributos serão ignorados. Algumas regras devem ser observadas para os atributos *min* e *max.*

a) Se a duração do elemento é maior que o valor de *max.* A duração é definida para ser igual ao valor de *max.* Observe o exemplo da fig. 4.22 onde o vídeo tem uma duração implícita de 15 segundos, porém tocará apenas durante 10 segundos.

```
<smil ...>
  ...
    <par >
        <video id="video_de_15s" max="10s".../>
    </par>
  ...
</smil>
```

Figura 4-22 - Atributo Max

b) Se a duração do elemento é menor que o valor de *min.* A duração do elemento fica igual ao valor de *min.* Alguns exemplos são demonstrados abaixo.

- Na fig. 4.23, se um evento acontecer antes de 10 segundos (por exemplo, um clique) não interromperá o vídeo imediatamente, o vídeo tocará até completar os 10 segundos e então pára. Se um evento acontece depois de 10 segundos, o vídeo toca até que o evento aconteça. Note que o *end* só é considerado se um evento acontecer depois de 10 segundos.

```
<smil ...>
  ...
    <par >
          <video id="video_de_15s" repeatDur="indefinite" end="activateEvent"
            min="10s".../>
    </par>
  ...
</smil>
```

Figura 4-23 - Atributo min

- Na fig. 4.24, se um evento acontecer no primeiro vídeo aos 5 segundos, então é computada a duração simples para 5 segundos. Em relação ao atributo *fill*, os dois vídeos ficarão congelados entre 5 e 12 segundos.

```
<smil ...>
  ...
    <par endsync="first" min="12s" fill="freeze" >
        <video id="video_of_15s" end="click" ...>
        <video id="video_of_10s" .../>
    </par>
  ...
</smil>
```

Figura 4-24 - Atributo fill

- Na fig. 4.25, é definida uma duração simples de 5 segundos, e o atributo *min* define a duração ativa para ser 12 segundos. Como o valor padrão de *fill* é "remover", nada é mostrado entre 5 segundos e 12 segundos.

```
<par dur="5s" min="12s" >
  <video id="video_de_15s"/>
  <video id="video_de_10s" />
</par>
```

Figura 4-25 - Atributo dur e min

- Se um elemento é definido para começar antes de seu pai (por exemplo, com um valor negativo), a duração de *min* não será afetada. No exemplo seguinte, fig. 4.26 a imagem será exibida durante 2 segundos.

```
<par>
  <img id="imagem" begin="-5s" min="7s" dur="5s" .../>
</par>
```

Figura 4-26 - Atributo begin com valor negativo – I

- Para o exemplo seguinte, fig. 4.27, a imagem não será exibida.

```
<par>
  <img id="imagem" begin="-5s" min="4s" dur="2s" .../>
</par>
```

Figura 4-27 - Atributo begin com valor negativo - II

- O atributo *min* também pode ser definido para aceitar valores de tempo negativos. O exemplo abaixo, fig. 4.28, a imagem será exibida durante 2 segundos.

```
<par>
  <img id="imagem" begin="-5s" min="7s" dur="5s" .../>
</par>
```

Figura 4-28 - Atributo min , begin e dur

Com a versão SMIL 2.0 muitos problemas relacionados ao controle de tempo e de sincronização apresentados na versão 1.0 foram resolvidos. Com a inclusão dos atributos *min* e *max*, ficou mais flexível para especificar a validade temporal dos objetos em relação à versão SMIL 1.0. Esta flexibilidade permite melhor qualidade de exibição mas não reduz as chances de ocorrerem inconsistências temporais devido ao congestionamento da rede.

4.5 Controlando a Sincronização em Tempo de Execução

Em ambientes distribuídos, os processamentos dos componentes estão sujeitos a variações temporais, ou seja, retardo da rede, tempo de acesso à base de dados, entre outros. Uma proposta da linguagem SMIL 2.0 para definir sincronização contínua entre elementos de mídia em tempo de execução é o atributo *syncBehavior*. Um exemplo de sincronização contínua é a sincronização labial, no qual áudio e vídeo de uma pessoa que esteja falando deve estar sincronizado.

4.5.1 Atributo syncBehavior

O atributo *syncBehavior* controla se o elemento de mídia pode deslizar enquanto o resto do documento continua sendo reproduzido, ou se tem que esperar até todo processo estar completo. Este controle é conseguido através de alguns atributos:

a) **canSlip** - permite que o elemento associado possa deslizar com respeito ao elemento pai - quando este valor for usado, qualquer valor do atributo *syncTolerance* é ignorado;

b) **locked** - força o elemento associado para manter sincronismo com o elemento pai - isto pode ser liberado com o uso do atributo *syncTolerance*;

c) **independent** - declara uma linha independente, e ignorará qualquer operação do elemento pai;

d) **default** - o comportamento de sincronização em tempo de execução para o elemento é determinado pelo valor do atributo *syncBehaviorDefault* - este é o valor padrão.

O *canSlip, locked, independent* e *inherit* especificam que o comportamento de sincronização do elemento em tempo de execução é o respectivo valor. Por outro lado, *inherit* especifica que o valor deste atributo é herdado do elemento pai do atributo *syncBehaviorDefault*. Se não houver nenhum elemento pai, o valor dependerá da implementação. O argumento *inherit* é o padrão.

No exemplo da fig. 4.29, os elementos de áudio e vídeo estão definidos em "locked", para manter o sincronismo labial, mas para o elemento *discurso,* de <par>, é permitido deslizar.

```
<par>
  <animacao src="..." />
          ...
          <par id="discurso" syncBehavior="canSlip" >
            <video src="discurso.mpg" syncBehavior="locked" />
            <audio src="discurso.au"  syncBehavior="locked" />
          </par>
          ...
</par>
```

Figura 4-29 - Atributo syncBehavior

Se o vídeo ou áudio tem que pausar devido a problemas de entrega na rede, o "discurso" inteiro pausará para manter o sincronismo. Porém, o resto do documento, inclusive o elemento de animação continuará normalmente a tocar.

4.5.2 Atributo syncTolerance

Com o atributo *syncTolerance* se consegue um controle adicional. Ele define a tolerância de sincronização, isto é, especifica a quantidade de deslizes que podem ser ignorados para um elemento. Na verdade, uma pequena variação pode ser ignorada permitindo um desempenho global. Por exemplo, ele define a diferença máxima de tempo entre o áudio e vídeo em uma sincronização labial.

Esta diferença pode ser definida por um valor de tempo (valor de relógio). O padrão é definido pelo valor do atributo *syncToleranceDefault*. Este atributo define um valor padrão (valor de relógio) de tolerância para estabelecer sincronização em tempo de execução. Ainda, possui o argumento `inherit` o qual especifica que o valor deste atributo é herdado do elemento pai do atributo syncToleranceDefault. Se não houver nenhum elemento pai, o valor dependerá da implementação, mas não deverá ser maior do que dois segundos. O argumento *inherit* é o padrão.

4.5.3 Atributo syncMaster

O atributo *syncMaster* permite definir explicitamente o elemento que vai controlar a sincronização. Na prática, mídia lineares necessitam ter o *syncMaster*, ao passo que mídias não lineares podem ser ajustadas mais facilmente. Todavia, um *player* nem sempre pode determinar qual mídia se comporta de uma forma linear e qual se comporta em uma forma não linear. Além disso, quando houver elementos lineares múltiplos ativos num determinado ponto, o *player* nem sempre pode tomar a decisão certa para solucionar conflitos de sincronismo. O atributo de *syncMaster* permite o autor especificar o elemento que tem mídia linear.

O atributo *syncMaster* é um atributo booleano que força outros elementos para sincronizar com o elemento em questão. O valor padrão é falso.

4.6 DTD SMIL

Como já definido anteriormente em XML, um DTD é uma descrição formal do documento. É um mecanismo para descrever cada objeto (elemento, atributo, etc.). Possui o papel de especificar quais elementos ou atributos e em que local do documento são permitidos.

A funcionalidade da linguagem SMIL é particionada em módulos, da mesma forma seu DTD. Cada um destes módulos define suas propriedades, seus elementos e atributos. Os módulos são complementares e se dividem em:

- Módulo de Animação
- Módulo de Controle de Conteúdo
- Módulo de Layout
- Módulo de Ligação
- Módulo de Objeto de Mídia
- Módulo de Meta Informação
- Módulo de Estrutura

- Módulo de Tempo e Sincronização
- Módulo de Efeito de Transição

Seguindo o mecanismo modular, a especificação SMIL 2.0 faz a declaração do elemento (por exemplo, <!ELEMENT...>) e declarações de listas de atributos (por exemplo, <!ATTLIST...>). Um módulo SMIL é provido para ser funcional na especificação SMIL 2.0, quer dizer, há um arquivo para cada módulo SMIL 2.0, por exemplo, um arquivo para módulo de animação, um para o módulo layout, um para módulo estrutura, etc.

A listagem abaixo é uma cópia do DTD do módulo de tempo e sincronização, o qual está disponível com os demais módulos, no endereço www.w3c.org/audiovideo.

```
1 <!—Módulo de Tempo e Sincronização SMIL ===== -->

2 <!-- ===== Elementos de tempo sincronização==================-->

3 <!ENTITY % BasicTimeContainers.module "IGNORE">
4 <![%BasicTimeContainers.module;[
5 <!ENTITY % par.content "EMPTY">
6 <!ENTITY % seq.content "EMPTY">
7 <!ENTITY % par.attrib  "">
8 <!ENTITY % seq.attrib  "">
9 <!ENTITY % seq.qname   "seq">
10 <!ENTITY % par.qname   "par">

11 <!ELEMENT %seq.qname; %seq.content;>
12 <!ATTLIST %seq.qname; %seq.attrib;
13  %Core.attrib;
14  %I18n.attrib;
15  %Description.attrib;
16 >

17  <!ELEMENT %par.qname; %par.content;>
18  <!ATTLIST %par.qname; %par.attrib;
19   %Core.attrib;
20   %I18n.attrib;
21   %Description.attrib;
22  >
```

```
23 ]]> <!—-fim de BasicTimeContainers.module -->

24 <!ENTITY % ExclTimeContainers.module "IGNORE">
25 <![%ExclTimeContainers.module;[
26 <!ENTITY % excl.content        "EMPTY">
27 <!ENTITY % priorityClass.content "EMPTY">
28 <!ENTITY % excl.attrib         "">
29 <!ENTITY % priorityClass.attrib  "">
30 <!ENTITY % excl.qname          "excl">
31 <!ENTITY % priorityClass.qname   "priorityClass">

32 <!ELEMENT %excl.qname; %excl.content;>
33 <!ATTLIST %excl.qname; %excl.attrib;
34 %Core.attrib;
35 %I18n.attrib;
36 %Description.attrib;
37 >

38 <!ELEMENT %priorityClass.qname; %priorityClass.content;>
39 <!ATTLIST %priorityClass.qname; %priorityClass.attrib;
40 peers     (stop|pause|defer|never) "stop"
41 higher    (stop|pause)          "pause"
42 lower     (defer|never)         "defer"
43 pauseDisplay (disable|hide|show )   "show"
44 %Description.attrib;
46 %Core.attrib;
47 %I18n.attrib;
48 >
49 ]]> <!—Fim de ExclTimeContainers.module -->
50 <!—Fim módulo -->
```

Este DTD consta de 50 linhas de códigos, as quais foram enumeradas para melhor esclarecimento e visualização. Algumas observações:

- quando fizer uso de comentários dentro de um DTD, deve-se usar os sinais <!-- e os sinais --> , conforme linha 1, 2 e 23;

- todos os elementos de um DTD são criados através da declaração ELEMENT, representada conforme linhas 11, 17 e 32 e 38;

- todos os atributos devem ser declarados na DTD através de ATTLIST, conforme linhas 12, 18, 33 e 39.

A linguagem SMIL, como a linguagem XML, não trabalha com arquivos e sim com entidades. Porém, geralmente, as entidades são armazenadas como arquivos, mas isto não é regra. A XML possui muitos tipos de entidade, classificadas de acordo com alguns critérios: entidades internas e externas, entidades gerais ou de parâmetros e entidades desmembradas ou não desmembradas.

As entidades gerais podem aparecer em qualquer lugar do texto ou na marcação. Quando uma entidade faz referência a imagem, som ou a um outro documento, diz-se que é uma entidade geral externa. As entidades gerais são declaradas com a marcação <!ENTITY seguida pelo nome da entidade e o tradicional sinal de maior. Nas entidades de parâmetros existe um caractere extra na declaração antes do nome da entidade, o % (símbolo de porcentagem). As referências de entidade de parâmetro também substituem o símbolo & pelo sinal de %.

Por outro lado, à medida que um DTD amadurece, poderá necessitar de alterações, que poderão ser incompatíveis com o DTD anterior. Durante a construção de um novo DTD, pelo fato de existirem documentos novos e antigos fica difícil manter o DTD. Para fazer este gerenciamento XML oferece a opção de incluir ou ignorar o DTD, esta possibilidade é chamada de *seções condicionais*. Para incluir ou ignorar um DTD, é suficiente alternar entre as palavras INCLUDE e IGNORE. O exemplo do DTD acima mostra como usar seções condicionais, conforme linhas 3 e 24, ambas entidades de parâmetros ignoradas através da declaração IGNORE.

Os módulos SMIL são usados nas definições da especificação do perfil da linguagem SMIL 2.0. É recomendado o uso destes módulos para definir outros perfis de SMIL.

4.7 Criando um DTD para um novo perfil da linguagem SMIL

A maioria dos arquivos de um DTD são reutilizados pelos diferentes perfis. São reusados os arquivos que definem os tipos de dados e os atributos comuns. Também pode ser reusado, se necessário, o arquivo *qname* que introduz o prefixo namespace e

ainda pode ser reutilizada a estrutura do arquivo, o qual cuida da ordem apropriada dos arquivos.

Os arquivos que devem ser diferentes para cada perfil da linguagem são o arquivo administrador do DTD e o arquivo modelo do documento. O arquivo administrador define que módulos são usados. O arquivo modelo de documento define o modelo do documento estendido.

4.7.1 Arquivo Administrador

O arquivo administrador faz referência ao documento através da declaração DOCTYPE. Seu trabalho principal é definir o arquivo modelo de documento e que o módulo de SMIL está sendo usado. Também pode definir um namespace opcional a ser usado em todos os prefixos namespace. Por exemplo, todo elemento SMIL é nomeado para "apto".

O exemplo seguinte pode ser adicionado para iniciar o arquivo:

```
<!ENTITY % SMIL.prefixed "INCLUDE" >
<!ENTITY % SMIL.prefix "apto" >
```

Os elementos definidos nos módulos, por exemplo <video>, será analisado gramaticalmente como <apto:video>. Isto também se refere a atributos SMIL que aparecem em outros elementos, por exemplo, *begin* se torna *apto:begin*. O padrão é que o prefixo qname seja vazio (*empty*).

Depois destas definições, o arquivo administrador inclui o arquivo de estrutura (que incluirá subseqüentemente os tipos de dados, os atributos comuns, qname e o arquivo modelo de documento) depois, os arquivos de módulos SMIL são incluídos para serem usado por este perfil.

4.7.2 Arquivo Modelo de documento

O arquivo modelo de documento contém as entidades XML que são usadas pelo módulo SMIL para definir os conteúdos dos modelos e listas de atributos dos elementos naquele perfil.

