PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Romualdo Monteiro de Resende Costa

Integração e Interoperabilidade de Documentos MPEG-4 e NCL

DISSERTAÇÃO DE MESTRADO

DEPARTAMENTO DE INFORMÁTICA
Programa de Pós-Graduação em Informática

Rio de Janeiro, abril de 2005

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Romualdo Monteiro de Resende Costa

Integração e Interoperabilidade de Documentos MPEG-4 e NCL

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio.

Orientador: Luiz Fernando Gomes Soares

Rio de Janeiro
Abril de 2005

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Romualdo Monteiro de Resende Costa

Integração e Interoperabilidade de Documentos MPEG-4 e NCL

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Luiz Fernando Gomes Soares
Orientador
Departamento de Informática - PUC-Rio

Prof. Markus Endler
Departamento de Informática - PUC-Rio

Prof. Rogério Ferreira Rodrigues
Departamento de Informática - PUC-Rio

José Eugênio Leal
Coordenador Setorial do Centro Técnico Científico - PUC-Rio

Rio de Janeiro, 06 de abril de 2005

Romualdo Monteiro de Resende Costa

Graduado em Ciência da Computação pela Universidade Federal de Minas Gerais (UFMG) em 2000. Desde 2002 é oficial de carreira do exército, na especialidade de informática. Atualmente, integra o grupo de pesquisadores do Laboratório TeleMídia da PUC-Rio, desenvolvendo pesquisas na área de Sistemas Hipermídia.

Ficha Catalográfica

Costa, Romualdo Monteiro de Resende

Integração e Interoperabilidade de Documentos MPEG-4 e NCL / Romualdo Monteiro de Resende Costa ; orientador: Luiz Fernando Gomes Soares. – Rio de Janeiro : PUC, Departamento de Informática, 2005.

180 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui referências bibliográficas.

1. Informática – Teses. 2. MPEG-4. 3. NCL. 4. Codificação audiovisual. 5. Sistemas hipermídia. 6. Integração. 7. Autoria. 8. Templates. 9. XMT-O. 10. XMT-A. 11. BIFS. I. Soares, Luiz Fernando Gomes. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

Este trabalho é dedicado

Á Aline, pelo amor e dedicação.

Á minha família, pelo apoio incondicional.

Agradecimentos

Ao meu orientador, professor Luiz Fernando Gomes Soares, agradeço a sua confiança, apoio e dedicação, presentes em todos os momentos, desde o primeiro dia de aula. Seus ensinamentos não se limitam apenas a este trabalho, serão importantes em toda a minha vida. Sinceramente obrigado.

Agradeço a toda a minha minha família e especialmente à Regina, Raquel e Raíssa. Agradeço também à Aline pela ajuda e compreensão dos momentos dedicados a este trabalho.

Agradeço a todos os amigos do Laboratório Telemídia, pois sem a ajuda de vocês este trabalho não seria possível. O ambiente de trabalho agradável nesse laboratório foi fundamental para recuperar o ânimo depois de uma manhã cansativa de trabalho.

Aos amigos agradeço pelo apoio, mesmo nas vezes em que me fiz ausente, tomado pelas tarefas diárias. Em especial agradeço aos amigos Lessandro e Bianca. Aos companheiros de longa data, amigos de Belo Horizonte, em especial Heitor, Daniel e Márcio. Agradeço também aos amigos, nem tão recentes, companheiros em Salvador e agora no Rio de Janeiro, Anderson e Cil Farne.

Ao professor Antônio Alfredo Ferreira Loureiro, outro mestre que tive a felicidade de conhecer, agradeço toda a orientação e ajuda. Agradeço também aos demais professores das Universidades Federais de Minas Gerais e Juiz de Fora.

Aos membros da banca pelos comentários pertinentes e pelas revisões precisas. Agradeço também a todos os professores e funcionários do Departamento de Informática da PUC-Rio, e à PUC-Rio, como instituição, pelo suporte financeiro.

Resumo

Costa, Romualdo Monteiro de Resende. **Integração e Interoperabilidade de Documentos MPEG-4 e NCL.** Rio de Janeiro, 2005. 180 p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A abordagem orientada a objetos do padrão MPEG-4, para a codificação de conteúdo audiovisual, é similar às utilizadas em vários modelos e linguagens de especificação de documentos multimídia/hipermídia. Entre essas linguagens, a NCL (*Nested Context Language*), utilizada no sistema HyperProp, introduz uma série de novos conceitos que podem ser integrados ao padrão, com vantagens. Esta dissertação propõe, inicialmente, a conversão de documentos especificados em NCL para MPEG-4 (XMT-O) e vice-versa, permitindo que ferramentas de autoria e formatação possam ser utilizadas na especificação e exibição de documentos de ambas as linguagens. Este trabalho também propõe a incorporação de cenas MPEG-4 tanto como objetos de mídia quanto composições da linguagem NCL, permitindo o estabelecimento de relacionamentos entre cenas. Para permitir a exibição desses novos objetos NCL, é incorporado ao Formatador HyperProp um exibidor MPEG-4 capaz de reportar ao controlador a ocorrência de eventos que, entre outras coisas, permite o sincronismo entre cenas MPEG-4 e outros objetos NCL, incluindo outras cenas MPEG-4. Por fim, explorando o conceito de templates introduzido pela linguagem NCL, a capacidade de autoria no MPEG-4 é estendida, através da definição de novas semânticas para as composições da linguagem XMT-O e da concepção de compiladores para essa linguagem.

Palavras-chave

MPEG-4; NCL; codificação audiovisual; sistemas hipermídia; integração; autoria; templates; XMT-O; XMT-A; BIFS.

Abstract

Costa, Romualdo Monteiro de Resende. **Integration and Interoperability of MPEG-4 and NCL Documents.** Rio de Janeiro, 2005. 180 p. Master Thesis – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The MPEG-4 standard object-oriented approach, employed to the encoding of audiovisual content, is similar to those used on many models and languages for multimedia/hypermedia document specification. Among those languages, the NCL (*Nested Context Language*), used in the HyperProp system, introduces a series of new concepts that can be integrated to the standard, with advantages. Initially, the proposal of this work is to convert NCL to MPEG-4 (XMT-O) documents and vice versa, allowing authoring and formatting tools to be used in the specification and presentation of documents in both languages. This work also proposes both the placing of MPEG-4 scenes as media objects and NCL language compositions, allowing the establishment of relationships among scenes. In order to allow displaying these new NCL objects, an MPEG-4 player is incorporated to the HyperProp Formatter. The MPEG-4 player is able to report to the controller the occurrence of events that, among other things, allows the synchronization between MPEG-4 scenes and other NCL objects, including other MPEG-4 scenes. Finally, exploring the concept of templates, introduced by the NCL language, the authoring in the MPEG-4 is improved, by means of the definition of new semantics for XMT-O language compositions and the design of compilers for this language.

Keywords

MPEG-4; NCL; audiovisual codification; hypermedia systems; integration; authoring; templates; XMT-O; XMT-A; BIFS.

Sumário

Lista de figuras

Lista de tabelas

1
Introdução

Um documento hipermídia é formado por um conjunto de informações relacionadas. Dessa forma, além do seu conteúdo (vídeo, áudio, texto, imagem etc.), um documento hipermídia agrega uma estrutura formada pelos relacionamentos definidos pelo autor, através de algum formato, estrutura esta tão importante quanto o próprio conteúdo das informações relacionadas.

O modelo de organização hipermídia não é uma proposição recente (Bush, 1945), entretanto, foi a partir do surgimento da WWW (*World Wide Web*) (Berners-Lee et al.,1994), no início dos anos 90, que esse paradigma para estruturação das informações tornou-se popular.

Atualmente, certos cenários, como os sistemas de TV digital interativa, têm despertado inúmeros desafios no desenvolvimento de sistemas hipermídia, motivando novos trabalhos de pesquisa diretamente relacionados a tecnologias aplicadas a modelos hipermídia, como o padrão MPEG-4 (Koenen, 2002), abordado nesta dissertação.

Este capítulo descreve, de forma sucinta, as características do MPEG-4 aplicadas no contexto de sistemas hipermídia, especificando os principais ambientes que compõem a arquitetura desses sistemas. Após essa breve contextualização, o capítulo descreve os objetivos desta dissertação e detalha a estrutura deste documento.

1.1.
Motivação

1.1.1.
O Formato MPEG-4

O MPEG (*Moving Picture Experts Group*) define uma família de padrões para codificação digital de informações de áudio e vídeo. Seus padrões iniciais, MPEG-1 (Chiariglione, 1996) e MPEG-2 (Chiariglione, 2000), focam principalmente em aspectos relacionados à codificação e decodificação dessas

mídias. Por outro lado, a codificação do MPEG-4, diferente da codificação linear de áudio e vídeo do MPEG 1 e 2, é baseada em objetos, isto é, as cenas audiovisuais são definidas em termos de objetos de mídia (vídeo, áudio, texto, imagem etc.) codificados separadamente.

A codificação audiovisual baseada em objetos, proposta pelo MPEG-4, possui algumas vantagens: a abordagem orientada a objeto permite aos autores reusarem material audiovisual; objetos podem ser codificados usando diferentes resoluções espaciais e temporais; técnicas de compactação e compressão podem ser aplicadas individualmente, de acordo com as características de cada mídia; e objetos sintéticos, que variam desde objetos simples de duas dimensões, como linhas e pontos, até objetos complexos de três dimensões, como animações faciais e corporais, podem fazer parte de uma apresentação, em conjunto com os demais objetos de mídia naturais[1].

Isoladamente, os objetos de mídia, resultantes da codificação do conteúdo audiovisual, não definem uma apresentação. Dado um conjunto qualquer de objetos de mídia, várias apresentações distintas podem ser definidas. Portanto, para que as vantagens da abordagem orientada a objeto sejam alcançadas na codificação do conteúdo audiovisual, faz-se necessário especificar informações complementares relativas à apresentação, como a forma de associação dos objetos e suas propriedades para exibição. Faz-se também necessário realizar a entrega síncrona dessas especificações, em conjunto com os objetos de mídia codificados, aos exibidores (receptores).

No MPEG-4 as informações relativas à apresentação, tais como os relacionamentos espaço-temporais, que definem a distribuição dos objetos de mídia em relação aos dispositivos de exibição e ao tempo da apresentação, e os relacionamentos interativos, que são aqueles associados a eventos acionados pela interação do usuário, encontram-se especificadas em documentos independentes da codificação do conteúdo.

Embora a especificação das informações relativas à apresentação seja independente da codificação do conteúdo dos objetos, no MPEG-4 elas são

---

[1] Objetos naturais são definidos como aqueles capturados por dispositivos tais como: câmeras, microfones etc; já os objetos sintéticos são produzidos por unidades computacionais (Koenen, 2002).

multiplexadas em conjunto. As informações relativas à apresentação são codificadas em um formato binário denominado BIFS (*BInary Format for Scenes*) (ISO/IEC, 2001). Esse formato favorece o armazenamento integrado da especificação dos relacionamentos com os objetos de mídia. O conteúdo de cada objeto, por sua vez, é individualmente codificado em um formato binário, de acordo com algoritmos específicos adotados na sua codificação (ISO/IEC, 2001). Na decodificação, o fluxo multiplexado permite aos exibidores realizar simultaneamente a decodificação do conteúdo e da estrutura, obtendo uma apresentação imediata, ainda que o documento multimídia/hipermídia seja parcialmente recebido.

Segundo Koenen (Koenen, 2002) os conceitos do MPEG-4, planejados como metas na definição do padrão, podem ser sintetizados em quatro itens:

- Representação através de objetos de mídia do conteúdo audiovisual de natureza natural ou sintética;
- Multiplexação e sincronização dos conteúdos dos objetos de mídia para transporte em redes que ofereçam garantias dos requisitos de qualidade de serviço (QoS) apropriados à natureza específica do conteúdo de cada objeto;
- Possibilidade de composição dos objetos de mídia, criando objetos compostos até o nível de formação de uma cena audiovisual;
- Interação no exibidor final (exibidor da apresentação) com a cena audiovisual gerada a partir da composição dos objetos de mídia.

1.1.2. MPEG-4 e os Sistemas Hipermídia

Os conceitos do padrão MPEG-4 podem ser abordados no contexto de sistemas hipermídia, que definem uma arquitetura normalmente formada por três ambientes: o ambiente de autoria; de armazenamento; e de execução de documentos. A esses ambientes correspondem as diferentes fases usualmente envolvidas no processo de produção de documentos multimídia/hipermídia. A Figura 1.1 ilustra a definição de um sistema hipermídia completo, onde os ambientes estão localizados em estações distintas. Num sistema hipermídia distribuído, como na Figura 1.1, a troca de informações entre seus ambientes tem por suporte um provedor de serviços de comunicações.

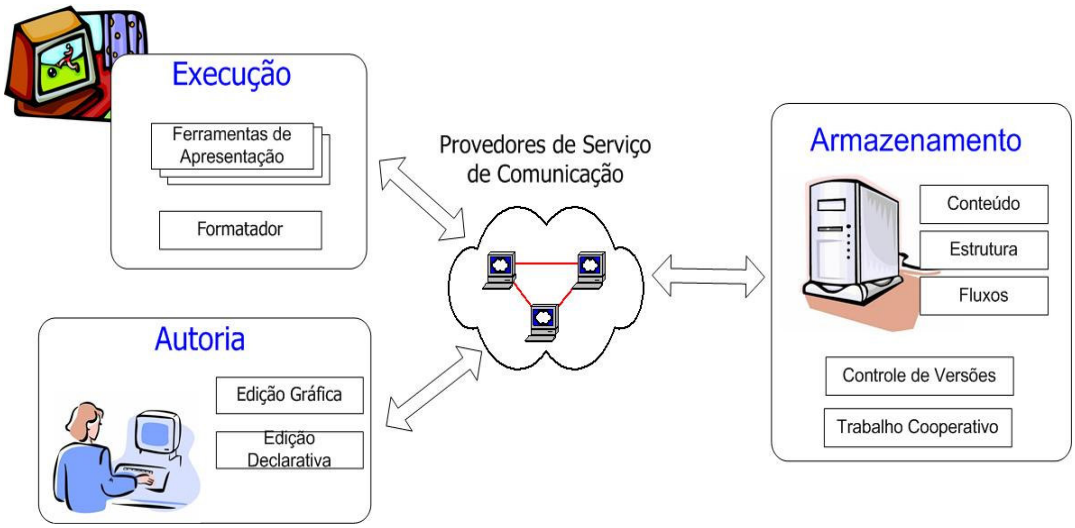

Figura 1.1 – Visão geral de um sistema hipermídia

O ambiente de autoria é o responsável por capturar as expressões do autor e defini-las através dos componentes do documento, das propriedades desses componentes, dos relacionamentos entre esses componentes e dos relacionamentos com componentes pertencentes a outros documentos. Para dar suporte às tarefas desse ambiente, ferramentas para autoria gráfica (edição gráfica) e textual (edição declarativa) podem ser utilizadas (Costa, 1996; Pinto, 2000; Moura, 2001; Coelho, 2004).

No MPEG-4, conforme anteriormente mencionado, os relacionamentos entre os objetos de mídia são especificados em um formato binário. Esse formato favorece o armazenamento e a distribuição linear da especificação das relações entre os objetos em conjunto com o seu conteúdo, no entanto, ele é inapropriado para autoria de aplicações multimídia/hipermídia.

Para facilitar a autoria, o MPEG-4 oferece duas linguagens declarativas: XMT-A e XMT-O (ISO/IEC, 2001). Em XMT-A são definidas todas as expressões existentes no formato binário, sem perda de representatividade. No entanto, ao contemplar todas essas expressões, essa linguagem tornou-se excessivamente complexa e extensa para a autoria. Como conseqüência, a linguagem XMT-O foi desenvolvida, com base na linguagem SMIL 2.0 (W3C, 2005a; Bulterman & Rutledge, 2004).

A linguagem XMT-O utiliza composições, que podem conter objetos de mídia e outras composições, para estruturar um documento. Nessa linguagem, as composições possuem semântica de sincronização, o que torna, a princípio, mais fácil a tarefa de autoria. No entanto, esse benefício é limitado pela existência de

um conjunto simples de composições (paralela, seqüencial e exclusiva) (ISO/IEC, 2001), o que dificulta a definição de relacionamentos complexos, podendo ser necessário, para esses casos, estabelecer composições com vários níveis de aninhamento. Além disso, essas composições obrigam o autor a estruturar o documento de acordo com a especificação para apresentação (Muchaluat-Saade, 2003).

No contexto da autoria, esta dissertação tem como foco facilitar a concepção pelos autores de documentos MPEG-4. Ao adotar a linguagem XMT-O para essa tarefa, esta dissertação propõe a utilização de templates hipermídia voltados para as características dessa linguagem, permitindo a definição de composições com outras semânticas, além das já existentes, favorecendo o reuso e, conseqüentemente, o processo de autoria.

Conceitualmente, templates hipermídia podem especificar diversos tipos de relacionamentos entre os objetos, se constituindo em estruturas hipermídia genéricas, que podem ser herdadas por composições. Como caso particular, na linguagem XMT-O, os templates podem definir relações de inclusão recursivas, oferecendo semânticas temporais a uma única composição, que antes somente eram obtidas através de composições com vários níveis de aninhamento.

Os templates adotados foram definidos inicialmente no módulo *XTemplate* (Muchaluat-Saade, 2003) de NCL 2.0 (*Nested Context Language*) (Silva et al., 2004b). NCL é uma linguagem declarativa e modular XML (W3C, 2000), baseada no modelo NCM (*Nested Context Model*) (Soares et al.,2003), que introduziu uma série de novos conceitos. Posteriormente, adaptações a esse módulo tornaram-no mais flexível, reestruturando suas funcionalidades em diferentes perfis, direcionados às linguagens de autoria onde ele pode ser empregado. Particularmente, o perfil SMIL+*XTemplate* (XT+SMIL) (Silva et al., 2004a), voltado para a linguagem SMIL 2.0, foi adotado neste trabalho como modelo base, uma vez que a linguagem XMT-O baseia-se em SMIL 2.0, possuindo estruturas semelhantes.

Documentos multimídia/hipermídia têm, geralmente, os objetos de mídia (vídeo, áudio, texto, imagem etc.) que os compõem e a especificação dos seus relacionamentos, armazenados em separado. Essa separação é proposta, principalmente, pelos requisitos exigidos pelos diferentes tipos de mídia, particularmente mídias contínuas (áudio e vídeo). Para essas mídias, quando o

armazenamento é realizado de forma distribuída, é desejável que o serviço de entrega dos dados seja capaz de estabelecer e manter acordos de qualidade de serviço (QoS) intramídia (Rodrigues, 2003).

No MPEG-4 o armazenamento dos documentos multimídia/hipermídia pode, eventualmente, ser realizado de forma a separar a especificação dos relacionamentos dos objetos relacionados. No entanto, ao contrário do citado no parágrafo anterior, no MPEG-4 pode-se considerar que o objetivo dessa separação consiste em facilitar as atualizações e modificações nos documentos. Nos documentos, durante a sua concepção, ou mesmo posteriormente, atualizações e modificações são usuais, principalmente quando eles são gerados de forma colaborativa, com múltiplos autores simultâneos ou mesmo subseqüentes (Gorini, 2001).

No entanto, como já mencionado na Seção 1.1.1, a principal forma de armazenamento dos documentos multimídia/hipermídia no MPEG-4, ao contrário da proposta tradicional de separação, segue um modelo integrado, onde as relações entre os objetos de mídia e seus conteúdos permanecem encapsuladas em uma estrutura única, formada por fluxos (*streaming*) que, posteriormente, são diretamente distribuídos aos exibidores.

Cada objeto de mídia pode corresponder a um fluxo e receber o tratamento apropriado à natureza específica do seu conteúdo (codificação individual, requisitos de QoS específicos para o transporte etc). Em alguns casos, objetos podem estar codificados em mais de um fluxo. A relação entre o objeto de mídia e o número de fluxos utilizados para sua codificação é uma opção que oferece aos exibidores padrões de codificação distintos, onde cada fluxo pode corresponder a um padrão diferente ou mesmo a um único padrão, com pequenas variações. Nessa abordagem, também é possível que um mesmo objeto seja codificado de forma escalável e que seus níveis de escalabilidade sejam transmitidos em vários fluxos individuais. Por exemplo, um vídeo pode ser codificado em três níveis de qualidade; a utilização no exibidor dos três fluxos permite apresentar o vídeo em alta qualidade, a utilização de apenas dois fluxos oferece uma apresentação de qualidade intermediária e, a disponibilidade de apenas um fluxo, implica em um vídeo de baixa qualidade.

O modelo de armazenamento integrado padronizado pelo MPEG-4, além de preservar os requisitos exigidos pelos diferentes tipos de mídia, é particularmente

útil em cenários onde a sincronização das mídias, principalmente daquelas provenientes de fontes distintas, é difícil de ser oferecida, como na distribuição de conteúdo em plataformas de TV interativa. Nos sistemas de TV interativa, além do conteúdo de áudio e vídeo principais que, tradicionalmente compõem a apresentação dos usuários (clientes), existe a possibilidade de que aplicações multimídia/hipermídia sejam adicionalmente transmitidas, possibilitando novas representações para as informações, tornando-as mais atrativas e inovadoras, quando comparadas às apresentações tradicionais. A codificação do MPEG-4, aplicada a esses sistemas, permite que todo o conteúdo, incluindo o audiovisual principal e as aplicações, independente da origem ou funcionalidade, seja codificado através de objetos de mídia, integrados à especificação dos seus relacionamentos.

A especificação dos relacionamentos em formato BIFS, anteriormente mencionado, corresponde a um novo fluxo, em sincronia com os demais fluxos dos objetos de mídia que relaciona. É importante mencionar que as especificações em BIFS, relativas à sincronização dos objetos de mídia são, por princípio, temporalmente lineares, isto é, as especificações temporais de sincronismo desse formato são relativas ao tempo de início da apresentação e não aos tempos dos demais objetos, definindo uma estrutura de dados em *timeline*, formada pelos objetos e seus respectivos instantes de exibição. A estrutura em *timeline* não permite determinar as especificações originais concebidas pelos autores, premissa básica necessária para a adoção de um formatador hipermídia (Rodrigues, 2003).

No contexto do armazenamento, o foco deste trabalho está inserido no desenvolvimento de ferramentas para a conversão de documentos multimídia/hipermídia, especificados através da linguagem NCL 2.0, para as linguagens do padrão MPEG-4. Essa conversão permite, entre outros benefícios, que documentos em NCL 2.0 possam ser armazenados e distribuídos de forma integrada, como documentos MPEG-4.

Nessa conversão, documentos especificados na linguagem NCL 2.0 podem ser transformados para a linguagem XMT-O, escolhida por ter uma representatividade mais próxima de NCL 2.0, em relação às outras linguagens e formatos do MPEG-4. Os documentos obtidos em XMT-O, a partir da conversão das especificações em NCL 2.0, podem ser convertidos, tanto para linguagem XMT-A, quanto para o formato binário. É no caso do formato binário, que a

multiplexação da especificação dos relacionamentos com os conteúdos dos objetos de mídia gera um formato para armazenamento (ou transmissão) integrado. Para complementar o intercâmbio entre os formatos, um conversor da linguagem XMT-O para NCL 2.0 também foi desenvolvido, possibilitando a exibição de programas MPEG-4 em formatadores NCL (Rodrigues, 2003).

Cabe ressaltar que a conversão entre os formatos agrega outros benefícios como, por exemplo, permitir que ferramentas de autoria e exibição desenvolvidas para um formato possam ser utilizadas por outro. Como caso particular, o ambiente de autoria NCL poderá ser usado na concepção de documentos MPEG-4 e documentos NCL poderão ser apresentados em exibidores MPEG-4 e vice-versa.

Como resultado da concepção de documentos multimídia/hipermídia, tem-se, primordialmente, apresentações audiovisuais interativas que devem ser exibidas aos usuários no ambiente de execução. É nesse ambiente que as especificações são interpretadas e apresentadas, por meio de dispositivos de saída correspondentes a alguma plataforma particular de exibição. Nesse ambiente também estão disponíveis os mecanismos de interação com os usuários, através de dispositivos de entrada, restringindo, no entanto, as possibilidades de interação àquelas definidas previamente pelos autores na especificação dos documentos.

Em uma visão geral (Figura 1.1), o formatador é o responsável pela exibição de um documento, possuindo um conjunto de ferramentas de exibição para o controle das características dos diversos objetos de mídia e por suas associações com os dispositivos de entrada e saída. Cada ferramenta de exibição é desenvolvida para um conjunto de objetos de mídia e gerencia aspectos relacionados à descompressão, decodificação, controle da taxa de apresentação, interação, entre outros (Rodrigues, 2003).

Compondo o formatador, um orquestrador intermídia é o responsável pelo controle da apresentação, preservando os relacionamentos definidos pelos autores. O orquestrador interage diretamente com as ferramentas de exibição que repassam, para cada objeto, o estado da apresentação, bem como as interações realizadas pelos usuários. Além de controlar as especificações originais dos autores, cabe ao formatador manter a consistência da apresentação através de ajustes no documento hipermídia. Esses ajustes podem ser necessários devido a ações realizadas pelos usuários, gerando eventos não previstos, devido a atrasos

no sistema de comunicação entre o ambiente de execução e o de armazenamento etc. (Rodrigues, 2003).

As ferramentas de exibição MPEG-4 são desenvolvidas para apresentação de cenas audiovisuais, onde cada cena é definida como uma composição que pode conter vários objetos, bem como os relacionamentos entre eles. Dentro de cada cena audiovisual não existem restrições para a definição de relacionamentos entre os objetos definidos internamente, no entanto, os relacionamentos entre cenas distintas são limitados. Nesse caso, o relacionamento, normalmente, possui semântica de referência, isto é, a cena de origem, quando apresentada em um exibidor MPEG-4, possui uma âncora associada a um objeto que, ao ser acionada, interrompe o desenvolvimento dessa cena e carrega a cena de destino, relacionada através de um elo.

No ambiente de execução, o foco deste trabalho está na integração de uma ferramenta de exibição MPEG-4 ao formatador HyperProp, que já possui implementadas ferramentas de exibição para tratar os principais tipos de mídia (texto, imagens, áudio e vídeo). Esta integração permitirá não apenas a exibição de objetos MPEG-4 em formatadores NCL, mas também permitirá a definição de novos relacionamentos entre cenas MPEG-4, superando a limitação atual, ou mesmo entre cenas MPEG-4 e composições, ou objetos, de outros tipos.

1.2. Objetivos

Esta dissertação tem como principal objetivo prover a integração dos recursos oferecidos pelo MPEG-4, contextualizados nos diferentes ambientes que compõem um sistema hipermídia completo, às características da linguagem NCL.

A proposta de integração é composta pelo desenvolvimento dos conversores entre a linguagem NCL 2.0 e XMT-O e vice-versa, além de integrar conversores dos documentos em XMT-O, obtidos através da conversão de NCL 2.0, para XMT-A e BIFS. Nos conversores entre as linguagens declarativas, um *meta-framework* para linguagens XML modulares é utilizado. Para o formato binário, são utilizados os softwares de referência do MPEG-4, e também implementações baseadas nesses softwares.

Complementarmente, no ambiente de execução, este trabalho integra uma ferramenta de exibição MPEG-4 ao sistema HyperProp, para que documentos especificados na linguagem NCL possam estabelecer relacionamentos entre cenas MPEG-4, ou mesmo entre cenas MPEG-4 e outras composições, ou mesmo objetos, de tipos diversos. Além de integrar as ferramentas existentes, este trabalho também pretende estendê-las com o objetivo de tratar as cenas MPEG-4 como composições efetivamente e não apenas como objetos, conforme apresentado no Capítulo 4.

Outro objetivo, desta vez diretamente relacionado à autoria de documentos MPEG-4, é a proposição de templates para composições da linguagem XMT-O, cuja função é facilitar a autoria através do reuso de estruturas de composições complexas. Os templates devem ser derivados daqueles propostos para SMIL 2.0, apresentados no Capítulo 5.

A proposta de integração do MPEG-4, em toda a sua amplitude, entre outras possibilidades, permite se pensar na alternativa de uso do formatador HyperProp como middleware do terminal de acesso para o Sistema Brasileiro de TV Digital, bem como a NCL como a linguagem para desenvolvimento de aplicações interativas, sem a perda de interoperabilidade com outros padrões, ainda que futuros, como no caso, o MPEG-4.

1.3. Organização da Dissertação

Esta dissertação encontra-se organizada como a seguir. O Capítulo 2 apresenta os conceitos do padrão MPEG-4 em detalhes, e os conceitos da linguagem NCL necessários ao entendimento do trabalho. Após descrever a arquitetura do padrão MPEG-4, o capítulo concentra-se nas suas linguagens de autoria, detalhando suas estruturas sintáticas, que serão necessárias nas conversões, e também em seus aspectos semânticos incluindo-se, principalmente, as peculiaridades dos relacionamentos entre os objetos de mídia em cada uma das suas linguagens. Por fim, o capítulo apresenta os conceitos básicos do modelo NCM e da sua representação declarativa (a linguagem NCL), com foco nos templates para autoria de documentos hipermídia.

O Capítulo 3 apresenta os conversores desenvolvidos entre as linguagens NCL 2.0 e XMT-O, e também o conversor dos documentos em XMT-O, obtidos através da conversão de NCL 2.0, para XMT-A, também desenvolvido nesta dissertação, e BIFS.

O Capítulo 4 trata da integração de ferramentas de exibição MPEG-4 ao formatador HyperProp. O capítulo detalha as possibilidades de relacionamentos entre cenas MPEG-4 e entre cenas e outras composições, ou objetos distintos. Nesse capítulo é proposto que cenas audiovisuais MPEG-4 sejam tratadas como composições, apresentando uma ferramenta de exibição estendida para esse fim.

O Capítulo 5 aborda a extensão dos perfis de templates, propostos inicialmente para a linguagem NCL. Ênfase especial é dada ao perfil XT+SMIL, base dos templates apresentados nesta dissertação, também abordados no Capítulo 5 com a denominação de perfil XT+XMT-O.

Finalmente, o Capítulo 6 tece as considerações finais desta dissertação, destacando as suas contribuições e os trabalhos futuros.

2
Conceitos Básicos

No capítulo anterior, foi mencionado que o padrão MPEG-4, entre os padrões para codificação do conteúdo audiovisual, destaca-se, com vantagens, pela sua abordagem orientada a objeto. A arquitetura dos componentes de um sistema MPEG-4, que serve como modelo para a codificação orientada a objeto, é definida na parte inicial desse padrão, denominada *Systems* (ISO/IEC, 2001). Além do modelo arquitetural, o MPEG-4 *Systems* define também os formatos adotados para a especificação das cenas audiovisuais. Pela sua importância, essa parte inicial do MPEG-4, que se dedica à produção de aplicações multimídia/hipermídia, será abordada neste capítulo.

Este capítulo tem como principal objetivo explicar, de forma precisa, os conceitos envolvidos na integração dos recursos oferecidos pelo MPEG-4 às características da linguagem NCL. Baseado nesse objetivo, além do MPEG-4 *Systems*, a linguagem NCL 2.0, originada no modelo NCM, também será abordada, com ênfase nas suas funcionalidades, particularmente nos templates para composições hipermídia.

Para cumprir seus propósitos este capítulo está organizado da forma a seguir. A Seção 2.1 apresenta o padrão MPEG-4, com destaque para sua parte 1 - *Systems*. A Seção 2.2 descreve os formatos para autoria nesse padrão. Por fim, na Seção 2.3, o modelo conceitual NCM – *Nested Context Model* e a linguagem NCL – *Nested Context Language* são resumidamente apresentados.

2.1.
MPEG-4

Os padrões MPEG são definidos pela ISO (*International Organization for Standardization*)[2] e a IEC (*International Electrotechnical Commission*)[3], e têm

[2] http://www.iso.org

[3] http://www.iec.ch

origem no trabalho associado entre várias empresas e grupos de pesquisa de instituições distribuídas por todo o mundo. Historicamente o MPEG-4 teve como objetivo inicial a codificação do conteúdo audiovisual para transmissão a baixas taxas (Pereira & Ebrahimi, 2002). Com esse objetivo, os estudos para o seu desenvolvimento foram iniciados em 1993, no entanto, a partir de 1994 e até 1995, seus objetivos sofreram alterações, voltando-se para uma convergência entre aplicações multimídia, baseadas em TV/filmes/entretenimento, computadores e telecomunicações (Koenen, 2002). Somente em outubro de 1998 foi apresentada a primeira versão desse padrão, que se encontra na segunda versão desde dezembro de 1999.

O MPEG-4, de forma similar aos padrões MPEG anteriores, é distribuído em partes, cada qual com um objetivo específico. Um sumário das partes do padrão é apresentado na Tabela 2.1.

Padrão	**Nomenclatura**
ISO/IEC 14496-1	*Systems*
ISO/IEC 14496-2	*Visual*
ISO/IEC 14496-3	*Audio*
ISO/IEC 14496-4	*Conformance Testing*
ISO/IEC 14496-5	*Reference Software*
ISO/IEC 14496-6	*DMIF (Delivery Multimedia Integration Framework)*
ISO/IEC 14496-7	*Optimized Visual Reference Software*
ISO/IEC 14496-8	*Carriage of MPEG-4 Contents over IP Networks*
ISO/IEC 14496-9	*Reference Hardware Description*
ISO/IEC 14496-10	*Advanced Video Coding*
ISO/IEC 14496-11	*Scene Description and Application Engine*
ISO/IEC 14496-12	*ISO Base Media File Format*
ISO/IEC 14496-13	*IPMP Extensions*
ISO/IEC 14496-14	*MP4 File Format*
ISO/IEC 14496-15	*AVC File Extensions*
ISO/IEC 14496-16	*Animation Framework eXtension (AFX)*
ISO/IEC 14496-17	*Streaming Text Format*
ISO/IEC 14496-18	*Font compression and streaming*
ISO/IEC 14496-19	*Synthesised texture streaming*

Tabela 2.1 – Sumário do padrão MPEG-4

A organização em partes distintas permite que as várias tecnologias, definidas individualmente, possam ser utilizadas em conjunto, mas também de forma independente. Em virtude dessa abordagem, partes do MPEG-4 podem ser utilizadas com tecnologias proprietárias. Considere, por exemplo, um sistema que utilize a codificação de vídeo do MPEG-4 em conjunto com alguma outra codificação de áudio proprietária. Diversas outras combinações são permitidas.

As partes apresentadas na Tabela 2.1 são aquelas atualmente definidas pelo MPEG-4, no entanto, essa lista é dinâmica, com a possibilidade de inserção de

novas partes, correspondentes a aspectos que o comitê da ISO/IEC voltado para o MPEG-4 (*JTC1/SC29/WG11*) julgue convenientes. A possibilidade de adicionar novos avanços tecnológicos ao padrão, através de novas partes, é outra vantagem da fragmentação do MPEG-4.

Como mencionado no início deste capítulo, grande parte da especificação baseada em objetos desse padrão é definida na sua Parte 1: *Systems*. O MPEG-4 *Systems* baseia-se no conceito de que uma cena audiovisual é composta de objetos cujas propriedades são definidas utilizando-se um formato para descrição de cenas. Complementarmente, as Partes 2 e 3 desse padrão dedicam-se, principalmente, aos algoritmos para codificação de vídeo e áudio, respectivamente, sendo que a Parte 10, definida também no ITU-T (*International Telecommunication Union*)[4], através da recomendação H.264 (ITU-T, 2004), complementa a codificação de vídeo apresentada na Parte 2. O conjunto de testes de conformidade, utilizados para validar implementações, é definido na Parte 4. As Partes 5, 7 e 9 definem ferramentas de software e de hardware que servem de referência para implementações, como codificadores e multiplexadores. Os modelos para transporte dos fluxos MPEG-4 são definidos nas Partes 6 e 8.

Da Parte 11 em diante podem ser encontrados os atuais projetos em desenvolvimento desse padrão, alguns dos quais ainda não publicados. Na Parte 11 são definidas as extensões para a descrição de cenas. Essas extensões são compostas por novos elementos textuais e gráficos, novos segmentos para áudio, além de novos atributos para as linguagens declarativas (ISO/IEC, 2004). Na Parte 12, o formato MP4 é definido em detalhes, destacando-se as facilidades para intercâmbio, gerência e apresentação do conteúdo de mídia encapsulado nesse formato (ISO/IEC, 2003). É ainda importante ressaltar que, em todas as suas partes, o MPEG-4 mantém compatibilidade com os padrões anteriores (MPEG-1 e MPEG-2).

2.1.1. MPEG-4 *Systems*

No MPEG-4 uma cena audiovisual pode ser composta por vários objetos (naturais ou sintéticos). Estruturalmente, cada cena é composta por objetos

[4] http://www.itu.int/ITU-T

organizados de forma hierárquica, formando um grafo dirigido acíclico, conforme ilustra a Figura 2.1. Os objetos de mídia são representados pelas extremidades do grafo e os demais nós representam composições criadas a partir desses objetos até o nível de uma cena. A estrutura da cena audiovisual é dinâmica, pois nós podem ser adicionados, substituídos ou removidos. As propriedades dos nós também podem ser alteradas, como, por exemplo, o posicionamento dos objetos em cena.

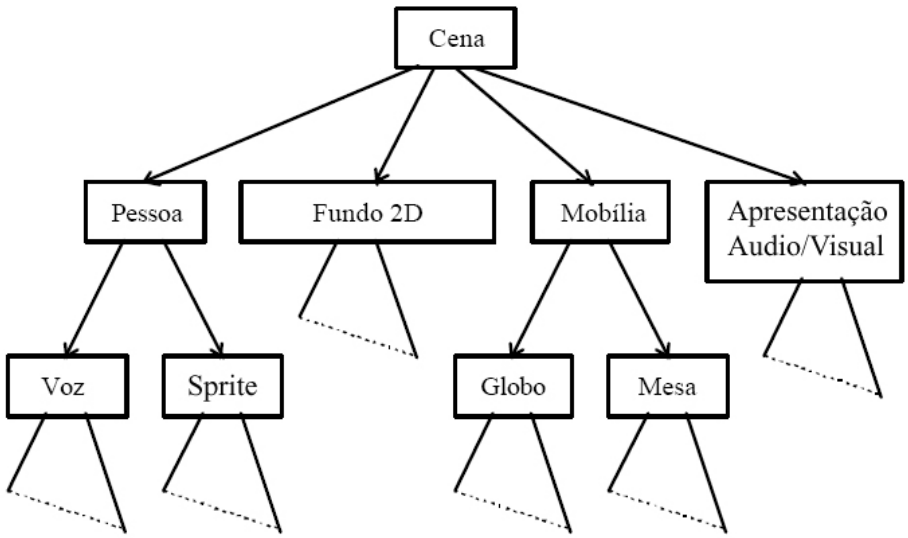

Figura 2.1 – Estrutura de uma cena MPEG-4

Na Figura 2.2, a estrutura representada pela Figura 2.1 é exemplificada através de uma cena cotidiana; nela tem-se a representação de uma aula, onde o conteúdo audiovisual é formado por objetos. Ainda na Figura 2.2, a exibição da cena, denominada visão hipotética, corresponde ao conjunto dos objetos distribuídos na cena e apresentados através de um exibidor MPEG-4. Esses objetos, correspondentes à realidade do conteúdo capturado, podem ser exibidos através de várias ferramentas audiovisuais (ISO/IEC, 2000b). Complementarmente, o MPEG-4 permite que sejam definidos eventos de interação para os objetos pertencentes à cena. Esses eventos, bem como os relacionamentos entre os objetos que compõem a cena audiovisual, são especificados através de formatos para descrição das cenas.

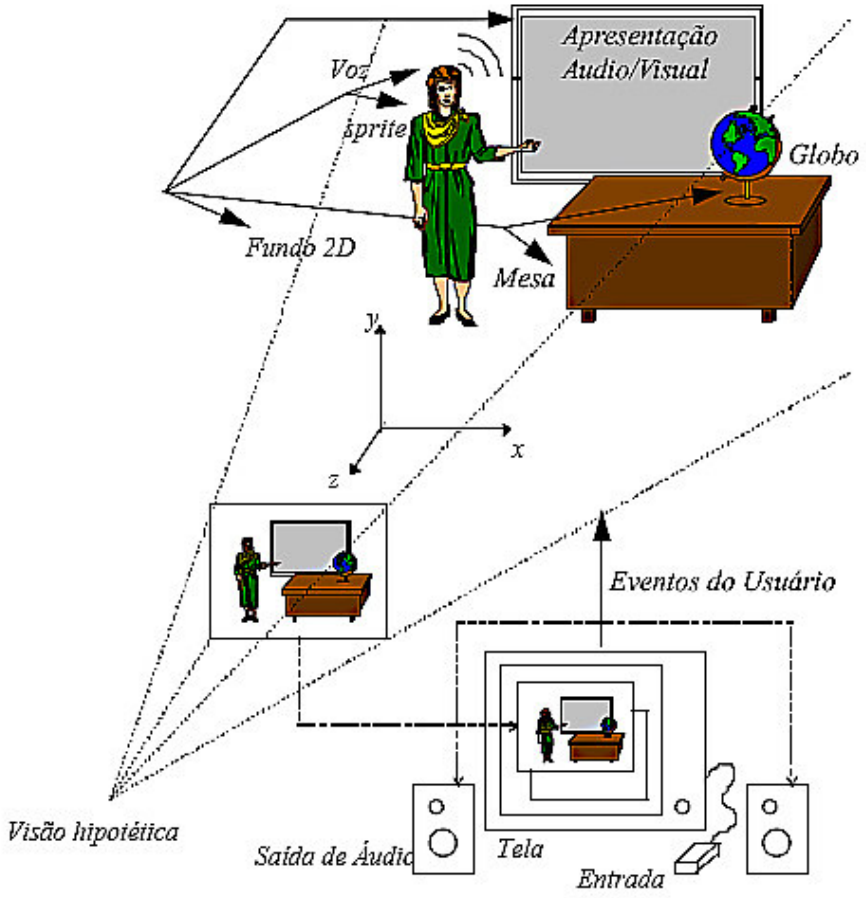

Figura 2.2 – Exemplo de uma cena MPEG-4

Conforme mencionado no Capítulo 1, no MPEG-4 cada objeto de mídia de um documento pode ser codificado como um ou mais fluxos individuais, denominados fluxos elementares. Além desses fluxos, dois outros complementam esse modelo: o fluxo descritor de cenas, que contém as informações das relações entre os objetos de mídia e é especificado através do formato BIFS e o fluxo descritor de objetos, que especifica, entre outras informações, quais são os fluxos elementares que representam cada objeto de mídia no documento.

A Figura 2.3 ilustra cada um dos fluxos citados no parágrafo anterior. Nela, um fluxo descritor de cenas, especificado em BIFS, contém as informações relativas a estrutura da cena a ser exibida. A Figura 2.3 contém ainda três fluxos elementares relativos a dois objetos da cena, um áudio e um vídeo. Um fluxo elementar corresponde ao objeto de áudio, enquanto os dois fluxos restantes correspondem a uma opção de escalabilidade para o vídeo. A relação entre os fluxos elementares existentes e os objetos em cena é definida pelos descritores de objetos pertencentes ao fluxo descritor de objetos.

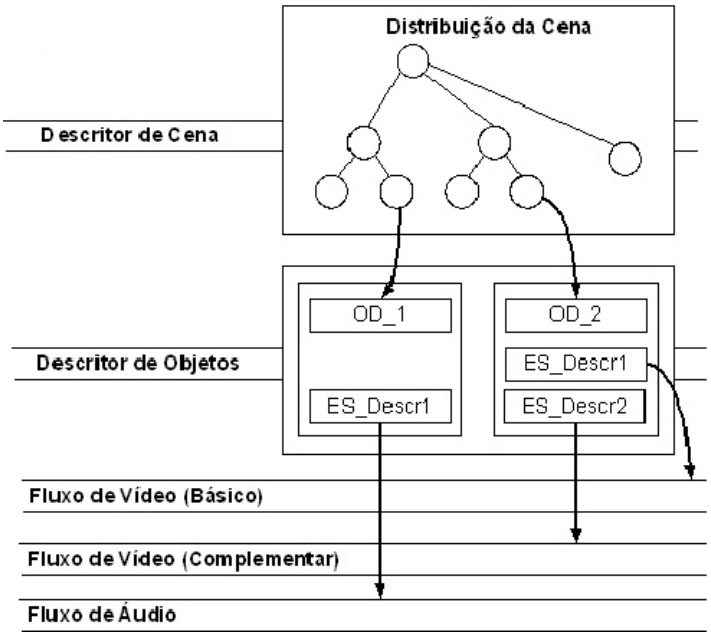

Figura 2.3 – Fluxos MPEG-4

No modelo de fluxos proposto pelo MPEG-4, além dos fluxos básicos citados, também existem outros fluxos específicos, como o fluxo OCI (*Object Content Information*), que contém informações sobre o conteúdo dos fluxos elementares e o fluxo IPMP (*Intellectual Property Management and Protection*), que armazena informações relativas à proteção de direitos autorais (Koenen, 2002).

O fluxo OCI define *metadados* sobre o conteúdo das cenas através da sua descrição semântica. Esse fluxo atribui palavras-chave, resumos ou textos completos sobre o conteúdo dos fluxos elementares. Atualmente, pode-se considerar a possibilidade de utilização do padrão MPEG-7 (Martinez, 2003) para realizar as funções do fluxo OCI (Pereira & Ebrahimi, 2002). O fluxo IPMP corresponde a uma estrutura genérica para implementar proteção ao conteúdo, permitindo atribuir um identificador de propriedade intelectual (IPI) aos fluxos elementares, com o objetivo de gerenciar o direito da apresentação (Koenen, 2002).

A seguir, a Figura 2.4 ilustra as etapas definidas na arquitetura dos componentes de um sistema MPEG-4 (Koenen, 2002). Nessa arquitetura, todas as informações, relativas ao conteúdo audiovisual, são transmitidas através de fluxos contínuos, onde técnicas de compactação e compressão podem ser aplicadas individualmente. Antes da transmissão, todo o conjunto deve ser sincronizado,

obedecendo às relações definidas entre os objetos de mídia, ainda na fase de autoria[5].

