Universidade Federal Fluminense
Instituto de Computação
Graduação em Ciência da Computação

Douglas Paulo de Mattos

Exibição da visão temporal de aplicações NCL para TV digital

Niterói-RJ
2013

DOUGLAS PAULO DE MATTOS

EXIBIÇÃO DA VISÃO TEMPORAL DE APLICAÇÕES NCL PARA TV DIGITAL

Trabalho de conclusão de curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Orientadora: Profa. Dra. Débora Christina Muchaluat Saade

Niterói-RJ

2013

DOUGLAS PAULO DE MATTOS

EXIBIÇÃO DA VISÃO TEMPORAL DE APLICAÇÕES NCL PARA TV DIGITAL

Trabalho de conclusão de curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Aprovado em Março de 2013.

BANCA EXAMINADORA

Prof.ª D.Sc. DÉBORA CHRISTINA MUCHALUAT SAADE - Orientadora
UFF

Prof. D.Sc. LEONARDO GRESTA PAULINO MURTA
UFF

Prof. D.Sc. IGOR MONTEIRO MORAES
UFF

Niterói-RJ
2013

Dedico este trabalho aos meus pais, Solange Paulo de Mattos e Levi Neves de Mattos, pelo constante apoio ao longo de toda minha educação e por estarem sempre ao meu lado e me incentivando ir em busca dos meus sonhos; à minha irmã, Cristiane Paulo de Mattos, que também sempre me apoiou dando forças nos momentos de saudades e dificuldades. E aos meus sobrinhos, Yan Paulo Clímaco e Nícolas Mattos Clímaco.

Agradecimentos

Agradeço, primeiramente, aos meus pais e irmã: Solange Paulo de Mattos, Levi Neves de Mattos e Cristiane Paulo de Mattos, por sempre estarem ao meu lado e me incentivando durante a realização deste trabalho, principalmente, nos momentos de dificuldades, me dando enorme força e energia para concretização deste trabalho da melhor forma possível.

À Professora Dra. Débora Christina Muchaluat Saade, pela excelente orientação, enorme dedicação e paciência, sempre incentivando para obter melhores resultados.

Aos amigos da faculdade que também me incentivaram e apoiaram no desenvolvimento do trabalho e a todos aqueles que, de alguma forma, contribuíram para a conclusão deste trabalho.

Ao grupo de TV Digital do Laboratório MídiaCom pelas ideias sugeridas na concepção deste projeto e, principalmente, ao Joel André Ferreira dos Santos pela disposição e enorme interesse em me orientar na implementação da ferramenta proposta neste trabalho; e a Júlia Varanda da Silva pelas experiências e conhecimentos técnicos adquiridos com ela que também me ajudaram no desenvolvimento deste projeto.

Power is not given to you, you have to take it. You play a part in a much bigger show. And that's what life is, it's the greatest show on earth. Life is but a dream.

Knowles, B.

Lista de Figuras

Lista de Tabelas

Sumário

Resumo

Em documentos NCL (*Nested Context Language*), para garantir que as mídias sigam uma programação de apresentação, deve-se construir elos que definem relacionamentos temporais entre os nós de mídia de acordo com a ocorrência de eventos. Este processo de encadear os nós mídia de um documento hipermídia no tempo é chamado de sincronismo temporal. E a representação, no tempo, da sequência de eventos previsíveis deste documento é denominada de cadeia temporal. Este trabalho discute uma estratégia de representação da cadeia temporal de documentos NCL e apresenta um editor gráfico para exibição da visão temporal de aplicações NCL. A interatividade nas aplicações multimídia torna o processo de representação da visão temporal uma tarefa não trivial, visto que existem eventos que dependem da interação do usuário com a aplicação, o que pode levar a diferentes representações na cadeia temporal. Esta questão é tratada neste trabalho. Neste trabalho, a implementação da ferramenta focou-se no modelo de dados baseado no HTG (*Hypermedia Temporal Graph*), que armazena as informações necessárias de um documento NCL para construir sua visão temporal, e na geração da cadeia temporal a partir destes dados. A ferramenta faz parte de um editor gráfico para autoria de documentos NCL com suporte a templates de composição, chamado NEXT (*NCL Editor Supporting XTemplate*), que facilita a construção de aplicações multimídia interativa para o sistema brasileiro de TV digital.

Palavras-chave: Aplicações hipermídia. TV digital. Documentos NCL. Visão Temporal. Sincronismo Temporal.

Abstract

In NCL documents, to ensure that media nodes follow a presentation sequence, it is necessary to define links, which specify temporal relationships among media nodes according to event occurrences. The process of linking hypermedia document nodes in time, is called temporal synchronization. This work discusses a temporal representation strategy and presents a graphical editor to display the temporal view of NCL applications. Interactivity in multimedia applications makes the process of representing the temporal view a non-trivial task, since there are events, which depend on user interaction, that can lead to different temporal representations. This issue is handled in our strategy. The proposed tool allows displaying a temporal view of NCL documents. In this work, the tool implementation focused on the data model based on HTG (Hypermedia Temporal Graph), which stores necessary information of NCL documents to create its temporal view, and on the timeline generation from these data. In a near future, the tool will be integrated into a graphical editor for authoring NCL documents supporting composite templates, which is called NEXT (NCL Editor Supporting XTemplate). NEXT facilitates the development of interactive multimedia applications for the Brazilian digital TV system.

Keywords: Hypermedia applications. Digital TV. NCL Documents. Temporal view. Temporal synchronization.

Capítulo 1

Introdução

Documentos hipermídia são compostos de objetos de mídia (texto, imagem, áudio, vídeo etc.) e relacionamentos entre eles. Dentre as relações hipermídia, a relação de sincronização é a que define o posicionamento temporal e espacial dos objetos de mídia, permitindo construir uma apresentação destes componentes a partir de um modelo conceitual hipermídia. O sincronismo temporal de objetos de mídia pode ser baseado em diversos paradigmas. Um deles é o paradigma baseado em eventos, empregado nas aplicações multimídia interativas para a TV digital brasileira.

O sistema de televisão digital interativa no Brasil utiliza a tecnologia de transmissão do padrão japonês e acrescenta inovações em algumas camadas do sistema de TV digital [2]. Na camada de middleware é apresentado o diferencial do padrão brasileiro, onde é especificado o GINGA, que pode utilizar linguagem declarativa ou procedural. No ambiente declarativo, utilizam-se as linguagens NCL (*Nested Context Language*) [32] e Lua [21] para o desenvolvimento de aplicações interativas, ambas desenvolvidas por universidades brasileiras.

1.1 Motivação

A linguagem NCL é baseada no modelo conceitual hipermídia NCM (*Nested Context Model*) [34] que permite estruturar logicamente um documento através de contextos (conjuntos de nós e elos) e faz uso do paradigma baseado em eventos para relacionar os objetos de mídia temporal e espacialmente. O modelo baseado em eventos é bastante expressivo para autoria de documentos multimídia interativos. Assim, a sincronização dos objetos de mídia (nós de contexto, vídeo, áudio, imagem, texto, programas Lua, etc.) é feita através de elos, cuja semântica da relação é definida por conectores hipermídia [24].

A visão temporal de um documento apresenta a programação de apresentação de seus objetos de mídia ao longo do tempo e permite ao autor visualizar a duração dos objetos e entender como estão relacionados temporalmente. Além disto, a possibilidade de editar graficamente a visão temporal de uma determinada aplicação facilita o processo de desenvolvimento de aplicações multimídia. A visão temporal de uma aplicação multimídia pode ser composta por várias cadeias temporais [26]. Cada cadeia temporal representa, ao longo do tempo, os eventos síncronos que acontecem durante a apresentação de

parte do documento, ou seja, aqueles que não dependem da interação do usuário. Quando existe um relacionamento que depende da interação do usuário, uma nova cadeia temporal é gerada apresentando a sequência dos eventos previsíveis relativos a interatividade. A Figura 1.1 apresenta a visão temporal de um documento simples onde um vídeo é exibido inicialmente e depois de alguns instantes uma imagem de um botão é apresentada indicando a possibilidade de interação do usuário. Caso ele interaja, outra imagem é mostrada apresentando informações sobre o vídeo e a exibição da imagem do botão é parada. Esta visão possui duas cadeias temporais (cadeia temporal 1 e 2) devido à ação interativa de seleção do botão.

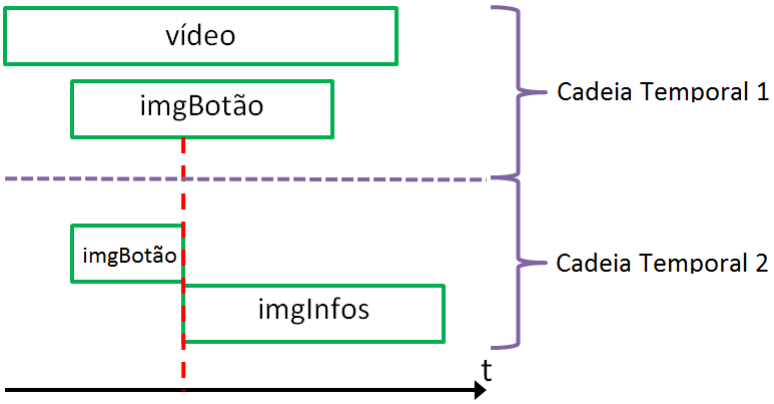


Figura 1.1: Visão temporal de um documento hipermídia

A representação da visão temporal de um documento NCL é uma tarefa não-trivial [18] [15], uma vez que aplicações NCL podem ser interativas, e diversas cadeias temporais são necessárias. Em um determinado instante da aplicação, há a necessidade da interação do usuário para conhecer, ao certo, o comportamento do documento adiante.

1.2 Objetivos

Visto que a visualização e edição gráfica da visão temporal tem grande importância na análise comportamental das aplicações hipermídia, este trabalho tem como objetivo representar a cadeia temporal de documentos NCL com interatividade propondo uma ferramenta gráfica para exibição da visão temporal de aplicações NCL para o sistema brasileiro de televisão digital e também discutir uma estratégia de edição da visão temporal. Assim, neste trabalho foi implementado o modelo de dados, que permite guardar as informações relevantes de um documento NCL para gerar sua visão temporal, e a representação gráfica das cadeias temporais.

A ferramenta permite, facilmente, a visualização da cadeia temporal e além disto, possibilitará a alteração da ordem de apresentação das mídias graficamente. Desta forma, ela também permitirá que usuários que não possuem conhecimento na linguagem NCL possam fazer a edição da ordem temporal das mídias de um documento NCL. Esta ferramenta faz parte de um editor gráfico para autoria de documentos NCL com suporte a templates de composição [28], chamado NEXT (*NCL Editor Supporting XTemplate*) [30], que facilita a construção de aplicações multimídia interativas para o sistema brasileiro de TV digital.

1.3 Estrutura da Monografia

A estrutura do trabalho é como se segue.

No Capítulo 2, são discutidos alguns conceitos básicos sobre modelos hipermídia e paradigmas para sincronização temporal. Além disto, o modelo NCM é introduzido, enfatizando sua sincronização temporal. A estrutura de um documento NCL e suas visões são apresentadas.

No terceiro capítulo são comentados alguns trabalhos relacionados, apresentando seus modelos de dados e interface gráfica para representar a visão temporal de documentos multimídia. Também são discutidas algumas ferramentas comerciais que representam o comportamento temporal de aplicações multimídia não-interativas. Ainda neste capítulo, é abordado o editor NEXT, apresentando sua arquitetura, seus *plugins* e como desenvolvê-los, e sua interface gráfica.