Os conteúdos dos módulos geralmente diferem do perfil, ou contêm elementos de outros módulos. Para evitar estas dependências no módulo SMIL, os conteúdos dos módulos precisam ser definidos no arquivo modelo de documento. O padrão do conteúdo é definido como vazio.

Os arquivos SMIL definem apenas uma lista de atributos padrão para os elementos. Esta lista padrão contém apenas os atributos que são definidos no módulo SMIL. Todos os outros atributos precisam ser adicionados a esta lista padrão definindo as entidades XML apropriadas. Por exemplo, o arquivo módulo objeto de mídia só adiciona os atributos de mídia nos objetos de mídia; outros atributos, como os atributos de tempo e cronometragem, são adicionados a esta lista pelo arquivo modelo de documento.

As tabelas abaixo descrevem todos os arquivos usados no DTD SMIL 2.0:

Tabela 4-2 - Arquivo administrador do DTD SMIL 2.0

Arquivo administrador	
-//W3C//DTD SMIL 2.0//EN	`http://www.w3.org/2001/SMIL20/SMIL20.dtd`

Fonte: W3C – http://w3.org/TR/smil2.0-profile.html

Tabela 4-3 - Arquivo modelo de documento do DTD SMIL 2.0

Arquivo modelo de documento	
-//W3C//ENTITIES SMIL 2.0 Document Model 1.0//EN	`http://www.w3.org/2001/SMIL20/smil-model-1.mod`

Fonte: W3C – http://w3.org/TR/smil2.0-profile.html

Tabela 4-4 - Arquivos dos módulos SMIL 2.0

Arquivos dos módulos SMIL 2.0	
-//W3C//ELEMENTS SMIL 2.0 Animation//EN	http://www.w3.org/2001/SMIL20/SMIL-anim.mod
-//W3C//ELEMENTS SMIL 2.0 Content Control//EN	http://www.w3.org/2001/SMIL20/SMIL-control.mod
-//W3C//ELEMENTS SMIL 2.0 Layout//EN	http://www.w3.org/2001/SMIL20/SMIL-layout.mod
-//W3C//ELEMENTS SMIL 2.0 Linking//EN	http://www.w3.org/2001/SMIL20/SMIL-link.mod
-//W3C//ELEMENTS SMIL 2.0 Media Objects//EN	http://www.w3.org/2001/SMIL20/SMIL-media.mod
-//W3C//ELEMENTS SMIL 2.0 Document Metainformation//EN	http://www.w3.org/2001/SMIL20/SMIL-metainformation.mod
-//W3C//ELEMENTS SMIL 2.0 Document Structure//EN	http://www.w3.org/2001/SMIL20/SMIL-struct.mod
-//W3C//ELEMENTS SMIL 2.0 Timing//EN	http://www.w3.org/2001/SMIL20/SMIL-timing.mod
-//W3C//ELEMENTS SMIL 2.0 Transition//EN	http://www.w3.org/2001/SMIL20/SMIL-transition.mod

Fonte: W3C – http://w3.org/TR/smil2.0-profile.html

Tabela 4-5 - Outros arquivos da linguagem SMIL 2.0

Outros serviços: tipos de dados, atributos comuns, nomes qualificados e estrutura dos arquivos	
-//W3C//ENTITIES SMIL 2.0 Datatypes 1.0//EN	http://www.w3.org/2001/SMIL20/smil-datatypes-1.mod
-//W3C//ENTITIES SMIL 2.0 Common Attributes 1.0//EN	http://www.w3.org/2001/SMIL20/smil-attribs-1.mod
-//W3C//ENTITIES SMIL 2.0 Qualified Names 1.0//EN	http://www.w3.org/2001/SMIL20/smil-qname-1.mod
-//W3C//ENTITIES SMIL 2.0 Modular Framework 1.0//EN	http://www.w3.org/2001/SMIL20/smil-framework-1.mod

Fonte: W3C – http://w3.org/TR/smil2.0-profile.html

A listagem de código abaixo é do DTD administrador da linguagem SMIL 2.0 Este módulo administra os demais módulos do DTD. Este módulo é identificado através dos identificadores PUBLIC e SYSTEM, conforme abaixo:

```
<!--        PUBLIC "-//W3C//DTD SMIL 2.0//EN"
            SYSTEM "http://www.w3.org/2001/SMIL20/PR/SMIL20.dtd"  - ->

<!ENTITY % NS.prefixed "IGNORE" >
<!ENTITY % SMIL.prefix "" >
```

```
<!-- Define o conteúdo do modelo -->
<!ENTITY % smil-model.mod
   PUBLIC "-//W3C//ENTITIES SMIL 2.0 Document Model 1.0//EN"
         "smil-model-1.mod" >

<!-- Sistema Modular..... -->
<!ENTITY % smil-framework.module "INCLUDE" >
<![%smil-framework.module;[
<!ENTITY % smil-framework.mod
    PUBLIC "-//W3C//ENTITIES SMIL 2.0 Modular Framework 1.0//EN"
         "smil-framework-1.mod" >
    %smil-framework.mod;]]>

<!-- O perfil de SMIL 2.0 inclui as seguintes seções:
Módulo de Animação
Módulo Controle de Conteúdo
Módulo de Layout
Módulo de Ligação
Módulo de Objeto de Mídia
Módulo de Meta Informação
Módulo de Estrutura
Módulo de Tempo e de Sincronização
Integração de SMIL Timing (SMIL Temporal) com outras linguagens
Módulo de Efeito de transição

    O módulo Objeto de Mídia Streaming (fluindo) é opcional
-->

<!ENTITY % streamingmedia.model "IGNORE">
<![%streamingmedia.model;[
 <!ENTITY % streaming-mod
  PUBLIC "-//W3C//ELEMENTS SMIL 2.0 Streaming Media Objects//EN"
  "SMIL-streamingmedia.mod">
 %streaming-mod;
]]>

<!ENTITY % anim-mod
 PUBLIC "-//W3C//ELEMENTS SMIL 2.0 Animation//EN"
 "SMIL-anim.mod">
<!ENTITY % control-mod
 PUBLIC "-//W3C//ELEMENTS SMIL 2.0 Content Control//EN"
 "SMIL-control.mod">
<!ENTITY % layout-mod
 PUBLIC "-//W3C//ELEMENTS SMIL 2.0 Layout//EN"
 "SMIL-layout.mod">
<!ENTITY % link-mod
 PUBLIC "-//W3C//ELEMENTS SMIL 2.0 Linking//EN"
 "SMIL-link.mod">
```

```
<!ENTITY % media-mod
 PUBLIC "-//W3C//ELEMENTS SMIL 2.0 Media Objects//EN"
 "SMIL-media.mod">
<!ENTITY % meta-mod
 PUBLIC "-//W3C//ELEMENTS SMIL 2.0 Document Metainformation//EN"
 "SMIL-metainformation.mod">
<!ENTITY % struct-mod
 PUBLIC "-//W3C//ELEMENTS SMIL 2.0 Document Structure//EN"
 "SMIL-struct.mod">
<!ENTITY % timing-mod
 PUBLIC "-//W3C//ELEMENTS SMIL 2.0 Timing//EN"
 "SMIL-timing.mod">
<!ENTITY % transition-mod
 PUBLIC "-//W3C//ELEMENTS SMIL 2.0 Transition//EN"
 "SMIL-transition.mod">

%struct-mod;
%anim-mod;
%control-mod;
%meta-mod;
%layout-mod;
%link-mod;
%media-mod;
%timing-mod;
%transition-mod;
```

A listagem de código seguinte é parte do DTD SMIL 2.0. Este arquivo define o modelo de documento da linguagem SMIL 2.0. Todos os atributos e o conteúdo dos módulo são definidos na segunda parte do arquivo. A primeira parte simplifica o uso dos módulos:

```
<!-- ================================================================== -->
<!-- SMIL 2.0 Módulo Modelo de Documento ============================== -->
<!--arquivo: smil-model-1.mod          -->

<!-- ================== Primeira parte ================================ -->


<!-- ================== Util: Head ==================================== -->
<!ENTITY % head-meta.content       "metadata">
<!ENTITY % head-layout.content     "layout|switch">
<!ENTITY % head-control.content    "customAttributes">
<!ENTITY % head-transition.content "transition+">
```

```
<!--========== Util: Body - Content Control ========== -->
<!ENTITY % content-control "switch|prefetch">
<!ENTITY % content-control-attrs "%Test.attrib; %CustomTest.attrib;
%SkipContent.attrib;">

<!--========== Util: Body - Animation ========== -->
<!ENTITY % animation.elements "animate|set|animateMotion|animateColor">

<!--========== Util: Body - Media ========== -->

<!ENTITY % media-object "audio|video|animation|text|img|textstream|ref|brush
                |%animation.elements;">

<!--========== Util: Body - Timing ========== -->
<!ENTITY % BasicTimeContainers.class "par|seq">
<!ENTITY % ExclTimeContainers.class "excl">
<!ENTITY % timecontainer.class
 "%BasicTimeContainers.class;|%ExclTimeContainers.class;">
<!ENTITY % timecontainer.content
 "%timecontainer.class;|%media-object;|%content-control;|a">

<!ENTITY % smil-basictime.attrib "
 %BasicInlineTiming.attrib;
 %BasicInlineTiming-deprecated.attrib;
 %MinMaxTiming.attrib;
">

<!ENTITY % timecontainer.attrib "
 %BasicInlineTiming.attrib;
 %BasicInlineTiming-deprecated.attrib;
 %MinMaxTiming.attrib;
 %RestartTiming.attrib;
 %RestartDefaultTiming.attrib;
 %SyncBehavior.attrib;
 %SyncBehaviorDefault.attrib;
 %FillDefault.attrib;
">

<!--========== Segunda Parte ========== -->
<!-- Nesta parte, todos os módulos estão definidos. Porém, para exemplificar, somente o
módulo de Controle de Tempo é descrito.
O modelo atual e as definições dos atributos seguem abaixo:
 -->

<!-- ========== Timing ========== -->
<!ENTITY % BasicInlineTiming.module     "INCLUDE">
<!ENTITY % SyncbaseTiming.module        "INCLUDE">
```

```
<!ENTITY % EventTiming.module          "INCLUDE">
<!ENTITY % WallclockTiming.module      "INCLUDE">
<!ENTITY % MultiSyncArcTiming.module   "INCLUDE">
<!ENTITY % MediaMarkerTiming.module    "INCLUDE">
<!ENTITY % MinMaxTiming.module         "INCLUDE">
<!ENTITY % BasicTimeContainers.module  "INCLUDE">
<!ENTITY % ExclTimeContainers.module   "INCLUDE">
<!ENTITY % PrevTiming.module           "INCLUDE">
<!ENTITY % RestartTiming.module        "INCLUDE">
<!ENTITY % SyncBehavior.module         "INCLUDE">
<!ENTITY % SyncBehaviorDefault.module  "INCLUDE">
<!ENTITY % RestartDefault.module       "INCLUDE">
<!ENTITY % FillDefault.module          "INCLUDE">

<!ENTITY % par.attrib "
          %Endsync.attrib;
     %Fill.attrib;
          %timecontainer.attrib;
          %Test.attrib;
          %CustomTest.attrib;
          %Region.attrib;
">
<!ENTITY % seq.attrib "
     %Fill.attrib;
          %timecontainer.attrib;
          %Test.attrib;
          %CustomTest.attrib;
          %Region.attrib;
">
<!ENTITY % excl.attrib "
          %Endsync.attrib;
     %Fill.attrib;
          %timecontainer.attrib;
          %Test.attrib;
          %CustomTest.attrib;
          %Region.attrib;
     %SkipContent.attrib;
">
      <!ENTITY %  par.content "(%timecontainer.content;)*">
      <!ENTITY %  seq.content "(%timecontainer.content;)*">
      <!ENTITY % excl.content "((%timecontainer.content;)*|priorityClass+)">

      <!ENTITY % priorityClass.attrib  "%content-control-attrs;">
      <!ENTITY % priorityClass.content "(%timecontainer.content;)*">
```

Observe que as entidades deste módulo estão com suas seções condicionais declaradas como INCLUDE, isto é, estão incluídas.

4.8 Expandindo a linguagem SMIL

Segundo o (W3C, 2002) a linguagem SMIL pode ser expandida através de outra recomendação W3C ou por extensões privadas, desde que sejam observadas algumas regras:

a) todos os elementos introduzidos em extensões devem aceitar o atributo "skip-content";

b) As expansões privadas têm de ser introduzidas através da sintaxe da especificação de namespace XML.

Abaixo, alguns exemplos de declaração namespace.

- Para um documento SMIL 1.0:

```
< smil xmlns="http://www.w3.org/TR/REC-smil " >
   ...
< / smil >
```

- Para um documento SMIL 2.0:

```
< smilxmlns="http://www.w3.org/2001/SMIL20/Language">
   ...
< / smil >
```

- Para um documento SMIL 1.0 que fosse estendido para usar o elemento excl:

```
< smil xml="http://www.w3.org/TR/REC-smil "
        xmlns:smil20="http://www.w3.org/2001/SMIL20/" >
   <smil20:excl >
      ....
   </smil20:excl >
< /smil >
```

- Um documento SMIL 2.0 que fosse estendido para usar o elemento fictício *apto* de uma versão fictícia de SMIL 3.0:

```
< smil xmlns="http://www.w3.org/2001/SMIL20/Language"
 xmlns:smil30="http://fw.uri.br/~elisa/2002/SMIL30/linguagem" >
   <smil30:apto>
     ...
   </smil30:apto>
</smil>
```

- Uma construção de uma extensão pode ser representada por:

```
< smil xmlns="http://www.w3.org/2001/SMIL20/Language"
   xmlns:a="http://fw.uri.br/~elisa/minhaextensao">
     ....
   <par a:novo="um valor legal">
     ...
   </par>
</smil>
```

4.8.1 Atributo skip-content

No módulo controle de conteúdo da linguagem Smil 2.0, existe o atributo skip-content, o qual é usado para controlar um elemento.

O atributo *SkipContent* controla se o conteúdo de um elemento é avaliado ou deve ser ignorado. Este atributo pode ser: true | false.

O atributo *SkipContent* foi introduzido para futuras expansões da linguagem SMIL. E é interpretado nos dois casos que seguem:

a) se um novo elemento for introduzido em uma futura versão SMIL, permitindo tal elemento a utilização de elementos SMIL 2.0 como conteúdo de elementos, o

atributo "skip-content" controla o processamento (ou não) desse conteúdo por parte dos apresentadores de SMIL;

b) se um elemento vazio em uma versão SMIL deixar de ser vazio em uma futura versão SMIL, o atributo "skip-content" controla se o conteúdo é ignorado ou não pelo apresentador; ou quando é que isso resulta num erro de sintaxe.

Se o valor do atributo "skip-content" for "true" (verdadeiro) e estivermos perante um dos casos indicados acima, o conteúdo do elemento é ignorado. Se o valor for "false", o conteúdo do elemento é processado.O valor predefinido para "skip-content" é "true". É responsabilidade do perfil da linguagem especificar que elementos têm atributos *skip-content* para habilitar este mecanismo de expansão.

4.8.2 O Perfil da Linguagem SMIL

A linguagem SMIL é definida como uma linguagem de marcação. A sintaxe é definida formalmente com um DTD ou um schema XML, e que esteja baseada nos módulos SMIL. As exigências para o perfil da linguagem SMIL são:

a) assegurar que o perfil esteja compatível com as versões anteriores;

b) assegurar que a semântica de todos os módulos mantenha a compatibilidade com a semântica da linguagem SMIL;

c) adotar novas recomendações da W3C quando apropriado sem se opor a outras exigências.