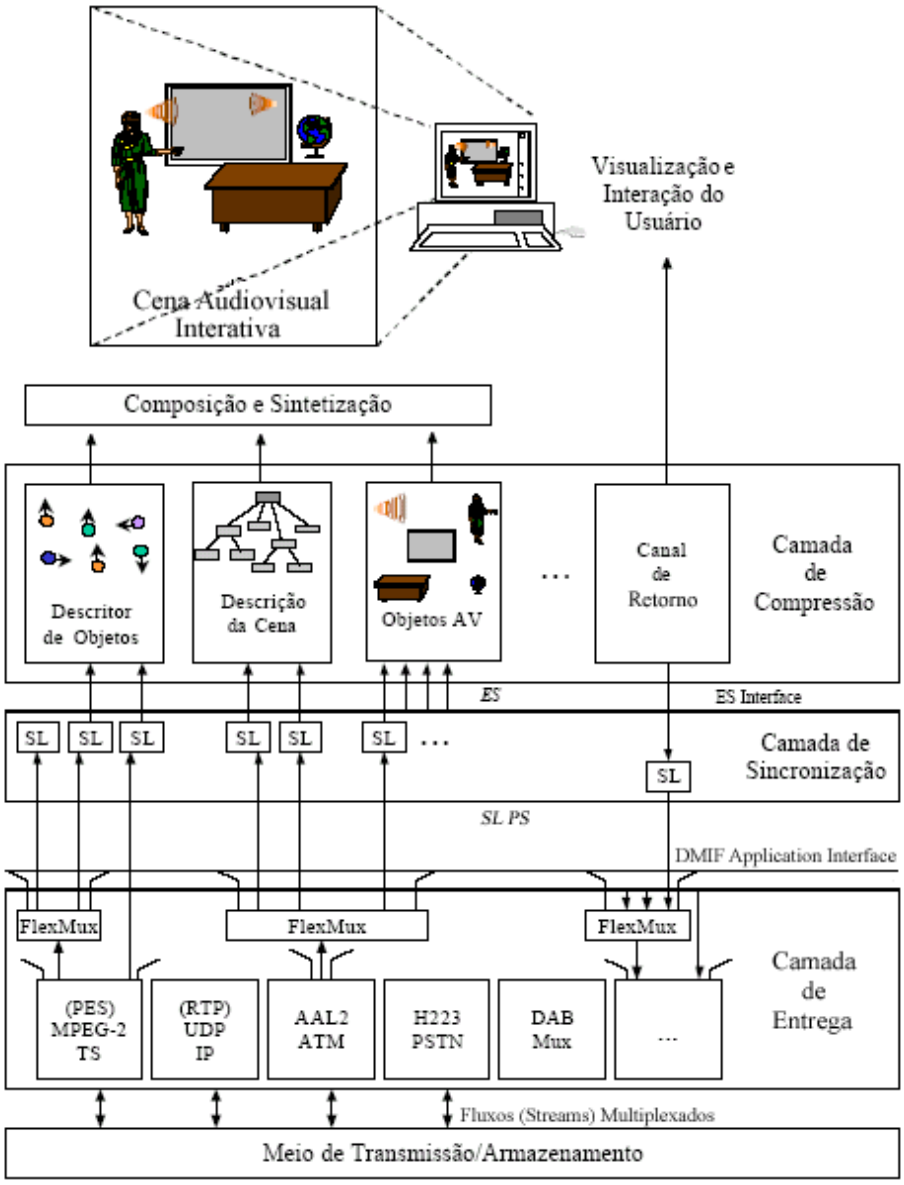

Figura 2.4 – Arquitetura de componentes MPEG-4

Na *camada de compressão*, localizada na parte superior da arquitetura, os objetos de mídia que compõem uma cena, o fluxo descritor de cenas e os descritores de objetos são codificados e decodificados individualmente. Nessa camada o fluxo descritor de cenas, contendo os relacionamentos entre os conteúdos e as características da apresentação, é codificado no formato BIFS. Os descritores de objetos, por sua vez, são codificados com as informações que

[5] Note que o sentido das setas indicam a recepção dos fluxos, mas basta invertê-las para termos a arquitetura para transmissão dos mesmos.

relacionam os objetos de mídia às instâncias de objetos nas cenas, e também com as descrições semânticas e regras para controle de acesso dos fluxos (fluxos OCI e IPMP). Finalmente, os objetos de mídia são codificados individualmente, de acordo com as características de cada mídia.

O sincronismo é definido temporalmente, e de forma relativa entre os fluxos elementares, na *camada de sincronização*. Nessa camada, os segmentos codificados são sincronizados através de marcas de tempo (*time stamp*), localizadas dentro das unidades de acesso individuais (ISO/IEC, 2001), em cada fluxo elementar. As unidades de acesso são elementos discretos de informação, existentes em todos os fluxos (descritor de cenas, descritor de objetos, OCI, IPMP, elementares etc.), que variam de acordo com as características de cada fluxo. Por exemplo, em um fluxo elementar representando um vídeo, as unidades de acesso podem ser formadas por cada um dos quadros desse vídeo, por outro lado, em um fluxo descritor de cenas, uma unidade de acesso pode ser definida por uma composição, que pode conter referências a vários objetos em cenas.

Ainda na *camada de sincronização*, os fluxos são encapsulados em pacotes de sincronização (*SyncLayer packet*). Nesses pacotes, as unidades de acesso podem ser fragmentadas para transmissão. Cada pacote de sincronização contém informações importantes no seu cabeçalho, como o seu número de seqüência, para o controle dos pacotes, o relógio do codificador (OCR - *Object Clock Reference*), que permite sincronizar o receptor com o transmissor, além de informações básicas sobre o fluxo encapsulado, como a ordem de fragmentação das unidades de acesso.

A *camada de entrega*, também denominada *camada de transporte*, é a última camada antes da passagem dos fluxos MPEG-4 para os meios de transmissão. No primeiro nível dessa camada existe um modelo de multiplexação denominado *FlexMux* (ISO/IEC, 2001). A adoção dessa multiplexação é opcional, no entanto, fluxos com características semelhantes e requisitos de qualidade de serviço (QoS) similares podem utilizá-la a fim de reduzir o número de conexões estabelecidas. Diferenças nas variações do atraso fim-a-fim entre fluxos correlacionados também podem ser minimizadas através da utilização desse módulo (Pereira & Ebrahimi, 2002). Na omissão dessa multiplexação, cada fluxo elementar, correspondente aos objetos de mídia codificados, é mapeado diretamente para um canal de transporte (Herpel, 1999).

O segundo nível da *camada de entrega* oferece vários serviços para o transporte das informações. Nessa camada somente a interface dos vários serviços é especificada, característica que permite ao MPEG-4 utilizar vários tipos de protocolos, adequados a aplicações e meios de transmissão distintos, tais como: RTP, UDP, TCP e ATM. Entre os protocolos disponíveis destaca-se a possibilidade de integração do MPEG-4 com o fluxo de transporte MPEG-2 (ISO/IEC, 2000a), adotado em praticamente todos os padrões atuais para TV digital interativa.

Todos os serviços oferecidos pela camada de transporte são definidos e coordenados pelo *framework* definido na Parte 6 do MPEG-4, denominado DMIF (*Delivery Multimedia Integration Framework*). Através do DMIF são definidos os canais de transporte, independente do protocolo adotado, utilizando a interface DAI (*DMIF Application Interface*). Essa interface abstrai, para as aplicações, o mapeamento dos fluxos MPEG-4 nos canais de transporte (ISO/IEC, 2000c).

Para exemplificar as funções do DMIF, pode ser interessante compará-lo a um protocolo com funções próximas e bastante conhecido: o FTP (*File Transfer Protocol*) (Postel & Reynolds, 1985). A diferença essencial está no fato de que o FTP retorna um conjunto de dados, enquanto que o DMIF retorna ponteiros indicando o caminho para a recuperação de um determinado fluxo de informações. Quando o DMIF é executado, do ponto de vista do cliente, a primeira ação realizada é o estabelecimento de uma sessão com o servidor remoto. Após o estabelecimento dessa sessão, os fluxos são selecionados e ocorre uma requisição para que eles comecem a ser enviados (*streaming*). Na camada de entrega, onde a conexão foi solicitada, ponteiros são retornados para as conexões de transporte onde os fluxos serão recuperados, estabelecendo assim a comunicação entre o cliente e o servidor.

Para obter acesso ao conteúdo MPEG-4 através do DMIF, segundo Herpel (Herpel, 1999), a comunicação entre o cliente e o servidor pode ser instanciada por requisições, a partir do cliente, como em um cenário típico de aplicações WWW, onde as requisições são realizadas através de URLs (*Uniform Resource Locator*) relativas ao servidor. Complementarmente, essa comunicação pode ser realizada por difusão (*broadcasting*) como, por exemplo, através de descritores MPEG-4 específicos inseridos nas tabelas de programa do MPEG-2 (*program-*

specific information-PSI) (ISO/IEC, 2000a). Em ambos os casos, para receber o conteúdo, as seguintes etapas são estabelecidas:

- O cliente recebe um único descritor de objetos denominado descritor de objetos inicial (IOD);
- O descritor de objetos inicial contém indicadores para o(s) fluxo(s) descritor de objetos e para o(s) fluxo(s) descritor de cenas;
- O cliente realiza requisições desses fluxos através dos seus indicadores usando a interface DAI;
- O cliente recebe uma mensagem indicando que os fluxos solicitados serão transmitidos;
- Em cenários interativos, o transmissor pode exigir que o cliente confirme que está pronto para receber esses fluxos;
- Os fluxos são transmitidos;
- A partir do fluxo descritor de cenas, o cliente traduz as expressões em BIFS e, com o auxílio do fluxo descritor de objetos, seleciona os identificadores dos fluxos elementares necessários à apresentação;
- O cliente realiza requisições desses fluxos elementares, através dos seus indicadores, usando a interface DAI (neste ponto o processo se repete para os fluxos elementares).

É importante ressaltar que o descritor de objetos inicial é uma entidade específica do início de uma transmissão que contém outras informações, além dos fluxos descritores de cenas e descritores de objetos. Esse descritor também armazena as informações relativas aos perfis e níveis (ISO/IEC, 2001) do conteúdo a ser transmitido. Os perfis e níveis indicam os recursos necessários ao receptor para que ele seja capaz de manipular o conteúdo de uma apresentação. A presença dessas informações no descritor de objetos inicial permite aos receptores tomar decisões sobre a possibilidade ou não de manipular o conteúdo a ser transmitido antes do mesmo ser recebido. Eventualmente, em apresentações com um grande número de cenas, onde exista uma variação na complexidade entre elas, é possível indicar os requisitos exigidos individualmente, por cada cena (Herpel & Eleftheriadis, 2000).

2.2. Linguagens para Documentos MPEG-4

Na arquitetura do MPEG-4, o fluxo descritor de cenas é codificado no formato BIFS. Esse formato possui construções baseadas nos conceitos da VRML (*Virtual Reality Modeling Language*) (WEB3D, 1996); linguagem que teve origem na década de 90 com o objetivo de descrever ambientes interativos tridimensionais aplicados à WWW, através de uma sintaxe textual e independente de plataforma. A primeira versão dessa linguagem (VRML 1.0) surgiu em 1995, porém, como essa versão possuía limitações de interatividade entre os usuários e as aplicações virtuais, em 1996 iniciou-se o desenvolvimento da segunda versão (VRML 2.0). Essa segunda versão acrescentou uma nova sintaxe, introduzindo várias modificações para permitir animações, som, interação etc., motivando, em 1997, a adoção dessa linguagem pela ISO e IEC. Após uma nova reestruturação, essa linguagem foi definida como um padrão ISO/IEC, com o nome de VRML97 (ISO/IEC, 1997).

Apesar de inspirado na VRML, BIFS possui características próprias, adequadas para o transporte MPEG-4. Ao contrário da VRML, onde seus objetos e ações são declarados textualmente, em BIFS todas as estruturas são especificadas em código binário. Essa diferença, entre outros aspectos, faz com que os documentos especificados em BIFS sejam extremamente menores do que em outras linguagens (Herpel, 1999).

Conforme citado na Seção 2.1, a adoção do formato binário é fundamental para a distribuição de cenas MPEG-4, no entanto esse formato pode dificultar o processo de autoria. Apesar da possibilidade de utilização da edição gráfica, onde o formato binário torna-se transparente para os autores, a edição declarativa pode ser a opção escolhida por muitos, por necessitar apenas de um editor de texto tradicional para escrever a representação textual do documento (Muchaluat-Saade, 2003).

Para contornar essa limitação do formato binário, o formato XMT (*eXtensible MPEG-4 Textual*) (ISO/IEC, 2001) foi proposto como um *framework* para especificar descrições de cenas MPEG-4 através de uma sintaxe textual. Esse *framework* é instanciado por duas linguagens, com diferentes sintaxes e

semânticas: a linguagem XMT-A e a linguagem XMT-O, apresentadas nas Seções 2.2.1 e 2.2.2, respectivamente.

2.2.1. Linguagem XMT-A

A linguagem XMT-A é uma versão declarativa de BIFS para representar descrições de cenas MPEG-4. Sua estrutura baseia-se em construções XML com representação direta das expressões existentes no formato BIFS. Por representar construções em BIFS, XMT-A também possui estruturas semelhantes às definidas em VRML. A Figura 2.5 ilustra parte de um documento MPEG-4 especificado em XMT-A.

```
<?xml version="1.0" encoding="US-ASCII" standalone="yes"?>
<XMT-A xmlns="urn:mpeg:mpeg4:xmta:schema:2002" ...>
 <Header>
  <InitialObjectDescriptor objectDescriptorID="IODID_1">
    <Profiles ODProfileLevelIndication="Unspecified" … />
      <Descr>
        <esDescr>
          <ES_Descriptor ES_ID="IOD_BIFS">
            <decConfigDescr>
              <DecoderConfigDescriptor  streamType="SceneDescription" …>
                ….
              </DecorderConfigDescriptor>
            </decConfigDescr>
              …
          </ES_Descriptor>
          <ES_Descriptor ES_ID="IOD_OD">
            <decConfigDescr>
              <DecoderConfigDescriptor streamType="ObjectDescriptor" ... />
            </decConfigDescr>
              …
          </ES_Descriptor>
        </esDescr>
      </Descr>
  </InitialObjectDescriptor>
 </Header>
 <Body>
    <ObjectDescriptorUpdate>
      <OD>
        <ObjectDescriptor binaryID="1" objectDescriptorID="od1">
          <Descr>
            <esDescr>
              <ES_Descriptor ES_ID="ESID4">
                <decConfigDescr>
                  <DecoderConfigDescriptor objectTypeIndication="5" streamType="4"../>
                  ….
                </decConfigDescr>
                <slConfigDescr>
                  …
                  <StreamSource url="../mediaContent/musicas/img/logo.jpg"/>
              </ES_Descriptor>
```

```
                </esDescr>
              </Descr>
            </ObjectDescriptor>
          </OD>
        </ObjectDescriptorUpdate>
        …
        <par begin="0.0">
          <Insert atNode="audioRegion1" position="END">
            <OrderedGroup DEF="5">
              …
                <AudioSource url="od:1"/>
              …
            </OrderedGroup>
          </Insert>
        </par>
    </Body>
</XMT-A>
```

Figura 2.5 – Exemplo de documento especificado em XMT-A

A primeira parte de um documento XMT-A é composta pelo cabeçalho (*Header*), que especifica o descritor de objetos inicial. Esse descritor de objetos é, segundo o modelo MPEG-4 *Systems*, o elemento inicial de uma apresentação. Nele são especificadas, através do elemento *Profiles*, as informações sobre os recursos necessários no receptor para interpretar o documento. Além de estabelecer os níveis e perfis do documento, esse descritor indica quais são os fluxos iniciais de uma apresentação. Na Figura 2.5, o descritor de objetos inicial especifica um fluxo descritor de cenas e um fluxo descritor de objetos, através dos elementos *DecoderConfigDescriptor*.

No corpo de um documento XMT-A (*Body*) são definidas as construções para apresentação dos objetos em cenas. O documento apresentado na Figura 2.5 tem como objetivo apresentar uma imagem (*logo.jpg*) através de um exibidor MPEG-4. Para isso, o comando BIFS, representado pelo elemento *ObjectDescriptorUpdate*, é utilizado para atualizar o fluxo descritor de objetos, instanciando nesse fluxo uma referência para essa imagem. A partir do descritor de objetos atualizado, um outro comando BIFS, responsável pela inserção (*insert*) é acionado. Esse comando atua no instante inicial da apresentação, determinado pelo elemento *par*, inserindo no nó *audioRegion1*, definido pelo atributo *atNode* do comando *Insert*, a imagem especificada pelo descritor de objetos.

A descrição detalhada da sintaxe da linguagem XMT-A pode ser encontrada na referência (ISO/IEC, 2001). Em relação ao formato BIFS, essa linguagem favorece o procedimento declarativo na autoria, pois permite ao autor escrever textualmente a especificação do documento multimídia/hipermídia.

Por outro lado, no aspecto semântico, XMT-A não acrescenta nenhuma funcionalidade. Na realidade, por representar diretamente todos os elementos presentes na arquitetura MPEG-4, XMT-A tornou-se excessivamente complexa e extensa para a autoria (Kim et al., 2000).

2.2.2. Linguagem XMT-O

Para diminuir a complexidade da especificação de cenas MPEG-4, a linguagem XMT-O foi desenvolvida. Nela os objetos audiovisuais e seus relacionamentos são descritos em um alto nível de abstração, facilitando o processo de autoria, pois os documentos multimídia/hipermídia são concebidos com base nas intenções do autor, ao contrário do que ocorre em XMT-A. Comparar a capacidade de autoria de XMT-O à de XMT-A é algo como comparar a facilidade de programação de C++ com *Assembly* (Kim & Wood, 2002).

Com o objetivo de melhor compreender a estrutura de XMT-O, a Figura 2.6 apresenta um documento MPEG-4 especificado nessa linguagem.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<XMT-O xmlns="urn:mpeg:mpeg4:xmto:schema:2002" ...>
 <head>
    <layout  type="xmt/xmt-basic-layout" metrics="pixel" >
         <topLayout  id="window1" ...>
            <region id="title" size="600 60" translation="0 145" .../>
          ...
         </topLayout>
    </layout>
    <defs>
      <img id="defImg" src="picture.jpg" … />
    </defs>
   <meta name="copyright" content="(c)Copyright Telemidia" … />
 </head>
 <body>
   <par id="ncl_example-processed">
      <rectangle region="title" dur="indefinite" size="600 60">
         <material color="white" filled="true"/>
      </rectangle>
      <par id="coisaPele">
          <par id="ncl-composite-1">
              <audio id="samba" src="coisa.mp3" region="audioRegion1"   .../>
                 <rectangle id="part1" begin="samba.begin+8.4s" end="samba.begin+18s" .../>
                  ...
                 </rectangle>
              </audio>
             <lines region="audioRegion1" … />
            …
          </par>
          ...
       </par>
```

```
        <string id="title-coisaPele" textLines="Coisa de Pele" …>
            <material color="black"/>
            <fontStyle family="TYPEWRITER" … size="22" …/>
        </string>
        …
        <rectangle id="logotele1" … begin="samba.begin" end="samba.end" …>
            <texture src="logo.jpg"/>
        </rectangle>
        …
        <use xlink:href="#defImg" dur="indefinite" />
  </body>
</XMT-O>
```

Figura 2.6 – Exemplo de documento especificado em XMT-O

A linguagem XMT-O divide a estrutura de um documento em duas partes, o cabeçalho (*head*) e o corpo (*body*), da mesma forma que outras linguagens padronizadas pelo W3C[6], como SMIL 2.0.

No cabeçalho encontram-se as informações gerais do documento, como o leiaute da apresentação, metadados para definição semântica do documento e as referências para conjuntos de elementos. O leiaute (*layout*) é definido por uma única janela (*topLayout*), que pode ser composta por várias regiões (*region*). A distribuição espacial das regiões é definida através de coordenadas cartesianas, com origem no centro da janela. As regiões definem os locais onde os elementos (objetos de mídia) serão apresentados nos dispositivos de exibição. Os metadados para definição semântica (*meta*) acrescentam informações sobre o documento especificado. Na Figura 2.6, no cabeçalho do documento é definido ainda um elemento *defs*, que contém a definição de um elemento *img*, correspondente ao objeto de mídia imagem. O elemento *defs* define conjuntos de elementos que podem ser referenciados no corpo do documento, através do elemento *use.*

O corpo do documento especificado em XMT-O é uma composição, que pode conter objetos de mídia e outras composições, recursivamente. Em XMT-O, de forma análoga a SMIL 2.0, as composições possuem semântica de sincronização, que serão abordadas ainda neste capítulo. O corpo de um documento XMT-O possui semântica temporal seqüencial. Nesse corpo, outras composições podem ser definidas, através dos elementos *par*, *seq* e *excl.* Os nomes dessas composições indicam a semântica temporal implementada (paralela, seqüencial e exclusiva). Cada composição pode conter, além de outras composições, objetos de mídia, desde os tradicionais como imagens (*img*), vídeos

[6] http://www.w3.org

(*video*), áudio (*audio*), até os objetos sintéticos, como retângulos (*rectangle*), linhas (*lines*), círculos (*circle*) etc.

Em XMT-O as características para apresentação dos objetos de mídia são especificadas nos próprios elementos, através dos seus atributos. Como exemplo, nos elementos que representam os objetos de mídia, a região para exibição é definida pelo atributo *region* desses elementos. Além das características para apresentação, os relacionamentos entre esses objetos também podem estar embutidos nos seus atributos. Dessa forma, além de composições com semântica de sincronização, relacionamentos podem ser estabelecidos através de eventos, especificados nos atributos dos elementos. Como exemplo, considere o elemento *rectangle*, contendo o atributo *id* igual a *logotele1*, especificado no final do documento apresentado na Figura 2.6. Seus atributos *begin* e *end* definem elos de sincronização temporal com o objeto áudio, representado pelo elemento *audio*, contendo o atributo *id* igual a *samba*. Nesse exemplo, o início e o fim da apresentação do objeto sintético retângulo são definidos pelo início e fim da apresentação do objeto de áudio.

2.2.2.1. Módulos da Linguagem XMT-O

Como apresentado no Capítulo 1, a linguagem XMT-O baseia-se na linguagem SMIL 2.0. XMT-O, além de declarativa, possui uma estrutura modular, onde vários de seus módulos possuem definições similares ou idênticas aos módulos definidos em SMIL 2.0. A Tabela 2.2 apresenta um agrupamento dos módulos da linguagem XMT-O em 12 áreas funcionais. É importante ressaltar que a estrutura de áreas funcionais apresentada nessa tabela não corresponde exatamente à estrutura organizacional definida pelo padrão MPEG-4 (XMT-O) (ISO/IEC, 2001). Na realidade, a estrutura da Tabela 2.2 tem como objetivo principal destacar as semelhanças existentes entre XMT-O e SMIL 2.0. Os módulos de XMT-O herdados diretamente de SMIL aparecem na Tabela 2.2 com a expressão “SMIL” em destaque.

Áreas Funcionais	Módulos
Timing	*AccessKeyTiming Module (SMIL)* *BasicInlineTiming Module (SMIL)* *BasicTimeContainers Module (SMIL)*

	EventTiming Module (SMIL) *ExclTimeContainers Module (SMIL)* *FillDefault Module (SMIL)* *MediaMarkerTiming Module (SMIL)* *MinMaxTiming Module (SMIL)* *MultiArcTiming Module (SMIL)* *RepeatTiming Module (SMIL)* *RepeatValueTiming Module (SMIL)* *RestartDefault Module (SMIL)* *RestartTiming Module (SMIL)* *SyncbaseTiming Module (SMIL)* *SyncBehavior Module (SMIL)* *SyncBehaviorDefault Module (SMIL)* *SyncMaster Module (SMIL)* *XMT Events Module*
Time Manipulations	*TimeManipulations Module (SMIL)* *FlexTime Module*
Animation	*BasicAnimation Module (SMIL)* *SplitAnimation Module (SMIL)* *DragAnimation Module*
Content Control	*BasicContentControl Module (SMIL)* *CustomTestAttributes Module (SMIL)* *PrefetchControl Module (SMIL)* *SkipContentControl Module (SMIL)*
Layout	*Layout Module*
Linking	*Linking Module*
Media Objects	*MediaClipping Module (SMIL)* *MediaClipMarkers Module (SMIL)* *MediaDescription Module (SMIL)* *MediaGroup Module* *xMedia Module* *MediaAugmentation Module* *CustomXMT-AMedia Module*
Metainformation	*Metainformation Module (SMIL)*
Structure	*Structure Module*
Transitions	*BasicTransitions Module (SMIL)* *InlineTransitions Module (SMIL)* *TransitionsModifiers Module (SMIL)*
Macros	*Macros Module*
DEFS	*DEFS Module*

Tabela 2.2 – Módulos que compõem a linguagem XMT-O

As próximas seções descrevem, de maneira sucinta, as principais características dos módulos apresentados na Tabela 2.2. Complementarmente, no Apêndice A podem ser encontrados os elementos e atributos de cada área funcional de XMT-O. A definição completa dos módulos de XMT-O pode ser encontrada na referência (ISO/IEC, 2001).

2.2.2.1.1. Área Funcional *Timing*

Os relacionamentos temporais na linguagem XMT-O podem ser declarados através de composições com semântica temporal. Essas composições são

especificadas pelos módulos *BasicTimeContainers*, que especifica composições com semântica paralela (os objetos pertencentes a essa composição são apresentados simultaneamente) e seqüencial (os objetos pertencentes a essa composição são apresentados um após o término do outro), e *ExclTimeContainers*, que especifica as composições com semântica exclusiva (somente um objeto é apresentado em cada instante de tempo, mas a ordem não é definida pela composição). Ambos os módulos são definidos originalmente em SMIL 2.0.

Relacionamentos temporais também podem ser especificados através de eventos definidos nos atributos dos objetos. Esses atributos são especificados nos módulos *BasicInlineTiming e EventTiming*. O *BasicInlineTiming* define valores numéricos, com algumas exceções (ISO/IEC, 2001), para a especificação da duração ideal (*dur*), início (*begin*) e término (*end*) da apresentação de um objeto qualquer. Além dos valores numéricos, o módulo *EventTiming* permite que eventos sejam definidos nos atributos relativos ao início (*begin*) e ao fim (*end*) da apresentação de um objeto. Os eventos que podem estar associados a esses atributos são especificados pelos módulos *AccessKeyTiming* e *SyncbaseTiming* de SMIL 2.0 e *XMTEvents* de XMT-O. Esses eventos exploram ações como manipulações do *mouse* (*mouse up, down, out* e *over*), início (*begin*) e fim (*end*) da apresentação de objetos, colisão e aproximação em animações, visibilidade de objetos, entre outros.

Complementarmente, existem alguns módulos de XMT-O que, apesar de não definirem composições e eventos, possuem elementos que são associados a atributos temporais. Esses módulos são o *FillDefault*, *MediaMarkerTiming*, *MinMaxTiming, MultiArcTiming*, *RepeatTiming*, *RepeatValueTiming*, *RestartDefault*, *RestartTiming, SyncBehavior*, *SyncBehaviorDefault* e *SyncMaster*, todos herdados de SMIL 2.0.

O módulo *FillDefault* estabelece a ação a ser tomada na apresentação de um objeto, quando a sua exibição termina. Uma das ações possíveis, definidas por esse módulo, consiste em manter a exibição do objeto mesmo após o término do tempo estabelecido para a sua apresentação.

O módulo *MediaMarkerTiming* permite que marcas de sincronização, existentes no conteúdo dos fluxos relativos aos objetos de mídia, sejam utilizadas como âncoras dos eventos estabelecidos. Como exemplo, em um objeto de mídia

vídeo, marcas na codificação dos seus quadros podem corresponder às âncoras especificadas nesse módulo.

O módulo *MinMaxTiming* define atributos que permitem ao autor especificar os limites superior e inferior do tempo para exibição de um determinado objeto.

O módulo *MultiArcTiming* permite que os atributos para início e fim de uma apresentação possam ter múltiplas condições, separadas por símbolos de ponto-e-vírgula, cuja semântica corresponde a uma expressão booleana, onde o ponto-e-vírgula representa o operador "ou" (*or*).

O módulo *RepeatTiming* permite aos autores especificar o tempo total para apresentação de um objeto qualquer, onde, durante esse tempo, um número estabelecido de repetições serão acionadas.

O módulo *RepeatValueTiming* permite aos autores especificar um número de ocorrências, relativas a um evento qualquer, como uma condição para um evento de apresentação. Como exemplo, considere a exibição de um objeto de vídeo, cujo término está condicionado a um número de cliques do mouse sobre um objeto qualquer da apresentação. Especificar a quantidade de eventos, que no caso, corrrespondem a cliques do mouse, é possível através das construções do *RepeatValueTiming*.

Os módulos *RestartTiming* e *RestartDefault* estabelecem condições para o reinício do evento de apresentação. Em XMT-O, quando um objeto está sendo apresentado, as especificações do módulo *RestartTiming* ou *RestartDefault* definem se essa apresentação pode ou não ser reiniciada. Por exemplo, a apresentação de um objeto de vídeo pode estar relacionada ao evento do clique do mouse sobre um objeto qualquer. Uma vez acionado o evento, tem início a apresentação do vídeo. Nesse ponto, caso o usuário clique novamente com o mouse sobre o objeto, o vídeo pode ser reiniciado ou não, dependendo da especificação no documento. O módulo *RestartDefault* corresponde à ação padrão a ser adotada quando as especificações do módulo *RestartTiming* não forem informadas.

Os atributos do módulo *SyncBehavior* podem ser utilizados nas composições XMT-O para definir o seu comportamento caso ocorram atrasos para a exibição dos seus componentes. Como exemplo, se uma composição paralela tiver o valor do atributo *syncBehavior* igual a *canSlip*, e se ocorrer um atraso com

algum dos seus componentes, toda a composição aguarda até a sincronização desse componente. O módulo *SyncBehaviorDefault* corresponde à ação de sincronismo padrão a ser adotada quando as especificações do módulo *SyncBehavior* não forem informadas.

O módulo *SyncMaster* define o fluxo de mídia principal, no qual todos os outros fluxos devem estar sincronizados, em relação ao relógio (*clock*) da apresentação (ISO/IEC, 2001).

2.2.2.1.2. Área Funcional Timing Manipulations

A linguagem XMT-O incorpora o módulo *TimeManipulations*, definido em SMIL 2.0, que permite a especificação de operações temporais sobre os objetos de mídia, como o controle da velocidade ou da taxa de apresentação dos objetos (*accelerate, decelerate, autoreverse, speed*). Esse tipo de manipulação é voltado para objetos de mídia contínua, como áudio e vídeo.

Além do módulo *TimeManipulations*, essa área funcional agrega o módulo *FlexTime*. Esse módulo, entre outros aspectos (Kim & Wood, 2002), define quais ações devem ser realizadas a fim de manter o sincronismo especificado pelos atributos do módulo *MinMaxTiming*, que especifica o tempo mínimo (*min*) e máximo (*max*) para apresentação de um objeto de mídia qualquer. Através dos atributos *flexBehavior e flexBehaviorDefault*, as ações a serem tomadas sobre um objeto de mídia são estabelecidas em ordem de prioridade. Entre essas ações destacam-se o descarte linear do conteúdo (*linear*), aplicado principalmente a exibições de vídeo que permitam o descarte de quadros, e a interrupção imediata da apresentação do objeto (*stop*).

2.2.2.1.3. Área Funcional *Animation*

A linguagem XMT-O incorpora os módulos de animação *BasicAnimation* e *SplineAnimation* de SMIL 2.0. O módulo *BasicAnimation* define elementos para animações responsáveis por alterações diversas como a definição do posicionamento dos objetos nos dispositivos para apresentação (*animate, set, animateMotion*), alterações da cor de preenchimento dos objetos (*animateColor*)

etc. O módulo *SplineAnimation* estende as possibilidades de animação, ao adicionar, aos elementos definidos pelo módulo *BasicAnimation*, atributos (*calcMode, keyTimes, keySplines*) para variações no cálculo da interpolação (*discrete, linear, paced*) e do tempo envolvidos na animação.

Complementarmente, a linguagem XMT-O define o módulo *DragAnimation* que, através de atributos específicos (*dragPlane, dragDisc, dragCylinder, dragSphere*), permite projetar animações interativas para os objetos sintéticos.

2.2.2.1.4. Área Funcional *Content Control*

Para o controle de conteúdo, a linguagem XMT-O incorpora os módulos *BasicContentControl, CustomTestAttributes, PrefetchControl e SkipContentControl*, todos definidos em SMIL 2.0.

O módulo *BasicContentControl* define um elemento (*switch*), que, em conjunto com atributos pré-definidos, tem por objetivo estabelecer condições para processar elementos nos documentos multimídia/hipermídia. Normalmente, esse módulo é utilizado com elementos para apresentação, onde caso o teste estabelecido seja verdadeiro, o elemento é apresentado, do contrário, o elemento é simplesmente ignorado.

O módulo *CustomTestAttributes* estende as funcionalidades do *BasicContentControl* permitindo que novas condições sejam estabelecidas (*customAttribute, customTest*), uma vez que originalmente, somente os testes pré-estabelecidos pela linguagem, especificados no módulo *BasicContentControl*, eram permitidos.

O módulo *PrefetchControl* oferece a possibilidade ao autor de definir operações de pré-busca (Jeong et al., 1997), influenciando o escalonamento das mídias, com o objetivo de favorecer aquelas que devem ser apresentadas imediatamente.

Finalmente, o módulo *SkipContent* define um elemento (*skip-content*), que permite que os elementos de um documento simplesmente não sejam considerados em uma apresentação.

2.2.2.1.5. Área Funcional *Layout*

A linguagem XMT-O especifica um módulo de leiaute com sintática semelhante à adotada em SMIL 2.0, no entanto, com uma estrutura semântica diferente. O leiaute da apresentação em XMT-O, da mesma forma que SMIL 2.0, é definido pelo elemento *layout*, no cabeçalho dos documentos multimídia/hipermídia. Porém, a estrutura do leiaute de apresentações XMT-O é formada por uma única janela (*toplayout*), que pode conter uma ou mais regiões (*region*), que, por sua vez, podem conter outras regiões recursivamente. Para serem exibidos, os objetos de mídia são associados às regiões. Em XMT-O, cada região é posicionada em relação aos eixos cartesianos, onde a posição inicial (coordenadas 0,0) corresponde ao centro da janela. Para posicionar as regiões em outros pontos, atributos de translação, definidos no módulo de *Layout*, devem ser utilizados.

2.2.2.1.6. Área Funcional *Linking*

A linguagem XMT-O especifica um único elemento para âncoras (*a* em conjunto com o atributo *href*) similar ao elemento homônimo definido em HTML (W3C, 1999). Particularmente, em XMT-O um elo somente pode ser definido entre duas cenas MPEG-4 distintas. Nesse caso, a cena de origem possui uma âncora associada a um objeto que, ao ser acionada, interrompe o desenvolvimento dessa cena e carrega a cena de destino.

2.2.2.1.7. Área Funcional *Media Objects*

Os módulos *MediaClipping*, *MediaClipMarkers* e *MediaDescription* definidos em SMIL 2.0 são totalmente incorporados pela linguagem XMT-O. Os módulos *MediaClipping* e *MediaClipMarkers* permitem, através de atributos (*clipBegin, clipEnd, readIndex*), destacar parte de um objeto de mídia contínua (áudio, vídeo), através da especificação de valores relativos ao tempo de início desse objeto. O módulo *MediaDescription* permite definir atributos com

descrições do conteúdo, como informações resumidas (*abstract*), informações de direitos autorais (*copyright*), autor (*author*) e título (*title*).

O módulo *MediaGroup*, definido pela linguagem XMT-O, especifica um único elemento, *group*, capaz de agrupar um conjunto de objetos. Na linguagem XMT-O esse elemento também é tratado como um objeto. Na realidade, esse elemento define um objeto composto dentro da hierarquia gráfica de uma cena MPEG-4, onde os demais objetos podem estar inseridos.

O módulo *xMedia*, também definido pela linguagem XMT-O, especifica os objetos de mídia tradicionais, tais como: imagens, vídeos e áudios (*img, video, audio*), definidos pelo padrão como objetos naturais. Nesse módulo também são definidos objetos que variam entre desenhos simples de duas dimensões como linhas e pontos, a objetos complexos de três dimensões (*box, cone, cylinder, sphere, mesh*). Esses objetos adicionais são denominados objetos sintéticos e representam uma importante funcionalidade da linguagem XMT quando comparada a outras linguagens como SMIL 2.0.

Entre os elementos sintéticos do módulo *xMedia*, alguns podem ser considerados blocos básicos para projetar objetos complexos. Os nomes dos elementos desse módulo abstraem a geometria dos objetos (*circle, rectangle* etc.) e, seus atributos, por sua vez, abstraem as características desses objetos. Eventualmente, algumas dessas características, quando mais complexas, podem ser especificadas através de elementos.

A linguagem XMT-O especifica também um módulo específico, denominado *MediaAugmentation*, com objetos que, apesar de serem parte da hierarquia gráfica de uma cena, são complementares aos objetos definidos em outros módulos. Os elementos desse módulo permitem especificar aspectos de luminosidade, aparência e plano de fundo (*backdrop, background, pointLight, spotlight*).

A linguagem XMT-O especifica ainda o módulo *CustomXMT-AMedia*, onde os elementos (*xmtaMedia, nodes, cmds*) oferecem mecanismos de escape para os autores que necessitarem embutir especificações da linguagem XMT-A diretamente na especificação de um documento na linguagem XMT-O.

2.2.2.1.8. Área Funcional *Metainformation*

A linguagem XMT-O incorpora o módulo *Metainformation* definido em SMIL 2.0. Esse módulo permite descrever o conteúdo de documentos XMT-O. Os elementos *metadata* e *meta* desse módulo definem metadados sobre o conteúdo do documento, que podem ser utilizados para especificar sua descrição semântica.

2.2.2.1.9. Área Funcional *Structure*

O módulo *Structure*, definido na linguagem XMT-O, especifica a estrutura básica de um documento XMT-O. São definidos o elemento raiz (*XMT-O*), o cabeçalho (*head*) e o corpo (*body*) do documento.

2.2.2.1.10. Área Funcional *Transitions*

A linguagem XMT-O permite realizar transições animadas dos objetos em apresentações incorporando os módulos *BasicTransitions*, *InlineTransitions* e *TransitionsModifiers*, definidos originalmente em SMIL 2.0. Esses módulos permitem especificar a forma como os objetos de mídia serão substituídos durante uma apresentação. As transições podem ser animadas de diversas formas, proporcionando um efeito visual agradável aos usuários.

2.2.2.1.11. Área Funcional *DEFS*

O módulo *DEFS*, especificado na linguagem XMT-O, permite que sejam especificadas referências (*def, use*) para conjuntos de elementos, facilitando o processo de autoria ao oferecer a possibilidade de reuso dos elementos no documento. Além do reuso, essa funcionalidade favorece a manutenção dos documentos. As definições e referências para conjuntos de elementos atuam como cópias idênticas do conjunto de elementos referenciados.

2.2.2.1.12. Área Funcional *Macros*

Complementarmente ao reuso de elementos e atributos, a linguagem XMT-O permite a utilização de macros. No módulo *Macros* essas estruturas são especificadas, permitindo definir, além de elementos e atributos, algumas operações, o que favorece ainda mais o reuso e conseqüentemente a autoria de documentos.

Quando uma macro é acionada, os valores dos atributos previamente definidos podem ser alterados. Esses atributos são denominados atributos virtuais (ISO/IEC, 2001) e podem variar de acordo com o contexto onde a macro for empregada. Como exemplo, pode-se definir condições para um conjunto de elementos, atributos ou valores associados em uma macro. Caso essas condições sejam satisfeitas, o conjunto de elementos, atributos ou valores a ela associados terão validade, caso contrário, outros elementos, atributos ou valores serão adotados, conforme a definição da macro (ISO/IEC, 2001).

2.3. Linguagem NCL (*Nested Context Language*) versão 2.0

A linguagem NCL 2.0, como citado no Capítulo 1, é uma linguagem declarativa e modular para autoria de documentos hipermídia baseada no modelo conceitual NCM (*Nested Context Model*). Devido ao fato de NCL ser baseada no modelo NCM, suas características são similares às desse modelo, compreendendo, por exemplo: o uso de composições para a estruturação lógica do documento; a especificação de relacionamentos temporais através de elos; o uso de conectores hipermídia (Muchaluat-Saade, 2003) para a autoria de elos; a possibilidade de escolha entre um conjunto de nós alternativos e a especificação da apresentação por meio de descritores. Em virtude da sua origem, para compreender as características da NCL é importante conhecer o modelo NCM. A Seção 2.3.1 realiza um breve resumo sobre a modelagem de documentos multimídia/hipermídia usando o NCM. Para obter maiores detalhes sobre esse modelo, o leitor deve consultar a referência (Soares et al., 2003).

Além dos aspectos herdados do modelo NCM, a linguagem NCL introduziu também o conceito de templates de composição hipermídia (Muchaluat-Saade,

2003) a fim de facilitar a autoria de documentos. A Seção 2.3.3 descreve os módulos de NCL, com destaque para o módulo *XTemplate* utilizado na especificação dos templates hipermídia.

2.3.1. Modelo Conceitual NCM (*Nested Context Model*)

No modelo conceitual NCM, um documento hipermídia é representado através de um nó de composição. Em NCM um nó é uma entidade básica que representa o conteúdo de um documento, podendo ser um objeto de mídia ou um nó de composição. Nós de composição podem conter um conjunto de nós, que podem ser objetos de mídia ou outros nós de composição, recursivamente. Os nós de composição também podem conter elos, relacionando seus nós internos. Uma restrição desse modelo impede que um nó contenha, recursivamente, ele próprio.

Elos definem um relacionamento fazendo referência a um conector hipermídia e a um conjunto de associações (*binds*). Um conector especifica a relação, sem mencionar os nós que irão participar do relacionamento. Conectores podem ser criados, através da linguagem de propósito específico *XConnector* (Muchaluat-Saade, 2003), representando relações de vários tipos, como: relações de sincronização, referência, derivação etc. Estruturalmente, um conector agrega um conjunto de pontos de interface, conhecidos como papéis (*roles*). Nos elos, os *binds* têm como finalidade associar os papéis de um conector a pontos de interface dos nós especificados em um documento, definindo a participação de cada elemento no relacionamento estabelecido. Como não há restrições ao número de papéis na definição estrutural de um conector, elos podem definir relacionamentos multiponto entre vários nós.

Os pontos de interface de um nó, associados através dos *binds* aos papéis de um conector, podem ser definidos como âncoras ou portas. Âncoras representam regiões (na verdade, um subconjunto das unidades de informação) de um nó qualquer. Por exemplo, para mídias contínuas como áudio e vídeo, âncoras podem ser definidas pelo intervalo temporal da exibição do objeto; ainda no caso do vídeo, âncoras podem também ser definidas pelos quadros que compõem o objeto. Portas são pontos de interface que existem somente em nós de composição. Um

mapeamento da porta para um ponto de interface de um dos nós internos dessa composição deve ser estabelecido.

No modelo NCM, os elos podem ser agrupados em bases de elos, permitindo o reuso. Pela mesma razão, os conectores hipermídia também podem estar agrupados em bases de conectores.

No modelo existe ainda a possibilidade de se agrupar um conjunto de nós alternativos, através de um nó *switch*. Esse nó permite que, na apresentação de documentos hipermídia, possam existir variações, resultantes da escolha entre os nós alternativos. Normalmente, as alternativas deverão ser escolhidas em virtude das características da plataforma de exibição, do usuário, ou de outros aspectos definidos como métricas para a escolha.

No NCM, a especificação para apresentação de um nó é feita separadamente de sua definição, através de um descritor. Descritores são elementos que reúnem informações, tais como: a ferramenta de exibição atribuída ao nó; a região do leiaute espacial de apresentação a que a exibição será associada; e outros parâmetros específicos de acordo com a mídia a ser exibida (volume, intensidade, aspecto etc.).

2.3.2. Documentos Hipermídia na Linguagem NCL

A Figura 2.7 apresenta a especificação de um documento hipermídia na linguagem NCL.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ncl id="coisadepeleNCL" ...>
  <head>
    <layout>
          <topLayout id="window1" ...>
             <region id="title" top=”0” left=”0” .../>
            ...
          </topLayout>
    </layout>
    <descriptorBase>
      <descriptor id="text_d0" region="title"/>
      ...
    </descriptorBase>
  </head>
  <body>
      <composite id="entrada1">
        <port id="coisadepele" .../>
          <audio descriptor="audio_d1" id="samba" ...">
            <area id="part1" begin="8.4s" end="18s"/>
            ...
```

```
        </audio>
        <img descriptor="img_d1" id="logoTelemidia" .../>
        ...
        <text descriptor="text_d0" id="title-coisaPele" .../>
        ...
        <linkBase>
          <link id="link01-01" xconnector="starts.xml" >
            <bind component="samba" role="on_x_presentation_begin"/>
             <bind component="logotele1" role="start_y"/>
          </link>
           ...
        </linkBase>
    </composite>
  </body>
</ncl>
```

Figura 2.7 – Exemplo de documento especificado em NCL 2.0

NCL divide a estrutura de um documento em duas partes, o cabeçalho (*head*) e o corpo (*body*), similar a SMIL 2.0, XMT-O e XMT-A. No cabeçalho, o leiaute da apresentação (*layout*) é definido, sendo composto por uma, ou mais janelas (*topLayout*), que, por sua vez, podem ser subdivididas em uma ou mais regiões (*region*). As regiões são as áreas onde os nós (objetos de mídia) serão apresentados.

No cabeçalho, as características de apresentação dos nós também são definidas, através do elemento *descriptor*. Esse elemento corresponde a uma instância declarativa do descritor de objetos do modelo NCM, definido na seção anterior.

O corpo do documento é, por si só, um nó de composição definido em NCM. Nesse corpo, outros nós de composição podem ser definidos, através do elemento *composite*. As composições, por sua vez, podem conter nós de mídia, como textos (*text*), imagens (*img*), vídeos (*video*), áudio (*audio*), outros nós de composição e bases de elos (*linkBase*).

Cada elo (*link*) de uma base de elos é formado por conectores, identificados pelo atributo *xconnector* e pelas associações entre os nós e esses conectores (*bind*). O exemplo da Figura 2.7 assume que o conector é apenas referenciado, sendo definido em um outro arquivo.

2.3.3. Módulos da Linguagem NCL

A especificação da linguagem NCL 2.0 foi realizada através de XML *Schema* (W3C, 2004), com a divisão de suas funcionalidades em módulos, assim

como a linguagem SMIL em sua versão 2.0, e a linguagem XMT-O. Essa abordagem modular facilita a interoperabilidade entre as linguagens, permitindo que módulos de uma sejam incorporados à outra. A Tabela 2.3 apresenta o agrupamento dos módulos da linguagem NCL, distribuídos nas suas 11 áreas funcionais. Como NCL 2.0 tem algumas funcionalidades idênticas às de SMIL 2.0, alguns módulos dessa linguagem são adotados em NCL. Os módulos de SMIL 2.0 utilizados em NCL aparecem na Tabela 2.3 com a expressão "SMIL" em destaque.

Áreas Funcionais	**Módulos**
Timing	*BasicTiming Module* *AdaptableTiming Module*
Components	*BasicMedia Module (SMIL)* *BasicComposite Module*
Presentation Specification	*BasicDescriptor Module* *CompositeDescriptor Module*
Presentation Control	*TestAttibutes Module* *ContentControl Module (similar ao BasicContentControl (SMIL))* *DescriptorControl Module*
Layout	*BasicLayout Module*
Linking	*Linking Module*
Interfaces	*MediaInterface Module* *CompositeInterface Module* *AttributeInterface Module* *SwitchInterface Module*
Metainformation	*Metainformation Module (SMIL)*
Structure	*StructureModule*
Connectors	*XConnector Module* *CompositeConnector Module*
Composite Templates	*XTemplate Module* *XTemplateUse Module*

Tabela 2.3 – Módulos que compõem a linguagem NCL 2.0

A definição completa dos módulos de NCL 2.0 pode ser encontrada na referência (Muchaluat-Saade, 2003), inclusive com suas restrições sintáticas, usando XML *Schema*. Entre as áreas funcionais dessa linguagem, destaca-se a *CompositeTemplates*, por oferecer facilidades não encontradas em nenhuma outra linguagem hipermídia (Muchaluat-Saade, 2003). Essa área funcional contém dois módulos, um chamado *XTemplateUse* e o outro *XTemplate*.

O módulo *XTemplateUse* permite que os templates definidos possam ser utilizados em nós de composição NCL. Isso é possível através do atributo (*xtemplate*), encontrado no elemento que representa o nó de composição, cujo conteúdo faz referência ao endereço (URI – *Uniform Resource Identifier*) de um template de composição pré-definido.

XTemplate consiste em uma linguagem de propósito específico XML voltada para a definição de templates de composição hipermídia. A especificação dessa linguagem, através de seu XML *Schema*, também pode ser encontrada na referência (Muchaluat-Saade, 2003).

Os templates de composição especificam tipos de componentes, tipos de relações, componentes e relacionamentos que uma composição possui ou pode possuir. Todos esses elementos são especificados sem identificar todos os componentes e relacionamentos, pois essa especificação fica sob responsabilidade da composição que utilizar o template. As especificações dos templates são definidas em duas partes, onde somente a primeira é obrigatória:

- *vocabulário*, onde os tipos de componentes (com seus pontos de interface) e conectores presentes em uma composição são definidos e,
- *restrições*, onde são definidas as regras sobre os elementos do vocabulário, incluídas instâncias de componentes e conectores, e relacionamentos entre os componentes.