O editor gráfico da visão temporal de aplicações NCL é apresentado no Capítulo 4, discutindo a estratégia de exibição da cadeia temporal de aplicações com interatividade. O modelo de dados da ferramenta também é estudado neste capítulo, analisando as estruturas de dados implementadas para armazenar as informações do comportamento temporal do documento NCL. Tais estruturas são as seguintes: o HTG (*Hypermedia Temporal Graph*) [15], o plano de apresentação e a cadeia temporal. Ademais, é feito um estudo de caso para demonstrar o funcionamento da ferramenta. Ao final do capítulo, é mostrado um protótipo da interface gráfica do editor, visto que este trabalho concentrou-se no desenvolvimento do modelo de dados.

No quinto capítulo, é discutida a implementação da ferramenta. São apresentados os requisitos do sistema e os diagramas de pacotes, de classes e de atividade. Em seguida, são analisados todos os pacotes em detalhes por funcionalidade (adaptação do editor ao NEXT, leitura do documento NCL, construção do HTG, do plano de apresentação, da visão temporal, interface gráfica do usuário e os pacotes utilitários). Neste capítulo, ainda é discutida a API aNa [27] que possibilita a leitura de documentos NCL.

Finalmente, no Capítulo 6, as contribuições são apresentadas, é feita uma análise comparativa com os trabalhos relacionados e projetos futuros são apontados.

Capítulo 2

Conceitos Básicos

Neste capítulo são apresentado alguns conceitos sobre sistemas hipermídia, a linguagem NCL e o modelo NCM, no qual ela é baseada para representar os documentos multimídia interativos para o Sistema Brasileiro de TV Digital.

2.1 Sistemas Hipermídia

Nesta seção, são apresentados alguns fundamentos de sistemas hipermídia como suas definições e descrição sobre seus elementos. Além disso, são estudados os modelos hipermídia usados para especificar os documentos multimídia e os paradigmas para sincronização temporal utilizados nos diversos modelos de dados para definir a ordem de apresentação das aplicações multimídia.

2.1.1 Definição

Sistemas multimídia manipulam mídias discretas ou contínuas, digitalmente, relacionando-as de forma a gerar uma sequência de apresentação das mesmas e também definindo de que modo elas são exibidas. Já sistemas hipermídia, além de lidar com mídias discretas ou contínuas como os sistemas multimídia, tratam a interação de usuários, isto é, alteram o fluxo da apresentação dos documentos hipermídia (multimídia interativos) de acordo com a ocorrência da interatividade.

Os documentos hipermídia possuem os seguintes elementos: os objetos de mídia que representam o conteúdo do documento (textos, imagens, vídeos, áudios etc.) e os relacionamentos entre estes objetos, como demonstrado na Figura 2.1.

2.1.2 Modelos Hipermídia

Os componentes de um documento hipermídia apresentados na seção anterior precisam ser expressados através de um modelo conceitual hipermídia. Assim, tais componentes são representados pelas entidades do modelo, sendo as principais, os nós, elos e nós de composição.

Os nós representam os objetos de mídia e cada um pode possuir âncoras que permitem criar relacionamentos entre as mídias do documento. Estas âncoras definem um trecho do conteúdo da mídia

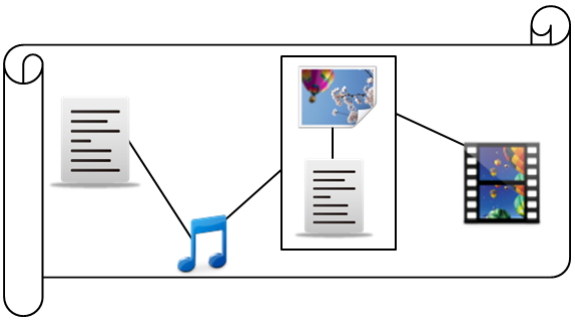


Figura 2.1: Representação de um documento hipermídia

que pode participar de um relacionamento com outra âncora de diferentes nós. O segmento definido por ela depende do tipo de mídia. Ou seja, se o conteúdo de uma mídia é um vídeo, seu trecho pode ser especificado através de um intervalo de tempo ou ainda pelo quadro inicial e final do trecho. No caso de áudio, também pode ser definido por um intervalo de tempo e, além disso, pela amostra inicial e final. Para imagem, um conjunto de pixels formando uma área fechada da mesma (área retangular definido por quatro coordenadas) define o trecho. E por fim, mídias do tipo texto têm o segmento especificado através da posição inicial e final no texto. Todas as âncoras de um nó definem sua interface.

Os elos de um documento hipermídia são utilizados para relacionar as âncoras (ponto de interface) de nós distintos. Estes elos permitem representar os relacionamentos de sincronização, os quais definem o posicionamento espacial e temporal dos objetos. Estes elos podem ter como origens e destinos diversos nós, sendo um elo multiponto.

Os nós de composição possibilitam o agrupamento de diversos nós somente ou também de diferentes nós e elos. Como os nós individuais, um nó de composição pode ter âncoras para permitir o relacionamento diretamente com nós de composição, representado pelo ponto de interface b simbolizado na Figura 2.2 e também possibilitar o elo entre nós internos (interface c na Figura 2.2) de um nó de composição com nós externos no documento. No caso de um ponto de interface que oferece acesso externo a um conteúdo de um nó de composição, a interface é chamada de porta.

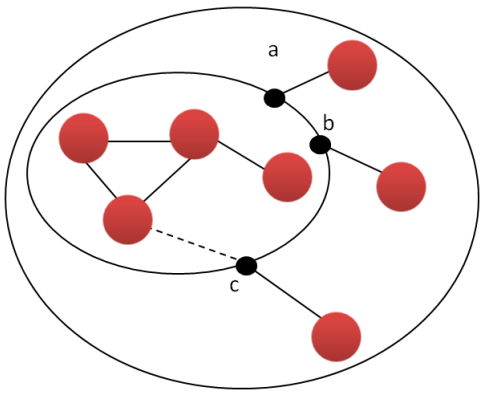


Figura 2.2: Nós de composição

Assim, os nós de composição são utilizados para estruturar um documento agrupando nós e elos, tornando-o mais organizado. Ademais, eles ainda permitem representar relações de sincronização

agrupando nós que são apresentados em sequência ou em paralelo, como na linguagem SMIL [3].

2.1.3 Paradigmas para Sincronização Temporal

Os relacionamentos temporais entre as mídias podem ser baseados nos seguintes paradigmas: scripts, *timeline* (linha do tempo), especificação formal, hierárquica, restrições e eventos. Adiante, serão analisados cada um dos modelos para sincronização temporal.

Sincronização baseada em Script

Este modelo faz o uso de programação em *scripts* para definir o relacionamento temporal entre as mídias do documento. Ele é bastante expressivo, já que usa linguagem imperativa para especificar o comportamento do documento. Porém, exige conhecimento em programação do autor do documento e a visualização de como as mídias estão organizadas na apresentação hipermídia torna-se difícil. *Adobe Flash* [5] é exemplo de ferramenta que faz o uso de scripts para criar apresentações multimídia.

Sincronização baseada em timeline

No caso da sincronização baseada em *timeline*, as mídias são posicionadas diretamente no instante que elas devem iniciar sua apresentação como mostrado na Figura 2.3. Este paradigma é muito intuitivo na criação de apresentações hipermídia como também na sua visualização temporal. Sendo assim este modelo é utilizado por várias ferramentas comerciais como visto na seção de trabalhos relacionados. Entretanto, esse modelo á bastante criticado na literatura por não representar relações temporais diretamente entre os nós que compõem o documento. Desta forma, não é possível garantir a sincronização temporal durante a exibição do documento realizando algum ajuste temporal, caso haja algum atraso na exibição de um nó específico.

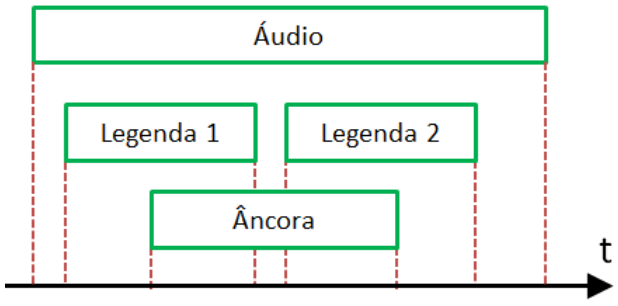


Figura 2.3: Timeline

Sincronização baseada em especificação formal

Este paradigma utiliza modelos formais para representar os relacionamentos temporais entre as mídias. Como exemplo destes modelos formais, têm-se *Flowcharts* e as *Redes de Petri*. O primeiro modelo é um diagrama que representa um processo, descrevendo passo-a-passo, no caso da sincronização temporal em sistemas hipermídia, o comportamento das mídias do documento através de elementos gráficos que

representam ações e pontos de decisão conectados por setas. Assim, construindo o fluxo de controle da apresentação. Como exemplo do uso deste modelo, tem-se o *Macromedia Authorware* [1]. Esta representação pode ser vista na Figura 2.4.

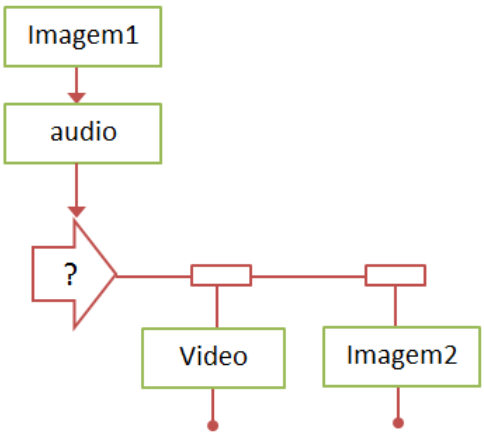


Figura 2.4: Flowchart

As *redes de Petri* são baseadas num grafo dirigido que possui como elementos: os *lugares*, *setas* e *transições*. Cada *lugar* possui duração e uma transição é disparada somente se todos seus *lugares de entrada* contiverem *tokens*. Quando ela é executada, os *tokens* são movidos para seus *lugares de saída* e ficam bloqueados neles no intervalo de tempo que corresponde a duração dos mesmos. Um exemplo de uso desta notação formal é o *HTSPN* [37]. A Figura 2.5 representa uma rede de Petri.

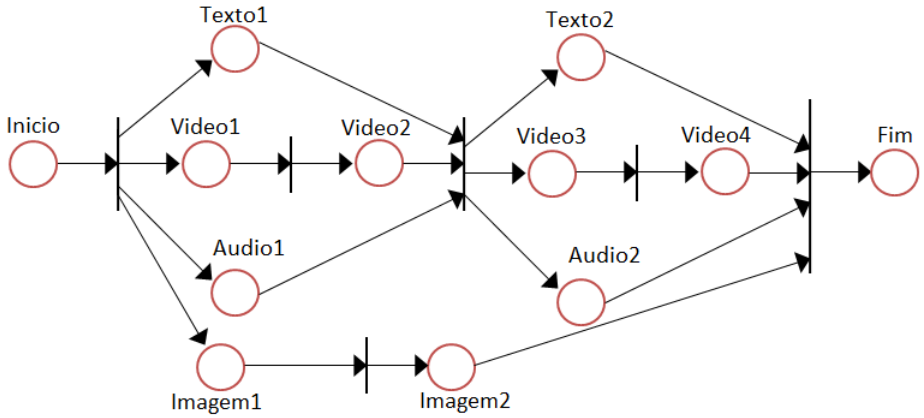


Figura 2.5: Rede de Petri

Este paradigma é bastante expressivo e permite modificações durante a execução da apresentação quanto a verificação formal do comportamento temporal do documento. Entretanto, ele requer um conhecimento das notações formais utilizadas e, nos casos de relacionar temporalmente partes de mídias, o modelo formal pode se tornar complexo.