4.9 Conformidade de um Documento SMIL

Um documento SMIL está em conformidade com a linguagem se aderir às especificações descritas no documento *Synchronized Multimedia Integration Language*

incluindo o *Document Type Definition.* Se um documento SMIL é um documento da versão 2.0, este documento deverá estar em conformidade com as especificações descritas no documento *Synchronized Multimedia Integration Language (SMIL) 2.0* incluindo o SMIL 2.0 DTD.

Os itens abaixo relacionados são critérios que um documento SMIL deve seguir.

a) O elemento raiz do documento deve ser o elemento "smil".

b) O documento deve ser XML bem formado.

c) Deve estar em conformidade com as seguintes recomendações do W3C:
   - especificação XML 1.0,
   - namespace XML,
   - todo o uso de CSS Styles e propriedades CSS em conformidade com Cascading Style Sheets e CSS2 Specification.

d) Um documento SMIL 2.0 pode conter a seguinte declaração:

```
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN"
        "http://www.w3.org/2001/SMIL20/WD/SMIL20.dtd">
```

   - se um documento contiver esta declaração, deve ser um documento XML válido. Isto implica a não permissão de extensões à sintaxe definida no DTD. Se o documento for inválido, um apresentador SMIL 2.0 deveria emitir um erro.

e) Um documento tem que declarar um namespace para seus elementos, com um atributo xmlns no elemento raiz <smil>, com seu identificador URI.

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  ...
</smil>
```

f) Um documento pode declarar um XML DOCTYPE e uma ou mais declarações XML namespace. Para ser reconhecido como um documento SMIL 2.0 em conformidade com SMIL 2.0, o documento deve incluir identificador namespace SMIL 2.0 como padrão namespace na tag <smil>. Por exemplo, declarar um documento SMIL 2.0 com as extensões feitas em conformidade a um DTD.

```
<!DOCTYPE smil SYSTEM
"http://www.fw.uri.br/~elisa/minhaextensaoSMIL.dtd">
<smil xmlns="http://www.w3.org/2001/SMIL20/Language"
 xmlns:minhatag=" http://fw.uri.br/~elisa/2002/SMIL30/Linguagem">
    <minhatag:novo>
    ...
    </minhatag:novo>
</smil>
```

g) Um documento que não declara nem um DOCTYPE nem uma declaração de namespace será processado como um documento de SMIL 1.0. Os elementos ou os atributos da extensão não definidos pelo namespace de SMIL 1.0 devem ser declarados usando o mecanismo do namespace de XML.

h) É dado que DTDs não têm nenhuma maneira de descrever a aceitabilidade de extensões de namespace qualificados, e que as extensões, em conformidade com documentos SMIL 2.0, devem ser namespace qualificados.

i) Muitos atributos de SMIL 2.0 são associados com diversos namespaces de SMIL 2.0. Os atributos dos namespaces de SMIL 2.0 que têm a mesma parte local são considerados o mesmo atributo. É ilegal usar um atributo de SMIL 2.0 diversas vezes em um elemento, mesmo se cada ocorrência é qualificada por um namespace diferente de SMIL 2.0, os agentes do usuário de SMIL tratarão este como um erro da sintaxe. No exemplo abaixo, o fragmento do documento resulta em um erro, pois o atributo *begin* aparece duas vezes no elemento *ref*.

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language"
xmlns:BasicInlineTiming="http://www.w3.org/2001/SMIL20/BasicInlineTiming"
>
```

```
...
   <ref begin="5s" BasicInlineTiming:begin="5s"/>
...
</smil>
```

A linguagem SMIL 2.0 ou os critérios de conformidade não estabelecem nenhum limite no número de elementos, na quantidade de dados, de caráter, ou do número de caracteres em valores do atributo.

O perfil da linguagem SMIL 2.0 suporta a recomendação XML Base. A XML Base é suportada em todos os elementos, e afeta a interpretação de URIs como especificada nos módulos individuais que definem os atributos de URI. Especificamente, todo o URI aplicável de XML Base deve ser aplicado à interpretação do atributo href dos elementos de ligação *a* , *área* e *anchor*, assim como o atributo *src* dos elementos de mídia *audio, video, img, animation, textstream, text,* and *ref.* XML Base deve também ser aplicada em atributos *longdesc* de todos os elementos de linguagem de SMIL 2.0.

4.10 Apresentador SMIL

Um apresentador SMIL é um programa que analisa gramaticalmente e processa um documento SMIL. Ao estender a linguagem SMIL, é necessário que o apresentador SMIL reconheça os novos elementos e atributos. Se o apresentador é um programa que analisa um documento SMIL 2.0, o documento deverá estar em conformidade com a linguagem SMIL 2.0.

Para que o documento e o apresentador estejam em conformidade, alguns critérios devem ser observados.

a) Em regra, ser consistente com a recomendação de XML 1.0.

b) O apresentador deve analisar gramaticalmente e avaliar um documento SMIL 2.0 para bem-formatado. Se o apresentador declarar ser um agente de validação do usuário, deve também validar documentos de encontro a seu DTD.

c) Quando o apresentador declarar suportar a funcionalidade de elementos SMIL através de elementos e atributos e da semântica associada, deve fazer de maneira consistente com a especificação.

d) O apresentador deve poder analisar gramaticalmente e processar com sucesso todos os documentos em conformidade com SMIL, e suportar e executar corretamente a semântica de todas as características do perfil da linguagem.

e) O parser XML do apresentador deve poder analisar gramaticalmente e processar as construções XML definidas dentro de XML1.0 e XML-NS (XML-Namespace).

f) O apresentador tem que ignorar elementos e atributos não implementados. Atributos não implementados devem ser tratados como se eles não fossem especificados. Elementos não implementados devem aplicar o mecanismo skip-content. Se nenhum atributo skip-content é declarado, o valor verdadeiro (true) é assumido.

g) O agente apresentador tem que avaliar todo o documento SMIL corretamente, os namespace e URIs.

h) O namespace reconhecido pelo apresentador poderá ser um dos três tipos abaixo:

- o namespace padrão reconhecido em sua totalidade pelo apresentador (O apresentador deve processar o documento como uma versão reconhecida. Todos os elementos, atributos devem ter namespace qualificados usando os mecanismos padrões de XML para declarar namespaces para os elementos e os atributos descritos em XML-NS. O mecanismo *skip-content*, definido no módulo SkipContent, será aplicado a elementos de extensão não reconhecida pelo

apresentador. Os elementos inadequados não partem do namespace padrão, são ilegais e devem resultar em um erro);

- quando nenhuma declaração namespace está presente no documento, será processado como um documento SMIL 1.0;

- quando a declaração namespace não é reconhecida, o apresentador SMIL não reconhecerá o documento como uma versão SMIL e tem que emitir um erro.

i) O apresentador SMIL tem que suportar as recomendações do W3C no que diz respeito ao conteúdo SMIL:

- suporte completo para XML 1.0;

- suporte completo para inclusão de namespaces em uma versão que não seja SMIL 2.0 dentro de SMIL 2.0; uma declaração xmlns pode ser usada em qualquer elemento dentro da linguagem SMIL 2.0;

- o agente usuário tem que emitir um erro para um valor de atributo que não está em conformidade com a sintaxe especificada para aquele atributo e nunca tocar a mídia.

j) Um apresentador da linguagem SMIL 2.0 têm que tocar também os documentos de SMIL 1.0, como especificado na recomendação SMIL1.0.

4.11 Ferramentas de suporte para SMIL

Editores, servidores e players são um conjunto de elementos que se fazem necessários para desenvolver, aplicar e reproduzir documentos multimídia. Existem no mercado várias ferramentas para exibição. Porém, como a linguagem SMIL é uma

linguagem emergente, as ferramentas disponíveis não são abundantes e são limitados o número de players e browsers, que suportam a linguagem SMIL 2.0.

Para edição de um documento SMIL, qualquer editor de texto dá suporte, porém um editor específico tem a capacidade de destacar a sintaxe, facilitando a construção.

Como XML é uma linguagem "aberta", SMIL habilita uma gama extensiva de ambientes de autoria. Variando de editores de texto simples a ferramentas gráficas. Abaixo, alguns exemplos de editores:

- GRINS – Graphical iNterface to SMIL da Oratrix;

- RealSlideshow (RealNetworks) - RealSlideshow permite a um autor combinar imagens digitais com música, com texto narrado e assim criar slideshows para o Web;

- SMIL Editor V2.0 – lançado pela DoCoMo.;

- Outros, como SMIL Composer SuperTool da Sausage Software e Alliare's SMIL Tag Pack da HomeSite e Cold Fusion Studio.

Os players são os apresentadores de mídia. Os apresentadores SMIL mais conhecidos são os seguintes:

- GRiNS da Oratrix Development – a Oratrix criou um player chamado GRiNS player for SMIL 1.0, porém já possuem um player que suporta a versão SMIL 2.0, o GRiNS player for SMIL 2.0;

- Apple – o player QuickTime de Apple, versão 4.1 e 5, ambos suportam SMIL 1.0;

- RealNetworks - a versão RealPlayer 8 implementa a versão SMIL1.0. O player RealOne Platform, um lançamento da Real, não é um simples upgrade da versão

RealPlayer 8 mas uma união de tecnologia oferecida por grande canais de televisão como CNN, FOX, Sports, E! Entertainment, Cnet, Weather Channel e NBA Channel. RealOne tem por finalidade tocar streaming de vídeo. Utiliza a tecnologia Turbo Play, para tentar diminuir os bufferings. Esta versão dá suporte ao perfil da linguagem SMIL 2.0;

- Microsoft - o Windows Mídia Player da Microsoft não suporta nem à versão SMIL 1.0.

- Microsoft - o browser, Internet Explorer 5.5, não suporta todos os módulos de SMIL 2.0. Internet Explorer 5.5 suporta os módulos de temporização e sincronização, animação, controle de conteúdo. O módulo Layout não é suportado, porém este problema pode ser contornado, usando HTML;

- outros - existem outros players ou browsers desenvolvidos para suportar SMIL. Por exemplo: HPAS da Compaq, Lp Player da Productivity Works, o aplicativo SOJA (SMIL Output in a Java Applet) desenvolvido pela organização francesa Helio (HELIO, 1999), S2M2 da NIST, Schmunzel produzido pela SunTREC Salzburg e X-SMILES implementado por TML laboratory. SOJA é um player SMIL basedo em java, S2M2 é um player SMIL baseado em Applet Java, Shmuzel é um player SMIL baseado em java e X-SMILES é um browser baseado em Java. Porém, muitos ainda apresentam problemas, outros possuem suporte parcial para a linguagem.

Exemplo de Players para SMIL 1.0:

- Realplayer 8 – pela RealNetworks;
- Quick Time 4.1 – pela Apple;
- Soja – pela Hélio ;
- X-SMILES – pelo laboratório TML ;
- S2M2 – SMIL Player baseado por NIST ;
- Grins – pela Oratrix;
- HPAS – pela Compact;

- Lp Player – pela Productivity WorksSchmunzel .

Exemplo de Players para SMIL 2.0:

- RealOne Platform – pela Real Networks;
- Grins para SMIL 2.0 – pela Oratrix;
- X-Smiles 0.5 – Provêm algum suporte para SMIL 2.0.

Exemplo de browsers para SMIL:

- Internet Explorer 5.5 – pela Microsoft;
- Browser Internet Explorer 6.0 – pela Microsoft incluindo implementação do HTML + SMIL Profile.

4.12 Conclusão

SMIL é uma linguagem declarativa para descrever apresentações multimídia na Web, projetada de maneira a facilitar a autoria. A idéia básica é nomear componentes de mídia com URLs e programar a apresentação deles dentro de uma seqüência e/ou em paralelo. A funcionalidade de SMIL 2.0 é particionada em módulos.

A linguagem SMIL é definida como uma linguagem de marcação. A sintaxe é definida formalmente com um DTD ou um schema XML, e que esteja baseada nos módulos SMIL.

Por descender de XML, SMIL é uma linguagem que pode ser expandida, através de outra recomendação W3C ou por extensões privadas, desde que sejam observadas algumas normas.

O próximo capítulo é um estudo sobre namespace. Namespace está diretamente ligado à extensão da linguagem SMIL, pois define a possibilidade de criar novos elementos e atributos e rotular com clareza quem é o responsável pela extensão, evitando conflitos de nomes e permitindo a reutilização de estruturas.

5 Namespaces

A linguagem XML é uma linguagem extensível, como o próprio nome diz: eXtensible Markup Language. Porém, em ambientes colaborativos como a Web, esta capacidade de extensão da linguagem deve ser gerenciada, para evitar conflitos. A proposta é usar namespace como mecanismo para identificar os elementos da XML.

O uso de namespace introduz a reusabilidade, isto é, os elementos podem ser reutilizados em vários documentos. Se um elemento de marcação existe e é reconhecido pelos softwares disponíveis, é melhor reusar este elemento em vez de criar tudo de novo.

Segundo W3C (1999b), um namespace XML é uma coleção de nomes, identificados por uma referência, isto é, um URI. Namespace provê um método fácil de qualificar elementos e atributos usados em documentos de linguagem de marcação extensível.

Nomes de namespace XML devem conter nomes qualificados. O nome do namespace é o URI e não o prefixo.Uma aplicação utiliza o URI e não o prefixo para comparar elementos (MARCHAL, 2000).

Um namespace é declarado usando uma família de atributos reservados. A sintaxe definida pelo W3C requer um atributo xmlns e um sinal de dois pontos, seguido por uma prefixo (abreviação/apelido) e uma parte local, a URI. A URI identifica uma descrição da sintaxe e da semântica dos elementos do namespace.

O exemplo abaixo ilustra a construção de um namespace:

```
< xmlns:prefixo="parte local (URI)">
```

A declaração associa um URI a um prefixo, como os URIs são exclusivos, este é um mecanismo para identificar sem ambigüidade quem desenvolveu determinado elemento.

O prefixo somente serve para conduzir a um nome de namespace. Aplicações deveriam usar o nome do namespace, não o prefixo, criando nomes cujo âmbito estende além do conteúdo do documento. O namespace é válido para o elemento onde está declarado e em todos os elementos dentro de seu conteúdo. Se um elemento não tem nenhum prefixo de namespace, é considerado um namespace padrão. Ele é aplicado para todos os elementos onde é declarado e para todos os elementos sem prefixo dentro do conteúdo daquele elemento. Exemplo, fig. 5.1.

```
<?xml version=1.0?>
<html xmlns="http://www.w3.org/TR/REC-thml140"
.....
</html>
```

Figura 5-1 - Namespace

Os exemplos, figs. 5.2 e 5.3, mostram uma declaração namespace, a qual está associada a um prefixo art, que aponta para um URI http://www.fw.uri.br/~elisa/artigos

```
<material xmlns:art='http://www.fw.uri.br/~elisa/artigos>
  <!-- O prefixo "art" está apontando para 'http://www.fw.uri.br/~elisa/artigos
       para o elemento <material> e seu conteúdo -->
</material>
```

Figura 5-2 - Namespace e URI

```
<material xmlns:art='http://www.fw.uri.br/~elisa/artigos>
  <!--o elemento "título" é um namespace para http://www.fw.uri.br/~elisa/artigos -- >
  <art:titulo>SMIL Boston</art:titulo
</material>
```

Figura 5-3 - Exclusividade de um URI

O URI é usado para garantir a exclusividade dos nomes. Portanto, é importante que os URIs sejam exclusivos. Eles são endereços e apontam para um arquivo em uma máquina. Dois URIs são considerados idênticos se forem idênticos em cada caractere. Por exemplo, o URI http://www.uri.br é diferente de http://www.URI.br, embora aponte para o mesmo documento. Atualmente a maioria dos URIs são URLs, a solução é criar URLs com base em seu próprio domínio.

