LEANDRO MARQUES RODRIGUES

INTEGRAÇÃO DE DOCUMENTOS SMIL AO SISTEMA HYPERPROP E DESENVOLVIMENTO DE FERRAMENTAS PARA EXIBIÇÃO DE OBJETOS COM RELACIONAMENTOS DE SINCRONIZAÇÃO

DISSERTAÇÃO DE MESTRADO

Departamento de Informática

Rio de Janeiro, 21 de Novembro de 2000

LEANDRO MARQUES RODRIGUES

INTEGRAÇÃO DE DOCUMENTOS SMIL AO SISTEMA HYPERPROP E DESENVOLVIMENTO DE FERRAMENTAS PARA EXIBIÇÃO DE OBJETOS COM RELACIONAMENTOS DE SINCRONIZAÇÃO

Dissertação de Mestrado apresentada ao Departamento de Informática da PUC-Rio como parte dos requisitos para obtenção do título de Mestre em Informática: Ciência da Computação.

Orientador: Luiz Fernando Gomes Soares

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, 21 de Novembro de 2000

Este trabalho é dedicado

À minha esposa Fabiana, por todo o amor, o carinho e o incentivo.

Aos meus pais Paulo Cesar e Maria José, por tudo o que fizeram para a minha realização pessoal e profissional.

A Deus, nosso guia em todos os momentos.

AGRADECIMENTOS

- Ao professor, orientador e amigo Luiz Fernando Gomes Soares, por toda a atenção dedicada a este trabalho, pela confiança que em mim depositou e pelo seu exemplo profissional.

- À minha esposa e aos meus pais, irmãos, cunhadas e sobrinhos, por todo o amor, o carinho e o apoio que sempre me dão.

- Aos meus amigos da PUC, da UFRJ e da Embratel, que me incentivaram e me apoiaram na realização deste trabalho.

- À equipe do Laboratório Telemídia, pela amizade adquirida ao longo destes anos e pela ajuda que prestaram para o desenvolvimento deste trabalho. Em especial, aos amigos Rogério Rodrigues e Débora Muchaluat, por todo o apoio e a atenção que dedicaram a este trabalho.

- A todos os professores e funcionários do Departamento de Informática da PUC-Rio que de alguma forma colaboraram para a realização deste trabalho.

- Ao CNPq e à Embratel, pelo apoio financeiro fornecido durante parte do mestrado.

- A Deus, pela oportunidade de viver estes momentos e concluir mais esta etapa de minha vida.

RESUMO

O uso da *World-Wide Web* para distribuir documentos multimídia que apresentam relacionamentos de sincronização temporal e espacial entre seus componentes é um tópico de pesquisa que tem sido crescentemente explorado. A linguagem SMIL teve a sua recomendação publicada pelo W3C para autoria declarativa desses documentos, porém ela possui algumas limitações. Com o objetivo de superar estas limitações, esta dissertação apresenta a integração de documentos SMIL ao HyperProp, um sistema hipermídia baseado no modelo conceitual NCM (*Nested Context Model*). Esta integração permite que documentos SMIL sejam refinados através das facilidades presentes no NCM e manipulados em um sistema hipermídia com suporte a controle de versões e ferramentas de autoria de alto nível. A dissertação também propõe algumas extensões ao modelo NCM, através da incorporação de alguns conceitos presentes na linguagem SMIL, com o objetivo de simplificar a autoria de documentos NCM que possuem relacionamentos de sincronização temporal. Por fim, a dissertação apresenta o desenvolvimento de ferramentas de exibição, integradas ao sistema HyperProp, para controlar a apresentação de mídias contínuas (áudio e vídeo) e imagens estáticas, suportando todas as relações de sincronização temporal e espacial definidas no modelo NCM, além das relações de interação do usuário.

Palavras-chave: sincronização multimídia, ferramentas de exibição, sistema HyperProp, NCM, SMIL

ABSTRACT

The use of the World-Wide Web to distribute multimedia documents with temporal and spatial synchronization relationships is a research topic that has recently gained a lot of attention. The SMIL language had its recommendation published by the W3C for declarative authoring of these documents, but it has some limitations. In order to overcome these limitations, this dissertation presents the integration of SMIL documents with HyperProp, a hypermedia system based on a conceptual model called NCM (Nested Context Model). This integration allows SMIL documents to be refined with NCM facilities and manipulated in a hypermedia system with version control and high-level authoring tools. The dissertation also proposes some extensions to NCM, adding some concepts found in SMIL, in order to simplify the authoring of NCM documents with temporal synchronization relations. Finally, the dissertation presents the development of presentation tools, integrated with the HyperProp system, to control the presentation of continuous media (audio and video) and static images, supporting all temporal and spatial relations defined in the NCM model, besides user interaction relations.

Keywords: multimedia synchronization, presentation tools, HyperProp system, NCM, SMIL

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

API	Application Programming Interface
HTML	HyperText Markup Language
HTML+TIME	Timed Interactive Multimedia Extensions for HTML
JMF	Java Media Framework
MIME	Multipurpose Internet Mail Extensions
NCM	Nested Context Model
QoS	Quality of Service
SMIL	Synchronized Multimedia Integration Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WWW	World Wide Web
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language

Capítulo 1

Introdução

1.1 Motivação

É cada vez maior o interesse em utilizar a World-Wide Web (WWW) para distribuir documentos que apresentem relacionamentos de sincronização temporal e espacial entre seus componentes, além dos relacionamentos convencionais de navegação, disparados por interação do usuário. Como o HTML, principal linguagem usada para criar documentos na Web, não suporta a definição destes relacionamentos de sincronização, diversos outros modelos e linguagens têm sido propostos para atingir este objetivo.

Uma destas propostas é a linguagem SMIL (*Synchronized Multimedia Integration Language*) [SMIL98], adotada como recomendação pelo W3C em junho de 1998. O SMIL é uma linguagem declarativa específica para autoria de documentos multimídia com relações de sincronização temporal e espacial, seguindo o mesmo objetivo de simplicidade do HTML. No entanto, esta simplicidade do SMIL implica algumas limitações como linguagem de autoria de documentos multimídia.

Um modelo mais genérico para especificação de documentos multimídia é o Modelo de Contextos Aninhados (NCM – *Nested Context Model*) [Soar00, SoCR95], desenvolvido no

Laboratório Telemídia da PUC-Rio. O NCM é um modelo orientado a objetos que permite estruturar logicamente os componentes de um documento através do conceito de composições aninhadas. Os relacionamentos entre os componentes são modelados através de composições e elos. As composições definem as relações de estruturação entre os nós, enquanto os elos definem relacionamentos síncronos e assíncronos entre eles.

O NCM é o modelo conceitual do HyperProp [RoMS98], um sistema para tratamento de documentos multimídia que também vem sendo desenvolvido no Laboratório Telemídia. Este sistema oferece uma interface gráfica de alto nível para a criação e a edição de documentos, e um módulo de exibição responsável por controlar a apresentação, denominado formatador. Entre os módulos do formatador estão o executor e as ferramentas de exibição [Rodr97]. O executor é responsável pela construção do plano de exibição do documento e por sua execução através de interações com as ferramentas de exibição. Cabe a estas ferramentas controlar a apresentação dos objetos de mídia, avisando ao executor a ocorrência de eventos e executando as ações solicitadas por ele.

Com o objetivo de superar as limitações da linguagem HTML, o sistema HyperProp foi integrado ao WWW, oferecendo seus recursos para manipular documentos HTML [RoMS98]. Com um objetivo similar, uma das propostas da dissertação é a integração de documentos escritos na linguagem SMIL ao ambiente do sistema HyperProp, através da conversão destes documentos para o modelo NCM. Esta integração permite oferecer aos usuários as facilidades do modelo e os recursos do sistema HyperProp para manipular, refinar e apresentar estes documentos.

Apesar de o NCM possuir um alto poder de expressão para especificar relacionamentos síncronos através de elos, a autoria de documentos que apresentam estes relacionamentos ainda pode gerar algumas dificuldades para o autor, em virtude da complexidade da definição dos elos. Portanto, a dissertação propõe extensões à hierarquia de classes do NCM, possibilitando representar de forma mais simples os relacionamentos de sincronização temporal, semelhante à que é utilizada em um documento SMIL.

O processo de mapeamento da linguagem SMIL para o modelo NCM e o desenvolvimento de extensões ao modelo a partir de elementos presentes nesta linguagem ressaltaram a necessidade de se desenvolverem ferramentas de exibição de mídias que implementassem

os conceitos do modelo relativos à sincronização temporal e espacial entre os objetos de mídia, de forma que os documentos especificados a partir destes conceitos pudessem ser exibidos.

Desta forma, a dissertação também apresenta o desenvolvimento de ferramentas de exibição integradas ao formatador do sistema HyperProp, para controlar a apresentação das mídias associadas aos nós de conteúdo NCM. Ferramentas de exibição de nós de texto em formato HTML já haviam sido implementadas no sistema HyperProp. Este trabalho contribui no que diz respeito ao desenvolvimento de ferramentas para exibição de mídias contínuas (áudio e vídeo) e de imagens estáticas, suportando todas as relações de sincronização temporal e espacial entre os objetos de mídia definidas no modelo, além das próprias relações de interação do usuário. A implementação das ferramentas de exibição foi feita com o suporte oferecido pelo Java Media Framework [JMF99a, JMF99b], que oferece uma API para a exibição de mídias contínuas bastante adequada aos objetivos desta implementação.

Um dos principais desafios no desenvolvimento de ferramentas de exibição é a sincronização temporal a partir do início (ou do término) da apresentação de uma sub-região temporal de uma mídia contínua. Este trabalho propõe uma forma de implementação que permite ajustar a precisão com que o evento é sinalizado em relação à quantidade de recursos de processamento que serão utilizados.

1.2 Objetivos

Os principais objetivos da dissertação são a realização de uma integração entre a linguagem SMIL e o modelo NCM, de modo a acrescer as funcionalidades do modelo aos documentos especificados nesta linguagem, e o desenvolvimento de ferramentas de exibição para a apresentação de documentos NCM obedecendo às relações de sincronização temporal e espacial especificadas. O processo de mapeamento da linguagem SMIL para o NCM também leva à definição de algumas extensões ao NCM para simplificar a autoria de documentos multimídia.

Inicialmente, o trabalho propõe uma integração da linguagem SMIL ao modelo NCM, a partir da conversão de documentos SMIL para este modelo. Nesta conversão, os relacionamentos temporais especificados no documento SMIL são representados através de elos NCM. A conversão proposta visa superar algumas limitações observadas na linguagem SMIL, através das facilidades oferecidas pelo NCM e dos recursos do sistema HyperProp para o tratamento de documentos multimídia.

A partir da especificação desta conversão, foi implementado um conversor automático da linguagem SMIL para o modelo NCM, que permite importar documentos SMIL para o sistema HyperProp. Este conversor é especificado em Java e totalmente integrado ao sistema.

Durante o processo de mapeamento dos elementos da linguagem SMIL para o NCM, verificou-se a utilidade de estender a estrutura de classes do NCM para incorporar alguns elementos da linguagem. Estas extensões permitem simplificar a autoria de documentos NCM que apresentam relações de sincronismo temporal e espacial. A implementação destas extensões do modelo também faz parte do escopo deste trabalho.

Por último, o trabalho propõe a implementação de ferramentas de exibição para a apresentação das diversas mídias (texto, imagem, áudio e vídeo). Estas ferramentas devem estar de acordo com a interface definida pelo formatador HyperProp, sinalizando para o executor a ocorrência de eventos durante a exibição da mídia e executando as ações por ele solicitadas.

Entre os eventos que podem ser gerados pelas ferramentas de exibição desenvolvidas na dissertação estão o início e o final da apresentação de qualquer conjunto de unidades de informação marcadas de um objeto, e a seleção de uma sub-região do componente visual da mídia por interação do usuário. Entre as ações que podem ser executadas sobre as ferramentas de exibição estão a preparação da apresentação de um objeto (alocação dos recursos e busca das unidades de informação), as solicitações de início e término da apresentação e a alteração de parâmetros como o ponto de exibição da mídia e a taxa de exibição.

Além de realizar toda a sincronização temporal determinada pelo executor, as ferramentas de exibição também são responsáveis pela realização de mudanças de comportamento durante a apresentação dos objetos e pela sincronização espacial, através da exibição das mídias nas regiões da janela de apresentação especificadas.

1.3 Organização da dissertação

O Capítulo 2 descreve a linguagem SMIL para especificação de documentos multimídia, discutindo suas vantagens e limitações, e apresenta o Modelo de Contextos Aninhados (NCM), concentrando-se nos aspectos relativos à especificação de relações de sincronização temporal e espacial. No final deste capítulo, é feita uma comparação entre a linguagem SMIL e o modelo NCM.

O Capítulo 3 descreve detalhadamente o mapeamento dos diversos elementos e atributos definidos na linguagem SMIL para o modelo NCM, apresentando também a implementação do conversor SMIL-NCM, que permite importar documentos SMIL para o sistema HyperProp. Em seguida, este capítulo descreve as extensões propostas ao modelo NCM, motivadas pelos conceitos presentes na linguagem SMIL.

O Capítulo 4 é dedicado ao processo de apresentação das diversas mídias pertencentes a um documento NCM, obedecendo às relações de sincronização temporal e espacial especificadas. Ele propõe uma arquitetura para as ferramentas de exibição e apresenta a sua implementação.

O Capítulo 5 apresenta alguns trabalhos relacionados, comparando-os com o trabalho realizado nesta dissertação. Entre os trabalhos relacionados estão o mapeamento da linguagem SMIL para o sistema multimídia Madeus, a descrição das novas funcionalidades da linguagem SMIL presentes em sua versão 2.0, que está em fase final de especificação, e algumas ferramentas de exibição de documentos multimídia existentes.

Finalmente, o Capítulo 6 apresenta as contribuições da dissertação e propõe alguns trabalhos futuros.

Capítulo 2

A Linguagem SMIL e o Modelo NCM

Este capítulo apresenta a linguagem SMIL e o modelo NCM, destacando os aspectos que serão relevantes nesta dissertação. Em seguida, é feita uma comparação entre eles em relação a uma série de características desejáveis em um modelo de especificação de documentos multimídia.

2.1 Especificação de documentos multimídia através da linguagem SMIL

A linguagem SMIL [SMIL98] oferece uma maneira declarativa de especificar a apresentação de documentos multimídia. Um documento é dividido em duas seções: o cabeçalho (*head*) e o corpo (*body*)[1]. O *cabeçalho* contém informações gerais relativas ao documento, tais como título e autor, e a definição do layout espacial da apresentação. O *corpo* contém a definição dos elementos componentes do documento e seus

[1] Como convenção, será apresentada entre parênteses a sintaxe, em SMIL, dos nomes dos elementos e atributos.

relacionamentos temporais, além da especificação de âncoras e elos de navegação. A Figura 1 apresenta um exemplo de documento SMIL.

```
<smil>
  <head>
    <meta name="title" content="matise"/>
    <layout>
      <root-layout id="matise" title="matise" background-color="#fcedc4" width="721" height="587"/>
      <region id="m_title" title="m_title" left="4%" top="4%" width="47%" height="22%"/>
      <region id="V-Main" title="V-Main" left="52%" top="5%" width="45%" height="42%"/>
      <region id="r_title" title="r_title" left="52%" top="57%" width="42%" height="22%"/>
      <region id="music" title="music"/>
    </layout>
  </head>
  <body>
    <par id="CNN-news-tree">
      <seq id="News-at">
        <par id="Opening-Segment">
          <text id="Leader-Title" region="m_ title" src="the.news/html/leader_title.html"/>
          <audio id="Leader-Music" region="music" src="the.news/audio/cnn.logo1.aiff"/>
        </par>
        <par endsync ="first">
          <text id="DCAB-Intro" region="m_title" src="the.news/html/dcab_intro.html" dur="8s"/>
          <a show="new" href="./archives-dcab.smil#1">
            <video id="Anchor" region="V-Main" src="the.news/mpeg/dcab.zoomin.mpv"/>
          </a>
        </par>
      </seq>
      <text id="remote-title" region="r_title" src="the.news/html/remote_title.html" dur="10s"/>
    </par>
  </body>
</smil>
```

Figura 1 – Exemplo de um documento SMIL[2].

O layout espacial do documento contém a definição de regiões (*region*) da janela de exibição onde os objetos serão apresentados. Existe uma região especial que representa a janela inteira onde o documento será exibido (*root-layout*), estando referente a ela a definição de todas as demais regiões.

Os objetos básicos, ou *objetos de mídia*, contidos em um documento SMIL podem ser de diversos tipos, tais como: texto, imagem, vídeo e áudio. Os objetos não contêm o conteúdo dos dados associados, e sim uma descrição onde está incluída uma referência (URI) para o conteúdo propriamente dito. Objetos de mídia podem conter ainda atributos de *corte inicial* (*clip-begin*) e *corte final* (*clip-end*), que limitam o conteúdo do objeto a ser apresentado.

[2] O exemplo apresentado é uma simplificação do documento *Webnews*, desenvolvido pelo CWI, disponível em http://www.cwi.nl/SMIL/webnews/index.html

A linguagem SMIL utiliza o conceito de composição para estruturar a apresentação do documento. Uma composição pode conter objetos de mídia e outras composições, determinando a forma como seus componentes deverão ser exibidos. Existem dois tipos de composição, a paralela (*par*) e a seqüencial (*seq*). Em uma *composição paralela*, os componentes são exibidos simultaneamente, enquanto em uma *composição seqüencial* os componentes são apresentados em seqüência. O corpo de um documento SMIL define implicitamente uma composição seqüencial.

Todo objeto de mídia, assim como toda composição, possui uma duração que determina o seu tempo de exibição. Objetos de mídia contínua, como áudio e vídeo, possuem uma duração implícita derivada de seu próprio conteúdo, enquanto em objetos de mídia discreta, como texto e imagem, a duração implícita é nula. A duração implícita de uma composição corresponde ao resultado das durações de seus componentes. Além da duração implícita, todo elemento, seja ele um objeto de mídia ou uma composição, pode ter a sua duração definida explicitamente pelo autor, através de um atributo opcional denominado *duração* (*dur*). Quando essa duração for menor que a duração implícita, a exibição será interrompida, ao passo que se a duração explícita for maior, a exibição será prolongada (no caso de um vídeo, por exemplo, o último quadro poderia permanecer na tela). A duração explícita de um elemento também pode receber o valor *indefinido* (*indefinite*), indicando que o término da exibição do elemento coincidirá com o término da exibição da composição que o contém.

Além do atributo duração, os elementos SMIL possuem outros dois atributos opcionais, denominados *início* (*begin*) e *término* (*end*), que também especificam o comportamento temporal da apresentação. Caso um elemento com o atributo início/término definido pertença a uma composição paralela ou seja o primeiro componente de uma composição seqüencial, ele inicia/termina sua apresentação decorrido o intervalo de tempo especificado desde o início da apresentação da composição que o contém. Caso o elemento seja um outro componente, diferente do primeiro, de uma composição seqüencial, o valor do atributo especifica um intervalo de tempo para o início/término da apresentação do componente, contado a partir do término do componente anterior. Tanto em composições paralelas quanto em seqüenciais, os atributos início e término dos elementos também

podem ser especificados em relação ao início ou término de qualquer outro elemento da composição.

Adicionalmente, as composições paralelas possuem o atributo *endsync*, que 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*) ou *último* (*last*), para especificar o término da composição em relação ao primeiro ou ao último componente a terminar a exibição, respectivamente. Caso esse atributo não seja especificado, as composições paralelas terminam pelo último componente.

Um mesmo elemento pode ser exibido diversas vezes em seqüência. O número de exibições é determinado pelo atributo *repetições* (*repeat*), cujo valor pode ser um número inteiro ou indefinido. Neste último caso, o elemento será exibido repetidamente até o término da composição que o contém.

A linguagem SMIL permite ainda a especificação de sub-regiões (âncoras) temporais e espaciais em objetos de mídia e a definição de hiper-elos entre elas. Os elos podem ser definidos não apenas entre sub-regiões de um objeto de mídia, mas também entre elementos (objetos de mídia inteiros ou composições), ou entre uma sub-região e um elemento. Elos também podem ser definidos de um elemento ou sub-região para um documento inteiro ou um elemento específico de um outro documento. Uma âncora de origem ou de destino correspondente a uma sub-região de um objeto de mídia é especificada através do elemento *anchor*, enquanto uma âncora de origem correspondente a um objeto de mídia inteiro ou a uma composição é especificada através do elemento *a*. Estes mesmos elementos especificam os elos SMIL nas âncoras de origem, quando a âncora de destino é identificada através do atributo *href*.

Um elo SMIL possui um atributo denominado *show*, que define o comportamento do objeto de origem quando o elo é percorrido. O valor desse atributo pode ser: *replace*, que especifica que a apresentação do objeto de origem será substituída pela apresentação do objeto de destino; *new*, que especifica que a apresentação do objeto de destino acontecerá em outro contexto, sem afetar o objeto de origem; e *pause*, que especifica que a

apresentação do objeto de origem será suspensa e retomada após o final da apresentação do objeto de destino, que ocorre em um outro contexto.

Por fim, a linguagem SMIL provê meios para definir comportamentos alternativos da apresentação de um documento, dependendo de fatores inerentes ao sistema (por exemplo, a largura de banda oferecida pela rede) ou de preferências do usuário (por exemplo, o idioma). Isto é feito através do elemento *switch*, que permite ao autor do documento especificar um conjunto de elementos alternativos, dos quais no máximo um será escolhido, de acordo com um ou mais testes a serem realizados em tempo de execução. Estes elementos são declarados em ordem, e somente o primeiro que satisfizer aos testes é apresentado.

2.2 O Modelo de Contextos Aninhados

O Modelo de Contextos Aninhados (NCM – *Nested Context Model*) [Soar00] é um modelo conceitual de dados hipermídia baseado nos conceitos usuais de nós e elos. No NCM, um *nó* possui como principais atributos seu conteúdo, uma lista ordenada de âncoras e um descritor. O *conteúdo* identifica um conjunto de unidades de informação. Uma *âncora* identifica um subconjunto de unidades de informação marcadas do conteúdo. A noção exata de unidade de informação faz parte da definição da classe nó. Por exemplo, um vídeo pode ter como unidade de informação um quadro, enquanto um texto pode ter como unidade de informação um caracter ou uma palavra. Todo nó NCM possui uma âncora especial denominada λ, que identifica o conteúdo total do nó. O *descritor* contém informações que determinam como o nó deve ser apresentado no tempo e no espaço.

O NCM distingue duas classes básicas de nós, denominadas nó de conteúdo e nó de composição. Intuitivamente, um *nó de conteúdo* contém dados cuja estrutura interna é dependente da aplicação, modelando um nó hipermídia tradicional. Os nós de conteúdo podem ser especializados em outras classes (texto, gráfico, áudio, vídeo, etc.), conforme requerido pelas aplicações. A estruturação lógica de um documento é feita introduzindo o conceito de composição. O conteúdo de um *nó de composição* é uma lista de elos e nós, que podem ser de conteúdo ou de composição, recursivamente. Uma restrição importante é feita: uma composição não pode estar recursivamente contida em si mesma.

Uma relação de estruturação lógica pode possuir diferentes semânticas, por exemplo, estruturação para definição de sincronização (apresentação), contextos, controle de versões, etc. Essas semânticas podem ser capturadas em subclasses da composição, conforme discutido em [SoMR98]. Uma destas subclasses é o *contexto de usuário*, que permite organizar os nós de uma forma estruturada, assim como um livro é organizado em capítulos, seções e subseções. A Figura 2 ilustra um exemplo de documento estruturado com o uso de composições do tipo contexto de usuário. A composição B_1 representa o documento (um livro), sendo composta por outras três composições: C_1, C_2 e C_3 (os capítulos). A composição C_1, por sua vez, contém as composições $S_{1.1}$ e $S_{1.2}$ (as seções), além de um nó de conteúdo O_1 (um parágrafo ou uma figura). Relações diferentes das relações de estruturação são representadas por elos. Por exemplo, um relacionamento entre os objetos O_1 e O_2 é representado através do elo l_1.

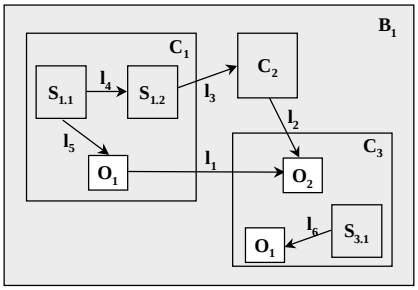

Figura 2 – Exemplo de documento estruturado em composições

Como um nó pode estar recursivamente contido em diferentes composições (por exemplo, nó O_1 nas composições C_1 e C_3 na Figura 2) e como nós de composição podem estar aninhados em qualquer profundidade, é necessário introduzir o conceito de perspectiva. Intuitivamente, a *perspectiva* de um nó identifica através de que seqüência de nós de composição aninhados uma dada instância de um nó está sendo observada. Note que pode haver diferentes perspectivas para um mesmo nó, se ele estiver contido em mais de uma composição. Por exemplo, na Figura 2, o nó O_1 possui duas perspectivas diferentes: P_1=(B_1, C_1, O_1) e P_2=(B_1, C_3, O_1). Além disso, um mesmo nó pode ter diferentes elos associados, dependendo da sua perspectiva, como é o caso do nó O_1.

No NCM, a unidade básica de sincronização para especificar relacionamentos entre nós é o evento. Um *evento*, conforme definido em [PeLi96], é uma ocorrência no tempo que pode

ser instantânea ou durar um período de tempo. O NCM define três tipos de eventos: apresentação, seleção e atribuição. Um *evento de apresentação* é definido pela exibição de uma âncora de um nó em uma dada perspectiva. Um *evento de seleção* é definido pela seleção de uma âncora de um nó em uma dada perspectiva. Um *evento de atribuição* é definido pela mudança no valor de um atributo de um nó ou de um descritor associado ao nó em uma dada perspectiva.

No NCM, um evento pode estar em um dos seguintes estados: *dormindo*, *preparando*, *preparado*, *ocorrendo*, *suspenso* e *abortado*. A máquina de estados dos eventos de apresentação está apresentada na Figura 3.

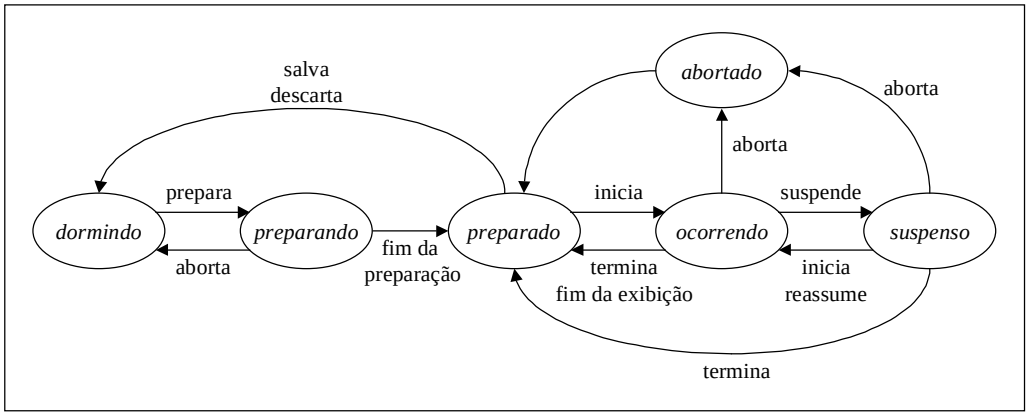

Figura 3 – Máquina de estados dos eventos de apresentação do modelo NCM

Intuitivamente, um evento de apresentação inicia no estado *dormindo*. O evento passa para o estado *preparando* e no mesmo permanece enquanto estiver sendo executado algum procedimento de *prefetch* de suas unidades de informação. Ao final do procedimento, o evento passa para o estado *preparado*[3]. Ao iniciar a apresentação de suas unidades de informação, o evento passa para o estado *ocorrendo*. Se a apresentação for temporariamente suspensa, o evento vai para o estado *suspenso* e no mesmo permanece enquanto a situação durar. Quando uma apresentação de um evento é interrompida abruptamente, através de um comando de aborto da exibição, o evento passa para o estado

[3] O final do procedimento de *prefetch* não implica que todas as unidades de informação relativas ao evento tenham sido buscadas, e sim um número de unidades suficiente para garantir a qualidade da exibição.

abortado e automaticamente volta ao estado *preparado*. O estado de um evento de apresentação pode mudar de *ocorrendo* para *preparado* em duas circunstâncias: como conseqüência do fim da exibição, ou seja, o término da duração da apresentação, ou devido a uma ação que termina o evento. As ações que geram as transições de estado serão apresentadas mais adiante nesta seção.

Assim como na linguagem SMIL, a duração de um evento pode ser intrínseca ao elemento que está sendo apresentado ou especificada pelo autor. No NCM, a *duração* de um evento de apresentação pode ser especificada como um valor fixo ou como uma função de custo, onde valores mínimo, ideal e máximo podem ser definidos. A Figura 4 apresenta exemplos de funções de custo. Intuitivamente, estas funções determinam o custo de reduzir ou estender a duração a partir de um valor ideal, que é dado pelo menor valor da função. Uma função de custo pode possuir múltiplos valores ideais. O NCM ainda permite que a duração seja especificada como um valor indefinido. Exemplos de eventos com duração indefinida são aqueles que são apresentados até que alguma ação externa os interrompa, e aqueles que terminam suas apresentações por conta própria em algum instante não conhecido previamente.

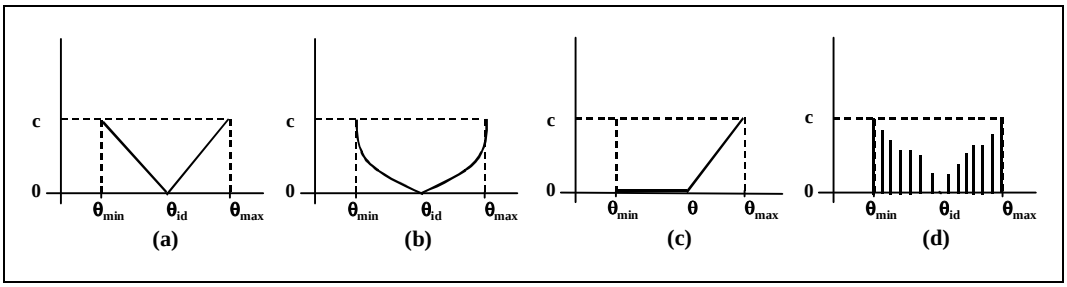

Figura 4 – Exemplos de funções de custo. (a) Função contínua linear com um valor ideal (θ_{id}). (b) Função contínua não-linear com um valor ideal (θ_{id}). (c) Função contínua linear com múltiplos valores ideais ($[\theta_{min}, \theta]$). (d) Função discreta com um valor ideal (θ_{id}).

Esta flexibilidade na especificação da duração de um evento de apresentação permite manter a consistência temporal do documento, satisfazendo a todos os requisitos temporais especificados pelo autor, mesmo que não seja possível utilizar os valores ideais (de custo mínimo) para as durações. Além disso, esta flexibilidade permite que ajustes sejam feitos para corrigir a apresentação quando eventos inesperados ocorrerem, como interações do usuário ou atrasos na transmissão dos dados. O formatador temporal procura ajustar as

durações dos eventos de modo a atingir a melhor qualidade possível para o documento como um todo, minimizando o custo global.

Todo evento possui ainda um atributo denominado *ocorrências*, que conta o número de vezes que o evento muda do estado *ocorrendo* para o estado *preparado* durante a apresentação do documento. Eventos de apresentação também possuem um atributo denominado *repetições*, que determina quantas vezes a âncora associada ao evento deve ser exibida em seqüência, de forma semelhante ao atributo *repeat* da linguagem SMIL.

O *elo* é uma entidade do modelo NCM utilizada para representar relacionamentos entre os objetos. Os relacionamentos especificados pelos elos podem ser causais ou de restrição.

Em relacionamentos causais, uma ação é executada quando uma condição específica é satisfeita. Um exemplo de relacionamento causal é o seguinte: "se o nó *A* está sendo apresentado (condição) e o nó *B* terminou a sua apresentação (condição), apresente o nó *C* (ação)."

Um elo causal representando um relacionamento *m:n* é composto por *m* pontos terminais de origem, *n* pontos terminais de destino e um ponto de encontro. Os pontos terminais definem eventos, enquanto o ponto de encontro possui a descrição do relacionamento entre os eventos. Uma vez que um nó pode pertencer a mais de uma composição, os *pontos terminais* (de origem ou de destino) de um elo são especificados como tuplas da forma $<(N_k,...,N_1), \alpha, tipo, D>$, onde: $(N_k,...,N_1)$ é a seqüência de nós aninhados para atingir N_1, em relação à composição que contém o elo; α é a identificação de uma âncora ou um atributo de N_1; *tipo* especifica o evento associado a α (apresentação, seleção ou atribuição); e *D* é um conjunto de descritores alternativos, dos quais um será escolhido para ser associado a N_1, dependendo da melhor QoS possível para uma dada plataforma.N_1 é denominado *nó âncora* do evento.

O *ponto de encontro* de um elo contém uma condição e uma ação. A condição é avaliada em relação aos pontos terminais de origem, enquanto a ação é executada sobre os pontos terminais de destino. Quando a condição é satisfeita, a ação associada é executada.