A Figura 2.8 apresenta, como exemplo, a definição de um template de composição NCL. Esse exemplo, baseado no template apresentado em (Silva et al., 2004a), tem por objetivo definir uma sincronização temporal entre um objeto de áudio e uma imagem. Para isso, esse template define uma instância de um objeto de mídia imagem, denominado *logo*, uma instância de um objeto de mídia áudio, denominado *song*, e duas instâncias de elos, *L* e *P*, que representam relacionamentos entre esses objetos, onde o início do áudio implica no início da apresentação da imagem (elo *L*) e o término do áudio implica no término da apresentação da imagem (elo *P*).

No template da Figura 2.8, o vocabulário (*vocabulary*) contém a definição dos componentes, através do elemento *component*. Nessa parte são definidos o componente *song*, do tipo (*ctype*) áudio, e o componente *logo*, do tipo imagem (*ctype="img"*). Entre os componentes declarados, *song* pode conter pontos de interface (*port*). Esses pontos de interface são âncoras, denominadas *track*, que podem ser utilizadas na composição para demarcar trechos do conteúdo do áudio. Ainda no vocabulário, os conectores, através do elemento *connector*, são definidos. Na realidade essa definição dos conectores corresponde a uma referência a conectores previamente definidos em uma base de conectores. Em NCL, conectores são necessários para a criação de elos e, portanto, necessitam ser

definidos sempre que elos forem instanciados. Adicionalmente, nesse template pode ser definida a cardinalidade, através dos atributos *minOccurs* e *maxOccurs*, para componentes, pontos de interface e conectores.

Ainda na Figura 2.8, a segunda parte do template define suas restrições (*constraints*). Especificamente, as restrições são definidas pelo elemento *constraint*, com relação aos componentes, conectores e pontos de interface definidos no vocabulário e referenciados, nessa parte do template, através de seus tipos (*type*). A semântica da restrição é estabelecida através do atributo *select*, utilizando expressões definidas na linguagem XPath (W3C, 2005b). Expressões nessa linguagem, quando processadas, devem retornar um valor booleano, indicando se a restrição estabelecida foi ou não obedecida. Adicionalmente, o atributo *description* é utilizado para reportar uma mensagem de erro, nos casos em que a restrição não é satisfeita, isto é, ao ser processada, a expressão retorna o valor booleano falso. A restrição, estabelecida na Figura 2.8, define que a composição somente pode conter componentes do tipo *song* e *logo*. Essa restrição é formada por uma comparação entre os elementos pertencentes à composição, mais especificamente, a expressão *count(child::*[@type!='song'] | child::*[@type!='logo'])* obtém o total de elementos da união entre os conjuntos dos elementos que não possuem o atributo *type* igual a *song* com os que não possuem o atributo *type* igual a *logo*. O valor obtido nessa expressão deve ser igual ao número total de elementos pertencentes a composição (*count(child::*)*), subtraídos os elos declarados na composição (*count(child::linkBase)*). Ainda nas restrições, instâncias de componentes podem ser estabelecidas, através do elemento *resource*. Essas instâncias fazem referência a um tipo de componente (*type*) declarado previamente no vocabulário, bem como, ao conteúdo da instância, através de uma referência ao endereço (URI) do conteúdo. Na Figura 2.8, uma instância do tipo de componente *logo* é definida e identificada, através do seu atributo *label* como *logotele*.

Finalmente, instâncias de conectores, que definem a semântica de um template de composição NCL são definidas. Para a definição de elos (*link*), devem ser referenciados os tipos de componentes e conectores declarados na parte do vocabulário. Nos elos também podem ser referenciadas instâncias de componentes, especificadas nas restrições. Ao serem especificados nos templates, os elos possuem elementos filhos *binds*. Cada elemento *bind* relaciona os

componentes através de seleções, utilizando expressões XPath no atributo *select*; os papéis, por sua vez, são definidos pelo atributo *role*. No exemplo da Figura 2.8, são definidos dois elos, o primeiro faz referência ao conector *L* e o segundo ao conector *P*. Os elementos *binds* desses elos relacionam os papéis *source* e *target*, definidos internamente nos conectores, com a instância do tipo de componente *song* e a imagem instanciada no template *logotele*. É importante ressaltar que a instância do tipo de componente *song* deverá ser declarada na composição que adotar o template.

```
<xtemplate id="audio-with-subtitles">
  <vocabulary>
    <component type="song" ctype="audio" maxOccurs="1">
      <port type="track" maxOccurs="unbounded" />
    </component>
    <component type="logo" ctype="img" maxOccurs="1" />
    <connector src="connector_base.xml#L" type="L" maxOccurs="unbounded" />
    <connector src="connector_base.xml#P" type="P"  maxOccurs="unbounded" />
  </vocabulary>
  <constraints>
   <constraint select="count(child::*[@type!='song'] |
          child::*[@type!='logo']) = (count(child::*)-count(child::linkBase))">
          description="All components must be songs or logos"/>
   <resource src="http://www.telemidia.puc-rio.br/logo.jpg" type="logo" label="logotele"/>
   <link type="L">
      <bind role="source" select="child::*[@type='song']"/>
      <bind role="target" select="child::*[@type='logotele']"/>
   </link>
   <link type="P">
      <bind role="source" select="child::*[@type='song']"/>
      <bind role="target" select="child::*[@type='logotele']"/>
   </link>
  </constraints>
</xtemplate>
```

Figura 2.8 – Exemplo de template NCL

A Figura 2.9 ilustra um documento NCL contendo uma composição hipermídia que herda o template definido na Figura 2.8. Essa composição (*audio-with-logo*) especifica apenas o nó de áudio. O restante da sua especificação (a imagem logo e os elos de sincronização) serão herdados do template, ao qual ela faz referência (Figura 2.8). Durante o período de tempo relativo a execução do áudio, o logo (*http://www.telemidia.puc-rio.br/logo.jpg*) será apresentado.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ncl id="templateNCL" ...>
 <head>
  <layout>
    ...
  </layout>
  <descriptorBase>
    ...
```

```
    </descriptorBase>
  </head>
  <body>
     <composition id="template">
        <port id="inputTemplate" ... />
        <composition id=”audio-with-logo” xtemplate=”audio-with-subtitles.xml”>
            <audio type="audio" …  id="samba" src="coisadepele.wav"/>
        </compositon>
     </composition>
  </body>
</ncl>
```

Figura 2.9 – Documento NCL com composição herdando do template

Embora o exemplo apresentado seja bastante simples, ele pode ser facilmente estendido. Na referência (Muchaluat-Saade, 2003) é apresentado um template, nos moldes do apresentado na Figura 2.8, onde várias âncoras relativas a um objeto de áudio são sincronizadas com objetos de texto, definindo legendas associadas ao áudio.

Neste ponto, é importante ressaltar que os templates NCL, referenciados pelas composições, devem ser processados, em conjunto com as composições originais, modificando-as a fim de refletir as definições semânticas existentes nos templates. No documento NCL da Figura 2.9, antes da sua apresentação, o template referenciado (*audio-with-subtitles*) é consultado e a composição original (*audio-with-logo*) modificada, agregando as definições existentes no template. Outro item importante na utilização do módulo *XTemplate* de NCL 2.0, é a sua dependência em relação ao módulo *XConnector*. Na versão 2.0 de NCL a definição de templates tem como ênfase os relacionamentos espaço-temporais definidos por conectores, impedindo, portanto, a implementação dos templates sem o uso do módulo *XConnector*. Atualmente, adaptações ao módulo *XTemplate* (Silva et al., 2004a) tornaram-no mais flexível, permitindo sua utilização independente do módulo *XConnector*. Essas adaptações, bem como a proposta de templates para XMT-O, serão abordadas no Capítulo 5.

3
Conversão entre os Formatos NCL e MPEG-4

Um dos motivos para se definir padrões de codificação do conteúdo audiovisual é flexibilizar o intercâmbio desse conteúdo entre diversas ferramentas, facilitando a sua concepção, distribuição e reutilização.

Em uma visão geral, esses benefícios podem ser estendidos às aplicações multimídia/hipermídia. Porém, para essas aplicações, faz-se necessário, além do intercâmbio do conteúdo, prover o intercâmbio dos formatos, utilizados na descrição das relações entre os objetos de mídia que compõem os documentos multimídia/hipermídia.

As linguagens declarativas NCL 2.0 e MPEG-4 (XMT-O, XMT-A) favorecem o intercâmbio dos documentos de/para outros sistemas, por se basearem no formato padrão XML. No entanto, mesmo entre linguagens declarativas XML, é necessário realizar a tradução das expressões na linguagem de origem quando se quer representá-las em uma outra linguagem de destino.

A conversão entre as linguagens NCL 2.0 e MPEG-4 (XMT-O, XMT-A) é introduzida na Seção 3.1, através da análise dos seus objetivos e requisitos. A seguir, na Seção 3.2, a conversão entre NCL 2.0 e XMT-O é discutida. Esse processo de tradução é finalizado na Seção 3.3, através da conversão de XMT-O para XMT-A. Finalmente, a implementação do conversor, automatizando as tarefas de tradução, é apresentada na Seção 3.4.

3.1.
Objetivos e Requisitos

A conversão primária, objeto deste capítulo, consiste em representar no formato XMT-O, documentos, originalmente especificados no formato NCL, com a menor perda de representatividade possível. O formato XMT-O, entre os formatos MPEG-4, foi escolhido por possuir uma representação declarativa mais próxima de NCL, em relação ao XMT-A e BIFS. Porém, como XMT-O não é o formato de apresentação usual dos exibidores MPEG-4 (Joung & Kim, 2002),

deve também existir um conversor, dos documentos XMT-O obtidos, para o formato XMT-A. Nesse ponto, a conversão para BIFS é direta, podendo inclusive ser realizada por ferramentas licenciadas como código aberto.

O intercâmbio de documentos NCL para linguagens MPEG-4 possui várias vantagens, entre as quais podem ser destacadas:

- O armazenamento integrado da especificação dos relacionamentos com o conteúdo dos objetos de mídia;
- A distribuição de documentos NCL através de servidores de fluxos, facilitando sua divulgação a vários usuários (clientes) simultâneos;
- A apresentação de documentos NCL em exibidores MPEG-4[7].

Entre os formatos MPEG-4, XMT-A e BIFS são diretamente intercambiáveis, ou seja, existe uma correspondência biunívoca entre suas entidades. A conversão de XMT-O para XMT-A, e conseqüentemente para BIFS, também é possível, apesar do fato de que algumas construções de XMT-O poderão corresponder a várias representações no formato XMT-A, cabendo ao conversor escolher uma das possíveis representações (Kim et al., 2000). Porém, a conversão de XMT-A, ou BIFS, para XMT-O não é trivial, pois nesses casos pode ser necessário descobrir as intenções do autor, num nível mais alto de abstração, a partir de especificações num paradigma de sincronização *timeline*, utilizadas pelas duas primeiras, como abordado na Seção 3.3.

A abordagem de conversão proposta nesta dissertação possui uma desvantagem em termos de eficiência, quando a conversão de documentos NCL tem por objetivo a representação no formato XMT-A, ou BIFS, pois, nesses casos, duas ou três conversões em seqüência precisam ser aplicadas. Por essa razão, pode-se, como trabalho futuro, sintetizar os aspectos da conversão em módulos únicos.

Como objetivo complementar, documentos no formato XMT-O devem poder ser convertidos para o formato NCL. Essa conversão segue os procedimentos inversos da conversão de documentos NCL para XMT-O. Entre as vantagens desse intercâmbio destacam-se:

---

[7] Espera-se desses exibidores, por adotarem um padrão ISO/IEC, uma distribuição em escala industrial, de forma similar ao que atualmente ocorre aos exibidores compatíveis com os padrões MPEG anteriores (MPEG-1, MPEG-2).

- A preservação da semântica dos relacionamentos especificados em XMT-O, através da sua representação utilizando conectores NCL;
- A disponibilidade de ferramentas para autoria (Costa, 1996; Pinto, 2000; Moura, 2001; Coelho, 2004), controle de versões (Batista, 1994), trabalho cooperativo (Gorini, 2001) e apresentação (formatação) (Rodrigues, 2003), fornecidas pelo sistema HyperProp.

Além das vantagens citadas, a integração NCL / XMT-O expõe as construções de XMT-O que não possuem representação em NCL, trazendo como contribuição a possibilidade de uma eventual inclusão dessas estruturas em uma nova versão de NCL.

3.2. Tradução entre NCL e XMT-O

Em alguns casos da conversão entre NCL 2.0 e XMT-O, tem-se uma conversão direta, uma vez que seus módulos são idênticos. Em outros casos, essa conversão necessita de uma reestruração completa, considerando a existência de módulos que possuem funções semelhantes, porém estruturas distintas. Os casos mais complexos, entretanto, são aqueles onde não existem representações comuns entre as linguagens, como, por exemplo, o módulo dos conectores NCL.

Com exceção dos casos onde a conversão entre os módulos de NCL 2.0 e XMT-O é direta, nos demais podem existir restrições para a tradução, sendo que, para alguns módulos, essas restrições podem inviabilizar a tradução. A fim de destacar os módulos onde a conversão é possível, nesta seção são propostos perfis[8], que definem documentos interoperáveis entre as linguagens.

Para favorecer a compreensão das tarefas desenvolvidas no processo de conversão, no restante desta seção apenas pontos críticos serão apresentados. Os detalhes envolvendo a tradução de atributos e entidades, bem como os exemplos das traduções, são abordados no Apêndice B. Durante todo o restante deste capítulo, as referências à linguagem NCL corresponderão a sua versão 2.0.

[8] Perfis de linguagem, normalmente, estabelecem um subconjunto dos módulos de uma linguagem.

3.2.1. Tradução de NCL para XMT-O

Os módulos de NCL, apresentados na Tabela 3.1, definem um perfil denominado NCL-XMT, cujos módulos podem ser convertidos para XMT-O e, conseqüentemente, para XMT-A. Esse perfil tem como objetivo conter todos os módulos de NCL que possuem alguma representação em XMT-O, mesmo que exista algum tipo de restrição para a tradução.

Áreas Funcionais	Módulos
Structure	*Structure Module*
Components	*BasicMedia Module*
Interfaces	*MediaInterface Module*
Linking	*Linking Module*
Connectors	*XConnector Module*
Composite Templates	*XTemplate Module* *XTemplate Use*
Timing	*BasicTiming Module* *AdaptableTiming Module*
Layout	*BasicLayout Module*
Presentation Specification	*BasicDescriptor Module*
Presentation Control	*ContentControl Module* *TestAttributes Module* *DescriptorControl Module*
Metainformation	*Metainformation Module*

Tabela 3.1 – Módulos de NCL que compõem o perfil NCL-XMT

Entre os módulos apresentados na Tabela 3.1, apenas os módulos *Structure*, *Metainformation* e *BasicTiming* podem ser diretamente convertidos para XMT-O, não apresentando, portanto, restrições nas traduções. Os dois primeiros são convertidos para os módulos homônimos de XMT-O e, o último, para o módulo de XMT-O denominado *BasicInlineTiming*.

Os módulos *BasicMedia*, *AdaptableTiming*, *BasicLayout*, *TestAttributes* e *ContentControl*, apresentados na Tabela 3.1, possuem especificações semelhantes às construções especificadas por alguns módulos de XMT-O. No entanto, para a tradução entre os respectivos módulos dessas linguagens, faz-se necessário algum tipo de reestruturação. Entre os módulos citados, destaca-se o *BasicMedia*, que define os objetos de mídia de um documento NCL, pois a maior parte da sua especificação pode ser diretamente traduzida para as construções do módulo *xMedia*, definido em XMT-O. Apenas alguns dos tipos básicos especificados no *BasicMedia*, bem como alguns de seus atributos, precisam ser interpretados e representados por outras construções em XMT-O, conforme discutido no Apêndice B.

Os módulos *BasicTiming* e *AdaptableTiming*, que especificam o comportamento temporal da apresentação de um componente em NCL, possuem construções próximas ao módulo *FlexTime* de XMT-O. No módulo *AdaptableTiming,* uma função de custo pode ser especificada. Nessa função são especificados os valores, relativos ao tempo da apresentação, onde a taxa de exibição do objeto pode ser aumentada ou diminuída, bem como o custo dessas operações. Por outro lado, no módulo *FlexTime,* a adaptação é especificada através da definição dos tempos mínimo, ideal e máximo para a apresentação, sem definir custos específicos relativos aos tempos mínimo e máximo. No *FlexTime* também é possível especificar as ações a serem executadas a fim de alcançar os limites de tempo estabelecidos. Dessa forma, na conversão das construções do módulo *AdaptableTiming* para XMT-O, os tempos para exibição podem ser representados sem a definição dos seus custos relativos.

O módulo *BasicLayout* de NCL, na conversão para XMT-O, também necessita de adaptações, pois define uma estrutura de apresentação diferente da estabelecida pelo módulo respectivo em XMT-O, denominado *Layout*. No módulo *BasicLayout* a estrutura para apresentação de um documento é formada por um conjunto de janelas, cada qual podendo conter um conjunto de regiões, que, por sua vez, podem conter outras regiões, recursivamente. No entanto, em XMT-O, o módulo *Layout*, apesar de especificar as regiões de forma similar ao *BasicLayout*, define uma estrutura formada por uma única janela. Assim, na tradução das construções especificadas pelo módulo *BasicLayout* para o módulo *Layout*, uma única janela deve ser construída, correspondendo a todas as janelas originais de NCL. Além disso, pelo fato de NCL adotar o sistema de coordenadas geométricas e XMT-O o sistema cartesiano, conversões nas especificações dos valores relativos ao posicionamento e às dimensões das regiões tornam-se necessárias na tradução entre as linguagens.

Os demais módulos de NCL que, embora possuam representação em XMT-O, necessitam de adaptações são definidos pela área funcional *Presentation Control*. Na conversão de documentos NCL para XMT-O, as construções dos módulos *ContentControl* e *TestAttributes* podem ser representadas através do módulo *BasicContentControl*. No *ContentControl,* um conjunto de componentes alternativos pode ser especificado através de um nó *switch*. Complementarmente, os atributos de teste, que estabelecem as regras a serem consideradas para a

escolha entre os componentes alternativos, são especificados através das construções do *TestAttributes*.

Em XMT-O, o módulo *BasicContentControl*, de forma similar ao *ContentControl* de NCL, também especifica um nó *switch* para a escolha de alternativas. No entanto, esse nó em XMT-O possui algumas diferenças em relação ao nó homônimo definido em NCL. Entre essas diferenças, destaca-se a possibilidade, oferecida por NCL, de associar, através de elementos específicos (*bindrule*), as regras estabelecidas para a escolha dos nós alternativos. Essa forma de associação permite o reuso dos nós, independente das regras estabelecidas sobre eles, pois as regras encontram-se relacionadas ao nó apenas no contexto da composição *switch*. Para realizar a conversão para XMT-O as regras estabelecidas devem ser diretamente distribuídas aos nós alternativos, na forma de atributos pertencentes aos nós.

Para a definição das regras, atributos de testes são pré-estabelecidos em XMT-O, através das construções especificadas pelo módulo *BasicContentControl*. Em NCL, os atributos de teste, especificados através do módulo *TestAttributes*, são, originalmente, iguais aos definidos pelo módulo *BasicContentControl*. No entanto, ao contrário do *BasicContentControl*, o módulo *TestAttributes* exige que os atributos de teste sejam declarados no cabeçalho do documento, onde o nome do atributo e o seu valor associado devem ser especificados (Muchaluat-Saade, 2003). Essa declaração explícita ocorre em XMT-O apenas quando os autores necessitam especificar seus próprios atributos de teste, além daqueles pré-definidos pela linguagem, utilizando para isso, as construções do módulo *CustomTestAttributes*. O módulo *TestAttributes* de NCL, pelas suas características, pode ser estendido, a fim de especificar também atributos específicos definidos pelos autores. Nesse caso, as construções desse módulo podem ser convertidas para construções do módulo *CustomTestAttributes*.

Os demais módulos do perfil NCL-XMT, apresentados na Tabela 3.1, correspondem aos casos mais complexos de conversão, pois não possuem módulos correspondentes em XMT-O. Entre eles, o módulo *BasicDescriptor* especifica elementos descritores, não definidos em XMT-O. Em XMT-O, as informações relativas aos descritores NCL encontram-se especificadas diretamente nos objetos. Assim, na conversão de NCL para XMT-O, as

informações contidas nos descritores devem ser distribuídas nos atributos dos objetos que os referenciam.

Também sem correspondência em XMT-O, o módulo *DescriptorControl* especifica descritores que definem um conjunto de opções alternativas, destinadas a adaptações no documento. Na conversão para XMT-O, um conjunto de descritores alternativos torna-se um conjunto de objetos alternativos, cujas diferenças são apenas as informações contidas originalmente nos seus descritores.

Outros módulos de NCL que não possuem representação em XMT-O são aqueles para especificação de interfaces. Em XMT-O, as únicas interfaces definidas são as âncoras embutidas nos elos dessa linguagem (ISO/IEC, 2001). Nos demais relacionamentos, como os estabelecidos através de eventos, XMT-O não define o uso de interfaces[9]. Apesar dessa restrição, as construções do módulo *MediaInterface*, pertencente ao perfil NCL-XMT, podem ser parcialmente representadas em XMT-O através dos uso dos objetos sintéticos, especificados no módulo *xMedia*. O módulo *MediaInterface* especifica âncoras para os objetos de mídia da linguagem NCL, cujas propriedades (Muchaluat-Saade, 2003) são similares às dos objetos sintéticos que representam figuras geométricas em XMT-O, como retângulos e círculos (ISO/IEC, 2001).

Para os elos e conectores de NCL, especificados pelos módulos *Linking* e *XConnector*, respectivamente, ambos pertencentes ao perfil NCL-XMT, também não existem módulos correspondentes em XMT-O. Como já mencionado, em XMT-O os relacionamentos são definidos através de composições com semântica de sincronização, especificadas através dos módulos *BasicTimeContainers* e *ExclTimeContainers*. Além das composições, eventos, especificados principalmente através dos módulos *XMTEvents* e *EventTiming* de XMT-O, também são empregados para especificar relacionamentos entre objetos de um documento.

Elos de XMT-O são especificados através do seu módulo *Linking*, definindo relacionamentos entre cenas audiovisuais distintas. Os elos e conectores definidos em documentos NCL podem ser representados, com algumas restrições, através das composições e eventos de XMT-O. Os conectores NCL são especializados de

[9] Exceto pelas âncoras que podem ser definidas no próprio conteúdo dos objetos de mídia, através do módulo *MediaMarkerTiming* (ISO/IEC, 2001).

acordo com a sua semântica, em conectores causais ou de restrição (Muchaluat-Saade, 2003). Independente do tipo do conector, a função dos participantes é determinada por um conjunto de papéis que serão associados a eventos.

Nos conectores causais, condições devem ser satisfeitas para a execução de ações. Nesses conectores, os papéis podem corresponder às condições ou às ações de uma relação. Na realidade, para cada evento, seus estados e transições, controlados pela máquina de estados específica desse evento (Muchaluat-Saade, 2003), podem ser empregados na especificação dos papéis de um conector.

Na conversão de documentos NCL contendo conectores causais para XMT-O, os eventos e papéis desses conectores devem ser interpretados para a tradução. Papéis que definem eventos de apresentação podem ser representados por construções especificadas através dos módulos *EventTiming* ou *SyncbaseTiming*, ambos de XMT-O. O módulo *EventTiming* define os atributos de início (*begin*) e fim (*end*), que podem ser utilizados para representar o evento de apresentação nas ações estabelecidas pelos conectores. Complementarmente, o módulo *SyncbaseTiming* define o uso das mesmas expressões (*begin* e *end*), relacionadas a algum objeto do documento, podendo representar os eventos de apresentação como condição do conector causal. Além do *SyncbaseTiming*, o módulo *XMTEvents* especifica uma série de outros eventos (ISO/IEC, 2001) que podem ser utilizados para representar as condições de um conector causal.

A especificação de como os papéis de um conector são relacionados é definida por um elemento denominado *glue*. Em um conector com semântica causal, o *glue* define tanto uma expressão de condição (*condition-expression*) quanto uma expressão de ações (*action-expression*), que podem ser simples ou composta (Muchaluat-Saade, 2003). Uma expressão de condição simples pode ser representada em XMT-O, desde que os eventos e transições relacionados ao conector sejam definidos nos módulos de XMT-O. Por outro lado, as expressões de condição composta, que permitem relacionar qualquer número de papéis através de operadores booleanos, somente podem ser representas no caso do operador "ou" (*or*), através do módulo *MultiArcTime* de XMT-O. De forma similar, uma expressão de ações simples pode ser representada em XMT-O, desde que os eventos relacionados a essa ação sejam definidos nos módulos dessa linguagem. As expressões de ações compostas são definidas por expressões utilizando os operadores *par*, *seq* ou *excl*, envolvendo outras expressões de ações.

Essas expressões, caso contenham eventos que iniciem a apresentação dos objetos, podem ser traduzidas para as composições da linguagem XMT-O, conforme apresentado no Apêndice B. Para os demais eventos essa tradução é restrita a casos específicos.

Ao contrário dos conectores com semântica causal, os conectores com semântica de restrição não possuem representação em XMT-O. Nesses conectores os papéis estão associados a alguma propriedade. Como exemplo, um conector desse tipo pode estabelecer uma relação onde dois objetos devem estar alinhados pelo topo, o que significa que as suas propriedades de deslocamento nessa dimensão devem ser iguais. Em XMT-O, esse tipo de especificação define uma referência cruzada entre as propriedades dos objetos da cena audiovisual, sem expressar o valor efetivo dessas propriedades.

Por fim, na Tabela 3.1, os módulos da área funcional *Composite Templates*, denominados *XTemplate Module* e *XTemplate Use*, também não possuem representação nos módulos de XMT-O, porém são considerados nesta dissertação componentes de pré-compilação. Portanto, antes da conversão efetiva entre NCL e XMT-O, esses módulos devem ser traduzidos para outras construções dentro da própria linguagem NCL.

Os módulos de NCL que apresentam uma forte restrição para a representação em XMT-O e que, portanto, não foram incluídos nesta dissertação no perfil NCL-XMT são apresentados na Tabela 3.2. Um desses módulos é o *BasicComposite*, que em NCL é responsável pela definição dos nós de composição. Composições em NCL representam relacionamentos de inclusão sem outra semântica associada. Por outro lado, as composições de XMT-O possuem semântica de sincronização embutida. Dessa forma não é possível representar em XMT-O as relações de inclusão isoladamente.

Áreas Funcionais	**Módulos**
Components	*BasicComposite Module*
Interfaces	*CompositeInterface Module* *AttributeInterface Module* *SwitchInterface Module*
Connectors	*CompositeConector Module*
Presentation Specification	*CompositeDescriptorModule*

Tabela 3.2 – Módulos de NCL não pertencentes ao perfil NCL-XMT

Devido as restrições na tradução das composições de NCL para XMT-O, outros módulos também tornam-se restritos, como os módulos *CompositeInterface*, *SwitchInterface*, *CompositeConector* e *CompositeDescriptor*

(Muchaluat-Saade, 2003). Além dos módulos relacionados às composições, o módulo *AttributeInterface,* que permite a definição de atributos como pontos de interface, também possui restrições de representação em XMT-O, pois nessa linguagem, de forma geral, não são previstas alterações ou comparações nos valores dos seus atributos.

3.2.2. Tradução de XMT-O para NCL

A Tabela 3.3 apresenta os módulos de XMT-O que fazem parte do perfil XMT-NCL. Os módulos desse perfil, de forma complementar aos módulos do perfil NCL-XMT, podem ser convertidos para os módulos da linguagem NCL, ainda que com algumas restrições para a tradução.

Áreas Funcionais	Módulos
Timing	*BasicInlineTiming Module* *BasicTimeContainers Module* *ExclTimeContainers Module* *EventTiming Module* *MinMaxTiming Module* *MultiArcTiming Module* *RepeatTiming Module* *SyncbaseTiming Module* *XMT Events Module*
Time Manipulations	*FlexTime Module*
Content Control	*BasicContentControl Module* *CustomTestAttributes Module* *PrefetchControl Module*
Layout	*Layout Module*
Linking	*Linking Module*
Media Objects	*xMedia Module* *MediaDescription Module*
Metainformation	*Metainformation Module*
Structure	*Structure Module*
Macros	*Macros Module*
DEFS	*DEFS Module*

Tabela 3.3 – Módulos de XMT-O que compõem o perfil XMT-NCL

Entre os módulos da Tabela 3.3, os módulos *Structure*, *Metainformation* e *BasicInlineTiming*, conforme citado na seção anterior, podem ser diretamente convertidos para os módulos *Structure*, *Metainformation* e *BasicTiming* de XMT-O.

Outros módulos, apresentados na Tabela 3.3, foram abordados anteriormente, como os módulos *xMedia*, *Flextime*, *Layout, BasicContentControl* e *CustomTestAttributes*, relativos, respectivamente, aos módulos *BasicMedia*, *AdaptableTiming*, *BasicLayout*, *ContentControl* e *TestAttributes*, especificados

em NCL. Entre esses módulos, o *xMedia* possui elementos sem representação no módulo *BasicMedia,* como os objetos sintéticos de duas e três dimensões (ISO/IEC, 2001). Ainda nesse módulo, as informações relativas à apresentação devem ser traduzidas para as especificações do módulo *BasicDescriptor* de NCL. Complementarmente, o *BasicDescriptor* contém as informações do módulo *MinMaxTiming* de XMT-O, relativas aos tempos para apresentação de um determinado objeto.

As construções relativas ao módulo *FlexTime* de XMT-O, quando convertidas para NCL, podem ser parcialmente representadas através das especificações do módulo *AdaptableTiming*. Como o módulo *FlexTime* não define o intervalo de tempo da apresentação em que a ação escolhida pode ser aplicada, nem o custo associado a essa ação, esses valores devem ser informados na conversão dos documentos.

As construções do módulo *Layout*, quando traduzidas para as especificações do *BasicLayout* apresentam restrições nas unidades dos valores, relativos às dimensões e ao posicionamento da janela e regiões, uma vez que XMT-O utiliza o sistema de coordenadas cartesianas e, NCL, o geométrico. Além dessa diferença, em XMT-O, os valores dessas dimensões podem ser especificados em *pixels* ou metros (*meter*). Caso o sistema métrico seja utilizado, na conversão para NCL é necessária a tradução dos valores dessas dimensões para *pixels*, medida adotada em NCL.

Para a conversão das construções do módulo *BasicContentControl* de XMT-O, as especificações dos módulos *ContentControl* e *TestAttributes* de NCL são empregadas. Em XMT-O, as regras estabelecidas para a escolha dos objetos são definidas nos seus atributos. Na conversão para NCL, essas regras, mesmo que pré-estabelecidas, devem ser explicitamente declaradas através das construções do módulo *TestAttributes*. Como esse módulo realiza a declaração das regras, ele pode ser estendido, a fim de declarar regras próprias dos autores, que em XMT-O são declaradas por construções do módulo *CustomTestAttributes*.

Os demais módulos apresentados na Tabela 3.3, exceto os módulos *Macros, DEFS* e *PrefetchControl*, são destinados à especificação de relacionamentos em documentos XMT-O. As composições de XMT-O com semântica de sincronização podem ser representadas através de nós e elos NCL. No Apêndice B os modelos para essas representações são apresentados. Em relação aos eventos,

os módulos *EventTiming*, *RepeatTiming*, *MultiArcTiming*, *SyncbaseTiming* e *XMTEvents* podem ser traduzidos para conectores causais, especificados através do módulo *XConnector*. Entre esses módulos, o *EventTiming* define os eventos, associados aos papéis do tipo ação do conector causal. Na tradução, apenas o evento NCL de apresentação (*presentation*) é utilizado, com sua transição de início (*starts*) ou de fim (*stops*), dependendo do atributo especificado em *EventTiming* (*begin*, *end*). Por outro lado, os módulos *SyncbaseTiming* e *XMTEvents* definem os eventos associados aos papéis do tipo condição do conector causal. A Tabela 3.4 apresenta os eventos definidos nesses módulos e a correspondência com os eventos definidos em NCL.

Módulo XMT-O	Eventos XMT-O	Eventos NCL	Transições
EventTiming	*begin*	*presentation*	*starts*
EventTiming	*end*	*presentation*	*stops*
SyncbaseTiming	*begin*	*presentation*	*starts*
SyncbaseTiming	*end*	*presentation*	*stops*
XMTEvents	*click*	*mouseclick*	*starts*
XMTEvents	*mouseover*	*mouseover*	*starts*

Tabela 3.4 – Correspondência entre os eventos do perfil XMT-NCL e NCL

O módulo *RepeatTiming* de XMT-O, quando traduzido para NCL, especifica o número de vezes que um evento de apresentação deve ocorrer. O módulo *MultiArcTiming* permite que múltiplas condições sejam associadas à ocorrência de um evento, correspondendo a uma expressão de condição composta do conector. Além desses módulos, o *PrefetchControl* de XMT-O também pode ser representado por conectores NCL que definem um evento de *prefetch* (Muchaluat-Saade, 2003).

Por fim, na Tabela 3.3 os módulos *Macros* e *DEFS* são considerados componentes de pré-compilação, de forma similar aos módulos dos templates de composição hipermídia da linguagem NCL, devendo ser traduzidos para outras construções de XMT-O, antes de serem convertidos para NCL.

Os módulos de XMT-O que apresentam uma forte restrição para a representação em NCL e que, portanto, não estão incluídos no perfil XMT-NCL são apresentados na Tabela 3.5. As especificações desses módulos são descritas no Capítulo 2.

Áreas Funcionais	Módulos
Timing	*AccessKeyTime Module* *FillDefault Module* *MediaMarkerTiming Module* *RepeatValueTiming Module* *RestartDefault Module* *RestartTiming Module* *SyncBehavior Module* *SyncBehaviorDefault Module* *SyncMaster Module*
Time Manipulations	*TimeManipulations Module*
Animation	*BasicAnimation Module* *SplitAnimation Module* *DragAnimation Module*
Media Objects	*MediaClipping Module* *MediaClipMarkers Module* *MediaGroup Module* *MediaAugmentation Module* *CustomXMT-AMedia Module*
Transitions	*BasicTransitions Module* *InlineTransitions Module* *TransitionsModifiers Module*

Tabela 3.5 – Módulos de XMT-O não pertencentes ao perfil XMT-NCL

Entre os módulos apresentados na Tabela 3.5, o *MediaMarkerTiming*, *SyncBehavior*, *SyncBehaviorDefault* e *SyncMaster* possuem funções específicas, herdadas de SMIL 2.0 e adaptadas às características do MPEG-4 *Systems*.

O módulo *AccessKeyTime* define eventos com origem no teclado, que poderiam estar incluídos aos eventos de NCL, de forma similar aos eventos obtidos através do mouse (*mouseclick*, *mouseover*).

Os módulos *FillDefault*, *RepeatValueTiming*, *RestartTiming* e *RestartDefault* definem condições para a execução de um evento ou, no caso do *FillDefault*, o estado em que um objeto deve se encontrar quando a sua exibição termina. O módulo *TimingManipulations* em XMT-O permite ao autor especificar operações temporais sobre os objetos de mídia contínua. Essas operações não são diretamente oferecidas em NCL, razão pela qual não foi explorada a conversão desse módulo nesta dissertação. Outros módulos que não possuem ainda representação em NCL, são os módulos de XMT-O relativos às áreas funcionais *Animation* e *Transitions*. Embora presentes na versão 1.0 de NCL e presentes na versão 2.0, ainda não foram exploradas as definições de conectores para especificação de relacionamentos espaciais e baseados em atributos que poderiam dar suporte à animação dos objetos de mídia pertencentes a uma apresentação (Moura, 2001).

Por fim, entre os módulos da área funcional *Media Objects*, o módulo *CustomXMT-AMedia*, que permite a inserção de especificações em XMT-A nos documentos XMT-O, poderia ter suas construções representadas em NCL, no entanto, como a tradução direta de construções em XMT-A para NCL não foi abordada, essa representação não foi considerada na tradução de XMT-O para NCL.

3.3. Tradução de XMT-O para XMT-A

Na referência (ISO/IEC, 2001) são definidas as traduções das construções no formato XMT-O para XMT-A. Nessa tradução a especificação em XMT-A, além de representar as expressões em XMT-O, deve atender aos requisitos definidos pela arquitetura MPEG-4 *Systems*. Para isso, pode ser necessário especificar outras informações, ausentes nos documentos XMT-O, tais como:

- Descritor de objetos inicial;
- Perfis e níveis adotados pela apresentação;
- Descritor de cenas para os comandos BIFS utilizados;
- Descritor de objetos, caso objetos de mídia sejam utilizados;
- Atualização no descritor de objetos dos objetos de mídia utilizados;
- Definição de outros fluxos que sejam considerados necessários (OCI, IPMP, MPEG-J etc.).

Para cada informação adicionada ao documento XMT-A, vários dados complementares podem ser fornecidos. Para o descritor de cenas, por exemplo, podem ser fornecidas informações sobre o tamanho do *buffer* para armazenamento no receptor, sobre a prioridadade desse fluxo entre os demais, sobre a sua dependência em relação a outros fluxos, entre outras.

Devido ao fato do padrão MPEG-4 já definir um mapeamento geral do formato XMT-O para XMT-A, esta seção irá concentrar-se na tradução dos seus relacionamentos, pela importância desse aspecto nos sistemas multimídia/hipermídia. Outros aspectos da conversão podem ser obtidos em (ISO/IEC, 2001).

Entre os relacionamentos definidos no formato XMT-A, destacam-se os relacionamentos de sincronização temporal. A especificação desses

relacionamentos tradicionalmente segue o paradigma *timeline*, onde a sincronização é definida através do posicionamento dos componentes de uma apresentação em relação a um eixo do tempo, ao contrário do paradigma de eventos de restrição e causalidade seguido por NCL e XMT-O.

O paradigma de *timeline* apresenta várias limitações, como, por exemplo, a restrição para representar eventos temporais de duração variável ou desconhecida antes da execução[10], e dificuldades no reuso das especificações e na manutenção das apresentações, pois as mudanças individuais nos elementos para apresentação podem alterar toda a distribuição da sincronização.

Além dessas limitações, a ausência de relacionamentos que especifiquem as intenções do autor em relação aos objetos na cena, uma vez que essas intenções são perdidas na linearização do *timeline*, pode comprometer o sincronismo de toda a apresentação. Geralmente, objetos sincronizados possuem um significado em conjunto, motivando o estabelecimento desse relacionamento, como, por exemplo, a exibição sincronizada de um vídeo e um áudio para apresentação de um filme. Como os atrasos em um dos objetos sincronizados pode comprometer toda a apresentação, pode-se adotar estratégias de ajuste (Rodrigues, 2003), a fim de garantir que os relacionamentos sejam obedecidos e, conseqüentemente, a qualidade da apresentação. Em um paradigma *timeline*, como as intenções do autor na concepção do documento não são conhecidas pelo exibidor, tais operações de ajustes tornam-se impossíveis, pois o exibidor não tem como inferir os relacionamentos pelo simples posicionamento temporal entre os objetos da apresentação.

Na Figura 3.1 um documento XMT-O define uma composição seqüencial (*seq*) contendo duas composições paralelas (*par*). Cada composição paralela, por sua vez, contém dois objetos, um do tipo áudio e outro do tipo vídeo, exibidos simultaneamente quando a composição é executada. As composições paralelas são subseqüentes, isto é, a segunda composição declarada será executada logo após a primeira.

---

[10] Documentos hipermídia especificados em XMT-O, onde relacionamentos de sincronização são estabelecidos com a participação de um nó *switch*, não podem ser representados em XMT-A, pois os componentes alternativos desse nó podem conter durações distintas para exibição, sendo desconhecido, antes da execução do documento, o componente escolhido.

```
<XMT-O xmlns=”urn:mpeg:mpeg4:xmto:schema:2002"
          xsi:schemaLocation="urn:mpeg:mpeg4:xmto:schema:2002 xmt-o.xsd"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xlink="http://www.w3.org/1999/xlink">
 <head>
  <layout type="xmt/xmt-basic-layout" metrics="pixel">
   <topLayout height="120" width="160"/>
  </layout>
 </head>
 <body>
  <seq>
   <par>
    <img src="../multimedia/imagem1.jpg" /><audio src="../multimedia/audio1.mp3"/>
   </par>
   <par>
    <img src="../multimedia/imagem2.jpg" /><audio src="../multimedia/audio2.mp3"/>
   </par>
  </seq>
 </body>
</XMT-O>
```

Figura 3.1 – Documento XMT-O com sincronização temporal

Na conversão do documento apresentado na Figura 3.1 para a linguagem XMT-A, ao invés de composições com semântica de sincronização seqüencial e paralela, por restrições dessa linguagem, apenas duas composições paralelas serão representadas. A primeira composição paralela de XMT-A, que será executada a partir do início da apresentação, será responsável pela exibição de dois objetos, relativos aos objetos da primeira composição paralela de XMT-O especificada na Figura 3.1. Na segunda composição paralela de XMT-A, dois objetos serão exibidos, relativos aos objetos de imagem e áudio da segunda composição paralela de XMT-O também especificada na Figura 3.1. Seguindo o paradigma *timeline*, o início da execução dessa segunda composição será determinado pelo cálculo do tempo de apresentação do objeto de áudio, especificado na primeira composição de XMT-A.

Durante a apresentação de um documento hipermídia, como o documento XMT-O especificado na Figura 3.1, atrasos, impossíveis de serem previstos, podem ocorrer. Esses atrasos podem ser provocados pelo sistema operacional utilizado na aplicação de exibição, pelo acesso ao conteúdo através de uma rede de comunicações etc. Nesse caso, a ferramenta de exibição, conhecendo as intenções do autor, pode realizar ajustes na apresentação, garantindo que os objetivos na concepção do documento sejam alcançados. No entanto, no caso dos documentos XMT-A, como a especificação para apresentação dos objetos é, tradicionalmente, temporalmente linear, ajustes não podem ser efetuados.

Devido ao fato do paradigma *timeline* possuir várias limitações, o próprio padrão MPEG-4 definiu o modelo *FlexTime* (ISO/IEC, 2001). Esse modelo permite especificar uma duração flexível, através do estabelecimento de limites mínimos e máximos para a apresentação de um conteúdo. Além dessa funcionalidade, o *FlexTime*, aplicado ao XMT-A, contrasta com o seu modelo temporal tradicional, ao permitir que relacionamentos possam ser efetivamente definidos entre objetos de mídia (Kim & Wood, 2002).

Para alcançar seus objetivos, o *FlexTime* define três tipos de relações temporais, apresentadas a seguir:

- CoStart: O início da execução dos objetos é realizado simultaneamente;
- Meet: O início da execução de um objeto é subseqüente ao término de outro;
- CoEnd: O término da execução dos objetos é realizado simultaneamente.

O modelo *FlexTime* possui uma representação declarativa em XMT-A, através, principalmente, dos elementos *TemporalTransform* e *TemporalGroup*. O primeiro permite especificar uma duração flexível para os objetos de mídia, estabelecendo limites temporais, equivalentes aos definidos no módulo *FlexTime* de XMT-O. O segundo define efetivamente os relacionamentos, através dos atributos booleanos, *coStart*, *coEnd* e *meet*, que podem, individualmente, ter valor booleano verdadeiro (*true*) ou falso (*false*). Os objetos a serem relacionados são definidos internamente a esse elemento, como em uma composição.

A Figura 3.2 apresenta o documento XMT-A obtido a partir da conversão do documento XMT-O apresentado na Figura 3.1, utilizando, desta vez, o módulo *FlexTime*. No documento da Figura 3.2 são especificados quatro elementos do tipo *TemporalTransform*, cada um relativo a um objeto de mídia (dois áudios e duas imagens). Cada elemento *TemporalTransform* define o tempo ideal da apresentação do objeto, através do seu atributo *optimalDuration*, e os tempos máximos e mínimos para apresentação, que podem ser utilizados como limites em operações de ajuste elástico (Bachelet et al., 2000). Para definir os relacionamentos entre esses objetos, um elemento *TemporalGroup* é especificado. Na Figura 3.2, esse elemento possui o atributo *meet* com valor verdadeiro (*true*). Dessa forma, os componentes internos a esse elemento serão apresentados seqüencialmente.

Ainda na Figura 3.2, internamente ao elemento *TemporalGroup*, dois outros elementos do mesmo tipo são especificados. Em cada um, seus atributos *coend* e *costart* possuem valores booleanos verdadeiros, significando que os componentes definidos internamente a eles irão iniciar e terminar simultaneamente suas apresentações. Nesse documento XMT-A, a semântica especificada no documento XMT-O original é preservada, pois os relacionamentos entre os objetos são mantidos no documento obtido na conversão.

```
<XMT-A>
...
<Body>
   <TemporalTransform DEF="Id_01" optimalDuration="30" scalability="20 40">
      <children><Sound2D><source>
            <AudioSource url="od://ODID_1"/>
      </source></Sound2D></children>
   </TemporalTransform>
   <TemporalTransform DEF="Id_02" optimalDuration="30" scalability="20 40">
      <children><Shape>
         <geometry><Bitmap/></geometry>
         <appearance><Appearance>
            <texture><ImageTexture url="od://ODID_2"/></texture>
         </Appearance></appearance>
      </Shape></children>
   </TemporalTransform>
   <TemporalTransform DEF="Id_03" optimalDuration="20" scalability="10 30">
      <children><Sound2D><source>
            <AudioSource url="od://ODID_3"/>
      </source></Sound2D></children>
   </TemporalTransform>
   <TemporalTransform DEF="Id_04" optimalDuration="30" scalability="20 40">
      <children><Shape>
         <geometry><Bitmap/></geometry>
         <appearance><Appearance>
            <texture><ImageTexture url="od://ODID_4"/></texture>
         </Appearance></appearance>
      </Shape></children>
   </TemporalTransform>
   <TemporalGroup DEF="Id_05" coend="false" costart="false" meet="true">
      <children>
         <TemporalGroup DEF="Id_05" coend="true" costart="true" meet="false">
            <children>
               <TemporalTransform USE="Id_01"/>
               <TemporalTransform USE="Id_02"/>
            </children>
         </TemporalGroup>
         <TemporalGroup DEF="Id_05" coend="true" costart="true" meet="false">
            <children>
               <TemporalTransform USE="Id_03"/>
               <TemporalTransform USE="Id_04"/>
            </children>
         </TemporalGroup>
      </children>
   </TemporalGroup>
</Body>
</XMT-A>
```

Figura 3.2 – Documento XMT-A com sincronização através do *FlexTime*

Um grande problema, contudo, vem do fato que, apesar das suas funcionalidades, o modelo *FlexTime* não é obrigatório e, portanto, não é usado na implementação de algumas ferramentas, como o exibidor OSMO4 (*Osmose*) (GPAC, 2002) e o conversor MPEG4Box (GPAC, 2000). A biblioteca dos softwares de referência do MPEG-4 (ISO/IEC, 2000b) possui uma implementação desse módulo, cujo desenvolvimento é creditado ao centro de pesquisas da IBM[11]. No entanto, as ferramentas desenvolvidas nesse centro de pesquisas, voltadas para a tradução de XMT-O para XMT-A, também não utilizam o modelo *FlexTime* (AlphaWorks, 2005). A Figura 3.3 apresenta um documento XMT-O, que especifica uma composição seqüencial contendo dois objetos de vídeo. Esse documento foi convertido para um documento XMT-A, apresentado na Figura 3.4, através da ferramenta XMTBatch, desenvolvida pelo centro de pesquisas da IBM[12]. No documento obtido a semântica da composição XMT-O foi perdida, mantendo-se apenas a seqüencialidade da apresentação.