Sincronização hierárquica

O modelo de sincronização hierárquico faz o uso de composições que definem a relação temporal entre seus conteúdos. Há dois tipos de composição: sequencial e paralela. Na primeira, todos os nós presentes dentro da composição são apresentados sequencialmente e no caso da segunda, os nós são exibidos simultaneamente no tempo. Na seção de trabalhos relacionados, foram analisados dois sistemas que empregam este paradigma, o *CMIFed* e *GRiNs*. A Figura 2.6 mostra a hierarquia dos nós de um documento.

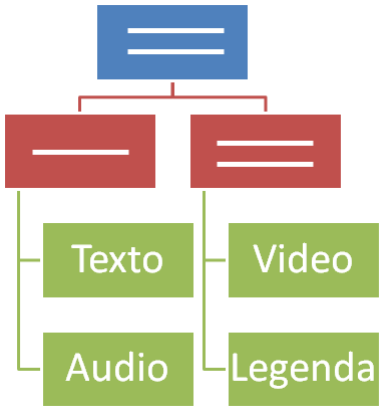


Figura 2.6: Sincronização Hierárquica

O modelo permite fazer alterações durante a execução, além de facilitar a autoria de documentos hipermídia e ser bastante expressivo. Contudo, os eventos de interatividade são dificilmente expressados por este modelo e o relacionamento entre partes de mídias também é dificultado.

Sincronização baseada em restrições

Este paradigma utiliza as definições das treze relações básicas de Allen [11] entre instantes ou intervalos de tempo. O modelo facilita a autoria de documentos hipermídia, é bastante expressivo e também permite fazer ajustes durante a execução. Porém, assim como o modelo hierárquico, possui limitações para representar interatividade e sincronização entre partes de mídias. O sistema *FireFly*, visto na seção de trabalhos relacionados, faz uso deste paradigma. A Figura 2.7 exibe como nós são ordenados no tempo com as relações de [11].

Sincronização baseada em eventos

Este modelo se baseia na ocorrência de eventos durante a execução do documento para especificar os elos temporais entre os nós. Tais eventos podem ser de apresentação (*start*, *stop* e *pause* de um nó), seleção (interatividade do usuário) ou atribuição (alteração do valor de uma propriedade de um objeto de mídia). Um exemplo de uso deste paradigma é o modelo NCM, utilizado pela linguagem NCL, para expressar documentos hipermídia. Ambos serão vistos nas próximas seções. Na seção de trabalhos relacionados, o Composer faz uso deste modelo, visto que o mesmo é uma ferramenta para autoria de documentos hipermídia pra TV digital que utiliza a linguagem NCL para especificar as aplicações

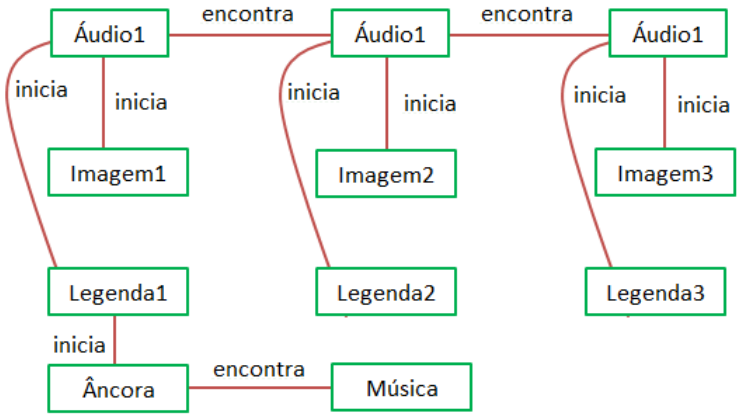


Figura 2.7: Sincronização baseada em restrições

multimídia interativas. Na Figura 2.8, são mostradas as relações temporais entre os objetos de mídia baseadas em eventos.

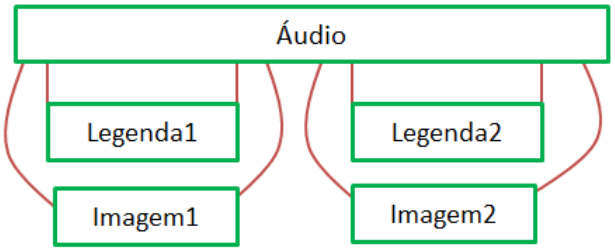


Figura 2.8: Sincronização baseada em eventos

Este modelo é bastante expressivo na definição dos relacionamentos temporais entre as mídias e facilmente trata a interatividade, além de permitir modificações durante a execução do documento.

2.2 O Modelo NCM

Esta seção descreve o modelo conceitual hipermídia NCM (*Nested Context Model*) [34] em sua versão 3.0. As principais características do modelo para este trabalho, como suas entidades, a máquina de estados de eventos e sua sincronização temporal, são descritas adiante.

2.2.1 Entidades

O modelo NCM possui as entidades nós (*node*) e elos (*links*), as quais são baseadas nos conceitos de modelos hipermídia estudados na Seção 2.1.2.

O nós (*node*) NCM se dividem em duas classes: nós de mídia ou de conteúdo (*content nodes*) e nós de composição (*composite nodes*). Os nós de mídia tem como propriedades básicas o conteúdo da mídia, uma lista de âncoras e um descritor (propriedade opcional). Uma âncora define um trecho da mídia de acordo com o tipo da mesma como dito na subseção 2.1.2. Este segmento de mídia é denotado como âncora de conteúdo. Neste modelo o conteúdo total da mídia é considerado uma âncora. Além

deste tipo, ela pode ser uma âncora de propriedade que define atributos associado a um nó.

Os nós de composição, no modelo NCM, são representados pelos nós de contexto (*context node*) e nós switch (*switch node*). Os nós de contexto podem conter nós de mídia, outros de contexto, nós switch e elos (*links*). Os nós switch são responsáveis por oferecer adaptação de conteúdo ao documento.

Os elos (*links*) NCM possuem duas propriedades: os conectores e associações (*binds*) a esse conector. Ou seja, o conector é a entidade que define a semântica de um relacionamento entre nós de um documento hipermídia, especificando papéis como seus pontos de interface. Os elos, por sua vez, são responsáveis por associar pontos de interface de nós (âncoras e atributos) aos papéis definidos pelo conector através de *binds*. Um conector pode ser reutilizados por vários elos que possuem a mesma semântica.

2.2.2 Máquina de Estado

Um evento é uma ocorrência no tempo que pode ser instantânea ou ter uma duração mensurável segundo Pérez-Luque e Little [25]. O modelo NCM define os seguintes tipos de eventos: evento de apresentação, evento de seleção e evento de atribuição. São definidos outros tipos de evento, entretanto, somente esses são relevantes para este trabalho.

O *evento de apresentação* é relativo a exibição de uma âncora de conteúdo e também de um nó de composição (contexto e switch) indicando a apresentação de todo o conteúdo da composição.

Um *evento de seleção* representa a seleção de uma âncora de conteúdo, que está sendo exibida, pelo usuário. É o único evento que depende da interação.

Um *evento de atribuição* modifica o valor de uma âncora de propriedade de um objeto de mídia.

Para cada um destes eventos descritos, o NCM define uma máquina de estados indicando em qual estado um determinado evento se encontra. Tais estados podem ser os seguintes: *ocorrendo* (*occurring*), *pausado* (*paused*) e *dormindo* (*sleeping*). Além disso, os eventos possuem algumas propriedades como: *ocorrências* (*occurrences*), que conta o número de vezes que o evento passou do estado *ocorrendo* para o *dormindo* ao longo da apresentação de um documento hipermídia, e *repetições* (*repetitions*), somente para os eventos de atribuição e apresentação, que especifica a quantidade de vezes que o evento deve ocorrer. A Figura 2.9 mostra a máquina de estados de eventos do modelo NCM.

É importante ressaltar que a duração de um evento é o tempo que o mesmo permanece no estado ocorrendo. Tal duração pode ser intrínseca ao objeto de mídia, explicitamente especificada pelo autor do documento ou ainda derivada de um relacionamento.

2.2.3 Sincronização Temporal

A sincronização temporal no modelo NCM é baseada em eventos que ocorrem durante a apresentação de um documento. O elemento conector, utilizado pelos elos, é responsável por especificar as relações de sincronização espaço-temporais. A seguir são estudados pontos para o entendimento básico de como o modelo NCM especifica os relacionamentos entre os objetos de um documento hipermídia.

As relações definidas pelos conectores NCM podem ter semântica causal ou de restrição. Porém,

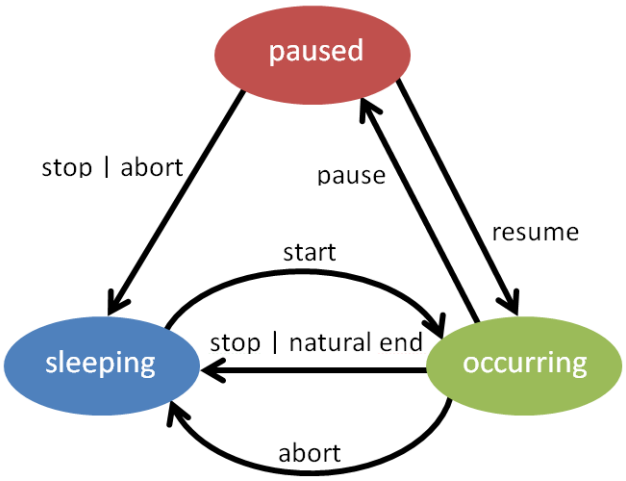


Figura 2.9: Máquina de estado de eventos

neste trabalho, são analisados somente os conectores causais, os quais foram levados em consideração na implementação da ferramenta proposta. Então, nos conectores causais, uma condição deve ser satisfeita para que uma ação seja executada. Estes conectores definem papéis (*roles*) que especificam como cada nó associado a eles devem participar na relação causal. Os papéis ainda possuem um identificador, tipo de evento e sua cardinalidade indicando o número mínimo e máximo de nós que podem ser associados ao papel. O tipo de evento (*event type*) corresponde a um dos eventos apresentados na Seção 2.2.2.

Os papéis podem ser do tipo condição (*condition*), ação (action) ou avaliação (*assessment*). Quanto aos papéis de ação (*action roles*), eles são responsáveis por causar uma transição na máquina de estado de um evento. A Tabela 2.1 descreve cada uma das transições disparadas pelas tipos de ações definidos pelo modelo na máquina de estado.

Tabela 2.1: Transições na Máquina de estado

Transição (Causada pela Ação)	**Nome da Transição**
sleeping –> occurring (start)	*starts*
occurring –> sleeping (stop)	*stops*
occurring –> sleeping (abort)	*aborts*
occurring –> paused (pause)	*pauses*
paused –> occurring (resume)	*resumes*
paused –> sleeping (stop)	*stops*
paused –> sleeping (abort)	*aborts*

Em relação aos papéis de condição (*condition roles*), estas devem ser satisfeitas para executar a ação definida pelo conector. Tanto as condições quanto as ações podem ser simples ou compostas.

Uma condição simples (*simple condition*) testa uma transição de estado de um evento e os valores de um atributo de um nó ou evento. A condição composta é uma expressão lógica que envolve duas ou mais condições simples ou compostas ligadas pelos operadores *and* ou *or*.

