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.

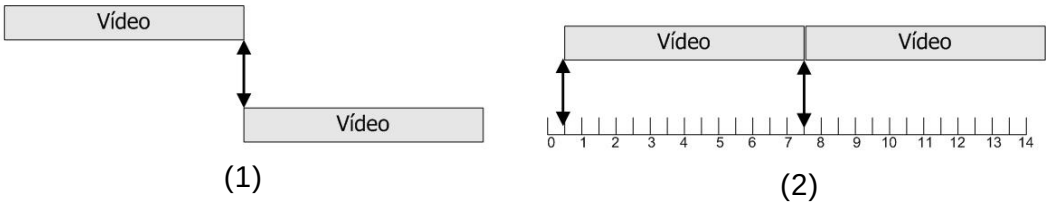

Figura 3.5 – Relacionamentos MPEG-4 em XMT-O e XMT-A

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.

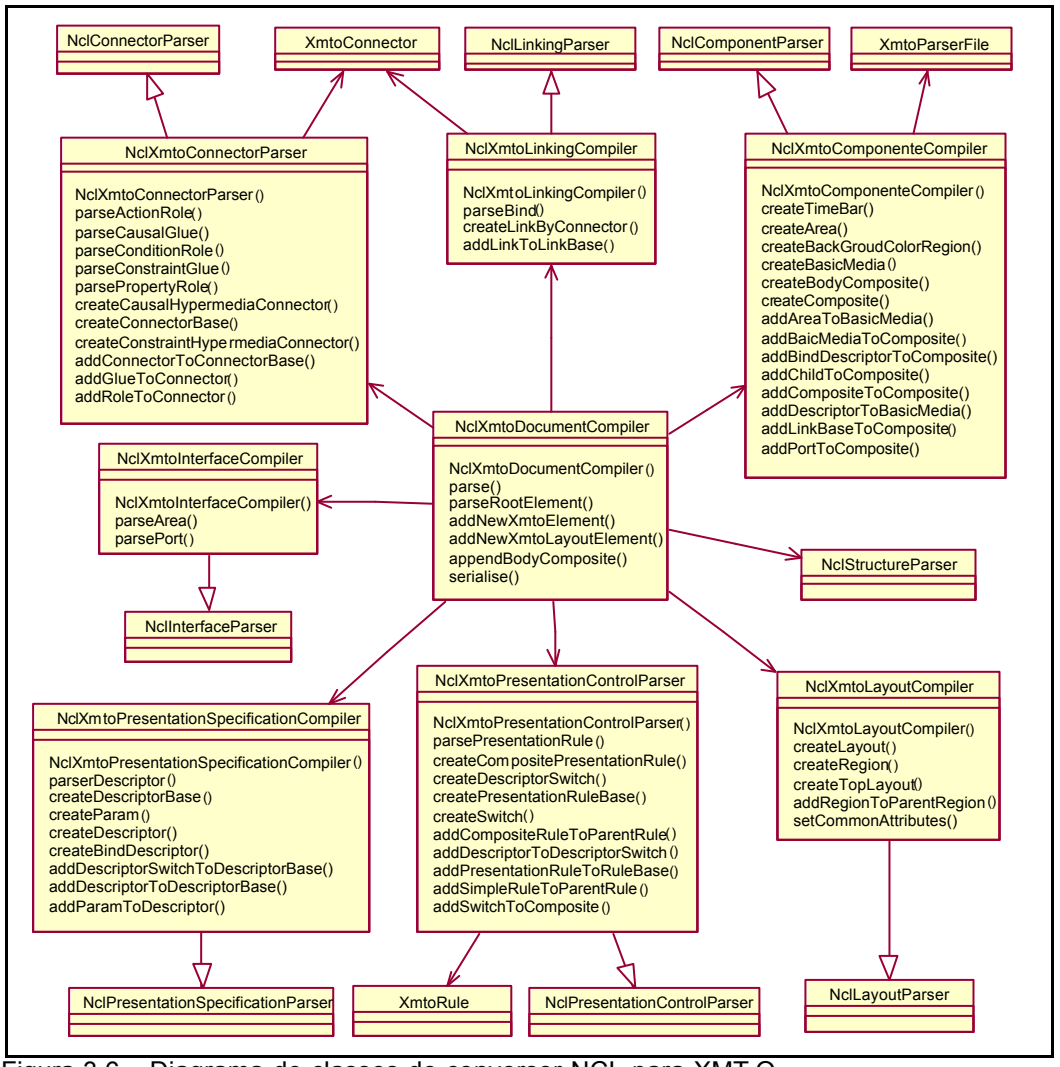

Figura 3.6 – Diagrama de classes do conversor NCL para XMT-O

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.