A *condição* do ponto de encontro avalia valores booleanos, podendo ser binária simples ou composta. Uma *condição binária simples* é formada por duas condições unárias: uma *condição prévia*, que deve ser satisfeita imediatamente antes do momento da avaliação, e uma *condição corrente*, que deve ser satisfeita no momento da avaliação. A condição binária simples é satisfeita se ambas as condições unárias são satisfeitas. Uma condição unária consiste em uma comparação realizada com respeito aos estados de um evento, aos atributos de um evento, ou aos atributos de um nó (ou descritor associado), no caso de um evento de atribuição. Ela também pode receber o valor *verdade*, caso não seja relevante na avaliação da condição binária simples. Uma condição composta é formada por qualquer expressão lógica envolvendo outras condições, simples ou compostas, baseada nos operadores $\wedge$ (e), $\vee$ (ou) e $\neg$ (negação). Além destes operadores, há um outro operador $\oplus$ (*retardo*), que pode ser aplicado a qualquer condição, simples ou composta. O operador de retardo aplicado a uma condição C é expresso como $C \oplus [t_1, t_2]$, onde $t_1, t_2 \in \Re$ e $0 \leq t_1 \leq t_2$. Dado que uma condição C é satisfeita em um instante t, a condição C', definida como $C \oplus [t_1, t_2]$, é satisfeita no intervalo $[t+t_1, t+t_2]$. Se $t_1=t_2$, o operador de retardo aplicado a uma condição C pode ser expresso simplesmente como $(C \oplus t_1)$.

A *ação* do ponto de encontro também pode ser simples ou composta. Uma ação simples é executada sobre um evento pertencente ao conjunto de pontos terminais de destino. As ações simples aplicadas sobre pontos terminais que definem eventos de apresentação e atribuição estão descritas a seguir.

a) Ações que se aplicam a pontos terminais que definem eventos de apresentação:

1. *Prepara(E)*: Se o evento *E* estiver no estado *dormindo*, ele passa para o estado *preparando*. Caso contrário, nada ocorre. Ao final do procedimento de preparação, o atributo *repetições* de *E* é iniciado com o valor especificado no descritor associado ao nó âncora. Caso não haja valor especificado, o atributo recebe o valor 1.
2. *Inicia(E, n)*: Se o evento *E* estiver nos estados *dormindo*, *preparando*, *preparado* ou *suspenso*, ele passa para o estado *ocorrendo* através da máquina de estados (ver Figura 3), iniciando a sua apresentação a partir da primeira unidade de informação. Caso contrário, nada ocorre. Essa ação possui um parâmetro opcional *n* para iniciar o

atributo *repetições* de *E*. Caso esse parâmetro não seja especificado, o atributo *repetições* é iniciado com o valor declarado no descritor associado ao nó âncora.

3. *Termina(E)*: Se o evento *E* estiver nos estados *ocorrendo* ou *suspenso*, ele passa para o estado *preparado*. Caso contrário, nada ocorre. Essa ação incrementa de uma unidade o atributo *ocorrências* e decrementa de uma unidade o atributo *repetições* de *E*. Se, após decrementado, o valor do atributo *repetições* for maior que zero, uma nova apresentação de *E* é iniciada automaticamente.
4. *Suspende(E)*: Se o evento *E* estiver no estado *ocorrendo*, ele passa para o estado *suspenso*. Caso contrário, nada ocorre.
5. *Reassume(E)*: Se o evento *E* estiver no estado *suspenso*, ele volta ao estado *ocorrendo*. Caso contrário, nada ocorre. A apresentação de *E* é retomada a partir da última unidade de informação exibida antes da apresentação do evento ser suspensa.
6. *Aborta(E)*: Se o evento *E* estiver nos estados *ocorrendo* ou *suspenso*, ele passa para o estado *abortado*, e imediatamente depois para *preparado*. O atributo *ocorrências* de *E* não é incrementado, e o atributo *repetições* recebe o valor zero. Se o evento *E* estiver no estado *preparando*, ele passa para o estado *dormindo*. Se o estado inicial for qualquer outro, nada ocorre.

b) Ações que se aplicam a pontos terminais que definem eventos de atribuição:

1. *Habilita(E)*: Habilita a lista de operações associada ao evento *E*. Essas operações são definidas nos objetos descritores, conforme será visto mais adiante nesta seção. O atributo α do ponto terminal que define o evento deve identificar uma âncora.
2. *Inibe(E)*: Inibe a lista de operações associada ao evento *E*. O atributo α do ponto terminal que define o evento deve identificar uma âncora.
3. *Ativa(E, c)*: Ativa a lista de operações associada ao evento *E*, passando como parâmetro a condição satisfeita *c* que desencadeou a ação. O atributo α do ponto terminal que define o evento deve identificar uma âncora. As operações são executadas somente se estiverem habilitadas.
4. *Atribui valor absoluto(E, i)*: O atributo especificado por α no ponto terminal que define o evento *E* passa a conter o valor *i*.
5. *Atribui valor relativo(E, i)*: O atributo especificado por α no ponto terminal que define o evento *E* passa a conter o valor anterior somado a *i*.

Uma ação composta é definida por uma expressão baseada nos operadores | (paralelo) e → (seqüencial), especificando a ordem de execução de cada ação da expressão. Além disso, toda ação simples ou composta possui um atributo opcional denominado *tempo de espera*. Esse atributo define um tempo que deve ser aguardado antes que a ação seja executada. Da mesma forma que a duração do evento, o *tempo de espera* pode ser especificado como uma função de custo.

Elos causais não são suficientes para especificar todas as relações possíveis em um documento multimídia. Existem relações que, semanticamente, especificam restrições (*constraints*) entre eventos, sem nenhum relacionamento causal. Um exemplo disto é uma restrição especificando que dois nós devem terminar sua apresentação ao mesmo tempo. O NCM define o *elo restrição* como um elo cujo conjunto de pontos terminais de destino é vazio. O ponto de encontro deste tipo de elo especifica uma restrição entre os eventos especificados nos pontos terminais de origem.

Um modelo de documentos hipermídia/multimídia deve permitir a definição do comportamento esperado de cada nó quando apresentado, especificando o layout da apresentação e como o comportamento da apresentação se altera com o tempo. No NCM, essas informações são especificadas de forma separada do nó, permitindo uma melhor reutilização dos objetos. Utilizando layouts distintos, um usuário pode definir diferentes apresentações para um mesmo objeto. Por exemplo, uma cadeia de caracteres poderia ser apresentada como texto ou sintetizada em áudio, dependendo do layout especificado. Uma vez que layouts distintos podem levar a diferentes durações de apresentação, diferentes requisitos de plataforma e diferentes qualidades da apresentação, todas essas especificações devem também fazer parte do layout, e não do nó propriamente dito.

No NCM, a especificação do layout espacial e temporal e as mudanças de comportamento durante a apresentação dos nós são definidas no objeto descritor. A apresentação de um nó do documento para o usuário final (leitor do documento) é feita a partir da associação deste nó a um descritor, dando origem a um *objeto de representação*. O modelo também permite que o descritor seja especificado pelo leitor durante a apresentação. Caso o leitor não especifique um descritor, o sistema deve utilizar um descritor previamente definido pelo autor. Nesse último caso, o descritor pode ser especificado no elo (parâmetro do ponto

terminal de destino), na composição que contém o nó ou como um atributo do nó, sendo esta a respectiva ordem de prioridade na escolha do descritor a ser utilizado. Caso nenhum descritor esteja especificado, é utilizado um descritor default definido para a classe do nó.

O descritor possui como principais atributos uma especificação de iniciação, uma especificação de término e uma coleção de descrições de eventos. Os atributos de um descritor podem ser estendidos com outros atributos dependentes da classe do nó.

A *especificação de iniciação* define o método de exibição/edição do nó, que pode identificar qualquer programa ou pode ser implementado no próprio descritor. Ela também especifica uma lista ordenada de operações que devem ser executadas para preparar a apresentação do nó. A *especificação de término* define os métodos que devem ser executados ao final de uma exibição, e possui uma lista ordenada de operações a serem executadas ao finalizar a exibição do nó.

A *lista de operações* contém uma seqüência ordenada de operações. Cada operação é composta por uma condição e uma ação, de forma semelhante às condições e ações do ponto de encontro de um elo. A única diferença é que o seu escopo está limitado aos eventos e atributos do nó associado ao descritor, ou aos atributos definidos no próprio descritor.

Cada *descrição de evento* consiste em uma tupla da forma <α, *tipo*, *dur*, *rep*, *oper*, *hab*>. O parâmetro α identifica uma âncora do nó ao qual o descritor será associado. *Tipo* especifica o tipo de evento associado à âncora (apresentação ou seleção) ou tem o valor especial ξ. *Dur* especifica uma função de custo, que será o valor inicial do atributo *duração* do evento. *Rep* especifica o número de repetições do evento, que será o valor inicial do atributo *repetições* do evento, caso o mesmo não seja iniciado por uma ação de um elo. *Dur* e *rep* são utilizados somente em eventos de apresentação. *Oper* especifica um objeto lista de operações. *Hab* indica se a lista de operações pode ser executada ou não. Se o parâmetro *tipo* especificar um evento de apresentação ou seleção, e o parâmetro *hab* permitir, a lista de operações é avaliada sempre que houver uma mudança no estado do evento. Caso o parâmetro *tipo* receba o valor ξ, e o parâmetro *hab* permitir, a lista de operações deve ser avaliada se o nó associado ao descritor receber uma mensagem de um elo especificando a âncora, resultado da ação *ativa* de um ponto de encontro.

2.3 Comparação entre a linguagem SMIL e o Modelo de Contextos Aninhados

A linguagem SMIL tem como uma de suas grandes vantagens a simplicidade para a especificação de documentos multimídia. Esta simplicidade permite que o SMIL seja bastante difundido e utilizado para a distribuição de documentos na Web, assim como ocorre com a linguagem HTML.

No entanto, a simplicidade do SMIL implica algumas limitações como linguagem de autoria de documentos multimídia. A expressividade limitada desta linguagem provém da expressividade limitada do modelo no qual ela está baseada.

O NCM consiste num modelo mais genérico e poderoso. A sua grande expressividade permite especificar relações mais abrangentes entre os componentes do documento multimídia. Por outro lado, ele apresenta um grau de complexidade mais elevado, em comparação com a linguagem SMIL.

A seguir, estão comentadas algumas limitações da linguagem SMIL, e como elas são superadas no modelo NCM:

1. Estrutura lógica vs. estrutura de apresentação

Os relacionamentos temporais na linguagem SMIL são definidos através de composições paralelas e seqüenciais. A estrutura lógica do documento é necessariamente a estrutura de apresentação, formada por uma hierarquia de composições aninhadas. Sem outras entidades para definir relacionamentos temporais, algumas alterações nas especificações de sincronização, por exemplo, a inclusão de novos relacionamentos temporais entre objetos, podem implicar uma reestruturação de todo o documento. Por sua vez, o NCM permite definir os relacionamentos de sincronização temporal entre componentes também através de elos, permitindo ao autor alterar a estrutura de apresentação do documento sem a necessidade de modificar a estrutura lógica do mesmo.

Além disso, a linguagem SMIL não possui um tipo de composição sem uma semântica de apresentação associada. Desta forma, não é possível estruturar um documento de forma

independente do modo como ele será apresentado. O NCM oferece composições não apenas para estruturar a apresentação de um documento, mas também para outras estruturações lógicas de seus componentes, como contexto, derivação e autoria cooperativa.

2. Especificação de relacionamentos complexos entre componentes

Relações assíncronas em um documento SMIL são sempre elos *1:1* ativados através da seleção da âncora de origem do elo pelo usuário, enquanto todas as relações síncronas são especificadas por composições paralelas e seqüenciais e por atributos dos objetos (início, término, duração e número de repetições da apresentação). Com isso, não é possível definir relacionamentos mais complexos entre os objetos. Por exemplo, a seguinte especificação não pode ser representada em SMIL: "se a apresentação do objeto *A* estiver suspensa quando a apresentação da âncora α do objeto *B* terminar, então reinicie a apresentação do objeto *A* e aumente a taxa de exibição do objeto *B* em 5%".

Por outro lado, o NCM oferece suporte para definição de relacionamentos *m:n* entre os componentes de um documento através do uso de elos, combinando eventos de interação do usuário (seleção), eventos síncronos (apresentação) ou ainda eventos que manipulam atributos dos objetos (atribuição). Além disso, o NCM ainda permite a definição de relacionamentos que não possuem uma semântica causal, mas que representam uma restrição entre eventos, conforme apresentado na Seção 2.2. Desta forma, é possível especificar relacionamentos bem mais complexos no modelo NCM.

3. Mudanças de comportamento durante a apresentação

A linguagem SMIL não oferece suporte para especificar mudanças de comportamento durante a apresentação de um objeto. Por exemplo, no caso de um áudio, uma especificação do tipo "aumente o volume para *X* db decorridos *t* segundos do início de sua apresentação" não pode ser definida. O NCM permite especificar estas mudanças de comportamento, através do objeto descritor.

4. Flexibilidade na especificação temporal

A linguagem SMIL não oferece muita flexibilidade para especificar a duração da apresentação dos objetos, por exemplo, permitindo a definição de intervalos temporais e funções de custo. O NCM oferece esta flexibilidade, permitindo ajustes necessários durante a apresentação do documento, por exemplo, em resposta a atrasos no sistema de comunicação, e reduzindo as chances de ocorrerem inconsistências temporais [BuZe93, KiSo95].

5. Especificação de características da apresentação

A linguagem SMIL não permite a especificação de características temporais da apresentação de um componente em um elemento separado, mas apenas das características espaciais. Esta facilidade, oferecida no NCM através do objeto descritor, permite associar layouts (temporais e espaciais) de apresentação diferentes a um mesmo componente, ou associar um mesmo layout de apresentação a diferentes componentes. O NCM permite ainda que o layout de apresentação de um componente seja escolhido dependendo da navegação feita para atingi-lo.

6. Reutilização de componentes do documento

A linguagem SMIL não provê a facilidade de reutilização das estruturas de apresentação através de referências. As estruturas só podem ser reutilizadas através de cópias, o que dificulta a manutenção. A linguagem só provê a reutilização do conteúdo dos objetos de mídia, que são identificados através de URIs, e de layouts espaciais de apresentação dentro de um mesmo documento. No NCM, como as composições podem estar contidas em diferentes composições, é possível reutilizar partes da estrutura de um documento (objetos e relacionamentos) no mesmo documento ou em outros documentos, além dos objetos de mídia (nós de conteúdo).

7. Elos dependentes do contexto

Na linguagem SMIL, os elos não são definidos no escopo da composição. Esta facilidade, presente no NCM, aliada à possibilidade de reutilização dos componentes, permite que um

mesmo componente tenha diferentes elos associados, dependendo do aninhamento de composições no qual ele está contido, conforme visto na Seção 2.2.

Capítulo 3

Mapeamento da Linguagem SMIL para o Modelo NCM

O capítulo anterior apresentou uma série de limitações da linguagem SMIL, quando comparada ao modelo NCM. Este capítulo apresenta a integração da linguagem SMIL ao modelo NCM, de modo a superar estas limitações. Esta integração é feita através de um mapeamento dos elementos definidos na linguagem SMIL para objetos do modelo NCM, permitindo a conversão de documentos especificados em SMIL para este modelo. O objetivo desta conversão é oferecer aos usuários as facilidades do modelo NCM e os recursos do sistema HyperProp para manipular e apresentar documentos SMIL. O HyperProp é um sistema para tratamento de documentos multimídia baseado no modelo NCM.

As facilidades do modelo NCM que serão acrescidas aos documentos SMIL a partir da conversão já foram discutidas no capítulo anterior. Resumindo, elas são as seguintes:

- Separação da especificação da estrutura lógica do documento de sua estrutura de apresentação;
- Especificação de relacionamentos mais complexos entre os componentes do documento;

- Especificação de mudanças de comportamento durante a apresentação de um documento;
- Maior flexibilidade na especificação temporal da apresentação dos componentes;
- Especificação do layout temporal e espacial da apresentação de forma separada do componente;
- Reutilização de estruturas de apresentação em um mesmo documento ou em documentos diferentes;
- Definição de elos no escopo da composição.

Os recursos do sistema HyperProp que poderão ser utilizados para manipular os documentos SMIL a partir de sua conversão para o NCM são os seguintes:

- Suporte a controle de versões e trabalho cooperativo, mantendo o histórico de documentos, realizando propagação automática de versões e fornecendo mecanismos de notificação [SSRM99].
- Utilização da ferramenta de autoria do sistema HyperProp, que oferece uma interface gráfica de alto nível para a criação e a edição de documentos, evitando a necessidade de especificá-los usando uma linguagem declarativa.
- Utilização do formatador HyperProp para apresentar documentos SMIL. O formatador é o elemento do sistema responsável por controlar a apresentação dos documentos. A versão corrente do formatador HyperProp não implementa, mas prevê, suporte para ajustes nas apresentações, a fim de garantir uma melhor qualidade da exibição de acordo com a plataforma.

Além dos benefícios mencionados, a integração NCM/SMIL sugere a definição de uma extensão da linguagem SMIL que ofereça pelo menos parte da funcionalidade do NCM, e ao mesmo tempo possa ser utilizada de uma forma simples, de acordo com a proposta do SMIL. Esta linguagem estendida está descrita em [AnSo00].

Durante o processo de mapeamento dos elementos da linguagem SMIL para o NCM, verificou-se a utilidade de estender a estrutura de classes do NCM para incorporar alguns elementos da linguagem, além de acrescentar novos atributos em classes já existentes. Essas extensões permitem simplificar a autoria de documentos NCM que apresentam relações de sincronismo temporal e espacial.

As Seções 3.1 a 3.6 apresentam o mapeamento dos diversos elementos da linguagem SMIL para o modelo NCM. Em seguida, a Seção 3.7 apresenta um exemplo de conversão de um documento SMIL para o modelo NCM. A implementação do conversor está discutida na Seção 3.8. Por fim, a Seção 3.9 apresenta as extensões do modelo NCM que foram propostas e implementadas neste trabalho.

3.1 Objetos de mídia e âncoras

Os objetos de mídia declarados no documento SMIL são convertidos no NCM em nós de conteúdo e descritores. Um nó de conteúdo é criado a partir dos atributos *tipo* e *src* declarados no objeto de mídia. Sua classe corresponde ao tipo do objeto de mídia (texto, imagem, etc.), e o seu atributo *conteúdo* recebe a referência (URI) contida no atributo *src* do objeto de mídia. Dois ou mais objetos de mídia distintos com os mesmos valores para os atributos *tipo* e *src* correspondem ao mesmo nó de conteúdo NCM, inserido em perspectivas diferentes, visto que o modelo suporta a reutilização de objetos. As informações armazenadas nos descritores serão tratadas na Seção 3.2.

Considere, por exemplo, um elemento de vídeo declarado da seguinte forma:

```
<video id="V1"/>
```

Este elemento é convertido em um nó de conteúdo de vídeo do modelo NCM, com o valor "V1" em seu identificador.

Para cada âncora definida em um objeto de mídia, uma âncora NCM correspondente é criada e inserida na lista de âncoras do nó. Se o objeto de mídia possuir atributos de corte (*clip-begin* e *clip-end*), uma âncora adicional definindo essa sub-região do conteúdo do nó é criada e inserida na lista de âncoras do nó. Nesse caso, todos os relacionamentos de sincronização envolvendo este nó farão referência a esta âncora, ao invés da âncora λ do nó (ver Capítulo 2). Para os exemplos apresentados neste capítulo, sem perda de generalidade, não serão considerados objetos de mídia contendo atributos de corte.

3.2 Layout espacial, duração dos objetos e número de repetições

Conforme apresentado no Capítulo 2, objetos da classe descritor do NCM são responsáveis por conter informações que descrevem como um nó deve ser apresentado no tempo e no espaço. Dessa forma, para cada objeto de mídia, é criado um descritor para guardar, entre outras coisas, as informações de layout espacial, duração de apresentação e número de repetições. Esse descritor é associado ao nó de conteúdo gerado a partir da conversão do objeto de mídia.

Dada a região de exibição do objeto de mídia, as informações de localização espacial, dimensões e cor de fundo são extraídas do layout do documento SMIL e armazenadas no descritor. Para armazenar a informação de duração e número de repetições do objeto, é criada uma entrada na lista de descrição de eventos do descritor (ver Capítulo 2), associada à âncora λ do nó. O tipo do evento é definido como apresentação. O parâmetro repetições (*rep*) recebe o valor do atributo repetições especificado no objeto de mídia (se estiver especificado). A duração do evento recebe um único valor, que é a duração especificada para o objeto de mídia, uma vez que o SMIL não provê flexibilidade na declaração das durações, conforme mencionado no Capítulo 2. Após a conversão, o valor da duração do evento poderá ser alterado para especificar uma função de custo mais elaborada, introduzindo esta facilidade em um documento SMIL.

Por exemplo, considere a seguinte declaração de um elemento SMIL com duração explícita de 5 segundos e número de repetições igual a 2:

```
<audio id=″A1″ dur=″5s″ repeat=″2″/>
```

Para converter este elemento para o modelo NCM, deve ser incluída na coleção de descrições de eventos do descritor associado ao nó de conteúdo de áudio a seguinte descrição de evento:

<α, *tipo*, *dur*, *rep*, *oper*, *hab*>

α = 1 (posição da âncora λ na lista ordenada de âncoras do nó de áudio);

tipo = apresentação;

dur = 5s;

rep = 2;

oper = nulo;

hab = falso.

Descritores também são criados para os nós de contexto de usuário NCM. Estes nós serão utilizados para representar as composições paralelas e seqüenciais SMIL, conforme será visto nas Seções 3.3 e 3.4. Nesse caso, os descritores serão importantes para guardar a duração, o número de repetições e alguns atributos adicionais.

Embora a linguagem SMIL permita a definição das características espaciais da apresentação de um componente em um elemento separado (região), as características temporais da apresentação, como início, duração, término e número de repetições, são embutidas na própria definição do objeto. Além disso, SMIL não permite que um objeto seja associado a diferentes especificações de apresentação (temporais e espaciais), dependendo da navegação realizada para atingir o objeto, ou mesmo de onde o objeto está incluído na estrutura de composições. Essas facilidades poderão ser introduzidas em um documento SMIL após sua conversão para o NCM.

3.3 Composições paralelas

Uma composição paralela SMIL é representada no NCM através de um nó de contexto de usuário contendo outros nós, de conteúdo ou de contexto de usuário, que correspondem aos componentes da composição paralela SMIL. O nó de contexto de usuário contém ainda elos para expressar a sincronização entre seus componentes. A estrutura genérica do nó de contexto de usuário NCM que representa uma composição paralela SMIL está apresentada na Figura 5, enquanto seus elos são descritos na Tabela 1.

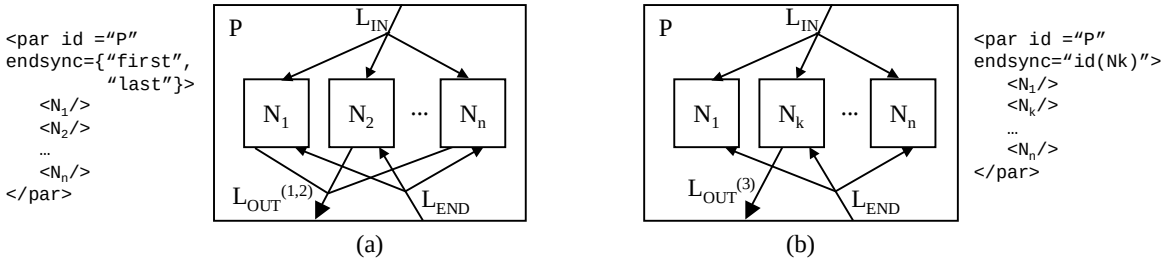

Figura 5 – Representação de composições paralelas SMIL no NCM. (a) Composições que terminam pelo primeiro ou último componente. (b) Composições que terminam por um componente específico.

Elo	**Eventos de Origem**	**Eventos de Destino**	**Ponto de Encontro**
L_{IN}	E_P: <(P), λ, *apres*, d_P>	E_1: <(N_1), λ, *apres*, d_1> E_2: <(N_2), λ, *apres*, d_2> … E_n: <(N_n), λ, *apres*, d_n>	***Condição***: <st(E_P)=preparado, st(E_P)=ocorrendo> ***Ação***: inicia(E_1) | inicia(E_2) | … | inicia(E_n)
$L_{OUT}^{(1)}$	E_1: <(N_1), λ, *apres*, d_1> E_2: <(N_2), λ, *apres*, d_2> … E_n: <(N_n), λ, *apres*, d_n>	E_P: <(P), λ, *apres*, d_P>	***Condição***: (<st(E_1)=ocorrendo, st(E_1)=preparado> ∨ <st(E_2)=ocorrendo, st(E_2)=preparado> ∨ … ∨ <st(E_n)=ocorrendo, st(E_n)=preparado>) ∧ (<verd, rep(E_1)=0> ∧ <verd, rep(E_2)=0> ∧ … ∧ <verd, rep(E_n)=0>) ***Ação***: termina(E_P)
$L_{OUT}^{(2)}$	E_1: <(N_1), λ, *apres*, d_1> E_2: <(N_2), λ, *apres*, d_2> … E_n: <(N_n), λ, *apres*, d_n>	E_P: <(P), λ, *apres*, d_P>	***Condição***: (<st(E_1)=ocorrendo, st(E_1)=preparado> ∧ <verd, rep(E_1)=0>) ∨ (<st(E_2)=ocorrendo, st(E_2)=preparado> ∧ <verd, rep(E_2)=0>) ∨ … ∨ (<st(E_n)=ocorrendo, st(E_n)=preparado> ∧ <verd, rep(E_n)=0>) ***Ação***: termina(E_P)
$L_{OUT}^{(3)}$	E_k: <(N_k), λ, *apres*, d_k>	E_P: <(P), λ, *apres*, d_P>	***Condição***: <st(E_k)=ocorrendo, st(E_k)=preparado> ∧ <verd, rep(E_k)=0> ***Ação***: termina(E_P)
L_{END}	E_P: <(P), λ, *apres*, d_P>	E_1: <(N_1), λ, *apres*, d_1> E_2: <(N_2), λ, *apres*, d_2> … E_n: <(N_n), λ, *apres*, d_n>	***Condição***: <st(E_P)=ocorrendo, st(E_P)=preparado> ∨ <st(E_P)=ocorrendo, st(E_P)=abortado> ***Ação***: aborta(E_1) | aborta(E_2) | … | aborta(E_n)

apres = apresentação
verd = verdade
(1) composições encerradas pelo último componente (*endsync* = "last" – default)
(2) composições encerradas pelo primeiro componente (*endsync* = "first")
(3) composições encerradas por um componente N_k específico (*endsync* = "id(N_k)")

Tabela 1 – Especificação dos elos em composições paralelas. Na condição dos elos, foram adotadas as seguintes convenções de representação: st(*E*) avalia o estado do evento *E*, e rep(*E*) avalia o valor do atributo repetições do evento *E*.

O elo L_{IN} é responsável pelo início da exibição paralela dos nós contidos no nó de contexto de usuário *P* assim que a exibição de *P* é iniciada. De acordo com o apresentado no Capítulo 2, a ação *inicia* do ponto de encontro de L_{IN} também inicia o atributo repetições do evento com o valor especificado no descritor do nó.

O elo $L_{OUT}^{(1,2,3)}$ é responsável pela finalização da exibição de *P*. Este elo verifica se os eventos de apresentação dos nós componentes já ocorreram o número de vezes especificado, comparando o seu atributo repetições com o valor zero. O nó contido em *P* responsável pela finalização de sua exibição depende do atributo *endsync* da composição paralela SMIL. Se a composição termina pelo último componente (elo $L_{OUT}^{(1)}$), a cada término de exibição de um nó contido em *P*, o ponto de encontro do elo verifica se todos os nós já terminaram as suas exibições (incluindo possíveis repetições). Em caso positivo, a exibição de *P* é finalizada. Se a composição termina pelo primeiro componente (elo $L_{OUT}^{(2)}$), o evento de apresentação de *P* é finalizado qualquer que seja o nó a terminar primeiro as suas exibições. Por fim, se a composição termina por um componente específico (elo $L_{OUT}^{(3)}$), o elo testa apenas o evento de apresentação deste componente.

Quando *P* deixa o estado *ocorrendo*, ele deve abortar a exibição de cada nó que ele contém, caso ela ainda esteja ocorrendo. Essa ação é desempenhada pelo elo L_{END}, que é disparado quando o estado do evento de apresentação de *P* passa de *ocorrendo* para *preparado*, ou de *ocorrendo* para *abortado*. A primeira transição pode ocorrer como conseqüência da ação *termina* de um elo (por exemplo, $L_{OUT}^{(1,2,3)}$) ou pelo término da duração especificada no descritor de *P*, caso seja definida uma duração explícita para a composição paralela. A segunda transição ocorre como conseqüência da ação *aborta* sobre o evento de apresentação de *P*, causada por um elo contido em uma outra composição na qual *P* está contido.

Em geral, os componentes de uma composição paralela SMIL iniciam suas exibições simultaneamente. Entretanto, o início de um componente pode ser definido explicitamente em relação à composição paralela. Supondo que um componente N_k da composição paralela apresentada na Figura 5 tenha um início explícito de *t* unidades de tempo em relação à composição paralela como um todo, um novo elo $L_{IN,k}$ é criado com um ponto terminal de origem definido em *P* e um ponto terminal de destino definido em N_k, de modo a garantir a espera do intervalo especificado antes de iniciar a apresentação de N_k. Além

disso, o elo L_{IN} é modificado para não mais iniciar a apresentação de N_k, mas somente prepará-la. O motivo da preparação de N_k é a necessidade de iniciar o atributo repetições do evento, para que, caso a composição paralela seja terminada pelo último componente, o elo $L_{OUT}^{(1)}$ não seja disparado sem que N_k tenha iniciado sua apresentação. A Tabela 2 apresenta as descrições do elo modificado (L_{IN}) e do novo elo criado ($L_{IN,k}$).

Elo	Eventos de Origem	Eventos de Destino	Ponto de Encontro
$\mathbf{L_{IN}}$	E_P: <(P), λ, *apres*, d_P>	E_1: <(N_1), λ, *apres*, d_1> … E_k: <(N_k), λ, *apres*, d_k> … E_n: <(N_n), λ, *apres*, d_n>	***Condição***: <st(E_P)=preparado, st(E_P)=ocorrendo> ***Ação***: inicia(E_1) | inicia (E_2) | …| prepara(E_k) | …| inicia(E_n)
$\mathbf{L_{IN,k}}$	E_P: <(P), λ, *apres*, d_P>	E_k: <(N_k), λ, *apres*, d_k>	***Condição***: (<st(E_P)=preparado, st(E_P)=ocorrendo> ⊕ t) ∧ <verd, st(E_P)=ocorrendo> ***Ação***: inicia(E_k)
$\mathbf{L_{m,k}}$	E_m: <(N_m), λ, *apres*, d_m> E_P: <(P), λ, *apres*, d_P>	E_k: <(N_k), λ, *apres*, d_k>	***Condição***: (<st(E_m)=preparado, st(E_m)=ocorrendo> ⊕ t) ∧ <verd, st(E_P)=ocorrendo> ***Ação***: inicia(E_k)

Tabela 2 – Especificação dos elos para início explícito em composições paralelas

O elo $L_{IN,k}$ possui uma condição que verifica se o evento de apresentação de *P* permanece no estado ocorrendo após o tempo de espera especificado. O estado pode ter sido alterado por três motivos: o término da apresentação de *P* era determinado por um nó, diferente do último, que encerrou sua exibição antes do tempo de espera especificado; a apresentação de *P* terminou como conseqüência da definição de uma duração explícita para a mesma; ou foi executada uma ação aborta/termina sobre o evento de apresentação de *P*.

O início explícito de um componente também pode ser definido em relação a um outro nó da composição. Nesse caso, o novo elo criado teria o ponto terminal de origem definido não mais na composição, mas sim no nó em relação ao qual o tempo para início explícito deve ser aguardado. A Tabela 2 descreve o novo elo ($L_{m,k}$) criado para expressar esse início explícito de um nó N_k em relação a um outro nó N_m contido na mesma composição. Pelo mesmo motivo anterior, a condição do ponto de encontro verifica se o evento de apresentação de *P* permanece no estado *ocorrendo* após o tempo de espera especificado.

3.4 Composições seqüenciais

Da mesma forma que a composição paralela, uma composição seqüencial SMIL é representada no NCM através de um nó de contexto de usuário que contém outros nós, de conteúdo ou de contexto de usuário, que correspondem aos componentes da composição seqüencial SMIL, e elos para expressar a sincronização entre seus componentes. O corpo do documento SMIL (*body*) é representado desta mesma forma. A estrutura genérica do nó de contexto de usuário NCM que representa uma composição seqüencial SMIL está apresentada na Figura 6, enquanto seus elos são descritos na Tabela 3.

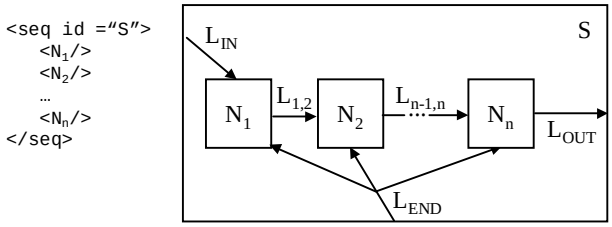

Figura 6 – Representação de composições seqüenciais SMIL no NCM.