```
<XMT-O xmlns="urn:mpeg:mpeg4:xmto:schema:2002"
          xsi:schemaLocation="urn:mpeg:mpeg4:xmto:schema:2002 xmt-o.xsd"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xlink="http://www.w3.org/1999/xlink">
 <head>
  <layout type="xmt/xmt-basic-layout" metrics="pixel">
   <topLayout height="120" width="160"/>
  </layout>
 </head>
 <body>
  <seq>
   <video src="../multimedia/FlatPanel160x120.cmp" />
   <video src="../multimedia/FlatPanel160x120.cmp"/>
  </seq>
 </body>
</XMT-O>
```

Figura 3.3 – Documento XMT-O com composição seqüencial

```
<XMT-A>
<Body>
<par begin="0.0">
   <Insert atNode="ROOT_ORDEREDGROUP_1">
     <OrderedGroup DEF="GROUP_1">
       <children><Shape>
          <geometry><Bitmap/></geometry>
          <appearance><Appearance>
            <texture>
              <MovieTexture url="od://ODID_1"/>
            </texture>
```

[11]IBM *Research* (T.J. Watson), http://www.research.ibm.com

[12]*Alphaworks emerging technologies* - IBM *Toolkit* for MPEG-4, http://www.alphaworks.ibm.com/tech/tk4mpeg4

```
            </Appearance></appearance>
          </Shape></children>
      </OrderedGroup>
    </Insert>
  </par>
  <par begin="39.888">
     <Delete atNode="GROUP_1"/>
  </par>
  <par begin="39.888">
    <Insert atNode="ROOT_ORDEREDGROUP_1">
     <OrderedGroup DEF="GROUP_4">
      <children><Shape>
           <geometry><Bitmap/></geometry>
           <appearance><Appearance>
              <texture><MovieTexture url="od://ODID_2"/></texture>
           </Appearance></appearance>
      </Shape></children>
     </OrderedGroup>
    </Insert>
  </par>
  <par begin="79.776">
    <Delete atNode="GROUP_4"/>
  </par>
</Body>
</XMT-A>
```

Figura 3.4 – Documento XMT-A convertido pela ferramenta XMTBatch

Na Figura 3.4, a composição seqüencial definida em XMT-O transforma-se em duas apresentações individuais dos seus componentes, com os intervalos das apresentações calculados pelo conversor. A partir do cálculo dos tempos de apresentação, os objetos são dispostos em composições paralelas. Dessa forma, quatro composições paralelas são instanciadas, a primeira, no início da apresentação, exibe o primeiro vídeo, através de uma referência ao descritor de objetos (*od://ODID_1*). Na segunda, o vídeo inicial é removido, no instante de tempo 39,888 segundos, decorridos do início da apresentação. No mesmo instante de tempo, calculado pelo conversor, a terceira composição é iniciada, exibindo o segundo vídeo, através da referência ao descritor de objetos (*od://ODID_2*). Finalmente, no instante de tempo 79,776 segundos, ao términio do segundo vídeo, sua referência é removida, através do comando *Delete* especificado na última composição paralela.

A Figura 3.5 apresenta os relacionamentos MPEG-4 das duas configurações abordadas para uma apresentação seqüencial de objetos de mídia, a primeira (1) no paradigma usado em XMT-O e a segunda (2) na representação tradicional de XMT-A.

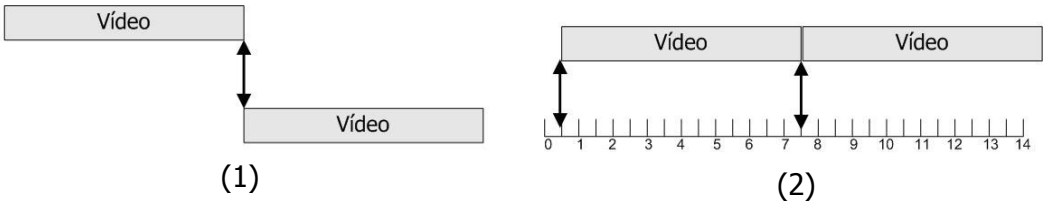

Figura 3.5 – Relacionamentos MPEG-4 em XMT-O e XMT-A

3.4. Instância do *Framework* para Compiladores

As conversões de documentos NCL para XMT-O e vice-versa, e dos documentos XMT-O obtidos para XMT-A são realizadas modularmente. Por exemplo, na conversão inicial (NCL para XMT-O), cada módulo do documento original é analisado e as estruturas equivalentes em XMT-O são geradas. Essa abordagem é proposta em (Silva, 2005), como um *meta-framework* para linguagens XML modulares, a partir do qual os conversores foram gerados.

A implementação do *meta-framework* para linguagens XML modulares (Silva, 2005) foi desenvolvida usando o pacote JAXP (Java API for XML Processing) (JAXP, 2003), biblioteca para processamento de documentos XML. Esse *framework* utiliza o modelo de documentos DOM – *Document Object Model* (W3C, 1998b), padronizado pelo W3C, com o objetivo de manipular a estrutura XML contendo os elementos do documento.

O *meta-framework* favorece a criação de instâncias, *frameworks*, de estruturas de classes voltadas para a compilação específica de uma linguagem, isto é, para a tradução de uma linguagem declarativa para várias outras. A estrutura de classes definidas nos *frameworks* é subjetiva, porém baseia-se na modularização das linguagens, onde, normalmente, uma classe é criada para cada área funcional definida na linguagem de origem. A partir da estrutura das classes, seus métodos, abstratos e concretos, são definidos. Os métodos concretos têm como objetivo realizar as tarefas genéricas envolvidas na conversão, enquanto os métodos abstratos devem ser instanciados de acordo com a linguagem destino da conversão.

Na conversão de NCL para XMT-O, a especificação do *framework*, com os métodos concretos para selecionar os módulos existentes em documentos NCL já se encontrava definida, sendo utilizada em trabalhos relacionados, por exemplo, na conversão de NCL para SMIL (Silva, 2005). Para conversão de NCL para

XMT-O, teve-se, no entanto, de instanciar as classes relativas aos métodos abstratos, a fim de definir as tarefas envolvidas na tradução.

Para a conversão de XMT-O para XMT-A, foi desenvolvido um *framework* para a linguagem XMT-O, permitindo a conversão dessa linguagem para várias outras. Como instâncias específicas, foram desenvolvidos os conversores para NCL e XMT-A, através da instanciação de classes relativas aos seus métodos abstratos. A Figura 3.6 apresenta o diagrama de classes da instância do *framework* para compiladores NCL aplicado à tradução para XMT-O. As classes principais são: *NclXmtoComponentsParser*, *NclXmtoLayoutParser*, *NclXmtoLinkingParser*, *NclXmtoConnectorParser*, *NclXmtoPresentationSpecificationParser*, *NclXmtoInterfaceParser* e *NclXmtoPresentationControlParser*. As áreas funcionais *Timing* e *Structure* de NCL, por incluírem poucos elementos tiveram suas funções absorvidas por outras classes. A área funcional *Metainformation* não é definida nessa versão do *framework*.

As classes principais apresentadas na Figura 3.6 são herdadas das classes originais do *framework* NCL, correspondentes às seguintes classes: *NclComponentsParser*, *NclLayoutParser*, *NclLinkingParser*, *NclConnectorParser*, *NclPresentationSpecificationParser*, *NclInterfaceParser* e *NclPresentationControlParser*. Além dessas classes, uma outra, chamada *NclXmtoDocumentCompiler*, é definida como a estrutura básica de gerência do compilador, centralizando as operações. Através do seu método *parse*, a conversão entre as linguagens é iniciada, utilizando a estrutura DOM do documento, onde o elemento raiz é passado para o método *parseRootElement*. Nessa classe, a estrutura do documento é analisada, através dos métodos definidos na classe *NclStructureParser*. Para cada parte específica do documento, a classe relativa à área funcional dessa parte é instanciada, por exemplo, no *layout* a análise da estrutura é responsabilidade da classe *NclXmtoLayoutParser*. Ao fim da conversão, a classe *NclXmtoDocumentCompiler*, através do seu método *serialize*, converte a estrutura DOM, obtida na transformação, para um arquivo declarativo.

As classes principais apresentadas na Figura 3.6 definem, além dos seus construtores, necessários para iniciar parâmetros gerais (como valores de variáveis, instâncias de listas e vetores etc.), uma série de métodos. As funções desses métodos podem, geralmente, ser identificadas de acordo com sua nomenclatura, definida na especificação do *meta-framework* (Silva, 2005).

Os métodos iniciados com o nome *parse* sempre retornam objetos no modelo da linguagem de destino. Os métodos iniciados com o nome *create* instanciam os objetos que representam o elemento declarativo na linguagem destino. Normalmente, os métodos nomeados como *create* criam os objetos nas instâncias finais dos elementos declarativos. É comum um método *parse*, concreto, acionar vários métodos *create*, criando vários filhos no elemento declarativo e, por fim, retornando um elemento composto por vários outros. Finalmente, os métodos iniciados com o nome *add* são responsáveis por adicionar objetos aos objetos pai do modelo de destino. Esses métodos são sempre chamados pelos métodos concretos do tipo *parse*, definidos nas classes herdadas do *framework*. Na Figura 3.6 algumas associações diretas entre classes, à exceção daquelas que contam com a participação de *NclXmtoDocumentCompiler*, foram omitidas, a fim de facilitar a compreensão do diagrama de classes e, conseqüentemente, do processo de tradução.

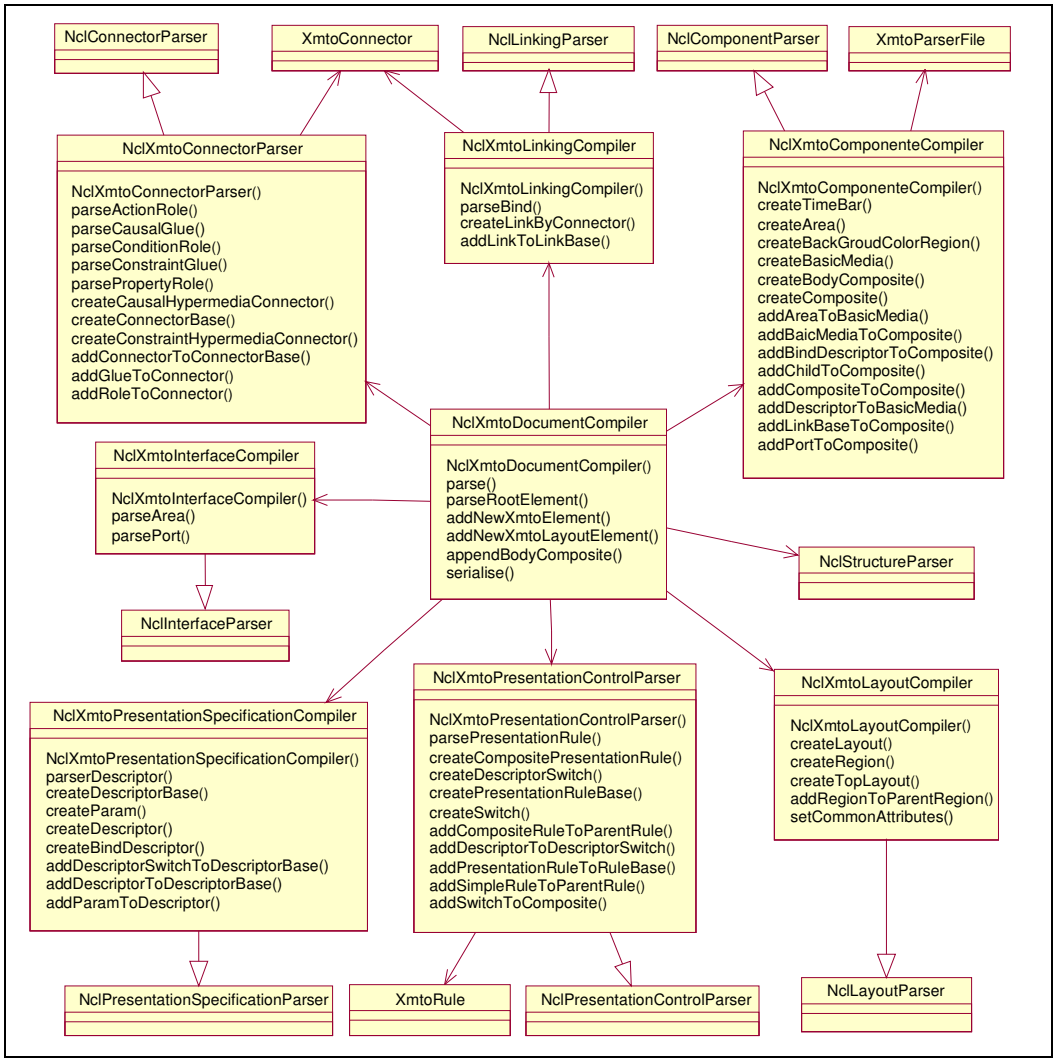

Figura 3.6 – Diagrama de classes do conversor NCL para XMT-O

Na Figura 3.6, além das classes e dos métodos citados, outros são empregados com o objetivo de favorecer a tradução entre as linguagens. Como exemplo, pode-se citar a classe *XmtoParserFile*, empregada na tradução de objetos de mídia textuais para objetos sintéticos XMT-O. Outras classes com funções similares são as classes *XmtoRule* e *XmtoConnector*, responsáveis, respectivamente pela tradução das regras para controle da apresentação e pela tradução da semântica dos conectores.

O diagrama de classes do *framework* para compiladores XMT-O é apresentado nas Figuras 3.7 e 3.8. A primeira define o diagrama da conversão para NCL e a segunda da conversão para XMT-A. As classes principais representam as seguintes áreas funcionais de XMT-O: *Timing* e *TimeManipulations* (*XmtoNclTimingCompiler, XmtoXmtaTimingCompiler)*; *Animation* (*XmtoNclAnimationParser, XmtoXmtaAnimationParser*); *ContentControl* (*XmtoNclContentControlCompiler, XmtoXmtaContentControlCompiler*); *Layout* (*XmtoNclLayoutCompiler, XmtoXmtaLayoutCompiler*); *Linking* (*XmtoNclLinkingCompiler*, *XmtoXmtaLinkingCompiler*); *MediaObjects* (*XmtoNclMediaObjectCompiler*, *XmtoXmtaMediaObjectCompiler*); *Transitions* (*XmtoNclTransitionsCompiler*, *XmtoXmtaTransitionsCompiler*); *Metainformation* (*XmtoNclMetaInformationCompiler*, *XmtoXmtaMetaInformationCompiler*); e *DEFS* (*XmtoNclDefinitionCompiler*, *XmtoXmtaDefinitionCompiler*).

A área funcional *Structure* tem suas funções especificadas pela classe *XmtoStructure* definida no *framework*. A área funcional *Macros*, como já anteriormente mencionado, deve ser compilada antes de qualquer conversão para outras linguagens.

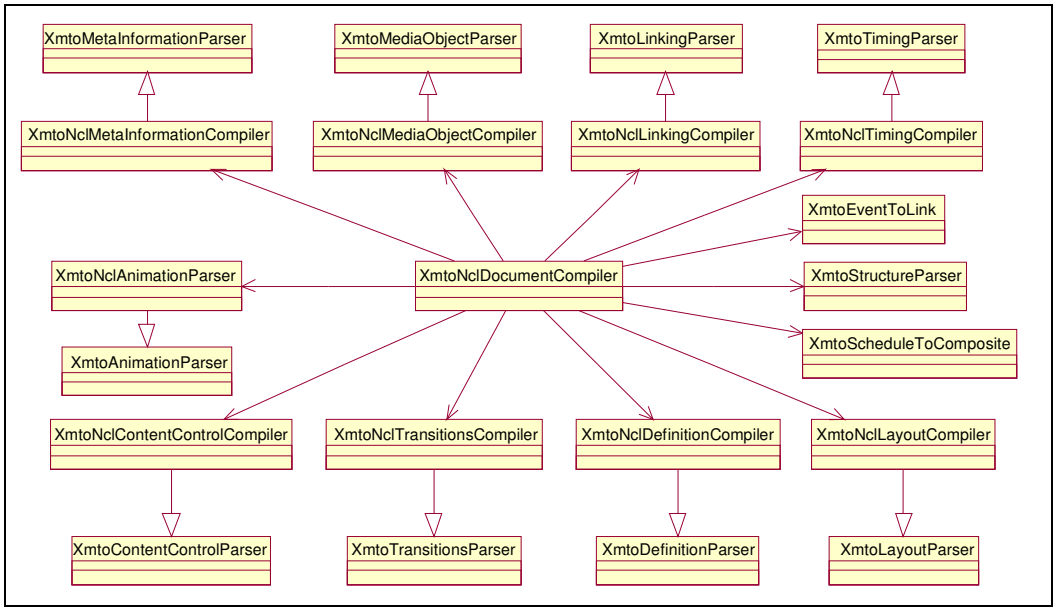

Figura 3.7 – Diagrama de classes do conversor XMT-O para NCL

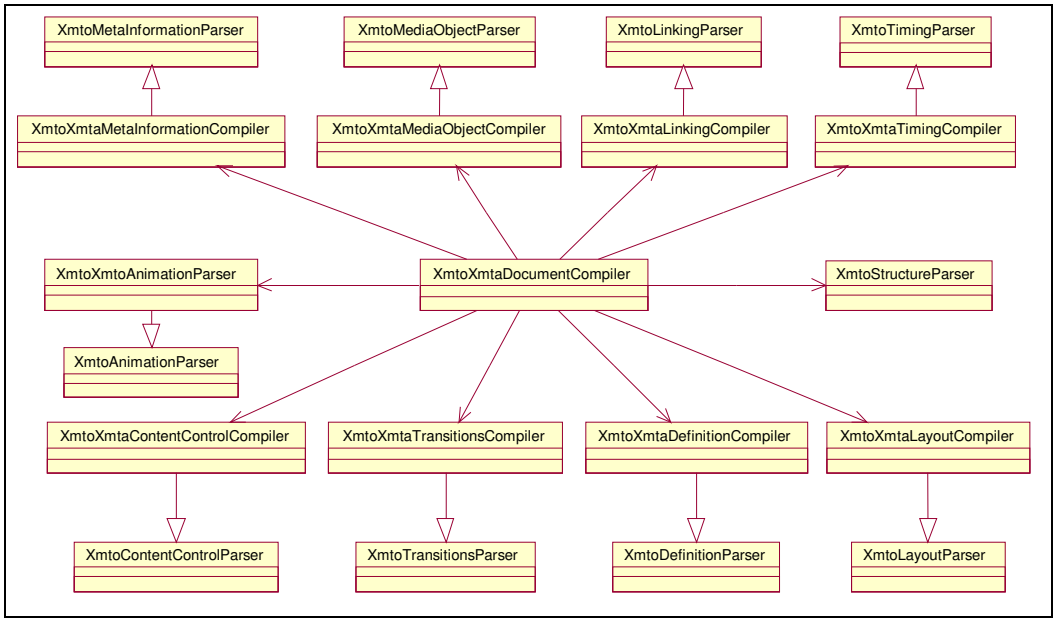

Figura 3.8 – Diagrama de classes do conversor XMT-O para XMT-A

3.4.1. Exemplo do Uso dos Compiladores

Os compiladores desenvolvidos para a conversão entre os formatos NCL e MPEG-4 podem ser utilizados tanto individualmente quanto em conjunto. Em (Costa et al., 2004) é apresentada uma arquitetura para emprego dos compiladores.

Nessa arquitetura, apresentada na Figura 3.9, o documento NCL é convertido para MPEG-4 em camadas. Inicialmente, na camada superior, o

documento é processado e representado em XMT-O. Na segunda camada, intermediária, o documento XMT-O, anteriormente obtido, é convertido para XMT-A. Finalmente, na camada inferior, a partir da descrição em XMT-A, é possível realizar um mapeamento direto para o formato BIFS. Nessa fase, os diversos objetos de mídia, referenciados no documento NCL original, também são convertidos em fluxos e agregados em um único arquivo.

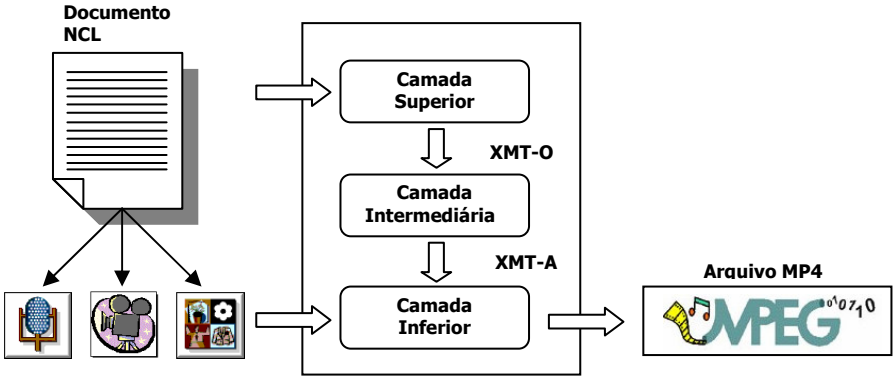

Figura 3.9 – Arquitetura em camadas para o conversor MPEG-4

Para a implementação da camada inferior, pode-se usar o pacote de software oferecido pelo GPAC (*GPAC Project on Advanced Content*)[13] devido às suas funcionalidades e também por ser licenciado como código aberto. GPAC é um subconjunto de implementações dos softwares de referência do padrão MPEG-4. Dentro desse subconjunto existe um codificador capaz de converter documentos especificados em XMT-A para descrições em BIFS. Além disso, ele também é capaz de converter os objetos de mídia em fluxos MPEG-4 e de realizar a multiplexação de todos os fluxos em um único arquivo.

A gerência das três camadas citadas é realizada por uma interface de controle da troca de informações e conteúdo entre os compiladores Java (NCL para XMT-O e XMT-O para XMT-A) e o codificador MP4Box, do GPAC, disponibilizado através de um arquivo executável, compilado previamente a partir do seu código fonte, escrito em C.

Para ilustrar o uso dos compiladores, considere um documento NCL onde, no cabeçalho, um leiaute é especificado definindo duas janelas com várias regiões. Ainda no cabeçalho, as informações para apresentação são especificadas através de descritores. Nesse documento, uma composição, identificada por “coisapele”, contém um nó de áudio com uma música. Nessa mesma composição, uma série de

[13] http://gpac.sourceforge.net

arquivos texto contém partes da letra dessa música. Esses arquivos são sincronizados com os respectivos intervalos do áudio, através de elos. A Figura 3.10 apresenta esse documento NCL.

```
<?xml version="1.0" encoding="UTF-8"?>
<ncl id="ncl_example-processed" xmlns="http://www.telemidia.puc-rio.br/specs/xml/NCL"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.telemidia.puc-rio.br/specs/xml/NCL
http://www.telemidia.puc-rio.br/specs/xml/NCL.xsd">
  <head>
   <layout>
   <topLayout id="window1" title="Music Lyrics" height="300" width="600" >
      <region id="title" height="20%" width="100%" />
      <region id="textRegion1" top="20%" height="80%" width="65%" />
      <region id="imageRegion1" top="20%" left="65%" height="80%" width="35%" />
    </topLayout>
 <topLayout id="audioWindow" title="Audio Control" top="300" height="50" width="600">
      <region height="100%" id="audioRegion1" width="100%" />
    </topLayout>
   </layout>
   <descriptorBase>
    <descriptor dur="94s" enableTimeBar="on" id="audio_d1" region="audioRegion1"/>
    <descriptor id="text_d0" region="title"/>
    <descriptor id="text_d1" region="textRegion1"/>
    <descriptor id="img_d1" region="imageRegion1"/>
   </descriptorBase>
 </head>
 <body>
  <port id="entrada1" component="coisaPele" port="musica"/>
  <composite id="coisaPele">
   <port id="musica" component="samba"/>
   <audio descriptor="audio_d1" id="samba" src=" coisa.mp3">
    <area id="part1" begin="8.4s" end="18s"/>
    <area id="part2" begin="18.5s" end="28s"/>
    <area id="part3" begin="29s" end="39s"/>
    <area id="part4" begin="39.5s" end="50s"/>
    <area id="part5" begin="50.5s" end="71.4s"/>
    <area id="part6" begin="72s" end="94s"/>
   </audio>
   <text descriptor="text_d0" id="title-coisaPele" src=" titulo01.txt"/>
   <text descriptor="text_d1" id="lyrics-part1" src="versos01.txt"/>
   <text descriptor="text_d1" id="lyrics-part2" src="versos02.txt"/>
   <text descriptor="text_d1" id="lyrics-part3" src="versos03.txt"/>
   <text descriptor="text_d1" id="lyrics-part4" src="versos04.txt"/>
   <text descriptor="text_d1" id="lyrics-part5" src="versos05.txt"/>
   <text descriptor="text_d1" id="lyrics-part6" src="versos06.txt"/>
   <img descriptor="img_d1" id="logotele1" src="logo.jpg"/>

   <linkBase>
    <link id="link1" xconnector="file:../mediaContent/connectors/starts.xml">
     <bind component="samba" role="on_x_presentation_begin"/>
     <bind component="logotele1" role="start_y"/>
    </link>
    <link id="link2" xconnector="file:../mediaContent/connectors/finishes.xml">
     <bind component="samba" role="on_x_presentation_end"/>
     <bind component="logotele1" role="stop_y"/>
    </link>
    <link id="link3" xconnector="file:../mediaContent/connectors/starts.xml">
     <bind component="samba" port="part1" role="on_x_presentation_begin"/>
```

```
        <bind component="lyrics-part1" role="start_y"/>
       </link>
       <link id="link4" xconnector="file:../mediaContent/connectors/finishes.xml">
        <bind component="samba" port="part1" role="on_x_presentation_end"/>
        <bind component="lyrics-part1" role="stop_y"/>
       </link>
       <link id="link5" xconnector="file:../mediaContent/connectors/starts.xml">
        <bind component="samba" port="part2" role="on_x_presentation_begin"/>
        <bind component="lyrics-part2" role="start_y"/>
       </link>
       <link id="link6" xconnector="file:../mediaContent/connectors/finishes.xml">
        <bind component="samba" port="part2" role="on_x_presentation_end"/>
        <bind component="lyrics-part2" role="stop_y"/>
       </link>
       <link id="link7" xconnector="file:../mediaContent/connectors/starts.xml">
        <bind component="samba" port="part3" role="on_x_presentation_begin"/>
        <bind component="lyrics-part3" role="start_y"/>
       </link>
       <link id="link8" xconnector="file:../mediaContent/connectors/finishes.xml">
        <bind component="samba" port="part3" role="on_x_presentation_end"/>
        <bind component="lyrics-part3" role="stop_y"/>
       </link>
       <link id="link9" xconnector="file:../mediaContent/connectors/starts.xml">
        <bind component="samba" port="part4" role="on_x_presentation_begin"/>
        <bind component="lyrics-part4" role="start_y"/>
       </link>
       <link id="link10" xconnector="file:../mediaContent/connectors/finishes.xml">
        <bind component="samba" port="part4" role="on_x_presentation_end"/>
        <bind component="lyrics-part4" role="stop_y"/>
       </link>
       <link id="link11" xconnector="file:../mediaContent/connectors/starts.xml">
        <bind component="samba" port="part5" role="on_x_presentation_begin"/>
        <bind component="lyrics-part5" role="start_y"/>
       </link>
       <link id="link12" xconnector="file:../mediaContent/connectors/finishes.xml">
        <bind component="samba" port="part5" role="on_x_presentation_end"/>
        <bind component="lyrics-part5" role="stop_y"/>
       </link>
       <link id="link13" xconnector="file:../mediaContent/connectors/starts.xml">
        <bind component="samba" port="part6" role="on_x_presentation_begin"/>
        <bind component="lyrics-part6" role="start_y"/>
       </link>
       <link id="link14" xconnector="file:../mediaContent/connectors/finishes.xml">
        <bind component="samba" port="part6" role="on_x_presentation_end"/>
        <bind component="lyrics-part6" role="stop_y"/>
       </link>
       <link id="link15" xconnector="file:../mediaContent/connectors/starts.xml">
        <bind component="samba" role="on_x_presentation_begin"/>
        <bind component="title-coisaPele" role="start_y"/>
       </link>
       <link id="link16" xconnector="file:../mediaContent/connectors/finishes.xml">
        <bind component="samba" port="part6" role="on_x_presentation_end"/>
        <bind component="title-coisaPele" role="stop_y"/>
       </link>
      </linkBase>
     </composite>
    </body>
   </ncl>
```

Figura 3.10 – Documento NCL “coisa de pele”

A partir do documento NCL da Figura 3.10, utilizando os compiladores de NCL para MPEG-4, apresentações audiovisuais podem ser exibidas diretamente em *players* MPEG-4, com aspecto visual e temporal similares aos obtidos caso o documento NCL fosse apresentado no formatador do sistema HyperProp. A Figura 3.11 apresenta o documento XMT-O obtido através da conversão do documento NCL apresentado na Figura 3.10.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<XMT-O xmlns="urn:mpeg:mpeg4:xmto:schema:2002"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xmto="urn:mpeg:mpeg4:xmto:schema:2002"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:mpeg:mpeg4:xmto:schema:2002 xmt-o.xsd">
<head>
 <layout metrics="pixel" type="xmt/xmt-basic-layout">
  <topLayout backgroundColor="white" height="350" id="window1" width="600">
   <region id="title" size="600 60" translation="0 145"/>
   <region id="textRegion1" size="390 240" translation="-105 -5"/>
   <region id="imageRegion1" size="210 240" translation="195 -5"/>
   <region id="audioRegion1" size="600 50" translation="0 -150"/>
  </topLayout>
 </layout>
</head>
<body>
 <par id="ncl_example-processed">
  <rectangle dur="indefinite" region="imageRegion1" size="210 240">
   <material color="white" filled="true"/>
  </rectangle>
  <rectangle dur="indefinite" region="audioRegion1" size="600 50">
   <material color="white" filled="true"/>
  </rectangle>
  <rectangle dur="indefinite" region="title" size="600 60">
   <material color="white" filled="true"/>
  </rectangle>
  <rectangle dur="indefinite" region="textRegion1" size="390 240">
   <material color="white" filled="true"/>
  </rectangle>
  <par id="coisaPele">
   <par id="ncl-composite-1">
    <audio dur="94s" id="samba" region="audioRegion1" src="coisa.mp3" />
    <rectangle begin="samba.begin+8.4s" end="samba.begin+18s" id="part1" />
    <rectangle begin="samba.begin+18.5s" end="samba.begin+28s" id="part2" />
    <rectangle begin="samba.begin+29s" end="samba.begin+39s" id="part3" />
    <rectangle begin="samba.begin+39.5s" end="samba.begin+50s" id="part4" />
    <rectangle begin="samba.begin+50.5s" end="samba.begin+71.4s" id="part5" />
    <rectangle begin="samba.begin+72s" end="samba.begin+94s" id="part6" />
    <rectangle begin="samba.begin" end="samba.end" region="audioRegion1" size="600
        50" >
     <material color="#C0C0C0" filled="true"/>
    </rectangle>
    <lines begin="samba.begin" color="#808080;#808080" colorPerVertex="false" coord="-
        300 -24;300 -24" end="samba.end" region="audioRegion1"/>
    <lines begin="samba.begin" color="#808080;#808080" colorPerVertex="false"
        coord="299 -24;299 24" end="samba.end" region="audioRegion1"/>
    <lines begin="samba.begin" color="#DDDCDC;#DDDCDC" colorPerVertex="false"
        coord="-300 24;300 24" end="samba.end" region="audioRegion1"/>
    <lines begin="samba.begin" color="#DDDCDC;#DDDCDC" colorPerVertex="false"
```

```
        coord="-300 24;-300 -24" end="samba.end" region="audioRegion1"/>
    <rectangle begin="samba.begin" end="samba.end" region="audioRegion1" size="540
        2">
     <material color="#000000" filled="true"/>
    </rectangle>
    <group begin="samba.begin" end="samba.end" id="groupnclprocessedsamba"
        region="audioRegion1" visibility="true">
     <animateMotion begin="groupnclprocessedsamba.begin" calcMode="linear" dur="94s"
         to="540 0"/>
     <rectangle size="16 16">
      <material color="#C0C0C0" filled="true"/>
      <transformation translation="-270 0"/>
     </rectangle>
     <lines begin="groupnclprocessedsamba.begin" color="#808080;#808080"
      colorPerVertex="false" coord="-278 -8;-262 -8" end="groupnclprocessedsamba.end"/>
     <lines begin="groupnclprocessedsamba.begin" color="#808080;#808080"
      colorPerVertex="false" coord="-262 -8;-262 8" end="groupnclprocessedsamba.end"/>
     <lines begin="groupnclprocessedsamba.begin" color="#DDDCDC;#DDDCDC"
      colorPerVertex="false" coord="-278 8;-262 8" end="groupnclprocessedsamba.end"/>
     <lines begin="groupnclprocessedsamba.begin" color="#DDDCDC;#DDDCDC"
      colorPerVertex="false" coord="-278 8;-278 -8" end="groupnclprocessedsamba.end"/>
    </group>
   </par>
   <rectangle begin="samba.begin" end="samba.end" id="logotele1" region="imageRegion1"
      size="210 240">
    <texture src="logo.jpg"/>
   </rectangle>
   <string begin="samba.begin" end="part6.end" id="title-coisaPele" region="title"
       textLines="&amp;quot;Coisa de Pele&amp;quot;;&amp;quot;(Jorge Aragão - Ivone Lara)&amp;quot;">
    <material color="black"/>
    <fontStyle family="&amp;quot;TYPEWRITER&amp;quot;" justify="MIDDLE;MIDDLE"
       size="15" style="PLAIN"/>
   </string>
   <string begin="part1.begin" end="part1.end" id="lyrics-part1" region="textRegion1"
       textLines="&amp;quot;Podemos sorrir, nada mais nos impede&amp;quot;;&amp;quot;Não dá para
       fugir dessa coisa de pele&amp;quot;">
    <material color="black"/>
    <fontStyle family="&amp;quot;TYPEWRITER&amp;quot;" justify="MIDDLE;MIDDLE"
       size="15" style="PLAIN"/>
   </string>
   <string begin="part2.begin" end="part2.end" id="lyrics-part2" region="textRegion1"
       textLines="&amp;quot;Sentida por nós, desatando os nós&amp;quot;;&amp;quot;Sabemos agora,
       nem tudo que é bom vem &amp;quot;;&amp;quot;de fora&amp;quot;">
    <material color="black"/>
    <fontStyle family="&amp;quot;TYPEWRITER&amp;quot;" justify="MIDDLE;MIDDLE"
       size="15" style="PLAIN"/>
   </string>
   <string begin="part3.begin" end="part3.end" id="lyrics-part3" region="textRegion1"
       textLines="&amp;quot;É a nossa canção, pelas ruas e bares&amp;quot;;&amp;quot;Que nos traz a
       razão, relembrando &amp;quot;;&amp;quot;palmares&amp;quot;">
    <material color="black"/>
    <fontStyle family="&amp;quot;TYPEWRITER&amp;quot;" justify="MIDDLE;MIDDLE"
       size="15" style="PLAIN"/>
   </string>
   <string begin="part4.begin" end="part4.end" id="lyrics-part4" region="textRegion1"
       textLines="&amp;quot;Foi bom insistir, compor e ouvir&amp;quot;;&amp;quot;Resiste quem pode,
       à força dos nossos &amp;quot;;&amp;quot;pagodes&amp;quot;">
    <material color="black"/>
    <fontStyle family="&amp;quot;TYPEWRITER&amp;quot;" justify="MIDDLE;MIDDLE"
       size="15" style="PLAIN"/>
```

```
    </string>
    <string begin="part5.begin" end="part5.end" id="lyrics-part5" region="textRegion1"
      textLines="&amp;quot;E o samba se faz prisioneiro pacato &amp;quot;;&amp;quot;dos nossos
      tantans&amp;quot;;&amp;quot;E um banjo liberta da garganta do povo &amp;quot;;&amp;quot;as suas
      emoções&amp;quot;;&amp;quot;Alimentando muito mais a cabeça de um
      &amp;quot;;&amp;quot;compositor&amp;quot;;&amp;quot;Eterno reduto de paz, nascente das
      &amp;quot;;&amp;quot;várias feições do amor&amp;quot;">
     <material color="black"/>
     <fontStyle family="&amp;quot;TYPEWRITER&amp;quot;" justify="MIDDLE;MIDDLE"
      size="15" style="PLAIN"/>
    </string>
    <string begin="part6.begin" end="part6.end" id="lyrics-part6" region="textRegion1"
      textLines="&amp;quot;Arte popular do nosso chão&amp;quot;;&amp;quot;É o povo quem produz o
      show e assina a &amp;quot;;&amp;quot;direção&amp;quot;">
     <material color="black"/>
     <fontStyle family="&amp;quot;TYPEWRITER&amp;quot;" justify="MIDDLE;MIDDLE"
      size="15" style="PLAIN"/>
    </string>
   </par>
  </par>
 </body>
</XMT-O>
```

Figura 3.11 - Documento XMT-O “coisa de pele”

As Figuras 3.12 e 3.14 exibem visões, em dois instantes de tempo (note a barra de rolagem do áudio), do documento MPEG-4 sendo apresentado.

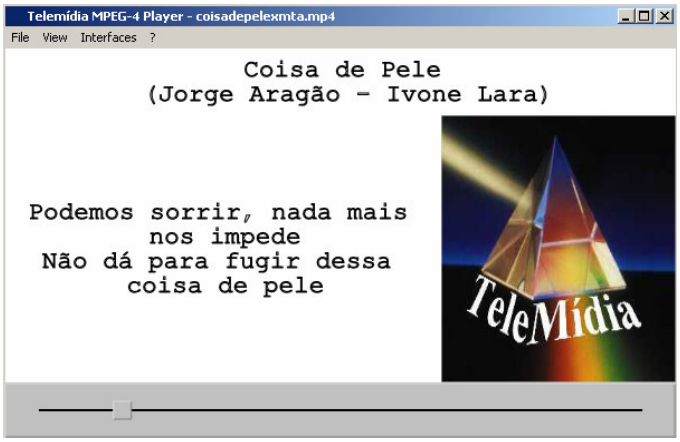

Figura 3.12 – Primeira visão da apresentação MPEG-4 no exibidor

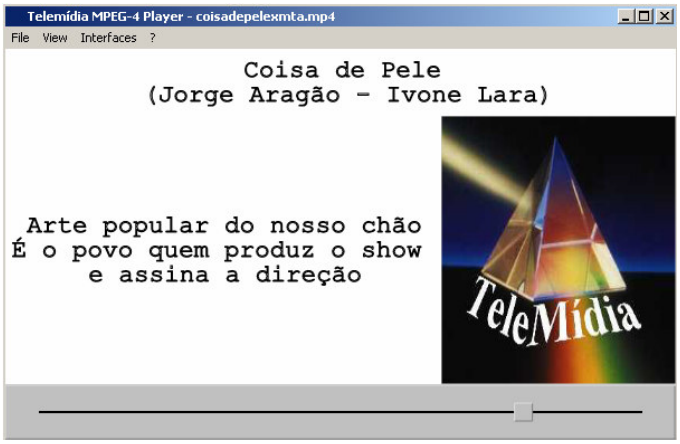

Figura 3.13 – Segunda visão da apresentação MPEG-4 no exibidor

4
Componentes MPEG-4 em Documentos NCL

O padrão MPEG-4 é semelhante aos padrões MPEG anteriores, embora possua características peculiares, como a abordagem orientada a objeto, que o aproxima da modelagem de aplicações multimídia/hipermídia.

A fim de explorar como os componentes MPEG-4 podem se relacionar entre si, estando em diferentes cenas, e também com outros objetos, externos às cenas audiovisuais MPEG, este capítulo discute a definição desses componentes como nós de documentos multimídia/hipermídia especificados na linguagem NCL.

O capítulo está organizado da forma a seguir. A Seção 4.1 aborda os aspectos necessários à definição de componentes MPEG-4 em documentos hipermídia especificados em NCL. A seguir, na Seção 4.2 é proposta a representação desses componentes como objetos NCL de mídia contínua, similares aos obtidos pela codificação audiovisual utilizando os padrões MPEG-1 e MPEG-2. Por fim, a Seção 4.3 propõe que os componentes MPEG-4 sejam tratados como composições NCL.

4.1.
Definição de Componentes MPEG-4 em Documentos NCL

Conforme definido no Capítulo 2, as apresentações MPEG-4 são formadas por cenas audiovisuais, onde cada cena corresponde a uma composição que pode conter vários objetos, bem como os relacionamentos entre eles. Dentro de cada cena não existem restrições para a definição de relacionamentos entre os seus objetos internos, no entanto, os relacionamentos entre cenas distintas são limitados.

Nos relacionamentos estabelecidos entre diferentes cenas a relação semântica é, normalmente, de referência: uma âncora é associada a um dos objetos da cena de origem, sendo acionada pela ação do clique do mouse. No entanto, algumas extensões XLink (W3C, 2001) podem ser utilizadas a fim de

estender a semântica desses relacionamentos, definindo relações com semântica de sincronização (ISO/IEC, 2001).

A limitação dos relacionamentos definidos entre cenas distintas não se deve somente a restrições existentes nas relações que podem ser estabelecidas, mas também ao fato dessas cenas serem apresentadas isoladamente nos exibidores MPEG-4. Em cada exibidor, durante um intervalo específico de tempo, somente uma cena pode ser apresentada. Dessa forma, quando relacionamentos entre cenas distintas são executados, a cena de origem tem que ser interrompida, para o início da cena de destino.

O modelo de referência dos exibidores MPEG-4 é definido na Parte 5 desse padrão (ISO/IEC, 2000b). Nesse modelo, as cenas audiovisuais são compostas por objetos de diferentes tipos, relacionados entre si, devendo o próprio exibidor MPEG-4 implementar todos os aspectos relacionados à apresentação desses objetos, como a descompressão, decodificação, controle da taxa de apresentação, entre outros, além de controlar, também, os relacionamentos estabelecidos entre eles.

Em outros modelos, mais genéricos, como o proposto pelo sistema HyperProp, o formatador (exibidor) dispõe de um conjunto de ferramentas de exibição a fim de tratar os diferentes tipos de objetos de mídia. Nesse modelo, é responsabilidade de cada ferramenta gerenciar os aspectos relacionados às características específicas dos tipos de objetos (padrões de codificação). Ao orquestrador (outro módulo do formatador) cabe a responsabilidade pelo controle da apresentação, preservando os relacionamentos definidos pelos autores (Rodrigues, 2003).

É importante notar que, inseridos no modelo proposto pelo sistema HyperProp, os exibidores MPEG-4 tornam-se apenas mais uma ferramenta de exibição, especializadas em componentes MPEG-4. Uma vez inseridos nesse sistema, esses exibidores favorecem a especificação de documentos NCL contendo componentes representando as cenas MPEG-4. Na realidade, a linguagem NCL não define restrições para os padrões de codificação adotados pelos objetos de mídia, no entanto cabe ao ambiente de execução, responsável pela apresentação dos documentos, oferecer aplicações capazes de exibir os objetos dos diferentes tipos de mídia, garantindo que os relacionamentos, concebidos na autoria, sejam obedecidos.

A fim de integrar os exibidores MPEG-4 ao sistema HyperProp, como parte do conjunto de ferramentas de exibição desse sistema, adaptações devem ser implementadas nesses exibidores, para que eles sigam o modelo de comunicação com o formatador, apresentado em (Rodrigues, 2003). Nesse modelo, uma interface padronizada para a troca de mensagens entre essas entidades é definida. Além dessas adaptações, os exibidores MPEG-4 devem ainda monitorar os eventos, que podem ser estabelecidos sobre os seus componentes.

Um evento especifica uma ocorrência no tempo resultante de alguma ação sobre um atributo ou sobre uma região do conteúdo (âncora) de um objeto de representação[27] (Rodrigues et al., 2001). Como exemplo de um evento, considere a seleção de uma âncora pelo usuário (evento de seleção). Essa seleção pode corresponder à ação do clique do mouse sobre uma região da apresentação de um vídeo, por exemplo. Caso a ferramenta de exibição, destinada a exibir esse vídeo, não monitore esse evento, relacionamentos causais, disparados por ele, ainda que especificados no documento multimídia/hipermídia, não serão considerados durante a apresentação.

Os eventos que podem ser estabelecidos sobre os componentes MPEG-4 variam de acordo com a forma de representação desses componentes. Em um dos cenários possíveis, esses eventos correspondem aos mesmos que podem ser definidos sobre os objetos NCL de mídia contínua (Rodrigues, 2003). Em outro cenário, onde os componentes MPEG-4 são definidos como composições NCL, esses eventos correspondem aos ocorridos internamente nessas composições, devendo ser exportados através do mapeamento dessas ocorrências internas para as portas da composição.

No primeiro cenário, os eventos podem estar associados a âncoras espaciais sobre a região de apresentação do componente MPEG-4. Nesse caso compete à ferramenta identificar a região estabelecida e controlar as ações nessa área. Ainda nesse cenário, âncoras temporais, relativas ao tempo da apresentação do componente podem ser definidas, devendo a ferramenta monitorar o tempo decorrido sobre a apresentação do componente.

---

[27] No modelo NCM, o objeto de representação corresponde ao componente apresentado, agregando os atributos do objeto e do seu descritor.

No segundo cenário, os componentes são apresentados como composições NCL, cuja estrutura semântica é estabelecida pelos comandos BIFS, contidos no fluxo descritor de cenas do componente em questão. Nesse cenário, além dos eventos apresentados no cenário anterior, outros podem ser estabelecidos, através do mapeamento dos eventos definidos internamente a essas composições.

Para exemplificar os diferentes cenários, considere uma apresentação audiovisual, onde os relacionamentos são definidos com a participação de um componente MPEG-4. Esse componente, apresentado na Figura 4.1, é formado por três objetos principais (1): o plano de fundo, correspondente a um objeto de imagem (VOP1); um objeto de vídeo (VOP2); e outro objeto de imagem (VOP3). Esses três componentes, definem planos de objeto de vídeo individuais (VOP - *video object plane*) que são sobrepostos para formar o componente MPEG-4 (2).

Suponha que quando o usuário selecionar com o mouse o objeto de vídeo VOP2, pertencente ao componente MPEG-4, um objeto de imagem, relativo a uma fotografia sobre o desmatamento, deva ser exibido e, quando o objeto de imagem VOP3, também pertencente a esse componente, for selecionado, um objeto de vídeo, contendo informações sobre a Amazônia, deva ser exibido.

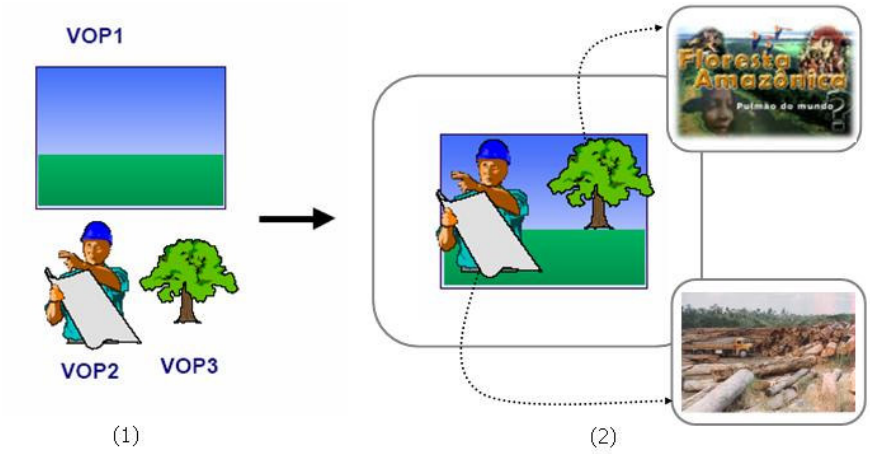

Figura 4.1 – Apresentação MPEG-4 com sincronização espacial e temporal

No cenário inicial, em relação ao exemplo apresentado, o componente MPEG-4 é tratado como um objeto NCL de vídeo. Nesse cenário, a fim de alcançar a concepção desejada, um documento NCL é especificado onde, além do componente MPEG-4, dois outros objetos são definidos, um objeto NCL de imagem, relativo à fotografia sobre o desmatamento, e um objeto NCL de vídeo, contendo informações sobre a Amazônia. Nesse documento, duas âncoras são definidas sobre o componente MPEG-4, cada uma sobre as áreas de exibição dos objetos de vídeo VOP2 e de imagem VOP3. As relações nesse documento são

definidas através de um único conector NCL com semântica causal. Esse conector estabelece como condição o evento de clique do mouse, e como ação a transição de início do evento de apresentação. Dois relacionamentos distintos são estabelecidos referenciando esse conector. No primeiro, um elo associa o componente MPEG-4, na sua âncora definida sobre VOP2, ao objeto NCL de imagem. No segundo, a associação é definida por um elo entre a âncora definida sobre VOP3, no componente MPEG-4, e o objeto NCL de vídeo.