Uma ação simples refere-se a um papel do tipo ação. A ação composta contém duas ou mais ações simples ou compostas ligadas pelos operadores *seq* ou *par* . O primeiro indica que as ações da composição são disparadas em uma determinada sequência e o segundo aponta que elas devem ser executadas em qualquer ordem. Além disso pode também ser especificado um *delay* indicando o intervalo de tempo que deve ser obedecido para então executar a ação de um elo após a condição definida pelo conector ser satisfeita.

Para exemplificar, considere um elo entre duas mídias (*vídeo* e *áudio*), que especifica que o término do vídeo deve ser sincronizado com o início do áudio. No relacionamento, a condição simples é o término da apresentação do *vídeo*, ou seja, a ocorrência da transição *stops* da máquina de estado do evento de apresentação deste objeto. A ação simples a ser disparada é iniciar a apresentação do objeto *áudio* (transição *starts*).

2.3 A Linguagem NCL

A linguagem NCL (*Nested Context Language*) [4] é baseada no modelo NCM e na linguagem XML, ou seja, ela usa elementos XML para definir as entidades do modelo NCM para especificar um documento hipermídia. A seguir serão estudadas a estrutura básica de um documento NCL e suas diferentes visões.

2.3.1 Estrutura Básica do Documento

O documento NCL possui, basicamente, duas partes: cabeçalho (*head*) e o corpo (*body*). A primeira contém as bases de região (*regionBase*), que possuem as regiões da tela onde as mídias são apresentadas, a base de descritores (*descriptorBase*) que define características de apresentação das mídias, e as bases de conectores (*connectorBase*) que especificam os conectores utilizados pelos elos do documento. A seguir é mostrado o cabeçalho de um documento NCL.

```
<?xml version=''1.0'' encoding=''ISO-8859-1''?>
<ncl id=''exemplo'' xmlns=''http://www.ncl.org.br/NCL3.0/EDTVProfile''>
    <head>
        <regionBase>
                    <region id=''backgroundReg'' width=''100%''
                    height=''100%'' zIndex=''1''/>
                    (...)
        </regionBase>

        <descriptorBase>
                    <descriptor id=''backgroundDesc''
                    region=''backgroundReg''/>
                    (...)
        </descriptorBase>
```

```
        <connectorBase>
                <causalConnector id=''onBeginStart''>
                    <simpleCondition role=''onBegin''
                    max=''unbounded'' qualifier=''and''/>
                    <simpleAction role=''start''
                    max=''unbounded'' qualifier=''par''/>
                </causalConnector>
        </connectorBase>
    </head>
    (...)
</ncl>
```

O corpo contém os nós de mídia (*media*), contextos (*context* e *switch*), no mínimo uma porta (*port*) indicando o início da apresentação do documento e os elos (*link*).

Para os nós podem ser definidas interfaces, a fim de criar relacionamentos entre eles, as quais são: *area* que define uma âncora de conteúdo (trecho de mídia), *port* que mapeia uma interface ou componente do contexto, *property* que define uma âncora de propriedade e *switchPort* que mapeia uma interface dos componentes de um *switch* (contexto com nós alternativos, isto é, apenas um deles é ativado na apresentação).

O elemento *link* possui *binds* cujos atributos principais são *role* e *component* que realizam a ligação dos nós com seus papéis na semântica da relação causal do conector usado pelo elo, o qual é indicado pelo atributo *xconnector* do *link*. Os papéis padrões, tanto para condição quanto ação, são mostrados na Tabela 2.2.

Além destas condições predefinidas, é possível construir outras que testem o estado de apresentação de um evento, atributos associados a eventos ou valores de propriedade. Isto é feito através do elemento *assessmentStatement*. Ele possui ainda os elementos *attributeAssessment* que define um atributo cujo valor é testado e o elemento *valueAssessment* que define o valor.

Na linguagem NCL, as condições simples e compostas são representadas pelos elementos *simpleCondition* e *compoundCondition* respectivamente. Este último contém uma ou mais condições simples ou compostas e ainda pode possuir um ou mais elementos *assessmentStatement*. Já as ações simples e compostas são simbolizados pelos elementos *simpleAction* e *compoundAction* respectivamente. Sendo que uma ação composta pode conter uma ou mais ações simples ou composta.

A seguir é mostrado um documento NCL como exemplo, apresentando a estrutura básica e os principais elementos discutidos.

Valor do *role*	Valor da Transição	Tipo do evento
onBegin	*starts*	*presentation*
onEnd	*stops*	*presentation*
onAbort	*aborts*	*presentation*
onPause	*pauses*	*presentation*
onResume	*resumes*	*presentation*
onSelection	*starts*	*selection*
onBeginAttribution	*starts*	*attribution*
onEndAttribution	*stops*	*attribution*

(a) Papéis de Condição

Valor do *role*	Valor do Tipo de Ação	Tipo do evento
start	*start*	*presentation*
stop	*stop*	*presentation*
abort	*abort*	*presentation*
pause	*pause*	*presentation*
resume	*resume*	*presentation*
set	*start*	*attribution*

(b) Papéis de Ação

Tabela 2.2: Papéis Padrões

```
<?xml version=''1.0'' encoding=''ISO-8859-1''?>
<ncl id=''exemplo'' xmlns=''http://www.ncl.org.br/NCL3.0/EDTVProfile''>
    <head>
        (...)
    </head>

    <body>
        <port id=''entry'' component=''animation''/>
        <media id=''background'' src=''background.png''
        descriptor=''backgroundDesc''/>
        <media id=''animation'' src=''animGar.mpg'' descriptor=''screenDesc''>
            <area id=''segPhoto'' begin=''12s''/>
            <area id=''segIcon'' begin=''5s'' end=''10s''/>
            <property name=''bounds''/>
        </media>
        <media id=''music'' src=''choro.mp3'' descriptor=''audioDesc''/>
        <media id=''photo'' src=''photo.png'' descriptor=''photoDesc''/>
```

```
        <context id=''advert''>
            <media id=''reusedAnimation'' refer=''animation''
            instance=''instSame''>
            (...)
            <link id=''lBeginShoes''
            xconnector=''onKeySelectionStopSetStart''>
                <bind role=''onSelection'' component=''icon''>
                    <bindParam name=''tecla'' value=''RED''/>
                </bind>
                <bind role=''start'' component=''shoes''/>
                <bind role=''start'' component=''ptForm''/>
                <bind role=''set'' component=''reusedAnimation''
                interface=''bounds''/>
                <bind role=''stop'' component=''icon''/>
            </link>
            (...)
        </context>

        <link id=''lMusic'' xconnector=''onBeginStart''>
            <bind role=''onBegin'' component=''animation''/>
            <bind role=''start'' component=''music''/>
        </link>
        (...)
</body>
</ncl>
```

2.3.2 Visões do Documento

O documento NCL possui diferentes visões que auxiliam o autor a compreendê-lo. A primeira delas é a visão estrutural que mostra como os nós estão relacionados. A Figura 2.10 exibe a visão estrutural de um documento NCL simples.

Outra visão é a de leiaute que representa as regiões onde as mídias do documento são apresentadas. É importante notar que esta visão apresenta as posições e dimensões inciais da mídia na tela da TV, visto que ao longo da execução do documento, modificações podem ser feitas nas propriedades das mídias. Na Figura 2.11, é mostrada a visão de leiaute onde a região do vídeo ocupa toda a tela da TV.

A visão temporal mostra, numa linha do tempo, a ordem em que as mídias do documento são apresentadas e sua duração. Nesta visão, a interatividade é difícil de ser representada, pois depende da interação do usuário com a aplicação hipermídia e não tem um instante predeterminado para acontecer.

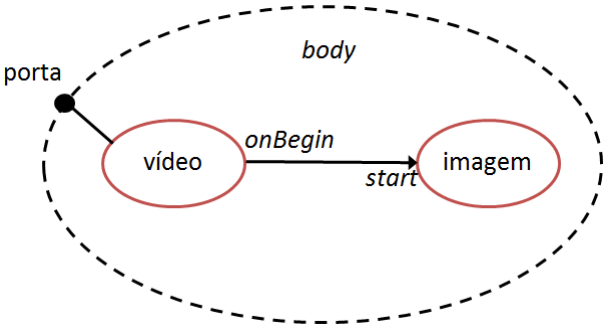


Figura 2.10: Visão estrutural do documento NCL exemplo

Figura 2.11: Visão de leiaute do documento NCL exemplo

Sendo assim, os instantes dos objetos de mídia que derivam do elo de interatividade não são determinados a priori. Desta forma, existem diferentes formas de representar a interatividade nesta visão. A Figura 2.12 exibe a visão temporal do documento NCL.

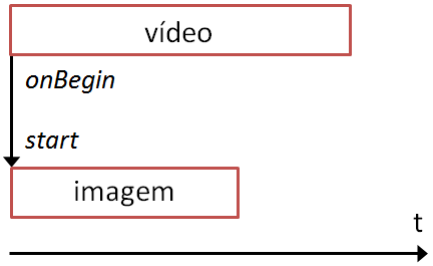


Figura 2.12: Visão temporal do documento NCL exemplo

Uma outra visão é a espacial. Ela apresenta como o documento é apresentado num determinado instante de tempo. Isto é, exibindo onde e como as mídias são apresentadas nos instantes especificados. A visão espacial do documento NCL é exibida na Figura 2.13.

Neste capítulo, foram estudados os principais fundamentos de sistemas hipermídia que são relevantes para este trabalho, definindo e descrevendo seus elementos, além de apresentar como os documentos hipermídia são expressados através de um modelo hipermídia. Também discutiu cada um dos paradigmas para a sincronização temporal entre os componentes de um documento hipermídia. Além disto, apresentou as entidades e a máquina de estado de eventos do modelo NCM no qual a linguagem NCL é baseada. E por fim, esta linguagem, que é utilizada para especificar aplicações para o sistema brasileiro de TV

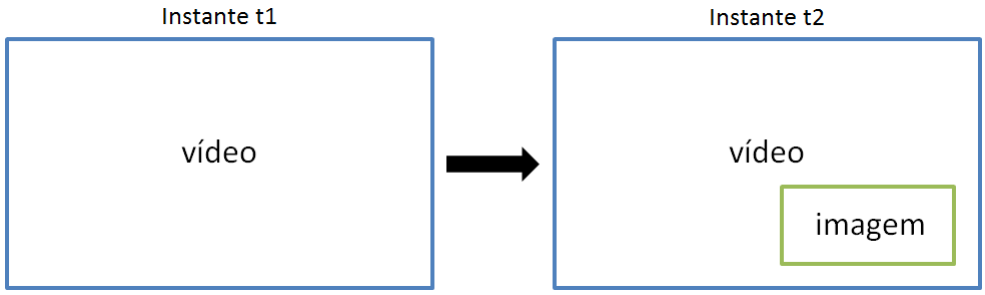


Figura 2.13: Visão espacial do documento NCL exemplo

digital, foi estudada apresentando a estrutura do documento NCL e suas diversas visões.

Capítulo 3

Trabalhos Relacionados

Para obter informações de visões temporais de documentos hipermídia já implementadas, inclusive no âmbito de TV digital, foram analisadas diversas ferramentas de autoria de documentos hipermídia, seus modelos de dados e suas interfaces gráficas. Também foram pesquisadas várias ferramentas comerciais bastante conhecidas para criação e edição de apresentações multimídia, como o *Final Cut Pro* [10], *iMovie* [6], *Adobe Prelude CS6* [8] e *Nero Vision* [7].

As ferramentas acadêmicas estudadas foram as seguintes: Composer [18], CMIFed [36], Firefly [12], GRiNs [13] e NEXT [30]. Dentre elas, o Composer e o NEXT são ferramentas de autoria para documentos NCL. O NEXT é o editor cuja funcionalidade é complementada pela ferramenta proposta neste trabalho.