Elo	Eventos de Origem	Eventos de Destino	Ponto de Encontro
L_{IN}	E_S: <(S), λ, *apres*, d_S>	E_1: <(N_1), λ, *apres*, d_1>	***Condição***: <st(E_S)=preparado, st(E_S)=ocorrendo> ***Ação***: inicia(E_1)
$L_{i-1,i}$ ($i \in [2,n]$)	E_{i-1}: <(N_{i-1}), λ, *apres*, d_{i-1}>	E_i: <(N_i), λ, *apres*, d_i>	***Condição***: <st(E_{i-1})=ocorrendo, st(E_{i-1})=preparado> ∧ <verd, rep(E_{i-1})=0> ***Ação***: inicia(E_i)
L_{OUT}	E_n: <(N_n), λ, *apres*, d_n>	E_S: <(S), λ, *apres*, d_S>	***Condição***: <st(E_n)=ocorrendo, st(E_n)=preparado> ∧ <verd, rep(E_n)=0> ***Ação***: termina(E_S)
L_{END}	E_S: <(S), λ, *apres*, d_S>	E_1: <(N_1), λ, *apres*, d_1> E_2: <(N_2), λ, *apres*, d_2> … E_n: <(N_n), λ, *apres*, d_n>	***Condição***: <st(E_S)=ocorrendo, st(E_S)=preparado> ∨ <st(E_S)=ocorrendo, st(E_S)=abortado> ***Ação***: aborta(E_1) | aborta(E_2) | …| aborta(E_n)

Tabela 3 – Especificação dos elos em composições seqüenciais

O elo L_{IN} inicia da exibição do nó correspondente ao primeiro componente da composição seqüencial SMIL, assim que a apresentação do nó de contexto de usuário *S* é iniciada. O início da exibição de cada nó subseqüente é determinado pelo fim da exibição do nó anterior (considerando possíveis repetições). Estas relações são estabelecidas pelos elos

$L_{i-1,i}$, com i ∈ [2,n]. Por fim, o elo L_{OUT} termina o evento de apresentação de *S* após o fim da exibição do último nó da seqüência.

Quando o evento de apresentação de *S* deixa o estado *ocorrendo*, as exibições de todos os nós contidos na composição devem ser abortadas. Essas ações são executadas pelo elo L_{END}, e ocorrem tipicamente quando o evento de apresentação de *S* é abortado ou quando ele termina como conseqüência da definição de uma duração explícita.

O início da apresentação de um componente pode ser definido explicitamente em relação ao término do componente anterior, especificando um tempo de espera (*t*). Quando a especificação explícita é feita no primeiro componente, o elo L_{IN} é ligeiramente modificado para que o início da apresentação de *S* seja verificado com um retardo de *t* segundos. Caso contrário, sendo a especificação feita para o componente *k*, o elo $L_{k-1,k}$ é modificado para que o término da exibição do nó N_{k-1} (considerando possíveis repetições) seja verificado com um retardo de *t* segundos. Em ambos os casos, pelos mesmos motivos já discutidos na Seção 3.3, a condição deve verificar se o evento de apresentação de *S* permanece no estado ocorrendo após o tempo de espera especificado. A Tabela 4 apresenta a descrição dos elos modificados para os casos especificados.

Elo	Eventos de Origem	Eventos de Destino	Ponto de Encontro
$\mathbf{L_{IN}}$	E_S: <(S), λ, *apres*, d_S>	E_1: <(N_1), λ, *apres*, d_1>	***Condição***: (<st(E_S)=preparado, st(E_S)=ocorrendo> ⊕ t) ∧ <verd, st(E_S)=ocorrendo> ***Ação***: inicia(E_1)
$\mathbf{L_{k-1,k}}$	E_{k-1}: <(N_{k-1}), λ, *apres*, d_{k-1}> E_S: <(S), λ, *apres*, d_S>	E_k: <(N_k), λ, *apres*, d_k>	***Condição***: ((<st(E_{k-1})=ocorrendo, st(E_{k-1})=preparado> ∧ <verd, rep(E_{k-1})=0>) ⊕ t) ∧ <verd, st(E_S)=ocorrendo> ***Ação***: inicia(E_k)

Tabela 4 – Especificação dos elos para início explícito em composições seqüenciais

3.5 Elos de navegação

A linguagem SMIL permite definir elos que são ativados por interação do usuário para navegação hipermídia tradicional. Esses elos são sempre *1:1*, ou seja, com uma âncora de origem e uma âncora de destino, e neles são definidas as âncoras. Conforme apresentado

no Capítulo 2, a âncora de origem corresponde a um elemento inteiro (objeto de mídia ou composição) ou a uma sub-região do elemento (somente se for objeto de mídia), e a âncora de destino pode ser um outro elemento do mesmo documento SMIL, uma sub-região de um elemento do mesmo documento, um outro documento inteiro, um elemento contido em outro documento ou uma sub-região de um elemento contido em outro documento[4].

Os elos SMIL são convertidos em elos NCM. Se o atributo *show* do elo SMIL tiver o valor *replace*, a ação do elo NCM termina a apresentação do documento de origem antes de iniciar a apresentação do objeto de destino. Se tiver o valor *pause*, a ação suspende a apresentação do objeto de origem, que é retomada por um outro elo NCM após o fim da apresentação do objeto de destino. Se tiver o valor *new*, a ação simplesmente inicia a apresentação do objeto de destino, sem afetar a apresentação do documento de origem.

Considere, por exemplo, a seguinte declaração contida em um documento SMIL denominado "origem.smil", especificando um elo de um elemento de vídeo ("elem_orig") deste documento para um elemento ("elem_dest") do documento "destino.smil", com o atributo *show* igual a *replace*:

```
<a href="destino.smil#elem_dest" show="replace">
  <video id="elem_orig" src="origem.mpv" region="reg1"/>
</a>
```

Sejam S_1 e S_2 os nós de contexto do usuário que representam o corpo dos documentos "origem.smil" e "destino.smil", respectivamente, e sejam N_1 e N_2 os nós NCM correspondentes aos elementos SMIL de origem ("elem_orig") e de destino ("elem_dest"), respectivamente. O elo *L*, criado no NCM para representar este elo SMIL, está especificado na Tabela 5. Seu ponto terminal de origem corresponde ao evento de seleção do nó N_1, e seus pontos terminais de destino correspondem aos eventos de apresentação de S_1 e N_2. Visto que o atributo *show* é igual a *replace*, a ação do elo termina a apresentação de S_1 antes de iniciar a apresentação de N_2. As listas de nós (S_1,.., N_1) e (S_2,.., N_2)

[4] A linguagem SMIL especifica que o documento que contém a âncora de destino deve ser exibido a partir da âncora até o final do próprio documento. Entretanto, o mapeamento para o modelo NCM considera que apenas a âncora de destino especificada seja exibida em conseqüência da seleção da âncora de origem.

especificam as perspectivas de N_1 e N_2 no contexto determinado pela conversão dos documentos "origem.smil" e "destino.smil", respectivamente. O elo *L* é inserido em um terceiro nó de contexto de usuário que também contém os nós S_1 e S_2. Este nó é denominado *Contexto SMIL*.

Elo	Eventos de Origem	Eventos de Destino	Ponto de Encontro
L	E_S: <(S_1,.., N_1), λ, *sel*, d_S>	E_1: <(S_1), λ, *apres*, d_1> E_2: <(S_2,.., N_2), λ, *apres*, d_2>	***Condição***: <st(E_S)=preparado, st(E_S)=ocorrendo> ***Ação***: termina(E_1) → inicia (E_2)

sel =seleção

Tabela 5 – Especificação de um elo de navegação hipermídia tradicional entre elementos inteiros.

Caso a âncora de origem e/ou de destino seja uma sub-região de um objeto de mídia do mesmo documento ou de outro documento, a especificação do elo no modelo NCM é semelhante à apresentada, a menos da identificação da âncora, que não se refere mais à âncora λ, mas à âncora resultante do mapeamento da sub-região. Considere, por exemplo, a seguinte declaração contida no documento "origem.smil", especificando um elo de uma sub-região de um elemento de imagem ("anc_orig") deste documento para uma sub-região de um elemento ("anc_dest") do documento "destino.smil", com o atributo *show* igual a *replace*:

```
<img id="I1" src="origem.gif" region="reg1">
  <anchor id="anc_orig" href="destino.smil#anc_dest"
          coords="0,0,5,5" show="replace"/>
</img>
```

Sejam S_1 e S_2 os nós de contexto do usuário que representam o corpo dos documentos "origem.smil" e "destino.smil", α_1 e α_2 as âncoras NCM resultantes do mapeamento das âncoras SMIL de origem ("anc_orig") e de destino ("anc_dest"), e N_1 e N_2 os nós NCM correspondentes aos elementos SMIL que contêm as âncoras de origem e de destino, respectivamente. O elo *L*, criado no NCM para representar este elo SMIL, está especificado na Tabela 6. Ele é bastante semelhante ao elo especificado na Tabela 5. A única diferença é que os pontos terminais de origem e de destino especificam âncoras que não são as âncoras λ dos nós, mas sim as que correspondem às sub-regiões especificadas nos elementos SMIL de origem e de destino do elo (âncoras α_1 e α_2, respectivamente).

Elo	Eventos de Origem	Eventos de Destino	Ponto de Encontro
L	E_S: <(S_1,.., N_1), α_1, *sel*, d_S>	E_1: <(S_1), λ, *apres*, d_1> E_2: <(S_2,.., N_2), α_2, *apres*, d_2>	***Condição***: <st(E_S)=preparado, st(E_S)=ocorrendo> ***Ação***: termina(E_1) → inicia (E_2)

Tabela 6 – Especificação de um elo de navegação hipermídia tradicional entre sub-regiões de elementos.

A âncora de origem definida no elo de navegação SMIL não precisa necessariamente ser um objeto de mídia ou uma sub-região deste, podendo ser também uma composição. Nesse caso, a âncora de origem é convertida em diversas âncoras NCM, uma para cada nó de conteúdo recursivamente contido no nó de contexto de usuário resultante do mapeamento da composição SMIL. Os pontos terminais de origem do elo *L* definem eventos de seleção dessas âncoras, estando na condição do ponto de encontro uma sentença que verifica, através de uma conjunção, se qualquer um dos eventos passou do estado *preparado* para *ocorrendo*.

É importante notar que os elos SMIL não são definidos no escopo de uma composição, diferentemente dos elos NCM. Além disso, os elos NCM podem representar relacionamentos causais *m:n* ou ainda relacionamentos de restrição entre diversos tipos de eventos NCM (apresentação, seleção ou atribuição), portanto são mais genéricos e oferecem um maior poder de expressão que os simples relacionamentos 1:1 disparados somente por interação do usuário. Todas estas facilidades poderão ser adicionadas a um documento SMIL após sua conversão para o modelo NCM.

3.6 Alternativas de apresentação

Conforme apresentado no Capítulo 2, a linguagem SMIL oferece o elemento *switch*, que permite definir alternativas para a apresentação de objetos de mídia e composições, escolhidas de acordo com características definidas pelo próprio usuário ou inerentes ao sistema, tais como idioma e banda passante da rede.

O NCM não oferece diretamente este tipo de opção. No lugar dela, o modelo propõe que os usuários criem estruturas e elos virtuais com consultas (*queries*) que definem o comportamento da apresentação de acordo com o contexto [Soar00]. Entretanto, entidades virtuais e linguagens de consulta ainda estão timidamente implementadas no sistema

HyperProp, sendo previstas como trabalho futuro suas extensões. Apesar disso, é possível simular a especificação de alternativas de apresentação através de nós de contexto do usuário, elos e listas de operações das descrições de eventos.

Este trabalho propõe duas formas para converter um elemento *switch* da linguagem SMIL para o modelo NCM. A primeira forma utiliza apenas os elementos do modelo já utilizados nas outras conversões realizadas (nós de contexto de usuário, elos e descritores). Esta forma, embora mais simples, possui uma limitação, que será apresentada mais adiante. A segunda forma utiliza um elemento do modelo mais sofisticado, que é a lista de operações da especificação de iniciação do descritor, porém seu comportamento é igual ao do elemento *switch* em todos os casos.

Na primeira forma de conversão, um nó de contexto de usuário é criado contendo nós que correspondem aos componentes do elemento `switch`. Associado a este nó de contexto do usuário, há um descritor com um atributo *attr*, sobre o qual o teste deve ser feito. O atributo *attr* pode ser qualquer um dos atributos de teste definidos na linguagem SMIL (por exemplo, idioma, banda passante da rede, etc.), e seu valor é obtido a partir da especificação da plataforma de exibição. A especificação da plataforma possui atributos que representam preferências do usuário (por exemplo, idioma) ou características do sistema (por exemplo, banda passante).

A Figura 7 descreve a estrutura genérica de uma composição para representar alternativas de escolha. A Tabela 7 contém a descrição dos elos mostrados na figura. O atributo *attr* está declarado no evento de atribuição A_S.

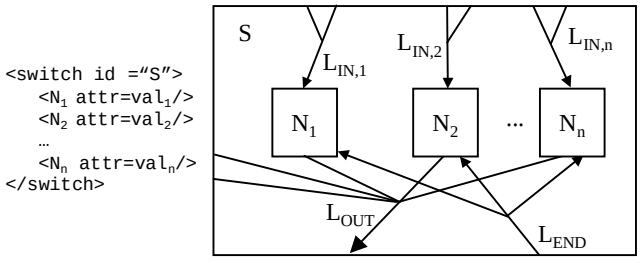

Figura 7 – Estrutura genérica de um elemento *switch* como um nó de contexto de usuário NCM.

Elo	Eventos de Origem	Eventos de Destino	Ponto de Encontro
$L_{IN,i}$ $(i \in [1,n])$	E_S: <(S), λ, *apres*, d_S> A_S: <(S), *attr*, *atrib*, d_S>	E_i: <(N_i), λ, *apres*, d_i>	***Condição***: <st(E_S)=preparado, st(E_S)=ocorrendo> ∧ <verd, at(A_S)=val_i> ***Ação***: inicia(E_i)
L_{OUT}	E_1: <(N_1), λ, *apres*, d_1> E_2: <(N_2), λ, *apres*, d_2> … E_n: <(N_n), λ, *apres*, d_n> E_S: <(S), λ, *apres*, d_S> A_S: <(S), *attr*, *atrib*, d_S>	E_S: <(S), λ, *apres*, d_S>	***Condição***: (<st(E_1)=ocorrendo, st(E_1)=preparado> ∧ <verd, rep(E_1)=0>) ∨ (<st(E_2)=ocorrendo, st(E_2)=preparado> ∧ <verd, rep(E_2)=0>) ∨ … ∨ (<st(E_n)=ocorrendo, st(E_n)=preparado> ∧ <verd, rep(E_n)=0>) ∨ (<st(E_S)=preparado, st(E_S)=ocorrendo> ∧ (¬<verd, at(A_S)=val_1> ∧ ¬<verd, at(A_S)=val_2> ∧ … ∧ ¬<verd, at(A_S)=val_n>)) ***Ação***: termina(E_S)
L_{END}	E_S: <(S), λ, *apres*, d_S>	E_1: <(N_1), λ, *apres*, d_1> E_2: <(N_2), λ, *apres*, d_2> … E_n: <(N_n), λ, *apres*, d_n>	***Condição***: <st(E_S)=ocorrendo, st(E_S)=abortado> ∨ <st(E_S)=ocorrendo, st(E_S)=preparado> ***Ação***: aborta(E_1) | aborta(E_2) |..| aborta(E_n)

atrib =atribuição

Tabela 7 – Especificação dos elos em composições *switch* (primeira forma de conversão)

Os elos $L_{IN,i}$, com i ∈ [1,n], iniciam a exibição do nó N_i cujo valor val_i é igual ao valor do atributo *attr* do descritor da composição. O elo L_{OUT} encerrará a apresentação de *S* quando a exibição do nó escolhido terminar, ou se nenhum componente satisfizer a condição. O elo L_{END} aborta a exibição do nó escolhido, caso haja algum, quando o evento de apresentação de *S* deixa o estado *ocorrendo* pela ação de um elo externo.

A especificação apresentada acima é válida para o mapeamento dos atributos de teste para os quais o nó que está sendo testado deve ser escolhido quando o valor definido na plataforma de exibição for igual ao valor de seu atributo. Este é o caso, por exemplo, do idioma escolhido pelo usuário. Entretanto, para alguns atributos de teste, o nó que está sendo testado deve ser escolhido não só quando o valor definido na plataforma de exibição for igual ao valor de seu atributo, mas também quando for maior. Este é o caso, por exemplo, da banda passante disponível na rede, onde um nó deve ser exibido se a banda passante for maior ou igual ao valor especificado em seu atributo de teste.

Uma das hipóteses para resolver este problema seria simplesmente substituir os operadores de comparação do atributo de teste nas condições dos pontos de encontro dos elos $L_{IN,i}$ e

L_{OUT} de "⊑" para " ≥". Porém, neste caso, mais de um elo $L_{IN,i}$ poderia ser satisfeito, iniciando a exibição de mais de um componente e, portanto, contrariando o que está especificado na linguagem SMIL. Esta é a limitação da primeira forma de conversão do elemento *switch*.

A segunda forma de conversão resolve a questão descrita acima através de uma programação na lista de operações da especificação de iniciação do descritor do contexto de usuário *S*. O descritor associado a *S* possui, além do atributo *attr* sobre o qual o teste deve ser feito, um atributo adicional denominado *nó atual*. Este atributo tem como valor a posição do nó que está sendo avaliado no momento, em relação aos outros nós contidos em *S*. A manipulação deste atributo é realizada na lista de operações da especificação de iniciação do descritor.

Consideremos a mesma estrutura apresentada na Figura 7 para representar alternativas de escolha no modelo NCM a partir da especificação SMIL desta mesma figura. A lista de operações da especificação de iniciação do descritor associado a *S* está apresentada na Tabela 8.

Eventos	**Operações**
E_S: <(S), λ, *apres*, d_S> $A_{S,nó\_atual}$: <(S), *nó_atual*, *atrib*, d_S> $A_{S,attr}$: <(S), *attr*, *atrib*, d_S >	***Operação 1***: *condição*: <st(E_S)=preparado, st(E_S)=ocorrendo> *ação*: atribui ($A_{S,nó\_atual}$, 1) ***Operação 2***: *condição*: <verd, at($A_{S,nó\_atual}$)=1> ∧ <verd, at($A_{S,attr}$)≠val_1> *ação*: atribui ($A_{S,nó\_atual}$, 2) ***Operação 3***: *condição*: <verd, at($A_{S,nó\_atual}$)=2> ∧ <verd, at($A_{S,attr}$)≠val_2> *ação*: atribui ($A_{S,nó\_atual}$, 3) … ***Operação n+1***: *condição*: <verd, at($A_{S,nó\_atual}$)=n> ∧ <verd, at($A_{S,attr}$)≠val_n> *ação*: atribui ($A_{S,nó\_atual}$, n+1)

atribui = atribui valor absoluto

Tabela 8 – Especificação de iniciação do descritor associado ao nó S

As operações da lista devem ser executadas seqüencialmente na ordem em que foram declaradas. Inicialmente, o atributo *nó_atual* recebe o valor 1 para avaliar o teste do primeiro nó. A cada operação, caso o teste não seja positivo, o valor do atributo *nó_atual* é incrementado para avaliar os testes dos nós seguintes, sempre respeitando a ordem em que estes nós são declarados. Ao final da execução da lista, o atributo *nó_atual* assume o valor

do primeiro nó que tenha o seu teste avaliado positivamente. Caso isso não ocorra em nenhum nó, o atributo *nó_atual* assume o valor *n+1*.

A Tabela 9 contém a descrição dos elos para a segunda forma de conversão. Os elos $L_{IN,i}$ baseiam-se no valor do atributo *nó_atual* para iniciar a exibição do nó correto. Dessa forma, no máximo um elo $L_{IN,i}$ será disparado, com *i* igual à posição do primeiro elemento contido no elemento *switch* que tem seu teste avaliado positivamente, caso exista algum. O elo L_{OUT} é responsável pela finalização da exibição de *S* após o término da exibição do nó escolhido (levando em conta possíveis repetições), ou se nenhum componente satisfizer os testes, caso em que o atributo *nó atual* possui o valor *n+1*. O elo L_{END}, assim como na primeira forma de conversão, aborta a exibição do nó escolhido, caso haja algum, quando o evento de apresentação de *S* deixa o estado *ocorrendo* pela ação de um elo externo.

Elo	**Eventos de Origem**	**Eventos de Destino**	**Ponto de Encontro**
$\mathbf{L_{IN,i}}$ **(i ∈ [1,n])**	E_S: <(S), λ, *apres*, d_S> A_S: <(S), *nó_atual*, *atrib*, d_S>	E_i: <(N_i), λ, *apres*, d_i>	***Condição***: <st(E_S)=preparado, st(E_S)=ocorrendo> ∧ <verd, at(A_S)=i> ***Ação***: inicia(E_i)
$\mathbf{L_{OUT}}$	E_1: <(N_1), λ, *apres*, d_1> E_2: <(N_2), λ, *apres*, d_2> … E_n: <(N_n), λ, *apres*, d_n> E_S: <(S), λ, *apres*, d_S> A_S: <(S), *nó_atual*, *atrib*, d_S>	E_S: <(S), λ, *apres*, d_S>	***Condição***: (<st(E_1)=ocorrendo, st(E_1)=preparado> ∧ <verd, rep(E_1)=0>) ∨ (<st(E_2)=ocorrendo, st(E_2)=preparado> ∧ <verd, rep(E_2)=0>) ∨ … ∨ (<st(E_n)=ocorrendo, st(E_n)=preparado> ∧ <verd, rep(E_n)=0>) ∨ (<st(E_S)=preparado, st(E_S)=ocorrendo> ∧ <verd, at(A_S)=n+1>) ***Ação***: termina(E_S)
$\mathbf{L_{END}}$	E_S: <(S), λ, *apres*, d_S>	E_1: <(N_1), λ, *apres*, d_1> E_2: <(N_2), λ, *apres*, d_2> … E_n: <(N_n), λ, *apres*, d_n>	***Condição***: <st(E_S)=ocorrendo, st(E_S)=abortado> ∨ <st(E_S)=ocorrendo, st(E_S)=preparado> ***Ação***: aborta(E_1) | aborta(E_2) |..| aborta(E_n)

Tabela 9 – Especificação dos elos em composições *switch* (segunda forma de conversão)

As especificações da lista de operações e dos elos apresentadas acima para a segunda forma de conversão é válida para os atributos de teste para os quais o valor definido na plataforma de exibição deve ser igual ao seu valor, como o idioma, por exemplo. Ao contrário da primeira forma de conversão, a segunda forma também pode ser utilizada para os atributos de teste para os quais o valor definido na plataforma de exibição deve maior ou igual ao valor de seu atributo, como a banda passante disponível na rede. Para isto, basta

substituir os operadores de comparação do atributo de teste na lista de operações da especificação de iniciação do descritor de *S* de "≠" para "<". A especificação dos elos permanece a mesma. Como o atributo *nó atual* do descritor possui apenas um valor ao final da execução das operações, somente um elo $L_{IN,i}$ será satisfeito, se *nó atual* for menor que *n+1*, ou nenhum elo $L_{IN,i}$ será satisfeito, se *nó atual* for igual a *n+1*. No primeiro caso, o elo $L_{IN,i}$ que for satisfeito iniciará a exibição do nó correspondente ao primeiro componente do elemento *switch* que satisfizer os testes, conforme programado na lista de operações. No segundo caso, nenhum nó será exibido, pois nenhum componente do elemento *switch* satisfaz os testes.

3.7 Exemplo de conversão

Esta seção apresenta um exemplo de conversão de um documento SMIL em um documento NCM. O documento SMIL está apresentado na Figura 8. O corpo do documento é composto por uma composição seqüencial contendo duas composições paralelas, cada uma com dois objetos de mídia. A primeira composição paralela é terminada pelo último componente, e a segunda é terminada por um componente específico (o objeto de vídeo).

```
<smil>
  <head>
    <meta name="title" content="matise"/>
    <layout>
      <root-layout id="matise" background-color="#fcedc4" width="721" height="587"/>
      <region id="m_title" left="4%" top="4%" width="47%" height="22%"/>
      <region id="V-Main"  left="52%" top="5%" width="45%" height="42%"/>
      <region id="music"/>
    </layout>
  </head>
  <body>
    <seq id="News-at">
      <par>
        <text id="Leader-Title" region="m_title" src="leader_title.html"/>
        <audio id="Leader-Music" region="music" src="logo1.aiff"/>
      </par>
      <par endsync ="id(DCAB-Video)">
        <text id="DCAB-Intro" region="m_title" src="intro.html" dur="8s"/>
        <video id=”DCAB-Video" region="V-Main" src="zoomin.mpv"/>
      </par>
    </seq>
  </body>
</smil>
```

Figura 8 – Documento SMIL a ser convertido para o NCM

O documento NCM gerado pela conversão está mostrado na Figura 9. O corpo do documento (elemento *body*) foi convertido em um nó de contexto de usuário (“example.smil”), contendo outro nó de contexto de usuário correspondente à composição seqüencial SMIL (“News-at”). Ambos os nós “example.smil” e “News-at” contêm elos para representar o comportamento seqüencial, conforme apresentado na Seção 3.4. As composições paralelas SMIL foram convertidas em nós de contexto de usuário (“$par1” e “$par2”) contendo nós de conteúdo que correspondem aos objetos de mídia SMIL, e elos para representar o comportamento paralelo, conforme descrito na Seção 3.3.

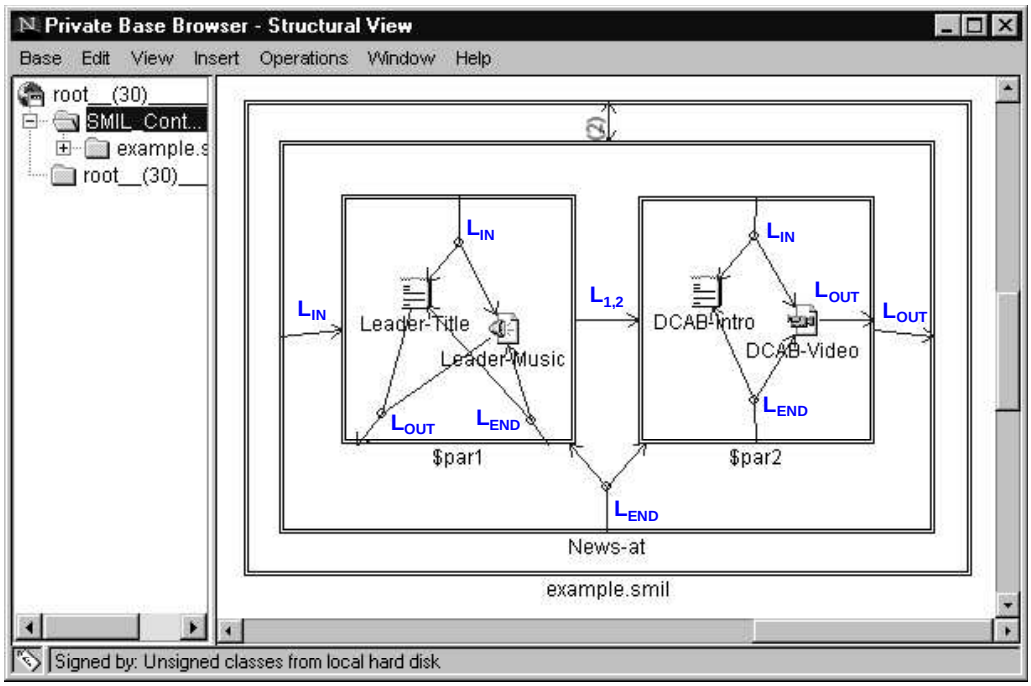

Figura 9 – Documento NCM gerado pela conversão do documento SMIL

3.8 Implementação do conversor

A janela apresentada na Figura 9 é o *browser* do cliente do sistema HyperProp. Este *browser* fornece uma visão estrutural gráfica dos documentos NCM, permitindo tanto operações de autoria quanto de navegação. A conversão de documentos SMIL em documentos NCM está totalmente integrada ao sistema, sendo ativada por um item de menu. O conversor importa automaticamente um conjunto de documentos SMIL e os

insere em um nó de contexto de usuário especial, denominado *Contexto SMIL*, que também contém os elos NCM correspondentes aos elos externos entre documentos SMIL.

Devido ao fato de o sistema HyperProp ser uma aplicação que utiliza a Internet como meio de distribuição dos documentos NCM, um dos requisitos mais importantes é a portabilidade, ou seja, o sistema deve ser capaz de ser utilizado em plataformas e sistemas operacionais diversos sem a necessidade de qualquer adaptação. Este requisito é atendido pela utilização da linguagem Java na implementação do sistema, em particular do conversor SMIL-NCM, pois esta é uma linguagem multiplataforma.

O processo de conversão é dividido em duas etapas. A primeira etapa é composta por um *parser* que lê o documento SMIL, verifica sua correção sintática e monta uma estrutura de dados que representa os elementos SMIL de modo orientado a objetos, segundo uma estrutura de classes. A segunda etapa traduz esta estrutura de dados para a estrutura de dados NCM, que representa documentos NCM. A razão para dividir a conversão nestas duas etapas é que o documento SMIL é lido apenas uma vez, e a tradução é iniciada apenas após verificar se o documento está especificado corretamente.

A Figura 10 apresenta o diagrama de classes desenvolvido para representar os documentos SMIL em um modo orientado a objetos, a partir da especificação feita em linguagem declarativa. Como pode ser observado na figura, a estrutura de classes apresenta herança múltipla, uma facilidade não suportada pela linguagem Java. Portanto, foi necessário alterar ligeiramente esta estrutura para adequá-la a uma implementação em Java, utilizando o conceito de interface. Desta forma, as classes *SMIL_CompositionChild* e *SMIL_CompositionFather* foram transformadas nas interfaces de mesmo nome, e seus atributos foram deslocados para as classes que as implementam. As classes *SMIL_GenericAnchor*, *SMIL_GenericSpatialAnchor* e *SMIL_GenericTemporalAnchor* foram transformadas nas interfaces *SMIL_AnchorInterface*, *SMIL_SpatialAnchorInterface* e *SMIL_TemporalAnchorInterface*, respectivamente, e seus atributos também foram deslocados para as classes que as implementam. Esta é uma forma aproximada de simular em Java uma herança múltipla do modelo orientado a objetos. O diagrama de classes adaptado à linguagem Java está apresentado na Figura 11, onde as interfaces estão representadas em cinza.