5.1 Restrições Namespaces

Ao criar um namespace, deve-se obedecer a certas normas que conformam as especificações da W3C. Um documento está conforme a esta especificação se todos os sinais, que são obrigatórios, estiverem compatíveis com a produção de nomes de prefixos e nomes de partes locais. Prefixos que começam com a seqüência das letras x, m, l, são reservados; portanto, não devem ser usados.

Segundo a especificação namespace (W3C, 1999b), nenhum tag pode conter dois atributos que:

- tenham nomes idênticos;
- nomes com a mesma parte local.

O exemplo abaixo, fig. 5.4 ilustra um modelo que apresenta atributos com nomes idênticos e os prefixos n1 e n2 com a mesma parte local.

```
<?xml version="1.0?>
<!-- http://www.w3.org está apontando para n1 e n2 -->
<x xmlns:n1="http://www.w3.org"
  xmlns:n2="http://www.w3.org" >
 <ruim a="1"    a="2" />
 <ruim n1:a="1"  n2:a="2" />
</x>
```

Figura 5-4 - Referência a atributos iguais - I

Ao contrário do exemplo acima, a fig. 5.5 mostra a forma correta de fazer esta referência:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE x SYSTEM "namespace2.dtd">
<!-- http://www.w3.org está apontando somente para n1 -->
<x xmlns:n1="http://www.w3.org"
  xmlns="http://www.w3.org" >
 <bom a="1"    b="2" />
 <bom a="1"    n1:a="2" />
</x>
```

Figura 5-5 - Referência a atributos iguais - II

Podem ser declarados vários prefixos de namespace como atributos de um único elemento, como mostra a fig. 5.6.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE art:material SYSTEM "namespace1.dtd">
<art:material xmlns:art="http://www.fw.uri.br/~elisa/artigos"
   xmlns:val="http://www.fw.uri.br/~elisa/valor">
 <art:titulo>SMIL Boston</art:titulo>
 <val:preço>35,00</val:preço>
</art:material>
```

Figura 5-6 - Prefixos namespace

Um documento XML é válido se tiver uma DTD. Nomes de elementos e tipos de atributos também são determinados como nomes válidos quando eles aparecem declarados na DTD (MARCHAL, 2000).

O exemplo da fig. 5.7 é o DTD gerado a partir do arquivo XML da fig. 5.5. O exemplo da fig. 5.8 é o DTD gerado a partir do arquivo XML do exemplo da fig. 5.6.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--DTD generated by XML Spy v3.5 (http://www.xmlspy.com)-->
<!ELEMENT bom EMPTY>
<!ATTLIST bom
  a CDATA #REQUIRED
  b CDATA #IMPLIED
  n1:a CDATA #IMPLIED
>
<!ELEMENT x (bom+)>
<!ATTLIST x
  xmlns:n1 CDATA #REQUIRED
  xmlns CDATA #REQUIRED
>
```

Figura 5-7 - Exemplo DTD - Referência a atributos iguais - II

```
<?xml version="1.0" encoding="UTF-8"?>
<!--DTD generated by XML Spy v3.5 (http://www.xmlspy.com)-->
<!ELEMENT art:material (art:titulo, val:preço)>
<!ATTLIST art:material
  xmlns:art CDATA #REQUIRED
  xmlns:val CDATA #REQUIRED
>
<!ELEMENT art:titulo (#PCDATA)>
<!ELEMENT val:preço (#PCDATA)>
```

Figura 5-8 - Exemplo DTD – Prefixos namespace

5.2 Expandindo XML através de Namespace

Namespaces são usados para estender a linguagem XML. Através dele, é possível desenvolver elementos que podem ser usados em vários documentos. Este é o princípio de reutilização. É nesse sentido que cada vez mais grupos trabalham para alcançar o objetivo da reusabilidade.

Primeiramente os documentos XML foram desenvolvidos para aplicações específicas.Os elementos definidos na DTD eram específicos para aquela aplicação. Poucos recursos reutilizáveis, no máximo, copiar e colar.

Como exemplo de reutilização, temos as folhas de estilo XML e a recomendação Xlink. As folhas de estilo combinam elementos da própria linguagem no namespace http://www.w3.or/XSL/transform/1.0 e elementos da HTML no namespace http://www.w3.org/TR/REC-htm140. Veja o fragmento de uma folha de estilo na fig. 5.9.

```
<?xml version="1.0"encoding=ISO-8859-1">
<xsl:stylesheet
  xmlns:xsl=http://www.w3.or/XSL/transform/1.0
  xmlns:=http://www.w3.org/TR/REC-htm140>
        .....
</xsl:stylesheet>
```

Figura 5-9 - Namespace folha de estilos

Em um documento XML que faz uso de elementos Xlink, a recomendação usa namespace para diferenciar os elementos Xlink do restante do documento, conforme ilustrado na fig. 5.10.

```
<?xml version="1.0" encoding="UTF-8"?>
<info>
  <para> Para obter mais informações sobre a recomendação Xlink
  <xlink:simple xmlns:xlink="http://www.w3.org/XML/Xlink/0.9"
    href="http://www.w3.org/TR/xLink"
  </xlink:simple>
  </para>
</info>
```

Figura 5-10 - Namespace Xlink

5.3 Nomes de namespaces qualificados em SMIL

As entidades de parâmetros requerem nomes qualificados para elementos e atributos da linguagem SMIL, através de namespaces. Esta seção declara entidades de parâmetro para suportar namespaces qualificados, declaração namespace e prefixos

namespace para extensões SMIL. Também mostra a relação de entidades de parâmetros com namespace qualificados para todos os elementos SMIL.

Segundo W3C (2002), a estrutura de um namespace SMIL deve observar várias normas.

a) A declaração de duas entidades de parâmetros. A declaração da entidade de parâmetro com o URI para que identifique o namespace (exemplo1) e a declaração do atributo namespace (exemplo2).

Exemplo1:

```
<!ENTITY % XLINK.xmlns "http://www.w3.org/1999/xlink" >
```

Exemplo2:

```
<!ENTITY % XLINK.xmlns.attrib
   "xmlns:xlink  %URI.datatype;         #FIXED '%XLINK.xmlns;'"
>
```

b) A declaração da entidade de parâmetro contendo o URI para SMIL e qualquer namespace incluído por SMIL. Por exemplo:

```
<!ENTITY % SMIL.xmlns http://www.w3.org/2001/SMIL20/PR/" >
```

c) A declaração de entidades de parâmetro que contêm o prefixo padrão, para serem usados quando o prefixo for habilitado. Isto pode ser ignorado pelo DTD administrador ou no subconjunto interno de uma instância do documento. Exemplo:

```
<!ENTITY % SMIL.prefix  "" >
```

- o namespace *prefix* serve como uma procuração para a referência de URI.

d) A declaração *%SMIL.prefixed* é uma chave de seção condicional, usada para ativar o prefixo. O valor padrão é herdado (%NS.prefixed;) do DTD administrador, de forma

que, se não for ignorado, o comportamento padrão segue o mesmo esquema do DTD administrador.

Exemplo:

```
<!ENTITY % NS.prefixed "IGNORE" >
<!ENTITY % SMIL.prefixed "%NS.prefixed;" >
```

e) A declaração de entidade de parâmetro (%SMIL.pfx;) contendo o prefixo (%SMIL.prefix;:) quando o prefixo está ativo. Um valor vazio, representado por "", quando não está ativo. Exemplo:

```
<![%SMIL.prefixed;[
<!ENTITY % SMIL.pfx  "%SMIL.prefix;:" >
]]>
<!ENTITY % SMIL.pfx  "" >
<!—declare as extensões de nome qualificadas aqui  -->
<!ENTITY % smil-qname-extra.mod "" >
%smil-qname-extra.mod;
```

f) A entidade de parâmetro *%SMIL.xmlns.extra.attrib*; pode ser redeclarada para conter qualquer declaração de atributos namespace não-SMIL para namespaces embutidos em SMIL. O padrão é um valor vazio, representado no exemplo por: "". XLink deveria ser incluído aqui se usado no DTD e não incluído numa declaração anterior. Exemplo:

```
<!ENTITY % SMIL.xmlns.extra.attrib "" >
```

g) A entidade de parâmetro *%NS.prefixed.attrib*; é definida para ser o prefixo para elementos SMIL se for *SMIL.xmlns.extra.attrib*. Exemplo:

```
<![%SMIL.prefixed;[
<!ENTITY % NS.prefixed.attrib
 "xmlns:%SMIL.prefix;  %URI.datatype;  #FIXED'%SMIL.xmlns;'
  %SMIL.xmlns.extra.attrib; " >
]]>
```

```
<!ENTITY % NS.prefixed.attrib "%SMIL.xmlns.extra.attrib;" >
```

O DTD abaixo mostra a declaração de entidades de parâmetro com namespace qualificados para todos os elementos SMIL.

```
<!ENTITY % animate.qname "%SMIL.pfx;animate" >
<!ENTITY % set.qname "%SMIL.pfx;set" >
<!ENTITY % animateMotion.qname "%SMIL.pfx;animateMotion" >
<!ENTITY % animateColor.qname "%SMIL.pfx;animateColor" >

<!ENTITY % switch.qname "%SMIL.pfx;switch" >
<!ENTITY % customTest.qname "%SMIL.pfx;customTest" >
<!ENTITY % customAttributes.qname "%SMIL.pfx;customAttributes" >
<!ENTITY % prefetch.qname "%SMIL.pfx;prefetch" >

<!ENTITY % layout.qname "%SMIL.pfx;layout" >
<!ENTITY % region.qname "%SMIL.pfx;region" >
<!ENTITY % root-layout.qname "%SMIL.pfx;root-layout" >
<!ENTITY % topLayout.qname "%SMIL.pfx;topLayout" >
<!ENTITY % regPoint.qname "%SMIL.pfx;regPoint" >

<!ENTITY % a.qname "%SMIL.pfx;a" >
<!ENTITY % area.qname "%SMIL.pfx;area" >
<!ENTITY % anchor.qname "%SMIL.pfx;anchor" >

<!ENTITY % ref.qname "%SMIL.pfx;ref" >
<!ENTITY % audio.qname "%SMIL.pfx;audio" >
<!ENTITY % img.qname "%SMIL.pfx;img" >
<!ENTITY % video.qname "%SMIL.pfx;video" >
<!ENTITY % text.qname "%SMIL.pfx;text" >
<!ENTITY % textstream.qname "%SMIL.pfx;textstream" >
<!ENTITY % animation.qname "%SMIL.pfx;animation" >
<!ENTITY % param.qname "%SMIL.pfx;param" >
<!ENTITY % brush.qname "%SMIL.pfx;brush" >

<!ENTITY % meta.qname "%SMIL.pfx;meta" >
<!ENTITY % metadata.qname "%SMIL.pfx;metadata" >

<!ENTITY % smil.qname "%SMIL.pfx;smil" >
<!ENTITY % head.qname "%SMIL.pfx;head" >
<!ENTITY % body.qname "%SMIL.pfx;body" >

<!ENTITY % seq.qname "%SMIL.pfx;seq" >
<!ENTITY % par.qname "%SMIL.pfx;par" >
<!ENTITY % excl.qname "%SMIL.pfx;excl" >
```