3.1 Composer

A ferramenta Composer [18] é um ambiente de autoria integrado (IDE) para o desenvolvimento de aplicações NCL para o sistema brasileiro de televisão digital, facilitando e agilizando a criação de aplicações interativas. A ferramenta se adapta a diversos perfis de autores e oferece suporte a requisitos não funcionais. A arquitetura é baseada em um micronúcleo que permite a troca de informações entre os diferentes módulos que complementam o ambiente. Além do micronúcleo, um modelo central e extensões compõem a arquitetura. Suas extensões são construídas através de *plugins*, que são programas que interagem com o micronúcleo e acrescentam novas funcionalidades à ferramenta.

O Composer foi desenvolvido baseado no editor HyperProp [35], assim ele também oferece ao autor várias visões do documento NCL (estrutural, leiaute, textual e temporal). A principal alteração feita no HyperProp foi em sua visão temporal quanto à sincronização dos objetos no tempo e a aplicações com interatividade que geram diferentes cadeias temporais. Na visão temporal do HyperProp, só é possível visualizar as cadeias temporais, diferente do Composer, que permite criar e editar os objetos de mídias ao longo do tempo. As modificações feitas no cadeia temporal são refletidas nas outras visões do documento NCL no Composer.

3.1.1 Modelo de Dados

O modelo de dados utilizado para representar a cadeia temporal hipermídia [16] tanto do Composer quanto do HyperProp, consiste em um grafo que é formado por um conjunto de nós que podem ser nós de início, de término ou de conectores; e arestas unidirecionais. Este modelo leva em consideração a máquina de estado de eventos da linguagem NCL versão 3.0. Ou seja, as transições de início (*starts*) e de recomeço (*resumes*) são representadas pelos nós de início. Já as transições de término (*stops*) e de pausa (*pauses*) são representadas pelos nós de término. Ainda neste modelo, as arestas do grafo podem representar os estados ocorrendo (*occurring*) e pausado (*paused*) de eventos de apresentação (*presentation*). O primeiro estado é representado pelas arestas com origem em nós de início e destino em nós de término. E o estado pausado é representado pelas arestas com origem em nós de término e destino em nós de início. Além disso, as arestas podem definir relacionamentos de causalidade representando condições que devem ser satisfeitas para a execução da transição de destino (nó de destino da aresta). Isto é, a transição de origem deve ser executada e depois de um determinado intervalo de tempo especificado na aresta, a transição de destino é executada. Não só os relacionamentos causais mas também os de restrições podem ser representados. Porém estes relacionamentos de restrições não são considerados no contexto de TV digital.

Assim, para cada nó do grafo é possível obter seu instante de execução na cadeia temporal pelo cálculo da soma dos tempos das arestas visitadas desde do vértice inicial pré-definido até o nó desejado. Neste modelo, há pontos que definem outra cadeias temporais hipermídia. Tais momentos indicam a existência de interatividade, onde os tempos a partir destes pontos, para serem calculados, dependem do instante da interação do usuário com a aplicação. Estas cadeias são consideradas cadeias auxiliares e a cadeia formada pelo nó inicial pré-definido é a principal. Quando esses instantes são definidos, as cadeias auxiliares passam a integrar a cadeia principal. A Figura 3.1 apresenta um exemplo da representação de um trecho de documento NCL no modelo de grafo temporal utilizado.

As letras I e T representam os nós de início e de término respectivamente. E as letras W, W', W'' representam, nas arestas, a duração de exibição dos objetos de mídia *audio1*, *video1* e *ads1* respectivamente. E a letra S é o arco que indica um relacionamento de interatividade, sendo assim, seu intervalo de tempo é indeterminado. E a partir da aresta *link3* é gerada uma cadeia temporal auxiliar. Cada retângulo pontilhado na figura representa uma cadeia temporal.

Na Figura 3.2, é mostrada a representação de um vértice que representa um conector pelo nó C. Este serve como ponto de união em relacionamentos multiponto, no caso o elo *link3*. Ele dispara duas ações simultâneas, a aresta P que representa a pausa do objeto *audio1* e a aresta W'' que inicia a mídia *ads1*. E o elo *link4* dispara o recomeço do objeto *audio1* quando a mídia *ads1* termina sua execução. A as arestas de pausa não entram no cálculo do tempo dos nós e elas obtém seu valor através da diferença dos intervalos de tempo dos nós interligados pela aresta.

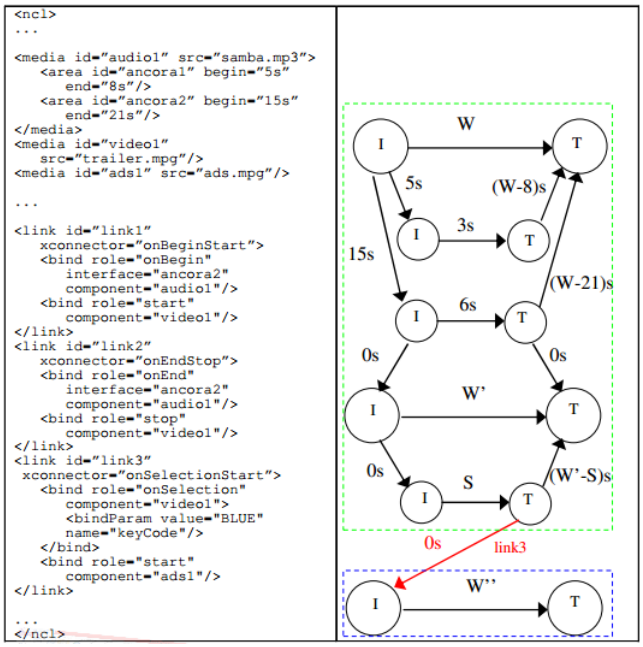


Figura 3.1: Documento NCL representado no modelo de grafo do Composer, retirado de [17]

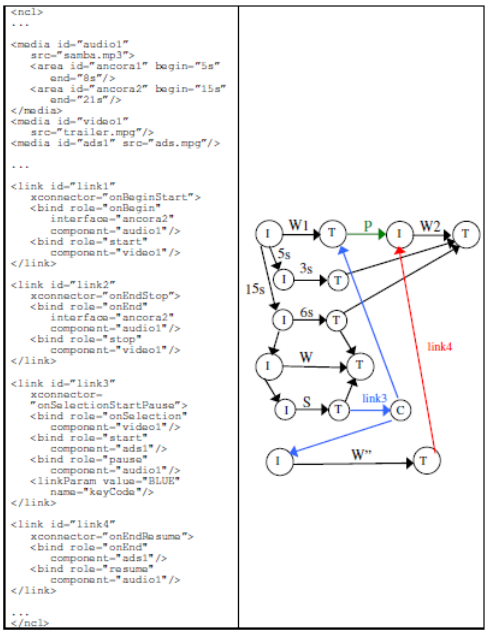


Figura 3.2: Nó de conectores no modelo de grafo do Composer, retirado de [17]

3.1.2 Interface Gráfica

Quanto a interface gráfica da visão temporal do Composer, ela possui uma escala de tempo que é alterada de acordo com o zoom utilizado, destacado pelo número 1 na Figura 3.3. Uma barra de status é apontada pelo número 2, que informa o nome da mídia selecionada na cadeia temporal e seu tempo de início e fim de apresentação. E uma barra de tempo indicada pelo número 3. Assinalado pelo número 4, está a representação gráfica de uma âncora. No número 5, tem-se a simbolização de um elo causal entre a âncora apontada pela numeração 4 e o objeto de mídia indicado por 6. Para indicar interatividade, um símbolo

é utilizado, indicado pelo número 7 na figura. Um instante de tempo indeterminado é representado por um ícone, destacado pelo número 8.

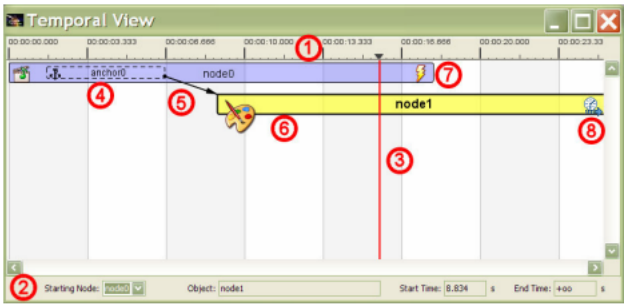


Figura 3.3: Visão temporal do Composer, retirado de [17]

A edição da cadeia temporal hipermídia é feita através de relações de alto nível, assim dividindo os relacionamentos do grafo em dois grupos: os elos de começo/fim e elos de pausa/recomeço. A especificação de relacionamentos temporais é realizada através de caixas de diálogo como a exibida na Figura 3.4.

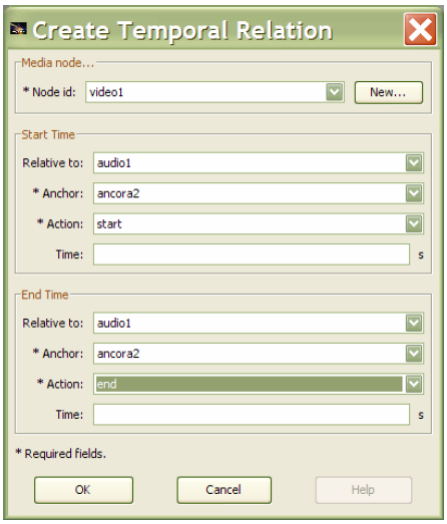


Figura 3.4: Definição de elos temporais no Composer, retirado de [17]

Os relacionamentos de interatividade também são definidos em caixas de diálogo como mostrado na Figura 3.5, especificando o momento que ela pode ocorrer, a tecla do controle remoto que deve ser pressionada pelo telespectador para interagir e a ação a ser executada após a interatividade. Esta ferramenta da visão temporal só permite a especificação de relações de interatividade um para um, o que limita a expressividade oferecida por NCL.

Também é possível fazer a simulação de eventos interativos e assim visualizar a cadeia temporal atribuindo um valor para os tempos indeterminados devido a interatividade. A Figura 3.6 demonstra uma cadeia temporal caso o telespectador interaja com a aplicação em um determinado instante durante a apresentação do objeto *node0*, o que finaliza a apresentação do *node1*.

Apesar desta visão temporal proposta pela versão inicial do *Composer* apresentar funcionalidades que facilitam a autoria de documentos NCL, ela não está disponível na nova versão da ferramenta, *Composer 3* [22].

Figura 3.5: Definição dos elos de interatividade do Composer, retirado de [17]

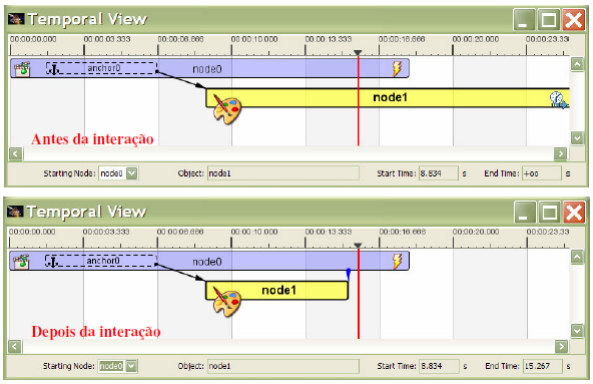


Figura 3.6: Simulação de eventos interativos do Composer, retirado de [17]

3.2 CMIFed

O editor CMIFed [36] é um ambiente de edição e apresentação de documentos hipermídia. Ele utiliza um modelo de dados, o *CMIF hypermedia model*, que se baseia em outro modelo hipermídia, o *Amsterdam hypermedia model* [20], que por sua vez estende o *Dexter hypertext reference model* [19]. Este modelo permite a criação de apresentações aninhadas, onde a estrutura de aninhamento representa o sincronismo temporal do documento. Um documento CMIF pode ser representado por uma estrutura em árvore, cujos nós folhas são apresentações atômicas, e os galhos possuem semântica temporal sequencial ou paralela.