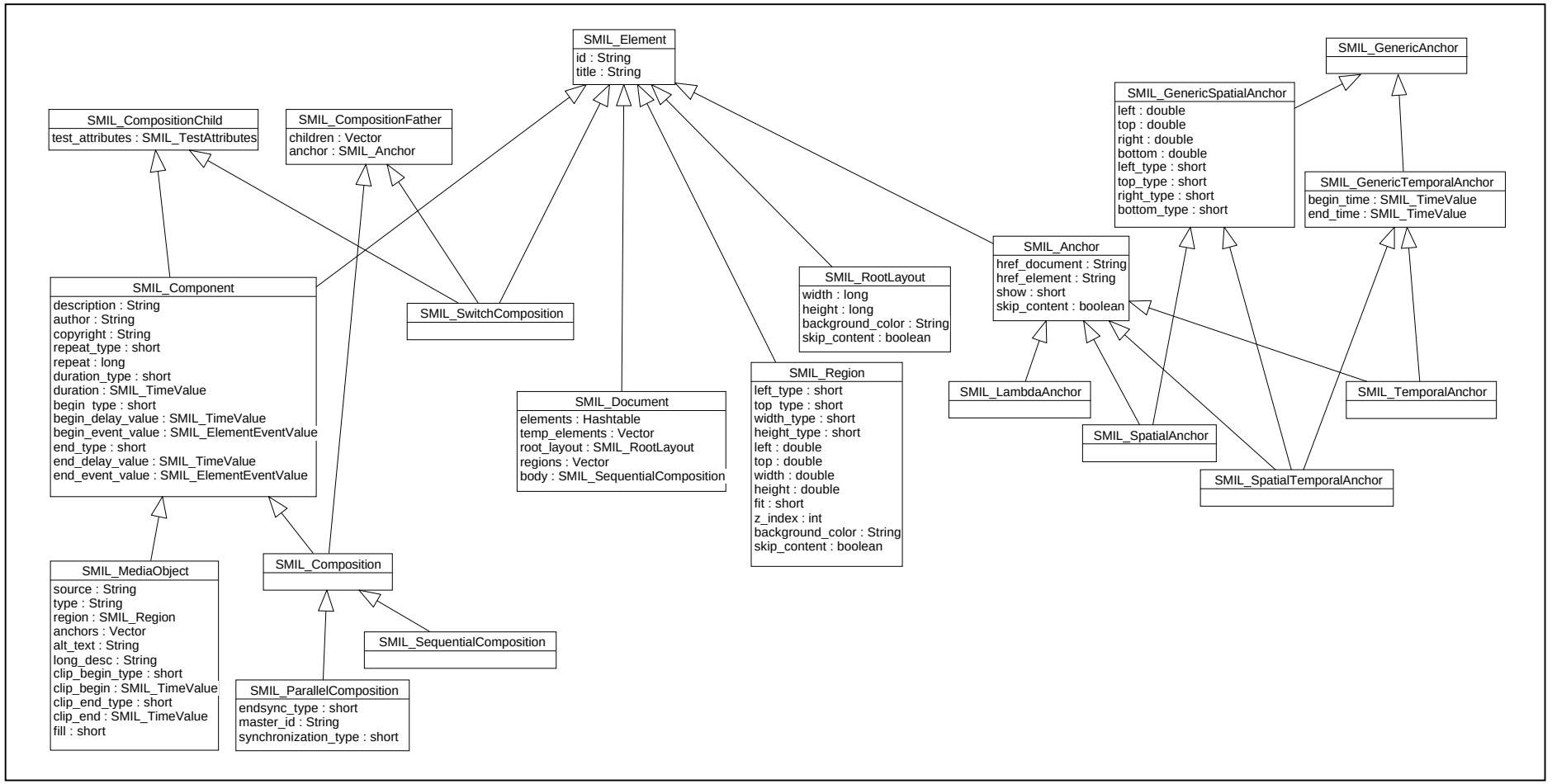

Figura 10 – Diagrama de classes dos elementos da linguagem SMIL

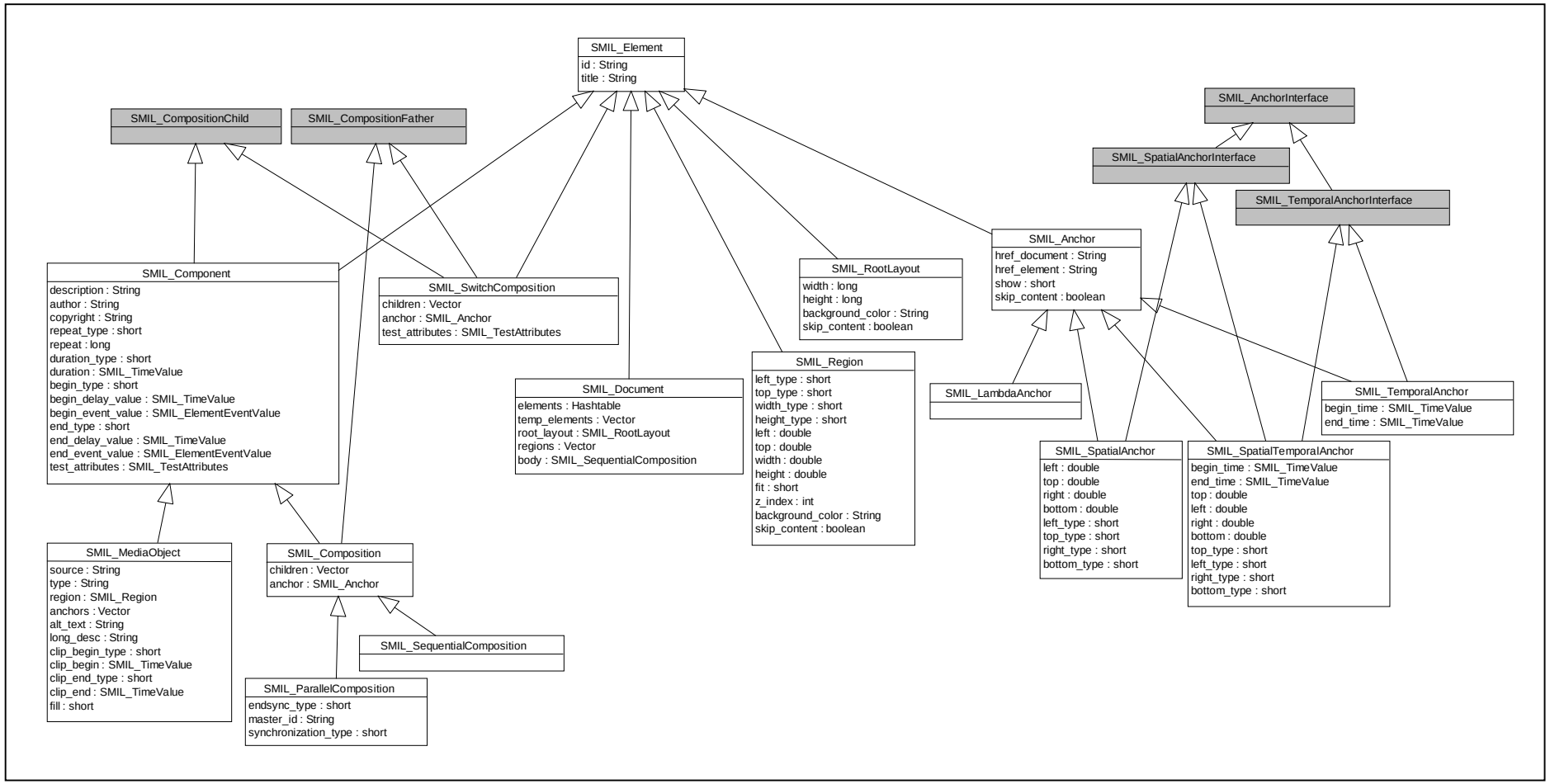

Figura 11 – Diagrama de classes dos elementos da linguagem SMIL adaptado à linguagem Java

A Figura 12 representa o diagrama de classes do conversor. O tradutor (*SMIL_NCM*) utiliza o método *ReadDocument* do *parser* (*SmilParser*) para realizar uma leitura do documento SMIL, convertendo-o para a estrutura de classes desenvolvida para a linguagem SMIL. Em seguida, realiza a conversão para a estrutura de classes do modelo NCM, utilizando como apoio as classes *PlatformSpecification*, que contém a especificação da plataforma de exibição, e *HyperLink*, que representa os hiper-elos especificados no documento SMIL.

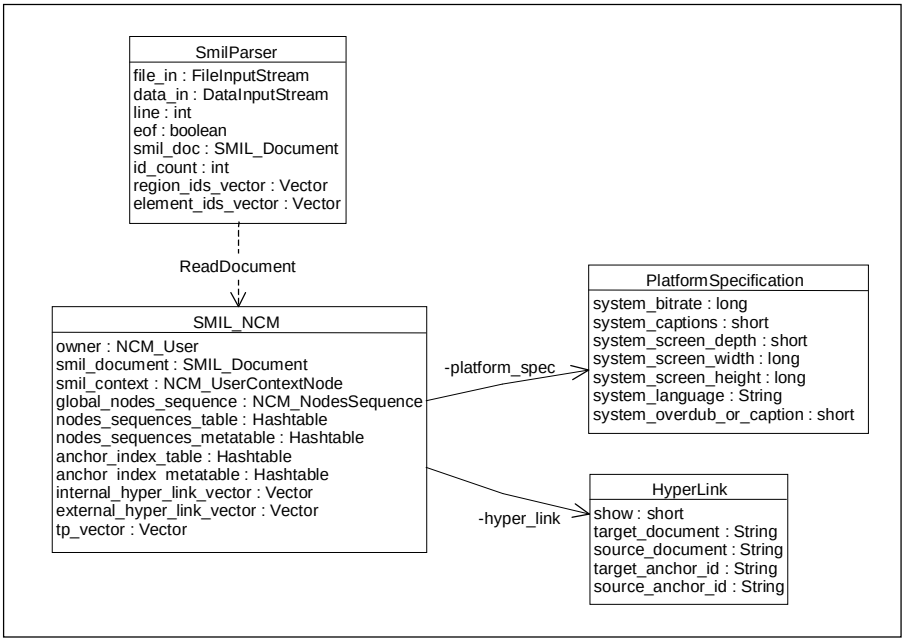

Figura 12 – Diagrama de classes do conversor SMIL-NCM

A Figura 13 representa o diagrama de estrutura do componente *parser* do conversor SMIL-NCM. Ele faz a leitura de todos os elementos especificados no documento SMIL, verificando sua correção sintática e convertendo-os para a estrutura de classes SMIL da Figura 11.

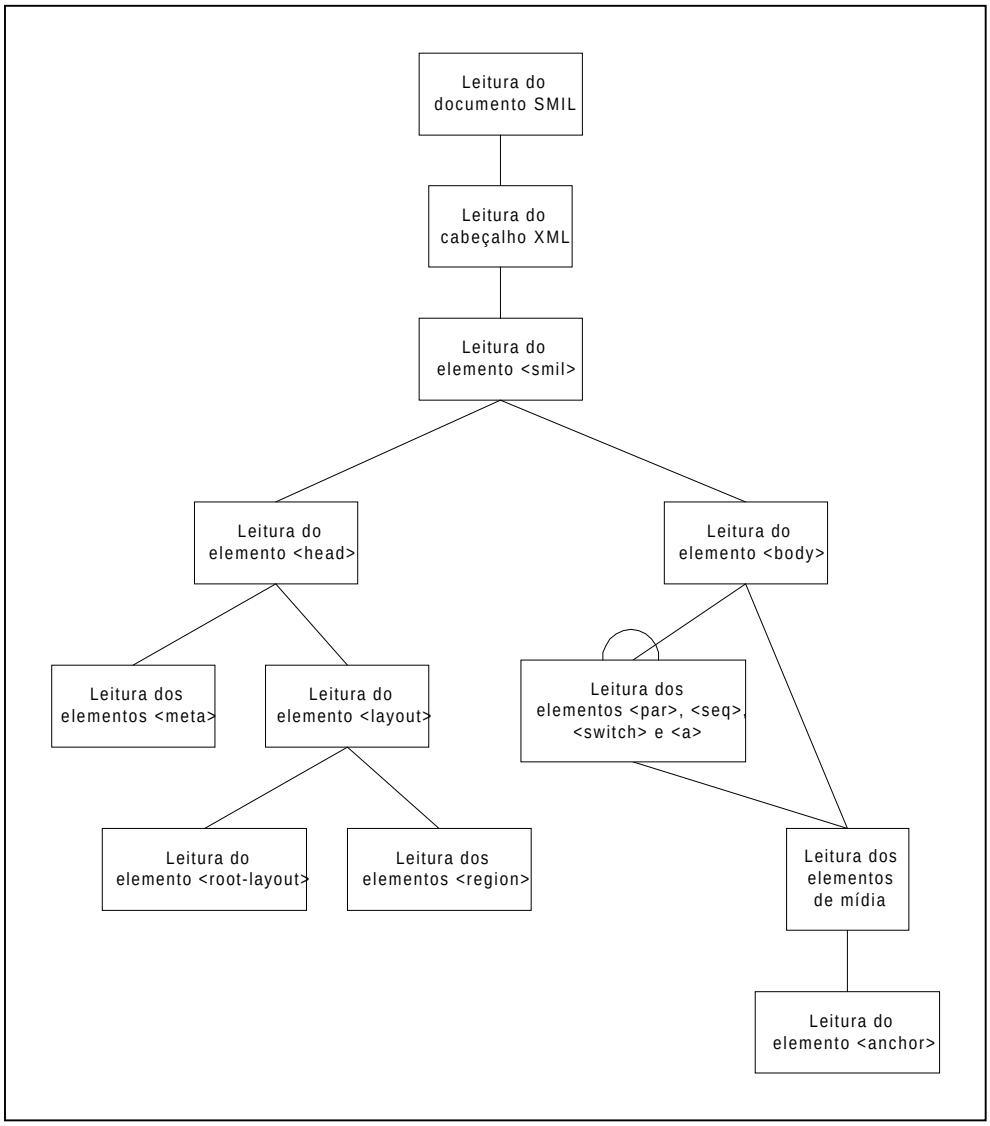

Figura 13 – Diagrama de estrutura do *parser*

A Figura 14 representa o diagrama de estrutura do componente *tradutor* do conversor SMIL-NCM. Este componente é responsável pela tradução da linguagem SMIL para o modelo NCM, convertendo cada objeto representado na estrutura de classes SMIL para a estrutura de classes do NCM. O tradutor recebe um conjunto de documentos SMIL a serem convertidos para o NCM. Para cada documento *i*, ele chama um método do *parser* para convertê-lo para a estrutura de classes SMIL, e em seguida realiza a tradução para o NCM, incluindo todos os elos de sincronização e os hiper-elos internos a cada documento. Cada documento convertido é inserido no *Contexto SMIL*, um nó de contexto de usuário NCM criado para este fim. Após terminar a

tradução de todos os documentos, o tradutor cria os hiper-elos externos (entre os documentos) especificados e os insere no *Contexto SMIL*.

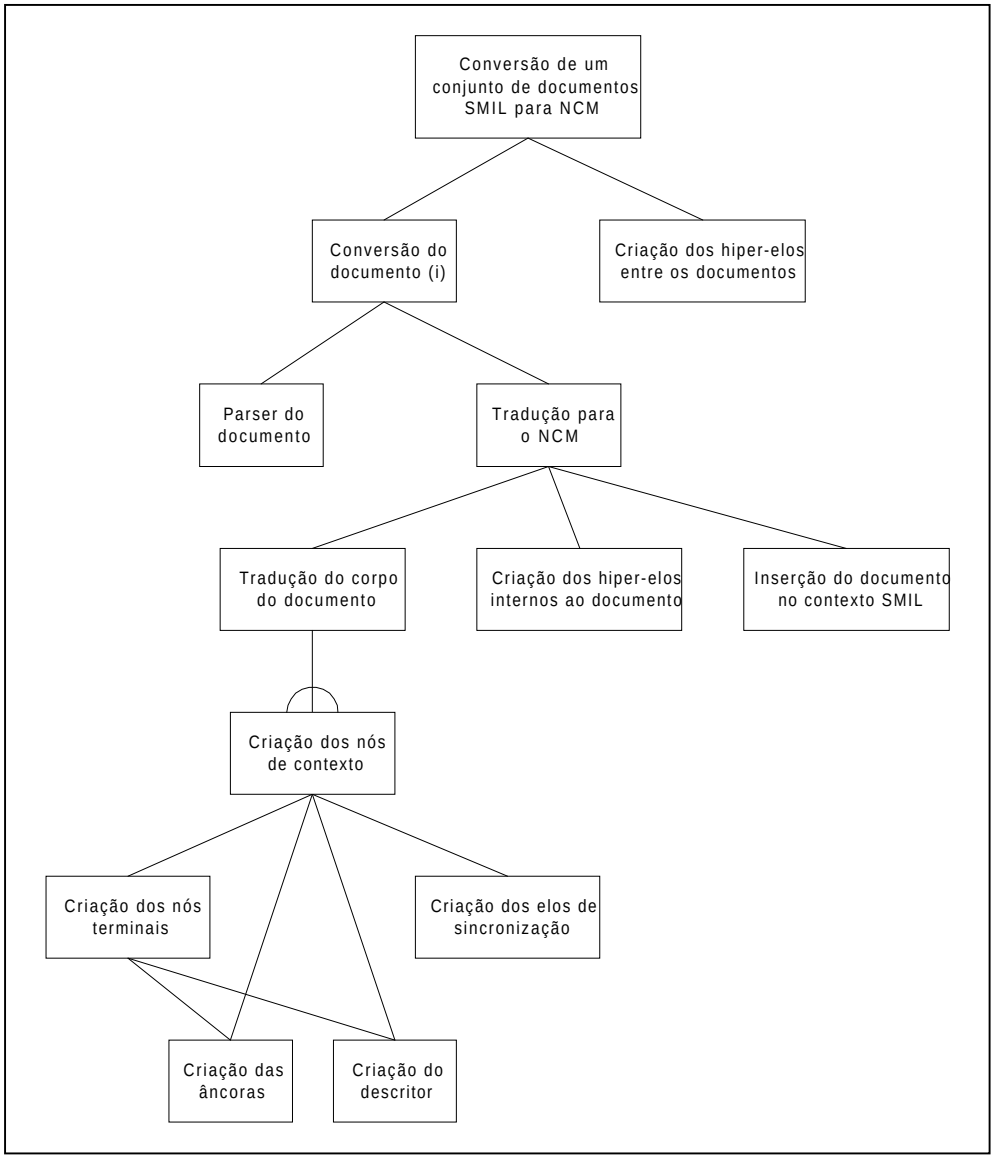

Figura 14 – Diagrama de estrutura do tradutor

Maiores detalhes a respeito do conversor de documentos SMIL para o modelo NCM podem ser obtidos em [Rodr99a].

3.9 Extensões ao Modelo NCM

Conforme comentado no início deste capítulo, durante o processo de mapeamento da linguagem SMIL para o modelo NCM, verificou-se a utilidade de estender a estrutura de classes do NCM para incorporar alguns elementos da linguagem, além de acrescentar novos atributos em classes já existentes.

Estas extensões, além de simplificar o processo de autoria de documentos no modelo NCM, permitem que um documento NCM seja convertido em um documento SMIL, dentro das limitações impostas pela linguagem. Deste modo, o ambiente de autoria implementado no sistema HyperProp pode ser utilizado para criar e editar documentos SMIL.

As extensões realizadas no modelo NCM estão apresentadas nas próximas seções.

3.9.1 Coleção de apresentação

Um *nó de contexto* do modelo NCM é um nó de composição do qual deriva, entre outras, a classe contexto de usuário. Na versão anterior do NCM (versão 2.2), um nó de contexto possuía como atributos adicionais um objeto descritor (ou o valor nulo) para cada nó nele contido.

Este trabalho propõe a substituição destes atributos por um novo atributo denominado coleção de apresentação. Uma *coleção de apresentação* contém, para cada nó contido no contexto, um grupo de conjuntos de descritores, ou o valor nulo. Um *conjunto de descritores* contém um conjunto de descritores alternativos do qual apenas um pode ser selecionado, dependendo da melhor QoS possível para uma dada plataforma. Assim, tem-se, para cada nó contido no contexto, um grupo de descritores selecionados (sem repetição), ou o valor nulo. No caso de o nó contido no contexto ser um outro nó de contexto, o grupo de descritores selecionados contém no máximo um objeto descritor.

A semântica por trás da definição do grupo de descritores selecionados para cada nó *N* contido em um nó contexto é permitir uma navegação em profundidade para *N* especificando várias exibições diferentes. Por exemplo, um nó de contexto pode possuir

um nó de conteúdo de vídeo que, dependendo da navegação feita pelo usuário, será exibido no canto ou no meio da tela, de acordo com o descritor escolhido no grupo.

3.9.2 Nós de contexto paralelo e seqüencial

Apesar de o NCM possuir um alto poder de expressão para especificar relacionamentos síncronos através de elos, a autoria de documentos que apresentam estes relacionamentos ainda pode gerar algumas dificuldades para o autor, em virtude da complexidade da definição dos elos. O processo de mapeamento dos elementos da linguagem SMIL para o NCM mostrou que os elementos de sincronização paralela (*par*) e seqüencial (*seq*) desta linguagem podem ser especificados através de nós de contexto de usuário e elos do modelo NCM, porém a representação torna-se bastante complexa. Portanto, verificou-se a utilidade de estender a estrutura de classes do NCM, criando novas composições com semânticas idênticas aos elementos de sincronização da linguagem SMIL. Desta forma, o autor poderá utilizar estas novas composições para expressar relacionamentos que antes ele só conseguia através da utilização de um conjunto complexo de elos, conforme apresentado nas Seções 3.3 e 3.4.

Estas novas classes do modelo serão utilizadas apenas nos módulos de autoria e armazenamento do sistema HyperProp. Na apresentação dos documentos, o formatador temporal traduzirá a semântica destas classes em elos do contexto de usuário e, para facilitar a formatação, só trabalhará com as classes já existentes anteriormente.

As novas classes introduzidas no modelo são denominadas *contexto paralelo* e *contexto seqüencial*, sendo ambas especializações da classe *contexto de usuário*. De forma simplificada, em um contexto paralelo os componentes são apresentados simultaneamente, enquanto em um contexto seqüencial os componentes são apresentados em série.

3.9.2.1 O contexto paralelo

A classe *contexto paralelo* possui os seguintes atributos adicionais: lista de início/término e endsync.

Um mesmo nó contido em um contexto paralelo pode dar origem a mais de uma exibição paralela de seu conteúdo, caso a coleção de apresentação do contexto paralelo especifique,

para este nó, um grupo contendo mais de um conjunto de descritores que, após a seleção pela melhor QoS possível, dará origem a um grupo de descritores selecionados contendo mais de um descritor. Isto permite, por exemplo, que uma mesma imagem seja exibida ao mesmo tempo em duas posições distintas da tela, a partir de um mesmo nó de conteúdo do tipo imagem, porém associado a dois descritores distintos.

O atributo *lista de início/término* do contexto paralelo contém, para cada entrada do grupo de conjuntos de descritores de cada um dos nós contidos no contexto paralelo, duas tuplas (início e término) <*t*, *comp*, *evt*>, indicando que ele deve iniciar/terminar sua apresentação decorrido o intervalo de tempo especificado (*t*) após o início da apresentação do contexto paralelo ou após o início ou o término de qualquer outro componente do contexto. O intervalo de tempo *t* pode ser especificado como uma função de custo, a exemplo do parâmetro *duração* do evento NCM (ver Capítulo 2). O parâmetro *comp* identifica o componente do contexto em relação ao qual o tempo é medido, ou recebe o valor nulo, no caso de o tempo ser medido em relação ao início do contexto paralelo (caso default). No primeiro caso, o parâmetro *evt* especifica o evento associado a *comp* que deve iniciar a contagem do tempo, podendo receber os valores *início*, para o início da apresentação de *comp*, ou *término*, para o término da apresentação de *comp*.

O atributo *endsync* determina o término do contexto paralelo em relação ao término de um de seus componentes. O valor desse atributo pode identificar um dos componentes, indicando que todos os demais serão finalizados quando o componente selecionado terminar. O atributo pode ainda assumir os valores *primeiro* ou *último*, para especificar o término do contexto em relação ao primeiro ou ao último componente a terminar a exibição, respectivamente. Caso esse atributo não seja especificado, o contexto paralelo termina pelo último componente.

Para efeitos de sincronização conjunta com a especificação de sincronização definida em outras classes (por exemplo, outros elos definidos no contexto paralelo ou em nós de contexto de usuário que contêm recursivamente o contexto paralelo), um contexto paralelo é equivalente a um contexto de usuário (nem paralelo nem seqüencial) que contém os mesmos nós contidos no contexto paralelo, além de elos para expressar a sincronização entre seus componentes, de acordo com as especificações descritas na Seção 3.3.

3.9.2.2 O contexto seqüencial

A classe *contexto seqüencial* possui como atributo adicional apenas a *lista de início/término*.

Um mesmo nó contido em um contexto seqüencial pode dar origem a mais de uma exibição de seu conteúdo, caso a coleção de apresentação do contexto seqüencial especifique, para este nó, um grupo contendo mais de um conjunto de descritores que, após a seleção pela melhor QoS possível, dará origem a um grupo de descritores selecionados contendo mais de um descritor. Um exemplo de aplicação deste conceito é a exibição em seqüência de uma mesma imagem em tamanhos cada vez maiores, como se ela estivesse se aproximando da tela. Neste caso, diversos descritores são associados ao mesmo nó de conteúdo do tipo imagem, cada um especificando um tamanho diferente.

Ao contrário do que ocorre no contexto paralelo, a coleção de apresentação deve ser ordenada, e é a sua ordem que especifica a ordem de apresentação dos nós contidos no contexto seqüencial. Assim, existe uma lista ordenada de conjuntos de descritores e nó associado, notando que um mesmo nó pode estar associado a mais de um conjunto de descritores. As exibições dos conteúdos dos diversos nós devem obedecer a seqüência determinada pela lista ordenada.

O atributo *lista de início/término* do contexto seqüencial contém, para cada entrada da lista ordenada de conjuntos de descritores e nó associado, duas tuplas (início e término) <*t*, *comp*, *evt*>, semelhantes às do contexto paralelo. Caso o parâmetro *comp* seja nulo (valor default) e o componente seja o primeiro a ser exibido, as tuplas indicam que ele deve iniciar/terminar sua apresentação decorrido o intervalo de tempo especificado após o início da apresentação do contexto seqüencial. Caso o parâmetro *comp* seja nulo e o componente seja um outro, diferente do primeiro, o valor da tupla determina um intervalo de tempo a partir do término do componente anterior para o início/término da apresentação do componente. Caso o parâmetro *comp* não seja nulo, ele identifica o componente do contexto em relação ao qual o tempo é medido. Neste caso, o parâmetro *evt* especifica o evento associado a *comp* que deve iniciar a contagem do tempo, podendo receber os valores *início*, para o início da apresentação de *comp*, ou *término*, para o término da apresentação de *comp*.

Para efeitos de sincronização conjunta com a especificação de sincronização definida em outras classes (por exemplo, outros elos definidos no contexto seqüencial ou em nós de contexto de usuário que contêm recursivamente o contexto seqüencial), um contexto seqüencial é equivalente a um contexto de usuário (nem paralelo nem seqüencial) que contém os mesmos nós contidos no contexto seqüencial, além de elos para expressar a sincronização entre seus componentes, de acordo com as especificações descritas na Seção 3.4.

3.9.3 Especificação da plataforma de exibição

A especificação plataforma de exibição do sistema HyperProp possui as características do ambiente onde os documentos serão apresentados. Alguns atributos da plataforma de exibição foram definidos a partir de conceitos presentes na linguagem SMIL. Estes atributos são semelhantes aos atributos da linguagem SMIL utilizados nos testes dos componentes do elemento *switch*, para escolher qual deles será exibido. Eles representam características do sistema ou preferências do usuário. São eles:

- *largura de banda*: Indica a largura de banda disponível para a transmissão das unidades de dados.
- *dimensões da tela*: Indica as dimensões em pixels do dispositivo de saída visual (por exemplo, o monitor de vídeo) onde o documento será exibido.
- *número de cores*: Indica o número de cores oferecidas pelo dispositivo de saída visual.
- *idioma*: Indica o idioma desejado pelo usuário.
- *legendas*: Indica se o usuário deseja a apresentação de legendas, caso o documento a ser exibido ofereça esta facilidade.
- *dublagem-ou-legendas*: Indica se o usuário prefere uma apresentação dublada ou legendada, caso o documento a ser exibido ofereça estas facilidades.

3.9.4 Novos atributos nos descritores e nos eventos de apresentação

Outras extensões realizadas no modelo NCM dizem respeito à especificação dos descritores e dos eventos de apresentação, definindo novos atributos que trazem maiores

informações a respeito da apresentação do objeto, a partir de conceitos presentes na linguagem SMIL.

Dois destes novos atributos são os parâmetros *repetições* do evento de apresentação e *rep* da descrição de evento. Estes parâmetros especificam quantas vezes a âncora associada ao evento deve ser exibida em seqüência, de forma semelhante ao atributo *repeat* da linguagem SMIL. O atributo *repetições* do evento de apresentação recebe como valor inicial o próprio valor do atributo *rep* da descrição do evento, caso o mesmo não seja iniciado por uma ação *inicia*[5] de um elo.

Outros atributos foram acrescentados à especificação de iniciação do descritor. São eles:

- *posição*: Seu valor é uma coordenada (*x*,*y*) que especifica a posição da extremidade superior esquerda da região de apresentação do conteúdo do nó, a exemplo dos atributos *top* e *left* da linguagem SMIL.
- *dimensões*: Seu valor é um par (*l*,*a*) que especifica a largura e a altura da região de apresentação do conteúdo do nó, a exemplo dos atributos *width* e *height* da linguagem SMIL.
- *cor de fundo*: Especifica a cor de fundo da região de apresentação, caso ela não seja totalmente ocupada pelas unidades de dados.
- *prioridade*: Possui um valor inteiro que indica a prioridade de exibição no caso de superposição espacial de regiões de apresentação, a exemplo do atributo *z-index* da linguagem SMIL.
- *ajuste*: Especifica o comportamento no caso de as dimensões intrínsecas de um objeto visual serem diferentes dos valores especificados para as dimensões da região de apresentação, a exemplo do atributo *fit* da linguagem SMIL. Ele pode assumir os seguintes valores:
  - *preencher*: Ajusta independentemente a largura e a altura do objeto, de forma a preencher toda a região. Pode haver distorção do objeto.

[5] O parâmetro opcional *n* da ação *inicia*, para iniciar o atributo *repetições* do evento de apresentação, também foi criado a partir destas extensões.

- *esconder*: Posiciona a extremidade superior esquerda do objeto na extremidade superior esquerda da região e mantém as dimensões intrínsecas do objeto. Caso alguma dimensão (largura ou altura) seja menor que a dimensão correspondente da região, o espaço restante é preenchido com a cor de fundo. Caso alguma dimensão seja maior que a dimensão correspondente da região, o objeto é cortado, ou seja, a parte do conteúdo que ultrapassa a região especificada não é exibida.
- *adequar*: Posiciona a extremidade superior esquerda do objeto na extremidade superior esquerda da região e ajusta as dimensões do objeto mantendo as proporções, até que a altura ou a largura coincidam com a dimensão correspondente da região, de forma que nenhuma parte do conteúdo é cortada. Pode sobrar um espaço da região, que é preenchido com a cor de fundo.
- *cortar*: Posiciona a extremidade superior esquerda do objeto na extremidade superior esquerda da região e ajusta as dimensões do objeto mantendo as proporções, até que a altura ou a largura coincidam com a dimensão correspondente da região, de forma que toda a região é preenchida. Parte do conteúdo do objeto pode ser cortada à direita ou abaixo.
- *rolar*: Mantém as dimensões intrínsecas do objeto. Caso alguma ou ambas as dimensões sejam maiores que as dimensões correspondentes da região, são fornecidas barras de rolagem para permitir a visualização de todo o conteúdo do objeto.

Além destes atributos, foi acrescentado ao descritor um conjunto de atributos de teste relacionados a características do sistema ou preferências do usuário, semelhantes aos definidos na linguagem SMIL. Estes atributos são comparados, em tempo de execução, aos valores dos atributos da especificação da plataforma, para decidir qual descritor do conjunto de descritores especificado para o nó será associado a ele, ou ainda se o nó será exibido ou não. Estes atributos possuem o mesmo nome dos atributos da especificação da plataforma (Seção 3.9.3), e estão descritos abaixo:

- *largura de banda*: Indica a largura de banda mínima necessária para a transmissão. O teste será positivo se a largura de banda disponível for maior ou igual ao valor deste atributo.

- *dimensões da tela*: Indica as dimensões mínimas (largura e altura), em pixels, necessárias para a apresentação do nó. O teste será positivo somente se as dimensões do dispositivo de saída visual forem maiores ou iguais a estes valores.
- *número de cores disponíveis*: Indica o número mínimo de cores necessárias para a apresentação do nó. O teste será positivo somente se o dispositivo de saída visual suportar este número de cores.
- *idioma*: O nó deverá ser exibido somente se o idioma escolhido pelo usuário for igual ao valor deste atributo.
- *legendas*: O nó deverá ser exibido somente se o valor deste atributo for igual ao valor do atributo *legendas* da plataforma de exibição, informado pelo usuário para indicar se ele deseja ou não a apresentação de legendas.
- *dublagem-ou-legendas*: O nó deverá ser exibido somente se a escolha feita pelo usuário na plataforma de exibição for igual ao valor deste atributo.

Uma das aplicações destes atributos de teste é a sua utilização em consultas especificadas em entidades virtuais do modelo NCM. Uma entidade virtual *X* é uma entidade que tem como valor de pelo menos um de seus atributos *A* uma expressão *E*, escrita em uma linguagem de consulta formalmente definida, cuja avaliação resulta em um objeto ou valor de tipo apropriado. O atributo *A* é chamado *virtual*. Quando alguma operação requer o valor de *A*, o valor resultante da avaliação de *E* é retornado.

Uma âncora virtual de um nó de contexto de usuário permite escolher um determinado nó dentre seus componentes, de acordo com os valores dos atributos de teste especificados nos descritores destes nós e com a especificação da plataforma. Do mesmo modo, um elo virtual permite escolher o nó âncora de seus pontos terminais dentre um conjunto de nós. Por exemplo, a escolha de um nó para ser exibido pode depender da largura de banda disponível. Caso ela seja baixa, poderá ser exibida uma imagem ao invés de um vídeo. Este comportamento é semelhante ao do elemento *switch* da linguagem SMIL. Maiores detalhes sobre entidades virtuais do modelo NCM podem ser obtidos em [Soar00].

Capítulo 4

Ferramentas de Exibição de Mídias com Relações de Sincronização

O capítulo anterior apresentou o processo de mapeamento da linguagem SMIL para o modelo NCM, bem como o desenvolvimento de extensões ao modelo a partir de elementos presentes nesta linguagem. Este exercício ressaltou a necessidade de se desenvolverem ferramentas de exibição de mídias que implementassem os conceitos do modelo relativos à sincronização temporal e espacial entre os objetos de mídia, de forma que os documentos especificados a partir destes conceitos pudessem ser exibidos.

Ferramentas de exibição de nós de texto em formato HTML já haviam sido implementadas no sistema HyperProp, com suporte aos eventos de interação do usuário, ou seja, a seleção de âncoras do texto. Este trabalho contribui no que diz respeito ao desenvolvimento de ferramentas para exibição de mídias contínuas (áudio e vídeo) e de imagens estáticas, suportando todas as relações de sincronização temporal e espacial entre os objetos de mídia definidas no modelo, além das próprias relações de interação do usuário. Os conceitos relativos à sincronização entre os objetos também foram aplicados às ferramentas de exibição de nós de texto que haviam sido desenvolvidas.

Este capítulo apresenta as ferramentas desenvolvidas para a exibição dos diversos tipos de mídia. Inicialmente, são dados alguns exemplos de relações de sincronização que podem ser especificadas em um documento multimídia. A seção seguinte mostra a arquitetura do formatador HyperProp, responsável pelo controle da apresentação. A última seção apresenta a estrutura e a implementação das ferramentas de exibição, tanto para mídias contínuas como para mídias discretas.

4.1 Relações de sincronização

Um dos requisitos de um sistema de tratamento de documentos multimídia é que ele seja capaz de apresentar os diversos objetos de mídia que compõem o documento, obedecendo às relações de sincronização especificadas pelo autor e respondendo às interações executadas pelo usuário durante a apresentação.

As relações de sincronização entre os objetos de um documento podem ser classificadas em dois tipos: temporais e espaciais.

As relações temporais determinam os tempos de início e fim da apresentação dos objetos de mídia ou de suas âncoras, em função de eventos gerados pelas apresentações dos outros objetos de mídia que compõem o documento (eventos síncronos) ou de interações do usuário (eventos assíncronos). Como exemplos de eventos síncronos podem ser citados o início e o final da apresentação de um objeto de mídia ou de uma âncora nele contida. Constituem eventos assíncronos a seleção de uma âncora através do *mouse* e a ativação de botões de controle da apresentação (*play*, *pause* ou *stop*).

As relações espaciais são responsáveis por definir, entre outras coisas, o tamanho e a posição das regiões a serem ocupadas pelos componentes visuais dos objetos de mídia e a prioridade de exibição em caso de superposição destas regiões.

Os exemplos a seguir ilustram casos simples de sincronização envolvendo os conceitos abordados anteriormente.