É importante ressaltar que nesse primeiro cenário, os relacionamentos NCL são definidos sobre as âncoras estabelecidas no componente MPEG-4, como mencionado, áreas espaciais cujo posicionamento varia no tempo, e não diretamente sobre os objetos que compõem esse componente. Portanto, algumas dificuldades podem ocorrer na definição desses relacionamentos, como, no exemplo, a definição de âncoras como função do tempo.

No segundo cenário, os eventos são definidos internamente ao componente MPEG-4, usando as facilidades definidas pelo padrão. Na Figura 4.1, os eventos de clique do mouse podem ser diretamente especificados sobre os objetos de vídeo VOP2 e de imagem VOP3, através de comandos BIFS. Nesse cenário, cabe à ferramenta de exibição capturar esses eventos, quando disparados, e ativar interfaces atribuídas ao componente MPEG-4. Essas interfaces, por sua vez, podem estar relacionadas a outros componentes MPEG-4, composições ou objetos de diversos tipos, definidos no documento NCL. Portanto, nesse segundo cenário, ao contrário do primeiro, os elos encontram-se associados, através do mapeamento de interfaces, diretamente aos objetos que compõem o componente MPEG-4. O componente MPEG-4 é tratado como uma composição NCL, com cada âncora interna associada a uma de suas portas.

Voltando ao exemplo da Figura 4.1, aplicado agora a esse segundo cenário, a especificação do documento NCL, destinado a representar a concepção desejada, contém dois relacionamentos. No primeiro, um elo NCL associa a interface do componente (composição) MPEG-4, mapeada ao evento do clique do mouse sobre o objeto de imagem VOP3. No segundo, através de outro elo NCL, uma associação é definida entre a interface do componente (composição) MPEG-4, relativa ao evento do clique do mouse sobre o objeto de vídeo VOP2, e o objeto NCL de imagem.

Independente do cenário considerado, a especificação de documentos NCL contendo componentes MPEG-4 permite que relacionamentos, anteriormente não definidos, com especialização semântica causal ou de restrição possam ser estabelecidos entre esses componentes. Nessa proposta, os documentos MPEG-4 concebidos pelo autor podem, a qualquer momento, ser relacionados entre si, ou participar de relacionamentos com outros componentes e objetos, externos às cenas audiovisuais, tais como: composições NCL ou outros objetos de mídia.

4.2. Objetos MPEG-4

Uma das possíveis representações de componentes MPEG-4, em documentos NCL, consiste em defini-los como objetos NCL de mídia contínua.

No ambiente de autoria, a representação declarativa dos componentes MPEG-4 pode ser definida pelos tipos básicos de objetos de mídia definidos em NCL (Muchaluat-Saade, 2003). Nesse ambiente, além da definição do tipo do objeto, a ferramenta de exibição, destinada ao componente MPEG-4, deve ser identificada. Essa identificação pode ser estabelecida pelo nome da classe que contém a implementação da ferramenta, descrita através do atributo *player*, no descritor de objetos destinado ao componente MPEG-4.

A Figura 4.2 apresenta um documento NCL contendo um componente MPEG-4. Nela, o documento MPEG-4, denominado *foodVideo320.mp4*, é atribuído ao componente MPEG-4 desse documento, através do tipo de objeto vídeo (*video*). Ainda na Figura 4.2, o descritor para o componente MPEG-4, denominado *videoMP4*, é especificado. Esse descritor estabelece, através do seu atributo *player*, a classe que instancia a ferramenta de apresentação destinada a esse componente, denominada *HYPF_MPEG4Player*.

```
<ncl>
  <head>
    …
    <descriptorBase>
        <descriptor id="logoDescriptor" region="logo"/>
        <descriptor id="textD1" region="text"/>
        <descriptor id="videoMP4" region="image" player="HYPF_MPEG4Player"/>
        <descriptor id="imagtextD2" region="image"/>
        <descriptor enableTimeBar="on" id="audio" region="time"/>
     </descriptorBase>
    …
  </head>
  <body>
  …
  <video descriptor="videoMP4" id="foodVideo" src="foodVideo320.mp4"/>
  …
  </body>
</ncl>
```

Figura 4.2 – Documento NCL contendo um componente MPEG-4

No ambiente de execução, a representação de componentes MPEG-4 como objetos NCL de mídia contínua aproxima as funcionalidades desejadas das ferramentas de exibição desse padrão, às atuais funcionalidades das ferramentas de exibição destinadas aos componentes dos padrões MPEG anteriores. Nas ferramentas, o controle de eventos (controle de âncoras), estabelecido sobre os objetos, é similar, característica que facilita a integração da ferramenta de exibição MPEG-4 ao conjunto de ferramentas de exibição do sistema HyperProp, onde adaptadores para exibidores MPEG-1 e MPEG-2 já haviam sido incluídos (Rodrigues, 2003).

Uma importante limitação para a inserção de uma ferramenta de exibição MPEG-4 no conjunto de ferramentas de exibição do sistema HyperProp deve-se ao fato do modelo de referência dos exibidores MPEG-4 disponibilizar apenas uma versão do exibidor MPEG-4, implementado, em grande parte, na linguagem C. Essa implementação impõe limitações operacionais para a integração com aplicações implementadas em outras linguagens, particularmente, o formatador do sistema HyperProp, que na sua versão atual é implementado na linguagem Java.

Com o objetivo de favorecer a interoperabilidade entre a ferramenta de exibição e o formatador, nesta dissertação, o projeto para desenvolvimento de aplicações relacionadas ao MPEG-4, denominado *IBM Toolkit for MPEG-4*[28], foi adotado. Esse projeto especifica um conjunto de ferramentas, implementadas na linguagem Java, cujas principais aplicações são:

[28] http://www.alphaworks.ibm.com/tech/tk4mpeg4.

- AVgen: aplicação destinada a criar arquivos MPEG-4, a partir de mídias de áudio e vídeo. Além da multiplexação, permite que os arquivos criados sejam transmitidos através de servidores de fluxos utilizando RTP/RTSP;
- XMTBatch: aplicação destinada a conversão de documentos no formato XMT-O para XMT-A ou de XMT-A para arquivos MPEG-4, especificados no formato binário e contendo as mídias relacionadas multiplexadas;
- M4Play: exibidor MPEG-4 destinado a exibir documentos desse padrão.

Cada aplicação desse projeto oferece um conjunto de APIs (*Application Program Interface*) que podem ser utilizadas no desenvolvimento de novas aplicações voltadas para o padrão MPEG-4 (*Software Developer Kit for MPEG-4 Systems*). Particularmente, as classes e métodos pertencentes à API da aplicação M4Play são diretamente utilizadas nesta dissertação no desenvolvimento de uma ferramenta de exibição para componentes MPEG-4 no sistema HyperProp.

A Figura 4.3 apresenta o diagrama de classes para a ferramenta de exibição MPEG-4 desta dissertação. A definição dessa ferramenta segue o *framework* proposto em (Rodrigues et al., 2001). A classe principal dessa ferramenta é definida em *HF_MPEG4PlayerAdapter*. Da mesma forma que a classe das ferramentas de exibição JMF (*Java Media Framework*), voltadas para a exibição de objetos MPEG-1, denominada *HF_JmfPlayerAdapter*, a classe *HF_MPEG4PlayerAdapter* herda das classes *HF_PlayerAdapter* e *HF_GenericPresentationTool* os métodos básicos oferecidos pelo formatador, implementados a partir da definição da interface *TF_PresentationTool*.

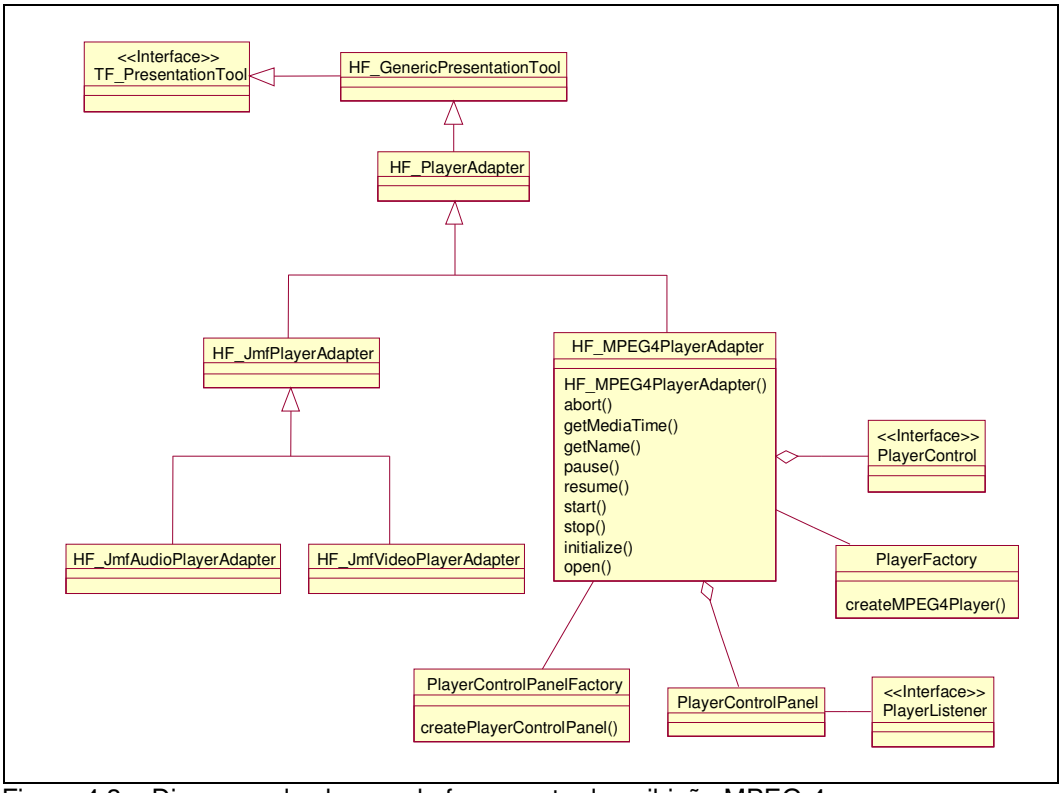

Figura 4.3 – Diagrama de classes da ferramenta de exibição MPEG-4

Durante a execução da apresentação no formatador, quando um objeto de representação específico deve ser exibido, a classe relativa à ferramenta de exibição desse objeto que implementa a interface *TF_PresentationTool* é instanciada e o seu método *initialize* executado, com os parâmetros necessários (Rodrigues et al., 2001). Quando esse objeto é um componente MPEG-4, o método *initialize* da classe *HF_MPEG4PlayerAdapter* é acionado, recebendo como parâmetro o objeto de execução, cuja estrutura possui quatro atributos principais: identificador único, objeto de dados, descritor e lista de eventos.

No método *initialize* da classe *HF_MPEG4PlayerAdapter*, as APIs para criação da ferramenta são utilizadas. Basicamente, essas APIs permitem a definição de duas regiões principais, que compõem a ferramenta. A primeira região corresponde ao local onde o componente MPEG-4 é executado, enquanto a segunda define onde o controle da apresentação dessa ferramenta pode ser disponibilizado. Esse controle da apresentação permite ao usuário interagir com o componente exibido, através de uma interface com botões relativos às ações de iniciar, parar, pausar, reiniciar etc. Além de oferecer opções de interação, esse controle fornece também informações, como o tempo transcorrido a partir do início da apresentação.

No método *initialize*, a ferramenta de exibição propriamente dita é instanciada, através da chamada ao método *createMPEG4Player* da classe *PlayerFactory*. Esse método retorna uma instância do exibidor, através da implementação da interface *PlayerControl*. Todos os acessos às funções dessa ferramenta são realizados através dos métodos dessa interface. Essas funções resumem-se ao controle dos seus estados e das suas propriedades. As propriedades definem aspectos da apresentação, como o controle do volume e o tempo transcorrido a partir do início da apresentação. Os estados, definidos internamente à ferramenta, são apresentados na Figura 4.4.

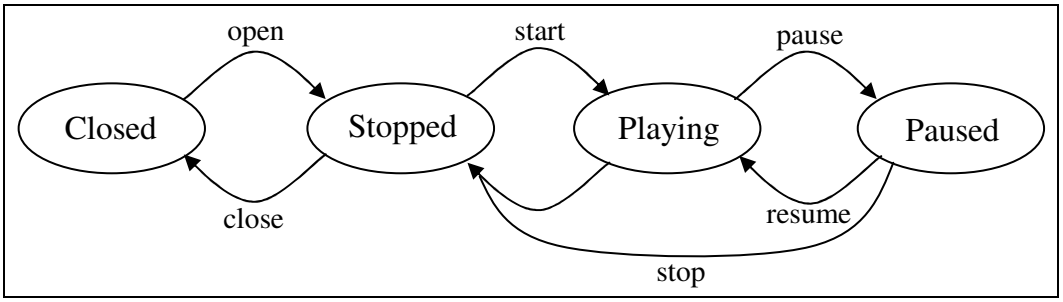

Figura 4.4 – Máquina de estados da ferramenta de exibição MPEG-4

Quando a ferramenta de exibição é criada, ela se encontra no estado *Closed*. O método *open*, definido na interface *PlayerControl*, recebe como parâmetro a URL de um documento MPEG-4 e carrega esse documento no exibidor. O método *open* somente é executado com sucesso se a ferramenta se encontra no estado *Closed*, alterando o estado da ferramenta para *Stopped*. Nesse estado, o método *start* pode ser acionado para iniciar a apresentação do componente MPEG-4 e, conseqüentemente, para alterar o estado da ferramenta para *Playing*. A execução de um componente MPEG-4 pode ser interrompida, através da chamada ao método *stop*. A execução também pode sofrer uma pausa, através da chamada ao método *pause*. Caso o método *stop* seja acionado, a ferramenta volta para o estado *Stopped*, por outro lado, sendo o método *pause* acionado, o estado da ferramenta torna-se *Paused*, estado no qual a ferramenta pode voltar a exibir o componente, do ponto interrompido, passando para o estado *Playing*, através do método *resume*, ou finalizar a apresentação, através do método *stop*, voltando para o estado *Stopped*. Finalmente, no estado *Stopped*, o documento MPEG-4 pode ser fechado, fazendo com que a ferramenta altere o seu estado para *Closed*.

Ao término natural da apresentação de um componente, a ferramenta encontra-se no estado *Playing*. Quando isso ocorre, ela pode voltar naturalmente para o estado *Stopped*, condição definida na ferramenta como

END_ACTION_STOP. Outra possibilidade consiste na ferramenta permanecer no estado *Playing* indefinidamente, até que o método *stop* seja acionado. Essa condição é definida na ferramenta como *END_ACTION_CONTINUE*. Uma terceira possibilidade consiste em voltar, quando o componente termina, ao estado *Stop* e, imediatamente, retornar para o estado *Playing*, condição definida como *END_ACTION_REPEAT*. A escolha do modo de término da apresentação pode ser realizada na ferramenta e, na última opção apresentada, o número de repetições pode ser previamente configurado.

Além dos métodos citados, a máquina de estados da ferramenta é complementada pelos métodos *playURL* e *stopURL*. No primeiro, a ferramenta, independentemente do estado original, passa para o estado *Playing*. Esse método recebe como parâmetro a URL de um documento MPEG-4. No segundo, a ferramenta passa para o estado *Stopped* independente do estado original em que se encontrava.

Voltando ao método *initialize*, o controle da ferramenta de apresentação pode ser instanciado, através da chamada ao método *createPlayerControlPanel* da classe *PlayerControlPanelFactory*. Esse método retorna uma instância para a classe *PlayerControlPanel*, que consiste em um pequeno painel contendo botões de controle para iniciar, pausar, interromper, controlar o volume etc. Os métodos associados aos botões desse painel são controlados diretamente pela interface *PlayerListener*. Dessa forma, mudanças no estado da ferramenta são refletidas diretamente no painel sem a necessidade de adicionar novos métodos. Como exemplo, se a ferramenta encontra-se no estado *Paused*, o botão relativo à ação de iniciar, encontra-se desabilitado.

Ao término do método *initialize*, o conteúdo do componente MPEG-4 é carregado na ferramenta através da chamada ao método *open*. A função básica desse método é acionar o método *open*, definido na interface *PlayerControl* da ferramenta. Vários outros métodos definidos na classe *HF_MPEG4PlayerAdapter* têm como função acionar chamadas aos métodos da máquina de estados da ferramenta, entre esses, estão os seguintes métodos: *abort* (*stop*), *pause* (*pause*), *resume* (*resume*), *start* (*start*) e *stop* (*stop*). Além de alterar os estados da ferramenta de exibição, os métodos definidos em *HF_MPEG4PlayerAdapter* devem informar ao formatador as transições nas máquinas de estados dos eventos NCM durante a apresentação dos objetos.

Como exemplo de aplicação da ferramenta, considere um componente MPEG-4, composto apenas por um objeto de vídeo. No documento NCL são definidos vários relacionamentos de sincronização desse componente MPEG-4 com objetos NCL de áudio, texto e imagem. A Figura 4.5 apresenta a visão temporal da apresentação do documento. Na Figura 4.5 pode ser observado que, conforme o componente MPEG-4 é exibido, vários objetos de áudio e de texto deverão ser exibidos, de forma síncrona. Durante toda a exibição do componente MPEG-4, uma imagem, relativa a um logotipo deve ser apresentada.

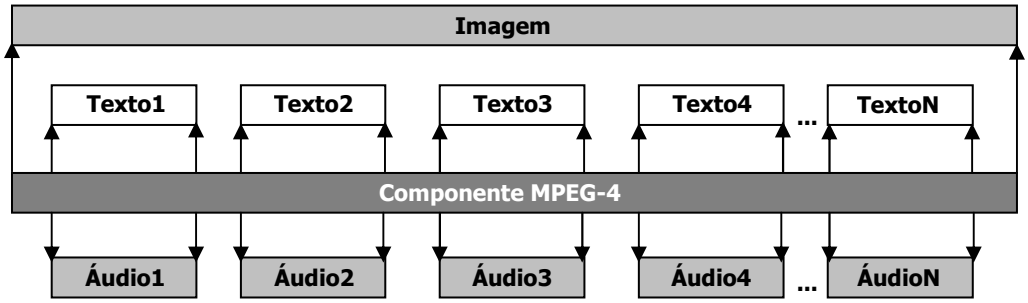

Figura 4.5 – Visão temporal de um documento NCL contendo um componente MPEG-4

A Figura 4.6 apresenta um instantâneo da exibição do documento NCL correspondente à visão temporal da Figura 4.5. O componente MPEG-4 é exibido através da ferramenta de exibição desenvolvida, onde se pode ver, além da sua região de exibição, uma barra de controle da apresentação. Através dessa interface é possível, por exemplo, pausar a apresentação do componente. Nesse caso, toda a apresentação permanece aguardando o reinício da exibição desse componente, preservando o sincronismo temporal estabelecido.

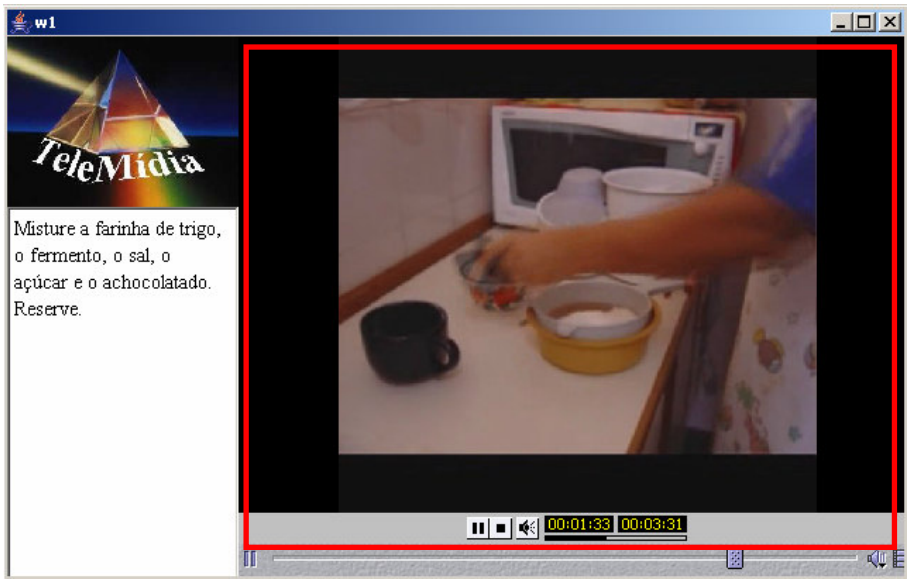

Figura 4.6 – Apresentação do documento NCL com um componente MPEG-4

4.3. Composições MPEG-4

Componentes MPEG-4 são formados por uma estrutura complexa, que pode ser comparada a uma composição NCL. Essa estrutura contém objetos internos, que podem estar relacionados entre si, ou com objetos definidos em cenas audiovisuais distintas, pertencentes a outros componentes. As propriedades e os relacionamentos definidos internamente a esses componentes são especificados em um formato próprio, que define a sua semântica de apresentação. Tradicionalmente, essa sincronização segue o paradigma *timeline*, no entanto, usando esse paradigma, com pequenas extensões, alguns relacionamentos são preservados.

Os relacionamentos sempre preservados são aqueles associados a eventos de interatividade, cuja ocorrência não pode ser prevista durante uma apresentação. Para definir esses eventos, rotas (*routes*) e sensores (*sensors*) (ISO/IEC, 2001), propostos originalmente em VRML (ISO/IEC, 1997), são adotados pelo MPEG-4.

No MPEG-4, conforme apresentado no Capítulo 2, uma cena audiovisual é construída como um grafo dirigido acíclico. Os nós desse grafo podem ser classificados em: nós de grupo (*grouping nodes*), com a função de agrupar outros nós; nós interpoladores (*interpolator nodes*), para o controle das animações; nós sensores (*sensor nodes*), que são utilizados em eventos interativos; e, finalmente, nós filhos (*children nodes*), que efetivamente representam os objetos (Bouilhaguet et al., 2000).

Independente do tipo do nó, suas propriedades são definidas nos seus campos. Existem vários tipos de campos, que se encontram especificados na referência (ISO/IEC, 2001). Do ponto de vista da definição dos eventos de interatividade, esses campos podem ser classificados nos seguintes tipos: campo (*field*), destinado aos campos cujos valores são estabelecidos apenas uma vez; evento de entrada (*eventIn*), que somente estão habilitados a receber eventos; evento de saída (*eventOut*), que somente podem emitir eventos; e campo exposto (*exposedField*), onde os campos podem emitir e receber eventos e ainda ter valores iniciais, de forma similar aos campos do tipo *field* (Bouilhaguet et al., 2000; Concolato & Feuvre, 2003).

Nós classificados como sensores geram eventos, que podem ser disparados pela interação do usuário com um nó específico ou, eventualmente, por modificações na cena (ISO/IEC, 2001). Os eventos gerados nos sensores alteram os valores dos seus campos do tipo evento de saída (*eventOut*). As alterações nesses campos, por sua vez, podem ser atribuídas aos campos do tipo evento de entrada (*eventIn*) e campo exposto (*exposedField*) de um nó qualquer da cena audiovisual, modificando o valor desse campo e, conseqüentemente, as propriedades do nó escolhido.

Para associar os campos definidos nos sensores aos campos dos demais nós, uma rota deve ser estabelecida entre eles. Nessas rotas, o escopo do sensor é limitado a sua cena audiovisual, isto é, rotas propagam eventos entre os elementos pertencentes a uma mesma cena. Como exemplo, a Figura 4.7 apresenta a representação gráfica da transmissão de eventos no formato binário do MPEG-4.

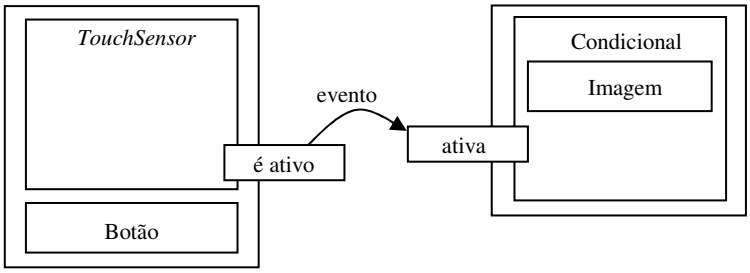

Figura 4.7 – Eventos em BIFS

Na Figura 4.7, o sensor (*TouchSensor*) é especificado internamente a um nó de grupo que contém também um nó filho, cujo conteúdo corresponde ao objeto sintético denominado “Botão”. Um nó do tipo sensor possui vários campos, entre eles, destaca-se o campo associado ao evento do clique do mouse sobre os objetos definidos internamente ao seu grupo, denominado “é ativo” (*isActive*), do tipo evento de saída (*eventOut*). Ainda na Figura 4.7, esse campo do sensor encontra-se vinculado, através de uma rota, ao campo que define se um nó condicional encontra-se ativo ou não, denominado “ative” (*activate*), do tipo evento de entrada (*eventIn*). Dentro desse nó condicional, é definido um outro nó, contendo um objeto de mídia imagem. Caso o evento de interatividade do usuário, definido pela ação do clique do mouse sobre o objeto sintético ocorra, o campo “é ativo” do sensor recebe o valor verdadeiro e, em virtude dessa mudança, através da rota, o campo “ative” do nó condicional é modificado, tendo o seu conteúdo substituído pelo valor do campo “é ativo”. Considerando que originalmente, o

valor do campo “ative” é definido como falso, quando o seu valor tornar-se verdadeiro, a imagem, definida internamente ao nó condicional, será apresentada.

A Tabela 4.1 apresenta alguns dos principais sensores definidos no MPEG-4. A lista completa desses sensores pode ser encontrada em (ISO/IEC, 2001). Todos os sensores apresentados possuem campos que representam alterações relacionadas aos eventos do mouse sobre os nós de uma cena audiovisual. Esses campos podem ter valores de diversos tipos, como no exemplo apresentado na Figura 4.7, onde o campo “é ativo” do sensor é do tipo booleano (*SFBool*).

Nós BIFS	Descrição
TouchSensor	Detecta o movimento do mouse e as ações dos botões.
DiscSensor	Detecta o movimento do mouse e o traduz em eventos de rotação, com origem no sistema de coordenadas cartesianas.
PlaneSensor2D	Detecta o movimento do mouse e o traduz em eventos de translação, com origem no sistema de coordenadas cartesianas.
ProximitySensor2D	Detecta a presença do mouse em uma região específica e gera eventos de posicionamento, tempo e estado (se está dentro ou fora da região).
Anchor	Detecta as ações dos botões do mouse.

Tabela 4.1 – Principais sensores BIFS

Sensores e rotas definidos em componentes MPEG-4 são elementos atípicos, não encontrados em outros padrões de codificação. Para que esses elementos possam ser representados em documentos multimídia/hipermídia especificados em NCL, é desejável representar os componentes MPEG-4 como composições NCL. Sensores devem ser mapeados em âncoras, associadas aos eventos. Caso se queira utilizar os eventos para definir relacionamentos com objetos externos à cena, as âncoras, correspondentes aos sensores, devem ser mapeadas em portas (interfaces) da composição. A Figura 4.8 apresenta uma composição, onde um elo é especificado associando um nó interno do componente MPEG-4 a um nó de mídia (*A*), especificado no documento multimídia/hipermídia representado pela composição mais externa.

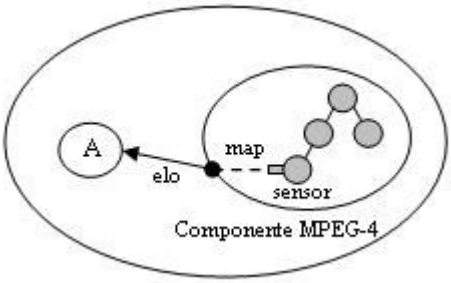

Figura 4.8 – Elo estabelecido entre uma composição MPEG-4 e um objeto

Na Figura 4.8, o relacionamento entre os nós é mapeado através de uma interface definida sobre o componente MPEG-4. Nesse componente os eventos originados pelos sensores definem mudanças nos valores dos seus campos (*eventOut* e *exposedField*). No mapeamento desses eventos para as interfaces (portas da composição), alterações nos valores dos campos pertencentes aos sensores devem ser externadas, indicando sua ocorrência. No sentido inverso, alterações nos eventos ligados às portas da composição devem ser mapeadas para os campos dos sensores.

Nos elos NCL, *binds* devem associar as interfaces dos componentes MPEG-4 a papéis do tipo ação ou propriedade, especificados nos conectores hipermídia. Nesses conectores os valores dessas interfaces definem condições para a execução de uma ação, como, por exemplo, para executar um papel do tipo ação. No sentido inverso, conectores, a partir de alguma condição, podem definir papéis para atribuir valores a essas interfaces. A Figura 4.9 apresenta um relacionamento entre um nó interno ao componente MPEG-4 e um nó definido no documento multimídia/hipermídia.

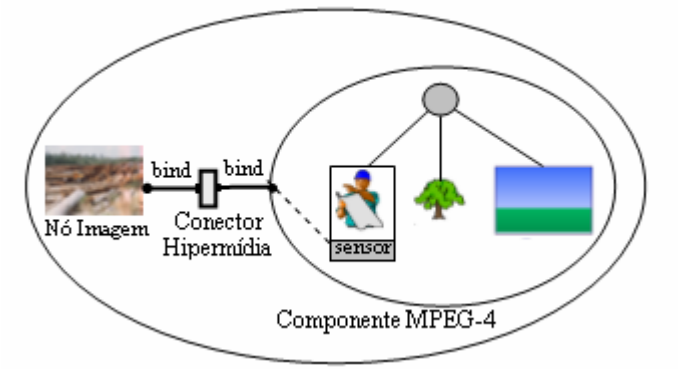

Figura 4.9 – Apresentação NCL contendo uma composição MPEG-4

No componente MPEG-4 da Figura 4.9, um nó sensor é definido internamente ao nó de grupo que contém um objeto de vídeo. No documento NCL (representado pela composição mais externa da figura), um conector hipermídia é definido, com especialização semântica causal, contendo dois papéis. Nesse documento, um elo é definido, estabelecendo um relacionamento entre o componente MPEG-4 e um nó representando uma mídia imagem. A relação desse elo é definida pelo conector, onde um dos seus papéis é associado, através de um *bind,* ao nó de imagem. O outro papel desse conector encontra-se associado, através de outro *bind*, à interface do componente MPEG-4.

Quando o evento relativo ao clique do mouse é acionado sobre o objeto de vídeo do componente MPEG-4, o campo "é ativo" (*isActive*) do sensor torna-se verdadeiro. Nesse momento, o valor desse campo deve ser mapeado para a interface correspondente do componente, que reporta os seus eventos internos. Essa interface encontra-se associada ao conector hipermídia, onde uma condição é estabelecida ao valor desse papel a fim de executar um papel do tipo ação, associado ao nó de imagem. A condição deve ser verdadeira quando o valor da interface também o for, realizando a apresentação da imagem.

A fim de apresentar os componentes MPEG-4 como composições NCL, a ferramenta de exibição associada a esses componentes deve mapear as ocorrências entre os seus eventos internos e suas interfaces. A Seção 4.2 discute como uma ferramenta de exibição MPEG-4 foi adicionada ao conjunto de ferramentas do sistema HyperProp. A partir dessa integração, para apresentar os componentes MPEG-4 como composições, são necessárias modificações na ferramenta, implementando o mapeamento entre os eventos e os pontos de interface dos componentes.

Como a ferramenta de exibição apresentada na Seção 4.2 não pode ser adaptada, pelo fato da sua licença de uso não permitir modificações no seu código, para implementar modificações no exibidor MPEG-4 foi necessário adotar um exibidor licenciado como código aberto, o que restringiu as opções disponíveis à ferramenta de exibição publicada pelo padrão (ISO/IEC, 2000b) e ao OSMO4 (*Osmose*) (GPAC, 2002). É importante ressaltar que, nas opções existentes, as implementações estão disponíveis apenas na linguagem C/C++.

O OSMO4 é um exibidor MPEG-4 desenvolvido pelo GPAC que segue o padrão de referência do MPEG-4. Os softwares desenvolvidos por esse grupo têm como objetivo serem alternativas de implementação menores, mais compreensíveis e flexíveis em relação aos softwares originais desse padrão (GPAC, 2002), motivos pelos quais o OSMO4 foi escolhido como a ferramenta de exibição a ser adaptada. A versão dessa ferramenta, adotada nesta dissertação, é denominada 2.0 e foi publicada em setembro de 2004. Nessa versão, o código fonte é fornecido na linguagem C, com algumas classes C++. Esse código encontra-se organizado sobre um projeto para o compilador Visual C++ 6.0[29].

[29] http://msdn.microsoft.com/visualc/default.asp

Na implementação do OSMO4, as classes *cMainFrame* e *WinGPAC* definem a sua interface gráfica principal. Nelas, as funções para manipulação dos documentos MPEG-4 são implementadas e disponibilizadas através de componentes gráficos. Essas classes implementam a região onde os componentes MPEG-4 são apresentados, bem como as interfaces relacionadas à apresentação, como a barra de informações da aplicação, a barra de ferramentas e o menu de opções, onde as ações de iniciar, parar, pausar, reiniciar, entre outras, aplicadas aos componentes MPEG-4 exibidos, são oferecidas.

Com exceção das funções presentes nas interfaces gráficas, que são especificadas através de classes, a maior parte das funcionalidades dessa ferramenta são especificadas através de bibliotecas C. A Tabela 4.2 apresenta os principais arquivos da implementação dessa ferramenta, selecionados nesta dissertação para a manipulação de cenas audiovisuais MPEG-4, em relação aos sensores e rotas. É importante mencionar que os arquivos listados na Tabela 4.2 foram sintetizados a partir da análise direta do código fonte da ferramenta, pois não há documentação disponível sobre o seu funcionamento (GPAC, 2002).

Arquivo	Descrição
m4_esm_dev.h	Define a estrutura da região para apresentação de um componente MPEG-4 e a estrutura da cena audiovisual apresentada nessa região.
m4_scenegraph_dev.h.	Define detalhes da estrutura audiovisual como rotas.
route.c	Implementa os procedimentos para manipulação das rotas.
m4_nodes.h	Define a estrutura dos nós pertencentes a uma cena audiovisual.
render2d.c	Implementa os procedimentos para manipulação da região onde componentes MPEG-4 são apresentados.
sensor_stacks.c	Implementa os procedimentos para manipulação dos sensores.
M4Term.c	Implementa os procedimentos para acesso através do terminal audiovisual. Nesse arquivo devem ser implementados os procedimentos a serem acessados a partir da interface gráfica.

Tabela 4.2 – Arquivos principais da ferramenta de exibição

Na Tabela 4.2, a biblioteca *m4_esm_dev.h* especifica as informações sobre a região da apresentação de um componente MPEG-4, através de uma estrutura denominada *_m4_client*. Nessa estrutura, o conteúdo da cena audiovisual apresentada é armazenado através de um ponteiro para o elemento raiz da cena, denominado *root_scene*.

No exibidor OSMO4, a própria cena audiovisual também é uma estrutura, denominada *_inline_scene*, especificada também na biblioteca *m4_esm_dev.h*. Essa estrutura armazena as informações que uma cena audiovisual deve conter, como os descritores de objeto para o conteúdo dos seus nós e o seu tempo para apresentação. As informações sobre rotas e sensores, definidos na cena

audiovisual, são armazenadas através de um ponteiro, denominado *graph*, para a estrutura *_tagSceneGraph*.

Na biblioteca *m4_scenegraph_dev.h* é definida a estrutura denominada *_tagSceneGraph*, que armazena as informações sobre os sensores. No entanto, as informações sobre os sensores não existem isoladamente, e sim, dentro das rotas especificadas em uma cena audiovisual MPEG-4. Evidentemente, essas informações podem ser obtida através das rotas, porém sensores que não estejam associados a nenhuma rota não serão identificados.

Dentro da estrutura *_tagSceneGraph* é especificada uma estrutura de fila, correspondente às rotas estabelecidas em uma cena audiovisual. Cada rota, por sua vez, é definida por uma estrutura, denominada *struct_route*, que também é especificada na biblioteca *m4_scenegraph_dev.h*. A Figura 4.10 apresenta parte dessa estrutura.

```
typedef struct _route
{
        SFNode *FromNode;
        u32 FromFieldIndex;
        FieldInfo FromField;

        SFNode *ToNode;
        u32 ToFieldIndex;
        FieldInfo ToField;

        const char *fromFieldName;
  …
} Route;
```

Figura 4.10 – Estrutura das rotas na ferramenta de exibição

Na Figura 4.10 as três variáveis iniciais da estrutura *Route* definem o nó de origem da rota, através da variável **FromNode*, e o campo, através das variáveis *FromFieldIndex* e **FromField*. As três variáveis seguintes nessa estrutura, definem o nó de destino da rota, através da variável **ToNode*, e o campo, através das variáveis *ToFieldIndex* e **ToField*. Ainda nessa estrutura, a variável **fromFieldName* define o nome do campo de origem do sensor associado à rota, isto é, o nome do campo que atua como evento de origem da rota.

A ativação das rotas é realizada através do procedimento *Node_OnEventOutSTR*, implementado no arquivo *route.c*. Esse procedimento recebe como parâmetros ponteiros para o nó do tipo sensor e para o seu campo, informações correspondentes às variáveis **FromNode* e **FromField*, respectivamente, da Figura 4.10. A partir desses parâmetros as rotas associadas ao sensor e ao seu campo são ativadas.

Na realidade, antes da ativação das rotas, o mapeamento das interfaces dos componentes MPEG-4 para os sensores deve ser realizado. A ativação das rotas sem a realização prévia de alterações nos valores dos campos pertencentes aos sensores não altera a exibição do componente MPEG-4. Para realizar esse mapeamento, a interface das composições NCL, representando componentes MPEG-4, deve ser especificada contendo as informações sobre qual é o tipo do sensor a ela associado, qual é o nó da cena audiovisual correspondente ao sensor associado e, por fim qual é o evento do sensor (campo) que a interface representa.

Note que essas informações são necessárias à configuração dos sensores MPEG-4, pois nos componentes MPEG-4 podem existir vários tipos de sensores, que possuem comportamentos distintos. Esses sensores, por sua vez, podem estar associados a praticamente todos os nós de uma cena audiovisual. Por fim, um mesmo sensor pode definir vários eventos, através dos seus campos, como, por exemplo, o evento de clique do mouse, posicionamento do mouse, pressionamento do teclado etc.

A biblioteca *m4_nodes.h* define uma estrutura de dados para cada tipo de sensor que pode ser especificado no MPEG-4. A Figura 4.11 apresenta a estrutura relativa ao sensor denominado *B_TouchSensor*. Na figura, os campos desse sensor estão classificados como evento de saída (*eventOut*) ou campo exposto (*exposedField*).

```
Typedef struct _tagTouchSensor
{
        SFBool enabled;                 /*exposedField*/
        SFVec3f hitNormal_changed;      /*eventOut*/
        SFVec3f hitPoint_changed;       /*eventOut*/
        SFVec2f hitTexCoord_changed;    /*eventOut*/
        SFBool isActive;                /*eventOut*/
        SFBool isOver;                  /*eventOut*/
        SFTime touchTime;               /*eventOut*/
} B_TouchSensor;
```

Figura 4.11 – Estrutura do *TouchSensor* na ferramenta de exibição

Para acionar eventos em um componente MPEG-4, foi implementado um procedimento, denominado *M4T_NCLActiveNode*, dentro do arquivo *M4Term.c*. Esse procedimento é executado quando um evento deve ser acionado a partir de um relacionamento especificado em um documento NCL. A Figura 4.12 apresenta parte desse procedimento, que recebe como parâmetros: o nó relativo ao sensor (**node*), o nome do campo desse sensor (**eventName*), seu tipo (*tag*) e o valor a ser atribuído ao seu campo. Os três primeiros parâmetros são definidos pela

interface do componente relativo ao sensor, e o último, pelo papel do conector associado a essa interface. Após a execução desse procedimento, basta executar o procedimento *Node_OnEventOutSTR*, anteriormente citado, a fim de acionar as rotas associadas ao sensor modificado.

```
void M4T_NCLActiveNode(SFNode *node, const char *eventName, u32 tag, u32 value)
{
        if (tag == TAG_TouchSensor) {

            B_TouchSensor *ts = (B_TouchSensor *)node;

            if (!stricmp(eventName, "isActive")) ts->isActive =value;
            if (!stricmp(eventName, "isOver"))    ts->isOver =value;
        }
    …
}
```

Figura 4.12 – Procedimento para acionar os sensores

Para demonstrar as interfaces que podem ser definidas em um componente MPEG-4, a ferramenta de exibição OSMO4 foi estendida com o objetivo de apresentar os sensores associados às rotas de uma cena audiovisual. Conhecer os sensores e, conseqüentemente, os eventos internos definidos em um componente MPEG-4 é necessário para que, durante a especificação do documento multimídia/hipermídia, o autor saiba quais são os nós internos ao componente MPEG-4 que contêm sensores associados.

A Figura 4.13 apresenta a exibição de um componente MPEG-4 nessa ferramenta, onde foi desenvolvido um menu que permite ao usuário acessar as possíveis interfaces do componente. Na figura, o componente MPEG-4 exibido contém cinco sensores, definindo relacionamentos com semântica de referência, que são acionados pelos eventos de clique do mouse sobre as legendas exibidas.

Figura 4.13 – Apresentação de um componente MPEG-4

A Figura 4.14 apresenta os sensores e as rotas existentes no componente MPEG-4 exibido pela ferramenta. Através de uma outra janela, as informações sobre o tipo do sensor, seu campo e o nó a ele relacionado, são apresentadas. Nessa janela, que representa as interfaces que podem ser estabelecidas em um documento multimídia/hipermídia, os eventos podem ser ativados através da sua seleção. Na atual versão, somente os campos do tipo booleano podem ser ativados, isto é, quando o evento é selecionado, o valor booleano do campo passa para verdadeiro. Para incluir campos de outros tipos, basta que seja definida uma interface para a passagem dos parâmetros relativos aos tipos dos campos.

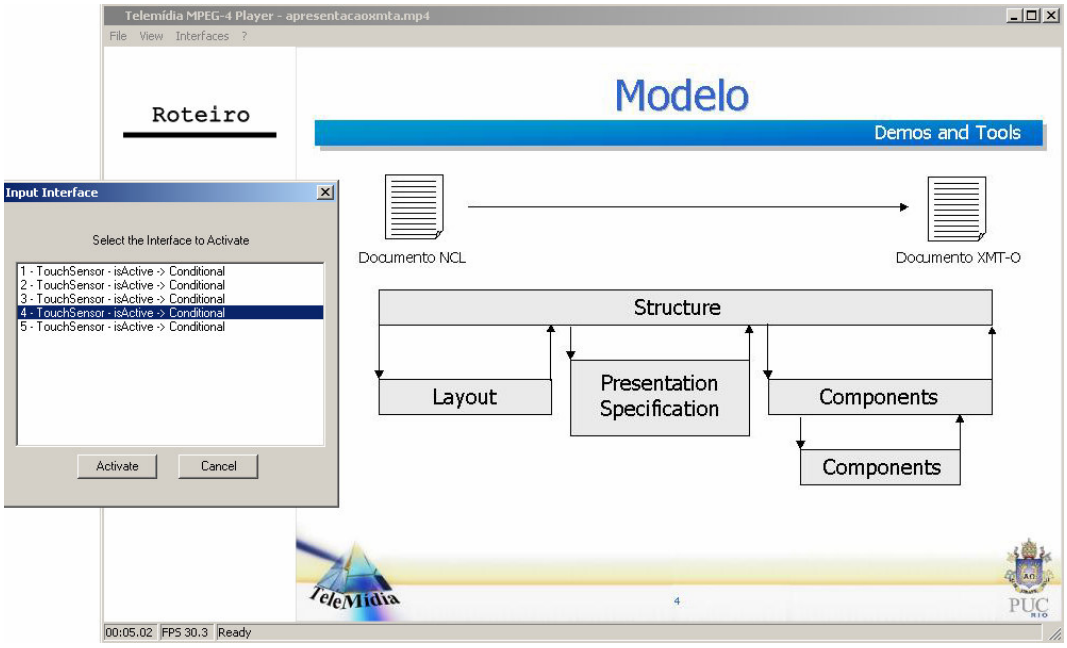

Figura 4.14 – Sensores definidos em um componente MPEG-4

Complementarmente, para mapear os eventos ocorridos em um componente MPEG-4 nas interfaces (portas) de composições NCL, representando esses componentes, outras adaptações na ferramenta devem ser realizadas. Essas adaptações têm como objetivo capturar as mudanças ocorridas nos campos dos sensores, provocadas por eventos de interatividade.

No arquivo *render2d.c*, o procedimento *R2D_ExecuteEvent* monitora todos os eventos que ocorrem na região de apresentação da ferramenta de exibição. Quando os eventos são detectados, procedimentos específicos definidos no arquivo *sensor_stacks.c* são executados, de acordo com o sensor envolvido. Nesses procedimentos, os campos dos sensores, relativos ao evento ocorrido, são ativados. Como exemplo, caso o procedimento *R2D_ExecuteEvent* observe a ação do clique do mouse sobre um nó da cena audiovisual, onde um sensor do tipo

TouchSensor é definido, o procedimento *OnTouchSensor*, definido no arquivo *sensor_stacks.c*, é acionado. Nesse caso, dentro desse procedimento, o campo "é Ativo" do sensor recebe o valor booleano verdadeiro.

Logo após o procedimento, relativo ao sensor ser executado, o procedimento *Node_OnEventOutSTR* é acionado, para executar as rotas associadas ao sensor. Nesse momento, métodos relativos ao mapeamento do sensor para as interfaces da composição NCL, representando o componente MPEG-4, devem ser executados. Esses métodos devem ser definidos na integração da ferramenta ao sistema HyperProp, de forma similar à integração apresentada na Seção 4.2. No entanto, essa integração, operacionalmente, é limitada pela restrição de comunicação entre as linguagens da ferramenta e da interface definida para comunicação com o formatador. Futuramente, pretende-se estender a ferramenta, através da definição de uma biblioteca de ligação dinâmica, que possa ser integrada com facilidade ao formatador do sistema HyperProp.

5
Extensões para Autoria no MPEG-4

Este capítulo apresenta a especificação de templates de composição hipermídia para a linguagem XMT-O. Com esse objetivo, o capítulo está organizado da forma a seguir. A Seção 5.1 propõe extensões para XMT-O, a partir dos recursos da linguagem NCL. A seguir, a Seção 5.2 aborda o perfil XT-SMIL, que corresponde à incorporação dos templates NCL à linguagem SMIL 2.0. Por fim, com base no perfil XT-SMIL, a Seção 5.3 apresenta o perfil XT-XMT-O.

5.1.
Templates para Autoria no MPEG-4

Na apresentação da linguagem NCL no Capítulo 2, os seus módulos *XConnector* e o *XTemplate* foram abordados. O módulo *XConnector* permite a especificação de conectores hipermídia. Um conector hipermídia especifica uma relação que pode ser utilizada para a criação de elos em um documento. Na realidade, o conector especifica a semântica, sem mencionar os nós participantes do relacionamento. O módulo *XTemplate*, por sua vez, permite a especificação de templates de composição hipermídia. Templates hipermídia podem especificar diversos tipos de relacionamentos entre os componentes, definindo estruturas hipermídia genéricas que podem ser herdadas por composições. Um template especifica os componentes e os relacionamentos que uma composição pode possuir, sem especificar todos os componentes e relacionamentos propriamente ditos. Essa especificação é realizada pelas composições, que herdam a semântica definida pelo template.

Conectores e templates são elementos importantes na autoria declarativa de documentos multimídia/hipermídia, pois aumentam o reuso nesse processo. Os conectores permitem que as relações sejam definidas independente dos relacionamentos, o que favorece o reuso de relações em vários relacionamentos distintos. Os templates, por sua vez, permitem o reuso de especificações semiprontas.