3.2.1 Modelo de Dados

No modelo conceitual do CMIF, uma apresentação atômica é uma coleção de eventos. Estes representam fragmentos de áudio, vídeo, imagem ou texto. Marcadores são âncoras que servem para relacionar, temporalmente, componentes do documento hipermídia. Propriedades específicas de apresentação são armazenadas com os eventos, como os marcadores também são. Existem três níveis de hierarquia: os descritores de eventos que especificam como eles ocorrem na apresentação, os descritores de dados que detalham propriedades estáticas dos dados e o dado propriamente dito.

Os eventos de uma apresentação atômica são ordenados através de relacionamentos temporais.

Nesse modelo são usados dois mecanismos para expressar relações de sincronismo temporal: composição paralela e sequencial, e arcos de sincronização (sinc-arcs). A hierarquia da árvore do documento CMIF já define os relacionamentos temporais através da composição paralela e sequencial, ou seja, o paradigma de sincronização temporal é hierárquico no CMIF. Quanto aos arcos de sincronização, eles fazem o relacionamento entre os marcadores de eventos de uma mesma apresentação atômica especificando um retardo. A interatividade nas apresentações é representada através dos elos de hipermídia (*hyperlinks*) que, como os sinc-arcs, ligam marcadores de eventos. Além disto, o modelo CMIF possui o conceito de *canal*, que agrupa os eventos de uma apresentação pelo tipo de mídia. Uma lista de atributos é definida para cada elemento do modelo detalhando as propriedades do mesmo.

3.2.2 Interface Gráfica

CMIFed possui duas visões da apresentação para a edição do documento. Uma delas é a visão de hierarquia onde os eventos são aninhados formando subapresentações. Na Figura 3.7, pode ser notado que a apresentação *time-mgmt* possui 3 apresentações compostas (*intros*, *video scene* e *summary*). A ordem como elas são dispostas ne janela (de cima pra baixo) definem a relação temporal por composição sequencial, ou seja, a primeira suapresentação a ser exibida é a *intros*, em seguida *video scene* e *summary*. A primeira apresentação contém as apresentações atômicas (*title*, *still*, *intro* e *music*) dispostas uma ao lado da outra. Isto representa uma composição paralela indicando que tais apresentações são exibidas ao mesmo tempo.

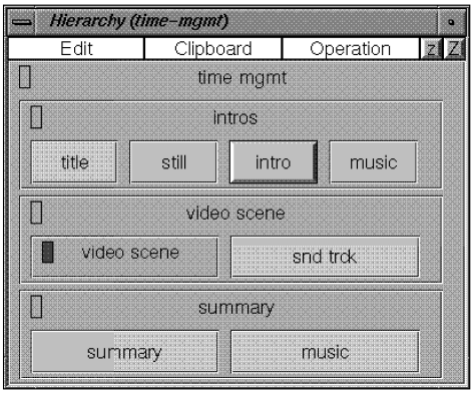


Figura 3.7: Visão hierárquica do CMIFed, retirado de [36]

A outra visão é a de canal onde os eventos são agrupados pelo tipo de mídia. A composição sequencial é representada pelo alinhamento na horizontal dos componentes e os mesmos dispostos na vertical dentro de um canal indicam a composição paralela. Esta visão é mostrada na Figura 3.8 onde, também, nota-se um arco de sincronização entre os componente *still* do canal *image* e *title* do *title text*.

O editor ainda possui um *player* onde é possível visualizar a apresentação desenvolvida. Ele possui uma janela de controle da apresentação e o leiaute das janelas pode ser alterado (geometria das janelas dos canais e o posicionamento das âncoras nas imagens). Esta visão é demonstrada na Figura 3.9.

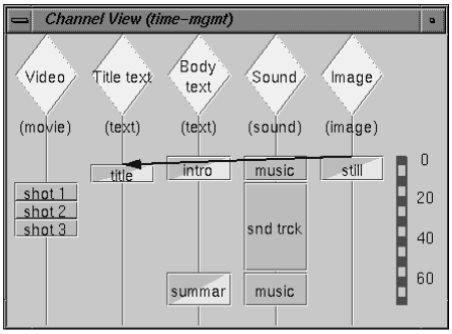


Figura 3.8: Visão de canal do CMIFed, retirado de [36]

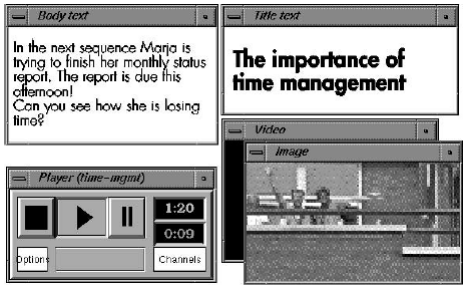


Figura 3.9: Player do CMIFed, retirado de [36]

3.3 FireFly

FireFly [12] é um sistema que facilita a criação, edição e apresentação de documentos hipermídia. Ou seja, é uma ferramenta de autoria para o desenvolvimento de aplicações hipermídia. Ela utiliza um modelo de dados que divide um documento em três partes: itens de mídias, relacionamentos temporais e listas de operações. O sistema ainda possui um formatador que processa os relacionamentos temporais para controlar a apresentação do documento.

3.3.1 Modelo de Dados

Os itens de mídias são fragmentos de objetos de mídia do documento que podem ser de vários tipos (vídeo, áudio, imagem, texto etc.). Tais itens especificam seu comportamento temporal na apresentação. Para isso, estes itens de mídia possuem eventos que indicam pontos, os quais podem ser relacionados com outros. Os eventos podem ser síncronos, o instante de tempo na apresentação do item pode ser previsto, ou assíncronos, quando o posicionamento no tempo é indeterminado. Cada item, também, tem associado um procedimento de controle que permite alterar o modo como as mídias se apresentam. As listas de operações realizam o mesmo que os procedimentos de controle, porém elas não são armazenadas juntamente com os itens de mídia. Desta forma, o modelo permite que as configurações de apresentação de uma lista de operações possam ser reutilizadas por outro item de mídia.

A sincronização temporal é realizada através de dois tipos de relacionamento temporal: de igualdade e desigualdade temporais. O primeiro indica que dois eventos são apresentados simultaneamente no tempo ou sequencialmente com intervalo de tempo fixo especificado entre eles. O relacionamento de desigualdade liga dois eventos que ocorrem sequencialmente, porém com intervalo de tempo, entre eles,

indeterminado.

3.3.2 Interface Gráfica

O FireFly apresenta duas ferramentas para visualização, criação e edição dos relacionamentos temporais. Uma delas é o visualizador de itens de mídia mostrado na Figura 3.10.

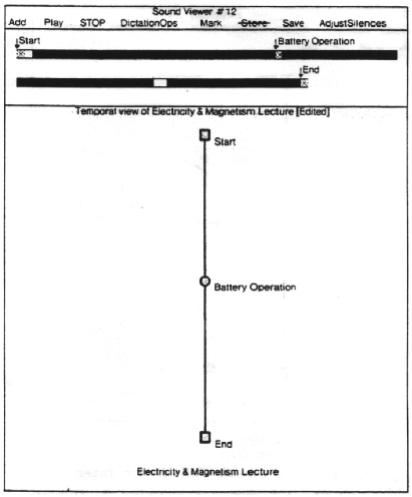


Figura 3.10: Visualizador de itens de mídia do FireFly, retirado de [12]

Na Figura 3.10, é representada, através de um grafo, a mídia de áudio, *Electricity & Magnetism Lecture* que possui dois retângulos, *Start* e *End* especificando os eventos de início e fim respectivamente. Além disso, um círculo indica um evento interno. As arestas que ligam tais eventos possuem um tamanho que é proporcional ao intervalo entres esses eventos.

A outra ferramenta, exibida na Figura 3.11, permite editar o documento hipermídia através de uma barra de menu. Note que o evento interno *Battery Operation* é usado para sincronizar com o evento de início do item de mídia *Battery Animation*.

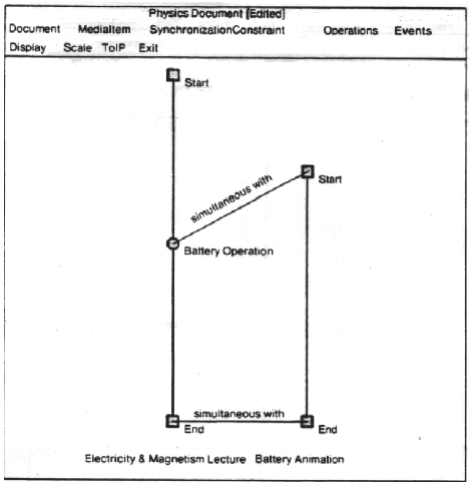


Figura 3.11: Editor interativo de documentos do FireFly, retirado de [12]

3.4 GRiNs

GRiNs [13] é uma ferramenta de autoria de documentos hipermídia, baseados na linguagem SMIL 2.0 [3], com ênfase no desenvolvimento das aplicações através da visão temporal utilizando recursos gráficos. O documento criado pode ser visualizado na própria ferramenta. A visão temporal cria os documentos de acordo com os elementos de sincronização temporal especificados na linguagem de autoria.

3.4.1 Modelo de Dados

Os elementos da linguagem SMIL que permitem definir a ordem temporal das mídias na apresentação são: sequencial, paralelo e exclusivo. Cada um pode conter vários objetos de mídias indicando, no caso do elemento sequencial, que todos são apresentados em sequência na execução do documento. É possível fazer o aninhamento destes elementos, ou seja, um contêiner sequencial pode possuir um paralelo e vice-versa, assim criando a sincronização entre as mídias conforme desejado.

3.4.2 Interface Gráfica

A visão hierárquica do GRiNs, como exibida na Figura 3.12, agrupa os objetos de mídia de acordo com sua semântica temporal. Isto é, Os objetos que se apresentam em sequência são colocados em retângulos que indicam uma ordem sequencial e analogamente para os objetos que são exibidos em paralelo ou que devem ser escolhidos (contêiner *switch*).

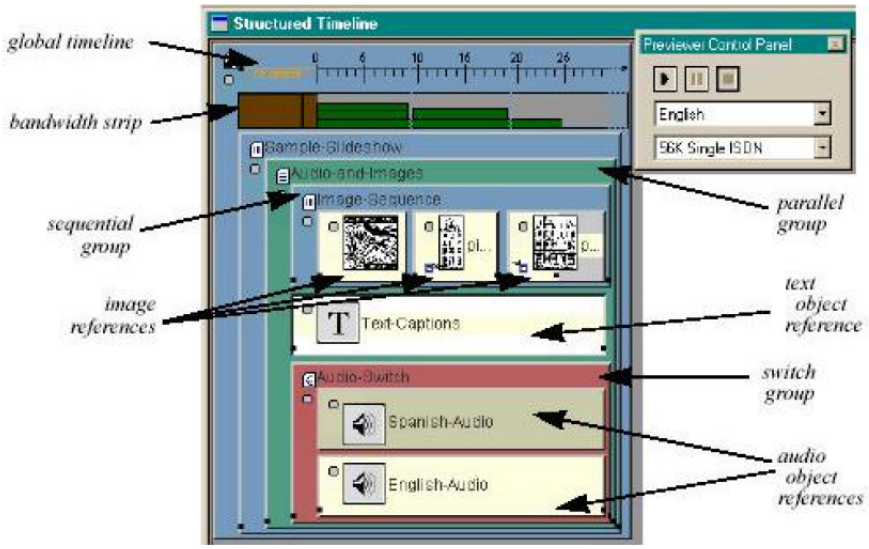


Figura 3.12: Visão temporal do GRiNs , retirada de [14]

Na Figura 3.12, o documento *Sample-Slideshow* possui o contêiner *Audio-and-Images* que representa um grupo paralelo, o *Image-Sequence* que indica um conjunto de mídias, as quais são apresentadas em sequência e o *Audio-Switch* que simboliza uma escolha, no caso, a opção entre o áudio espanhol ou inglês.