Exemplo 1: Sejam dois objetos de mídia: um vídeo V_1 e um áudio A_1. A duração intrínseca de A_1 é maior que a de V_1. Deseja-se que o áudio seja apresentado durante toda a exibição

vídeo, sem a necessidade de uma sincronização fina (o áudio pode ser, por exemplo, uma música de fundo para o vídeo). A sincronização entre estes objetos pode ser especificada da seguinte forma: o início da apresentação de V_1 deve iniciar a apresentação de A_1, e o término da apresentação de V_1 deve terminar a apresentação de A_1. A Figura 15 representa esta sincronização.

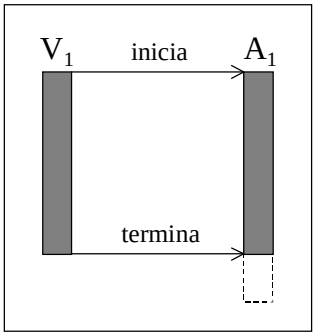

Figura 15 – Sincronização entre objetos de mídia (exemplo 1)

Exemplo 2: Consideremos os mesmos objetos de mídia do exemplo 1. Agora, porém, deseja-se que o áudio só seja apresentado durante a exibição de um determinado trecho do vídeo, identificado por uma âncora temporal α. A sincronização entre estes objetos pode ser especificada da seguinte forma: o início da apresentação de α deve iniciar a apresentação de A_1, e o término da apresentação de α deve terminar a apresentação de A_1. A Figura 16 representa esta sincronização.

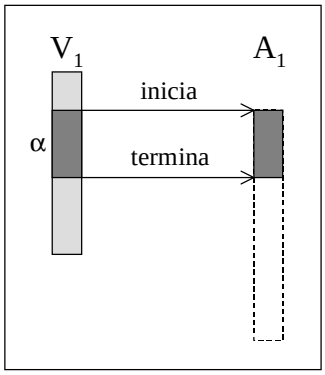

Figura 16 – Sincronização entre objetos de mídia (exemplo 2)

Exemplo 3: Sejam dois objetos de mídia, um vídeo V_1 e um áudio A_1, com a mesma duração intrínseca. Deseja-se estabelecer uma sincronização fina entre V_1 e A_1, ou seja, um

atraso em um dos fluxos de dados deve interromper a exibição do outro fluxo, até que os tempos de exibição das mídias voltem a se sincronizar. Este é o caso, por exemplo, de um vídeo de uma pessoa falando e o áudio correspondente à sua voz, em dois fluxos de dados distintos. Podemos dividir cada um dos objetos de mídia em um conjunto de *n* âncoras temporais α_i e β_i (para o vídeo e o áudio, respectivamente), cada uma correspondendo ao intervalo de tempo $[t_{i-1},t_i]$, para $i \in [1,n]$. Os tempos t_0, t_1, ... t_n são chamados pontos de sincronização. A sincronização fina pode ser especificada através das seguintes relações, para $i \in [1,n\text{-}1]$:

- O término da apresentação de α_i deve interromper a apresentação de V_1.
- O término da apresentação de β_i deve interromper a apresentação de A_1.
- Quando as apresentações de α_i e β_i houverem ambas terminado, deve-se prosseguir com as apresentações de V_1 e A_1.

A Figura 17 representa a terceira relação acima, em um dado ponto de sincronização t_i. Quanto maior o valor de *n*, mais fina é a sincronização entre os objetos de mídia.

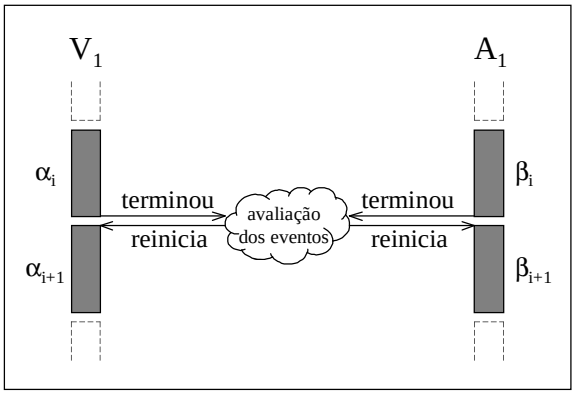

Figura 17 – Sincronização entre objetos de mídia (exemplo 3)

Exemplo 4: Seja um conjunto de *n* objetos de mídia do tipo áudio: A_1, A_2, ..., A_n. Deseja-se apresentar em seqüência os 10 primeiros segundos de cada música. Esta apresentação pode ser especificada através de uma relação de sincronização temporal entre âncoras dos objetos de mídia. Seja a âncora temporal α_i correspondente aos 10 primeiros segundos do objeto A_i, para $i \in [1,n]$. Considerando que a apresentação do documento seja iniciada pela âncora α_1, o resultado desejado pode ser obtido através da seguinte relação: o término da

apresentação de α_i inicia a apresentação de α_{i+1}, para $i \in [1,n\text{-}1]$. A Figura 18 representa esta sincronização.

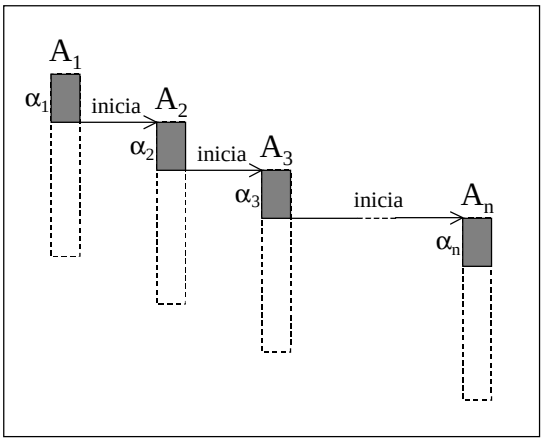

Figura 18 – Sincronização entre objetos de mídia (exemplo 4)

Exemplo 5: Sejam dois objetos de mídia: uma imagem I_1 e um texto T_1. A imagem I_1 é um mapa do Brasil, e o texto T_1 é uma página HTML sobre o estado do Rio de Janeiro. Deseja-se que, quando o usuário selecionar com o *mouse* o estado do Rio de Janeiro em I_1, seja apresentado o texto T_1 ao lado da imagem, com largura de 150 pixels e altura de 200 pixels. A sincronização entre estes objetos pode ser especificada da seguinte forma: quando o usuário selecionar em I_1 a âncora espacial β, correspondente ao estado do Rio de Janeiro, deve ser iniciada a apresentação de T_1 ao lado de I_1 e com o tamanho especificado. A Figura 19 representa esta sincronização.

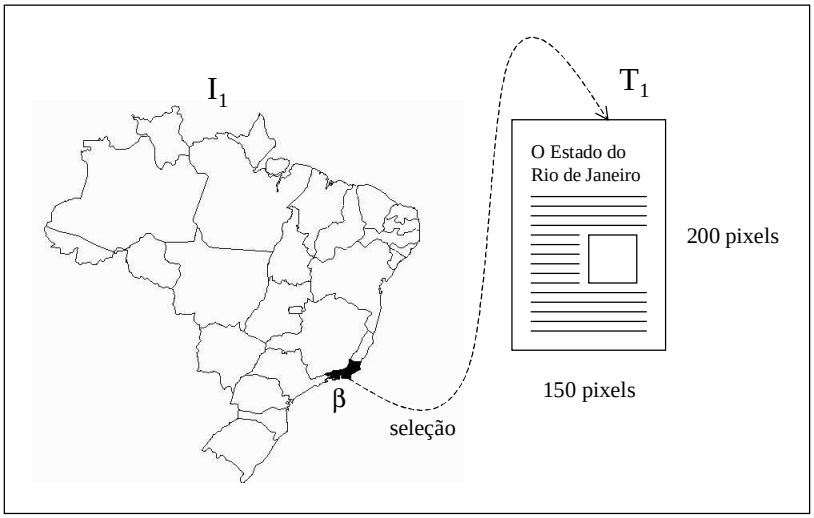

Figura 19 – Sincronização entre objetos de mídia (exemplo 5)

4.2 O formatador HyperProp

O *formatador* é o componente dos sistemas hipermídia/multimídia responsável pela apresentação dos documentos, a partir da especificação feita pelo autor e da descrição da plataforma onde eles serão exibidos.

A arquitetura do formatador temporal e espacial do sistema HyperProp [Rodr97, Rodr99b, SoMR98] está apresentada na Figura 20. Nela podemos identificar três módulos principais: o pré-compilador, o compilador e o executor. Além disso, são definidas as interfaces com o subsistema de autoria, o subsistema de armazenamento e as ferramentas de exibição.

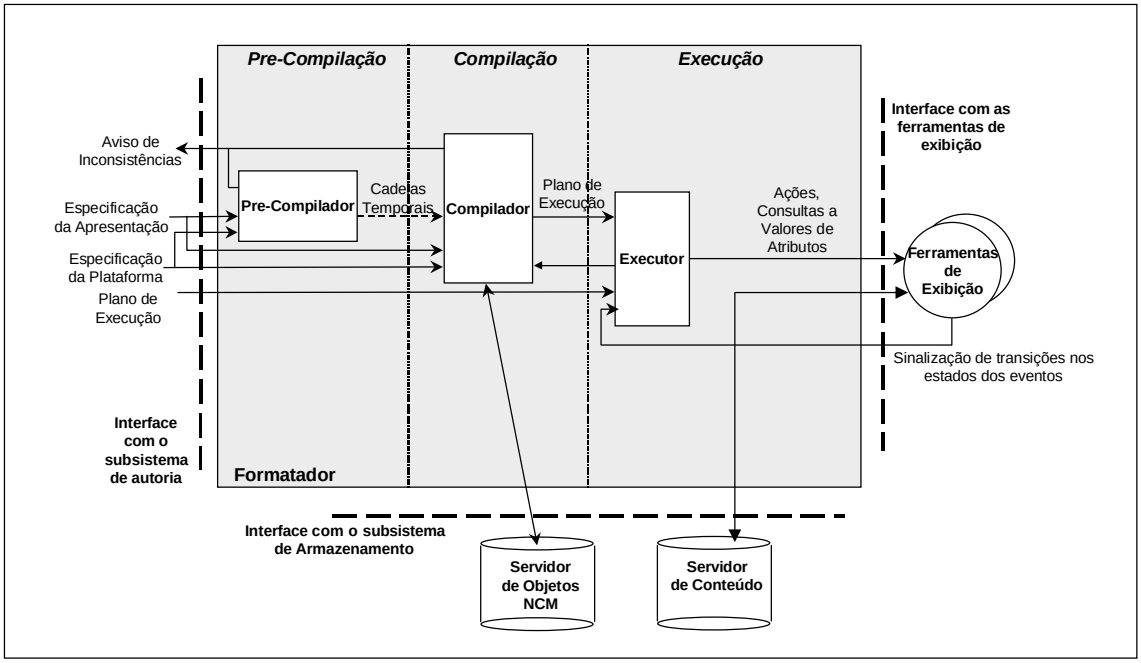

Figura 20 – Arquitetura do formatador HyperProp

O *pré-compilador* e o *compilador* têm como função converter a estrutura de dados do documento para uma estrutura de dados própria do formatador, adequada para o controle da apresentação. Nestes módulos podem ser implementados mecanismos de verificação de consistência, programação de *prefetch* do conteúdo dos objetos de mídia e ajuste das durações de exibição dos componentes. A principal diferença entre estes dois módulos é que o pré-compilador trabalha com subconjuntos do documento, verificando a ocorrência de inconsistências temporais e espaciais ainda em fase de autoria, enquanto o compilador

trabalha com a especificação completa do documento para gerar a estrutura de dados a ser usada pelo executor.

A função do *executor* é realizar o controle da apresentação. Este módulo realiza a avaliação dos elos NCM especificados no documento, de forma que as ações de controle da exibição sejam disparadas sempre que as respectivas condições forem satisfeitas.

As ações de controle da exibição, disparadas pelo executor, provocam mudanças de estado nos eventos de apresentação aos quais se aplicam, conforme apresentado no Capítulo 2. A máquina de estados dos eventos de apresentação do modelo NCM está novamente apresentada na Figura 21. O executor é responsável por manter atualizados os estados dos eventos.

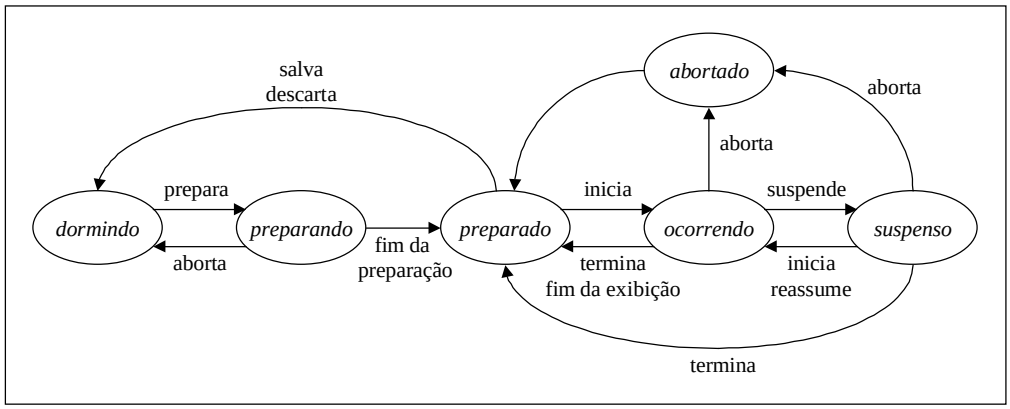

Figura 21 – Máquina de estados dos eventos de apresentação do modelo NCM

As *ferramentas de exibição* são iniciadas pelo executor sempre que for iniciada a apresentação de um novo objeto de mídia. Elas são responsáveis pela apresentação dos objetos de mídia propriamente ditos, avisando ao executor a ocorrência de eventos e executando as ações solicitadas por ele, em decorrência da avaliação dos elos. Estas ferramentas podem ser implementadas no próprio formatador, obedecendo à interface especificada por ele. Também podem ser usados como ferramentas de exibição aplicativos comerciais, como editores de texto, editores de imagem, *browsers* HTML e controladores de áudio e vídeo. Neste caso, é necessária a existência de um módulo intermediário entre a ferramenta e o formatador, de modo a tornar as interfaces compatíveis. Há ainda a possibilidade de que os métodos de exibição sejam implementados no próprio objeto de

dados ou em seu descritor. O escopo deste trabalho limita-se às ferramentas de exibição implementadas no formatador, que serão tratadas com maiores detalhes na Seção 4.3.

4.3 Ferramentas de exibição

As ferramentas de exibição, conforme apresentadas na Seção 4.2, são os elementos do sistema HyperProp responsáveis pela apresentação de cada um dos objetos de mídia que compõem o documento. Estas ferramentas podem ser classificadas de duas formas: quanto ao tipo de mídia que elas suportam (texto, imagem, áudio ou vídeo) e quanto à forma de apresentação espacial (janela ou painel).

As ferramentas do tipo *janela* provêem uma janela de exibição independente para cada objeto do documento, de modo que os botões de controle da apresentação (*play*, *pause* e *stop*) são individualizados para cada ferramenta, e o tamanho e a posição de cada uma delas podem ser alterados pelo usuário sem causar interferência nas outras. Por outro lado, as ferramentas do tipo *painel* caracterizam-se pelo fato de serem inseridas em uma mesma janela, que passa a ser única para a apresentação do documento e oferece um único componente de controle para todos os objetos de mídia. A escolha do tipo de ferramenta a ser utilizada deve ser feita na especificação da plataforma de exibição, sendo portanto independente da especificação do documento. Isto oferece ao usuário uma flexibilidade de escolha no momento da apresentação.

A Figura 22 apresenta o diagrama de classes das ferramentas de exibição que foram implementadas no formatador. A classe *HF_GenericPlayer* define uma ferramenta de exibição genérica, contendo os atributos e os métodos comuns aos diversos tipos de mídia. Como subclasses desta ferramenta genérica estão as ferramentas específicas para cada tipo de mídia. As ferramentas de exibição de mídias discretas (*HF_TextPlayer* e *HF_ImagePlayer*) derivam diretamente das ferramentas genéricas. As ferramentas de exibição de mídias contínuas (áudio e vídeo), por terem uma série de características em comum, foram representadas pela classe *HF_JMFPlayer*, de onde derivam as classes específicas para áudio (*HF_AudioPlayer*) e vídeo (*HF_VideoPlayer*).

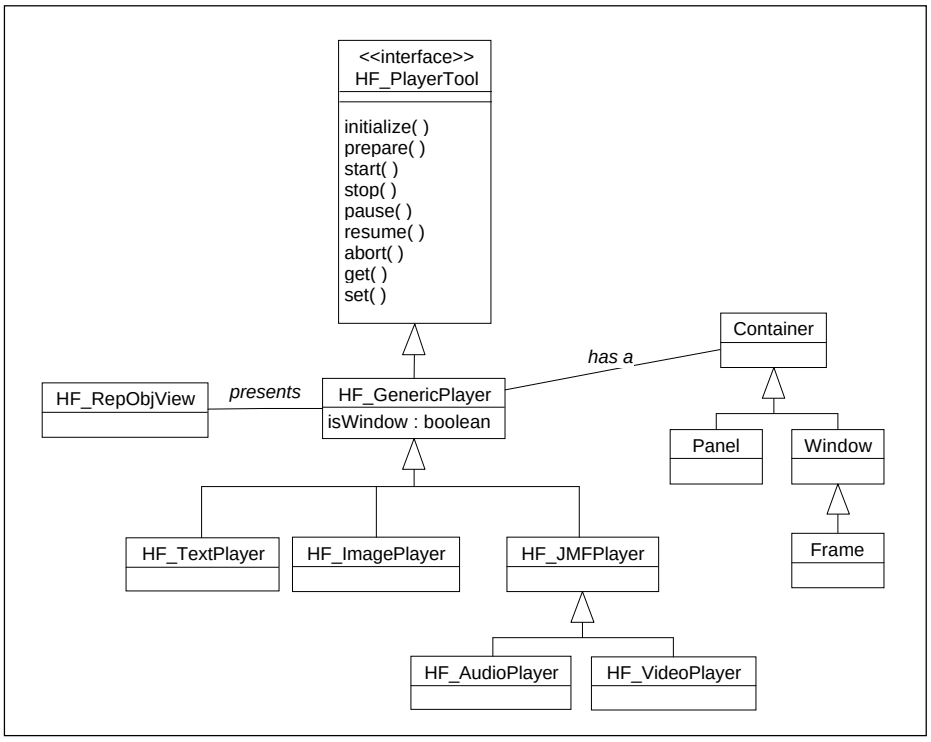

Figura 22 – Diagrama de classes das ferramentas de exibição

As ferramentas de exibição podem se comportar como janela ou painel, dependendo de um parâmetro de iniciação (*isWindow*). As que são iniciadas como janela instanciam um *Container* (pacote *java.awt*) do tipo *Frame*, onde é inserido o componente visual da mídia (se houver) e, opcionalmente, um componente de controle dedicado à apresentação daquele objeto de mídia em particular. As que são iniciadas como painel instanciam um *Container* do tipo *Panel*, contendo apenas o componente visual da mídia. Todos os painéis são inseridos em uma mesma janela, com um único componente de controle para todos os objetos de mídia da apresentação. Esta janela é definida no próprio formatador, e suas características (tamanho, posição, cor de fundo, etc.) fazem parte da especificação da plataforma. Neste trabalho, as janelas (*Frame*) e os painéis (*Panel*) serão referidos genericamente pela palavra *display*, exceto nos casos onde é necessária a diferenciação entre eles. A Figura 23 ilustra ferramentas de exibição dos tipos janela e painel.

Uma ferramenta de exibição controla a apresentação de um determinado objeto de representação do modelo NCM, em uma dada perspectiva. Estas informações estão presentes no objeto da classe *HF_RepObjectView*, que o formatador passa como parâmetro para a ferramenta ao iniciá-la.

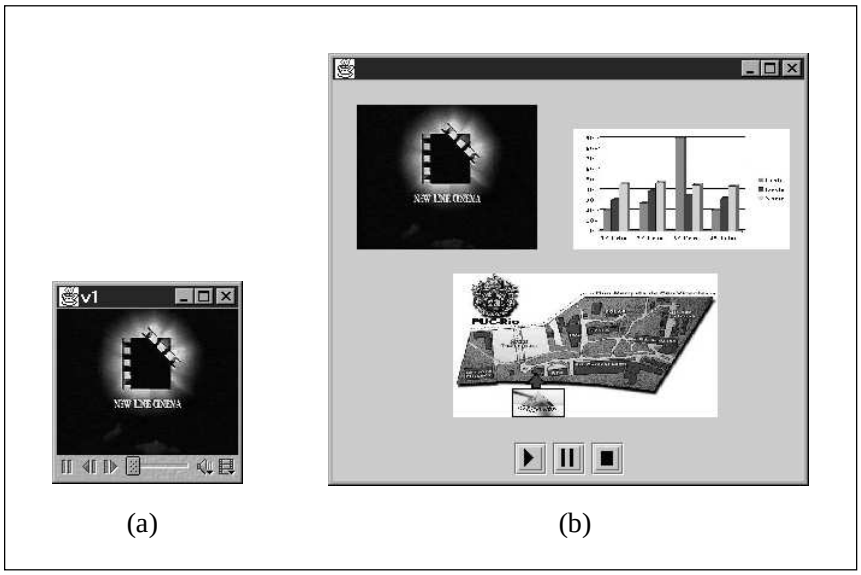

Figura 23 – (a) Ferramenta do tipo janela. (b) Ferramentas do tipo painel inseridas em uma janela de apresentação.

Uma nova ferramenta de exibição é instanciada toda vez que o executor prepara a apresentação de um novo objeto de representação. Para controlar a instanciação destas ferramentas, o executor possui um registro daquelas que são suportadas pelo formatador. Este registro é baseado em duas tabelas. A primeira determina a classe da ferramenta de exibição a partir da ferramenta especificada no descritor do nó. Caso ela não esteja especificada, uma segunda tabela determina a classe da ferramenta a partir do tipo MIME do objeto de dados.

Uma ferramenta de exibição do sistema HyperProp desempenha duas funções fundamentais: executar as ações solicitadas pelo executor sobre a apresentação e sinalizar para o executor a ocorrência de eventos durante a apresentação da mídia.

As ações a serem executadas pelas ferramentas de exibição são as próprias ações simples sobre os eventos de apresentação especificadas no modelo NCM: preparar a apresentação (*prepara*), iniciar a apresentação (*inicia*), terminar a apresentação (*termina*), suspender temporariamente uma apresentação em andamento (*suspende*), reiniciar uma apresentação suspensa (*reassume*) e cancelar a apresentação (*aborta*). Além disso, as ferramentas de exibição também devem permitir ao executor consultar e alterar seus atributos de apresentação. Estes atributos podem ser comuns a todas as ferramentas, como o tamanho e

a posição da região de apresentação[6], ou específicos para cada tipo de ferramenta, como a taxa de exibição de uma mídia de áudio ou vídeo.

Todas as ferramentas de exibição devem implementar a interface *HF_PlayerTool*, especificada pelo formatador. Esta interface define um método para iniciação (*initialize*) da ferramenta e os métodos correspondentes às ações citadas acima[7]. O método *initialize* recebe como parâmetros um objeto de representação em uma dada perspectiva, os eventos NCM associados a ele e o tipo de ferramenta (janela ou painel). Os métodos correspondentes às ações simples recebem como parâmetro o evento ao qual a ação se aplica. Além destes, foram acrescentados os métodos *get* e *set*. O primeiro recebe como parâmetro o nome de um atributo da ferramenta de exibição e retorna o seu valor, enquanto o segundo recebe o nome de um atributo e o valor a ser atribuído, e realiza a atribuição.

Os eventos a serem sinalizados pelas ferramentas de exibição correspondem às mudanças de estado dos eventos do modelo NCM[8]. São eles:

- Início da preparação da apresentação de um objeto (alocação dos recursos e busca das unidades de informação);
- Final da preparação da apresentação de um objeto;
- Início da apresentação de um objeto;
- Final da apresentação de um objeto, que pode ocorrer em três situações: a apresentação de sua última unidade de informação (por exemplo, o último quadro de um objeto de vídeo), o alcance de um tempo de término preestabelecido ou uma ação *termina* executada sobre a apresentação;

---

[6] Mesmo um objeto de áudio pode ter uma região de apresentação associada, onde são exibidos componentes de controle da apresentação e/ou alguma representação visual, como o formato da onda.

[7] Os nomes dos métodos estão em inglês para manter compatibilidade com a implementação do sistema HyperProp.

[8] Esta sentença possui duas noções de evento. Os eventos sinalizados pelas ferramentas de exibição são instantâneos, enquanto os eventos do modelo NCM possuem, em geral, uma duração finita.

- Suspensão temporária da apresentação de um objeto devido a uma ação *suspende* executada sobre a apresentação;
- Reinício da apresentação de um objeto devido a uma ação *reassume* executada sobre uma apresentação suspensa;
- Cancelamento da apresentação de um objeto devido a uma ação *aborta* executada sobre a apresentação;
- Seleção de uma âncora pelo usuário.

Todos os eventos citados acima, com exceção do último, correspondem a mudanças de estado nos eventos de apresentação do modelo NCM. A seleção de uma âncora pelo usuário corresponde a uma mudança de estado em um evento de seleção do modelo NCM.

4.3.1 Ferramentas de exibição de mídias contínuas

As ferramentas de exibição de mídias contínuas realizam a apresentação das mídias de áudio e vídeo. A sua modelagem foi feita com base no Java Media Framework. A seguir está apresentada uma breve descrição dos componentes do framework relevantes para este trabalho. A partir dos conceitos apresentados, é feito o detalhamento das ferramentas de exibição de mídias contínuas que foram desenvolvidas.

4.3.1.1 Java Media Framework

O Java Media Framework (JMF) [JMF99a, JMF99b] é uma API que permite a incorporação de mídias contínuas em aplicações e *applets* Java. A versão 1.0 da API JMF suporta a reprodução de mídias de áudio e vídeo. A versão 2.0 estende a anterior, provendo suporte para a aquisição, o armazenamento e a transmissão destas mídias, a utilização de codificadores e decodificadores personalizados e a manipulação dos dados de mídia antes de sua reprodução. A Figura 24 apresenta os principais componentes do JMF, dentro do escopo deste trabalho.

A interface *Clock* define as operações básicas de temporização e sincronização para controlar a apresentação dos dados de mídia, devendo ser implementada por todas as classes que suportam o modelo de tempo do JMF. Entre estas operações está a manipulação de atributos como a posição corrente no fluxo da mídia, o tempo de término

da exibição e a taxa de exibição. Para ter controle sobre a passagem do tempo enquanto um fluxo de mídia está sendo apresentado, um *Clock* utiliza um objeto que implementa *TimeBase* (base de tempo). Uma base de tempo provê seu tempo corrente, normalmente com base no relógio do sistema. Ferramentas de exibição capazes de informar a duração total de um fluxo de mídia devem implementar ainda a interface *Duration*.

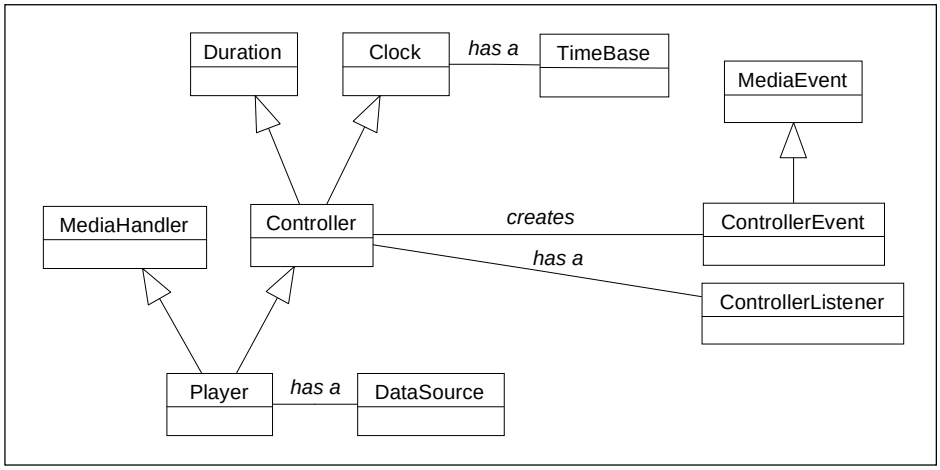

Figura 24 – Principais componentes do JMF

Durante a apresentação de um fluxo de mídia, a posição corrente no tempo é calculada da seguinte forma:

$$\texttt{MediaTime} = \texttt{MediaStartTime} + \texttt{Rate} \times (\texttt{TimeBaseTime} - \texttt{TimeBaseStartTime}),$$

onde *MediaTime* é a posição corrente no fluxo da mídia, *MediaStartTime* é a posição no fluxo de mídia onde a apresentação iniciou, *TimeBaseTime* é o tempo corrente informado pela base de tempo, *TimeBaseStartTime* é o tempo informado pela base de tempo quando a apresentação iniciou, e *Rate* é a taxa de exibição da mídia. Quando a apresentação é suspensa, o tempo da mídia (*MediaTime*) pára, enquanto a base de tempo (*TimeBaseTime*) continua a avançar. Se a apresentação for reiniciada a partir deste mesmo ponto, é feito um novo mapeamento entre o tempo da mídia e o tempo da base de tempo.

O processo de apresentação de mídias contínuas no JMF é modelado pela interface *Controller*. Um *Controller* notifica mudanças em seu estado através da sinalização de eventos. Para uma classe receber eventos gerados por um *Controller*, ela deve implementar a interface *ControllerListener* e ser registrada como seu observador.

A interface *MediaHandler* deve ser implementada por todos os objetos que obtêm e controlam o conteúdo de uma mídia fornecido por um *DataSource*. Um objeto da classe *DataSource* é responsável pela transmissão do fluxo de dados, a partir das informações de sua localização e do protocolo utilizado.

A interface *Player* estende as interfaces *Controller* e *MediaHandler*. A sua função é processar um fluxo de dados de mídia e apresentá-lo nos dispositivos de saída. Para controlar todo o processo de alocação de recursos, busca das informações e apresentação de um objeto de mídia, é definida uma máquina de estados para o *Player*, representada na Figura 25.

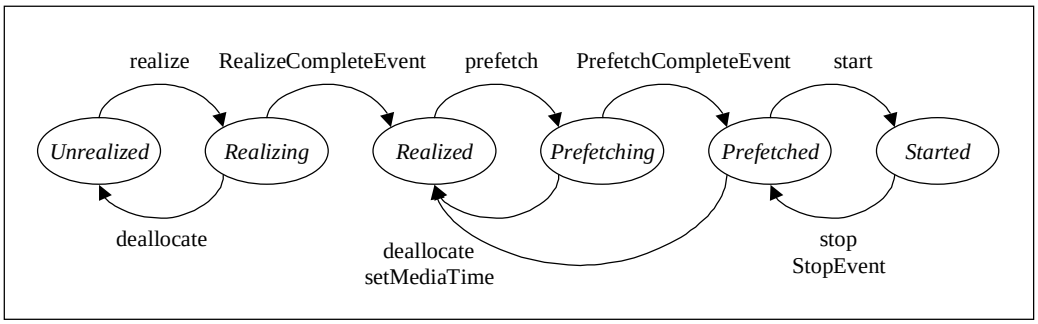

Figura 25 – Máquina de estados de um *Player*

Um novo *Player* é criado no estado *unrealized*. Quando o método *realize* é chamado, o *Player* passa para o estado *realizing*, iniciando o processo de verificação dos recursos necessários e alocação daqueles que não são de uso exclusivo. Ao fim deste processo, o *Player* passa para o estado *realized*, onde ele já possui informação sobre o tipo de mídia que irá apresentar. O método *prefetch* leva o *Player* para o estado *prefetching*, onde são iniciados os processos de carregamento dos dados de mídia e alocação dos recursos de uso exclusivo. Ao passar para o estado *prefetched*, o *Player* está pronto para iniciar a apresentação da mídia. No momento em que o método *start* é chamado, o *Player* passa para o estado *started*, quando a apresentação começa efetivamente. Um *Player* no estado *started* retorna ao estado *prefetched* quando o método *stop* é chamado, quando atinge o fim da apresentação ou quando a transmissão dos dados é interrompida.

Sempre que há mudança de estado em um *Player*, ele dispara eventos de transição para todos os objetos registrados como seus observadores, conforme explicado anteriormente.

Desta forma, quando um programa chama um método assíncrono em um *Player* (por exemplo, *realize*, *prefetch*, *start* ou *stop*), ele deve observar o evento associado para determinar quando a operação foi completada. Um *Player* também dispara eventos quando ocorrem alterações em seus atributos (por exemplo, duração e taxa de exibição) e quando ele é finalizado.

A instanciação de um *Player* JMF é feita através da chamada a um método estático da classe *Manager* da API JMF. Um novo *Player* pode ser criado a partir das informações de localização dos dados de mídia e do protocolo a ser utilizado para a sua transferência, ambas presentes na URL do arquivo de dados. Os passos seguidos pelo *Manager* para a criação de um novo *Player* são os seguintes, de forma simplificada:

1. Instanciar um objeto da classe *DataSource* para o protocolo especificado.
2. Instanciar um objeto que implemente a interface *Player*, de acordo com o formato dos dados da mídia que será exibida. O pacote JMF possui um conjunto de classes que implementam a interface *Player*, para diversos formatos de dados.
3. Conectar o novo *Player* ao *DataSource*.

Maiores detalhes sobre a criação de novos *Players* fogem do escopo deste trabalho, podendo ser encontradas na especificação da classe *Manager* da API JMF [JMF99b].