```
<!ENTITY % transition.qname "%SMIL.pfx;transition" >
<!ENTITY % transitionFilter.qname "%SMIL.pfx;transitionFilter" >
```

5.4 Conclusão

Namespaces são usados para estender a linguagem XML, impedindo que conflitos se estabeleçam. Este capítulo apresentou como os namespaces complementam a facilidade de extensão da XML, como devem ser declarados os prefixos namespaces para possíveis extensões e como usar namespace em DTDs.

A proposta é usar namespace como mecanismo para identificar os elementos da XML. Um namespace XML é uma coleção de nomes, identificados por uma referência, um URI.

O uso de namespace introduz a reusabilidade, isto é, os elementos podem ser reutilizados em vários documentos. Se um elemento de marcação existe e é reconhecido pelos softwares disponíveis, é melhor reusar este elemento em vez de criar tudo de novo.

A aplicabilidade dos namespaces será introduzida no capítulo 7, o qual refere-se à extensão da linguagem SMIL. O próximo capítulo descreve os principais requisitos que um modelo multimídia deveria atender para ser um modelo adequado à criação de cenários multimídia sobre a Internet.

6 Requisitos de um Modelo Multimídia para a Web

As ferramentas de autoria de documentos multimídia são baseados em um modelo multimídia que induz uma técnica de descrição de documentos multimídia. Grande parte das ferramentas de autoria de documentos multimídia permitem gerar cenários multimídia interativos que podem ser disponibilizados via Web. O sistema de comunicação da Web, a Internet, não garante certos requisitos importantes para garantir o sincronismo e a qualidade das apresentações multimídia. Isto, pois a rede Internet é do tipo melhor esforço, não garantindo taxa de bits, atraso e variação de atraso. Portanto, a Internet gera um indeterminismo temporal, que deveria ser tratado pelos cenários multimídia. Para isto, as ferramentas de autoria deveriam permitir ao autor tratar este problema.

Este capítulo levanta os principais requisitos que um modelo multimídia deveria atender, para ser um modelo adequado à criação de cenários multimídia sobre a Internet. Estes requisitos são exigências importantes que toda linguagem voltada ao projeto de documentos multimídia, considerando em redes temporalmente não determinísticas, deveria satisfazer. Além disso, são levantados os principais requisitos básicos de um modelo de autoria.

6.1 Estrutura de um Documento

A estruturação de um documento em uma linguagem de autoria é um componente Fundamental. Os documentos devem ser especificados de forma estruturada, e esta estrutura deve fazer uso de módulos independentes, de forma que estes módulos possam ser agrupados.

Existem algumas abordagens para especificar a estrutura ideal de um documento multimídia. Segundo WILLRICH (2000) um modelo ideal deveria permitir as seguintes estruturas:

a) **estrutura conceptual** - descreve a estrutura lógica das diferentes partes de um documento e suas relações lógicas, e em que instante os componentes serão apresentados;

b) **estrutura de apresentação** - descreve como e onde os componentes serão apresentados;

c) **estrutura de conteúdo** - descreve as informações que constituem os componentes.

A divisão de um documento em três estruturas, conforme citadas acima, oferecem algumas vantagens, tais como (WILRICH, 2000):

a) se a estrutura lógica for separada da estrutura de apresentação, esta permite reutilização da estrutura lógica quando da apresentação em dispositivos diferentes (FLUCKIGER, 1995);

b) a separação entre a estrutura de apresentação e do conteúdo permite que os documentos utilizem os mesmos dados em diferentes documentos ou diferentes contextos (SCHLOSS, 1994).

As próximas seções apresentarão os tipos de estrutura que permitem esta descrição. Além disso, elas identificam os requisitos de um modelo de autoria para cada nível da estrutura.

6.1.1 Estrutura de conteúdo

A fase inicial de um documento multimídia se refere à obtenção dos materiais (imagens, textos, sons e outros) que irão compor o documento. Estes materiais podem ser definidos como um conjunto de dados primitivos. Por sua vez, a estrutura de conteúdo é responsável pela descrição dos dados primitivos, ou seja, descreve as informações que constituem os componentes.

"Neste nível, um modelo multimídia deve permitir a especificação das informações de acesso e de manipulação dos dados primitivos e os valores originais das características espaciais, sonoras e temporais de apresentação" (WILLRICH, 2000).

6.1.2 Estrutura conceptual

Esta estrutura descreve a estrutura lógica, suas relações lógicas e a composição temporal dos componentes, isto é, especifica em que instantes os diferentes componentes serão apresentados.

Esta estrutura, por conter a definição dos componentes semânticos, permite a divisão de um documento em várias partes. Um exemplo claro de um documento dividido em partes é um livro, onde temos, capítulos, seções, subseções, etc. Porém, quanto maior o documento, mais complexa se torna a tarefa de especificação. Esta estrutura, no entanto, é utilizada para construção de apresentações complexas, porém requer a construção a partir de pequenos grupos ou seções, independentes entre si. A idéia de que, estas seções possam ser reutilizadas e agrupadas, visto que cada seção pode ser criada independentemente de outras, introduzindo, assim, o benefício de modularidade e encapsulação.

KLAS (1990) define que, os componentes do documento devem ser realizados independentemente das informações acerca da representação de seu conteúdo. No que se refere aos componentes da estrutura conceptual, estes devem fazer menção somente às informações que estes componentes representam. Esta menção deve ser independente do tipo de informação e da localização geográfica, isto é, se é local ou remota.

Desta forma, o armazenamento e a transmissão do conteúdo ou da estrutura de um documento podem ser feitos implicitamente ou explicitamente. No caso de uma inclusão implícita, a estrutura do documento contém os dados primitivos. Desta forma, a estrutura e o conteúdo são armazenados no mesmo arquivo e assim transferidos. Na inclusão implícita, a estrutura do documento faz referência aos dados primitivos ou a outras partes da estrutura. Neste caso, somente uma parte do documento é transferida.

Quando o documento for apresentado, o sistema cliente acessa remotamente as partes incluídas implicitamente no documento.

Os caminhos de percurso de um documento são definidos também na estrutura conceptual. A estrutura de uma informação pode ser linear, hierárquica e em rede. Conforme ilustra a fig. 6.1.

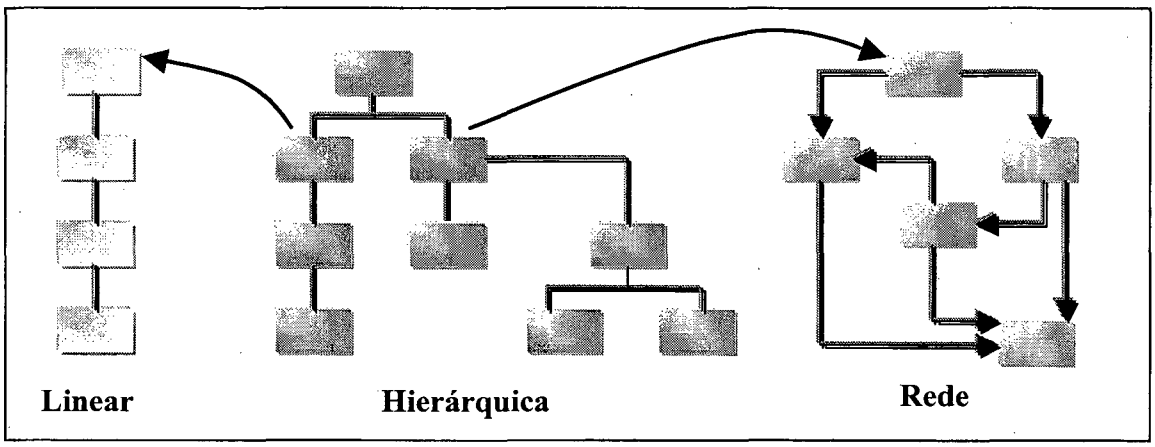


Figura 6-1 - Estruturas de Navegação

As estruturas lineares são utilizadas nos documentos do tipo "visita guiada", de maneira seqüencial. As hierárquicas, por sua vez, podem ser comparadas com um livro, isto é, organizados em capítulos e seções. No caso das estruturas em rede, estas contêm ligações associativas, não sendo seqüências. Os links são de natureza semântica e pragmática. Um exemplo é a organização das informações na forma de enciclopédia. O modelo ideal de um documento multimídia deveria suportar os tipos de estrutura de navegação definidos acima.

A estrutura conceptual também define o modelo temporal e condicional de uma apresentação. Esta estrutura deve prever os eventos que poderão ocorrer. Existem dois tipos de eventos. Eventos determinados previamente, isto é, eventos *síncronos* e os eventos *assíncronos*, os quais não podem ser determinados previamente; são eventos imprevisíveis. Nos eventos síncronos, a posição dos eventos pode ser determinada

somente sob condições ideais, ou seja, não considera imprevistos, como, por exemplo, sobrecarga na rede.

As relações temporais definem sincronizações que podem ser entre eventos de uma mesma apresentação, os quais são chamados de *sincronização intramídia.* As relações temporais entre eventos de diferentes mídias, ou intervalos definidos entre diferentes apresentações são chamados de *sincronização intermídia.*

De acordo com WILLRICH (2000) as sincronizações intramídia são dependências temporais naturais que são definidas implicitamente quando da produção dos dados primitivos (na estrutura do conteúdo). As sincronizações intermídia são dependências temporais especificadas pelo autor da estrutura conceitual de um documento. Assim, as sincronizações intermídia são descritas pela estrutura conceitual do documento, e as sincronizações intramídia fazem parte da estrutura de apresentação.

As relações temporais de um documento podem ser expressas de forma estática ou de forma dinâmica. A estática consiste em organizar explicitamente as relações temporais entre os componentes. Na forma dinâmica, são definidas as relações temporais denominadas relações ao vivo. A estrutura conceptual define apenas os aspectos da sincronização estática da apresentação multimídia, visto que, as relações ao vivo não podem ser especificadas em avanço, no momento da criação do documento (FLUCKIGER, 1995).

WAHL (1994) descreve as relações temporais de um documento, utilizando dois modelos:

- **modelos temporais baseados em pontos** - neste modelo a unidade temporal são os eventos; dois eventos podem definir três relações temporais, um evento pode ocorrer antes, em simultâneo e após outro evento;

- **modelos temporais baseados em intervalos** - neste modelo a unidade temporal são os intervalos.

Conforme WILLRICH (2000), modelos temporais baseados em intervalos são melhores que aqueles baseados em pontos. Pois, as apresentações podem ser vistas como intervalos temporais, com um início, fim e uma duração.

Um modelo hipermídia ideal deveria fornecer mecanismos para especificar uma relação condicional. Por exemplo, após o término da apresentação X, se o *link* Y está ativado, então apresentar Z.

O indeterminismo da duração das informações em sistemas multimídia distribuídos, causado pela rede de comunicação, atrasos de acesso à base de dados e outros, faz com que a relação temporal síncrona, em sistemas distribuídos, não pode ser garantida. Para contornar este problema, um modelo multimídia ideal deve permitir a especificação de métodos de tolerância de sincronização (WYNBLATT, 1995). Assim, o autor pode expressar quais compromissos de sincronização são aceitáveis e os meios de tratar as exceções quando da violação.

6.1.3 Estrutura de apresentação

Esta estrutura descreve as características espaciais, sonoras e temporais de cada componente que será apresentado. O autor deve especificar as características da apresentação dos componentes (por exemplo, descrições espaciais e sonoras) e sua composição espacial em um dado instante dado.

Nesta fase, um modelo multimídia deve permitir (WILLRICH, 2000):

a) especificar valores temporais de apresentação das informações dinâmicas, como a velocidade, posição de início e de fim de um vídeo e o número de repetições;

b) especificar valores espaciais de apresentação de informações visuais, por exemplo, tamanho, posição e estilo de apresentação;

c) especificar as apresentações sonoras, por exemplo, volume;

d) a especificação de dispositivos de saída, chamados de canais, na qual as informações serão apresentadas e vistas pelo leitor (por exemplo, uma janela, um canal de áudio);

e) apresentações alternativas a fim de repor uma apresentação principal se ela não puder ser apresentada em um certo sistema, devido a problemas de acesso, ou restrições temporais não satisfeitas. Permitindo a criação de documentos adaptáveis aos recursos disponíveis.

6.2 Interações

Em aplicações interativas, tais como multimídia interativa, hipertexto e hipermídia, o usuário deve dispor de mecanismos que permitam o controle da apresentação. HARDMAN (1995) descreve quatro métodos: navegação, controle de apresentação, controle do ambiente e interações da aplicação.

- **Navegação** - este método permite que o usuário possa selecionar um contexto entre vários. Ela é geralmente definida através da criação de um link, que liga as âncoras origens às âncoras destinos.

- **Controle da apresentação** - este método permite que o usuário interaja com a apresentação, podendo parar, recomeçar, avançar ou retroceder.

- **Controle do ambiente** - este método permite a particularização do ambiente de apresentação do documento. Por exemplo, o leitor pode desativar o canal de áudio ou ainda pode alterar o tamanho de uma janela.

- **Interações da aplicação** - nos métodos citados acima, o autor cria o documento e o usuário interage com ele. Existem aplicações que requerem mecanismos específicos, por exemplo, nas aplicações de tele-ensino, o modelo deve permitir a especificação da noção de acompanhamento dos alunos e de avaliação. Outro método de interação é a pesquisa por palavras-chave. Este tipo de mecanismo de interação é suportado por ferramentas especializadas.

Para WILLRICH (2000) "um modelo multimídia ideal deveria permitir a especificação de todos estes tipos de interação e fornecer uma abordagem uniforme de representação dos componentes, das relações lógicas e temporais, e dos mecanismos de interação".

6.3 Sincronização Multimídia

A sincronização multimídia se refere aos mecanismos usados para assegurar que dois ou mais eventos estejam coordenados no domínio do tempo e do espaço. A sincronização temporal representa o alinhamento de relacionamentos em relação ao domínio do tempo e a sincronização espacial se baseia em definir como combinar os elementos a serem apresentados, num dado instante de tempo. Para que uma aplicação multimídia seja satisfatória, é necessário que a apresentação dos componentes esteja sincronizada.

A problemática da sincronização multimídia está em ordenar a apresentação das mídias, a interação do usuário e os dispositivos físicos, a fim de satisfazer as relações temporais entre eles. Para isso existem alguns modelos, dos quais alguns são citados abaixo.

a) **Modelo baseado em linha temporal** – este modelo posiciona os componentes do documento hipermídia em relação ao eixo do tempo. Todos os relacionamentos de sincronização são vinculados de forma absoluta ao próprio eixo do tempo, ou seja, não possuem vinculação relativa a outros componentes. Exemplo: Flash e Director.

b) **Modelo baseado em eventos** – neste modelo, as ações são realizadas com base na ocorrência de eventos. A linguagem SMIL, por exemplo, emprega este modelo e MHEG-5.

c) **Modelo baseado em intervalos e modelo avançado baseado em intervalos** – Intervalo é a duração da apresentação de um componente multimídia. Neste modelo o tempo de apresentação de uma mídia está diretamente relacionado com a apresentação de outra. Devido à dificuldade de implementar este modelo, visto que é necessário o conhecimento prévio do tempo de apresentação de mídia, um novo modelo é criado. O modelo avançado baseado em intervalos, o qual possui algumas características que facilitam a manipulação.

d) **Modelo script** – A apresentação de um documento multimídia é realizada através de uma linguagem de especificação do seu comportamento. É ideal para apresentações curtas. Produções mais complexas tornam a apresentação e a manutenção trabalhosa. Exemplo: Toolbook.

Referente à sincronização, aplicações diferentes requerem requisitos de sincronização diferentes. Porém é importante reconhecer os requisitos de maneira a satisfazer a qualidade de serviço. Segundo FLUCKIGER (1995), as aplicações podem ser divididas em duas grandes classes:

- **aplicações pessoa-a-sistema** – pessoas comunicam-se com sistemas remotos - esta classe ainda pode ser subdividida em aplicações interativas (vídeo sob demanda) e de distribuição (difusão de áudio e vídeo);

- **aplicações pessoa-a-pessoa** – prover comunicação entre pessoas - esta classe ainda pode ser subdividida em aplicações em tempo real (teleconferência e videofonia) e aplicações assíncronas (e-mail multimídia).

Para uma comunicação multimídia síncrona, é necessário que sejam observados parâmetros de desempenho da rede. Abaixo segue relação de alguns destes parâmetros.

a) **Velocidade de acesso** – a velocidade de acesso refere-se à freqüência em que os bits podem ser enviados.

b) **Largura de banda** - a largura de banda é determinada pelo meio de transmissão utilizado. É a capacidade que a rede possui de transportar por unidade de tempo, medida em bits.

c) **Atraso** – mede o atraso fim-a-fim das aplicações conversacionais e o tempo de resposta nas aplicações baseadas em servidores. Atraso fim-a-fim é o tempo que leva para transmitir um bloco de dados de um transmissor a um receptor.

d) **Variação de atraso** - mede o desvio do tempo de apresentação, do tempo de apresentação original.

e) **Vazão** – é a taxa de bits efetiva, ou seja, a largura de banda efetiva. Assim, é a diferença entre a taxa de bits de ligação e os vários *overheads* (sobrecarga) associados à tecnologia de transmissão empregada.

f) **Taxa de erro tolerável** - a taxa de erro pode ser definida por erro de bits e erros de pacotes permitidos. A taxa de erro de bits é a razão entre o número médio de bits corrompidos e o número total de bits que são transmitidos. No caso de erro por pacotes, ocorre o mesmo que ocorre para bits, porém em pacotes. Um terceiro erro é no caso de redes ATM[9], é a taxa de erros em quadros, definidos como número de quadros (células) errados.

[9] Asynchronous Transfer Mode, desenvolvida inicialmente pela AT&T, 1980, com uma técnica rápida de comutação de pacotes para possibilitar mixar voz e transmissão digital sobre uma única rede digital. Em 1988, ATM foi adotada pelo ITU-T (International Telecommunication Union) como o mecanismo de multiplexação e comutação para B-ISBN (Broadband Integrated Service Digital Network).

g) **Distorção intermídia** - mede a diferença entre o tempo efetivo e o tempo ideal definido na relação temporal especificada.

Muitos são os mecanismos necessários para sincronização multimídia. Seguem abaixo alguns itens considerados importantes:

- redes por comutação de pacotes sofrem variações de atrasos, se não uniformizadas podem causar perdas de sincronização;
- taxas de consumo e de transmissão de dados são baseadas nas taxas dos relógios locais, portanto os relógios do transmissor e do receptor devem estar sincronizados;
- problemas de perdas de pacote - uma abordagem comum é anexar a cada pacote um número de seqüência de envio;
- apresentações advindas de múltiplas fontes distribuídas requerem que a transmissão seja coordenada;
- a estação de trabalho e a estação servidora devem fornecer requisitos de arquitetura de hardware para mídias contínuas.

O tempo de resposta e a interatividade são pontos essenciais. É interessante ter o conhecimento antecipadamente do tempo de resposta do sistema e que este seja satisfeito.

Uma solução para obter bons efeitos de apresentações é utilizar a técnica em que saltam-se ou repetem-se algumas unidades de dados de maneira que a qualidade de apresentação seja degradada suavemente. Para obter sincronização intermídia, o áudio é apresentado de forma mais cadenciada possível e o vídeo salta ou repete os quadros. Outra solução seria alterar o ritmo de transmissão do dado ou degradar a qualidade de serviço a fim de manter a sincronização.

6.4 Outros Requisitos Desejáveis

Além de um documento permitir a especificação de forma estruturada conforme proposto neste capítulo, outros itens não menos importantes, devem ser observados para uma linguagem de autoria (ANTONACCI, 2000).

a) Uma característica desejável é que um objeto possa ser especificado em separado de parte de seu conteúdo. Por exemplo, uma mídia pode não ter somente conteúdo específico dela, mas descrições como título, autor e outras definições.

b) Oferecer a possibilidade de reuso. Reusar é poder utilizar componentes de um documento no próprio documento ou em outros. A reusabilidade é um requisito fundamental, pois facilita o processo de autoria.

c) Permitir que os elos associados a um determinado componente fossem válidos no escopo de uma composição. Esta possibilidade juntamente com a possibilidade de reuso, permite que um mesmo componente tenha diferentes elos associados. Observe o exemplo nas figuras abaixo. Na figura 6.2A, o componente A possui três elos a ele associado através de links. Na figura 6.3B, o mesmo componente (A), possui somente um elo a ele associado, o elo E.

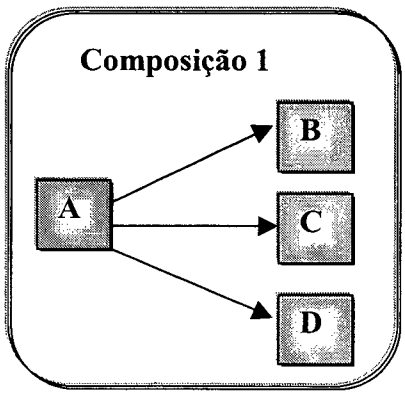


Figura 6-2 - Elo A associado aos elos B,C e D

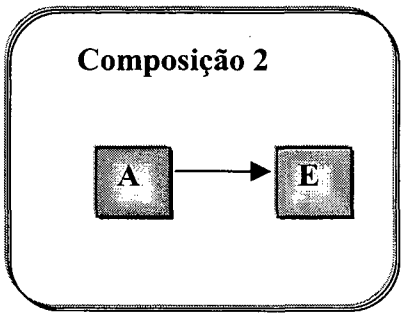


Figura 6-3B - Reuso do elo A

d) Permitir que uma relação ancore em pontos internos do objeto de mídia e não apenas no início ou no fim do objeto.

e) Oferecer a possibilidade de definir o relacionamento de dois ou mais componentes não apenas no modo tradicional de hipermídia, mas de forma que possua a semântica de causalidade ou restrições. Um exemplo de causalidade seria, se os componentes X e Y estão sendo exibidos e o componente Y termina sua exibição, exibir o componente Z. Para o caso de restrições entre os componentes, por exemplo, uma restrição que especifica que dois componentes X e Y devam iniciar juntos, não existe uma relação causal, visto que a apresentação dos componentes somente se verifica, se e somente se os dois iniciarem juntos.

f) Permitir a definição de elos entre componentes de composições diferentes e não limitar-se a elos de uma mesma composição, permitindo assim uma flexibilidade maior.

g) Possibilitar a especificação do comportamento de componentes de forma flexível. Esta flexibilidade pode permitir ajustes na exibição quando eventos imprevisíveis ocorrerem, tais como, atrasos na rede.

h) Adaptar de um documento a diferentes situações. Esta adaptação pode ser feita de acordo com alguns parâmetros que especificam informações, tais como idioma, qualidade de serviço, banda passante e outros.

6.5 Conclusão

As ferramentas de autoria de documentos multimídia permitem gerar cenários multimídia interativos que podem ser disponibilizados em redes de comunicação. No entanto, o sistema de comunicação via Web não garante certos requisitos. A Internet é do tipo melhor esforço, não garantindo taxa de bits, atraso e variação de atraso. Por isto, as ferramentas de autoria deveriam permitir ao autor tratar este problema.

Este capítulo levantou requisitos que um modelo multimídia deveria atender, para ser um modelo adequado à criação de cenários multimídia sobre a Internet.

Documentos multimídia devem ser especificados de forma estruturada, e esta estrutura, deve fazer uso de módulos independentes, de forma que estes módulos possam ser agrupados. A estruturação de um documento em uma linguagem de autoria é um componente Fundamental.

Para que uma aplicação multimídia seja satisfatória, é necessário que a apresentação dos componentes esteja sincronizada. A problemática da sincronização multimídia está em ordenar a apresentação das mídias, a interação do usuário e os dispositivos físicos, a fim de satisfazer as relações temporais entre eles.

Definidos os requisitos de um modelo multimídia e alguns mecanismos usados para assegurar a comunicação de forma sincronizada, o próximo capítulo descreverá uma análise da linguagem SMIL 2.0 em relação aos requisitos básicos para uma ferramenta de autoria multimídia na Web, de maneira que algumas deficiências desta linguagem sejam verificadas. Além disso, apresenta a extensão proposta para a linguagem SMIL, definindo passo a passo os procedimentos de extensão usando namespace XML.

7 Análise e Extensão da linguagem SMIL 2.0

SMIL 2.0 é uma extensão da linguagem SMIL 1.0, acrescida de novas características, as quais acrescentam poder de expressão da linguagem. Por outro lado, quando o assunto é dispor multimídia na Web, o problema está diretamente relacionado ao comportamento da rede. A Internet é uma rede temporalmente não determinista, onde não existe a possibilidade de garantir taxa de bits, atraso e variação de atraso na entrega da mídia. Problemas como este, e outros, deveriam ser contornados pela linguagem que projeta a mídia.

O capítulo anterior descreve os requisitos para uma linguagem de composição de documentos multimídia. Este capítulo tem como objetivo verificar se a linguagem SMIL 2.0 atende a estes requisitos, visto que, ela é uma linguagem de autoria multimídia. Este capítulo também propõe algumas extensões para a linguagem SMIL 2.0 a fim de torná-la mais adaptada a tratar dos problemas causados pelo indeterminismo temporal das redes de computadores. Esta extensão é realizada utilizando namespaces XML.

7.1 Análise da Linguagem SMIL 2.0

Esta seção apresenta uma análise da linguagem SMIL 2.0 em relação aos requisitos básicos para uma linguagem de autoria multimídia na Web. A fig 7.1 ilustra algumas das vantagens em usar SMIL como ferramenta de autoria. As subseções descrevem a análise dos requisitos em confronto com a linguagem SMIL 2.0.

Para descrever a análise, foi usado o mesmo método do capítulo anterior, onde um documento é dividido em três estruturas: estrutura de conteúdo, estrutura conceptual e estrutura de apresentação.

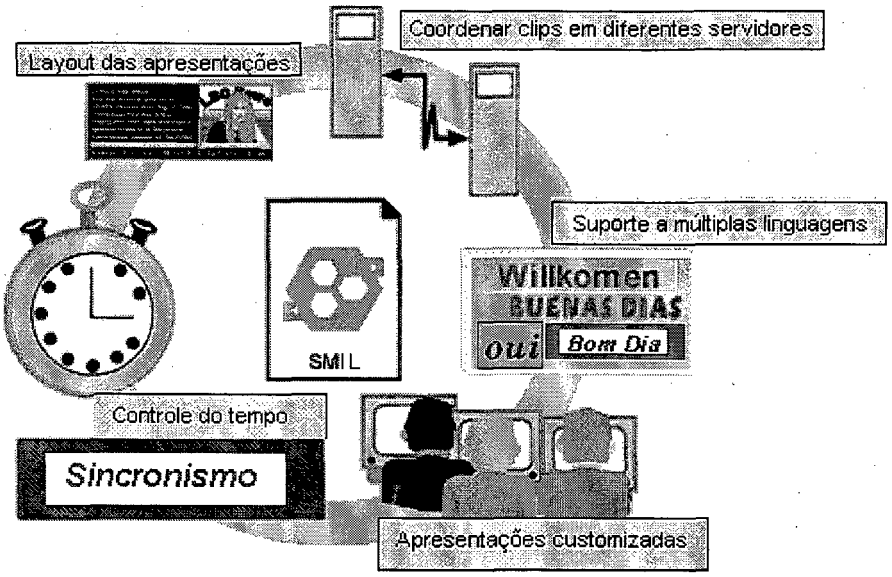


Figura 7-1 – Vantagens em usar SMIL 2.0

7.1.1 Estrutura do conteúdo

Dentro deste nível, o requisito mínimo que uma linguagem de autoria multimídia deve atender é o suporte aos tipos básicos de mídia (texto, áudio, imagem e vídeo). Quanto a estes requisitos, a linguagem SMIL garante suporte e, ainda, produz texto streaming e animações em texto ou imagem.

7.1.2 Estrutura conceptual

Como visto, esta estrutura descreve a estrutura lógica, suas relações lógicas e a composição temporal dos componentes. A análise dos requisitos desta estrutura com relação à linguagem SMIL, está relacionada abaixo.

a) **Mecanismo de estruturação** - Por apresentar composições com semântica de apresentação associada, SMIL possui a estrutura conceitual do documento, diretamente relacionada com a estrutura de apresentação, formada por uma hierarquia de composições aninhadas. Portanto, SMIL não permite ao autor alterar a estrutura da

apresentação sem a necessidade de modificar a estrutura lógica do mesmo. Por exemplo, a inclusão de novos relacionamentos temporais pode implicar em uma reestruturação de todo o documento.

b)**Reusabilidade dos componentes** – SMIL não provê a facilidade de reutilização de composições em um mesmo documento ou em outros documentos, a não ser através de cópias.

c) **Mecanismos de interação** – As aplicações interativas, por exemplo, navegação, controle de apresentação e controle do ambiente, estão presentes em SMIL 2.0.

d) **Especificação das relações temporais** - SMIL permite definir relações temporais, tais como duração, repetição, tempo de início e de fim, congelamento, velocidade, aceleração e outros. No entanto, todos esses eventos são determinados previamente, considerando condições ideais, isto é, nenhum imprevisto é considerado, por exemplo, sobrecarga da rede. A exceção fica para os atributos min e Max que conferem a possibilidade de definir um tempo mínimo ou máximo de duração da mídia, definindo assim uma tolerância no tempo.

e) **O relacionamento temporal entre componentes** - O relacionamento temporal entre componentes de um mesmo documento é alcançado por SMIL através de atributos que possibilitam a exibição de dois ou mais objetos de mídia de forma paralela ou seqüencial. Outra possibilidade é a definição do atributo *excl*, o qual define que somente um dos objetos pode ser apresentado em um dado momento.

f) **Relação temporal entre componentes a partir de eventos** - Os tempos de início e término de uma apresentação podem ser especificados em relação a outros eventos, por exemplo, um clique do mouse.

g) **Caminhos de percurso** – Quando aos caminhos de percurso, para navegação em um documento SMIL, estes são definidos através de links. Estes links são elos entre documentos que são ativados por interação do usuário. A navegação nos documentos

SMIL ainda é restrita entre elos 1:1 (âncora origem para âncora destino). O que a linguagem apresenta é a possibilidade de incorporar um atributo que remete a uma parte interna de um objeto de mídia.

h) **Sincronização temporal entre os componentes** – Um dos itens mais significativos quando o assunto é comunicação em rede, é o controle temporal das mídias a fim de manter sincronismo. SMIL provê controle sobre estas ocorrências, permitindo definir que elementos devem permanecer em sincronismo.

i) **Especificação de informações adaptáveis** - A linguagem SMIL 2.0 oferece recursos para adaptar um documento a diferentes situações. Esta adaptação é feita através do elemento *switch,* o qual possui atributos que possibilitam especificar informações, tais como o idioma, a taxa de transferência disponível na rede, fornecer legenda, o tamanho da tela, especificar a CPU e outros. A funcionalidade destes atributos faz com que o usuário tenha um maior controle na apresentação. Outra funcionalidade desta linguagem é o elemento *prefetch.* Este elemento especifica a quantidade de dados a serem carregados, em tempo ou em bytes, e a largura de banda da rede a ser utilizada.

7.1.3 Estrutura de Apresentação

Nesta estrutura são especificadas as características da apresentação dos componentes da linguagem SMIL.

a) **Especificação dos valores temporais** – Estes requisitos são definidos em SMIL. A linguagem disponibiliza recursos como, posição de início e de fim, recursos de repetição, velocidade e outros. Confere ainda a possibilidade de especificar um valor de início ou término negativo.

b) **Especificação dos valores espaciais** - Os valores espaciais de uma apresentação, tal como, posição, tamanho, estilo de apresentação, animação, são fatores bem tratados na linguagem SMIL.

c) **Especificação dos valores sonoros** - Além de SMIL 2.0 apresentar recursos de sons, incorporou a capacidade de alterar o volume do áudio, da origem e do destino, quando o elo for percorrido.

d) **Apresentações Alternativas** - Apresentações alternativas, que tenham a finalidade de repor uma apresentação devido a problemas de acesso ou restrições temporais, também é verificado em SMIL 2.0. A linguagem apresenta um atributo que substitui a apresentação principal, caso ela não puder ser apresentada.

A seção 7.1 descreveu uma análise da linguagem SMIL 2.0 em relação aos requisitos multimídia. A seção 7.2 descreve como estender SMIL, criando um módulo, um elemento e um atributo, com finalidade de prover alguns dos requisitos faltantes.

7.2 Estendendo SMIL

A linguagem SMIL é uma linguagem emergente, por este motivo o W3C mantém em constante atualização. Desde seu lançamento várias especificações foram verificadas, apesar de ter apenas duas versões. Como SMIL descende de XML e a linguagem XML é extensível, conseqüentemente é possível estender a linguagem SMIL.

Esta proposta visa esclarecer os procedimentos, que são definidos pelo W3C para estender SMIL, a fim de possibilitar a criação de novos módulos, elementos e atributos, para poder suprir possíveis deficiências encontradas na linguagem.

7.3 Módulo Composite

SMIL força uma estrutura hierárquica, não apresentando recursos para poder reutilizar um bloco de informações (cenário) no mesmo documento. Se SMIL fornece-se recursos para reutilizar estes cenários, o nível de hierarquia estabelecido por SMIL poderia ser "quebrado", facilitando a autoria.

O exemplo da fig 7.2 descreve um modelo de como a linguagem SMIL poderia ser estendida para suprir o problema de reusabilidade e hierarquia dentro de um documento. Neste caso, um novo elemento denominado *scene* faz uma chamada através de um atributo *id* identificado pelo nome de *teste,* a um cenário com o mesmo identificador.