3.5 NEXT

NEXT [31] [23] é uma ferramenta de autoria gráfica de documentos NCL com suporte a templates de composição, estruturas genéricas responsáveis por especificar os relacionamentos entre os componentes do documento. Tais templates são especificados na linguagem XTemplate 3.0 [28]. Assim, a ferramenta facilita e agiliza o desenvolvimento de aplicações por autores com pouco ou nenhum conhecimento em NCL para o sistema brasileiro de televisão digital interativa. NEXT oferece diferentes visões do documento que são disponibilizadas através de *plugins*, que são adicionados ao NEXT por meio de uma arquitetura extensível e adaptável a diferentes perfis de autores.

3.5.1 Arquitetura

A arquitetura do NEXT é baseado em um núcleo apto a tratar documentos NCL e a comunicação com os *plugins*. Sendo assim, ela permite que novas funcionalidades sejam adicionadas ao editor facilmente. Na Figura 3.13, é possível notar que o núcleo é completamente independente dos *plugins*.

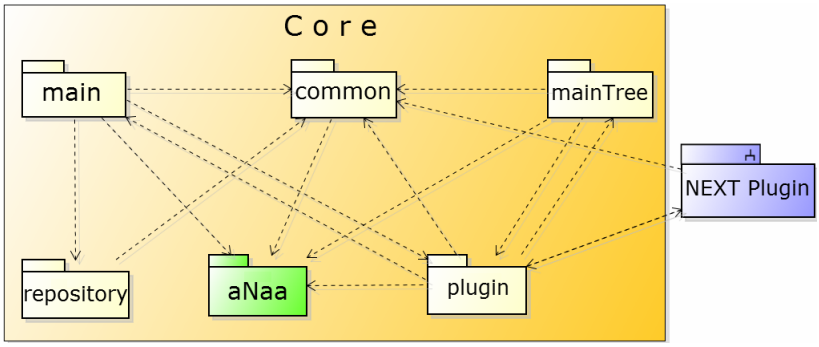


Figura 3.13: Arquitetura do NEXT

O pacote *aNaa* [27] é responsável pela importação, criação e edição de documentos NCL utilizando a API aNa [29] . Além disso, ela é capaz de analisar a consistência estrutural e comportamental do documento. Os pacotes *main* e *common* são responsáveis pela interface gráfica e pelo idioma da ferramenta respectivamente. O pacote *plugin* encarrega-se da troca de informações entre os *plugins* e o núcleo do NEXT, além de guardar informações sobre os mesmos e informá-los também sobre alterações no documento NCL. E o pacote *repository* armazena as mídias que são importadas ao NEXT para ser utilizadas na construção da aplicação.

3.5.2 Desenvolvimento de Plugins

Para adicionar uma nova funcionalidade ao NEXT através de criação de um *plugin*, deve-se desenvolver um programa em java com a extensão *jar*. Além disso, para prover a comunicação do *plugin* com o NEXT, deve-se desenvolver um pacote chamado de *myPlugin* contendo as classes *StartPlugin* e *PluginInfo*. A primeira deve estender uma classe do pacote *javax.swing*, *JInternalFrame*, para inicializar o *plugin*, o qual é chamado pelo núcleo do NEXT. Ela deve implementar os métodos *getUpdateAtNCLDocument* para notificar os *plugins* sobre modificações no documento NCL. A classe *PluginInfo* armazena informações

dos *plugins* que deve conter dois métodos: *getAlias* que corresponde ao nome do *plugin* no NEXT e o *getNCLInterest* que retorna uma lista de elementos de interesse do *plugin*, que quando forem modificados, o *plugin* é notificado. A seguir são mostradas estas duas classes e seus métodos que são necessários para incluir uma nova funcionalidade ao NEXT através de *plugins*.

```
public class StartPlugin extends JInternalFrame{
    public StartPlugin(Language language, HandleNCLDocument ne){ ... }
    public void getUpdateAtNCLDocument (ModType type,
                NCLElement elem, Object[] values) { ... }
}

public class PluginInfo{
    public static String getAlias(){
        return ''Nome do plugin'';
    }
    public static List<Element> getNCLInterest(){
        List<Element> elements = new ArrayList();
        elements.add(Element.CONNECTOR);
        elements.add(Element.CONNECTOR_BASE);
        ...
        return elements;
    }
}
```

3.5.3 Plugins

O NEXT possui quatro *plugins* sendo que um deles é o próprio editor gráfico de criação de documentos NCL com uso de templates de composição. O segundo permite visualizar estruturalmente, criar e editar o documento. Outro proporciona a criação de regiões e descritores, elementos NCL responsáveis pelo leiaute e apresentação da mídia respectivamente. O último permite criar conectores hipermídia graficamente que são utilizados por elos, relacionando os componentes do documento. Além disso, o NEXT permite a visualização textual e em árvore de documentos NCL como pode ser notado na Figura 3.14. Note que essas funcionalidades estão implementadas dentro do núcleo do NEXT, e não como *plugins*.

Wizard para uso de templates

O editor para criação de documentos NCL com uso de templates, um *wizard* para uso de templates, facilita a autoria de aplicações em NCL para TV digital brasileira. Qualquer template especificado em XTemplate 3.0 pode ser utilizado com o *wizard*. Logo, esta ferramenta permite a construção de aplicações por autores que não possuem conhecimento algum sobre NCL. Além disto, ela agiliza a criação de aplicações que possuem um grande número de linhas de código, muitas vezes repetitivas, através do

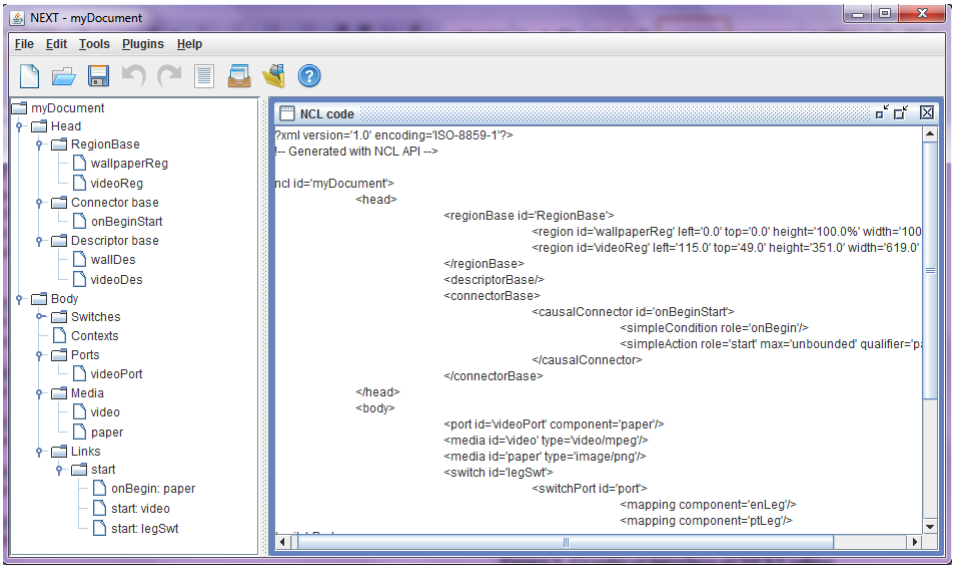


Figura 3.14: Visão textual do NEXT

reúso de estruturas genéricas do comportamento da aplicação.

O *plugin* utiliza um template como entrada e através da especificação do mesmo suas telas são geradas, indicando graficamente como os nós de mídia da aplicação serão apresentados simultaneamente na TV. O NEXT possui uma base de templates que podem ser escolhidos pelo autor para construir sua aplicação. Após a escolha do template, ele só necessita preencher os componentes das telas do template utilizando o repositórios de mídias. Os relacionamentos entre os componentes do documento já são especificados pelo template. A Figura 3.15 mostra como o repositório de mídias é usado com o editor para preencher o template através do recurso *drag-and-drop*.

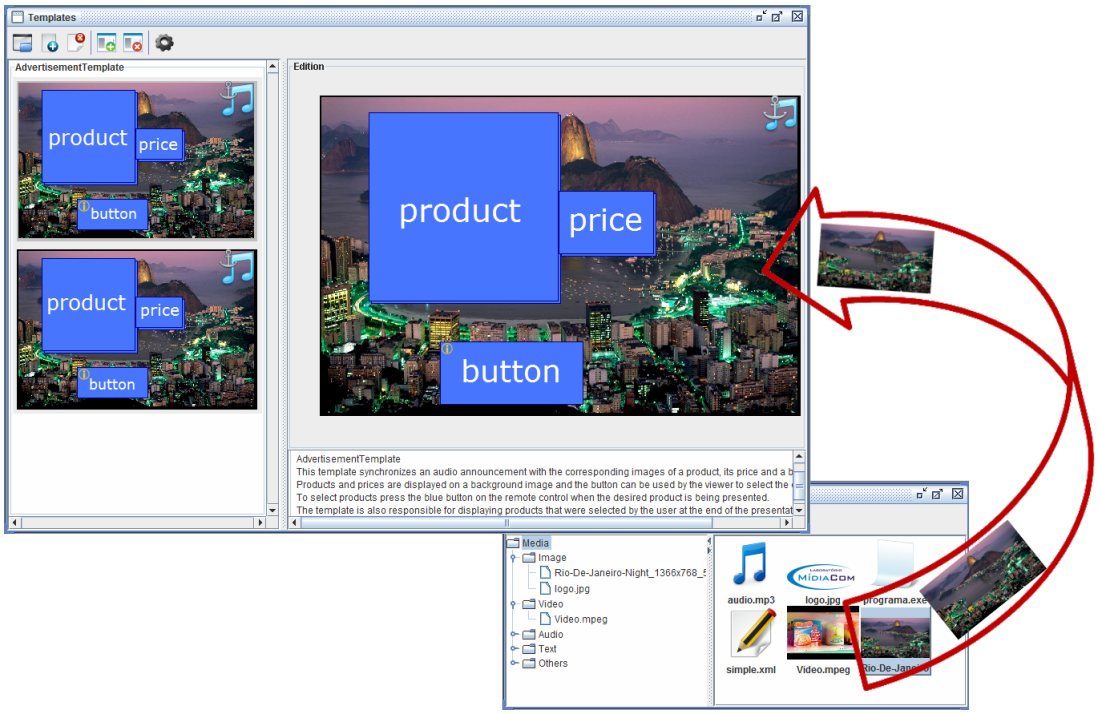


Figura 3.15: *Drag-and-drop* no NEXT

Visão Estrutural

O *plugin* de visão estrtural permite a edição gráfica do documento NCL. Ainda que facilite o desenvolvimento de aplicações para TV digital, este *plugin* exige que o autor tenha pelo menos um conhecimento básico da linguagem NCL. A interface gráfica pode ser vista na Figura 3.16.