4.3.1.2 Diagrama de classes

A Figura 26 apresenta o diagrama de classes das ferramentas de exibição de mídias contínuas baseadas no Java Media Framework.

A classe *HF_JmfPlayer* contém todos os métodos para controle da apresentação de áudio e vídeo. Cada objeto desta classe possui uma instância de um *Player* da API JMF, responsável por processar o fluxo de dados de mídia e apresentá-lo nos dispositivos de saída (visual e/ou acústico). O papel da ferramenta de exibição (*HF_JmfPlayer*) é de um mediador entre o formatador HyperProp e o *Player*, acrescendo uma série de funcionalidades requeridas pelo modelo NCM e não contempladas pelo JMF, como será visto no decorrer desta seção.

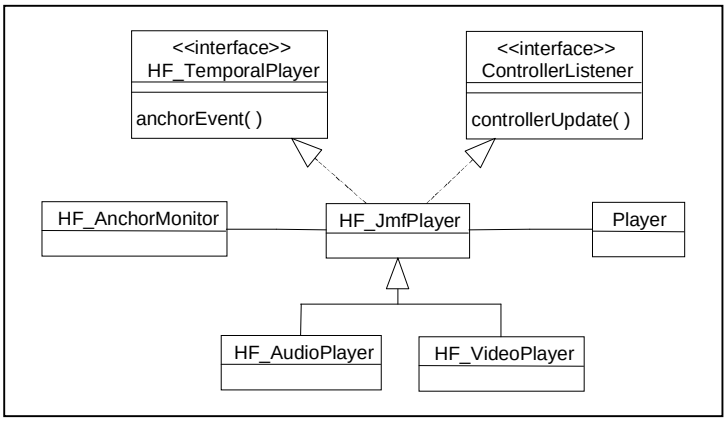

Figura 26 – Ferramentas de exibição de áudio e vídeo baseadas no JMF

A interface *ControllerListener* da API JMF deve ser implementada pela ferramentas de exibição para que elas possam receber eventos gerados pelos *Players*, que são tratados no método *controllerUpdate*. Os eventos gerados pelos *Players* JMF serão apresentados com mais detalhes na Seção 4.3.1.6.

As ferramentas de exibição devem implementar ainda a interface *HF_TemporalPlayer*, para que possam receber do monitor de âncoras (*HF_AnchorMonitor*) notificações de início ou fim da apresentação de âncoras temporais. A implementação dos monitores de âncoras temporais será apresentada na Seção 4.3.1.7.

4.3.1.3 Instanciação do Player JMF

A instanciação do *Player* da API JMF é feita no método *initialize* da ferramenta de exibição, a partir do objeto de representação passado como parâmetro. Este método possui basicamente o seguinte algoritmo:

1. Obter a URL do arquivo de dados a partir do objeto de representação.
2. Criar um novo *Player* através de uma chamada ao método estático *createPlayer* da classe *Manager*, passando como parâmetro a URL, conforme apresentado na Seção 4.3.1.1.
3. Registrar a ferramenta de exibição como observador do *Player*, para que ela receba os eventos gerados por ele.
4. Chamar o método *realize* do *Player*, para iniciar o processo de verificação dos recursos necessários e alocação daqueles que não são de uso exclusivo.

Assim que termina o processo de alocação dos recursos não exclusivos, a ferramenta de exibição já possui um novo *Player* JMF instanciado e pronto para receber chamadas de métodos, em conseqüência das ações NCM a serem solicitadas pelo formatador.

4.3.1.4 Relação entre estados dos eventos NCM e dos Players JMF

No Capítulo 2 foi descrita a máquina de estados dos eventos NCM, enquanto a Seção 4.3.1.1 apresentou a máquina de estados de um *Player* da API JMF. O mapeamento entre estes estados deve ser feito pela ferramenta de exibição, de acordo com a Tabela 10.

NCM	**JMF**
dormindo	*unrealized, realizing, realized*
preparando	*prefetching*
preparado	*prefetched*
ocorrendo	*started*
suspenso	*prefetched*
abortado	---

Tabela 10 – Mapeamento entre estados de eventos NCM e de *Players* JMF

O estado *dormindo* do NCM corresponde a três estados distintos no JMF: *unrealized*, *realizing* e *realized*. Isto ocorre porque o método *realize* do *Player* é chamado durante a sua iniciação, conforme visto na Seção 4.3.1.3, e não a partir da ação *prepara* solicitada pelo formatador. O principal motivo de se fazer o mapeamento desta forma é que o formatador HyperProp pode fazer uso de um recurso do *Player* que estima o tempo de carregamento dos dados, para utilização em um eventual algoritmo de *prefetch*. Este recurso é oferecido pelo método *getStartLatency*[9], que não pode ser chamado enquanto o *Player* ainda está no estado *unrealized*.

O estado *preparando* do NCM corresponde ao estado *prefetching* do JMF, onde as unidades de informação estão sendo carregadas. Quando a apresentação está pronta para

[9] Apesar de este método estar especificado na API JMF, é tecnicamente difícil estimar o tempo de carregamento dos dados. De fato, a implementação de referência da Sun retornará sempre a constante *LATENCY_UNKNOWN*, indicando que o tempo necessário para iniciar a apresentação do objeto não pode ser determinado.

ser iniciada, o evento NCM está no estado *preparado*, correspondendo ao estado *prefetched* do JMF. O evento NCM está no estado *ocorrendo* quando a apresentação está sendo realizada, ou seja, quando o *Player* está no estado *started*.

Assim como o estado *preparado*, o estado *suspenso* do NCM também é mapeado no estado *prefetched* do JMF. Para o *Player*, a única diferença entre estes estados do NCM é o ponto de exibição da mídia, que no estado *preparado* do NCM é o tempo de início da âncora, e no estado *suspenso* do NCM é o ponto onde a apresentação foi suspensa. O estado *abortado* do NCM só dura um tempo instantâneo, retornando imediatamente para *preparado*. Desta forma, ele não possui nenhum estado equivalente no JMF.

4.3.1.5 Relação entre ações NCM e métodos dos Players JMF

As ações solicitadas pelo formatador HyperProp, apresentadas no Capítulo 2, devem ser mapeadas pela ferramenta de exibição em ações a serem executadas pelo *Player*, através de chamadas a seus métodos. Os principais métodos oferecidos pelo *Player* para controle da apresentação são os seguintes:

- *realize*: Inicia os processos de verificação dos recursos necessários para a apresentação da mídia e a alocação daqueles que não são de uso exclusivo.
- *prefetch*: Inicia os processos de carregamento dos dados da mídia e alocação dos recursos de uso exclusivo.
- *start*: Inicia a apresentação da mídia.
- *stop*: Suspende a apresentação, sem alterar a posição corrente no fluxo da mídia.
- *setMediaTime*: Altera a posição corrente no fluxo da mídia.
- *setStopTime*: Estabelece a posição no fluxo da mídia onde a apresentação será suspensa.
- *setRate*: Estabelece a taxa de exibição da mídia.
- *deallocate*: Libera os recursos de uso exclusivo e minimiza o uso de recursos não exclusivos.
- *close*: Libera todos os recursos que estavam sendo utilizados e termina a sua atividade.

O mapeamento entre as ações NCM e as ações a serem executadas sobre o *Player* da API JMF está apresentado na Tabela 11.

NCM	JMF
prepara	*setMediaTime(T_i) + setStopTime(T_f) + prefetch*
inicia	*start*
termina *se for repetir* *se não for repetir*	 *stop + setMediaTime(T_i)* *stop + deallocate + close*
suspende	*stop*
reassume	*start*
aborta	*stop + deallocate + close*

T_i = posição de início da apresentação, T_f = posição de término da apresentação

Tabela 11 – Mapeamento entre ações NCM e ações a serem executadas sobre os *Players* JMF

Todas as ações NCM são solicitadas pelo formatador à ferramenta de exibição, que por sua vez realiza a conversão para a chamada de métodos ao *Player* JMF. As ações NCM possuem como parâmetro o evento de apresentação NCM ao qual se aplicam.

A ação *prepara* resulta na chamada de três métodos do *Player*. O método *setMediaTime* estabelece a posição no fluxo de mídia onde a apresentação será iniciada, correspondente ao tempo de início da âncora[10]. No caso particular da âncora λ ou de uma âncora estritamente espacial, o início da apresentação ocorre na posição zero do fluxo. O método *setStopTime* determina a posição de término da apresentação, que é igual ao menor valor entre o tempo de término da âncora e a soma do tempo de início da âncora com a duração especificada no descritor (caso ela esteja especificada). No caso específico da âncora λ ou de uma âncora estritamente espacial, se a duração não estiver especificada, o método *setStopTime* não é chamado, de modo que a apresentação será conduzida até o término do fluxo de dados ou até que o método *stop* seja chamado. Após a definição das posições de início e fim da apresentação, o método *prefetch* é chamado para iniciar a busca dos dados de mídia.

A ação *inicia* do NCM gera uma chamada ao método *start* do *Player*, para iniciar a apresentação dos dados da mídia. A ação *termina* do NCM gera uma chamada ao método

[10] Quando o início da âncora é definido por unidades de dados (por exemplo, quadro inicial de um objeto de vídeo), é necessário fazer uma conversão para unidades de tempo (por exemplo, dividindo o valor pela taxa de exibição do vídeo). Esta observação também é válida para o término da âncora.

stop do *Player* e, a partir daí, pode assumir dois comportamentos distintos. Se o atributo *repetições* do evento NCM indicar que a apresentação deve ser repetida, a ferramenta de exibição reposiciona o ponto de exibição corrente no fluxo de mídia, através da chamada ao método *setMediaTime*, para que na próxima chamada ao método *start* do *Player* a apresentação reinicie na posição correta do fluxo. É importante observar que a próxima chamada ao método *start* do *Player* ocorre como conseqüência de uma nova ação *inicia* solicitada pelo formatador à ferramenta de exibição, pois é ele que é responsável por iniciar a próxima repetição da apresentação. Caso a apresentação não seja repetida, a ferramenta de exibição libera os recursos do sistema e termina as atividades do *Player*.

A ação *suspende* gera simplesmente uma chamada ao método *stop* do *Player*, sem alterar o ponto de exibição corrente. Desta forma, a ação *reassume* reinicia a apresentação a partir do ponto onde ela foi suspensa, através da chamada ao método *start* do *Player*. A ação *aborta* finaliza a apresentação, libera os recursos ocupados pelo *Player* e termina as suas atividades, de forma semelhante ao que ocorre como conseqüência da ação *termina*, sem que haja uma repetição da apresentação.

4.3.1.6 Transições de estado dos eventos NCM a partir de eventos gerados pelo Player JMF

Os eventos gerados pelo *Player* são recebidos e tratados pela ferramenta de exibição, que implementa a interface *ControllerListener* e é registrada como um observador do *Player*, conforme apresentado na Seção 4.3.1.1. No contexto deste trabalho, os principais eventos que são gerados pelo *Player* são os seguintes:

- *RealizeCompleteEvent*: Ocorre em resposta à chamada ao método *realize*, ao final da alocação dos recursos que não são de uso exclusivo.
- *PrefetchCompleteEvent*: Ocorre em resposta à chamada ao método *prefetch*, quando todos os recursos já foram obtidos e a quantidade de dados carregados é suficiente para iniciar a exibição.
- *StartEvent*: Ocorre em resposta à chamada ao método *start* do *Player*, quando a exibição é iniciada ou reiniciada.
- *StopByRequestEvent*: Ocorre quando a exibição é suspensa em resposta à chamada ao método *stop* do *Player*.

- *StopAtTimeEvent*: Ocorre quando o ponto de exibição no fluxo de mídia atinge a posição de término, estabelecida previamente através do método *setStopTime*.
- *EndOfMediaEvent*: Ocorre quando o ponto de exibição atinge o final do fluxo de mídia.
- *ControllerClosedEvent*: Ocorre em resposta à chamada ao método *close*, quando o *Player* é finalizado.
- *RateChangeEvent*: Ocorre quando a taxa de exibição é alterada.

Alguns destes eventos devem ser mapeados em transições de estado dos eventos de apresentação do modelo NCM pela ferramenta de exibição, a serem sinalizadas ao formatador HyperProp. Este mapeamento está apresentado na Tabela 12.

JMF	**NCM**
PrefetchCompleteEvent	*preparando* → *preparado*
StartEvent	*preparado/suspenso* → *ocorrendo*
StopByRequestEvent *em resposta à ação NCM termina* *em resposta à ação NCM suspende* *em resposta à ação NCM aborta*	 *ocorrendo/suspenso* → *preparado* *ocorrendo* → *suspenso* *ocorrendo/suspenso* → *abortado* → *preparado*
StopAtTimeEvent	*ocorrendo* → *preparado*
EndOfMediaEvent	*ocorrendo* → *preparado*
ControllerClosedEvent	*ocorrendo/suspenso* → *abortado* → *preparado*

Tabela 12 – Mapeamento entre eventos JMF e transições de estado nos eventos de apresentação NCM

Ao receber do *Player* o evento de término da busca dos dados, a ferramenta de exibição informa ao formatador que o estado do evento de apresentação NCM deve passar de *preparando* para *preparado*. Um evento de início (ou reinício) de apresentação sinalizado pelo *Player* gera a mudança de estado do evento NCM de *preparado* ou *suspenso* para *ocorrendo*.

Um evento de suspensão da apresentação por solicitação explícita (chamada ao método *stop* do *Player*) pode ser conseqüência de três ações NCM distintas solicitadas pelo formatador. Caso seja uma resposta à ação *termina*, a ferramenta de exibição deve informar ao formatador a mudança de estado do evento de apresentação de *ocorrendo* ou *suspenso* para *preparado*. Caso seja uma resposta à ação *suspende*, o estado do evento de apresentação deve mudar de *ocorrendo* para *suspenso*. Finalmente, se for uma resposta à

ação *aborta* solicitada pelo formatador, o evento de apresentação passa de *ocorrendo* ou *suspenso* para *abortado*, e daí para o estado *preparado*, sendo que esta última mudança de estado fica a cargo do próprio formatador.

Quando o *Player* sinaliza um evento de suspensão da apresentação por alcance do ponto de término preestabelecido, o estado do evento de apresentação deve passar de *ocorrendo* para *preparado*, significando que foi atingido o tempo de término da âncora ou a duração especificada no descritor. O evento de término das unidades de informação tem um tratamento semelhante.

Quando o *Player* é finalizado pela chamada ao método *close*, o estado do evento de apresentação NCM deve passar de *ocorrendo* ou *suspenso* para *abortado*. O próprio formatador se encarrega de realizar imediatamente a transição de *abortado* para *preparado*.

4.3.1.7 O monitor de âncoras temporais

O modelo NCM permite a definição de relações de sincronização entre sub-regiões temporais dos objetos de mídia, conforme ilustram os exemplos 2 e 3 da Seção 4.1. Portanto, uma das funcionalidades que devem ser oferecidas pelas ferramentas de exibição é a capacidade de sinalizar ao formatador HyperProp o início e o fim de eventos de apresentação de âncoras temporais.

Conforme foi apresentado na Seção 4.3.1.5, quando o formatador solicita à ferramenta de exibição a apresentação de uma âncora temporal (através das ações *prepara* e *inicia*), os métodos *setMediaTime* e *setStopTime* do *Player* JMF determinam as posições de início e término da apresentação. Desta forma, os eventos de início e término da apresentação desta âncora são informados pelo *Player* à ferramenta de exibição através dos eventos *StartEvent* e *StopAtTimeEvent*, respectivamente. Por sua vez, a ferramenta de exibição sinaliza a ocorrência dos eventos ao formatador.

Por outro lado, o evento de apresentação de uma âncora temporal pode ser interno a um outro evento de apresentação que foi iniciado. Por exemplo, consideremos a apresentação de um vídeo com duração de 30 minutos contendo uma âncora entre os minutos 10 e 15. Quando o vídeo atingir 10 e 15 minutos de apresentação, a ferramenta de exibição deve sinalizar ao formatador, respectivamente, o início e o fim da apresentação da âncora, sem

interferir no processo de apresentação do vídeo. Esta é uma funcionalidade que não está especificada na API dos *Players* JMF.

O *monitor de âncoras temporais* (classe *HF_AnchorMonitor*) foi projetado e implementado para incorporar esta funcionalidade às ferramentas de exibição. A sua função é verificar continuamente se a apresentação do objeto de mídia atingiu o início ou fim de qualquer uma das âncoras temporais nele especificadas, para informar à ferramenta de exibição. Esta, por sua vez, sinaliza a ocorrência destes eventos ao formatador.

Para cada ferramenta de exibição de mídia contínua instanciada, há um monitor de âncoras associado. O monitor de âncoras é implementado como uma *thread*, que é disparada pela ferramenta de exibição sempre que a apresentação é iniciada pela primeira vez ou reiniciada após uma interrupção. A *thread* termina a sua execução quando a apresentação é interrompida ou finalizada, ou quando não há mais eventos de apresentação de âncoras a serem sinalizados a partir do ponto de exibição corrente.

Assim que o monitor de âncoras é iniciado, ele monta uma tabela com todos os eventos de apresentação de âncoras temporais que devem ser sinalizados. Cada registro desta tabela possui as seguintes informações: o evento que deve ser sinalizado, o tempo em que a sinalização deve acontecer[11] e o tipo de ocorrência (início ou fim da apresentação da âncora em questão). Esta tabela é ordenada pelo tempo em que o evento deve ser sinalizado, através de um dos algoritmos de ordenação já conhecidos (por exemplo, *quicksort* [CoLR90]). No caso particular de uma âncora pontual, ou seja, com tempo de término igual ao tempo de início, deve-se garantir que o registro correspondente ao início da apresentação da âncora seja anterior ao correspondente ao término, para que os eventos sejam sinalizados nesta ordem.

[11] Quando a âncora é definida por unidades de tempo, os valores de início e fim são diretamente incluídos nos registros da tabela. Quando a âncora é definida por unidades de dados (por exemplo, quadro inicial e quadro final de um objeto de vídeo), é necessário fazer uma conversão para unidades de tempo (por exemplo, dividindo os valores pela taxa de exibição do vídeo).

Quando está sendo apresentada a âncora λ do objeto de mídia (ou seja, todo o seu conteúdo), o monitor deve sinalizar todos os eventos presentes na tabela, à medida que os tempos definidos vão sendo atingidos. Entretanto, no caso da apresentação de uma âncora temporal, o monitor deve considerar apenas os eventos cujos tempos de sinalização estão contidos no intervalo da âncora. Para isso, após montar a tabela de eventos, o monitor atualiza o seu índice, de modo que ele aponte para o primeiro evento imediatamente após o tempo de início da apresentação da âncora.

O funcionamento do monitor consiste na verificação freqüente do ponto de exibição corrente da mídia, através do método *getMediaTime* do *Player* JMF. Após cada verificação, o monitor localiza na tabela ordenada de eventos, a partir de seu índice atual, aqueles cujos tempos de sinalização já foram atingidos, mas que ainda não foram sinalizados até a última verificação realizada. Neste momento, estes eventos são sinalizados à ferramenta de exibição, através de uma chamada ao método *anchorEvent*, definido na interface *TemporalPlayer*. O método *anchorEvent* das ferramentas de exibição é responsável por repassar a ocorrência destes eventos ao formatador HyperProp. A interface *TemporalPlayer* deve ser implementada por todos as ferramentas de exibição de mídias contínuas que desejem tratar a ocorrência de eventos de apresentação de âncoras temporais por meio de um monitor de âncoras temporais. Após a sinalização dos eventos para a ferramenta de exibição, o monitor atualiza o índice de sua tabela de eventos para apontar para o próximo evento ainda não sinalizado.

A freqüência da verificação do ponto de exibição corrente da mídia deve levar em conta dois fatores: a precisão do momento da sinalização do evento e a quantidade de recursos utilizados pela *thread*.

Numa primeira abordagem, o monitor poderia verificar constantemente o ponto de exibição corrente, ou seja, sempre que a *thread* estivesse sendo executada pelo processador, esta estaria em um *loop* chamando o método *getMediaTime* do *Player* JMF e verificando na tabela de eventos se há algum a ser sinalizado. Esta forma de implementação caracteriza-se por uma grande precisão do momento de sinalização do evento, porém utiliza um grande volume de recursos de processamento.

Uma segunda forma de implementação divide a linha de tempo da apresentação em *slots* de tamanho fixo. O tamanho destes *slots* de tempo é denominado *granularidade*. O monitor de âncoras verifica a ocorrência dos eventos apenas no final de cada *slot* de tempo, de modo que todos os eventos que ocorreram no *slot* corrente são sinalizados neste momento. Em seguida, a *thread* dorme[12] pelo tempo da granularidade, para acordar novamente ao final do *slot* seguinte. Portanto, no pior caso, o atraso na sinalização dos eventos é o valor da granularidade. Esta forma de implementação apresenta uma precisão menor que a anterior, porém consome menos recursos do sistema, pois a *thread* fica suspensa na maior parte do tempo. Quanto menor a granularidade, maior é a precisão e mais recursos são utilizados. No caso extremo de granularidade igual a zero, o comportamento do monitor é igual à primeira forma de implementação apresentada.

A Figura 27 ilustra um monitor de âncoras com granularidade de 0,5s, para a apresentação de um vídeo com duração de 20s contendo uma âncora α de 10,2s a 16,8s. Neste caso, o monitor verifica a ocorrência dos eventos a cada 0,5s, de modo que o início e o término da apresentação de α são sinalizados respectivamente nos tempos 10,5s e 17s. O monitor termina sua execução aos 17s, quando ele sinaliza o último evento de apresentação de âncoras, realizando um total de 34 verificações do ponto de exibição corrente da mídia.

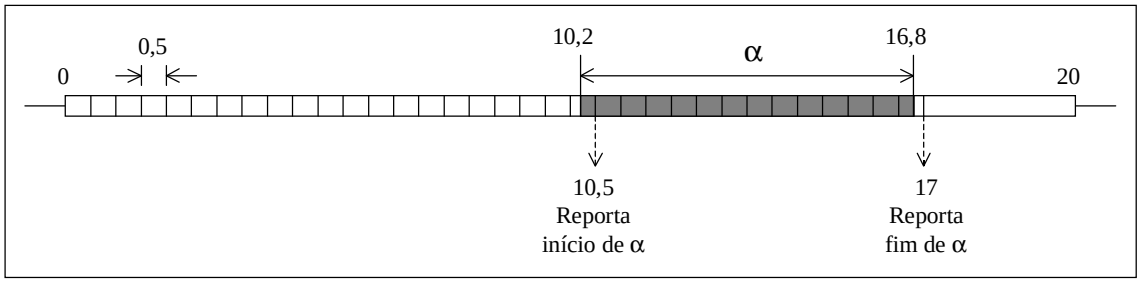

Figura 27 – Implementação do monitor de âncoras com granularidade de 0,5s

A terceira forma de implementação complementa a anterior, adicionando o conceito de *sleep-factor*. Este conceito aproveita o conhecimento que o monitor tem de todos os eventos a serem sinalizados, ordenados na tabela por tempo de sinalização. Quando um

[12] Os conceitos de "dormir" e "acordar" correspondem, respectivamente, à saída da *thread* da fila de execução e ao seu retorno.

evento é sinalizado, o monitor pode fazer uma previsão do momento de sinalização do evento seguinte, a partir do tempo de sinalização presente no próximo registro da tabela e do ponto de exibição corrente. Desta forma, se o próximo evento estiver previsto para ser sinalizado após um período de tempo suficientemente longo, a *thread* do monitor pode dormir por um tempo maior que a granularidade, diminuindo o gasto de recursos do sistema. Cada vez que a *thread* é executada, o valor do tempo durante o qual ela irá dormir em seguida é calculado como o valor máximo entre a granularidade e a seguinte expressão:

$$(t\_next - t\_now) \times sleep\_factor$$

onde *t_next* é o tempo de sinalização do próximo evento, *t_now* é o ponto de exibição corrente e *sleep_factor* é um fator multiplicativo real pertencente ao intervalo [0,1]. Este fator tem como objetivo precaver-se contra pequenas variações no fluxo de mídia, que poderiam eventualmente adiantar a ocorrência do início ou fim da exibição de uma âncora[13], ou mesmo contra a influência dos tempos de processamento sobre o momento da sinalização dos eventos. Caso o *sleep-factor* seja igual a 1, o monitor irá dormir exatamente durante o intervalo de tempo previsto para a ocorrência do próximo evento, se este for maior que a granularidade, ou durante o próprio valor da granularidade, caso contrário. Um valor menor que 1 e maior que zero faz com que o monitor durma sucessivamente durante intervalos cada vez menores, à medida que o ponto de exibição se aproxima do tempo de ocorrência do próximo evento, até que seja atingido o valor da granularidade. Se o *sleep-factor* for igual a zero, monitor irá dormir sempre durante o tempo da granularidade, recaindo na segunda forma de implementação apresentada.

Um exemplo de monitor de âncoras implementado desta forma está apresentado na Figura 28. Ele possui uma granularidade de 0,5s, um *sleep-factor* igual a 0,8 e está monitorando a apresentação do mesmo vídeo do exemplo anterior, isto é, com duração de 20s e contendo uma âncora α de 10,2s a 16,8s. Desprezando as variações no fluxo da mídia e os tempos de processamento, o monitor verificará se houve a ocorrência do início da exibição de α nos

[13] Esta afirmação é válida para as implementações dos *Players* JMF cuja base de tempo não está vinculada ao relógio do sistema, mas às unidades de dados que estão sendo exibidas. Neste caso, uma variação no fluxo de mídia é refletida na base de tempo.

tempos 8,16s, 9,79s e 10,29s, quando então sinalizará o evento à ferramenta de exibição. Em seguida, ele passará a monitorar o término da exibição de α, fazendo a verificação nos tempos 15,5s, 16,54s e 17,04s, quando então sinalizará este evento. Portanto, a quantidade de verificações do ponto de exibição corrente efetuadas nesta implementação do monitor é igual a 6, sendo muito inferior em relação à segunda forma de implementação.

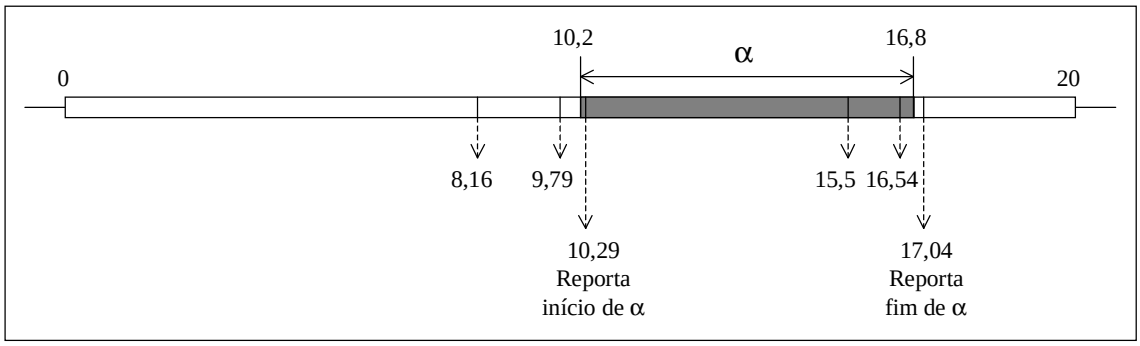

Figura 28 – Implementação do monitor de âncoras com granularidade de 0,5s e *sleep-factor* igual a 0,8

Esta terceira forma de implementação foi a escolhida para os monitores associados às ferramentas de exibição de mídias contínuas do sistema HyperProp, em razão das vantagens já apresentadas. Ao ser instanciado, o monitor recebe como parâmetro a ferramenta de exibição que o instanciou, para sinalizar os eventos ocorridos para ela. Além disso, ele recebe como parâmetros a granularidade e o *sleep-factor*. O valor destes parâmetros pode ser definido pelo usuário na especificação da plataforma, de acordo com a precisão desejada na sincronização temporal entre âncoras e com a disponibilidade de recursos do sistema.

Existem alguns casos em que a ferramenta de exibição deve acordar a *thread* do monitor assincronamente para fazer alguns ajustes decorrentes de eventos imprevisíveis. Um destes eventos é a alteração descontínua do ponto de exibição da mídia durante a apresentação, por meio de interação do usuário com o componente de controle ou de chamada ao método *set* da ferramenta. Neste caso, o índice da tabela ordenada de eventos deve ser recalculado para refletir a alteração do ponto de exibição. Um outro evento imprevisível é a alteração da taxa de exibição durante a apresentação, por meio de chamada ao método *set* da ferramenta. Quando isto ocorre, os parâmetros granularidade e *sleep-factor* do monitor devem ser proporcionalmente alterados para que seja mantido o mesmo comportamento.

4.3.1.8 Sincronização espacial

A sincronização espacial diz respeito ao tamanho e à localização dos componentes visuais das mídias. No NCM, estas informações estão presentes no descritor associado ao nó que está sendo exibido.

Um *Player* JMF provê dois componentes de interface com o usuário, chamados *componente visual* e *componente de controle*.

O componente visual é obtido através de uma chamada ao método *getVisualComponent*. Este componente corresponde à representação visual da mídia, se houver. As mídias de vídeo sempre possuem uma representação visual. Embora seja menos comum, as mídias de áudio também podem possuir uma representação visual, como indicadores de nível ou o formato da onda.

O componente de controle é obtido através de uma chamada ao método *getControlPanelComponent*. Ele apresenta um conjunto de componentes gráficos para o usuário controlar a apresentação, como botões de *play*, *stop* e *pause*, para iniciar, terminar ou suspender a apresentação, respectivamente.

De acordo com o que foi apresentado no início da Seção 4.3, uma ferramenta de exibição do tipo janela possui um objeto *Frame*, onde são inseridos o componente visual, se houver, e o componente de controle. Por sua vez, uma ferramenta de exibição do tipo painel possui um objeto *Panel*, onde é inserido apenas o componente visual, se houver. Os painéis são inseridos em uma única janela, com um único componente de controle para todos os objetos de mídia do documento que está sendo apresentado.

As informações das dimensões (largura e altura) e posição (coordenada do canto superior esquerdo) são obtidas do descritor associado ao nó. Caso as dimensões não sejam especificadas no descritor, a ferramenta de exibição utiliza as dimensões intrínsecas do objeto, através da chamada ao método *getPreferredSize* do componente visual. A largura do componente de controle é definida com o mesmo valor da largura do componente visual, e sua altura é obtida através da chamada ao seu método *getPreferredSize*. A posição das ferramentas do tipo janela é calculada em relação à tela do monitor de vídeo, enquanto nas ferramentas do tipo painel ela é calculada em relação à janela de apresentação. Os

atributos *left*, *top*, *width* e *height* guardam os valores da posição esquerda, posição superior, largura e altura do componente visual, respectivamente, enquanto o atributo *ctr_height* guarda o valor da altura do componente de controle.

As ferramentas de exibição devem garantir que os componentes visuais das mídias sejam redimensionados sempre que o usuário alterar o tamanho da janela de apresentação. Quando isto ocorre, as ferramentas do tipo painel inicialmente recalculam a posição e o tamanho do objeto *Panel*, de forma proporcional à alteração do tamanho da janela de apresentação. Em seguida, são recalculados a posição e o tamanho dos componentes visual e de controle que estão inseridos no *Panel*. Uma ferramenta do tipo janela só precisa recalcular a posição e o tamanho dos componentes visual e de controle, pois estes já estão diretamente inseridos nela.

A sincronização espacial da apresentação de âncoras espaciais possui algumas particularidades, que serão discutidas na próxima seção.

4.3.1.9 Apresentação de âncoras espaciais

Uma âncora espacial corresponde a uma área do componente visual da mídia. Neste trabalho, considera-se apenas o caso de uma âncora em formato retangular.

Quando é solicitada a apresentação de uma âncora espacial, a ferramenta de exibição utiliza um recurso para limitar a apresentação do componente visual à área definida na âncora. Este recurso consiste no reposicionamento e redimensionamento do componente visual dentro do *display* (janela ou painel), de forma que apenas a âncora fique visível. A Figura 29 ilustra a apresentação de uma âncora espacial de um vídeo.

O efeito de visualização de apenas uma sub-região do componente visual da mídia está ilustrado na Figura 30, onde a âncora espacial está indicada por um quadrado pontilhado na Figura 30a. Este efeito é obtido por uma diminuição das dimensões do *display* sem alterar as dimensões do componente visual (Figura 30b), seguida de um possível deslocamento das posições esquerda e/ou superior do *display* (Figura 30c), compensado por um deslocamento negativo do componente visual em relação ao *display* (Figura 30d). Pode-se observar que a região que está sendo visualizada na Figura 30d corresponde exatamente à âncora assinalada na Figura 30a.