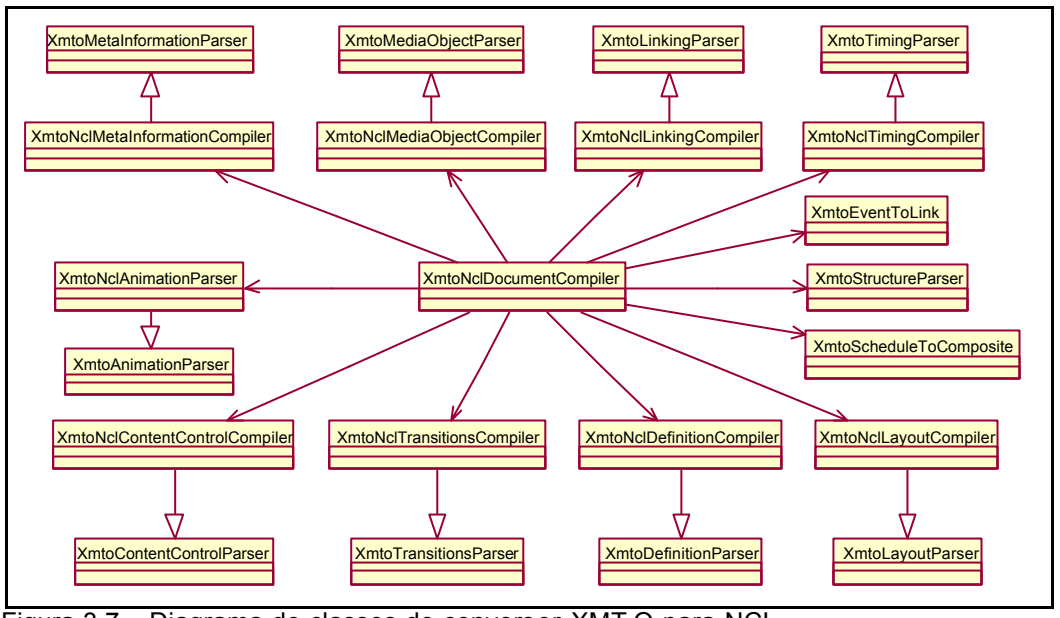

Figura 3.7 – Diagrama de classes do conversor XMT-O para NCL

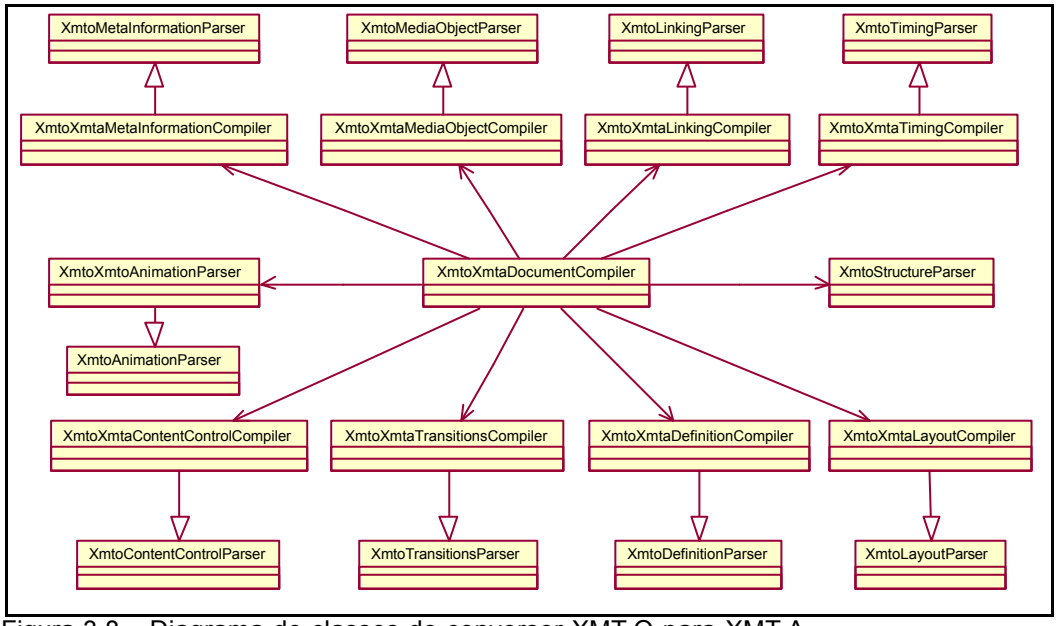

Figura 3.8 – Diagrama de classes do conversor XMT-O para XMT-A

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.

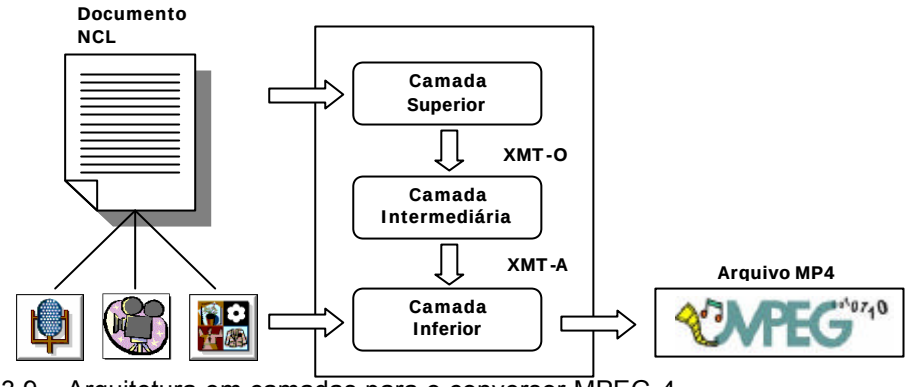

Figura 3.9 – Arquitetura em camadas para o conversor MPEG-4