Além de favorecer o reuso, conectores e templates são elementos importantes para a usabilidade. A linguagem NCL possui elevado poder de expressão (Muchaluat-Saade, 2003) e, conseqüentemente, é uma linguagem bastante complexa, quando usada em toda a sua plenitude. No entanto, a definição de uma base de conectores e de templates, por usuários especialistas dessa linguagem, torna o uso de NCL bem mais simples, pois as bases de conectores e de templates encapsulam a complexidade das especificações.

As vantagens obtidas através do uso de conectores e templates podem ser estendidas a outras linguagens de autoria hipermídia. Particularmente, este capítulo propõe a inclusão de templates de composição para a autoria de documentos MPEG-4. O uso de templates nesse padrão é apresentado em alguns trabalhos (Boughoufalah et al., 2000; Boughoufalah et al., 2001). No entanto esses trabalhos têm como foco o reuso do design da apresentação, principalmente dos componentes audiovisuais formados a partir das composições de objetos sintéticos simples, e não o reuso da especificação de relacionamentos complexos, que podem ser estabelecidos entre os objetos de uma apresentação.

Entre as linguagens para autoria de documentos do padrão MPEG-4, XMT-O, apresentada no Capítulo 2, destaca-se por permitir, através das suas construções, que a concepção dos documentos seja realizada com base nas intenções do autor. XMT-O é uma linguagem declarativa e modular, características que favorecem a incorporação de novas funcionalidades, através da adição de módulos definidos em outras linguagens, como *XConnector* e *XTemplate*, definidos na linguagem NCL.

Linguagens declarativas compostas por módulos podem ter suas funcionalidades agrupadas em perfis de linguagem. Cada perfil pode, de acordo com a necessidade, reunir um subconjunto dos módulos componentes da linguagem, gerando subconjuntos da própria linguagem, apropriados para casos específicos. Além disso, como essas linguagens podem incorporar módulos originalmente definidos em outras, perfis podem ser utilizados a fim de identificar essa incorporação.

A extensão da linguagem XMT-O, através da adição dos módulos *XConnector* e *XTemplate*, cria dois novos perfis de XMT-O, propostos nesta dissertação como: XC-XMT-O e XT-XMT-O. O primeiro perfil tem como

objetivo aplicar o conceito de conectores à linguagem XMT-O e, o segundo, introduzir as vantagens do uso de templates nessa linguagem.

A incorporação do módulo *XConnector* à linguagem XMT-O torna necessário distinguir, nos relacionamentos especificados nessa linguagem, seus aspectos semânticos, definidos como relações, dos seus participantes. As adaptações necessárias a esse primeiro perfil não serão abordadas nesta dissertação, porém, futuramente, esse perfil pode ser implementado, obtendo, por exemplo, um perfil único, que utilize tanto conectores quanto templates.

O segundo perfil, definido como XT-XMT-O, permite aos autores a definição de estruturas hipermídia que, quando herdadas pelas composições, ofereçam novas semânticas temporais, além das obtidas com as composições tradicionais da linguagem XMT-O.

Na versão original do módulo *XTemplate* (versão 2.0), os relacionamentos, necessários para a definição das estruturas hipermídia nos templates são especificados através de elos, que necessitam dos conectores para definir sua semântica. Dessa forma, a utilização desse módulo por um determinado perfil de linguagem, exigiria a utilização do módulo *XConnector*.

Recentemente, uma nova versão da linguagem *XTemplate* (versão 2.1) tornou-a mais flexível, permitindo também a definição de relações de inclusão (Silva et al., 2004a). A nova linguagem XT-XMT-O proposta nesta dissertação utiliza, como já mencionado, apenas o perfil da linguagem *XTemplate* que permite relações de inclusão. Fica como trabalho futuro a definição de uma nova extensão para XMT-O utilizando o perfil completo *XTemplate*, incorporando relações de inclusão e as definidas por conectores.

Independente da linguagem onde o módulo *XTemplate* for incorporado, é importante mencionar que o conceito de template de composição é uma facilidade da linguagem de autoria. Dessa forma, um documento multimídia/hipermídia que, na sua especificação, faz referência a um template, deve ser previamente processado a fim de ser exibido. Um processador de templates para a linguagem NCL é proposto em (Muchaluat-Saade, 2003). Posteriormente esse processador foi estendido, para templates do perfil X-SMIL (Silva et al., 2004a). O processamento de templates para a linguagem XMT-O, que gera um novo documento XMT-O a partir de um documento especificado em XT-XMT-O e das definições dos templates relacionados, é abordado na seção final deste capítulo.

5.2. Perfil *XTemplate* de SMIL

Na linguagem NCL os relacionamentos são definidos através de elos e composições, sendo que as últimas representam apenas relacionamentos de inclusão. Nessa linguagem, o uso de templates tem como objetivo embutir semântica de relacionamentos, dos mais diversos tipos, nas composições, facilitando a tarefa de autoria do usuário. As composições com semântica embutida podem tornar, a princípio, mais fácil a tarefa de autoria, uma vez que, em uma única composição, podem ser estabelecidas relações para as quais seria necessário o uso de vários elos e outras composições.

Em algumas linguagens, como SMIL, as composições podem representar relações de sincronização temporal ou mesmo de escolha seletiva. No entanto, nessas linguagens, a existência de um conjunto limitado de composições pode restringir as facilidades para autoria, pois para definir relacionamentos complexos pode ser necessário estabelecer composições com vários níveis de aninhamento (Rodrigues et al., 2002).

Com o objetivo de facilitar a autoria em linguagens como SMIL, templates podem definir estruturas semânticas contendo relações de inclusão recursivas, que podem ser herdadas pelas composições. Essas relações permitem especificar composições com novas semânticas temporais. A especificação dos objetos relacionados fica a cargo das composições que herdam os templates.

A estrutura dos templates, especificados segundo o perfil XT-SMIL, é formada pelos nós de composição denominados *par, seq* e *excl*, todos definidos em SMIL. Além desses nós, que representam as relações de sincronização tradicionais da linguagem, nós *switch*, que agrupam um conjunto de componentes alternativos, também podem fazer parte das relações especificadas no template. A linguagem SMIL também permite a definição de relacionamentos através de eventos, especificados nos atributos dos seus elementos. Coerentemente, a estrutura dos templates da linguagem XMT-O também se utiliza de eventos.

A fim de possibilitar a definição do perfil XT-SMIL, alguns elementos devem estar presentes na definição da linguagem *XTemplate*, apresentada na Seção 2.3.3. A Tabela 5.1 apresenta os elementos, atributos e conteúdos, definidos na linguagem *XTemplate*, com as modificações propostas em sua versão 2.1 (Silva

et al., 2004a). Os símbolos da Tabela 5.1 possuem os seguintes significados: "?" opcional, "|" ou, "*" zero ou mais ocorrências e "+" uma ou mais ocorrências.

Elemento	Atributo	Conteúdo
xtemplate		*(vocabulary, constraints?)*
vocabulary		*(component+,conectors*)*
component	*type, ctype, maxOccurs, minOccurs*	*(port | component)**
port	*type, maxOccurs, minOccurs*	
constraints		*(constraint | resource | linkBase | XSLT)**
constraint	*select, description*	
resource	*src, type, label*	*resource*
conectors	*src, type, maxOccurs, minOccurs*	
linkBase	*id*	*(link | XSLT)**
link	*type*	*bind**
bind	*role, select*	

Tabela 5.1 – Elementos, atributos e conteúdo da linguagem *XTemplate*

Conforme comentado na seção 2.3.3, a especificação de um template é composta por duas partes: o vocabulário, definido pelo elemento *vocabulary*, e as restrições, definidas pelo elemento *constraints*.

No vocabulário, onde os componentes do template são especificados, esses podem conter, além de pontos de interface definidos pelo elemento *ports*, outros componentes (*component*) recursivamente. Conectores não são permitidos no perfil *XTemplate* da linguagem XT-SMIL, bem como outros elementos, apontados ainda nesta seção.

Nas restrições são definidas as regras sobre os elementos especificados no vocabulário, bem como instâncias de componentes, definidas através do elemento *resource*, que também podem conter outras instâncias de componentes (*resource*) recursivamente.

Para incluir atributos nos elementos de uma composição, os templates, especificados para a linguagem SMIL, utilizam as instruções da linguagem XSLT (W3C, 2005c) em conjunto com as expressões da linguagem XPath. As linguagens XPath e XSLT são, sobretudo, empregadas para a especificação de restrições sobre os componentes e pontos de interface, mas também permitem a especificação de atributos e de conteúdos para as instâncias de componentes, representadas pelo elemento *resource*.

Ainda na Tabela 5.1, alguns elementos não são utilizados para a especificação de templates SMIL. Como os relacionamentos, especificados nesses templates, a princípio, não são estabelecidos através de elos e conectores, os elementos *conectors*, *linkBase*, *link* e *bind*, definidos na parte final da Tabela 5.1,

não são empregados por esse perfil. Esses elementos fazem parte do perfil completo de *XTemplate*.

Os templates SMIL, depois de definidos, podem ser diretamente referenciados por composições XT-SMIL. A Figura 5.1 (1) apresenta a visão estrutural de uma composição que referencia um template. Nessa composição, que possuirá nova semântica oferecida pelo template, são definidos objetos de mídia imagem e texto, além de um objeto de áudio com vários pontos de interface associados (âncoras).

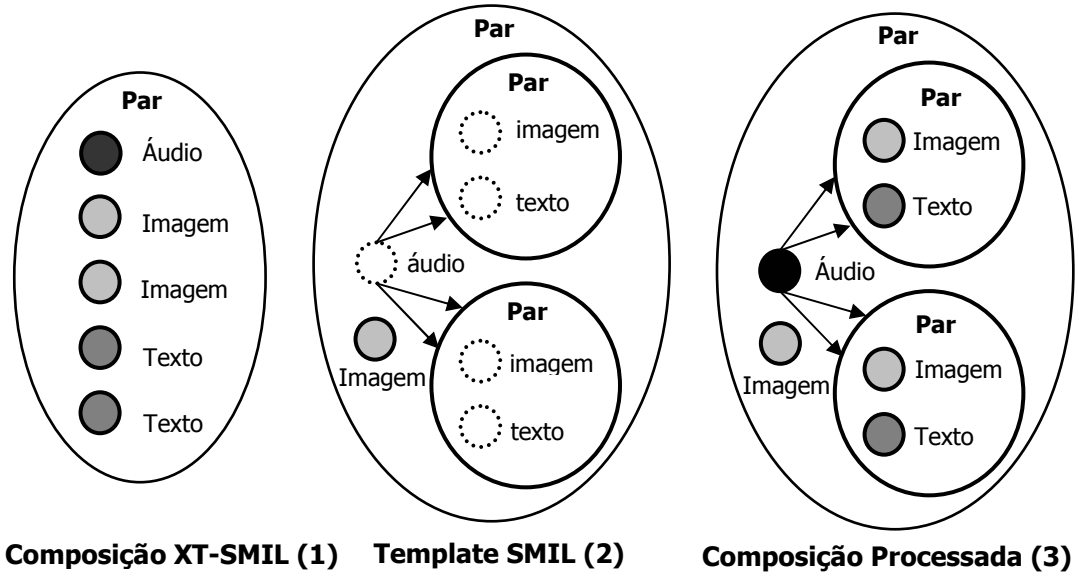

Figura 5.1 – Composição SMIL com semântica definida por um template hipermídia

Na Figura 5.1 (2), o template, referenciado pela composição XT-SMIL (1), especifica uma estrutura semântica formada por uma composição paralela, que contém outras composições, também com semântica paralela. Cada uma dessas composições terminais contém duas referências para objetos, uma para o objeto do tipo imagem e outra para o objeto do tipo texto. Na composição paralela mais externa, também se encontram definidas uma instância de um objeto de mídia imagem e uma referência para um objeto do tipo áudio. Esse objeto de áudio contém vários pontos de interface que se relacionam com as composições paralelas mais internas do template, através dos atributos definidos nessas composições. Esses relacionamentos possuem semântica de sincronização temporal, onde os pontos de interface acionam eventos de apresentação dessas composições, relacionados às transições de início e término de cada apresentação. Note que nesse template somente o objeto de mídia imagem, pertencente à composição mais externa, é instanciado. Os demais objetos correspondem a

referências, a serem instanciadas pelas composições onde esse template for herdado. O mesmo se aplica aos pontos de interface do objeto do tipo áudio. Essas âncoras participam de relacionamentos, porém somente serão especificadas pela composição que vier a herdar esse template.

Ainda na Figura 5.1, após o processamento do template (2), referenciado por essa composição, uma nova semântica para a apresentação é obtida (3). A nova apresentação da composição exibe, inicialmente, os objetos do tipo áudio e do tipo imagem (definida pelo template). Cada ponto de interface do áudio é relacionado a uma composição paralela, de forma que, quando esse ponto é alcançado, a composição é executada. Internamente às composições paralelas, são definidos um objeto do tipo imagem e um objeto do tipo texto, que são apresentados simultaneamente, quando cada composição é executada.

5.3. Perfil *XTemplate* de XMT-O

Para a especificação de templates no perfil XT-XMT-O, não são necessárias modificações na estrutura do perfil *XTemplate*, apresentado na seção anterior e utilizado por XT-SMIL, pois os relacionamentos em XMT-O, de forma idêntica à SMIL, são definidos através de eventos, especificados nos atributos dos seus elementos, ou pelo aninhamento das suas composições. Como exemplo, a Figura 5.2 apresenta um template especificado nesse perfil.

```
<?xml version=”1.0” ?>
<xtemplate name="XTemplate Example" id="audio-with-subtitles"
 xmlns="http://www.telemidia.puc-rio.br/specs/xml/XTemplate" … >

<vocabulary>
  <component type="audio" ctype="audio" maxOccurs="1" minOccurs="1"/>
  <component type="logoResource" ctype="img" maxOccurs="1" />

    <component type="seq" ctype="seq">
    <component type="par" ctype="par">
              <component type="image" ctype="img" maxOccurs="1" minOccurs="1" />
              <component type="text" ctype="string" maxOccurs="1" minOccurs="1" />
    </component >
    </component >
 </vocabulary>

<constraints>
  <resource type="logoResource" label="logo" src="../img/BackDrop.jpg" />

  <xsl:template match="/*/*/*">
     <xsl:if test="not(./@type='image') and not(./@type='text')">
       <xsl:copy-of select="." />
     </xsl:if>
```

```
    </xsl:template>

  <xsl:template name="imageElement">
   <xsl:param name="i"></xsl:param>
   <xsl:for-each select="/*/*/child::*[@type='image'][$i]" >
                    <xsl:copy>
                              <xsl:for-each select="text()|@*"><xsl:copy/></xsl:for-each>
                              <xsl:copy-of select="*"/>
                    <xsl:apply-templates/>
                    </xsl:copy>
   </xsl:for-each>
  </xsl:template>

  <xsl:template name="textElement">
    <xsl:param name="i"></xsl:param>
      <xsl:for-each select="/*/*/child::*[@type='text'][$i]" >
                    <xsl:copy>
                      <xsl:for-each select="text()|@*"><xsl:copy/></xsl:for-each>
                      <xsl:copy-of select="*"/>
                    <xsl:apply-templates/>
                    </xsl:copy>
    </xsl:for-each>
  </xsl:template>

  <resource type="seq">
            <xsl:for-each select="child::*[@type='image']" >
                  <resource type="par">
                  <xsl:variable name="i" select="position()"/>
                  <xsl:attribute name="id">par<xsl:value-of select="$i"/></xsl:attribute>
                  <xsl:call-template name="imageElement">
                        <xsl:with-param name="i" select="$i"></xsl:with-param>
                     </xsl:call-template>
                   <xsl:call-template name="textElement">
                        <xsl:with-param name="i" select="$i"></xsl:with-param>
                     </xsl:call-template>
               </resource>
            </xsl:for-each>
  </resource>

</constraints>
</xtemplate>
```

Figura 5.2 – Template para o perfil XT-XMT-O

Na Figura 5.2, dentro do vocabulário (*vocabulary*) são declarados três componentes (*component*), um componente do tipo *audio*, relativo a um objeto de mídia áudio; um componente do tipo *logoResource*, relativo a um objeto de mídia imagem e um componente composto do tipo *seq*. Dentro desse componente composto é declarado um componente do tipo *par*. Na realidade, pelo fato desse componente, do tipo *par*, não possuir restrições sobre a sua quantidade de ocorrências (*minOccurs*, *maxOccurs*), vários desses componentes podem ser instanciados. Cada componente do tipo *par*, por sua vez, contém declarados sempre dois outros componentes, um do tipo *image*, relativo a um objeto de mídia imagem e outro do tipo *text*, relativo a um objeto sintético *string* (texto).

Ainda na Figura 5.2, dentro das restrições (*constraints*), é especificada uma instância (*resource*) para o componente do tipo *logoResource*, declarado no vocabulário. Essa instância atribui o objeto de mídia *BackDrop.jpg* a esse componente. A seguir, uma instrução XSLT é especificada (<*xsl:template match="/*/*/*"*>). Essa instrução especifica que os componentes do tipo *image* e *text* devem estar contidos em outros componentes. Isso significa que as instâncias desses componentes serão retiradas do conteúdo da composição, onde esse template for referenciado, a fim de serem inseridas em outros componentes.

Ao final das restrições no template, apresentado na Figura 5.2, o componente composto do tipo *seq* é instanciado através do elemento *resource*, tendo como conteúdo instâncias dos componentes do tipo *par*. Cada componente do tipo *par* instanciado recebe, para o seu atributo *id*, um valor diferente, representando a ordem da sua criação. Dentro de cada componente do tipo *par*, são realizadas chamadas a instruções XSLT, previamente declaradas (*xsl:call-template*). Essas instruções têm como objetivo instanciar os objetos de mídia, relativos aos componentes do tipo *image* e *text*, a partir do conteúdo da composição onde esse template for referenciado. Através dessas instruções são estabelecidas as relações de inclusão das composições, relacionadas aos objetos de mídia e suas propriedades, obtidas na composição XT-XMT-O que referencia esse template.

A Figura 5.3 apresenta um documento especificado no perfil XT-XMT-O, que faz referência ao template apresentado na Figura 5.2. Nos elementos desse perfil são adicionados atributos, em relação aos elementos originais da linguagem XMT-O. Para as composições que herdam as estruturas semânticas especificadas nos templates, o atributo *xtemplate*, adicionado nesse perfil, define o endereço (URI) do template utilizado. Complementarmente, nos elementos que podem estar contidos nessas composições, o atributo *type*, também adicionado no perfil, define a referência a um tipo de componente declarado no vocabulário do template referenciado.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<XMT-O xmlns="urn:mpeg:mpeg4:xmto:schema:2002" ...>
  <head>
    <layout metrics="pixel" type="xmt/xmt-basic-layout">
      <topLayout width="400" height="300" backgroundColor="#CCCCCC ">
      </topLayout>
    </layout>
  </head>
```

```
  <body>
   <par id="propaganda" xtemplate="file:../templates/templateaudiocomlegenda.xml">
       <audio type="audio" region="audioRegion1" id="samba" src="../img/aquarela.mp3"/>
       <img type="image" id="slide1" src="../img/riodejaneiro.jpg" dur="10s">
          <transformation translation="0 16"/>
       </img>
       <img type="image" id="slide2" src="../img/saopaulo.jpg" dur="10s">
          <transformation translation="0 16"/>
       </img>
       <img type="image" id="slide3" src="../img/belohorizonte.jpg" dur="10s">
          <transformation translation="0 16"/>
        </img>
        <img type="image" id="slide4" src="../img/florianopolis.jpg"  dur="10s">
          <transformation translation="0 16"/>
        </img>
        <string type="text" id="text1" textLines="&amp;quot;Rio de Janeiro&amp;quot;">
            <material color="black" filled="true"/>
            <fontStyle  justify="MIDDLE; MIDDLE" size="18" style="BOLD"/>
             <transformation translation="0 -126"/>
        </string>
         <string type="text" id="text2" textLines="&amp;quot;Sao Paulo&amp;quot;">
            <material color="black" filled="true"/>
            <fontStyle  justify="MIDDLE; MIDDLE" size="18" style="BOLD"/>
            <transformation translation="0 -126"/>
         </string>
          <string type="text" id="text3" textLines="&amp;quot;Belo Horizonte&amp;quot;">
             <material color="black" filled="true"/>
             <fontStyle  justify="MIDDLE; MIDDLE" size="18" style="BOLD"/>
             <transformation translation="0 -126"/>
          </string>
          <string type="text" id="text4" textLines="&amp;quot;Florianopolis&amp;quot;">
             <material color="black" filled="true"/>
             <fontStyle  justify="MIDDLE; MIDDLE" size="18" style="BOLD"/>
             <transformation translation="0 -126"/>
          </string>
   </par>
  </body>
 </XMT-O>
```

Figura 5.3 – Documento especificado segundo o perfil XT-XMT-O

Na Figura 5.3 uma composição paralela herda a estrutura semântica do template apresentado na Figura 5.2. Essa composição declara um objeto de mídia áudio, quatro objetos de mídia imagem e quatro objetos sintéticos do tipo *string*. Pela semântica herdada do template, após o processamento do documento e do template referenciado pela composição, um novo documento XMT-O é obtido. Nesse novo documento, uma composição paralela contém, além do objeto de mídia áudio, um objeto de mídia imagem, instanciado pelo template, e uma composição seqüencial. Nessa composição seqüencial são definidas outras quatro composições paralelas, onde, em cada uma dessas composições, estão dispostos um objeto de mídia imagem e um objeto sintético *string*. Esses objetos, declarados na composição do documento XT-XMT-O apresentado na Figura 5.3, são agrupados de acordo com a ordem dessa declaração. Além disso, todas as

propriedades originais desses objetos são preservadas no documento obtido. A Figura 5.4 apresenta o documento XMT-O resultante do processamento.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<XMT-O xmlns="urn:mpeg:mpeg4:xmto:schema:2002" ...>
<head>
 <layout metrics="pixel" type="xmt/xmt-basic-layout">
  <topLayout backgroundColor="#CCCCCC" height="300" width="400"/>
 </layout>
</head>
<body>
 <par id="propaganda">
  <audio id="samba" src="img/aquarela.mp3"/>
  <img id="propaganda-logoResource0" src="img/BackDrop.jpg"/>
  <seq>
   <par id="par1">
    <img dur="10s" id="slide1" src="img/riodejaneiro.jpg">
     <transformation translation="0 16"/>
    </img>
    <string id="text1" textLines="&amp;quot;Rio de Janeiro&amp;quot;">
     <material color="black" filled="true"/>
     <fontStyle justify="MIDDLE; MIDDLE" size="18" style="BOLD"/>
     <transformation translation="0 -126"/>
    </string>
   </par>
   <par id="par2">
    <img dur="10s" id="slide2" src="img/saopaulo.jpg">
     <transformation translation="0 16"/>
    </img>
    <string id="text2" textLines="&amp;quot;Sao Paulo&amp;quot;">
     <material color="black" filled="true"/>
     <fontStyle justify="MIDDLE; MIDDLE" size="18" style="BOLD"/>
     <transformation translation="0 -126"/>
    </string>
   </par>
   <par id="par3">
    <img dur="10s" id="slide3" src="img/belohorizonte.jpg">
     <transformation translation="0 16"/>
    </img>
    <string id="text3" textLines="&amp;quot;Belo Horizonte&amp;quot;">
     <material color="black" filled="true"/>
     <fontStyle justify="MIDDLE; MIDDLE" size="18" style="BOLD"/>
     <transformation translation="0 -126"/>
    </string>
   </par>
   <par id="par4">
    <img dur="10s" id="slide4" src="img/florianopolis.jpg">
     <transformation translation="0 16"/>
    </img>
    <string id="text4" textLines="&amp;quot;Florianopolis&amp;quot;">
     <material color="black" filled="true"/>
     <fontStyle justify="MIDDLE; MIDDLE" size="18" style="BOLD"/>
     <transformation translation="0 -126"/>
    </string>
   </par>
  </seq>
 </par>
</body>
</XMT-O>
```

Figura 5.4 – Documento XMT-O obtido através do processador de templates

Para realizar a apresentação do documento XMT-O obtido, ele deve ser convertido para o formato BIFS. A Figura 5.5 apresenta duas visões da exibição do documento MPEG-4, correspondente ao documento XMT-O apresentado na Figura 5.4, em dois instantes distintos de tempo. No instante de tempo relativo a oito segundos do início da apresentação, a imagem inicial da composição XMT-O está sendo apresentada, com uma legenda que corresponde ao conteúdo do primeiro objeto sintético *string* declarado na composição. Posteriormente, decorridos 22 segundos do início da apresentação, a imagem apresentada corresponde à terceira imagem da composição, também com a legenda correspondente ao terceiro objeto sintético declarado na composição.

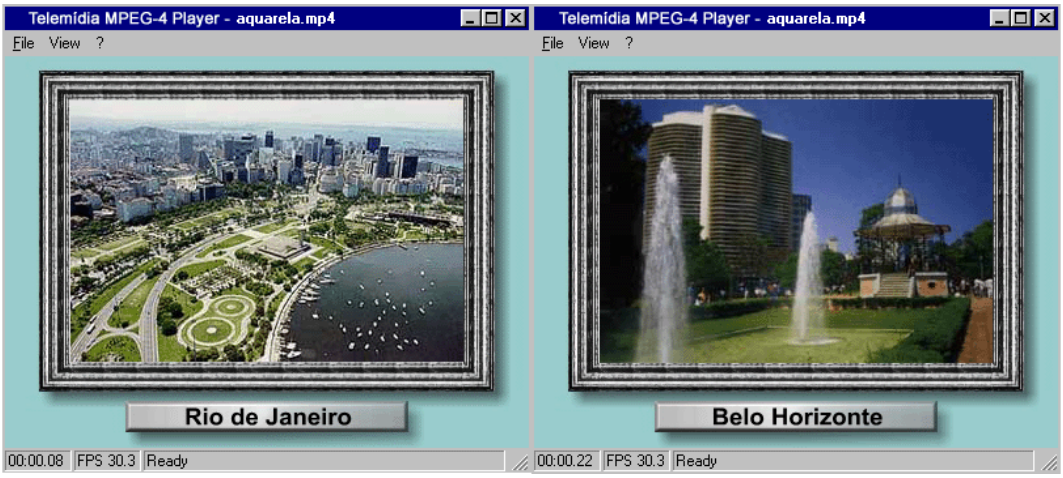

Figura 5.5 – Apresentação do documento MPEG-4 obtido através do template

Devido às semelhanças entre o perfil XT-XMT-O e o perfil XT-SMIL, o processador de templates proposto em (Silva et al., 2004a) pôde ser estendido a fim de ser aplicado também ao novo perfil.

Em (Silva et al., 2004a), a classe *TemplateProcessor* é a responsável pelo processamento de uma composição, que herda um template, gerando uma nova composição. Essa nova composição contém a estrutura semântica especificada pelo template. Nessa classe, o processamento de templates é iniciado através de uma cópia da composição original. Após essa cópia, instâncias de componentes, representadas pelo elemento *resource*, definidas sem o uso de instruções XSLT, são adicionadas à nova composição. Posteriormente, os recursos definidos através das instruções XSLT, como as relações de inclusão, são aplicados à nova composição, através de um processador XSLT (JAXP, 2003). Por fim, as restrições em XSLT são verificadas (Silva et al., 2004a).

O diagrama de classes para estender o processador de templates é apresentado nas Figura 5.6 e 5.7. A primeira apresenta as classes necessárias à instanciação desse processador para o perfil XT-XMT-O. A segunda apresenta uma instância do *framework* para compiladores XMT-O, discutido no Capítulo 3, que tem como objetivo percorrer os documentos especificados pelo perfil XT-XMT-O, gerando as cópias das composições originais.

Na Figura 5.6 são definidas duas novas classes: *XxmtoStructureParser* e *XxmtoTemplateProcessor*. Na primeira, o corpo (*body*) de um documento especificado segundo o perfil XT-XMT-O, onde se encontram definidas as composições que podem referenciar os templates, é analisado. Para isso, essa classe herda os métodos da classe *XmtoStructureParser*, especificada no *framework* para compiladores XMT-O. Nessa classe também é definida uma instância para a classe *XxmtoTemplateProcessor*, que herda as funcionalidades definidas na classe *TemplateProcessor* para o processamento de templates.

Na classe *XxmtoStructureParser*, o método *parseBody*, utilizado para analisar o corpo de um documento, chama o método *processComposite*, herdado pela classe *XxmtoTemplateProcessor*. Esse método realiza as operações necessárias ao processamento de templates, definidas em (Silva et al., 2004a), caso, no corpo do documento, existam composições que façam referências a templates.

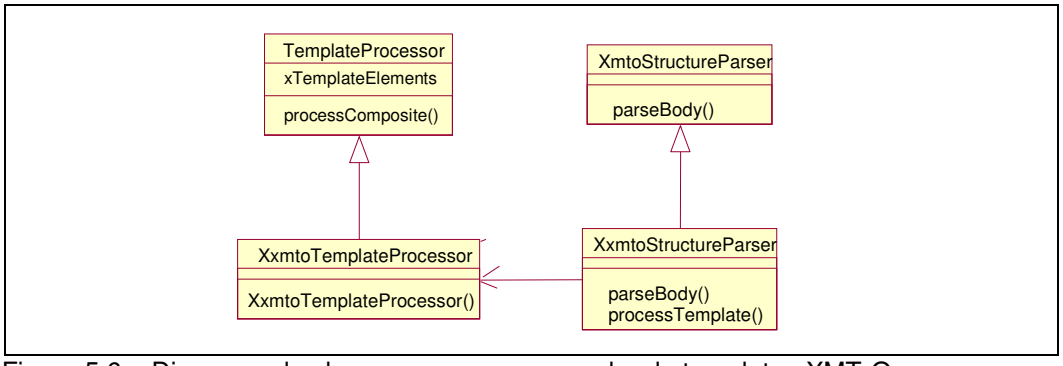

Figura 5.6 – Diagrama de classes para o processador de templates XMT-O

Na Figura 5.7 são definidas várias classes, relativas às áreas funcionais da linguagem XMT-O. Essas classes instanciam o *framework* para compiladores XMT-O, com o objetivo de realizar a tradução das especificações no documento, do perfil XT-XMT-O para a linguagem XMT-O. Na realidade, essas classes percorrem o documento XT-XMT-O, fazendo uma cópia do documento original. Nessa cópia, para as composições que fazem referência a templates, a adição de

instâncias de componentes e a análise das instruções XSLT é realizada pela classe *XxmtoTemplateProcessor*, apresentada na Figura 5.6. Para as demais composições, bem como para as outras partes do documento, como, por exemplo, o *layout*, é realizada uma simples cópia para o novo documento na linguagem XMT-O.

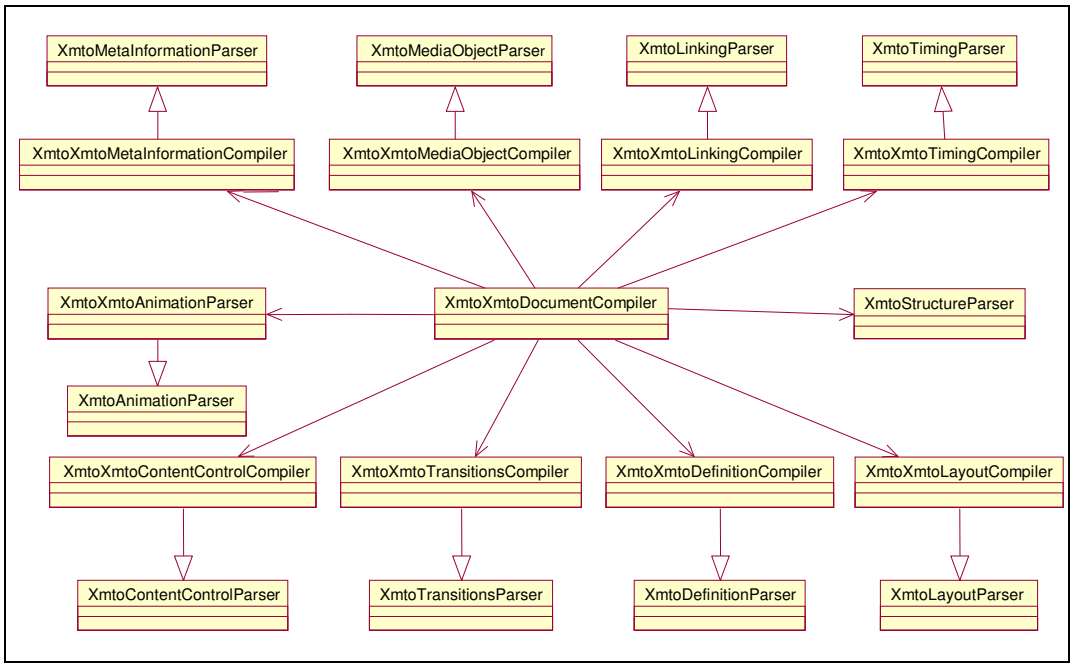

Figura 5.7 – Diagrama de classes para percorrer composições XMT-O

Por fim, é importante destacar que a proximidade entre as linguagens XMT-O e SMIL favorece a interoperabilidade entre documentos especificados nessas linguagens (Kim & Wood, 2004). No caso dos templates para composições hipermídia, essa proximidade permite que templates possam ser herdados, com algumas restrições, pelas composições definidas tanto no perfil XT-XMT-O quanto no perfil XT-SMIL. Templates interoperáveis entre esses perfis podem ser especificados utilizando elementos pertencentes a vários módulos dessas linguagens, entre os quais destacam-se:

- *BasicTimeContainers*, *ExclTimeContainers* e *BasicContentControl*. Todos esses módulos são definidos em XMT-O a partir de módulos SMIL. Esses módulos especificam as composições da linguagem XMT-O (*par*, *seq, excl* e *switch*).
- *BasicInlineTime* e *EventTiming*. Esses módulos, definidos em XMT-O a partir de módulos SMIL, especificam os atributos *begin, end* e *dur*, que podem ser qualificados através de eventos. A única exceção é o atributo *dur*, que somente pode conter valores absolutos em XMT-O.

Existem, no entanto, algumas diferenças entre os eventos que podem ser especificados nas linguagens XMT-O e SMIL. Em XMT-O, os eventos são, sobretudo, especificados através do módulo *XMT Events*, onde alguns eventos, que correspondem a extensões destinadas à interação dos objetos MPEG-4, não possuem representação em SMIL. Em (ISO/IEC, 2001) são apresentados os principais eventos de XMT-O, comparativamente aos definidos em SMIL. Entre os eventos definidos em ambas as linguagens incluem-se: clique do mouse (*click*), pressionamento do mouse (*mousedown*), mouse fora do elemento (*mouseout*), início da apresentação (*begin*) e término da apresentação (*end*). Os seguintes eventos de XMT-O não possuem representação em SMIL: despressionamento do mouse (*mouseup*), mouse sobre o elemento (*mouseover*), visibilidade do elemento (*viewable*), proximidade do elemento (*near*), colisão de elementos (*collide*).

A principal restrição do emprego de templates, especificados segundo o perfil XT-SMIL nos documentos do perfil XT-XMT-O, corresponde à definição de pontos de interface sobre os componentes do template. A linguagem XMT-O, ao contrário de SMIL, não define pontos de interface nos nós de conteúdo.

No sentido contrário, isto é, na utilização de templates especificados segundo o perfil XT-XMT-O em documentos do perfil XT-SMIL, a principal restrição é relativa à diversidade de componentes definidos pela linguagem XMT-O. Nessa linguagem, o módulo *xMedia* especifica, além dos objetos de mídia tradicionais, tais como: imagens, vídeos e áudio (*img, vídeo, audio*), definidos pelo padrão como objetos naturais, objetos sintéticos, produzidos por unidades computacionais. Na linguagem SMIL somente os objetos naturais podem ser representados.

Como exemplo de interoperabilidade, o template do perfil XT-XMT-O, apresentado na Figura 5.2, à exceção do objeto sintético denominado *string*, pode ser utilizado por documentos do perfil XT-SMIL.

6
Conclusões

Nesta dissertação foram apresentadas propostas voltadas para a integração das facilidades do padrão MPEG-4 às funcionalidades da linguagem NCL e do sistema HyperProp. Entre essas propostas foram abordadas as conversões de documentos NCL para XMT-O e vice-versa. Além da tradução entre esses formatos, também foi apresentada a incorporação de componentes, codificados segundo o MPEG-4, a documentos multimídia/hipermídia especificados na linguagem NCL. Outra proposta, relativa à autoria de documentos MPEG-4, foi o uso de templates na linguagem XMT-O, a fim de estender a semântica original das suas composições.

6.1.
Contribuições da Dissertação

As vantagens alcançadas pelas propostas desta dissertação representam as principais contribuições deste trabalho, que podem ser sintetizadas em:

- Representação de documentos hipermídia em ambos os formatos NCL e MPEG-4 (XMT-O);
- Exibição no sistema HyperProp de componentes MPEG-4 incorporados a documentos NCL;
- Definição de relacionamentos, anteriormente não permitidos, entre componentes MPEG-4 e outros objetos e composições NCL, onde se incluem outros componentes MPEG-4;
- Especificação de templates hipermídia para a linguagem XMT-O.

Para obter a representação de documentos nos formatos NCL e MPEG-4, uma instância do *framework* para compiladores NCL foi implementada. Além desse *framework*, um outro, para a conversão de documentos XMT-O foi proposto, a partir do qual duas outras instâncias foram implementadas, uma voltada à conversão de documentos XMT-O para NCL e outra à conversão dos documentos XMT-O, obtidos a partir de documentos NCL, para XMT-A.

As conversões implementadas permitem que as ferramentas existentes, particularmente aquelas voltadas para a autoria e apresentação, possam ser utilizadas nos documentos especificados nos formatos NCL e XMT-O. Para realizar a conversão entre esses formatos foi conduzido um estudo, onde perfis de interoperabilidade foram definidos.

Com o objetivo de especificar componentes MPEG-4 em documentos NCL, foi adicionado ao conjunto de ferramentas de exibição do sistema HyperProp um exibidor MPEG-4. Esse exibidor controla a apresentação desses componentes, informando ao controlador do sistema os eventos nele ocorridos. No exibidor, os componentes MPEG-4 comportam-se como objetos NCL de mídia contínua. Porém, a fim de estender a representação semântica desses componentes, foi proposta também a sua representação como composições NCL.

Para definir os relacionamentos entre os componentes MPEG-4, elos e conectores, definidos na linguagem NCL, foram utilizados, estabelecendo semânticas diversas aos relacionamentos entre esses componentes. Aplicados a um ambiente de TV digital interativa, os componentes MPEG-4 correspondem à codificação do conteúdo audiovisual existente nos canais para transmissão. Nesse ambiente e em outros mais, onde a codificação do MPEG-4 for adotada, torna-se possível estabelecer relacionamentos, anteriormente não permitidos, entre os componentes desse padrão (MPEG-4) e outros nós NCL, que podem ser outros componentes MPEG-4 ou outros tipos de objetos NCL.

Ainda com relação à autoria de documentos MPEG-4, esta dissertação apresentou como estruturas hipermídia semiprontas podem ser utilizadas no estabelecimento de novas semânticas para composições XMT-O. Essas estruturas, definidas como templates hipermídia, permitem a especificação de relações de inclusão e eventos que podem ser herdados, após o seu processamento, pelas composições XMT-O.

6.2. Trabalhos Futuros

As propostas desta dissertação oferecem a possibilidade de alguns trabalhos futuros, entre os quais têm-se os seguintes:

- Implementação da conversão direta entre os formatos MPEG-4 (XMT-O, XMT-A) e objetos do modelo de dados (modelo de apresentação) do formatador HyperProp;
- Desenvolvimento e integração ao sistema HyperProp de novas ferramentas de exibição MPEG-4;
- Representação de componentes MPEG-4 através de nós atômicos especificados em documentos hipermídia;
- Especificação de conectores hipermídia aplicados à linguagem XMT-O;
- Implementação de uma estrutura para o armazenamento integrado dos objetos de mídia em conjunto com a especificação dos documentos NCL;
- Desenvolvimento de uma arquitetura distribuída para os módulos do formatador no sistema HyperProp.

Conversão MPEG-4 no modelo de apresentação

Os conversores apresentados nesta dissertação possuem uma estrutura modular, idealizada para a utilização em uma arquitetura em camadas, onde as conversões são processadas seqüencialmente. Embora essa arquitetura seja flexível, permitindo diversas organizações dos módulos, o que favorece sua aplicação a vários projetos, em termos de eficiência essa abordagem pode não ser satisfatória. Para otimizar as tarefas de conversão e ao mesmo tempo integrá-las ao sistema HyperProp, é necessário que os documentos MPEG-4, em qualquer dos três formatos utilizados para a sua especificação, possam ser diretamente convertidos para os objetos do modelo de apresentação do formatador do sistema, representados internamente ao formatador desse sistema e vice-versa.

Novas ferramentas de exibição MPEG-4 integradas ao HyperProp

Nesta dissertação uma ferramenta de exibição para componentes MPEG-4 foi adicionada ao sistema HyperProp. No entanto, essa ferramenta limita os tipos de eventos informados ao formatador. Como exemplo, foi apresentada nesta dissertação uma ferramenta capaz de reportar eventos internos aos componentes MPEG-4, especificados através dos sensores e rotas. Esse tipo de evento não tem suporte na ferramenta atualmente integrada ao sistema. Além desses eventos, outros poderiam ser informados, como, por exemplo, eventos de transição

estabelecidos pelas relações temporais no modelo *FlexTime*. Para abranger todos esses eventos, novos adaptadores devem ser desenvolvidos para exibidores MPEG-4.

Representação de componentes MPEG-4 através de nós atômicos especificados em documentos hipermídia

A codificação linear do conteúdo audiovisual, adotada nos padrões MPEG 1 e 2, gera objetos de mídia que, normalmente, são representados em documentos multimídia/hipermídia através de nós atômicos. Por outro lado, no padrão MPEG-4, ao contrário dos padrões MPEG anteriores, a codificação é baseada em objetos individuais, cujas propriedades e relacionamentos são especificados internamente ao componente MPEG-4, através de um formato próprio. Dessa forma, conforme apresentado no Capítulo 4, a especificação dos componentes MPEG-4 em documentos multimídia/hipermídia pode ser realizada através de nós de composição.

A representação dos componentes MPEG-4 utilizando nós de composição necessita que os objetos internos a essas composições (objetos definidos na cena MPEG-4) sejam especificados no documento, além disso, devem ser especificados os possíveis mapeamentos entre os eventos definidos internamente a essa composição (sensores) e os eventos externos, definidos no documento multimídia/hipermídia.

Uma solução mais abrangente poderia unificar a representação do conteúdo audiovisual, gerado pelos padrões MPEG, nos documentos multimídia/hipermídia. Nessa solução os objetos de mídia codificados segundo o padrão MPEG-4 também poderiam estar representados através de nós atômicos, facilitando a tarefa de especificação dos autores, pois os objetos internos à cena MPEG-4, bem como o mapeamento dos eventos internos à cena com os eventos especificados no documento, não necessitariam estar especificados no documento multimídia/hipermídia. No entanto, essa solução necessita que sejam definidos, na linguagem de autoria dos documentos multimídia/hipermídia, um conjunto de âncoras, relativas aos nós atômicos MPEG-4, capazes de representar o universo dos eventos que possam ocorrer internamente às cenas MPEG-4.

Novos perfis XMT-O com conectores e templates

Templates aplicados à linguagem XMT-O facilitam a autoria de documentos MPEG-4, através da construção de estruturas semânticas formadas por relações de inclusão e eventos, reusáveis por diferentes composições. Além dos templates, conectores hipermídia podem ser empregados a fim de aumentar a capacidade de autoria da linguagem XMT-O. Através do uso de conectores, relacionamentos podem ser estabelecidos por elos que, ao contrário daqueles definidos originalmente em XMT-O, podem associar tanto objetos externos quanto internos à cena audiovisual. Para a inclusão de conectores, um novo perfil de XMT-O deve ser especificado, aqui denominado XC-XMT-O, através da adição dos módulos *XConnector* e *Linking* de NCL, aos módulos de XMT-O. Além desse perfil, um outro pode ser definido, a fim de permitir o uso simultâneo de conectores e templates, aqui denominado X-XMT-O. Esse perfil deve conter, além dos módulos *XConnector* e *Linking,* o módulo *XTemplate*.

Armazenamento integrado da estrutura e conteúdo de um documento

Em relação ao armazenamento, no MPEG-4 o conteúdo e as especificações dos documentos podem ser armazenados e distribuídos de forma integrada, o que facilita a obtenção do sincronismo entre as diversas mídias de uma cena ou programa. A fim de utilizar essa estrutura integrada, documentos NCL podem ser convertidos para MPEG-4, porém, nesse caso, esses documentos deverão ser apresentados em exibidores MPEG-4. Para prover o armazenamento integrado de documentos NCL, e ao mesmo tempo, permitir que o formatador do sistema HyperProp realize a apresentação, é necessário definir uma estrutura de dados específica para o formato NCL, que contenha a especificação do documento e o conteúdo dos objetos de mídia nele referenciados; uma espécie de BIFS-NCL.

Formatador HyperProp distribuído

Nos exibidores MPEG-4 não são implementados métodos que tenham por objetivo garantir a qualidade das apresentações, de acordo com a sincronização entre objetos especificadas pelo usuário, a não ser a simples linearização em *timeline*. Nesse padrão, as especificações em BIFS, por princípio temporalmente lineares, impedem que, durante a apresentação, as especificações originais dos

autores sejam conhecidas, premissa básica para garantir a correta execução dos relacionamentos nos documentos multimídia/hipermídia.

No formatador do sistema HyperProp, ao contrário dos exibidores MPEG-4, o recebimento da especificação dos relacionamentos temporais de um documento é a etapa inicial para a apresentação. A partir dessa especificação, o formatador define um plano de execução, criando cadeias temporais baseadas nos relacionamentos entre os conteúdos da apresentação (Rodrigues, 2003). As cadeias temporais podem ser representadas por grafos, onde seus nós correspondem ao conteúdo da apresentação e suas arestas às relações previsíveis entre os nós. Essas cadeias preservam as dependências entre os conteúdos da apresentação, permitindo que ajustes possam ser realizados de acordo com as intenções do autor. O conteúdo dos documentos é obtido através de requisições realizadas pelo formatador ao servidor de armazenamento, no tempo previsto para a exibição dos objetos de uma apresentação qualquer.

Nos sistemas de transmissão de TV digital interativa, a comunicação é, usualmente, assimétrica. Na utilização do canal de comunicação prevalece a transmissão do servidor de difusão para os aparelhos receptores. Nesse cenário, cada apresentação é transmitida a um grande número de usuários (clientes) sem requisições individuais. Os conteúdos chegam temporalmente sincronizados (*timeline*) à medida que é necessária sua exibição. Na maioria dos sistemas atuais, existe uma limitação da interatividade entre os ambientes de execução e armazenamento. Na realidade, as requisições por parte do ambiente de execução se restringem a aspectos relacionados à interatividade local (informações já enviadas por difusão) e à personalização da apresentação (não da informação), ainda assim, quando permitidos pelo autor do programa.

Uma solução mais abrangente poderia unir as vantagens dos dois cenários descritos nos dois parágrafos anteriores, levando ainda em conta a possibilidade da integração, em um único fluxo, como discutido no item anterior, da especificação dos relacionamentos de um documento e o conteúdo de seus objetos. Um fluxo, nesse caso, integraria uma cadeia temporal aos dados dos objetos que relaciona. A concatenação dos vários fluxos, correspondentes às várias cadeias temporais de um documento, permitiria a exibição completa do mesmo. Um fluxo seria requisitado pelo controlador da exibição (formatador), mas não haveria necessidade de haver uma requisição para cada objeto do fluxo,

como hoje acontece no sistema HyperProp. Mais ainda, o conteúdo dos objetos transportados pelo fluxo poderia ser recebido à medida que fosse necessária sua exibição. No contexto do sistema HyperProp, essa solução requer uma distribuição dos módulos do seu formatador, com o objetivo de compartilhar suas tarefas entre os ambientes.