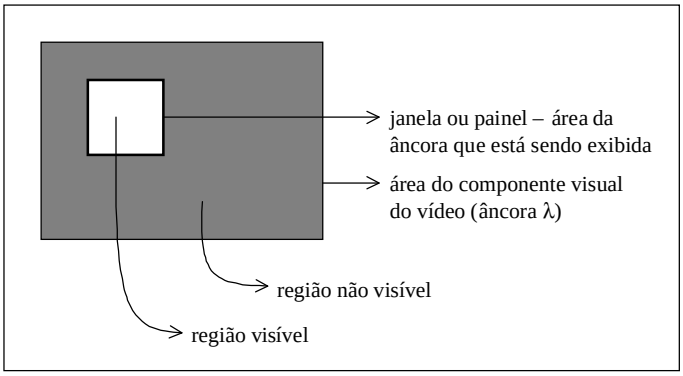

Figura 29 – Apresentação de uma âncora espacial

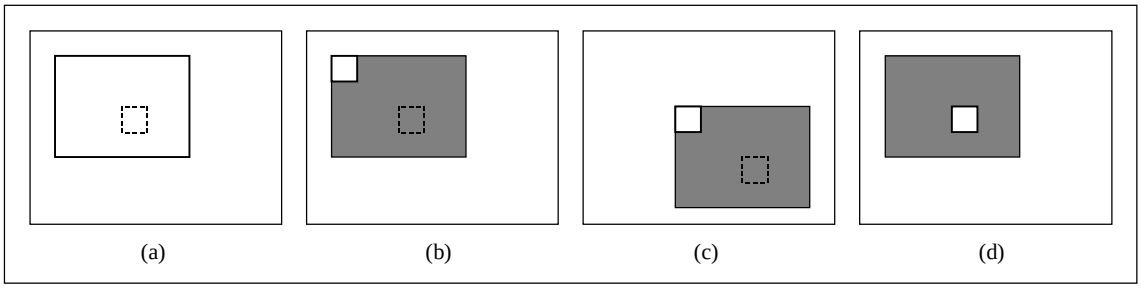

Figura 30 – Efeito de visualização de uma âncora espacial

Para calcular as posições e as dimensões do *display* e do componente visual correspondente a uma âncora espacial, a ferramenta de exibição possui quatro atributos adicionais, além dos atributos *left*, *top*, *width*, *height* e *ctr_height* citados anteriormente:

- *rel_left*: Posição esquerda da âncora em relação à largura do componente visual.
- *rel_top*: Posição superior da âncora em relação à altura do componente visual.
- *rel_width*: Largura da âncora em relação à largura do componente visual.
- *rel_height*: Altura da âncora em relação à altura do componente visual.

No caso particular da âncora λ ou de âncoras estritamente temporais, os atributos *rel_left* e *rel_top* são iguais a 0 e os atributos *rel_width* e *rel_height* são iguais a 1, o que corresponde a toda a região do componente visual.

As posições e as dimensões do *display* e dos componentes visual e de controle são calculadas antes de se iniciar a apresentação dos dados de mídia. As posições e dimensões dos componentes visual e de controle são recalculadas sempre que o tamanho do *display* se

alterar, seja por interação do usuário ou por chamada ao método *set* da ferramenta de exibição.

Sejam *d_width* e *d_height* os valores (iniciais ou alterados) da largura e da altura do *display*, respectivamente. A Tabela 13 apresenta os cálculos das posições e dimensões iniciais do *display*, bem como os cálculos das posições e dimensões dos componentes visual e de controle, em relação ao próprio *display*.

	***display* (inicialmente)**	**componente visual**	**componente de controle**
posição esquerda	*left* + *rel_left* × *width*	– *rel_left* × (*d_width* / *rel_width*)	0
posição superior	*top* + *rel_top* × *height*	– *rel_top* × ((*d_height* – *ctr_height*) / *rel_height*)	*d_height* – *ctr_height*
largura	*rel_width* × *width*	*d_width* / *rel_width*	*d_width*
altura	*rel_height* × *height* + *ctr_height*	(*d_height* – *ctr_height*) / *rel_height*	*ctr_height*

Tabela 13 – Cálculos das posições e dimensões do *display* e dos componentes visual e de controle

Nos cálculos das posições iniciais do *display*, as parcelas (*rel_left* × *width*) e (*rel_top* × *height*) correspondem ao deslocamento de suas posições esquerda e superior, respectivamente. Nos cálculos das dimensões iniciais do *display*, os fatores *rel_width* e *rel_height* correspondem ao redimensionamento de sua largura e de sua altura, respectivamente. O cálculo da altura do *display* também deve considerar a altura do componente de controle (*ctr_height*) nas ferramentas do tipo janela.

Os valores iniciais de largura e altura do componente visual, calculados antes de se iniciar a apresentação, devem ser iguais aos atributos *width* e *height* da ferramenta de exibição, respectivamente. De fato, estes são os valores obtidos pela substituição dos parâmetros *d_width* e *d_height* pelos valores iniciais de largura e altura do *display* nos cálculos das dimensões do componente visual. Os valores iniciais das posições esquerda e superior do componente visual devem compensar o deslocamento sofrido pelo *display*. Isto pode ser observado pela substituição dos parâmetros *d_width* e *d_height* pelos valores iniciais de largura e altura do *display* nos cálculos das posições do componente visual.

O componente de controle, se existir, localiza-se sempre na região inferior do *display*. Este componente possui a mesma largura do *display* e sua altura é fixa, ficando armazenada no atributo *ctr_height*, conforme visto na Seção 4.3.1.8.

4.3.1.10 Apresentação de objetos com múltiplas âncoras

Quando o objeto que está sendo exibido possui mais de uma âncora, algumas regras devem ser estabelecidas para decidir quais eventos devem ser sinalizados para o formatador durante a apresentação.

Consideremos inicialmente um objeto de mídia contínua com múltiplas âncoras temporais. As seguintes regras foram estabelecidas[14]:

1. O início da apresentação de uma âncora deve ser sinalizado sempre que o ponto de exibição corrente atingir o tempo de início especificado, contanto que o evento NCM de apresentação da âncora esteja no estado *preparado*. Esta restrição deve ser verificada pela ferramenta de exibição ao receber do monitor de âncoras a sinalização de um evento, antes de repassá-lo para o formatador HyperProp.
2. O término da apresentação de uma âncora deve ser sinalizado sempre que o ponto de exibição corrente atingir o tempo de término especificado, contanto que o evento NCM de apresentação da âncora esteja no estado *ocorrendo*. Assim como na regra anterior, esta restrição deve ser verificada pela ferramenta de exibição ao receber do monitor a sinalização de um evento, antes de repassá-lo para o formatador.
3. Quando a apresentação acaba de ser preparada pela ação NCM *prepara*, deve ser sinalizado o final da preparação de todos os eventos NCM de apresentação, de modo que eles passem para o estado *preparado*. Isto permite que, à medida que as âncoras temporais sejam apresentadas, os eventos NCM associados passem diretamente do estado *preparado* para o estado *ocorrendo*.
4. Quando a apresentação é iniciada, seja por conseqüência da ação NCM *inicia* ou de interação do usuário com o componente de controle, deve ser sinalizado apenas o início

[14] Nestas regras, a âncora λ comporta-se como uma âncora temporal, onde o intervalo de tempo a ser apresentado compreende toda a duração do objeto de mídia.

da apresentação da âncora em questão e de outras que porventura venham a ter o mesmo tempo de início. Não deve ser sinalizado o início da apresentação das âncoras com tempos de início anteriores a este momento, mesmo que o ponto de exibição atual esteja contido em seu intervalo.

5. Quando a apresentação é finalizada, seja por conseqüência do término dos dados, do término da duração especificada, da ação NCM *termina* ou de interação do usuário com o componente de controle, deve ser sinalizado o término da apresentação de todas as âncoras cujos eventos NCM se encontram no estado *ocorrendo* ou *suspenso*, de modo que eles passem para o estado *preparado*. As âncoras cujos eventos NCM não se encontram nos estados citados não devem ter os términos de suas apresentações sinalizados.
6. Quando a apresentação é suspensa, seja por conseqüência da ação NCM *suspende* ou de interação do usuário com o componente de controle, deve ser sinalizada a suspensão da apresentação de todas as âncoras cujos eventos NCM se encontram no estado *ocorrendo*, de modo que eles passem para o estado *suspenso*.
7. Quando a apresentação é reiniciada após uma suspensão, seja por conseqüência da ação NCM *reassume* ou de interação do usuário com o componente de controle, duas situações devem ser consideradas. Se o ponto de exibição não tiver sido alterado, deve ser sinalizado o reinício da apresentação de todas as âncoras cujos eventos NCM se encontram no estado *suspenso*, de modo que eles retornem ao estado *ocorrendo*. Se o ponto de exibição tiver sido alterado durante a suspensão da apresentação, deve ser sinalizado o reinício da apresentação das âncoras cujos eventos NCM se encontram no estado *suspenso* e cujos intervalos de tempo contêm o novo ponto de exibição. No caso das âncoras cujos eventos NCM se encontram no estado *suspenso* e cujos intervalos de tempo não contêm o novo ponto de exibição, deve ser sinalizado o final de sua exibição, de modo que os eventos NCM passem diretamente do estado *suspenso* para o estado *preparado*.

A seguir estão apresentados quatro casos de aplicação destas regras. Consideremos um objeto de mídia contínua contendo duas âncoras temporais: a âncora α, correspondente ao intervalo de tempo $[T_{\alpha i}, T_{\alpha f}]$, e a âncora β, correspondente ao intervalo de tempo $[T_{\beta i}, T_{\beta f}]$. Consideremos também que foi solicitada à ferramenta de exibição a apresentação da

âncora α do objeto, ou seja, apenas os dados de mídia contidos no intervalo de tempo [$T_{\alpha i}$, $T_{\alpha f}$] serão exibidos. Os quatro casos estão ilustrados na Figura 31.

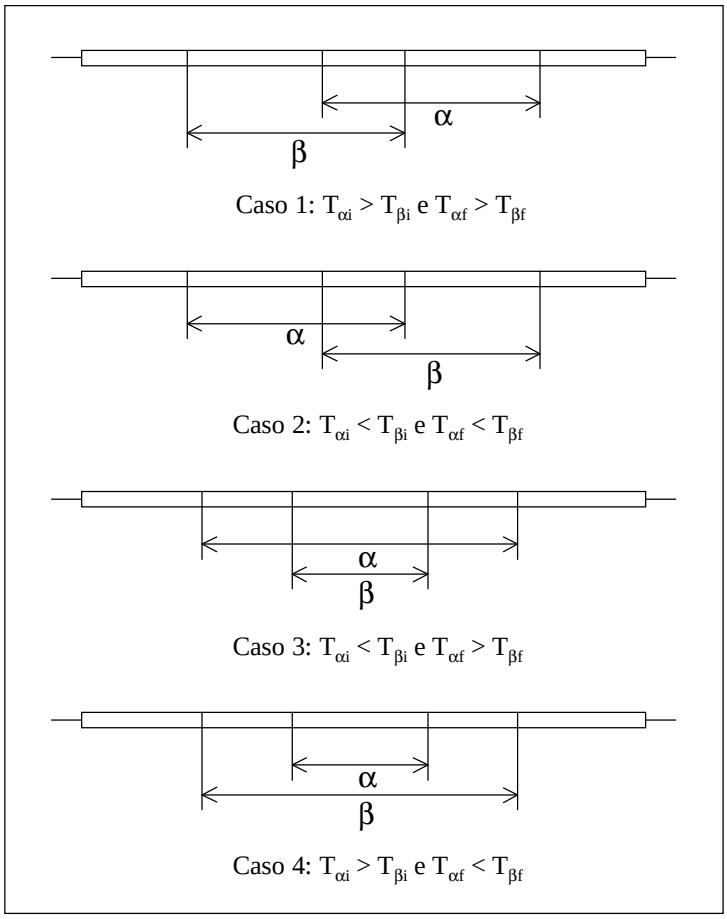

Figura 31 – Apresentação de um objeto de mídia contínua com duas âncoras temporais: $\alpha = [T_{\alpha i}, T_{\alpha f}]$ e $\beta = [T_{\beta i}, T_{\beta f}]$

Caso 1: $T_{\alpha i} > T_{\beta i}$ e $T_{\alpha f} > T_{\beta f}$

A ferramenta de exibição deverá sinalizar para o formatador apenas o início e o término da apresentação da âncora α, nos tempos $T_{\alpha i}$ e $T_{\alpha f}$, respectivamente. A regra 4 impede a sinalização do início da apresentação da âncora β, e a regra 2 impede a sinalização do seu término.

Caso 2: $T_{\alpha i} < T_{\beta i}$ e $T_{\alpha f} < T_{\beta f}$

A ferramenta de exibição deverá sinalizar para o formatador o início da apresentação da âncora α em $T_{\alpha i}$, o início da apresentação da âncora β em $T_{\beta i}$, e o término das apresentações de α e β em $T_{\alpha f}$. A sinalização do início da apresentação de β no tempo $T_{\beta i}$ está de acordo com a regra 1, e a sinalização do término da apresentação de β no tempo $T_{\alpha f}$ está de acordo com a regra 5.

Caso 3: $T_{\alpha i} < T_{\beta i}$ e $T_{\alpha f} > T_{\beta f}$

A ferramenta de exibição deverá sinalizar para o formatador o início e o final da apresentação de ambas as âncoras nos respectivos tempos em que elas foram definidas. As sinalizações do início e do término da apresentação de β estão de acordo com as regras 1 e 2, respectivamente. Este caso pode ser aplicado à âncora λ do objeto, que se comporta como a âncora α em relação a todas as âncoras temporais nele contidas.

Caso 4: $T_{\alpha i} > T_{\beta i}$ e $T_{\alpha f} < T_{\beta f}$

A ferramenta de exibição deverá sinalizar para o formatador apenas o início e o término da apresentação da âncora α, nos tempos $T_{\alpha i}$ e $T_{\alpha f}$, respectivamente. A regra 4 impede a sinalização do início da apresentação da âncora β no momento em que a apresentação de α é iniciada, e a regra 5 impede a sinalização do término da apresentação de β no momento em que a apresentação de α é finalizada.

Estes quatro casos podem ser estendidos para determinar os eventos que devem ser sinalizados durante a apresentação de um objeto que contenha um número qualquer de âncoras temporais.

Consideremos agora um objeto de vídeo com múltiplas âncoras espaciais[15]. Neste caso, foi estabelecida a seguinte regra:

---

[15] Aqui a âncora λ comporta-se como uma âncora espacial, onde a região espacial compreende toda a parte visual do objeto de mídia.

- Os eventos de apresentação das âncoras espaciais que especificam regiões distintas são independentes uns dos outros.

Desta forma, quando é solicitada a apresentação de uma âncora espacial, através da ação NCM *inicia* aplicada sobre o evento de apresentação desta âncora, a ferramenta de exibição não deve sinalizar nenhum evento associado à apresentação de outras âncoras espaciais que especifiquem regiões diferentes da âncora que está sendo exibida. Por outro lado, caso o objeto contenha outras âncoras espaciais que especificam a mesma região, os respectivos eventos de apresentação devem ser sinalizados[16].

A Figura 32 apresenta o exemplo de um objeto de vídeo contendo duas âncoras espaciais que especificam regiões distintas: a âncora α e a âncora β. Consideremos que foi solicitada à ferramenta de exibição a apresentação da âncora α. Durante a apresentação, a ferramenta não deve sinalizar nenhum evento associado à apresentação de β, mesmo que todas as suas unidades de informação estejam contidas em α, como exemplifica a figura. Da mesma forma, considerando agora a âncora λ como uma âncora espacial, os eventos associados a ela não devem ser sinalizados.

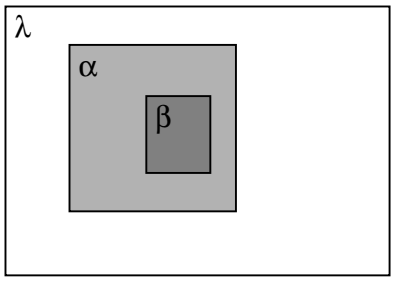

Figura 32 – Apresentação de um objeto de vídeo com duas âncoras espaciais

As âncoras espaço-temporais, específicas dos objetos de vídeo, combinam as características das âncoras temporais e espaciais descritas anteriormente. Quando o formatador solicita à ferramenta de exibição a apresentação de uma âncora espaço-temporal, são considerados para efeito de sinalização ao formatador apenas os eventos

[16] É possível a especificação de duas ou mais âncoras espaciais que denotam a mesma região de um objeto de mídia. A aplicação deste conceito será melhor observada no caso das âncoras espaço-temporais.

associados às âncoras espaço-temporais cujas regiões espaciais sejam iguais à da que está sendo apresentada. Neste caso, as sinalizações destes eventos são submetidas às mesmas regras que regem a apresentação das âncoras temporais, apresentadas anteriormente.

Todos os tipos de âncoras em objetos de vídeo podem ser considerados espaço-temporais. As âncoras temporais são casos particulares de âncoras espaço-temporais onde a região espacial compreende toda a parte visual do vídeo. As âncoras espaciais são âncoras espaço-temporais onde o intervalo de tempo a ser apresentado compreende toda a duração do vídeo. A âncora λ é uma âncora espaço-temporal com ambas as características citadas acima.

4.3.1.11 Eventos de seleção

Os eventos de seleção de uma âncora são disparados por ação do usuário, através da seleção das unidades de informação especificadas na âncora, geralmente com o uso do *mouse*.

As ferramentas de exibição devem verificar a ocorrência de eventos de seleção sempre que o usuário selecionar com o *mouse* o componente visual da mídia. Para isso, elas devem implementar a interface *MouseListener*, registrar-se como observadoras de eventos de *mouse* sobre o componente visual e tratar os eventos de seleção na implementação do método *mouseClicked*. Neste método, devem ser percorridos todos os eventos de seleção especificados para o objeto de mídia que está sendo exibido, para verificar se eles foram efetivamente disparados, e, em caso positivo, sinalizá-los para o formatador HyperProp.

Um evento de seleção de uma âncora deve ser sinalizado ao formatador nos seguintes casos, quando o usuário seleciona com o *mouse* um ponto do componente visual da mídia:

1. A âncora é a própria âncora λ.
2. A âncora é temporal e o ponto de exibição corrente está contido no intervalo de tempo nela especificado.
3. A âncora é espacial e o ponto do componente visual que foi selecionado está contido na região nela especificada.

4. A âncora é espaço-temporal, o ponto de exibição corrente está contido no intervalo de tempo nela especificado e o ponto do componente visual que foi selecionado está contido na região espacial nela especificada.

4.3.1.12 Ações de controle do usuário

As ferramentas de exibição devem disponibilizar ao usuário um conjunto de ações para controle da apresentação. Estas ações devem ser pelo menos as seguintes:

- iniciar a apresentação;
- suspender temporariamente a apresentação;
- reiniciar a apresentação;
- finalizar a apresentação.

Estas ações de controle do usuário comportam-se, respectivamente, da mesma forma que as ações NCM solicitadas à ferramenta de exibição pelo formatador HyperProp: *inicia*, *suspende*, *reassume* e *termina*.

Além destas, algumas outras ações de controle opcionais podem ser oferecidas ao usuário pelas ferramentas de exibição de mídias contínuas, por exemplo:

- alterar o ponto de exibição corrente da mídia;
- alterar o volume do áudio.

O componente de controle dos *Players* JMF normalmente oferece todas estas ações de controle[17], dependendo do tipo da mídia que está sendo exibida. A ativação de uma destas ações por interação do usuário através do *mouse* gera o evento correspondente sobre a apresentação da mídia. A Figura 33 apresenta um exemplo de componente de controle fornecido por um *Player* JMF e contido em uma ferramenta de exibição do tipo janela.

[17] Exceto a finalização da apresentação, que em geral os *Players* JMF não oferecem. Uma alternativa a esta ação é o fechamento da janela de exibição, que se comporta como uma ação NCM *aborta*.

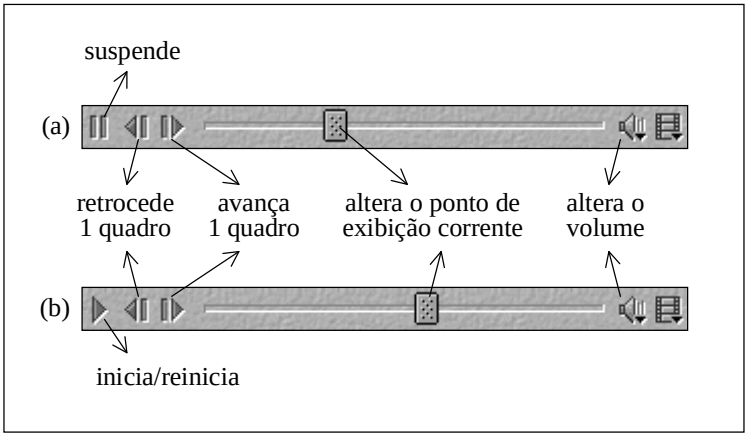

Figura 33 – Componente de controle de um *Player* JMF
(a) Apresentação em andamento. (b) Apresentação suspensa ou ainda não iniciada.

No caso da exibição de vários objetos de mídia em uma mesma janela (ferramentas de exibição do tipo painel), esta deve conter um componente de controle único para todos os objetos, conforme mencionado no início da Seção 4.3. Um exemplo de componente de controle que foi implementado está apresentado na Figura 34.

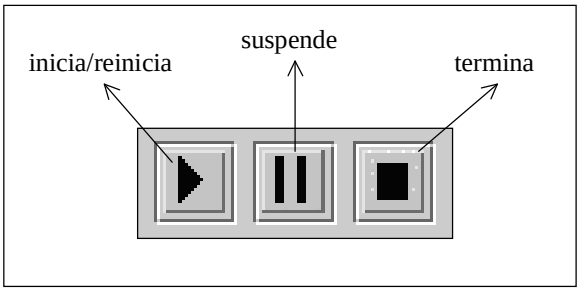

Figura 34 – Componente de controle da janela de apresentação

A implementação da comunicação entre o componente de controle único (*HF_ControlPanel*) e as ferramentas de exibição do tipo painel é feita através da interface *ActionListener*, implementada pela janela principal (*HF_MainWindow*). A janela principal instancia um componente de controle e registra-se como observadora dos eventos de cada um de seus botões de ação (*play*, *pause* e *stop*). Quando qualquer um destes botões é selecionado pelo usuário, é chamado o método *actionPerformed* de *HF_MainWindow*. Por sua vez, este método avisa a ocorrência da ação de controle a cada ferramenta de exibição contida na janela principal, através da chamada ao método de tratamento destas ações. Cada ferramenta de exibição deve implementar este método de tratamento das ações de

controle do usuário, para que os seus resultados sejam iguais aos das ações NCM correspondentes.

4.3.2 Ferramentas de exibição de mídias discretas

As ferramentas de exibição de mídias discretas realizam a apresentação das mídias de texto e imagem. Conforme mencionado no início deste capítulo, as ferramentas de exibição de nós de texto (em formato HTML) já haviam sido implementadas no sistema HyperProp, com suporte aos eventos de seleção de âncoras [RoMS98]. Foi acrescida a estas ferramentas a funcionalidade de exibir textos pelo tempo determinado pela duração do evento, e foi especificado o seu comportamento em resposta às ações NCM *suspende* e *reassume*, com as respectivas influências na duração da apresentação. Por outro lado, assim como as ferramentas de exibição de áudio e vídeo, as ferramentas de exibição de imagens estáticas foram totalmente desenvolvidas dentro do escopo deste trabalho, suportando as relações de interação do usuário (seleção de âncoras) e de sincronização temporal e espacial definidas no modelo NCM.

4.3.2.1 O cronômetro da apresentação

Conforme apresentado no Capítulo 2, o modelo NCM especifica um atributo denominado *duração* para os eventos de apresentação. Nas ferramentas de exibição de mídias contínuas, o término da apresentação causado pelo fim da duração especificada era implementado com o suporte oferecido pelo JMF, através da interface *Clock* (Seção 4.3.1.1). Visto que as ferramentas de exibição de texto e imagem não possuem o suporte do JMF, foi desenvolvido um mecanismo para medir o tempo decorrido da apresentação e determinar o seu término quando este tempo atingir a duração especificada. Este mecanismo é denominado *cronômetro da apresentação*, e está implementado na classe *HF_Timer*, conforme mostra a Figura 35.

O cronômetro da apresentação é implementado como uma *thread*, disparada pela ferramenta de exibição sempre que a apresentação é iniciada através da ação NCM *inicia*. Cada instância de ferramenta de exibição de mídia discreta possui um cronômetro associado. O cronômetro tem como função avisar à ferramenta de exibição o momento em que o tempo decorrido atinge a duração especificada, para que ela finalize a apresentação.

Este aviso é feito através da chamada ao método *endOfTime*, definido na interface *HF_StaticPlayer*. Todas as ferramentas de exibição de mídias discretas devem implementar esta interface para usarem o cronômetro da apresentação.

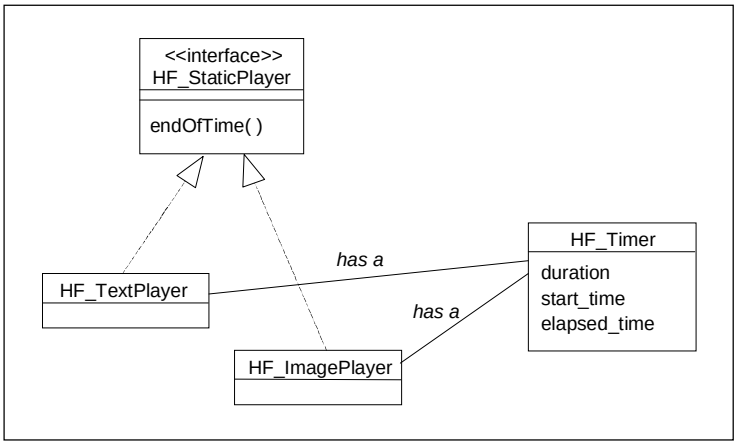

Figura 35 – Cronômetro da apresentação de mídias discretas

O controle do tempo da apresentação pelo cronômetro é feito através dos atributos *duration*, *start_time* e *elapsed_time*. O atributo *duration* possui o valor da duração da apresentação, sendo passado como parâmetro no construtor da classe *HF_Timer*. O atributo *start_time* corresponde ao tempo de início da apresentação, sendo obtido do relógio do sistema assim que a *thread* inicia a sua execução[18]. O atributo *elapsed_time* é iniciado com zero, sendo utilizado para controlar o cronômetro quando são executadas ações NCM *suspende* ou *reassume* sobre a apresentação, conforme será visto mais adiante nesta seção.

O mecanismo de funcionamento do cronômetro é a princípio bastante simples. Se não houver nenhuma ação NCM durante a apresentação, o cronômetro dorme (sai da fila de execução) durante o tempo especificado no atributo *duration*. Ao acordar, ele avisa à ferramenta associada o término da duração especificada e termina a sua execução logo em seguida.

[18] Uma outra interpretação do atributo *start_time* será dada mais adiante, quando for apresentado o comportamento do cronômetro em resposta à ação NCM *resume*.

O comportamento do cronômetro em resposta às ações NCM permite controlar o tempo decorrido da apresentação, quando esta é suspensa ou reiniciada. Quando a ferramenta de exibição recebe do formatador uma ação NCM *suspende*, ela ativa um *flag* indicador de pausa no cronômetro e acorda (recoloca na fila de execução) a *thread*. Neste momento, o cronômetro calcula o tempo decorrido (*elapsed_time*) da seguinte forma:

```
elapsed_time = elapsed_time + current_time – start_time ,
```

onde *current_time* é o tempo corrente obtido do relógio do sistema e *start_time* é tempo de início (ou reinício) da apresentação. Em seguida, a *thread* dorme por um tempo indeterminado.

Quando a apresentação é reiniciada por uma ação NCM *reassume* sobre a ferramenta de exibição, esta ativa um *flag* indicador de reinício no cronômetro e acorda a *thread*. Neste caso, o cronômetro recalcula o atributo *start_time*, atribuindo-lhe o valor do tempo corrente, ou seja, o tempo em que a apresentação foi reiniciada. Além disso, ele calcula o novo tempo em que a *thread* irá dormir (*sleep_time*), que deve ser igual ao tempo restante para atingir a duração especificada:

```
sleep_time = duration - elapsed_time .
```

Desta forma, o cronômetro da apresentação funciona como se fosse um cronômetro digital, podendo ser suspenso e reiniciado a qualquer momento, sem perder o controle do tempo decorrido para sinalizar o término da duração especificada.

As ações NCM *termina* e *aborta* finalizam a execução do cronômetro sem que este gere a notificação de término da duração para a ferramenta de exibição.

4.3.2.2 Exibição de imagens estáticas

A ferramenta de exibição de imagens estáticas permite a apresentação de objetos de mídia do tipo imagem cujo arquivo de dados é identificado por uma URL. Esta ferramenta possui as seguintes funcionalidades:

- Apresentação de âncoras espaciais;
- Redimensionamento automático da imagem (ou de sua âncora espacial) em caso de alteração das dimensões da janela de apresentação.
- Eventos de seleção de âncoras espaciais;
- Tempo de apresentação especificado por uma duração.

A Figura 36 apresenta o diagrama de classes das ferramentas de exibição de imagens estáticas (*HF_ImagePlayer*). Cada instância desta ferramenta possui um objeto da classe *ImagePanel*, criada como uma derivação da classe *JPanel* do pacote *javax.swing*. A classe *ImagePanel* está associada a duas outras classes, ambas do pacote *java.awt*: *Image*, que representa imagens gráficas, e *MediaTracker*, responsável pelo carregamento dos dados da imagem. O objeto da classe *ImagePanel* é inserido no *display* da ferramenta de exibição, seja ela do tipo *janela* ou do tipo *painel*.

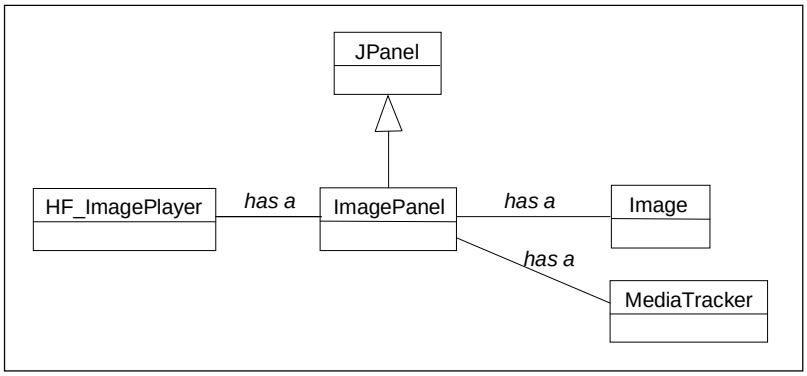

Figura 36 – Ferramenta de exibição de imagens estáticas

As características relativas à sincronização espacial nas ferramentas de exibição de imagens estáticas são semelhantes às apresentadas na Seção 4.3.1 para as ferramentas de exibição de vídeo. As informações das dimensões e posição são obtidas do descritor associado ao nó. Caso as dimensões não sejam especificadas no descritor, a ferramenta de exibição utiliza as dimensões intrínsecas do objeto, através da chamada aos métodos *getWidth* e *getHeight* do objeto da classe *Image*. O redimensionamento da imagem em conseqüência da alteração do tamanho da janela de apresentação comporta-se da mesma forma que nas ferramentas de exibição de vídeo, assim como a apresentação de âncoras espaciais (Seção 4.3.1.9).

Os eventos de seleção de âncoras espaciais das imagens estáticas são implementados da mesma forma que nas ferramentas de exibição de vídeo (Seção 4.3.1.11), ou seja, estes eventos são sinalizados ao formatador sempre que o usuário selecionar com o *mouse* um ponto do componente visual da mídia contido na região especificada pela âncora. Em particular, a âncora λ comporta-se como uma âncora espacial cuja região compreende todas as unidades de informação da imagem.

É de responsabilidade do cronômetro da apresentação a finalização da apresentação das imagens estáticas por esgotamento do tempo determinado pela duração, conforme visto na Seção 4.3.2.1.

Capítulo 5

Trabalhos Relacionados

5.1 Conversão da linguagem SMIL para o sistema Madeus

Madeus [JLRST98] é um sistema de autoria e apresentação de documentos multimídia pertencente ao projeto Opera, desenvolvido no instituto de pesquisa francês INRIA (*Institute National de Recherche en Informatique et en Automatique*). Seu modelo baseia-se no conceito de restrições (*constraints*) temporais e espaciais para especificar relações de sincronização entre os objetos de um documento, porém oferece também composições para sua estruturação lógica.

O relatório [Coll98] descreve um trabalho semelhante ao que foi apresentado no Capítulo 3 para o modelo NCM. Ele apresenta um conversor da linguagem SMIL para o modelo conceitual do sistema Madeus, que permite importar documentos SMIL para o ambiente de autoria e apresentação deste modelo. Esta conversão tem como principal objetivo a utilização do ambiente do Madeus para apresentar documentos multimídia escritos em linguagem SMIL e localizar possíveis inconsistências através dos algoritmos de detecção de inconsistências implementados no sistema. O relatório também faz uma análise do modelo temporal da linguagem SMIL e uma comparação entre esta linguagem e o sistema Madeus em relação ao poder de expressão. Entretanto, o trabalho concentra-se nos

aspectos temporais da linguagem SMIL, em detrimento das relações espaciais, dos hiperelos e das composições *switch*.

5.2 Extensões ao SMIL: a versão 2.0

O trabalho apresentado nesta dissertação concentrou-se na versão 1.0 da linguagem SMIL, cuja recomendação [SMIL98] foi publicada em junho de 1998. Atualmente, está em fase final de especificação a versão 2.0 da linguagem, anteriormente chamada de SMIL Boston. Nesta seção, foi utilizada a publicação da última chamada do *working draft* de SMIL 2.0, de setembro de 2000 [SMIL00b].

A versão 2.0 da linguagem SMIL está organizada em um conjunto de módulos específicos, permitindo que a sintaxe e a semântica dos seus elementos e atributos sejam reutilizados em outras linguagens baseadas no padrão XML [XML98], em particular aquelas que precisam representar relações de sincronização temporal. Cada módulo pode ser incorporado de forma independente dos outros, gerando uma grande flexibilidade na reutilização dos componentes. Por exemplo, alguns módulos de SMIL 2.0 foram utilizados para integrar relações temporais à linguagem XHTML [XHTML00], dando origem à linguagem HTML+SMIL, ainda em fase de especificação [SMIL00a].

A nova versão apresenta uma série de extensões à versão 1.0, aumentando o poder de expressão da linguagem. Algumas das novas funcionalidades presentes na nova versão podem ser encontradas no modelo NCM, enquanto outras poderiam ser acrescentadas a este modelo para aumentar o seu poder de expressão e facilitar a autoria, assim como foi feito nesta dissertação com a versão 1.0.

