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.

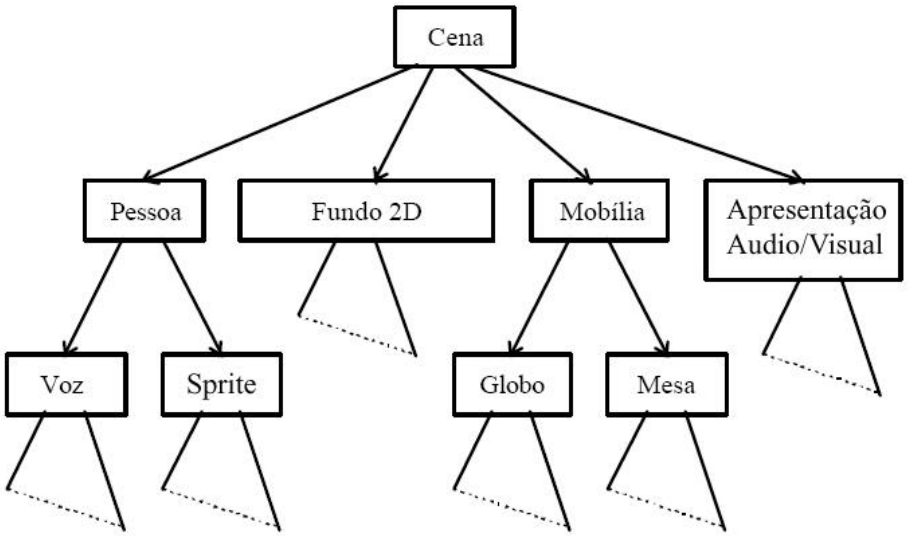

Figura 2.1 – Estrutura de uma cena MPEG-4

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.

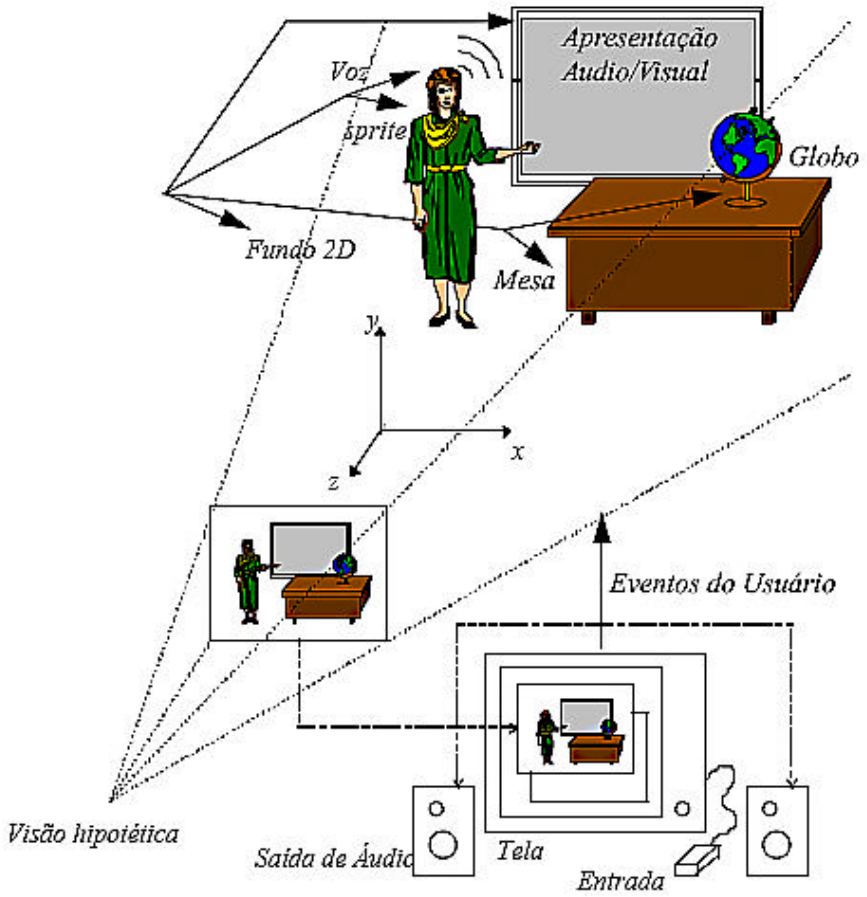

Figura 2.2 – Exemplo de uma cena MPEG-4

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.