Em um formatador distribuído, adequado a esse novo cenário, a construção das cadeias temporais poderia ser realizada no ambiente de armazenamento para posterior transmissão ao ambiente de execução, integrada ao conteúdo dos seus objetos de mídia, através de uma estrutura em *timeline* contendo os seus instantes de exibição. Essa estrutura poderia, inclusive, ser gerada em tempo real (transmissões ao vivo), por módulos do formatador alocados no ambiente de armazenamento.

É importante mencionar que a definição de uma arquitetura distribuída para os módulos do formatador hipermídia pode também ser aplicada a outros cenários, que não envolvam, necessariamente, a comunicação entre os ambientes de armazenamento e execução. Como exemplo, considere o cenário colaborativo, caracterizado pela distribuição das ferramentas de exibição do formatador em múltiplos usuários. Individualmente, as interações de um usuário constituem eventos imprevisíveis, que levam o formatador a alterar a cadeia temporal hipermídia principal de um usuário específico. No entanto, nesse cenário colaborativo, as alterações em uma apresentação devem ser atualizadas para todos que a estejam compartilhando, como no caso de aplicações multimídia/hipermídia que utilizem grupos de colaboração, onde se incluem aplicações de teleconferência, ensino à distância, jogos, entre outras. De uma forma mais geral, pode-se então pensar em um formatador distribuído com parte de seus módulos nos vários servidores (ambiente de armazenamento), outra parte de seus módulos (por exemplo as ferramentas de exibição) espalhados entre várias plataformas de exibição e, ainda, com a possibilidade de outros módulos em algum ponto central do controle da exibição.

7
Referências

ALPHAWORKS Discussion List for MPEG-4 Toolkit. List maintained by the IBM AlphaWorks. Disponível em http://www.alphaworks.ibm.com/tech/tk4mpeg4. Acesso em: 20 jan. 2005.

BACHELET, B.; MAHEY, P.; RODRIGUES, R. F.; SOARES, L. F. G. **Elastic Time Computation for Hypermedia Documents**. In: Simpósio Brasileiro em Sistemas Multimídia e Hipermídia – SBMídia 2000, Natal. Anais do VI Simpósio Brasileiro em Sistemas Multimídia e Hipermídia, 2000. p. 47-62.

BATISTA, T. V. **Controle de Versões no Modelo Hipermídia de Contextos Aninhados**. 1994. 120 p. Dissertação (Mestrado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, 1994. Disponível em ftp://ftp.telemidia.puc-rio.br/pub/docs/theses/1994_03_batista.pdf. Acesso em: 20 jan. 2005.

BERNERS-LEE, T.; CAILLIAU, R.; LUOTONEN, A.; NIELSEN, H. F.; SECRET, A. **The World-Wide Web**. Communications of the ACM, v. 37, n. 8, p. 76-82, Agosto de 1994.

BOUILHAGUET, F.; DUFOURD, J. C.; BOUGHOUFALAH, S.; HAVET, C. **Interactive Broadcast Digital Television – The OpenTV Plataform versus the MPEG-4 Standard Framework**. In: IEEE International Symposium on Circuits and Systems – ISCAS 2000, Geneva, Suiça. Procedings of the International Symposium on Circuits and Systems, 2000, p. 1312-1315.

BOUGHOUFALAH, S.; DUFOURD, J. C.; BOUILHAGUET, F. **MPEG-Pro, an Authoring System for MPEG-4 with Temporal Constraints and Template Guide Editing**. In: IEEE International Conference on Multimedia and Expo – ICME 2000, Nova Iorque, Estados Unidos. Procedings of the International Conference on Multimedia and Expo, 2000, v. 1, p.175-178.

BOUGHOUFALAH, S.; BRELOT, M.; BOUILHAGUET, F.; DUFOURD, J. C. **A Template-Guide Authoring Environment to Produce MPEG-4 Content for the Web**. International Conference on Media Futures - ICMF, Florença, Itália, 2001.

BULTERMAN, D.; RUTLEDGE, L. **SMIL 2.0: Interactive Multimedia for Web and Mobile Devices**. 1 ed. Springer, 2004. 439 p.

BUSH, V. **As We May Think**. The Atlantic Monthly, Julho de 1945. Disponível em http://www.w3.org/History/1945/vbush. Acesso em: 20 jan. 2005.

CHIARIGLIONE, L. **Short MPEG-1 Description**. ISO/IEC – International Organisation for Standardisation – ISO/IEC JTC1/SC29/WG11 – NMPEG96 – Coding of Moving Pictures and Audio, Junho de 1996. Disponível em http://www.chiariglione.org/mpeg/standards/mpeg-1/mpeg-1.htm. Acesso em: 20 jan. 2005.

CHIARIGLIONE, L. **Short MPEG-2 Description**. ISO/IEC – International Organisation for Standardisation – ISO/IEC JTC1/SC29/WG11 – NMPEG00 – Coding of Moving Pictures and Audio, Outubro de 2000. Disponível em http://www.chiariglione.org/mpeg/standards/mpeg-2/mpeg-2.htm. Acesso em: 20 jan. 2005.

COELHO, R. M. **Integração de Ferramentas Gráficas e Declarativas na Autoria de Arquiteturas Modeladas através de Grafos Compostos**. 2004. 106 p. Dissertação (Mestrado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, 2004. Disponível em ftp://ftp.telemidia.puc-rio.br/pub/docs/theses/2004_08_coelho.pdf. Acesso em: 20 jan. 2005.

CONCOLATO, C.; FEUVRE, J. **MPEG-4 BIFS and XMT Tutorial**. Tutorial sobre o padrão MPEG-4 – Telecom Paris, 2003. Disponível em http://gpac.sourceforge.net/tutorial/bifs_intro.htm. Acesso em: 20 jan. 2005.

COSTA, F. R. **Um Editor Gráfico para Definição e Exibição do Sincronismo de Documentos Multimídia/Hipermídia**. 1996. 106 p. Dissertação (Mestrado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, 1996. Disponível em ftp://ftp.telemidia.puc-rio.br/pub/docs/theses/1996_08_costa.pdf. Acesso em: 20 jan. 2005.

COSTA, R. M. R.; RODRIGUES, R. F.; SOARES, L. F. G. **Armazenamento e Distribuição de Documentos NCL Através do Padrão MPEG-4**. In: II Workshop de Ferramentas e Demonstrações do X Simpósio Brasileiro de Sistemas Multimídia e Web, Ribeirão Preto, São Paulo. Anais do WebMedia & LA-Web, 2004. p. 47-62.

GORINI, R. A. C. **Um Ambiente de Suporte à Autoria Cooperativa de Documentos Hipermídia.** 2001. 132 p. Dissertação (Mestrado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, 2001. Disponível em ftp://ftp.telemidia.puc-rio.br/pub/docs/theses/2001_09_rgorini.pdf. Acesso em: 20 jan. 2005.

GPAC Project on Advanced Content. **MP4Box – MPEG-4 File Conversion**. http://gpac.sourceforge.net/auth_mp4box.php. 2000.

GPAC Project on Advanced Content. **OSMOSE Player for MPEG-4 – OSMO 4**. http://www.comelec.enst.fr/osmo4/. 2002.

HERPEL, C. **Elementary Stream Management in MPEG-4**. In: IEEE Transactions on Circuits and Systems for Video Technology, v.9, n. 2, p. 315-324. 1999.

HERPEL, C.; ELEFTHERIADIS, A. **MPEG-4 Systems: Elementary Stream Management**. Special Issue on MPEG-4 Signal Processing: Image Communication, v. 15, p. 299-320, Elservier, 2000.

ISO/IEC International Organisation for Standardisation. **14772-1:1997. The Virtual Reality Modeling Language – VRML97**. 1997.

ISO/IEC International Organisation for Standardisation. **13818-1:2000. Generic coding of moving pictures and associated audio information – Parte 1: Systems**. 2000.

ISO/IEC International Organisation for Standardisation. **14496-5:2000. Coding of Audio-Visual Objects – Part 5: Reference Software.** 2º Edition, 2001.

Disponível em http://www.iso.ch/iso/en/ittf/PubliclyAvailableStandards/. Acesso em: 20 jan. 2005.

ISO/IEC International Organisation for Standardisation. **14496-6:2000. Coding of Audio-Visual Objects – Part 6: Delivery Multimedia Integration Framework (DMIF).** 2º Edition, 2000.

ISO/IEC International Organisation for Standardisation. **14496-1:2001. Coding of Audio-Visual Objects – Part 1: Systems.** 2º Edition, 2001.

ISO/IEC International Organisation for Standardisation. **14496-12:2003. Coding of Audio-Visual Objects – Part 12: ISO Base Media File Format.** 2003.

ISO/IEC International Organisation for Standardisation. **14496-11:2004. Coding of Audio-Visual Objects – Part 11: Scene description and application engine/Amd 4**. **XMT & MPEG-J extensions**. 2004.

ITU-T International Telecomunication Union – Telecommunication Standardization Sector. **H.264 Advanced Video Coding for Generic Audiovisual Services**. 2004.

JAXP **Java API for XML Processing**. http://java.sun.com/xml/jaxp. 2003.

JEONG, T.; HAM, J.; KIM, S. **A Pre-scheduling Mechanism for Multimedia Presentation Synchronization**. In: IEEE International Conference on Multimedia Computing and Systems – ICMCS 1997, Ottawa, Canadá. Procedings of the International Conference on Multimedia Computing and Systems, 1997. p. 379-386.

JOUNG, J.; KIM, K. **An XMT API for Generation of the MPEG-4 Scene Description**. In: International Multimedia Conference, Juan-les-Pins, França. Procedings of 10 ACM International Conference on Multimedia, 2002. p. 307-310.

KIM, M.; WOOD, S.; CHEOK, L. **Extensible MPEG-4 textual format (XMT)**. In: International Multimedia Conference, Los Angeles, Estados Unidos. Proccedings of the 2000 ACM Workshop on Multimedia, 2000. p. 71-74.

KIM, M.; WOOD, S. **MPEG-4 Flexible Timing Standard (FlexTime)**. Overview of FlexTime. IBM Research. 2002. Disponível em http://www.research.ibm.com/mpeg4/Projects/Flextime.htm. Acesso em: 20 jan. 2005.

KIM, M.; WOOD, S. **XMT: MPEG-4 Textual Format for Cross-Standard Interoperability**. Overview of XMT. IBM Research. 2004. Disponível em http://www.research.ibm.com/mpeg4/Projects/XMTInterop.htm. Acesso em: 20 jan. 2005.

KOENEN, R. **MPEG-4 Overview (V21 – Jeju Version).** ISO/IEC – International Organisation for Standardisation – ISO/IEC JTC1/SC29/WG11 – N4668 – Coding of Moving Pictures and Audio. Março de 2002. Disponível em http://www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm. Acesso em: 20 jan. 2005.

LIM, Y.; SINGER, D. **MIME Type Registration for MPEG-4**. Audio-Video Transport Working Group. Internet Draft. 2004. Disponível em http://community.roxen.com/developers/idocs/drafts/draft-lim-mpeg4-mime02.html. Acesso em: 20 jan. 2005.

MARTINEZ, M. J. **MPEG-7 Overview.** ISO/IEC – International Organisation for Standardisation – ISO/IEC JTC1/SC29/WG11 – N5525 – Coding of Moving Pictures and Audio. Março de 2003. Disponível em http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm. Acesso em: 20 jan. 2005.

MOURA, M. S. **Relações Espaciais em Documentos Hipermídia**. 2001. 153 p. Dissertação (Mestrado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, 2001. Disponível em ftp://ftp.telemidia.puc-rio.br/pub/docs/theses/2001_08_moura.pdf. Acesso em: 20 jan. 2005.

MUCHALUAT-SAADE, D. C. **Relações em Linguagens de Autoria Hipermídia: Aumentando Reuso e Expressividade**. 2003. 251 p. Tese (Doutorado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, 2003. Disponível em ftp://ftp.telemidia.puc-rio.br/pub/docs/theses/2003_03_muchaluat.pdf. Acesso em: 20 jan. 2005.

PEREIRA, F.; EBRAHIMI, T. **The MPEG-4 Book**. 1 ed. Prentice Hall PTR, 2002. 849 p.

PINTO, L. A. F. **Autoria Gráfica de Estruturas de Documentos Hipermídia no Sistema HyperProp**. 2000. 118 p. Dissertação (Mestrado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, 2000. Disponível em ftp://ftp.telemidia.puc-rio.br/pub/docs/theses/2000_08_pinto.pdf. Acesso em: 20 jan. 2005.

POSTEL, J.; REYNOLDS, J. **File Transfer Protocol (FTP)**. Request for Comments – RFC – 959. Outubro de 1985. Disponível em http://www.faqs.org/rfcs/rfc959.html. Acesso em: 20 jan. 2005.

RODRIGUES, R. F.; RODRIGUES, L. M.; SOARES, L. F. G. **Desenvolvimento e Integração de Ferramentas de Exibição em Sistemas de Apresentação Hipermídia**. In: Simpósio Brasileiro em Sistemas Multimídia e Hipermídia – SBMídia 2001, Florianópolis. Anais do VII Simpósio Brasileiro em Sistemas Multimídia e Hipermídia, 2001. p. 70-88.

RODRIGUES, L. M.; ANTONACCI, M. J.; RODRIGUES, R. F.; MUCHALUAT-SAADE, D. C.; SOARES, L. F. G. **Improving SMIL with NCM Facilities**. Journal of Multimedia Tools and Applications, v. 16, n. 1, p. 29-54, Kluwer Academics. 2002.

RODRIGUES, R. F. **Formatação e Controle de Apresentações Hipermídia com Mecanismos de Adaptação Temporal**. 2003. 170 p. Tese (Doutorado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, 2003. Disponível em ftp://ftp.telemidia.puc-rio.br/pub/docs/theses/2003_03_rodrigues.pdf. Acesso em: 20 jan. 2005.

SILVA, H. V. O.; RODRIGUES, R. F.; SOARES, L. F. G. **SMIL+XTemplate**. In: X Simpósio Brasileiro de Sistemas Multimídia e Web, Ribeirão Preto, São Paulo. Anais do WebMedia & LA-Web, 2004. p. 79-86.

SILVA, H. V. O.; MUCHALUAT-SAADE, D. C.; RODRIGUES, R. F.; SOARES, L. F. G. **NCL 2.0: Integrating New Concepts to XML Modular Languages**. In: The ACM Symposium on Document Engineering – DocEng'04, Milwaukee, Estados Unidos. Procedings of the 2004 ACM Symposium on Document Enginnering, 2004. p. 188-197.

SILVA, H. V. O. **X-SMIL: Aumentando o Reuso e Expressividade em Linguagens de Autoria Hipermídia**. 2005. 185 p. Dissertação (Mestrado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, 2005.

SOARES, L. F. G.; RODRIGUES, R. F.; MUCHALUAT-SAADE, D. C. **Modelo de Contextos Aninhados – versão 3.0**. Relatório Técnico, Laboratório Telemídia, PUC-Rio. Março de 2003.

WEB3D Consortium. **The Virtual Reality Modeling Language – VRML 2.0 Specification**. WEB3D Recommendation. Agosto de 1996. Disponível em http://www.web3d.org/VRML2.0/FINAL/. Acesso em: 20 jan. 2005.

W3C World-Wide Web Consortium. **Cascading Style Sheets, level 2 – CSS**. W3C Recommendation. Maio de 1998. Disponível em http://www.w3.org/TR/CSS2/. Acesso em: 20 jan. 2005.

W3C World-Wide Web Consortium. **Document Object Model – DOM Level 1 Specification**. W3C Recommendation. Outubro de 1998. Disponível em http://www.w3.org/TR/RECDOM-Level-1/. Acesso em: 20 jan. 2005.

W3C World-Wide Web Consortium. **HyperText Markup Language – HTML 4.01**. W3C Recommendation. Dezembro de 1999. Disponível em http://www.w3.org/TR/html4. Acesso em: 20 jan. 2005.

W3C World-Wide Web Consortium. **Extensible Markup Language – XML 1.0 – Second Edition**. W3C Recommendation. Outubro de 2000. Disponível em http://www.w3.org/TR/REC-xml. Acesso em: 20 jan. 2005.

W3C World-Wide Web Consortium. **XML Linking Language – XLink 1.0**. W3C Recommendation. Junho de 2001. Disponível em http://www.w3.org/TR/xlink/. Acesso em: 20 jan. 2005.

W3C World-Wide Web Consortium. **XML Schema Part 1: Structures Second Edition**. W3C Recommendation. Outubro de 2004. Disponível em http://www.w3.org/TR/xmlschema-1/. Acesso em: 20 jan. 2005.

W3C World-Wide Web Consortium. **Synchronized Multimedia Integration Language – SMIL 2.0 Specification**. W3C Recommendation. Janeiro de 2005. Disponível em http://www.w3.org/TR/smil. Acesso em: 20 jan. 2005.

W3C World-Wide Web Consortium. **XML Path language – XPath 2.0**. W3C Recommendation. Fevereiro de 2005. Disponível em http://www.w3.org/TR/xpath. Acesso em: 20 fev. 2005.

W3C World-Wide Web Consortium. **XSL Transformations – XSLT 2.0**. W3C Recommendation. Fevereiro de 2005. Disponível em http://www.w3.org/TR/xslt20. Acesso em: 20 fev. 2005.

8
Apêndice A

Este apêndice descreve os elementos e atributos das áreas funcionais de XMT-O, definidas no Capítulo 2. Os símbolos nas tabelas relativas a cada área funcional possuem os seguintes significados: "?" opcional, "|" ou, "*" zero ou mais ocorrências e "+" uma ou mais ocorrências. Nessas tabelas, os nomes iniciados com letras maiúsculas correspondem a grupos de elementos ou atributos. Ao final deste apêndice, esses grupos são apresentados.

8.1.
Áreas Funcionais Timing e Time Manipulations

Elementos	Atributos	Conteúdo
par	CoreAttrs,	(ScheduleGroup | MediaContentGroup | ContentControlGroup | a | AnimationGroup | transitionFilter)*
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	SystemTestAttrs, customTest,	
	region,	
	MediaAnnotateAttrs	
seq	CoreAttrs,	(ScheduleGroup | MediaContentGroup | ContentControlGroup | a | AnimationGroup | transitionFilter)*
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	SystemTestAttrs, customTest,	
	region,	
	MediaAnnotateAttrs,	
excl	CoreAttrs,	(ScheduleGroup | MediaContentGroup | ContentControlGroup | a | AnimationGroup | transitionFilter)*
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	SystemTestAttrs, customTest,	
	region,	
	MediaAnnotateAttrs	

8.2.
Área Funcional *Animation*

Elementos	Atributos	Conteúdo
animate	CoreAttrs,	vazio
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs	

	skip-content,	
	attributeName,attributeType, targetElement, values, calcMode, accumulate, additive, from, to, by,	
	?calcMode=”spline” (keyTimes, keySplines)	
set	CoreAttrs,	vazio
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	skip-content,	
	attributeName, attributeType, targetElement, to,	
animateMotion	CoreAttrs,	vazio
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	skip-content,	
	targetElement, values, calcMode, accumulate, additive, from, to, by, origin,	
	?calcMode=”spline” (keyTimes, keySplines)	
animateColor	CoreAttrs,	vazio
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	skip-content,	
	attributeName, attributeType, targetElement, values, calcMode, accumulate, additive, from, to, by,	
	?calcMode=”spline” (keyTimes, keySplines)	
dragPlane	CoreAttrs,	vazio
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	skip-content,	
	maxPosition, minPosition, autooffset, offset,	
dragDisc	CoreAttrs,	vazio
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	skip-content,	
	minAngle, maxAngle, autooffset, offset	
dragCylinder	CoreAttrs,	vazio
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	skip-content,	
	diskAngle, minAngle, maxAngle, autooffset, offset	
dragSphere	CoreAttrs,	vazio
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	skip-content,	
	autooffset, offset	

8.3.
Área Funcional *Content Control*

Elementos	Atributos	Conteúdo
switch	CoreAttrs,	(layout | ScheduleGroup | MediaContentGroup | ContentControlGroup | a | AnimationGroup | transitionFilter)*
	SystemTestAttr,	
	customTest	
prefetch	CoreAttrs,	vazio
	MediaClippingAttrs,	
	skip-content,	
	src, mediaSize, mediaTime, bandwidth	
customAttributes	CoreAttrs,	(customTest)*
	skip-content,	
customTest	CoreAttrs,	vazio
	skip-content,	
	title, defaultState, override, uid	

8.4.
Área Funcional *Layout*

Elementos	Atributos	Conteúdo
layout	CoreAttrs,	toplayout
	skip-content,	
	type, metrics	
toplayout	CoreAttrs ,	(region)*
	skip-content ,	
	backgroundColor, height, width, enableSubtitles	
region	CoreAttrs,	(region)*
	skip-content,	
	backgroundColor, regionName, translation, size, z-index, soundLevel	

8.5.
Área Funcional *Linking*

Elementos	Atributos	Conteúdo
a	CoreAttrs,	(ScheduleGroup | MediaContentGroup | ContentControlGroup | AnimationGroup)*
	SystemTestAttrs,	
	customTest,	
	href, actuate	

8.6.
Área Funcional *Media Objects*

Elementos	Atributos	Conteúdo
retangle	CoreAttrs,	(switch | use | transitionFilter | AnimationGroup | material | texture | transformation | hotspots)*
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	SystemTestAttrs, TermCapTestAttrs, customTest,	
	region,	
	erase, sensitivity,	
	TransInOutAttrs,	
	MediaAccessAttrs, MediaAnnotateAttrs,	
	size	
circle	CoreAttrs,	(switch | use | transitionFilter | AnimationGroup | material | texture | transformation | hotspots)*
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	SystemTestAttrs, TermCapTestAttrs, customTest,	
	region,	
	erase, sensitivity,	
	TransInOutAttrs,	
	MediaAccessAttrs, MediaAnnotateAttrs,	
	radius	
points	CoreAttrs,	(switch | use | transitionFilter | AnimationGroup | material | texture | transformation | hotspots)*
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	SystemTestAttrs, TermCapTestAttrs, customTest,	
	region,	
	erase, sensitivity,	
	TransInOutAttrs,	
	MediaAccessAttrs, MediaAnnotateAttrs,	
	color, coord	
lines	CoreAttrs,	(switch | use | transitionFilter | AnimationGroup | material | texture | transformation | hotspots)*
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	SystemTestAttrs, TermCapTestAttrs, customTest,	
	region,	
	erase, sensitivity,	
	TransInOutAttrs,	
	MediaAccessAttrs, MediaAnnotateAttrs,	
	color, coord, colorIndex, colorPerVertex, coordIndex	
polygons	CoreAttrs,	(switch | use | transitionFilter |
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	

	SystemTestAttrs, TermCapTestAttrs, customTest,	AnimationGroup | material | texture | transformation | hotspots)*
	region,	
	erase, sensitivity,	
	TransInOutAttrs,	
	MediaAccessAttrs, MediaAnnotateAttrs,	
	color, coord, colorIndex, colorPerVertex, coordIndex, texCoord, texCoordIndex	
curve	CoreAttrs,	(switch | use | transitionFilter | AnimationGroup | material | texture | transformation | hotspots)*
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	SystemTestAttrs, TermCapTestAttrs, customTest,	
	region,	
	erase, sensitivity,	
	TransInOutAttrs,	
	MediaAccessAttrs, MediaAnnotateAttrs,	
	points, fineness, type	
material	StructureModuleAttrs,	(AnimationGroup | switch)*, (outline | use)*, (AnimationGroup | (switchGroup)*
	ambientInsensity, diffuseColor, color, shininess, specularColor, transparency, filled	
texture	StructureModuleAttrs,	(AnimationGroup | switch)*
	src, repeatS, repeatT, center, rotation, scale, translation	
transformation	StructureModuleAttrs[1],	(AnimationGroup | switch)*
	center, rotation, scale, scaleOrientation, translation, billboardAxis, order, visibility, billboard	
hotspots	StructureModuleAttrs	(xMediaObjectsGroup | switch | a | group | xmtaMedia | use | useMacro)*
outline	StructureModuleAttrs,	(AnimationGroup | switch)*
	color, width, style	
img	CoreAttrs,	(switch | use | transitionFilter | AnimationGroup | material | chromakey | transformation | hotspots)*
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	SystemTestAttrs, TermCapTestAttrs, customTest,	
	region,	
	erase, sensitivity,	
	TransInOutAttrs,	
	MediaAccessAttrs, MediaAnnotateAttrs, MediaClippingAttrs,	
	mediaRepeat, type , src	
chromakey	StructureModuleAttrs,	(AnimationGroup | switch)*
	isKeyedr, isRGB, keyColor, lowThreshold, highThreshold, transparency	

light	StructureModuleAttrs, ambientIntensity, color, direction, intensity, on	(AnimationGroup | switch)*
video	CoreAttrs, TimingAttrs, TimeManipAttrs, FlexTimeAttrs, SystemTestAttrs, TermCapTestAttrs, customTest, region, erase, sensitivity, TransInOutAttrs, MediaAccessAttrs, MediaAnnotateAttrs, MediaClippingAttrs, mediaRepeat, type , src	(switch | use | transitionFilter | AnimationGroup | material | chromakey | transformation | hotspots)*
text	CoreAttrs, TimingAttrs, TimeManipAttrs, FlexTimeAttrs, SystemTestAttrs, TermCapTestAttrs, customTest, region, erase, sensitivity, TransInOutAttrs, MediaAccessAttrs, MediaAnnotateAttrs, MediaClippingAttrs, mediaRepeat, type , src	(switch | use | transitionFilter | AnimationGroup | material | texture | transformation | hotspots | fontstyle)*
string	CoreAttrs, TimingAttrs, TimeManipAttrs, FlexTimeAttrs, SystemTestAttrs, TermCapTestAttrs, customTest, region, erase, sensitivity, TransInOutAttrs, MediaAccessAttrs, MediaAnnotateAttrs, textLines	(switch | use | transitionFilter | AnimationGroup | material | texture | transformation | hotspots | fontstyle)*
subtitles	CoreAttrs, TimingAttrs, TimeManipAttrs, FlexTimeAttrs, SystemTestAttrs, TermCapTestAttrs, customTest, region, erase, sensitivity, TransInOutAttrs, MediaAccessAttrs, MediaAnnotateAttrs, textLines	(switch | use | transitionFilter | AnimationGroup | material | texture | transformation | hotspots | fontstyle)*
audio	CoreAttrs, TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	(switch | use | transitionFilter | AnimationGroup | transformation | hotspots |

	SystemTestAttrs, TermCapTestAttrs, customTest,	sound)*
	region,	
	erase, sensitivity,	
	TransInOutAttrs,	
	MediaAccessAttrs, MediaAnnotateAttrs, MediaClippingAttrs,	
	mediaRepeat, type , src	
sound	StructureModuleAttrs,	(AnimationGroup | switch)*
	location, intensity, spatialize, direction, maxBack, maxFront, minBack, minFront, priority	
audioClip	CoreAttrs,	(switch | use | transitionFilter | AnimationGroup | transformation | hotspots | sound)*
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	SystemTestAttrs, TermCapTestAttrs, customTest,	
	region,	
	erase, sensitivity,	
	TransInOutAttrs,	
	MediaAccessAttrs, MediaAnnotateAttrs, MediaClippingAttrs,	
	mediaRepeat, type , src	
inline	CoreAttrs,	vazio
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	SystemTestAttrs, TermCapTestAttrs, customTest,	
	region,	
	erase, sensitivity,	
	TransInOutAttrs,	
	MediaAccessAttrs, MediaAnnotateAttrs, MediaClippingAttrs,	
	mediaRepeat, type , src	
applicationWindow	CoreAttrs,	vazio
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	SystemTestAttrs, TermCapTestAttrs, customTest,	
	region,	
	erase, sensitivity,	
	TransInOutAttrs,	
	MediaAccessAttrs, MediaAnnotateAttrs, MediaClippingAttrs,	
	mediaRepeat, type , src	
	description, parameter, size	
delay	CoreAttrs,	(switch | transitionFilter | AnimationGroup)*
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	

	SystemTestAttrs, customTest,	
	MediaAnnotateAttrs	
box	CoreAttrs,	(switch | use | transitionFilter | AnimationGroup | material | texture | transformation | light | hotspots)*
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	SystemTestAttrs, TermCapTestAttrs, customTest,	
	region,	
	erase, sensitivity,	
	TransInOutAttrs,	
	MediaAccessAttrs, MediaAnnotateAttrs, MediaClippingAttrs,	
	size	
cone	CoreAttrs,	(switch | use | transitionFilter | AnimationGroup | material | texture | transformation | light | hotspots)*
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	SystemTestAttrs, TermCapTestAttrs, customTest,	
	region,	
	erase, sensitivity,	
	TransInOutAttrs,	
	MediaAccessAttrs, MediaAnnotateAttrs, MediaClippingAttrs,	
	radius, height, hasSide, hasBase	
cylinder	CoreAttrs,	(switch | use | transitionFilter | AnimationGroup | material | texture | transformation | light | hotspots)*
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	SystemTestAttrs, TermCapTestAttrs, customTest,	
	region,	
	erase, sensitivity,	
	TransInOutAttrs,	
	MediaAccessAttrs, MediaAnnotateAttrs, MediaClippingAttrs,	
	radius, height, hasSide, hasBase, hasTop	
sphere	CoreAttrs,	(switch | use | transitionFilter | AnimationGroup | material | texture | transformation | light | hotspots)*
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	SystemTestAttrs, TermCapTestAttrs, customTest,	
	region,	
	erase, sensitivity,	
	TransInOutAttrs,	
	MediaAccessAttrs, MediaAnnotateAttrs, MediaClippingAttrs,	
	radius	
mesh	CoreAttrs,	(switch | use | transitionFilter |

	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	AnimationGroup | material | texture | transformation | light | hotspots)*
	SystemTestAttrs, TermCapTestAttrs, customTest,	
	region,	
	erase, sensitivity,	
	TransInOutAttrs,	
	MediaAccessAttrs, MediaAnnotateAttrs, MediaClippingAttrs,	
	coord, colorIndex, color, colorIndex, colorPerVertex, normal, normalIndex, normalPerVertex, texCoord, texCoordIndex, creaseAngle, ccw, convex, solid	
backdrop	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	(switch | AnimationGroup)*
	SystemTestAttrs, customTest,	
	region,	
	MediaAnnotateAttrs, MediaClippingAttrs,	
	color, mediaRepeat, type, src	
background	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	(switch | AnimationGroup)*
	SystemTestAttrs, customTest,	
	region,	
	MediaAnnotateAttrs, MediaClippingAttrs,	
	skyColor, skyAngle, groundColor, groundAngle, backSrc, bottomSrc, frontSrc, leftSrc, leftSrc, rightSrc, topSrc	
fog	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	(switch | AnimationGroup)*
	SystemTestAttrs, customTest,	
	region,	
	MediaAnnotateAttrs, MediaClippingAttrs,	
	color, fogType, visibilityRange	
pointLight	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	(switch | AnimationGroup)*
	SystemTestAttrs, customTest,	
	region,	
	MediaAnnotateAttrs, MediaClippingAttrs,	
	ambientIntensity, attenuation, color, intensity, location, on, radius	
spotLight	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	(switch | AnimationGroup)*
	SystemTestAttrs, customTest,	
	region,	
	MediaAnnotateAttrs, MediaClippingAttrs,	

	ambientIntensity, attenuation, beamWidth, color, intensity, cutOffAngle, direction, location, on, radius	
group	CoreAttrs,	(a | switch | transitionFilter AnimationGroup | MediaContentGroup)*
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	region,	
	erase, sensitivity,	
	TransInOutAttrs,	
	MediaAccessAttrs, MediaAnnotateAttrs,	
	center, rotation, scale, sacaleOrientation, translation, billboardAxix, size, order, backgroundColor, visibility, billboard, collide	
cmds	vazio	XMT-A (gramática)
nodes	vazio	XMT-A (gramática)
xmtaMedia	CoreAttrs,	(nodes, cmds)*
	BasicInlineTimingAttrs,	
	SystemTestAttrs, customTest,	
	region,	
	MediaAnnotateAttrs,	
	type, hasOD	

8.7. Área Funcional *Metainformation*

Elementos	Atributos	Conteúdo
meta	CoreAttrs,	vazio
	skip-content,	
	content, name	
metadata	CoreAttrs,	(Elementos RDF)
	skip-content	

8.8. Área Funcional *Structure*

Elementos	Atributos	Conteúdo
XMT-O	StructureModuleAttrs	head?, body?
head	CoreAttrs	meta*, customAttributes*, metadata*, (layout | swicth)*, transition*, defs*, macros*
body	CoreAttrs,	(ScheduleGroup | MediaContentGroup | ContentControlGroup | a |
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	MediaAnnotateAttrs,	

	region	AnimationGroup)*

8.9. Área Funcional *Transitions*

Elementos	Atributos	Conteúdo
transition	CoreAttrs,	(param)*
	SystemTestAttrs customTest, skip-content,	
	type, subtype, startProgress, endProgress, direction, fadeColor	
param	CoreAttrs,	vazio
	name, value	
transitionFilter	CoreAttrs,	vazio
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	skip-content,	
	mode, targetElement, type, subtype, fadeColor, from, to, by, values, calcMode	

8.10. Área Funcional *DEFS*

Elementos	Atributos	Conteúdo
def	CoreAttrs	(xMediaObjectsGroup | MediaAugmentationGroup | material chromakey | texture | light | group | useMacro | xmtaMedia | outline | fontStyle | sound)*
use	CoreAttrs,	vazio
	TimingAttrs, TimeManipAttrs, FlexTimeAttrs,	
	SystemTestAttrs, termCapTestAttrs, customTest,	
	region,	
	erase, sensitivity,	
	TransInOutAttrs,	
	MediaAccessAttrs, MediaAnnotateAttrs,	
	href	

8.11. Área Funcional *Macros*

Elementos	Atributos	Conteúdo
macro	CoreAttrs,	(attrib*, group | useMacro)
	Name,	

useMacro	CoreAttrs,	(AnimationGroup | switch)*
	TimingAttrs, timeManipAttrs, FlexTimeAttrs,	
	systemTestAttrs, TermCapTestAttrs, customTest,	
	region,	
	erase, sensitivity,	
	TransInOutAttrs,	
	MediaAccesAttrs, MediaAnnotateAttrs,	
	href	
put	CoreAttrs,	vazio
	target, value,	
value	CoreAttrs,	(put)*
	name, base,	
target	CoreAttrs,	vazio
	targetElement, attributeName	
targets	CoreAttrs	(target)*
attrib	CoreAttrs,	(targets, value*)
	name, values,	
macros	CoreAttrs	(macro)*

8.12. Grupos de Elementos

Nome	**Elementos**
AnimationGroup	animate | set | animateMotion | animateColor | dragPlane | dragDisc | dragCylinder | dragSphere
ContentControlGroup	switch | prefetch
MediaAugmentationGroup	backdrop | background | pointLight | spotlight
MediaContentGroup	rectangle | circle | points | lines | polygons | curve | img | video | text | string | subtitles | audio | audioClip | inline | applicationWindow | delay | box | cone | cylinder | sphere | mesh | backdrop | background | pointLight | spotlight | xmtaMedia | group | use | useMacro
ScheduleGroup	par | seq | excl
xMediaObjectsGroup	rectangle | circle | points | lines | polygons | curve | img | video | text | string | subtitles | audio | audioClip | inline | applicationWindow | delay | box | cone | cylinder | sphere | mesh

8.13. Grupos de Atributos

Nome	**Atributos**
BasicInlineTimingAttrs	begin, end, dur
CoreAttrs	Id, class, alt, longdesc
FlexTimeAttrs	flexBehaviour , flexBehaviourDefault
MediaAccessAttrs	readIndex
MediaAnnotateAttrs	abstract, author, copyright, title
MediaClippingAttrs	clipBegin, clipEnd

StructureModuleAttrs	id, class
SystemTestAttrs	systemAudioDesc, systemBitrate, systemCaptions, systemComponent, systemCPU, systemLanguage, systemOperatingSystem, systemOverdubOrSubtitle, systemRequired, systemScreenDepth, systemScreenSize
TermCapTestAttrs	termCapFrameRate, termCapColorDepth, termCapScreneSize, termCapGraphicsHardware, termCapAudioOutputFormat, termCapMaxAudioSamplingRate, termCapSpatialAudioCapability, termCapCPULoad, termCapMemoryLoad
TimeManipAttrs	accelerate, decelerate, autoreverse, speed
TimingAttrs	dur, begin, end, min, max, repeatDur, repeatCount, fill, fillDefault, endsync, restart, restartDefault, syncBehavior, syncBehaviorDefault
TransInOutAttrs	transIn, transOut

9
Apêndice B

Este apêndice tem por objetivo descrever a conversão de documentos NCL 2.0 para XMT-O e vice-versa. Como essas linguagens possuem estruturas distintas, as formas de mapeamento entre seus módulos serão apresentadas através de uma abordagem que contempla os principais aspectos definidos por essas linguagens, tais como: a estrutura do documento, estrutura de apresentação, sincronização etc. Os casos mais simples, relacionados à conversão, serão abordados em conjunto. Para os demais casos, considerados nesta dissertação como mais complexos, a conversão será dividida em traduções isoladas. Neste apêndice todas as referências à linguagem NCL correspondem a sua versão 2.0.

9.1.
Estrutura do Documento

As linguagens NCL e XMT-O dividem a estrutura de um documento em duas partes, o cabeçalho e o corpo. Envolvendo esses elementos, encontram-se os indicadores da linguagem, que contêm informações como a identificação do documento (*id*) e as restrições de vocabulário (*xmlns*).

Em NCL a estrutura do documento é especificada pelo módulo *Structure* e em XMT-O pelo módulo homônimo. A Figura 9.1 apresenta os elementos que definem a estrutura de documentos multimídia/hipermídia, o primeiro em NCL e o segundo em XMT-O.

<ncl id="NCL" xmlns=”Schema” ...> <head> ... </head> <body> … </body> </ncl>	<XMT-O id="XMT-O" xmlns=”Schema” ...> <head> ... </head> <body> … </body> </XMT-O>

Figura 9.1 – Estrutura dos documentos especificados em NCL e XMT-O

A conversão das estruturas desses documentos é realizada através da tradução entre seus elementos. Além da tradução, na conversão da estrutura de

documentos XMT-O para NCL também é necessário instanciar elos e conectores NCL a fim de representar a semântica de sincronização de uma composição seqüencial, pois, em XMT-O, o corpo de um documento possui essa semântica. Os aspectos de sincronização das linguagens, incluindo as composições, serão abordados na Seção 9.3.

9.2. Estrutura da Apresentação

Nas linguagens NCL e XMT-O a estrutura da apresentação é definida no cabeçalho do documento. Em NCL essa estrutura é, normalmente, especificada pelo módulo *BasicLayout*, embora possam ser utilizados módulos definidos em outras linguagens (Muchaluat-Saade, 2003). Em XMT-O, essa estrutura é especificada pelo módulo *Layout*, que possui sintaxe similar à definida pelo *BasicLayout*, porém uma estrutura semântica diferente.

Em NCL o elemento *layout* inicia a definição da estrutura da apresentação de um documento. Dentro desse elemento podem ser definidos um conjunto de janelas, através do elemento *topLayout*, contendo um conjunto de regiões, definidas pelo elemento *region*, que, por sua vez, pode conter outras regiões, recursivamente. A altura e largura desses elementos são definidas através dos seus atributos *height* e *width*, respectivamente, que podem conter valores absolutos (*pixels*) ou percentuais, sendo esses, relativos às dimensões do elemento pai, que contém a janela ou região.

O posicionamento das janelas e regiões é definido em relação à distância da sua borda superior e da sua borda esquerda, com as bordas superior e esquerda, respectivamente, do seu elemento pai. Esses valores são representados, respectivamente, pelos atributos *top* e *left*, que também podem conter valores absolutos (*pixels*) ou percentuais, relativos às dimensões do seu elemento pai. Na referência (Moura, 2001) o modelo para apresentação de documentos NCL é descrito em detalhes.

A Figura 9.2 ilustra o modelo de apresentação NCL. Nela encontram-se definidas duas janelas e três regiões. Cada janela contém uma região e uma dessas regiões contém uma terceira região.

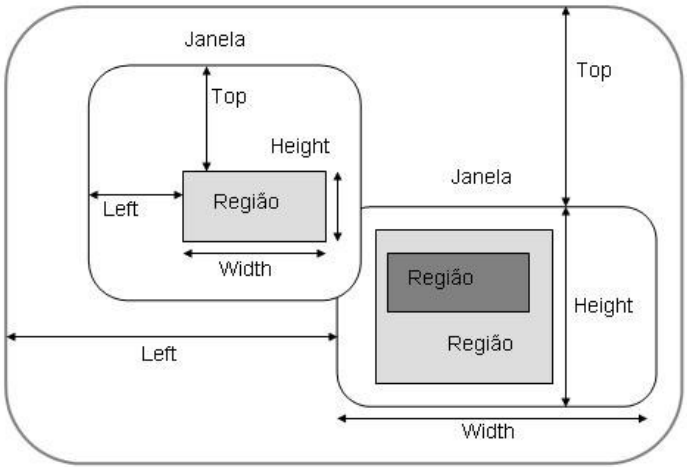

Figura 9.2 – Estrutura da apresentação de documentos NCL

Na linguagem XMT-O a estrutura da apresentação também é definida pelo elemento *layout*. Dentro desse elemento existe apenas uma única janela, definida pelo elemento *topLayout*. As propriedades dessa janela são sua altura e largura, definidas através dos atributos *height* e *width*, respectivamente, que armazenam valores absolutos em *pixels*.

A janela em XMT-O, de forma similar a NCL, pode conter um conjunto de regiões, definidas pelo elemento *region*, que, por sua vez, podem conter outras regiões recursivamente. As dimensões dessas regiões são especificadas através do atributo *size,* que contém os valores da sua largura e altura, separados apenas por um espaço em branco.

O posicionamento das regiões em XMT-O, ao contrário de NCL, é definido pela distância entre o seu centro geométrico e o centro do seu elemento pai, que pode ser a janela ou uma outra região. Essas distâncias são definidas através do atributo denominado *translation*, pertencente ao elemento *region*. XMT-O adota o sistema de coordenadas cartesianas, onde a origem dos eixos é definida no centro do elemento pai. Dessa forma, o atributo *translation* contém, os valores da abscissa e da ordenada, relativos ao centro geométrico da região, separados por um espaço em branco.

Em uma região, seu tamanho (*size*) e translação (*translation*) são sempre especificados através de valores absolutos, cuja unidade pode ser *pixels* ou metros (*meter*). Ao contrário de NCL, onde todas as dimensões são definidas em *pixels*, em XMT-O, através do atributo *metrics*, pertencente ao elemento *layout*, todas as dimensões de um documento, à exceção das dimensões da janela, podem ser definidas pelas expressões *pixels* ou *meter* (ISO/IEC, 2001).

A Figura 9.3 ilustra o modelo de apresentação de um documento XMT-O, onde estão definidas duas regiões com as respectivas medidas de tamanho e translação. Na Figura 9.3, um eixo cartesiano fictício foi representado para demonstrar como essas medidas são obtidas.

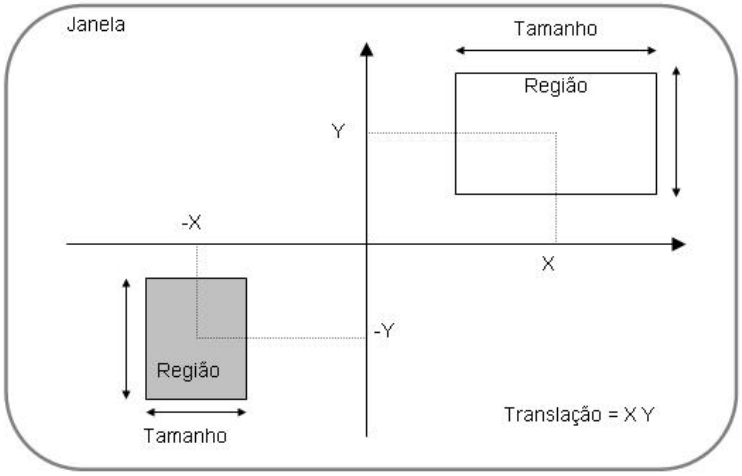

Figura 9.3 – Estrutura da apresentação de documentos XMT-O

9.2.1. Conversão de NCL para XMT-O

Na conversão de documentos NCL para XMT-O que contenham mais de uma janela, a estrutura da apresentação deve ser remodelada, a fim de transformar as janelas originais em uma única, capaz de abranger todas as anteriores.

Para definir uma única janela, são necessárias apenas as maiores medidas, pertencentes às janelas originais de NCL, da altura e largura, acrescidas, respectivamente, das distâncias das bordas superior e esquerda. Como exemplo, na Figura 9.2, a janela que contém duas regiões possui, aparentemente, a maior medida de altura somada à distância da borda superior e, também, a maior medida de largura somada à distância da borda esquerda. Ao contrário do exemplo da Figura 9.2, essas medidas poderiam ser obtidas a partir de janelas distintas, isto é, uma janela poderia possuir a maior medida de altura somada à distância da sua borda superior e outra possuir a maior medida da largura somada à distância da sua borda esquerda. As maiores medidas obtidas correspondem, em XMT-O, à altura, atributo *height*, e à largura, atributo *width*, do elemento *topLayout*.

Para definir as regiões em XMT-O, as medidas de altura e largura, definidas originalmente nas regiões de NCL, não sofrem alterações. Durante a conversão dos documentos, basta agrupar esses valores no atributo *size* das regiões de

XMT-O. A única restrição existente é a necessidade de transformar os valores relativos, que em NCL podem ser expressos através de percentuais, para valores absolutos, expressos em *pixels*.

As dimensões de posicionamento das regiões NCL, que são relativas às distâncias da borda superior e esquerda, na conversão para XMT-O, devem ser transformadas em coordenadas cartesianas. Nessa transformação, deve-se considerar que, se o elemento pai da região NCL era uma janela, agora, em XMT-O, ele será relativo a uma nova janela, criada a partir das dimensões das janelas originais de NCL. Nesse caso, as distâncias da borda superior e esquerda da janela original de NCL devem ser adicionadas às distâncias da borda superior e esquerda, respectivamente, da região a ser convertida.

Para transformar as distâncias originais em NCL para coordenadas cartesianas, faz-se necessário converter genericamente essas medidas e, a partir dessa sincronização, adicionar ou subtrair, dependendo do caso, as distâncias superior e esquerda de cada região. A Figura 9.4 ilustra uma região com distâncias superior e esquerda nulas em relação ao seu elemento pai. Em um sistema de coordenadas cartesianas essas distâncias equivalem aos valores da metade da largura do elemento pai, subtraída da metade da largura da região, sendo o valor obtido, atribuído, com sinal negativo, a ordenada. Para a abscissa, esse cálculo corresponde à metade da altura do elemento pai, subtraída da metade da altura da região.

Com as medidas convertidas, o valor da distância superior de uma região deve ser diretamente subtraído do valor obtido para a abscissa e, o valor da distância superior diretamente adicionado ao valor obtido para a ordenada.

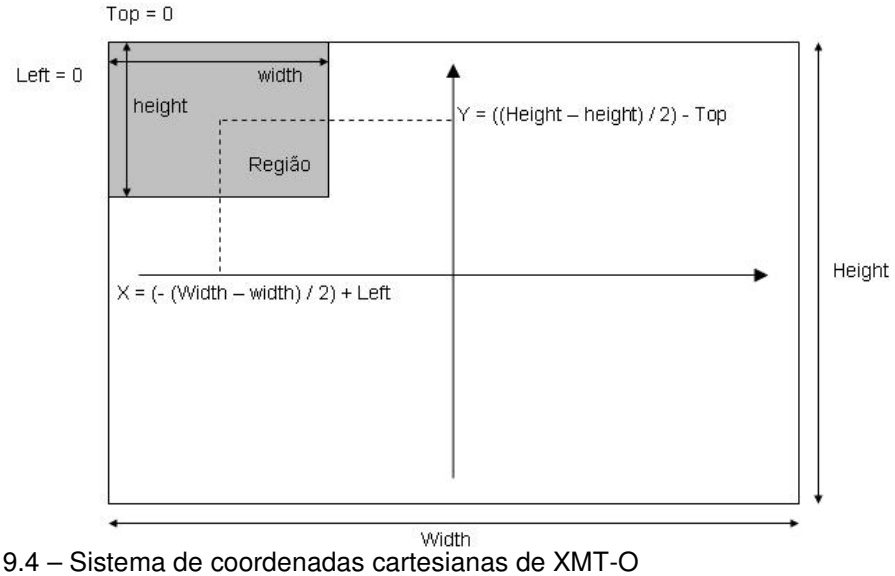

Figura 9.4 – Sistema de coordenadas cartesianas de XMT-O

Na conversão de documentos NCL para XMT-O, alguns atributos, pertencentes às janelas (*topLayout*), e regiões (*region*) podem ser diretamente convertidos, onde incluem-se os atributos *id*, *backgroundColor* e *z-index*. Complementarmente, os atributos *visible* e *title*, definidos em NCL para janelas e regiões podem ser representados, em XMT-O, pelos elementos *skip-content* e *regionName*, respectivamente, sendo que, esse último, encontra-se disponível apenas para regiões.