```
<!DOCTYPE smil PUBLIC "-//MYORG//DTD SMIL-MYSMIL //EN"
                "http://www.my.org/dtd/mysmil.dtd">
      <smil xmlns="http://www.w3.org/2001/SMIL20/Language"
            xmlns:mysmil= "http://www.my.org/SMIL/composite">

          ....
<head>
 ...
</head>
  <body>
    <seq>
          <text src="texto"/>
          <mysmil:scene="teste"/>
    </seq>
    <par>
          <video src="video1" />
          <audio src="musica1" />
    <par/>
     <mysmil:bodyScene
          <mysmil:scene id="teste">
            <par>
                  <audio scr="musica.au"/>
                  <video scr="video.mpg"/>

<par/>
          <mysmil:scene/>
   <mysmil:bodyScene/>
 </body>
</smil>
```

Figura 7-2 – Documento SMIL para criação de Cenários

Para definir o documento acima é necessário criar um módulo. O novo módulo foi denominado *composite* e os elementos que compõe este módulo são, *bodyscene* e *scene*. B*odyscene* é um elemento que possui como definição à capacidade de implementar cenários. Este módulo deve estar dentro do elemento *body*, e deve aceitar os mesmos elementos e atributos que o elemento *body* aceita. Porém, *bodyscene* recebe um elemento adicional, o elemento *scene*. O elemento *scene* possui o atributo *id*, o qual deve identificar um elemento ou um conjunto de elementos, dentro de um documento.

7.3.1 Definição do Módulo Composite

Para definir este novo módulo é necessário criar um DTD. Basicamente, o autor precisa descrever a declaração dos elementos e atributos se necessário. A fig. 7.3 descreve a declaração do novo módulo.