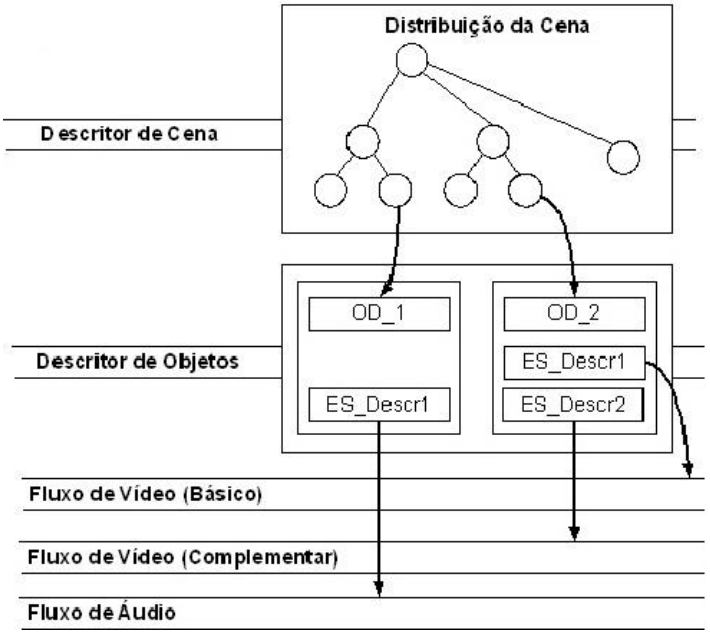

Figura 2.3 – Fluxos MPEG-4

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].

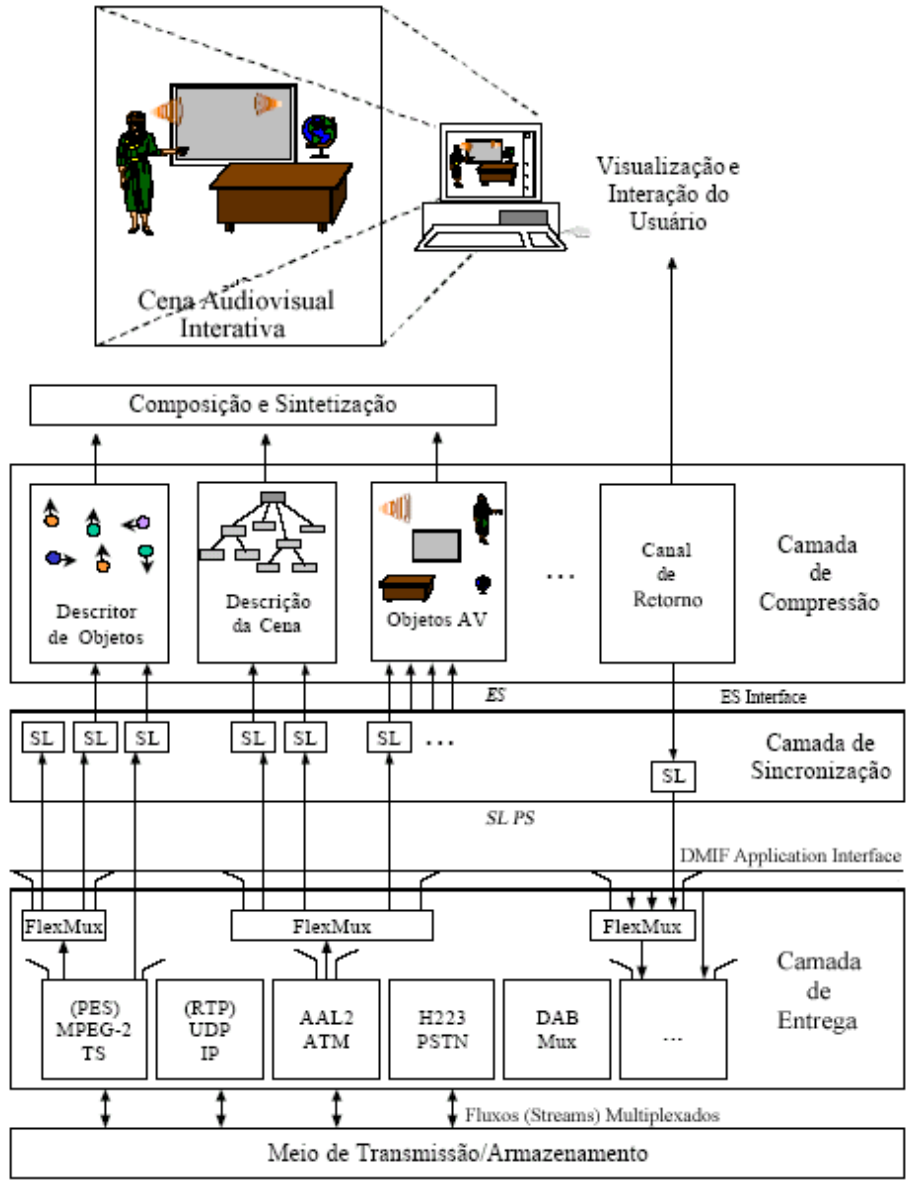

Figura 2.4 – Arquitetura de componentes MPEG-4

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.