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="text Region1"
      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.

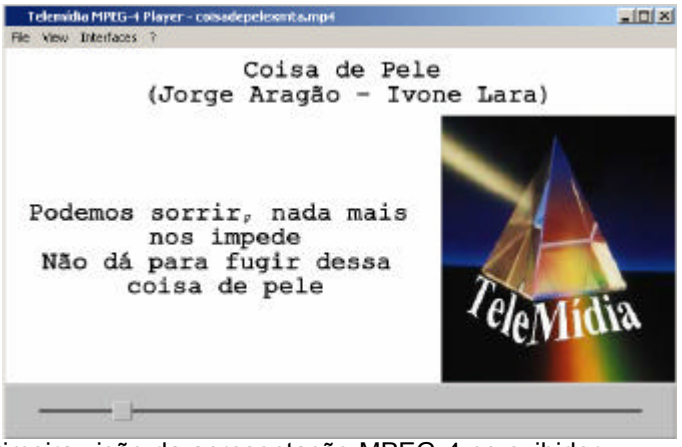

Figura 3.12 – Primeira visão da apresentação MPEG-4 no exibidor

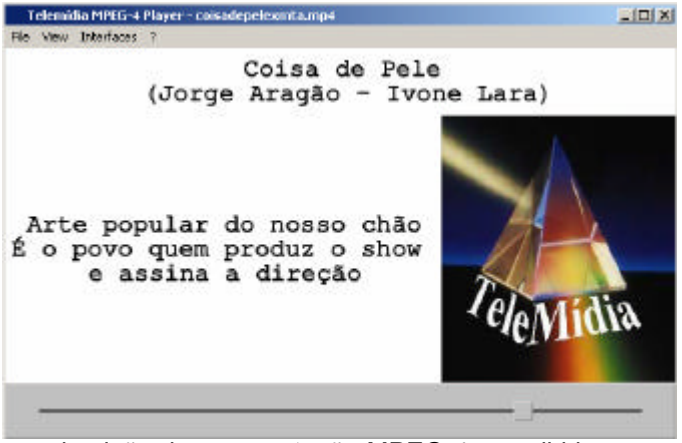

Figura 3.13 – Segunda visão da apresentação MPEG-4 no exibidor