```
<!-- My Composite Module ..... -->
<!-- file: MYSMIL - composite.mod
          PUBLIC "-//MY COMPANY//ELEMENTS SMIL-MY composite//EN"
          SYSTEM "http://www.my.org/DTDs/composite.mod"
   xmlns:mysmil="http://www.my.org/DTDs/mysmil.dtd"
 -->
<!-- My composite Module
   mysmil:bodyScene
   mysmil:scene
-->
<!ELEMENT mysmil: bodyScene ( #PCDATA | mysmil:scene )* >
<!ATTLIST mysmil: bodyScene "EMPTY">
<!ELEMENT mysmil:scene "EMPTY" >
<!ATTLIST  mysmil:scene
                    %id-attr;
                    %title-attr;
<!-- end of composite.mod -->
```

Figura 7-3 – DTD do Módulo Composite

Criado o novo módulo, é preciso defini-lo no DTD administrador. A listagem abaixo ilustra o DTD administrador, onde constam todos os módulos SMIL 2.0 e inclusive o novo módulo *composite*.

```
<!-- MYSMIL DTD ..................................................... -->
<!-- file: mysmil.dtd -->
<!-- This is the DTD driver for mysmil.
     Please use this formal public identifier to identify it:
         "-//MY COMPANY//DTD SMIL-MYSMIL //EN"
     And this namespace for mysmil-unique elements:
         xmlns:mysmil=http://www.my.org/DTDs/mysmil.dtd  -->
<!ENTITY % SMIL.version  "-//MY COMPANY//DTD SMIL-MYSMIL//EN" >
<!-- Reserved for use with the XLink namespace:-->
<!ENTITY % XLINK.ns "" >
<!ENTITY % XLinkns.attrib "" >
...
<!ENTITY % anim-mod
  PUBLIC "-//W3C//ELEMENTS SMIL 2.0 Animation//EN"
  "SMIL-anim.mod">
<!ENTITY % control-mod
  PUBLIC "-//W3C//ELEMENTS SMIL 2.0 Content Control//EN"
  "SMIL-control.mod">
<!ENTITY % layout-mod
  PUBLIC "-//W3C//ELEMENTS SMIL 2.0 Layout//EN"
  "SMIL-layout.mod">
<!ENTITY % link-mod
  PUBLIC "-//W3C//ELEMENTS SMIL 2.0 Linking//EN"
  "SMIL-link.mod">
<!ENTITY % media-mod
  PUBLIC "-//W3C//ELEMENTS SMIL 2.0 Media Objects//EN"
  "SMIL-media.mod">
<!ENTITY % meta-mod
  PUBLIC "-//W3C//ELEMENTS SMIL 2.0 Document Metainformation//EN"
  "SMIL-metainformation.mod">
<!ENTITY % struct-mod
  PUBLIC "-//W3C//ELEMENTS SMIL 2.0 Document Structure//EN"
  "SMIL-struct.mod">
<!ENTITY % timing-mod
  PUBLIC "-//W3C//ELEMENTS SMIL 2.0 Timing//EN"
  "SMIL-timing.mod">
<!ENTITY % transition-mod
  PUBLIC "-//W3C//ELEMENTS SMIL 2.0 Transition//EN"
 "SMIL-transition.mod">
<!-- My composite Module   ..................................... -->
<!ENTITY % composite-mod
     PUBLIC "-//MY COMPANY//ELEMENTS SMIL-MYSMIL composite//EN"
            "http://www.my.org/DTDs/composite.mod" >
%composite.mod;
%struct-mod;
%anim-mod;
%control-mod;
%meta-mod;
%layout-mod;
%link-mod;
%media-mod;
%timing-mod;
%transition-mod;
```

A especificação do novo módulo juntamente com elementos e atributos deve estar bem definida e disponível para acesso. Por isso, na construção do arquivo SMIL é necessário constar um namespace *xmlns:mysmil*, como no exemplo da fig. 7.3, apontando para um endereço. Neste endereço (por exemplo, http://www.my.org/ SMIL/composite) devem estar todas as especificações (sintaxe e semântica) referentes aos novos elementos e seus atributos, se esses existirem.

7.4 Atributo prepDur

Esta seção visa criar um atributo SMIL, o qual tem a finalidade de controlar o tempo de duração para preparação dos elementos de mídia contínua, permitindo definir um tempo máximo de tolerância em resposta a eventos não previsíveis, como por exemplo, inconsistência da rede.

Esta tolerância de tempo para o preparo das mídias é introduzida através do atributo *prepDur*. A idéia básica é que o algoritmo deva computar o tempo para preparação da entrega da mídia. Neste caso, deve computar o tempo de entrega da mídia utilizando um contador de tempo, o relógio.

O autor do documento deve definir o valor do atraso para ele aceitável, através do atributo *prepDur*. O exemplo da fig. 7.4 descreve um vídeo e um áudio em paralelo, aonde o tempo de duração para ambos é de 15 segundos. Caso o vídeo esteja com problemas na entrega da mídia, o atributo *prepDur* garante a espera do vídeo até o tempo determinado, 5 segundos. Excedido este tempo, o vídeo não poderá mais ser apresentado e um texto poderá ser mostrado em substituição, neste exemplo através do atributo alt. Para o áudio o tempo de tolerância na entrega é de no máximo 2s, caso contrário, também não será apresentado. A figura 7.4 ilustra como poderia ser definido este tempo de tolerância dentro de um documento SMIL.

```
<par>
    <myattrib: video src=video.mpg" dur="15s" prepDur="5s" alt="TEXTO" />
    <myattrib: audio src="audio.rm"  dur="15s" prepDur="2s" />
   ...
  </par>
```

Figura 7-4 – Documento SMIL para o Atributo prepDur

7.4.1 Definindo o atributo prepDur

O módulo *Timing* de SMIL 2.0 contém o submódulo *BasicLineTiming*, o qual possui os atributos, *dur, begin, end, repeatCount* e *repeatDur*. Este novo atributo *prepDur* poderia estar inserido neste submódulo, visto que, vai tratar também da validade temporal de uma mídia. A fig. 7.5 ilustra o DTD para o novo atributo predDur.

```
<!ELEMENT attribute EMPTY>
<!ATTLIST attribute
   myattrib:prepDur  CDATA #IMPLIED
>
```

Figura 7-5 – Definição do atributo prepDur

Para edição de um DTD, qualquer editor de texto dá suporte, porém um editor específico tem a capacidade de destacar a sintaxe, facilitando a construção. Porém para formar e validar o DTD deve ser utilizado um software que possui um parser de validação, por exemplo, XML Spy[10]. Este software é um Ambiente de Desenvolvimento Integrado (IDE) para XML e contém funções úteis que podem simplificar a linguagem. Ele possui ferramentas para editar o documento e um parser que testa o documento para bem-formado e valida este documento, seja um DTD ou XSD Schema, ou ainda do tipo DCD ou XDR.

A figuras abaixo ilustram como o XML Spy testa o DTD. A fig. 7.6 é um exemplo de um documento bem formado. Neste caso, o software testa através de um parser se o DTD foi bem elaborado, isto é, se todas as linhas estão formadas de acordo com as especificações XML (a seta ilustrada na figura abaixo aponta para o resultado do

[10] Disponível em: www.xmlspy.com

documento). Se o documento não está bem formado, ele retorna em um erro e não validará o documento, caso este erro não for corrigido.

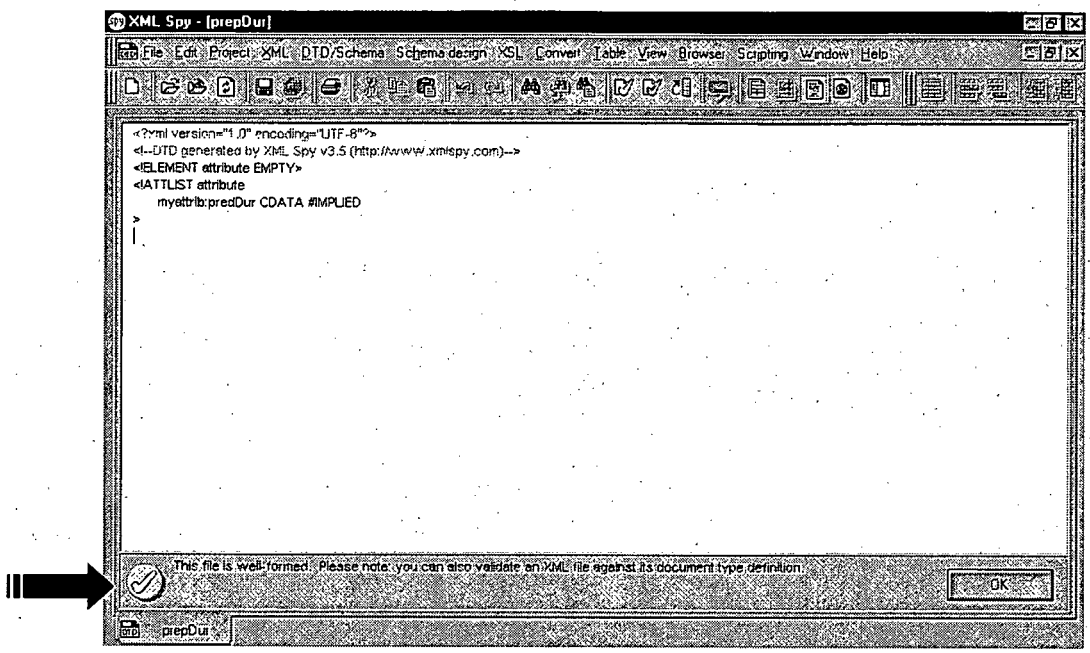


Figura 7-6 - DTD bem formado

A figura 7.7 corresponde ao mesmo documento, porém validado. A seta aponta para o resultado da validação.

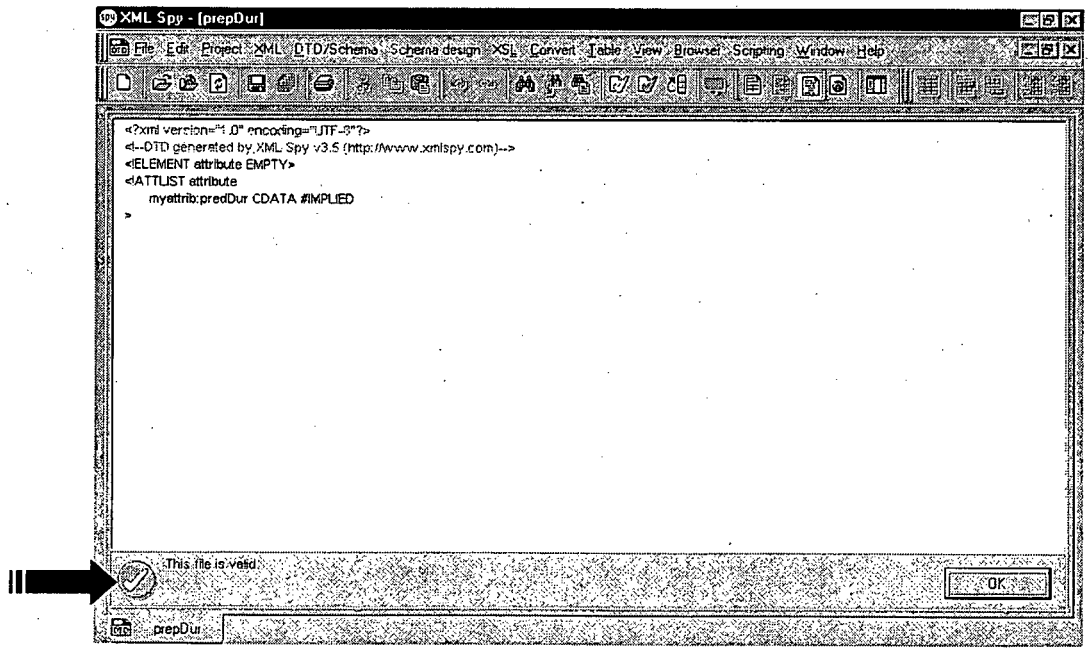


Figura 7-7 - DTD validado

Após ter formatado e validado o DTD, o atributo criado pode ser usado dentro de um documento SMIL. Veja o exemplo abaixo, fig. 7.8, o qual possui um namespace denominado *myattrib,* que aponta para o URI http://www.my.org/SMIL/prepDur. Neste endereço devem estar todas as especificações (sintaxe e semântica) referentes ao novo atributo.

```
< smil xmlns="http://www.w3.org/2001/SMIL20/Language"
  xmlns:myattrib="http://www.my.org/SMIL/prepdur">
   ....
   <par>
     <myattrib: video src=video.mpg" dur="15s" prepDur="5s" alt="TEXTO" />
     <myattrib: audio src="audio.rm"  dur="15s"  prepDur="2s" />
   ...
   </par>
</smil>
```

Figura 7-8 – Namespace para o Atributo prepDur

7.5 Elemento altMedia

O atributo *predDur* definido na seção anterior tem o objetivo de controlar o tempo de duração para preparação dos elementos de mídia, permitindo definir um tempo máximo de tolerância em resposta a eventos não previsíveis. Caso este tempo de tolerância exceda, o ideal seria poder substituir por outra mídia (estática ou contínua) e não somente por um texto, como é o caso do atributo *alt* ilustrado na fig 7.8.

A criação deste novo elemento *altMedia* tem o objetivo de substituir uma mídia, caso esta exceda o tempo de tolerância pré-definido pelo autor no atributo *predDur.* Como mostra a fig. 7.9, onde um vídeo e um áudio tocam em paralelo durante 15 segundos. Caso o vídeo esteja com problemas na entrega da mídia, o atributo *prepDur* garante a espera do vídeo até o tempo determinado, 5 segundos. Após este tempo, o vídeo não deve ser apresentado e uma imagem será apresentada em substituição, através do elemento *altMedia.* Para o áudio o tempo de tolerância na entrega é de no máximo 2s, caso contrário, não será apresentado.