Ainda nessa conversão, algumas das funcionalidades definidas na estrutura da apresentação de NCL não possuem representação em XMT-O. Esses atributos são: *scroll*, *open*, *close*, definidos para os elementos *topLayout* e, o atributo *fit*, definido nos elementos *region*. Todos esses atributos são especificados, originalmente, nos módulos da linguagem SMIL 2.0, à exceção do atributo *scroll*, que permite ao usuário definir como deseja configurar essa operação em uma janela ou região.

9.2.2. Conversão de XMT-O para NCL

Na conversão de documentos XMT-O para NCL não são necessárias modificações estruturais, assim, a janela e as regiões, definidas em XMT-O, podem ser diretamente traduzidas para NCL.

Para a janela, as medidas de altura e largura não sofrem alterações na representação em NCL. No caso das regiões, essas medidas, quando expressas em *pixels*, devem ser simplesmente retiradas do valor definido no atributo *size* e

destinadas aos atributos para largura e altura dos elementos que definem as regiões NCL. No entanto, quando essas medidas são definidas em metros, elas devem ser convertidas para *pixels*. Essa conversão pode gerar múltiplas versões para um mesmo documento, pois a relação entre o número de *pixels* por unidade de comprimento depende do dispositivo de exibição utilizado.

A tradução da translação, a partir do sistema de coordenadas cartesianas, para as distâncias das bordas superior e esquerda, relativas ao elemento pai, podem ser obtidas através das seguintes equações:

- X = (-(Width – width) / 2) + Left => Left = X + (Width - width) / 2
- Y = (Height - height) / 2 - Top => Top = –Y + (Height - height) / 2

Nessas equações, apresentadas na Figura 9.4, as variáveis *X* e *Y* correspondem aos valores das abscissas e ordenadas, encontradas no atributo *translation*, as variáveis *Width* e *Height* representam a largura e altura, respectivamente, do elemento pai da região e, finalmente, as variáveis *width* e *height*, as respectivas largura e altura da região. A primeira equação obtém a distância da borda esquerda, através da variável *Left* e, a segunda, a distância da borda superior, através da variável *Top*.

Na conversão de XMT-O para NCL, somente o atributo *soundLevel*, definido para regiões de XMT-O, não possui representação na estrutura da apresentação de NCL, porém pode ser definido através de descritores, apresentados na Seção 9.6.

9.3. Relações de Sincronização e Referência

Na linguagem NCL, as relações de referência e de sincronização espaço-temporal hipermídia são especificadas através dos conectores hipermídia. Os relacionamentos, por sua vez, são especificados através de elos, que fazem referência a um conector e aos participantes desses relacionamentos. Os elos NCL são especificados no módulo *Linking* e os conectores nos módulos *XConnector* e *CompositeConnector*.

Na linguagem XMT-O, diferentemente de NCL, as relações de sincronização podem ser estabelecidas através de um conjunto de composições. Além dessas composições, eventos, especificados nos atributos dos elementos

dessa linguagem, também podem estabelecer relações, tanto de sincronização como de referência. Tanto as composições quanto os eventos são especificados em módulos pertencentes à área funcional *Timing Manipulations* de XMT-O, definida no Capítulo 2.

O escopo das relações de XMT-O, especificadas através de composições e eventos, restringe-se a relacionamentos estabelecidos entre objetos pertencentes a uma mesma cena audiovisual. Para especificar relacionamentos entre objetos definidos em diferentes cenas, elos XMT-O, propriamente ditos, são utilizados, especificados no seu módulo *Linking*.

9.3.1. Conversão de NCL para XMT-O

A conversão da estrutura de sincronização de NCL para XMT-O é realizada pela tradução dos conectores NCL, representando relações de sincronização, através de composições e atributos XMT-O. Além dos conectores, os componentes associados a esses conectores devem estar contidos nessas composições ou, então, conter esses atributos.

Os conectores possuem especialização semântica causal ou de restrição (Muchaluat-Saade, 2003). Independente da sua especialização, os conectores são definidos por um conjunto de papéis (*roles*), determinando as funções dos participantes da relação, e por uma descrição de como esses papéis interagem (*glue*). A Figura 9.5 apresenta um conector com semântica causal.

```
<?xml version="1.0"?>
<hypermedia-connector id="starts" xsi:type="CausalHypermediaConnector" …>
    <condition-role id="on_x_presentation_begin" event-type="presentation">
        <condition xsi:type="EventTransitionCondition" transition="starts"/>
    </condition-role>
    <action-role id="start_y" event-type="presentation" action-type="start"/>
    <glue>
        <condition-expression xsi:type="SimpleConditionExpression"
            condition-role="on_x_presentation_begin" />
        <action-expression xsi:type="SimpleActionExpression" action-role="start_y"/>
    </glue>
</hypermedia-connector>
```

Figura 9.5 – Conector hipermídia com semântica causal

Nos conectores com semântica causal qualquer tipo de papel pode ser utilizado, embora sejam mais comuns papéis do tipo condição (*condition-role*) e ação (*action-role*). Na Figura 9.5, o papel do tipo condição, representado pelo elemento *condition-role* avalia se o evento de apresentação (*presentation*) foi

iniciado (*starts*) e, o papel do tipo ação, representado pelo elemento *action-role* define que o evento de apresentação (*presentation*) deve ser iniciado (*start*).

Os tipos básicos de eventos definidos em NCL são: apresentação (*presentation*), clique do mouse (*mouseClick*), posicionamento do mouse (*mouseOver*), foco no objeto (*focus*), pré-busca (*prefetch*) e atribuição (*attribution*). O comportamento desses eventos, por sua vez, é controlado por uma máquina de estados, cujas transições encontram-se sintetizadas na Tabela 9.1.

Transição	Nome
preparado para ocorrendo (início)	*starts*
ocorrendo para preparado (fim ou término natural)	*stops*
ocorrendo para preparado (encerramento)	*aborts*
ocorrendo para terminado (término natural)	*ends* (somente se aplica ao pre-fetch)
ocorrendo para aguardando (aguarda)	*pauses*
aguardando para ocorrendo (reinício ou início)	*resumes*
aguardando para preparado (fim ou encerramento)	*abortsFromPaused*

Tabela 9.1 – Transições de eventos em NCL

Quando a estrutura de sincronização de NCL é convertida para XMT-O, os eventos e transições estabelecidos nos conectores podem ser representados através de atributos XMT-O. O evento de apresentação (*presentation*) pode ser representado pelo atributo XMT-O de início (*begin*) caso esse evento defina uma transição de início (*starts*), e pelo atributo XMT-O de fim (*end*) caso esse evento defina as transições de fim (*stops*) ou encerramento (*aborts*).

Além da apresentação, os eventos de clique (*mouseClick*) e de posicionamento do mouse (*mouseOver*) de NCL também podem ser representados em XMT-O. Esses eventos são definidos como qualificações sobre os atributos de início e fim dos elementos, portanto, somente podem representar as condições e não ações a serem realizadas. Na Figura 9.6 a qualificação relativa ao clique do mouse (*.click*) e ao seu posicionamento (*.mouseover*) são exemplificadas. Nos módulos *SyncbaseTiming* e *XMTEvents* de XMT-O são especificadas diversas qualificações de eventos para os seus atributos de início e fim (ISO/IEC, 2001).

```
<XMT-O>
<body>
  <par>
   …
   <circle radius=”20” begin=”m1.click”>
     <material color =”red”>
   </circle>
   …
   <img id=”m1” src=”logo.jpg” …/>
  </par>
</body>
</XMT-O>
```

```
<XMT-O>
<body>
 <par>
    …
    <circle radius=”20” begin=”m1.mouseover”>
      <material color =”red”>
    </circle>
     …
    <img id=”m1” src=”logo.jpg” …/>
 </par>
</body>
</XMT-O>
```

Figura 9.6 – Eventos em XMT-O

Na Figura 9.6, os documentos XMT-O apresentados exibem o objeto círculo (*circle*) quando o evento de clique do mouse ou o evento de posicionamento do mouse ocorrerem sobre o objeto *m1*. Na realidade, em cada exemplo da Figura 9.6 podem ser encontrados dois eventos, um de apresentação e outro de clique ou posicionamento do mouse. Em uma analogia com as definições de NCL, os eventos de clique ou de posicionamento do mouse de XMT-O definem os papéis de condição, que estariam relacionados ao objeto *m1* e, o evento de apresentação corresponde ao papel de ação, relacionado ao objeto círculo.

Nos conectores, após a definição dos papéis é necessário especificar como eles interagem. No elemento *glue* do conector essa especificação é realizada sobre todos os papéis do conector. Em um conector com semântica causal, como o da Figura 9.5, o seu *glue* define tanto uma expressão de condição (*condition-expression*), relacionando papéis do tipo condição ou propriedade, quanto uma expressão de ações (*action-expression*), relacionando papéis do tipo ação.

Na conversão para XMT-O, a expressão de condição é representada apenas para as expressões simples (*SimpleConditionExpression*), como na Figura 9.5. A expressão de condição composta (*CompoundConditionExpression*) é formada por uma expressão lógica binária, relacionando qualquer número de papéis de condição e de propriedade, utilizando os operadores *and* ou *or*. Na realidade, a expressão de condição composta pode definir vários eventos e transições que devem ser satisfeitos para que a ação seja executada. Em XMT-O essa estrutura somente possui representação para expressões de condição lógicas utilizando o operador *or*, através das construções especificadas no seu módulo *MultiArcTiming*.

A Figura 9.7 apresenta um documento XMT-O representando uma expressão de condição composta. Nessa expressão, três objetos *m1*, *m2* e *m3* possuem eventos definidos que são condições para o evento de início da apresentação do objeto círculo (*circle*). Qualquer ocorrência dos eventos definidos para os objetos *m1*, *m2* e *m3* dispara a ação definida para o círculo.

```
<XMT-O>
…
<body>
  <par>
…
    <circle radius=”20” begin=”m1.click; m2.click; m3.begin”>
```

```
      <material color =”red”>
    </circle>

    <img id=”m1” src=”telemidia.jpg” …/>
    <img id=”m2” src=”puc-rio.jpg” …/>
    <img id=”m3” begin=”10” src=”riodejaneiro.jpg” …/>
</par>
…
</body>
</XMT-O>
```

Figura 9.7 – Expressão de condição composta representada em XMT-O

No *glue* do conector, a expressão de ações também pode ser simples ou composta. A expressão de ações simples (*simpleActionExpression*), como na Figura 9.5, refere-se somente a um papel do tipo ação, sem maiores implicações na conversão para XMT-O. Uma expressão de ações composta (*compoundActionExpression*) é definida por uma expressão, utilizando os operadores *par*, *seq* ou *excl*, envolvendo outras expressões de ações.

Os significados semânticos dos operadores *par*, *seq* e *excl,* em uma expressão composta, são semelhantes aos das composições homônimas definidas em XMT-O. Portanto, na conversão de uma expressão de ações composta para XMT-O, as composições *par*, *seq* e *excl* podem ser utilizadas. A Figura 9.8 apresenta um exemplo de uma expressão de ações composta, com semântica paralela, representada em XMT-O. Nela, quando o evento de clique do mouse sobre o objeto *m1* for acionado, os objetos círculos serão apresentados no mesmo instante de tempo.

```
<XMT-O>
…
<body>
  <par>
    …
    <par begin=”m1.click”>
      <circle radius=”20” >
        <material color =”red”>
      </circle>
      <circle radius=”20” >
        <material color =”blue”>
      </circle>
    </par>
   …
    <img id=”m1” src=”telemidia.jpg” …/>
  </par>
…
</body>
</XMT-O>
```

Figura 9.8 – Expressão de ações composta representada em XMT-O

Para realizar a conversão completa da estrutura de sincronização NCL para XMT-O, além dos conectores, os elos NCL devem ser consultados a fim de obter

os participantes da relação. Na Figura 9.9, um documento NCL contendo dois objetos de mídia, um áudio (*samba*) e uma imagem (*logotele1*), são definidos como participantes, através do elo *link1*, da relação definida pelo conector apresentado na Figura 9.5.

```
<ncl>
…
    <audio descriptor="audio_d1" id="samba" src="coisadepele.mp3">
    <img descriptor="img_d1" id="logotele1" src=" logo.jpg"/>
…
   <linkBase>
       <link id="link1" xconnector="starts.xml">
          <bind component="samba" role="on_x_presentation_begin"/>
          <bind component="logotele1" role="start_y"/>
       </link>
    </linkBase>
…
</ncl>
```

Figura 9.9 – Documento especificado em NCL com uma base de elos

O documento XMT-O obtido a partir do documento NCL apresentado na Figura 9.9 é exibido na Figura 9.10. Nesse documento o objeto imagem (*img*) é exibido no mesmo instante do início da execução do objeto de áudio (*audio*).

```
<XMT-O>
…
    <audio id="samba" src="coisadepele.mp3" region="audioRegion1" dur="94s"/>
    <img src="logo.jpg" id="logotele1" begin="samba.begin"/>
…
</XMT-O>
```

Figura 9.10 – Documento especificado em XMT-O com eventos

Além dos conectores com semântica causal, documentos NCL podem conter conectores com semântica de restrição. Nesses conectores somente papéis do tipo propriedade podem ser utilizados (Muchaluat-Saade, 2003). Papéis desse tipo sempre retornam um valor, dependendo da propriedade representada. Esses papéis podem definir também um valor de deslocamento, representado pelo atributo *offset*. Como exemplo, o atributo *offset* pode determinar que uma propriedade contenha um intervalo de tempo após um evento ter ocorrido, como na especificação: “10 segundos após o evento de clique do mouse”, onde o atributo *offset* é utilizado na definição do tempo transcorrido.

Os papéis do tipo propriedade também podem ser empregados em conectores causais. Como exemplo, considere um conector causal que defina um papel do tipo propriedade, com semântica similar ao citado no exemplo do parágrafo anterior, e um papel do tipo ação, associado a um evento de apresentação. A conversão desse conector para XMT-O corresponde a um atributo

de início com 10 segundos de espera após o evento de clique do mouse (*begin=”m1.click+10”*).

O emprego dos papéis do tipo propriedade em conectores de restrição é apresentado na Figura 9.11. Nela um conector define uma relação onde os componentes a serem associados ao conector através de *binds* devem terminar sua apresentação simultaneamente.

```
<?xml version="1.0"?>
<hypermedia-connector id="finishes " xsi:type="ConstraintHypermediaConnector" … >
  <property-role id="x_presentation_end" event-type="presentation">
    <property xsi:type="EventTransitionProperty" transition="stops"/>
  </property-role>
  <property-role id="y_presentation_end" event-type="presentation">
    <property xsi:type="EventTransitionProperty" transition="stops"/>
  </property-role>
  <glue>
    <property-expression xsi:type="PropertyToPropertyExpression" comparator="eq"
     first-property-role="x_presentation_end" second-property-role="y_presentation_end"/>
  </glue>
</hypermedia-connector>
```

Figura 9.11 – Conector hipermídia com semântica de restrição

Na Figura 9.11 são definidos dois papéis do tipo propriedade, através do elemento *property-role*, relativos a ocorrência de uma transição no evento de apresentação, definida pelo atributo *event-type*. Essa transição corresponde à mudança do estado ocorrendo para o preparado, definida pelo atributo *transition*. Nesse mesmo conector, seu *glue* compara, através do elemento *property-role*, os papéis de propriedades do mesmo tipo, utilizando o operador *eq* (=). A sintaxe completa dos conectores pode ser encontrada na referência (Muchaluat-Saade, 2003).

Ao contrário dos conectores com semântica causal, os conectores com semântica de restrição não possuem representação direta em XMT-O. No conector apresentado na Figura 9.11, caso fossem utilizados atributos definidos em XMT-O para representá-lo, considerando dois objetos associados aos papéis desse conector, *m1* e *m2*, seus atributos XMT-O de fim seriam definidos como “*m1.end = “m2.end””* e “*m2.end = “m1.end””*. Essa construção, a princípio, não é válida, pois define uma referência cruzada entre os atributos, sem expressar o valor efetivo para o término da apresentação do objeto.

A dificuldade da representação semântica de restrições NCL em XMT-O se aplica às demais construções realizadas pelos conectores, incluindo aquelas onde os papéis de propriedade contêm valores de atributos dos objetos participantes de um relacionamento. Por exemplo, dada uma restrição de relação de sincronização

espacial especificando que “dois objetos devem ter a mesma largura”. Na representação em XMT-O, os atributos relativos à largura dos objetos fariam referências entre si, sem estabelecer o valor definido para essa medida.

9.3.2. Conversão de XMT-O para NCL

Na conversão de XMT-O para NCL todas as composições, eventos e elos XMT-O com semântica de sincronização ou referência devem ser representados através de conectores NCL, onde os objetos, contidos nessas composições, ou que contenham os atributos relacionados aos eventos, ou mesmo, que estejam definidos como âncoras em elos XMT-O, devem estar unidos, através de *binds*, aos conectores obtidos.

Elos XMT-O são definidos pelo elemento *a*, contendo, primariamente, o atributo *href*. Esse atributo é utilizado para definir o endereço (URI) de um outro documento MPEG-4[33]. O elemento *a* pode conter vários componentes, de diversos tipos, como composições e objetos de mídia. Todos os componentes contidos nesse elemento são utilizados como âncoras no relacionamento especificado pelo elo.

Nos elos XMT-O a relação semântica é, normalmente, de referência, onde uma âncora é associada a um dos objetos da cena de origem, sendo acionada pela ação do clique do mouse. Porém esses elos podem ser especializados por atributos XLink, alterando seu evento de iniciação através do atributo *actuate*. Quando esse atributo contém o valor *onLoad*, o elo é acionado no momento em que seus componentes são apresentados. Caso seus componentes possuam atributos XMT-O de início (*begin*), as condições estabelecidas nesses atributos devem ser previamente satisfeitas para iniciação do elo. A Figura 9.12 apresenta um documento XMT-O definindo um elo.

```
<XMT-O id="XMT-O"  xmlns=”Schema” ...>
  <head>
    <layout type="xmt/xmt-basic-layout" metrics="pixel">
      <topLayout height="240" width="320"/>
    </layout>
  </head>
  <body>
```

[33] Esse documento deve estar no formato de apresentação, estruturado de acordo com a arquitetura definida pelo MPEG-4 *Systems*, apresentada no Capítulo 2.

```
    <par>
     <a href="documento.mp4" actuate="onLoad">
      <circle begin="rectangle1.click" size="1 1">
        <transformation visibility="false" />
      </circle>
     </a>
     <rectangle id="rectangle1" size="100 20">
       <material color="white" filled="true" />
     </rectangle>
    </par>
  </body>
</XMT-O>
```

Figura 9.12 – Documento especificado em XMT-O com um elo definido

No exemplo apresentado na Figura 9.12, o elo de referência para o arquivo "documento.mp4" é acionado quando o objeto sintético do tipo círculo (*circle*) é apresentado, no entanto, esse objeto somente é apresentado quando o objeto sintético do tipo retângulo (*rectangle*), iniciado através de uma composição paralela, é clicado com o mouse.

Na conversão para NCL, um elo XMT-O corresponde a um elo NCL cujos *binds* devem associar as âncoras XMT-O, definidas internamente ao elemento *a*. As relações, por sua vez, podem ser especificadas por conectores com semântica causal. Nesses conectores, papéis do tipo condição podem avaliar o evento de clique do mouse (*mouseClick*), na sua transição de ocorrendo para preparado (*stops*), ou, caso o elemento *a* possua o atributo *actuate* com valor *onLoad*, o evento de apresentação (*presentation*), na transição de preparado para ocorrendo (*starts*). Complementarmente, a definição de um papel do tipo ação nesses conectores permite que o evento de apresentação (*presentation*) seja iniciado (*start*). No *glue* desse conector sua expressão de condição composta (*CompoundConditionExpression*) define uma expressão lógica binária, utilizando o operador *or*, composta pelos seus papéis de condição.

No caso das composições, as que definem semântica para apresentação paralela e seqüencial de XMT-O são herdadas de SMIL 2.0 (Pereira & Ebrahimi, 2002). Na referência (Rodrigues et al., 2002), o modelo de representação dessas composições para nós e elos NCM é apresentado. Em (Muchaluat-Saade, 2003), esse modelo é instanciado em NCL através de templates hipermídia. A Figura 9.13 apresenta o modelo de representação das composições paralela e seqüencial de XMT-O através de composições e elos NCM.

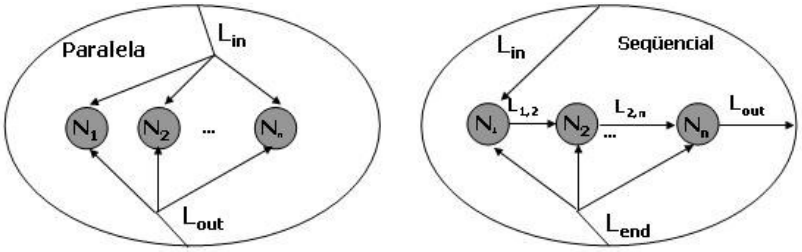

Figura 9.13 – Composições paralela e seqüencial XMT-O representadas por composições e elos NCM

Na Figura 9.13, a composição paralela é representada através de um nó de composição, contendo outros nós, correspondentes aos componentes da composição original. No nó de composição seus componentes são sincronizados pelos elos, conforme a semântica da composição paralela de XMT-O. O elo L_{in} associa o início da apresentação dos componentes ao início da apresentação da composição e o elo L_{out} associa o fim da apresentação dos componentes ao término da apresentação da composição.

Ainda na Figura 9.13, a composição seqüencial, de forma similar à composição paralela, é representada por um nó de composição. O elo L_{in} inicia a exibição do componente correspondente ao primeiro elemento da composição seqüencial de XMT-O quando a apresentação da composição é iniciada. Em seguida são iniciados, ao término do componente anterior e em ordem definida pela composição original de XMT-O, os componentes subseqüentes, associados através dos elos L_{ii+1}, onde i corresponde à ordem do componente atual e $i+1$ ao próximo componente. O elo L_{out} associa o término da composição ao fim da apresentação do seu último componente. Finalmente, o elo L_{end} é utilizado para terminar as apresentações dos componentes, caso a composição seja interrompida antes do fim da apresentação dos seus componentes.

A composição exclusiva de XMT-O é a única não herdada diretamente de SMIL 2.0. Os componentes pertencentes a essa composição são exibidos individualmente, sem uma ordem pré-estabelecida. Ao contrário de SMIL 2.0, onde podem ser definidas prioridades para esses componentes, indicando qual pode ou não ser interrompido por outros, em XMT-O todos os componentes têm a mesma ordem de prioridade, isto é, quando um objeto é acionado por um evento e outro já está sendo apresentando, esse último é interrompido, independente de qualquer fator.

A Figura 9.14 apresenta a proposta desta dissertação para representar, através de nós e elos NCM, a semântica da composição exclusiva de XMT-O. Nessa figura, a composição exclusiva é representada através de um nó de composição, que contém outros nós, correspondentes aos componentes da composição original. No nó de composição são definidos elos L_i, onde $0 > i \leq n$, tal que n corresponde ao número de componentes da composição. Cada elo L_i associa a um componente todos os demais pertencentes à mesma composição. Quando um componente é apresentado, através do seu elo L_i, a apresentação dos demais é interrompida. O elo L_{out} é utilizado para terminar as apresentações de todos os componentes, caso a composição seja previamente interrompida.

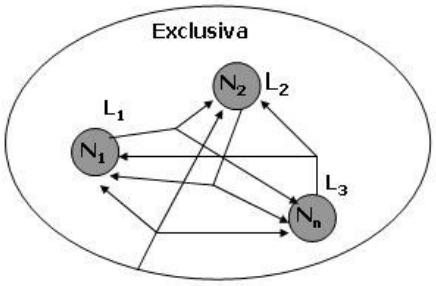

Figura 9.14 – Composição exclusiva XMT-O representada por composição e elos NCM

A proposta apresentada na Figura 9.14 é válida somente quando os nós internos à composição não possuem relacionamentos de sincronização com outros nós. Para exemplificar quando essa representação não é válida considere, como exemplo, o documento XMT-O apresentado na Figura 9.15.

```
<XMT-O>
…
  <par>
     <video id="vid" …/>
     <excl>
        <par begin="englishBrn.click" >
           <audio  id="english" begin="vid.begin" src="english.mp3"/>
        </par>
        <par begin="portugueseBtn.click" >
           <audio  id="portuguese" begin="vid.begin" src="portuguese.mp3"/>
        </par>
     <excl/>
  </par>
…
</XMT-O>
```

Figura 9.15– Composição exclusiva de XMT-O

O conteúdo da composição exclusiva, apresentada na Figura 9.15, é formado por duas composições paralelas, iniciadas através do evento do clique do mouse sobre dois objetos distintos. Dentro de cada composição paralela dois objetos de mídia áudio encontram-se sincronizados temporalmente com um objeto

de mídia vídeo, declarado fora da composição exclusiva. Durante a exibição do documento da Figura 9.15, os dois objetos de áudio serão sincronizados com o vídeo, porém não serão executados até que alguma das composições paralelas seja apresentada. Nesse caso, independente da composição escolhida, os objetos de áudio estarão sempre sincronizados com o objeto de vídeo, pois essa composição impede apenas a apresentação simultânea dos objetos de áudio, não interrompendo o seu sincronismo.

Voltando à composição paralela, esta apresenta ainda três semânticas distintas para o seu término. A Figura 9.16 apresenta composições paralelas definidas em (Muchaluat-Saade, 2003). Essas composições são representadas por nós e elos NCM, com diferentes semânticas para sua finalização. Complementando o nó de composição e os elos apresentados na Figura 9.13, o elo L_{last} (1) associa o término da composição ao término do seu último componente. O elo L_n (2) associa o término da composição ao término de um componente previamente escolhido e, finalmente, o elo L_{first} associa o término da composição ao término do seu primeiro componente.

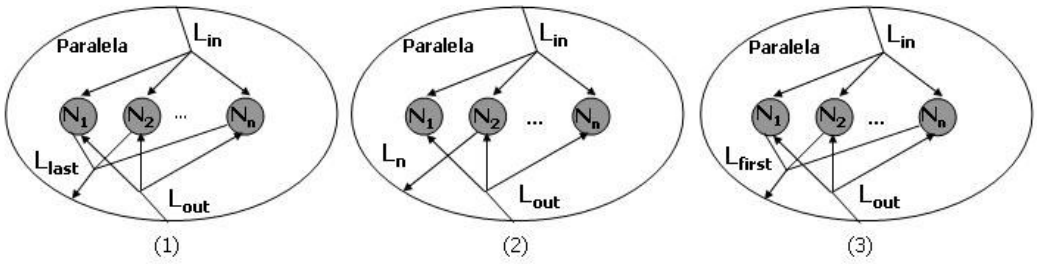

Figura 9.16 – Composição paralela terminada pelo último componente (1), terminada por um componente específico (2), terminada pelo primeiro componente (3)

É importante destacar que, nas composições XMT-O, outros relacionamentos podem ser definidos sobre os seus componentes. Inclusive, tais relacionamentos têm maior prioridade do que os definidos pela composição.

Alguns eventos de XMT-O não possuem representação nos eventos básicos de NCL, entre esses, têm-se os eventos de visibilidade de um componente (*viewable*), que detectam a sua visibilidade na apresentação, os eventos de animação (*near*, *collide*), que detectam a proximidade e a colisão entre componentes e as especializações dos eventos do mouse (*mouseup*, *mousedown*, *mouseout*).

9.4. Interfaces

Na linguagem NCL as interfaces são especificadas nos módulos *MediaInterface*, *CompositeInterface*, *AttributeInterface* e *SwitchInterface*. Por outro lado, a linguagem XMT-O não especifica elementos destinados, especificamente, a representação de interfaces, além daquelas existentes diretamente nos elos dessa linguagem.

Na conversão das interfaces especificadas em NCL para XMT-O, esses elementos podem ser parcialmente representados através dos objetos sintéticos de XMT-O. Âncoras NCL são definidas pelo seu elemento *area*, que permite especificar partes espaciais dos objetos de mídia, através do seu atributo *coords*, bem como partes temporais, através dos seus atributos *begin*, *end* e *dur*. Todos esses atributos são definidos nos objetos sintéticos de XMT-O, que representam figuras geométricas, como retângulos e círculos. Na realidade o elemento *area*, originalmente definido em SMIL, especifica, além dos atributos citados, a forma geométrica da âncora espacial, através do seu atributo *shape* (W3C, 2005a).

Os objetos sintéticos de XMT-O não são capazes de representar algumas das funcionalidades do elemento *area* de NCL. Entre essas limitações, destacam-se a impossibilidade de definir âncoras textuais, através dos atributos *text* e *position* e a limitação na definição de âncoras baseadas nas propriedades dos objetos de mídia contínua, como números de quadros de vídeo e amostras de áudio, através dos atributos *first* e *last.*

As Figuras 9.17 e 9.18 apresentam, respectivamente, um trecho de um documento NCL contendo um objeto de mídia com várias âncoras e, um documento XMT-O contendo um objeto de mídia com vários objetos sintéticos do tipo retângulo. O documento da Figura 9.18 corresponde ao documento da Figura 9.17 convertido. No documento apresentado na Figura 9.18, além do objeto de áudio NCL, convertido diretamente para o objeto de áudio XMT-O, cada âncora (*area*) foi representada por objetos sintéticos do tipo retângulo (*rectangle*). As âncoras definidas são temporais, especificando uma parte específica do tempo do áudio. Para representar esses intervalos de tempos, as propriedades de início e fim das âncoras, representadas pelos atributos *begin* e *end*, respectivamente, foram copiadas para os atributos homônimos definidos nos objetos sintéticos de

XMT-O. Porém, como esses intervalos de tempo são relativos ao início da apresentação do objeto de áudio, nos objetos sintéticos XMT-O essas propriedades foram qualificadas com a propriedade de início do áudio (*samba.begin*).

```
<ncl>
…
    <audio descriptor="audio_d1" id="samba" src="coisadepele.mp3">
      <area id="part1" begin="8.4s" end="18s"/>
      <area id="part2" begin="18.5s" end="28s"/>
      <area id="part3" begin="29s" end="39s"/>
      <area id="part4" begin="39.5s" end="50s"/>
      <area id="part5" begin="50.5s" end="71.4s"/>
      <area id="part6" begin="72s" end="94s"/>
    </audio>
…
</ncl>
```

Figura 9.17 – Documento NCL contendo um objeto de mídia e suas âncoras

```
<ncl>
…
      <audio region="audioRegion1" id="samba" src="coisadepele.mp3" />
      <rectangle id="part1" begin="samba.begin+8.4s" end="samba.begin+18s" />
      <rectangle id="part2" begin="samba.begin+18.5s" end="samba.begin+28s" />
      <rectangle id="part3" begin="samba.begin+29s" end="samba.begin+39s" />
      <rectangle id="part4" begin="samba.begin+39.5s" end="samba.begin+50s" />
      <rectangle id="part5" begin="samba.begin+50.5s" end="samba.begin+71.4s" />
      <rectangle id="part6" begin="samba.begin+72s" end="samba.begin+94s" />
…
</ncl>
```

Figura 9.18 – Documento XMT-O contendo objetos

Na Figura 9.18, os objetos sintéticos não estão associados a nenhuma região para apresentação, assim, esses objetos se destinam apenas a delimitar os intervalos temporais do áudio. No caso de âncoras espaciais, os retângulos devem estar associados à mesma região do objeto destino da âncora, porém, sua propriedade de transparência deve estar ativada (ISO/IEC, 2001).

Além da tradução das âncoras NCL, a conversão de documentos NCL para XMT-O deve retirar o mapeamento entre portas e interfaces dos nós internos. As portas NCL, definidas em nós de composição, especificam esse mapeamento com o objetivo de garantir a propriedade de composicionalidade (Muchaluat-Saade, 2003), no entanto, como XMT-O não define portas, os relacionamentos devem ser estabelecidos diretamente entre as âncoras.

Complementarmente, as interfaces em XMT-O, embora não sejam explicitamente definidas, podem assim ser consideradas para os objetos sintéticos

definidos sem regiões associadas e com especificações temporais totalmente qualificadas (início de outros objetos de mídia).

9.5. Objetos de Mídia

Na linguagem NCL os objetos de mídia são especificados no módulo *BasicMedia*. Os tipos básicos desses objetos, definidos em NCL são: animação (*animation*), áudio (*audio*), imagem (*img*), texto (*text*), fluxo de texto (*textstream*) e vídeo (*video*). Além desses, o elemento de referência (*ref*) permite estabelecer referências a outros objetos. Os objetos podem adotar diversos padrões de codificação, compactação ou compressão, sendo necessário especificar apenas, em cada instância desses elementos, o seu tipo MIME através do seu atributo *type*.

Na linguagem XMT-O, à exceção dos tipos animação, fluxo de texto e referência, os demais tipos básicos, definidos em NCL, são encontrados. Além desses tipos, a área funcional *Media Objects*, definida no Capítulo 2, especifica muitos outros, com destaque para os tipos dos objetos sintéticos.

Apesar do XMT-O, e o MPEG-4 de forma geral, definir um grande número de tipos de objetos de mídia, o padrão MPEG-4 *Systems* limita os padrões de codificação de objetos que podem ser empregados (Lim & Singer, 2004). Essa medida favorece a compatibilidade entre os exibidores MPEG-4, porém limita o processo de autoria ao dificultar a utilização do conteúdo audiovisual.

Na conversão de documentos da linguagem NCL para XMT-O, em relação aos objetos de mídia, os tipos áudio (*audio*), imagem (*img*), texto (*text*) e vídeo (*video*) podem ser diretamente traduzidos para XMT-O. Esses tipos são representados em XMT-O por elementos declarativos com os mesmos nomes. Entre os atributos, o *id*, que define a identificação do objeto, e o *src*, que define o endereço (URI) para o conteúdo do objeto, possuem representações homônimas em XMT-O. Além desses, o atributo *label* pode ser representado em XMT-O pelos atributos *abstract*, *author*, *copyright* e *title*, que definem informações sobre o seu conteúdo. O atributo relativo ao descritor NCL, denominado *descriptor* contém um ponteiro para informações relativas ao controle da apresentação do conteúdo, abordadas na Seção 9.6, e os atributos de teste (*TestAttributes*) possuem representação completa em XMT-O, que herda o módulo *BasicContentControl* de

SMIL 2.0. Dessa forma, a conversão de objetos de mídia de NCL para XMT-O é limitada pela restrição nos padrões de codificação empregados, e não por restrições das linguagens.

Na conversão da linguagem XMT-O para NCL, à exceção dos tipos básicos citados no parágrafo anterior, todos os demais não possuem representação em NCL. A lista de todos os tipos de objetos definidos em XMT-O pode ser encontrada no Apêndice A, na área funcional *Media Objects*.

9.6. Especificação da Apresentação

Na linguagem NCL o controle da apresentação é definido nos descritores. Descritores NCL são estruturas que reúnem informações referentes às características para exibição dos objetos de mídia do documento multimídia/hipermídia. Ao contrário de NCL, em XMT-O essas informações encontram-se distribuídas diretamente nos objetos.

Em NCL a definição dos descritores é realizada no cabeçalho do documento, através de uma base de descritores (*descriptorbase*). Dentro dessa base, cada descritor, denominado *descriptor*, é unicamente identificado através do seu atributo *id*. A partir da definição dos descritores, cada objeto, definido no corpo do documento NCL, pode referenciá-los através dos seus identificadores.

Entre os principais atributos pertencentes ao elemento *descriptor* estão os atributos temporais *dur*, *min* e *max,* que informam, respectivamente, a duração esperada, mínima e máxima para a apresentação de um objeto qualquer. Além desses, os descritores definem também o local, isto é, a região onde um objeto será apresentado, através do seu atributo *region*. Em relação ao local de apresentação, o descritor contém ainda um atributo chamado *enableTimeBar*, usado para habilitar a barra de rolagem no tempo.

Na conversão para XMT-O as informações, contidas nos valores dos atributos dos descritores, podem ser diretamente traduzidas para os atributos dos objetos de XMT-O. No entanto, algumas propriedades de apresentação não possuem representação em XMT-O. Nessa lista de propriedades inclui-se uma possível referência a uma folha de estilo (W3C, 1998a), através do atributo *style*, e outros atributos específicos para objetos de áudio, definidos pelos atributos

balanceLevel, *trebleLevel* e *bassLevel*. Além desses, um atributo denominado *player*, cuja função é identificar a ferramenta de exibição utilizada na apresentação de um objeto, também não possui representação em XMT-O.

Na conversão de documentos XMT-O para NCL, os descritores de NCL devem ser gerados a partir das informações encontradas nos atributos dos objetos. Essas informações são relativas às características de apresentação e controle do tempo de exibição. Algumas dessas características foram anteriormente citadas na conversão de NCL para XMT-O, como a região para exibição (*region*) e a duração da apresentação (*dur*, *min* e *max*).

Alguns atributos definidos em XMT-O não possuem representação em NCL. Entre esses atributos, destacam-se as especificações de apresentação sobre objeto de mídia contínua, como a taxa de aceleração (*speed*) e a opção de apresentação em retrocesso (*autoreverse*).

9.7. Controle da Apresentação

Na linguagem NCL um conjunto de nós alternativos podem ser agrupados em um único elemento, cujo objetivo é especificar documentos hipermídia adaptáveis ao contexto da apresentação. Esse elemento, denominado *switch*, é especificado no módulo *ContentControl*. Além do *switch*, que pode conter objetos de mídia, bem como nós de composição, o elemento *descriptorSwitch*, especificado no módulo *DescriptorControl*, permite que seja definido um conjunto de descritores alternativos para as adaptações no documento.

O controle da apresentação em documentos XMT-O é, em alguns pontos, similar ao definido em NCL. A sintaxe e as construções estabelecidas nas duas linguagens são similares, sendo que, em ambas a escolha entre as alternativas disponíveis é estabelecida por regras (*rule*). Em XMT-O o elemento *switch*, especificado no módulo *BasicContentControl*, agrupa apenas conjuntos de objetos de mídia, onde as regras são referenciadas como atributos desses objetos.

Em XMT-O, assim como em NCL, regras podem ser previamente declaradas na região do cabeçalho de um documento, através do elemento *customAttributes*, especificado no módulo *CustomTestAttributes*. No conjunto de objetos XMT-O de um *switch*, aquele cuja regra estabelecida é atendida primeiro

é selecionado e as demais opções descartadas. As regras também podem ser estabelecidas sem o elemento *switch*, diretamente sobre os objetos, evitando a escolha entre as opções disponíveis.

Como XMT-O herda as definições do controle da apresentação de SMIL 2.0, várias diferenças no controle da apresentação entre SMIL e NCL, citadas em (Silva et al., 2004b), se aplicam também a XMT-O. Entre essas se destaca a possibilidade da associação entre as regras e os nós através do elemento *bindRule*, permitindo o reuso dos nós, independentemente de regras estabelecidas sobre um contexto específico.

A Figura 9.19 apresenta um documento NCL onde o elemento *presentationRuleBase* define o conjunto de regras para a apresentação, especificadas pelos elementos *presentationRule*. Nesse elemento o atributo *id* realiza sua identificação e os demais atributos estabelecem a regra propriamente dita. Ainda na Figura 9.19, o conjunto de opções de nós é definido dentro do elemento *switch* e a associação das regras aos nós é definida pelo elemento *bindRule*.

```
<ncl>
<head>
<presentationRuleBase>
   <presentationRule id="ruleEn" var="systemLanguage" op="eq" value="en" />
   <presentationRule id="rulePt" var="systemLanguage" op="eq" value="pt-br" />
</presentationRuleBase>
<descriptorBase>
    ...
<body>
 ...
 <switch id=" subt">
     <bindRule rule="ruleEn" component="subtEn" />
     <bindRule rule="rulePt" component="subtPt" />
     <img id="subtEn" src=”logoEn.jpg” .. />
     <img id="subtPt" src=”logoPt.jpg” .. />
 </switch>
 …
</body>
</ncl>
```

Figura 9.19 – Documento NCL com controle de apresentação

A Figura 9.20 apresenta o documento XMT-O obtido a partir da conversão do documento NCL especificado na Figura 9.19. As regras estabelecidas em NCL podem ser representadas diretamente em XMT-O. Quando as regras NCL, definidas no atributo *var* do elemento *presentationRule*, correspondem as regras definidas no seu módulo *TestAttributes*, elas podem ser representadas diretamente nos elementos de XMT-O, como no caso da Figura 9.19. Porém, quando essas

regras são definidas no documento NCL, elas também devem ser definidas através do elemento *customTest* nos documentos XMT-O, sendo posteriormente referenciadas pelos atributos *customTest* nos objetos. Na Figura 9.19, apesar do elemento *descriptorSwitch* não ter sido utilizado, a tradução desse elemento para XMT-O envolve a duplicação dos elementos por ele descritos, onde cada elemento deve possuir a regra estabelecida para a escolha dos descritores.

```
<XMT-O>
…
<body>
...
<switch >
     <img id="subtEn" src=”logoEn.jpg” systemLanguage="en"  .. />
     <img id="subtPt" src=”logoPt.jpg” systemLanguage="pt-br"   .. />
 </switch>
…
</body>
</XMT-O>
```

Figura 9.20 – Documento XMT-O com controle de apresentação

Na conversão para XMT-O o elemento *portSwitch,* especificado no módulo *SwitchInterface* de NCL, não é representado. Esse elemento realiza o mapeamento dos elos para os componentes pertencentes a uma composição *Switch*. Seu mapeamento não é necessário, uma vez que XMT-O não define a propriedade de composicionalidade, conforme citado anteriormente.

Na conversão contrária, de XMT-O para NCL, a representação dos elementos é similar à da conversão original, de NCL para XMT-O, à exceção do módulo *PrefetchControl* de XMT-O, que especifica o elemento *pre-fetch*. Esse elemento, através do seu atributo *src*, define o endereço (URI) para o conteúdo de um objeto a ser recuperado com prioridade. Na linguagem NCL, as operações de pré-busca são especificada através de eventos nos conectores hipermídia (Muchaluat-Saade, 2003).

9.8. Animação

A linguagem NCL não define estruturas para animação dos objetos de mídia pertencentes a uma apresentação. Nela as animações podem estar definidas como conteúdos nos objetos de mídias apresentados. A linguagem XMT-O, por outro lado, incorpora os módulos de animação definido em SMIL 2.0, além de definir os seus próprios. Esses módulos permitem que as animações, definidas sobre os

objetos pertencentes a uma apresentação, sejam especificadas na própria linguagem.

Como NCL não possui representação para módulos de animação, não é possível representar em NCL os módulos de animação XMT-O. Porém, na transformação inversa, algumas animações, que ocorrem na exibição de documentos NCL, podem ser representadas em XMT-O.

Como um exemplo, a barra de rolagem no tempo, definida pelos descritores NCL através de um atributo chamado *enableTimeBar*, aplicada, normalmente, a objetos de áudio, pode ser representada como uma animação MPEG-4. A Figura 9.21 apresenta a descrição em XMT-O de uma animação representando a barra de rolagem NCL.

```
<audio id="samba" src="coisadepele.mp3" region="audioRegion1" .../>
<group region="audioRegion1" id="groupncl" begin="samba.begin" end="samba.end" …>
    <animateMotion begin="groupncl.begin" calcMode="linear" dur="94s" to="xx xx" … />
    <rectangle size="16 16">
        <material color="#C0C0C0" filled="true"/>
        <transformation translation="xx xx"/>
    </rectangle>
    <lines begin="groupncl.begin" end="groupncl.end" coord="xx xx;xx xx" …/>
    <lines begin="groupncl.begin" end="groupncl.end" coord="xx xx;xx xx" …/>
    <lines begin="groupncl.begin" end="groupncl.end" coord="xx xx;xx xx" …/>
    < lines begin="groupncl.begin" end="groupncl.end" coord="xx xx;xx xx" …/>
</group>
```

Figura 9.21 – Barra de rolagem construída através de animações XMT-O

Na Figura 9.21 um grupo XMT-O, definido pelo elemento *group*, é utilizado para facilitar a especificação da animação, permitindo que suas ações sejam definidas apenas uma vez e aplicadas a todos os objetos pertencentes ao grupo. O grupo é associado à região *audioRegion1* e sua apresentação encontra-se sincronizada com o objeto de áudio *samba* (*begin="samba.begin" end="samba.end"*). Dentro do grupo são definidos um retângulo (*rectangle*) e quatro linhas (*lines*). Esse retângulo corresponde ao botão da barra de rolagem e as linhas oferecem um efeito tridimensional a esse botão, fazendo um contorno sombreado nele.

O posicionamento do retângulo, apresentado na Figura 9.21, é definido pelo elemento *transformation*, através do seu atributo *translation*. Complementarmente, o posicionamento das linhas é definido pelos atributos *coord*. Os valores desses posicionamentos são calculados pela disposição da região dentro da janela, definido pela estrutura da apresentação (Seção 9.1). Após definidos os objetos e suas posições, a animação, representada pelo elemento

animateMotion, atua sobre o grupo *groupncl*, movimentando todos os seus elementos da sua posição inicial até as coordenadas finais.

Durante o período de 94 segundos, definido pelo atributo *dur* do elemento *animateMotion*, as coordenadas iniciais serão linearmente incrementadas (*calcMode="linear"*). O período de tempo, definido no atributo *dur*, corresponde ao tempo de exibição do áudio. O valor desse período de tempo foi previamente calculado e atribuído a esse atributo, que não aceita referências a eventos, como o término da exibição do objeto de áudio (*samba.end*). Nesse caso, variações no tempo da apresentação do áudio podem comprometer o sincronismo com a animação.

9.9. Informações do Documento

Na linguagem NCL as informações do documento são definidas através do módulo *Metainformation*, idêntico ao módulo *Metainformation* de SMIL 2.0. Como em XMT-O essas informações são definidas pelo mesmo módulo, também herdado de SMIL 2.0, as conversões entre essas linguagens é direta e sem perda de representatividade.

9.10. Elementos de Pré-compilação

Os elementos de pré-compilação são estruturas NCL e XMT-O, definidas nesta dissertação, que não possuem representação direta na linguagem de destino da conversão ou que, apesar de possuírem algum tipo de representação, pode ser compensatório representar essas estruturas através de outras construções dentro da própria linguagem para, posteriormente, realizar a tradução.

Na linguagem NCL os templates de composição hipermídia são processados antes da apresentação dos documentos, adicionando o valor semântico destinado às composições. Apesar dos templates hipermídia facilitarem a representação de algumas construções XMT-O (como as composições, na conversão de documentos NCL para a linguagem XMT-O), se documentos possuírem templates, eles deverão ser previamente processados. A arquitetura para o

processamento de templates NCL pode ser encontrado na referência (Muchaluat-Saade, 2003).

Na linguagem XMT-O as áreas funcionais *DEFS* e *Macros*, definidas no Capítulo 2, possuem mecanismos para facilitar a autoria através da reutilização de declarações das estruturas da linguagem. Como essas estruturas não possuem representação na linguagem NCL, durante a conversão de documentos XMT-O seus elementos devem ser substituídos pelas construções reais que eles representam (ISO/IEC, 2001).