Em relação à sincronização temporal, SMIL 2.0 oferece uma maior flexibilidade para especificar os tempos de início e término da exibição dos elementos. Estes atributos podem ser especificados não só como um intervalo de tempo em relação à composição à qual o elemento pertence, ou aos outros elementos desta composição, como em SMIL 1.0, mas também em relação a outros eventos que podem ocorrer durante a apresentação do documento: início ou término de um outro elemento qualquer do documento, início da n-ésima repetição de um outro elemento e seleção pelo usuário de um elemento visual

qualquer com o *mouse*. Os tempos de início e término podem ser calculados ainda em relação ao momento da exibição de marcações temporais em objetos de mídia que suportam esta facilidade. Todos estes relacionamentos de sincronização temporal podem ser expressos no NCM através de elos entre eventos de apresentação e seleção.

Os atributos de início e término também podem ser especificados em SMIL 2.0 como uma lista de eventos, de modo que o elemento seja iniciado ou terminado pela ocorrência de qualquer um dos eventos da lista. No NCM, este tipo de relacionamento é especificado através de um elo com múltiplos eventos de origem e cuja condição é formada por uma expressão lógica entre estes eventos baseada no operador *ou*.

É importante observar que a linguagem SMIL 2.0, assim como a sua primeira versão, especifica os relacionamentos de sincronização temporal nos próprios elementos (objetos de mídia ou composições), por meio de seus atributos. Desta forma, mesmo que a linguagem oferecesse algum mecanismo de reutilização destes elementos, não seria possível reutilizá-los sem a herança obrigatória dos seus relacionamentos. O NCM, por sua vez, especifica estes relacionamentos nos elos, de forma separada da especificação dos nós. Assim, um mesmo nó pode ser associado a elos distintos, dependendo do aninhamento de composições no qual ele está contido, o que permite a reutilização dos dados e das estruturas de apresentação de forma independente dos relacionamentos entre eles.

Há ainda em SMIL 2.0 a possibilidade de especificar um intervalo de tempo negativo em relação ao evento que determina o início ou o término. Quando o tempo de início resulta em um instante no passado no momento em que é calculado, o elemento inicia imediatamente, porém na posição do fluxo de mídia que ele estaria caso tivesse começado no instante calculado. Além disso, um elemento pode ser iniciado ou terminado por meio de chamadas a métodos definidos no modelo de objetos do documento [SMIL00c]. Em qualquer caso, há uma restrição que deve ser obedecida: um elemento só pode estar sendo exibido durante o intervalo de tempo que a composição que o contém também o estiver. Em outras palavras, os elementos estão sujeitos às restrições temporais de seus respectivos elementos pais.

A versão mais recente do NCM inclui a possibilidade de especificar um retardo negativo a ser aplicado uma condição, assim como em SMIL 2.0. As ações definidas no NCM sobre

os eventos de apresentação correspondem aos métodos definidos no modelo de objetos do documento da linguagem SMIL 2.0. Porém, ao contrário de SMIL 2.0, um nó NCM não está sujeito às restrições temporais da composição onde ele está inserido, pois ele pode ser exibido mesmo que o evento de apresentação da composição não esteja no estado *ocorrendo*. Caso seja desejada, esta restrição pode ser definida explicitamente pelo autor, através de condições que verificam o estado do evento de apresentação da composição.

SMIL 2.0 permite especificar valores mínimo e máximo para a duração total (ou seja, incluindo possíveis repetições). Estes valores não são utilizados em algoritmos de ajuste elástico do tempo de apresentação, mas representam apenas restrições adicionais a serem consideradas. Por exemplo, caso o valor mínimo para a duração total seja 10 segundos e o término, especificado como um evento de seleção, ocorrer antes de 10 segundos, a apresentação só será terminada 10 segundos após o início da apresentação, e não no momento em que a seleção foi efetuada.

No NCM, o valor mínimo da duração pode ser especificado por meio de uma condição que verifica, em todos os elos que terminam a apresentação, se o início da apresentação ocorreu há mais tempo que este valor, e de um atributo (com um evento de atribuição associado) para indicar se algum evento de origem destes elos ocorreu antes que o tempo de apresentação atingisse o valor mínimo, de modo que a apresentação seja finalizada no momento em que este valor for atingido. O valor máximo da duração pode ser especificado simplesmente através de um elo cuja condição verifica o início da apresentação com um retardo igual a este valor, e cuja ação termina a apresentação.

Enquanto um documento está sendo exibido, atrasos na rede e outros fatores podem interferir nos tempos de exibição das mídias. SMIL 2.0 provê um maior controle sobre estas ocorrências, permitindo aos autores definir quais elementos (objetos de mídia ou composições) devem permanecer em sincronismo forte com as respectivas composições que os contém, assim como o tempo de tolerância para o desvio. No NCM, o sincronismo forte entre dois ou mais objetos é especificado por meio de um elo de restrição que determina que estes objetos iniciam e terminam ao mesmo tempo.

SMIL 2.0 define um novo tipo de composição: o elemento *excl*. Este elemento especifica que apenas um dos componentes nele contido pode estar sendo apresentado em um dado

momento. Se algum componente tentar iniciar enquanto outro está sendo exibido, vários comportamentos podem ser definidos: o primeiro elemento é finalizado e o novo elemento inicia a sua exibição; o primeiro elemento é interrompido, aguardando para ser reiniciado quando o novo elemento terminar; o novo elemento aguarda o primeiro terminar; ou simplesmente o novo elemento não é exibido. Uma aplicação do elemento *excl* seria, por exemplo, a apresentação de um conjunto de músicas, uma por vez, onde o usuário troca a música selecionando o seu nome com o *mouse*.

O elemento *excl* de SMIL 2.0 não possui um equivalente no NCM, porém, assim como os elementos *par* e *seq*, ele pode ser expresso por meio de nós de contexto de usuário e elos. As condições destes elos devem verificar o início da apresentação dos nós correspondentes aos componentes do elemento *excl*, de modo que somente um possa estar sendo exibido em um dado instante. A exemplo das composições paralelas e seqüenciais, o elemento *excl* poderia ter uma representação direta no NCM, através de uma especialização do nó de contexto de usuário. Esta extensão do NCM seria útil para facilitar a autoria deste tipo de relacionamento expresso pelo elemento *excl*.

Outras facilidades relacionadas à sincronização temporal consistem na manipulação da base de tempo. SMIL 2.0 permite alterar a velocidade da exibição, bem como especificar efeitos de aceleração e desaceleração. A nova versão também especifica uma série de efeitos visuais para a transição entre objetos exibidos em uma mesma região, como, por exemplo, uma apresentação de *slides*.

No NCM, a velocidade da exibição pode ser manipulada através de um atributo *velocidade* no descritor dos objetos de mídia contínua (áudio e vídeo), e de um evento de atribuição para alterar o seu valor. Os efeitos de aceleração e desaceleração ainda não são suportados pelo NCM, mas uma forma de suportá-los futuramente é através da especificação de funções temporais para o atributo *velocidade*. Os efeitos visuais para a transição entre objetos exibidos em uma mesma região não podem ser expressos no NCM. Esta facilidade poderia ser acrescentada ao modelo, porém é necessário um estudo mais profundo para definir a forma de expressão.

Os módulos de animação de SMIL 2.0 permitem a variação de valores de atributos no momento da exibição, de acordo com uma função do tempo. Um exemplo característico

desta funcionalidade é a variação da posição ou das dimensões de um objeto de mídia. Os valores dos atributos também podem ser alterados por interação do usuário no momento da exibição. Desta forma, podem ser especificadas animações bastante complexas, envolvendo simultaneamente eventos temporais e de interação do usuário.

No NCM, os eventos de atribuição podem ser usados como destinos de elos para alterar os valores dos atributos dos descritores durante a exibição, como por exemplo a posição e as dimensões do objeto. Uma possível extensão do NCM seria suportar a especificação de funções temporais para estes atributos, de modo a permitir a animação dos objetos de uma forma contínua. A especificação de mudanças de comportamento nos descritores também poderia ser usada para este fim, porém ela ainda requer uma maior formalização.

Os elos de navegação em SMIL 2.0 ainda estão restritos a uma âncora de origem e uma âncora de destino, porém apresentam algumas novas funcionalidades. A nova versão permite alterar o volume do áudio da origem e do destino quando o elo é percorrido, além de definir a região onde o documento de destino será exibido. Âncoras especificadas internamente a um documento HTML também poder ser usadas como âncoras de origem de elos de navegação em um documento SMIL 2.0, através de referências a elementos *a* ou *area* definidos no próprio documento HTML.

No NCM, a alteração do volume do áudio dos documentos de origem e de destino pode ser especificada através de ações sobre eventos de atribuição associados ao atributo do descritor que representa o volume do áudio. A região de apresentação do documento de destino pode ser especificada no descritor do ponto terminal de destino do elo. Entretanto, o NCM não permite que âncoras especificadas internamente a um documento HTML sejam utilizadas em seus elos. Para isto, seria necessário especificar um meio de identificar estas âncoras nos nós de conteúdo do tipo HTML, possivelmente através de um atributo da própria âncora.

Em relação à sincronização espacial, SMIL 2.0 permite que um documento seja apresentado em múltiplas janelas, com suas respectivas regiões associadas. Além disso, as regiões podem ser aninhadas, ou seja, definidas dentro de outras regiões. O posicionamento dos objetos dentro das regiões também está mais flexível, com a definição de atributos espaciais (posição e dimensões) no próprio objeto de mídia, relativos à região

à qual ele está associado. Os objetos podem ainda ser alinhados em relação a pontos predefinidos na região, permitindo, por exemplo, que eles sejam centralizados dentro da região.

O modelo NCM não possui meios de associar regiões de apresentação a janelas distintas nem permite aninhar regiões. Conforme apresentado no Capítulo 4, há apenas duas alternativas de apresentação: todas as regiões são inseridas em uma mesma janela principal, ou cada região é associada a uma janela distinta, sendo a escolha feita pelo usuário na plataforma de exibição. Uma possível extensão ao modelo NCM seria permitir a definição de regiões compostas, que conteriam recursivamente outras regiões. As regiões mais externas, ou seja, as que não estivessem contidas em nenhuma outra região, seriam exibidas como uma janela independente, onde seriam apresentados todos os nós cujos descritores especificassem uma das regiões recursivamente contidas nesta região mais externa. O descritor também poderia conter atributos para posicionar a apresentação no nó em relação à região especificada, oferecendo a flexibilidade encontrada em SMIL 2.0.

No que se refere aos objetos de mídia, houve poucas alterações em relação à versão 1.0 da linguagem SMIL. Basicamente, a nova versão permite a passagem de parâmetros para objetos de mídia (principalmente animações) e a definição de âncoras a partir de marcações temporais feitas no próprio objeto de mídia (por exemplo, o início de cada música em um CD). Em ambos os casos, os objetos de mídia devem suportar as funcionalidades mencionadas.

No NCM, a passagem de parâmetros para os objetos de mídia poderia ser realizada através de novos atributos nos descritores associados aos nós de conteúdo, de acordo com o tipo de mídia a ser exibido. A utilização de marcações temporais feitas nos próprios objetos de mídia requer uma forma de identificar estas marcações, possivelmente através de atributos nas âncoras NCM, a exemplo da identificação das âncoras HTML citadas anteriormente nesta seção.

SMIL 2.0 também provê uma maior flexibilidade na especificação de comportamentos alternativos (elemento *switch*). Foram definidos novos atributos de teste em relação às características do sistema, havendo ainda a possibilidade de inclusão de atributos definidos pelo próprio autor. No NCM, estes novos atributos de teste poderiam ser acrescentados à

especificação da plataforma de exibição e aos descritores, assim como foi feito com os atributos de teste de SMIL 1.0 (ver Seção 3.9).

Uma outra funcionalidade oferecida pela nova versão da linguagem SMIL é a solicitação explícita da busca dos dados a serem exibidos, através do elemento *prefetch*. São especificadas a quantidade de dados (em tempo ou em bytes) a serem carregados e a largura de banda da rede a ser utilizada. No NCM, esta funcionalidade já é oferecida através da ação *prepara* sobre um evento de apresentação. A quantidade de dados a serem carregados e a largura de banda a ser utilizada não são explicitamente especificadas pelo autor, mas podem ser calculadas pelo formatador através de algoritmos de *prefetch* dos dados, para otimizar a qualidade da apresentação. Estes algoritmos também calculam o momento ideal para o início do processo de *prefetch*.

Observa-se que algumas das limitações da versão 1.0 da linguagem apontadas no Capítulo 2 foram superadas, enquanto outras permanecem na nova versão. SMIL 2.0 permite definir relacionamentos temporais mais complexos entre os componentes do documento, porém ainda não possui o poder de expressão do NCM neste aspecto. As especificações de mudanças de comportamento estão suportadas na nova versão, através dos módulos de animação, que permitem a alteração de valores de atributos durante a apresentação do documento, causada por eventos síncronos ou por interação do usuário.

Por outro lado, a estrutura lógica do documento ainda está intimamente relacionada com a estrutura de apresentação, pois ainda não há composições de contexto sem uma semântica de apresentação associada. A especificação da duração dos objetos não permite a definição de funções de custo a serem utilizadas para ajustar o tempo da apresentação em resposta a eventos imprevisíveis. As características temporais da apresentação ainda são especificadas nos próprios componentes, impedindo a reutilização destas especificações em outros componentes ou a associação de características distintas a um mesmo componente, dependendo, por exemplo, da navegação feita pelo usuário. Também não há a facilidade de reutilização das estruturas de apresentação através de referências, e nem de associar diferentes conjuntos de elos a um mesmo componente, dependendo do aninhamento de composições no qual ele está contido.

5.3 Ferramentas de exibição de documentos multimídia

Existem atualmente várias ferramentas para exibição de documentos multimídia com sincronização temporal e espacial. Algumas delas são específicas para documentos escritos em linguagem SMIL (GRiNS Player, SOJA), enquanto outras já existiam e passaram a suportar a linguagem SMIL em suas versões mais recentes (RealPlayer, QuickTime). Há ainda o navegador Internet Explorer, que passou a suportar alguns módulos de SMIL 2.0 em sua versão 5.5.

O aplicativo GRiNS Player (*Graphical Interface to SMIL*) [Grins00] foi desenvolvido pela empresa holandesa Oratrix, que tem suas raízes no centro de pesquisas CWI. Atualmente na versão 2.0 beta, este aplicativo suporta a maioria dos elementos da linguagem SMIL 2.0, antes mesmo de ela se tornar uma especificação do W3C. O GRiNS Player é atualizado com as novas funcionalidades que surgem ao longo das especificações SMIL, desde a versão 1.0, permitindo que a comunidade científica avalie na prática estas especificações e colabore com novas sugestões.

Apesar de ser a ferramenta de exibição de documentos SMIL mais atualizada, o GRiNS Player possui algumas limitações em relação à sincronização entre sub-regiões temporais e espaciais. A ferramenta não permite sincronizar o início ou o término de um objeto em relação ao início ou ao término de uma âncora temporal interna a um outro elemento que está sendo exibido, mas apenas se esta for exatamente a âncora que está sendo exibida, definida através de atributos de corte *clip-begin* e *clip-end* (ver Capítulo 2). No que se refere aos elos de navegação (disparados por interação do usuário), a ferramenta aceita a definição de âncoras temporais, espaciais e espaço-temporais como origem destes elos. No entanto, estas âncoras não podem ser usadas como destino dos elos, pois a ferramenta sempre considera o objeto inteiro como destino (mais uma vez, exceto no caso das âncoras temporais definidas através de atributos de corte). Além disso, as dimensões espaciais das regiões não são ajustadas quando o usuário altera o tamanho da janela, mesmo que elas estejam definidas como um percentual da janela principal.

O aplicativo SOJA (*SMIL Output in a Java Applet*) [Soja99], desenvolvido pela organização francesa Helio, permite apresentar documentos SMIL em navegadores comuns

através de um *applet* Java. A versão atual, SOJA Cherbourg 2, ainda é bastante limitada. Além de não suportar objetos de vídeo, ela implementa apenas uma parte da versão 1.0 da linguagem. O aplicativo não exibe sub-regiões temporais ou espaciais, e suporta apenas uma navegação simples, onde um outro documento inteiro (HTML ou SMIL) pode ser iniciado a partir de uma seleção do usuário sobre um texto ou uma imagem em exibição.

O RealPlayer [Real00] é uma ferramenta de exibição desenvolvida pela empresa RealNetworks, sendo bastante utilizado para a apresentação de mídias distribuídas na Internet. A partir da versão G2, esta ferramenta passou a suportar a linguagem SMIL para exibir simultaneamente diversos tipos de mídias, em uma única apresentação sincronizada. Atualmente na versão 8, o RealPlayer implementa a versão 1.0 da linguagem SMIL. Esta ferramenta apresenta as mesmas limitações do GRiNS em relação à sincronização entre sub-regiões temporais e espaciais. Além disso, não implementa âncoras de origem de elos de navegação em objetos de vídeo (ao menos em vídeo MPEG). Em relação à sincronização espacial, a ferramenta não ajusta o tamanho das imagens e dos vídeos ao tamanho da região associada. Por outro lado, as dimensões da mídia são ajustadas sempre que o usuário altera o tamanho da janela principal da apresentação, mantendo as proporções relativas a ela.

O QuickTime [Qtime00] foi desenvolvido pela Apple. Assim como o RealPlayer, esta ferramenta também é bastante utilizada para apresentar objetos de mídia distribuídos na Internet. Em sua versão 4.1, o QuickTime passou a suportar a versão 1.0 linguagem SMIL, inclusive definindo algumas extensões através de novos atributos para os seus elementos. Estas extensões permitem, entre outras coisas, especificar se um documento deve aguardar o comando do usuário para iniciar sua apresentação, ou ainda informar a taxa de transmissão de um vídeo em tempo real, para que a ferramenta calcule o volume de dados a serem carregados antes de iniciar a exibição, de acordo com a largura de banda disponível na rede. As limitações do QuickTime em relação à sincronização entre âncoras são semelhantes às do GRiNS. Assim como o RealPlayer, o QuickTime não ajusta o tamanho das imagens e dos vídeos ao tamanho da região associada, porém as dimensões da mídia são ajustadas sempre que o usuário altera o tamanho da janela principal da apresentação, mantendo as proporções relativas a ela.

O navegador Microsoft Internet Explorer, em sua versão 5.5, é compatível com a linguagem HTML+TIME [TIMEH98]. De acordo com [Newm00], o HTML+TIME (já na versão 2.0) é a implementação realizada pela Microsoft da linguagem HTML+SMIL [SMIL00a]. O Internet Explorer 5.5 inclui suporte aos módulos de sincronização temporal, animação e objetos de mídia do SMIL 2.0. As âncoras e os elos da linguagem SMIL não estão incluídos na linguagem HTML+TIME, visto que existem elementos HTML que oferecem funcionalidades semelhantes.

Capítulo 6

Conclusão

6.1 Contribuições da dissertação

As principais contribuições desta dissertação foram: a comparação entre a linguagem SMIL e o modelo NCM; o mapeamento SMIL-NCM; a implementação de um conversor de documentos SMIL para o NCM; a definição e a implementação de extensões ao modelo NCM; a implementação de ferramentas de exibição para os objetos de um documento NCM; e a definição de regras de comportamento para os eventos de apresentação NCM das âncoras internas a um objeto que está sendo apresentado.

A comparação entre a linguagem SMIL e o modelo NCM permitiu identificar as limitações da linguagem que poderiam ser superadas através de um mapeamento para o modelo NCM. A possibilidade de acrescer facilidades NCM a documentos SMIL foi a principal motivação para a realização do mapeamento e da conversão SMIL-NCM.

O mapeamento da linguagem SMIL para o modelo NCM mostrou como as relações de sincronização temporal e espacial definidas na linguagem SMIL podem ser representadas em um modelo conceitual multimídia baseado em eventos. Toda a semântica especificada para as composições paralelas e seqüenciais da linguagem, além dos atributos que definem

com mais precisão a sincronização temporal entre os diversos objetos de mídia de um documento SMIL, foram mapeados em nós, elos e descritores do modelo NCM. Além disso, foram propostas duas formas de implementar a composição *switch* da linguagem, utilizada para selecionar em tempo de execução os objetos a serem exibidos. Uma destas formas é mais simples, porém não atende a todos os casos previstos na linguagem, enquanto a outra, embora mais complexa, representa a semântica da composição *switch* em todos os casos. Esta segunda forma permitiu exercitar a utilização da lista de operações da especificação de iniciação do descritor NCM, até então pouco discutida. Por fim, a sincronização espacial, os objetos de mídia, as âncoras e os hiper-elos da linguagem SMIL também foram representados no modelo NCM.

A implementação do conversor de documentos SMIL para o modelo NCM foi uma conseqüência do processo de mapeamento. A sua função é converter documentos SMIL, especificados em linguagem declarativa, em documentos NCM, integrando-os ao sistema HyperProp. Esta conversão permite refinar os documentos SMIL através das facilidades oferecidas pelo modelo NCM e dos recursos de autoria implementados no sistema HyperProp, além de utilizar o formatador do sistema HyperProp para apresentar documentos SMIL. O conversor foi implementado em Java e totalmente integrado ao sistema.

A utilidade da definição de extensões ao modelo NCM foi descoberta durante o próprio processo de mapeamento. A representação das composições paralelas e seqüenciais da linguagem SMIL foi possível com as classes do modelo já existentes, porém mostrou-se bastante complexa devido à quantidade de elos envolvidos. Com o objetivo de facilitar a autoria destas composições no modelo NCM, a classe nó de contexto de usuário foi especializada em duas novas classes: a composição paralela e a composição seqüencial, com as mesmas semânticas das composições SMIL homônimas. Outras extensões, como a coleção de apresentação, para especificar os descritores associados a um nó de contexto, e a definição de novos atributos nos descritores e nos eventos de apresentação, mostraram-se úteis para aumentar o poder de expressão do NCM. Além disso, alguns atributos da plataforma de exibição foram definidos a partir de conceitos presentes na linguagem SMIL, para representar características do sistema, como a largura de banda disponível na rede, e preferências do usuário, como o idioma. Estas extensões permitem ainda que o

ambiente de autoria implementado no sistema HyperProp possa ser utilizado para criar e editar documentos SMIL.

A implementação de ferramentas de exibição para o sistema HyperProp permitiu a apresentação de documentos NCM que apresentassem relacionamentos de sincronização temporal e espacial entre os seus componentes, além das próprias relações de interação do usuário. Estas ferramentas obedecem à interface definida pelo executor do sistema, sinalizando para ele os eventos que ocorrem durante a apresentação e realizando as ações por ele solicitadas, além de estabelecerem a sincronização espacial especificada no documento. Foram desenvolvidas ferramentas de exibição de mídias contínuas (áudio e vídeo), com o suporte do Java Media Framework, e ferramentas de exibição de imagens estáticas. Os conceitos relativos à sincronização entre os objetos também foram aplicados às ferramentas de exibição de nós de texto, que já haviam sido desenvolvidas.

Uma importante contribuição do desenvolvimento das ferramentas de exibição foi o monitor de âncoras temporais, que permite detectar os eventos de início e término da exibição de âncoras temporais internas ao objeto de mídia contínua que está sendo exibido, para sincronizar com outros objetos do documento. A precisão do momento em que estes eventos são sinalizados pode ser ajustada em relação à quantidade de recursos de processamento que serão utilizados, ou seja, quanto maior a precisão desejada, maior será a utilização de recursos de processamento.

Uma outra característica destas ferramentas é a capacidade de exibir uma sub-região espacial de mídias de imagem e vídeo, inclusive efetuando os ajustes das dimensões em resposta às ações do usuário de redimensionamento da janela de apresentação.

Por fim, o processo de desenvolvimento destas ferramentas de exibição levou à definição de regras de comportamento para os eventos de apresentação NCM de âncoras temporais, espaciais e espaço-temporais de um objeto de mídia. Estas regras definem quais eventos de apresentação de âncoras devem mudar de estado e em que momento, quando um objeto de mídia está sendo exibido.

6.2 Propostas de trabalhos futuros

Durante o mapeamento dos elementos da linguagem SMIL para o NCM e a definição de extensões para o modelo, percebeu-se a utilidade de representar a semântica do elemento *switch* da linguagem SMIL através de entidades virtuais. Desta forma, a escolha dos elementos a serem exibidos poderia ser determinada em tempo de execução, através de consultas realizadas em relação aos atributos da especificação da plataforma. Esta representação do elemento *switch* através de entidades virtuais é uma proposta de trabalho futuro.

Com a finalização da versão 2.0 da linguagem SMIL, novas funcionalidades estão surgindo em relação à versão anterior. Desta forma, como trabalho futuro, estas funcionalidades poderiam ser comparadas com o modelo NCM, para verificar quais delas já estão suportadas e quais ainda não estão. A funcionalidades suportadas podem ser mapeadas no modelo NCM, enquanto as não suportadas podem ser incluídas no modelo através da definição de novas extensões, conforme apresentado na Seção 5.2.

Em relação às ferramentas de exibição, um trabalho futuro seria um estudo para verificar a possibilidade de unificação das bases de tempo das diversas ferramentas. Atualmente, cada ferramenta de exibição possui a sua própria base de tempo. Uma base de tempo unificada permitiria oferecer ao usuário uma barra de tempo deslizante (para ajuste do ponto de exibição) única para todo o documento, de forma que ele pudesse alterar o ponto de exibição do documento como um todo, e não apenas de cada ferramenta individualmente.

As ferramentas de exibição desenvolvidas nesta dissertação podem ser de dois tipos, quanto à forma de apresentação espacial: janela ou painel. As ferramentas do tipo *janela* provêem uma janela de exibição independente para cada objeto do documento, enquanto as ferramentas do tipo *painel* caracterizam-se pelo fato de serem inseridas em uma janela única. A escolha do tipo de ferramenta a ser utilizada é feita na especificação da plataforma de exibição, sendo independente da especificação do documento. Desta forma, o usuário pode escolher entre duas opções: ou cada ferramenta dá origem a uma janela distinta, ou todas as ferramentas são incluídas em uma mesma janela. Como trabalho futuro, propõe-se que o tipo de ferramenta possa ser especificado dentro do próprio documento, de forma

independente para cada objeto a ser exibido. Isto permitirá definir conjuntos de objetos de forma que os objetos de um mesmo conjunto sejam associados a uma mesma janela, e cada conjunto seja associado a uma janela distinta, resultando em uma maior flexibilidade na forma de apresentação dos objetos de um documento.

Referências Bibliográficas

[AnSo00] Antonacci, M.J.; Soares, L.F.G. "Especificação X-SMIL". *Relatório Técnico do Laboratório Telemídia, Departamento de Informática da PUC-Rio*, Rio de Janeiro, Fevereiro 2000.

[ARMS00] Antonacci, M.J.; Rodrigues, R.F.; Muchaluat-Saade, D.C.; Soares, L.F.G. "NCL: Uma Linguagem Declarativa para Especificação de Documentos Hipermídia na Web". *Anais do VI Simpósio Brasileiro de Sistemas Multimídia e Hipermídia*, Natal, Julho 2000.

[BuZe93] Buchanan, M.C.; Zellweger, P.T. "Automatically Generating Consistent Schedules for Multimedia Documents". *Multimedia Systems*, Springer-Verlag, Abril 1993.

[Coll98] Collomb, P. "De Smil à Madeus : deux langages de synchronisation de documents multimédia". *Rapport de Maîtrise Informatique de l' Université Joseph Fourier (Grenoble 1)*, Setembro 1998.

[CoLR90] Cormen, T.H.; Leiserson, C.E.; Rivest, R.L. "Introduction to Algorithms". *MIT Press*, 1990.

[Grins00] "GRiNS Player beta for SMIL 2.0". *Oratrix Development BV*, Setembro 2000. Disponível em: http://www2.oratrix.nl/W3C/index_html

[JLRST98] Jourdan, M.; Layaïda, N.; Roisin, C.; Sabry-Ismail, L.; Tardif, L. "Madeus, an Authoring Environment for Interactive Multimedia Documents". *Proceedings of the ACM Multimedia Conference 98*, pp. 267-272, Inglaterra, Setembro 1998.

[JMF99a] "Java Media Framework API Guide". *Sun Microsystems*, Novembro 1999. Disponível em: http://java.sun.com/products/java-media/jmf/2.1/specdownload.html

[JMF99b] "Java Media Framework, v2.0 API Specification". *Sun Microsystems*, 1999. Disponível em: http://java.sun.com/products/java-media/jmf/2.1/specdownload.html

[KiSo95] Kim, M.Y.; Song J. "Multimedia Documents with Elastic Time". *Proceedings of the ACM Multimedia Conference 95*, San Francisco, Novembro 1995.

[Newm00] Newman, D. "Spice Up Your Web Pages with HTML+TIME". *Msdn Online Web Workshop*, Maio 2000. Disponível em: http://msdn.microsoft.com/workshop/Author/behaviors/htmltime.asp

[PeLi96] Pérez-Luque, M.J.; Little, T.D.C. "A Temporal Reference Framework for Multimedia Synchronization". *IEEE Journal on Selected Areas in Communications – Special Issue: Synchronization Issues in Multimedia Communication*, Vol. 14, No. 1, pp. 36-51, Janeiro 1996.

[Qtime00] "QuickTime 4.1". *Apple Technical Publications*, Janeiro 2000. Disponível em: http://developer.apple.com/techpubs/quicktime/qtdevdocs/PDF/QuickTime41.pdf

[Real00] "Real Player 8 Plus User Manual". *RealNetworks*, 2000. Disponível em: http://service.real.com/help/player/plus_manual.8/rppmanual.htm

[Rodr97] Rodrigues, R.F. "Formatação Temporal e Espacial no Sistema HyperProp". *Dissertação de Mestrado, Departamento de Informática da PUC-Rio*, Rio de Janeiro, Maio 1997.

[Rodr99a] Rodrigues, L.M. "Conversor de Documentos SMIL para o Modelo NCM". *Projeto Final de Programação, Departamento de Informática da PUC-Rio*, Rio de Janeiro, Dezembro 1999.

[Rodr99b] Rodrigues, R.F. "Projeto OO do Formatador Temporal e Espacial do Sistema HyperProp". *Relatório Técnico, Departamento de Informática da PUC-Rio*, Rio de Janeiro, Dezembro 1999.

[RoMS98] Rodrigues, R.F.; Muchaluat-Saade, D.C.; Soares, L.F.G. "Composite Nodes, Contextual Links and Graphical Structural Views on the WWW". *Special Issue on World Wide Web of the Journal of the Brazilian Computer Society (JBCS)*, Vol. 5, No. 2, pp. 31-44, Novembro 1998.

[RRMS99] Rodrigues, L.M.; Rodrigues, R.F.; Muchaluat-Saade, D.C.; Soares, L.F.G. "Improving SMIL Documents with NCM Facilities". *Proceedings of the International Conference on Multimedia Modeling (MMM'99)*, pp. 19-38, Ottawa, Outubro 1999.

[SMIL98] "Synchronized Multimedia Integration Language (SMIL) 1.0 Specification". *W3C Recommendation*, Junho 1998. Disponível em: http://www.w3.org/TR/REC-smil/

[SMIL00a] "Synchronized Multimedia Integration Language (SMIL) Boston Specification". *W3C Working Draft*, Junho 2000. Disponível em: http://www.w3.org/TR/2000/WD-smil-boston-20000622/

[SMIL00b] "Synchronized Multimedia Integration Language (SMIL 2.0) Specification". *W3C Working Draft*, Setembro 2000. Disponível em: http://www.w3.org/TR/2000/WD-smil20-20000921/

[SMIL00c] "Synchronized Multimedia Integration Language Document Object Model (DOM)". *W3C Working Draft*, Fevereiro 2000. Disponível em: http://www.w3.org/TR/2000/WD-smil-boston-dom-20000225/

[Soar00] Soares, L.F.G. "Modelo de Contextos Aninhados – Versão 2.3". *Relatório Técnico do Laboratório Telemídia, Departamento de Informática da PUC-Rio*, Rio de Janeiro, 2000.

[SoCR95] Soares, L.F.G.; Casanova, M.A.; Rodriguez, N.L.R. "Nested Composite Nodes and Version Control in an Open Hypermedia System". *International Journal on Information Systems – Special Issue on Multimedia Information Systems*, Vol. 20, No. 6, pp. 501-519, Elsevier Science Ltd, Inglaterra, 1995.

[Soja99] "SOJA Cherbourg 2". *Helio*, 1999. Disponível em: http://www.helio.org/products/smil/

[SoMR98] Soares, L.F.G.; Muchaluat-Saade, D.C.; Rodrigues, R.F. "Authoring and Formatting of Hypermedia Documents". *Workshop on Multimedia Authoring Systems in the IEEE International Conference on Multimedia Computing and Systems*, Austin, Junho 1998.

[SSRM99] Soares, L.F.G.; Souza, G.L.; Rodrigues, R.F.; Muchaluat-Saade D.C. "Versioning Support in the HyperProp System". *Multimedia Tools and Applications – An International Journal*, Kluwer Academic Publishers, Vol. 8, No. 3, Maio 1999.

[TIMEH98] "Timed Interactive Multimedia Extensions for HTML (HTML+TIME)". *W3C Note*, Setembro 1998. Disponível em: http://www.w3.org/TR/1998/NOTE-HTMLplusTIME-19980918

[XHTML00] "The Extensible HyperText Markup Language: A Reformulation of HTML 4.0 in XML 1.0". *W3C Recommendation*, Janeiro 2000. Disponível em: http://www.w3.org/TR/xhtml1/

[XML98] "Extensible Markup Language (XML) 1.0". *W3C Recommendation*, Fevereiro 1998. Disponível em: http://www.w3.org/TR/REC-xml