```
<par>
    <myattrib: video src=video.mpg" dur="15s" prepDur="5s" >
        <mysmil:altMedia src=imagem.mpg"/>
    </myattrib:vídeo>
    <myattrib: audio src="audio.rm"  dur="15s"  prepDur="2s" />
  ...
  </par>
```

Figura 7-9 - Documento SMIL para o Elemento altMedia

7.5.1 Definindo o elemento altMedia

O elemento *altMedia* pode utilizar o mesmo DTD do módulo Objeto de Mídia, o qual define os elementos, ref., áudio, img, vídeo, text, textstream e animation. A diferença é que usa um ou mais desses elementos como elemento raiz. Esta integração deve seguir os seguintes passos:

- decidir qual o elemento deve ser o elemento raiz;
- decidir aonde o novo elemento deve ser anexado na árvore.

Para anexar o elemento *altMedia* como filho do elemento *video*, e somente do elemento *video*, a linha do DTD que faz referência ao elemento vídeo deve ser:

```
<!ENTITY % video.content "( mysmil:altMedia )*">
```

Neste caso, o documento SMIL, fig. 7.10, também deve possuir um namespace denominado *mysmil,* que aponta para o URI http://www.my.org/SMIL/ altMedia. Neste endereço devem estar todas as especificações (sintaxe e semântica) referentes ao novo elemento.

Se o autor quiser substituir a mídia por outro tipo e não uma imagem, como definido na fig. 7.10, não haverá problemas, pois como qualquer elemento SMIL, a*ltMedia* é somente um nome de referência, o que deve ser válido é o URL. Porém, se

o autor necessitar definir uma substituição também para o elemento áudio, o DTD para este elemento também deve sofrer as alterações necessárias.

```
< smil xmlns="http://www.w3.org/2001/SMIL20/Language"
  xmlns:myattrib=http://www.my.org/SMIL/prepdur
  xmlns:mysmil= http://www.my.org/SMIL/altmedia >
   ....
<par>
     <myattrib: video src=video.mpg" dur="15s" prepDur="5s" >
        <mysmil:altMedia src=imagem.mpg"/>
    </myattrib:vídeo>
     <myattrib: audio src="audio.rm"  dur="15s"  prepDur="2s" />
    ...
   </par>
</smil>
```

Figura 7-10 - Namespace para o Elemento altMedia

Por outro lado, quando um apresentador SMIL encontra um elemento e/ou um atributo através de namespace ele poderá validar, ignorar ou retornar em erro. No capítulo 4, seção 4.10, há uma descrição de como o apresentador SMIL se comporta em referência a novos elementos e atributos, e o que é necessário para que um documento SMIL e o apresentador estejam em conformidade.

7.6 Conclusão

SMIL é uma linguagem de autoria para dispor multimídia na Web. A análise apresentada neste capítulo centrou-se na versão SMIL 2.0. Para realizar esta análise foi usada à mesma abordagem de verificação de requisitos, que um modelo multimídia na Web deve oferecer.

Dentro dos requisitos mínimos, que uma linguagem de autoria multimídia deve atender, é o suporte aos tipos básicos de mídia, como texto, imagem, áudio e vídeo. SMIL garante suporte a todas estas mídias e, ainda, produz texto streaming e animações em texto ou imagem.

Em relação à estrutura conceptual, foram analisados os requisitos de especificação das relações temporais, sincronização temporal, mecanismos de interação, reusabilidade dos componentes, caminhos de percurso entre outros. Na estrutura de apresentação, foram analisadas as especificações dos valores temporais, espaciais e sonoros e apresentações alternativas. SMIL 2.0 possui vários dos requisitos apontados para uma ferramenta de autoria multimídia, mas novas funcionalidades poderiam ser acrescentadas para aumentar o seu poder de expressão e facilitar a autoria.

Por outro lado, o maior problema para implementar multimídia na Web, decorre principalmente do fato que a Internet é uma rede com comportamento temporal não determinístico, isto é, oferece um serviço do tipo melhor esforço, onde o tráfego é tratado tão rápido quanto possível, mas não há garantias temporais, tornando muitas vezes as sincronizações previstas pelo autor da aplicação inviáveis de serem respeitadas.

Este capítulo descreveu também, uma proposta de extensão da linguagem SMIL 2.0 visando esclarecer os procedimentos, que são definidos pelo W3C, para estender SMIL, a fim de possibilitar a criação de novos módulos, elementos e atributos, para poder suprir possíveis deficiências encontradas na linguagem e em resposta aos problemas da rede de comunicação.

8 Conclusão

O objetivo deste estudo foi levantar requisitos para uma linguagem de composição de documentos multimídia usando como suporte de comunicação, redes temporalmente não deterministas e verificar se a linguagem SMIL 2.0 atende estes requisitos. Esta dissertação também propôs algumas extensões para a linguagem SMIL 2.0, a fim de torná-la mais adaptada a tratar dos problemas causados pelo indeterminismo temporal das redes de computadores.

A linguagem SMIL 2.0 foi o foco de estudo desta dissertação. SMIL é uma linguagem declarativa para descrever apresentações multimídia na Web, que permite integrar um conjunto de objetos multimídia independentes numa apresentação multimídia sincronizada. Os documentos em SMIL são documentos em XML.

XML é uma metalinguagem, pois descreve outras linguagens, tal como SMIL. Para poder entender como funciona a linguagem SMIL e como é possível prover sua extensão, foi necessário fazer um mapeamento da linguagem XML, descrevendo sua estrutura, sua sintaxe e seu funcionamento.

Da mesma forma, foi feito um estudo de namespace XML. Namespace XML está diretamente ligado à extensão da linguagem SMIL, pois define a possibilidade de criar novos elementos e atributos e rotular com clareza quem é o responsável pela extensão, evitando conflitos de nomes e permitindo a reutilização de estruturas.

Este trabalho também apresentou um estudo dos requisitos de um modelo multimídia na Web e uma análise destes requisitos em relação à linguagem SMIL 2.0. A partir desta análise foi elaborada uma proposta para poder possibilitar a extensão de SMIL.

Quanto aos requisitos para um modelo multimídia na Web, a linguagem SMIL possui vários destes requisitos, tais como mecanismos de interação, sincronização temporal entre os componentes e especificação de informações adaptáveis. Outra

vantagem significativa de SMIL é a possibilidade de usar mídias de diferentes servidores. Por exemplo, um vídeo pode estar em um servidor de multimídia e o texto em um servidor Web.

SMIL 2.0 permite definir relações temporais, tais como duração, repetição, tempo de início e de fim, congelamento, velocidade, aceleração e outros. No entanto, todos esses eventos são determinados previamente, considerando condições ideais, isto é, nenhum imprevisto é considerado, por exemplo, sobrecarga da rede. A exceção fica para os atributos min e Max que conferem a possibilidade de definir um tempo mínimo ou máximo de duração da mídia, definindo assim uma tolerância no tempo.

Em relação às características temporais da apresentação, estas são especificadas nos próprios componentes, impedindo a reutilização destas especificações em outros componentes. SMIL também, não provê a facilidade de reutilização de composições em um mesmo documento ou em outros documentos, a não ser através de cópias. Por apresentar composições com semântica de apresentação associada, SMIL possui a estrutura conceitual do documento, diretamente relacionada com a estrutura de apresentação, formada por uma hierarquia de composições aninhadas.

Em SMIL 2.0 não há um limite de tempo tolerável na espera da preparação de uma mídia, causando muitas vezes uma espera prolongada ou até infinita, devido a problemas de comunicação. SMIL 2.0 oferece um atributo para substituir a mídia, caso ocorra um problema de entrega na rede. Porém, este atributo é limitado a aceitar somente texto como substituição da mídia.

Apesar da variedade de elementos e atributos SMIL, para controle temporal das mídias, o problema para implementar multimídia na Web, decorre principalmente do fato que a Internet é uma rede com comportamento temporal não determinístico. A sincronização intramídia e intermídia exige que o sistema de comunicação atenda certos requisitos, como, largura de banda garantida, limitação do atraso e da variação de atraso. Como a Internet não atende estes requisitos, muitas vezes as sincronizações definidas pelo autor das aplicações multimídia se tornam inviáveis de serem respeitadas.

Em resposta a algumas deficiências levantadas na linguagem SMIL, problemas na rede de comunicação e em relação à proposta de expansão da linguagem SMIL, foi desenvolvido três extensões SMIL: um módulo, um elemento e um atributo. O módulo foi criado com o objetivo de fornecer recursos para reutilização de cenários, possibilitando a introdução de reusabilidade dentro de um mesmo documento SMIL. O novo atributo tem a finalidade de controlar o tempo de duração para preparação dos elementos de mídia, permitindo definir um tempo máximo de tolerância em resposta a eventos não previsíveis, visto que SMIL 2.0 não faz este tratamento. A criação do novo elemento tem a finalidade de substituir uma mídia, caso esta exceda o tempo de tolerância pré-definido pelo autor no novo atributo. O módulo, o elemento e o atributo que foram criados possuem além de tudo, o objetivo de elucidar como é feita a expansão da linguagem SMIL.

A sintaxe da linguagem SMIL é simples, muito similar a HTML, denotando facilidade para sua compreensão. Porém, como é uma linguagem nova, os apresentadores disponíveis ainda são limitados, alguns nem suportam todos os elementos que a versão 2.0 possui. Assim sendo, mesmo sendo extensível, um novo módulo, elemento e/ou atributo só será implementado se o apresentador conhecer e entender este novo componente.

Como sugestão para um trabalho futuro, seria realizar um estudo, de forma detalhada, do funcionamento de um apresentador (player) SMIL. Entendendo como um apresentador funciona, tornaria mais compreensível uma análise de uma linguagem de autoria multimídia e conseqüentemente facilitaria o processo de extensão.

Outra sugestão como trabalho futuro, é referente à recomendação *XML Schema Definition Language,* nova proposta do W3C. Um DTD é muito útil para criação de novos componentes, porém por motivos históricos, é um pouco limitado. Desta forma, a recomendação *XML Schema Definition Language* provê recursos para superar algumas limitações da DTD. Como proposta para um trabalho futuro, seria a extensão de SMIL

usando Schemas e DTDs, de modo a comparar suas limitações, verificando o poder de expressão desta nova recomendação.

9 Bibliografia

ANTONACCI, M. J., Muchaluat, D.C., Rodrigues, R.F., Soares, L.F.G. NCL: Uma linguagem Declarativa para Especificação de Documentos Hipermídia na Web. Anais do VI Simp. Brasileiro de Sistemas Multimídia e Hipermídia (SBMidia'2000), pp. 79-95. Natal, 2000.

BOX, Don; SKONNARD, Aaron; LAM, John. Essencial XML, Ed. Addison Wesley, 2000.

BRADLEY, Neil. XML Companion. Ed. Addison Wesley, 2000.

BRAY, D. Hollander. LAYMAN, A. Namespaces in XML. W3C Recommendation, 14 January 1999. Disponível em: http://www.w3.org/TR/REC-xml-names

CEREDA, Ronaldo Luiz Dias e outros. ATM – O Futuro das Redes. Ed. Makron Books do Brasil Editora Ltda, 1997.

COURTAUD, Didier. Introdução ao SMIL, disponível em: http://www.teotonio.org, 1999.

DRISCOLL, Margaret. Web – Based Training. Ed. Jossey-Bass Pfeiffer, São Francisco - California, 1998.

EAGER, Bill. Por dentro da World Wide Web. Ed. Berkeley Brasil Editora, SP - 1995.

FLUCKIGER, François. Understanding Networked Multimedia: Applications and Technology. Prentice Hall International (UK) Limited, 1995.

HELIO. A tutorial to demonstrate SMIL, disponível em: http://www.helio.org/products/smil/tutorial/chapter1/index.html.

KLAS, W. , E. J. Neuhold e M. Schrefl, Using na Object-Oriented Approach to Model Multimedia Data. Computer Communication 13(4):204-216, em 1990.

MARCHAL, Benoit. XML Conceitos e Aplicações. Ed. Berkeley - SP, 2000.

MARTIN, James. Hiper Documentos e como Criá-los, Ed. Campus - RJ, 1992.

MCCANNELL, Steve. SMIL: Multimedia for the Masses, disponível em: http://hotwired.lycos.com/webmonkey/00/41/inde x4a.html, 2000.

MCGRATH, Sean. XML Aplicações Práticas. Ed. Campus-RJ, 1998.

OTTE, Peter. A Super-Rodovia da Informação. Ed. Axcel Books do Brasil editora - RJ, 1995.

PAULA FILHO, Wilson de Pádua. Multimídia Conceitos e Aplicações. Ed. JC Editora – RJ, 2000.

RANDALL, Neil. Discover the World Wide Web with your Sportster, Ed. 2ª Edição, 1997.

RATHBONE, Andy. Multimídia & CD-ROMs. Ed. Berkeley Brasil Editora –SP, 1995.

REAL - Tutorial RealSystem G2. Disponível em: http://service.real.com/help/library/guides/production8/htmfiles/notice.htm, março 2002.

SCHLOSS, G.A., WYNBLATT, M.J.. Building Temporal Streuctures in a Layered Multimedia Data Model. In proc. ACM Multimedia'94, pg. 271-278, 1994

SCHULZRINNE H., RTP: A transport Protocol for Real-Time Applications. Internet Draft draft-ietf-avt-rtp-02.new-01.ps, 1997.

VAUGHAN, Tay. Multimídia na Prática. Ed. Makron Books, SP, 1994.

WAHL, T. Wahl, K. Rothermel. Representing Time in Multimedia Systems. In proc. of the International Conference on Multimedia Computing and Systems (ICMCS'94), pp. 538-543, Boston, 1994.

WILLRICH, Roberto. Sistemas Multimídia Distribuídos. Apostila da disciplina de Sistemas Multimídia Distribuídos, do Curso de Pós Graduação em Ciência da Computação da UFSC, 2000.

W3C, World Wide Web Consortium, SGML - Standard Generalized Markup Language - Comparation of SGML and XML. World Wide Web Consortium, 15 December 1997.

W3C, World Wide Web Consortium. SMIL -Synchronized Multimedia Integration Language Smil 1.0- W3C Recommendation.Disponível em: http://www.w3.org/TR/ REC-smil/, junho 1998.

W3C, World Wide Web Consortium. SMIL - Synchronized Multimedia Integration Language Smil Boston Specification. W3C Recommendation. Disponível em: http://www.w3.org/TR/smil-boston, novembro 1999a.

W3C, World Wide Web Consortium, Namespaces in XML - W3C Proposed Recommendation 14 January 1999. Disponível em http://www.w3.org/TR/REC-xml-names, 1999b.

W3C, World Wide Web Consortium, XML - Extensible Markup Language. Disponível em: http://www.w3.org/XML/, 2000.

W3C, World Wide Web Consortium, SMIL - Synchronized Multimedia Integration Language Smil Boston Specification, W3C Working Draft 05 June 2001. Disponível em: http://www.w3.org/TR/smil-boston/, 2001a.

W3C, World Wide Web Consortium, SMIL - Synchronized Multimedia Integration Language Smil Boston Specification, W3C Proposed Recommendation September 2001. Disponível em: http://www.w3.org/ TR/smil/, 2001b.

W3C, World Wide Web Consortium. Disponível em: http://www.w3.org.br, 2002.

W3C, World Wide Web Consortium, HTML – HyperText Markup Language. Disponível em http://www.w3c.org/ MarkUp, 2002a

W3C, World Wide Web Consortium, SMIL - Synchronized Multimedia Integration Language. Disponível em: http://www.w3.org/smil2.0, 2002b.