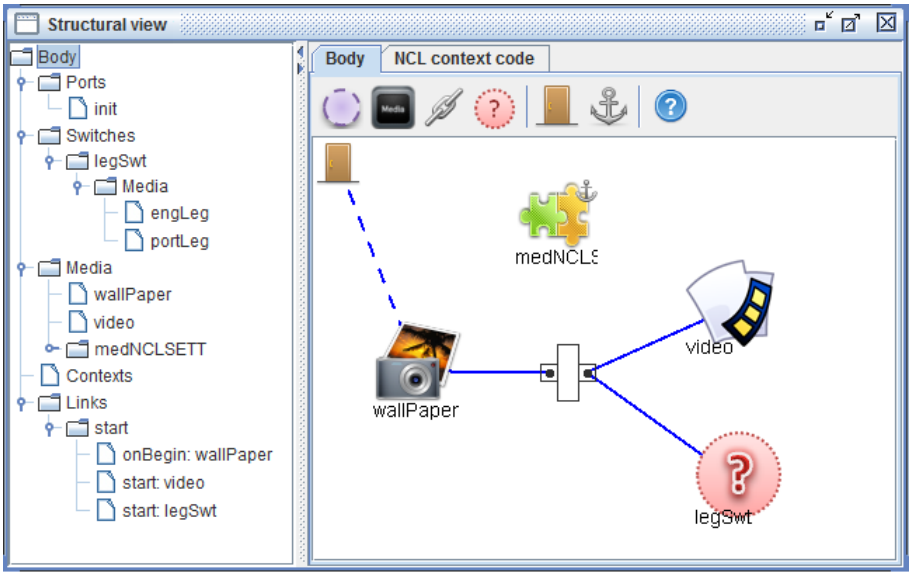


Figura 3.16: Visão Estrutural do NEXT

Visão de Leiaute

O *plugin* de visão de leiaute possibilita a criação e edição das regiões e descritores do documento NCL. Em sua interface, exibida na Figura 3.17, as regiões são construídas graficamente e redimensionadas apenas usando o mouse. Quanto aos descritores, estes devem ser editados através de caixas de diálogo que exijem do autor conhecimento básico em NCL. Esta ferramenta pode ser usada independentemente do NEXT, criando bases de regiões e descritores que poderão ser usadas por qualquer documento NCL.

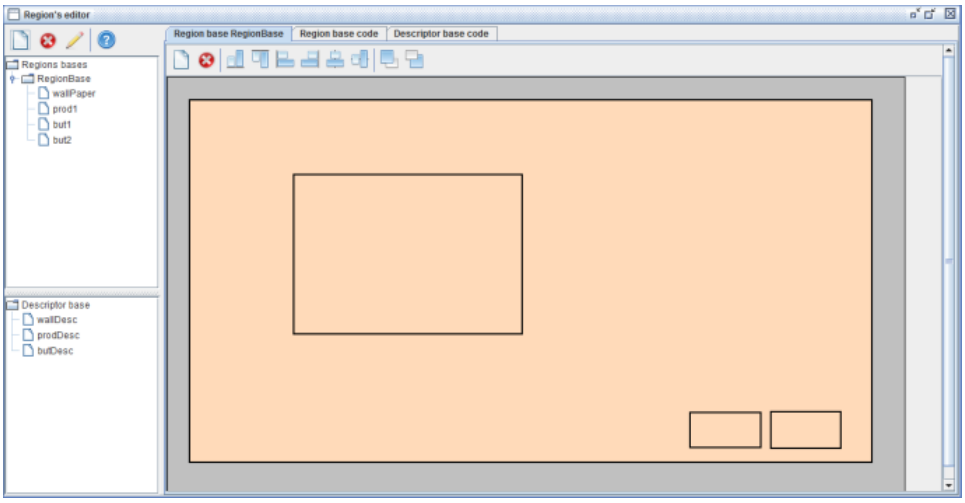


Figura 3.17: Visão de Leiaute do NEXT

Editor de Conectores

Este *plugin* permite criar e editar conectores hipermídia graficamente para documentos NCL. Como o *plugin* de visão de leiaute, ele permite ser usado separadamente do NEXT para criar uma base de conectores que pode ser utilizada por qualquer documento NCL. Sua interface gráfica é exibida na Figura 3.18.

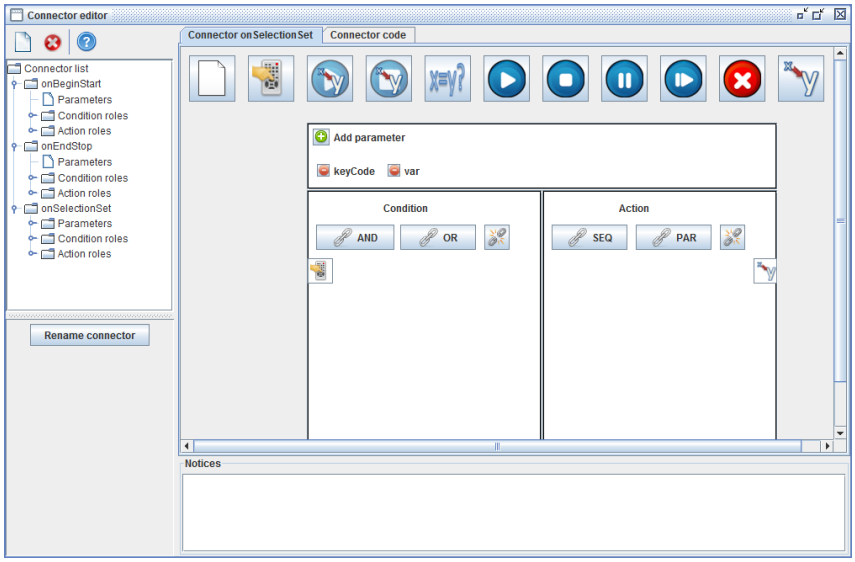


Figura 3.18: Editor de conectores do NEXT

Um dos objetivos do presente trabalho é desenvolver um novo *plugin* para o NEXT oferecendo uma visão temporal do documento NCL que permita a visualização e edição do documento.

3.6 Ferramentas Comerciais

O desenvolvimento de apresentações multimídia em softwares comerciais é comumente realizado com o paradigma de linha do tempo (*timeline*). Ou seja, os objetos de mídia são posicionados diretamente no eixo temporal e, intuitivamente, o autor cria seu documento multimídia colocando os componentes da apresentação nos instantes em que eles devem ser exibidos. Como exemplos destas ferramentas, têm-se o *Final Cut Pro* [10] e *iMovie* [6] da *Apple*, que fazem edição profissional de vídeos e editam filmes caseiros respectivamente. As interfaces dessas ferramentas são mostradas na Figura 3.19.

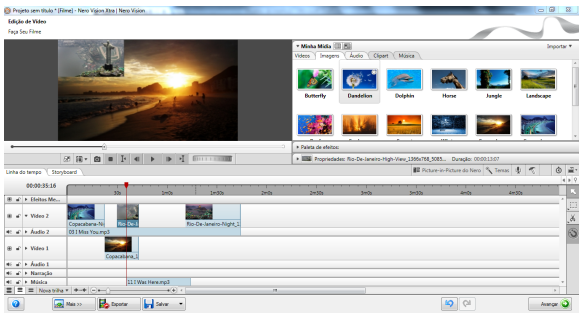
Além destas, há o editor *Adobe Prelude CS6* [8] da *Adobe* que simplifica a criação e edição de apresentações multimídia. Sua interface gráfica é exibida na Figura 3.20a. Outra é o *neroVision* [7] da *Nero AG* que separa, explicitamente, as mídias da linha do tempo em canais como mostrado na Figura 3.20b.

Este capítulo discutiu diversos trabalhos que apresentam ferramentas que auxiliam na autoria de documentos hipermídia, enfatizando como a visão temporal destes documentos é tratada por estes trabalhos. Dentre elas, a ferramenta *CMIFed* que apresenta uma visão temporal estruturada hierarquicamente através de composições paralelas e sequenciais como a ferramenta *GRiNs*. E o sistema *FireFly* que utiliza grafos para representar a visão temporal. Além destas ferramentas, foram analisados alguns softwares

(a) Final Cut Pro (b) iMovie

Figura 3.19: Ferramentas de edição de conteúdo multimídia, Apple

(a) Adobe Prelude CS6, Adobe



(b) Nero Vision, Nero AG

Figura 3.20: Ferramentas comerciais para edição de apresentações multimídia

comerciais que permitem a criação de apresentações multimídia e utilizam o paradigma de sincronização temporal baseado em *timeline*. Porém, de todas estas ferramentas estudadas, apenas o *Composer* e o NEXT auxiliam na autoria de documentos NCL para TV digital. A primeira ferramenta apresenta diversas visões de um documento NCL, inclusive a visão temporal que utiliza um modelo de grafo para representar a cadeia temporal, e trata a interatividade em documentos hipermídia. Entretanto, na nova versão do *Composer* a visão temporal não está disponível. Além desta ferramenta para documentos NCL, foi apresentado o NEXT que é um editor gráfico com suporte a templates de composição. Este editor também possui diversas visões do documento NCL, porém não tem a visão temporal que será implementada por este trabalho.

Capítulo 4

O Editor Gráfico da Visão Temporal

Neste capítulo é apresentada a ferramenta proposta neste trabalho, mostrando sua estratégia para exibir e editar a cadeia temporal de documentos NCL. Além disto, discute as estruturas de dados desenvolvidas na implementação (o foco deste trabalho), a qual concentrou-se na construção do modelo de dados e na geração da cadeia temporal, definindo-as e apresentando os fundamentos para a geração de cada uma delas. Tais estruturas são: o HTG (*Hypermedia Temporal Graph*), o plano de apresentação e a cadeia temporal. E ainda discute a forma como a ferramenta trata a interatividade nas aplicações NCL para representá-la na visão temporal. Em seguida, é feito um estudo de caso, tendo uma aplicação NCL como entrada para a ferramenta e exibindo a saída da mesma, isto é, o HTG, o plano de apresentação e a cadeia temporal gerados para o documento NCL de entrada. Por fim, é apresentado o protótipo da interface gráfica do usuário para a ferramenta.

4.1 Exibição da Cadeia Temporal

A cadeia temporal de um documento hipermídia exibe a sequência de apresentação dos objetos de mídia do documento na linha do tempo. Diversas ferramentas de criação e edição de documentos hipermídia fornecem a cadeia temporal da apresentação multimídia a fim de facilitar o entendimento do comportamento da mesma. Ou ainda, permitem criar um documento utilizando a cadeia temporal. Algumas destas ferramentas foram apresentadas na seção de trabalhos relacionados. Com base nestas ferramentas analisadas, foi definida a representação gráfica da cadeia temporal de documentos NCL da ferramenta proposta que pode ser vista na Figura 4.1.

Na cadeia temporal, as mídias são dividas em dois canais, um canal de vídeo e outro de áudio. O primeiro contém as mídias do tipo vídeo, imagem e texto que são exibidas no dispositivo de vídeo de saída. O segundo canal, apresenta as mídias do tipo áudio. Caso seja necessário mais de um canal de vídeo ou áudio como nas aplicações NCL de múltiplos dispositivos, basta criar uma nova trilha na cadeia temporal para criar a ordem de apresentação das mídias do outro dispositivo de vídeo ou áudio de saída. Na Figura 4.1, as mídias “clipe”, “fundo”, “logo” e “legenda” estão no “canal de vídeo1”, visto que elas são do tipo vídeo, imagem e texto. No “canal de vídeo 2”, é apresentada a mídia “trailer” do tipo vídeo.

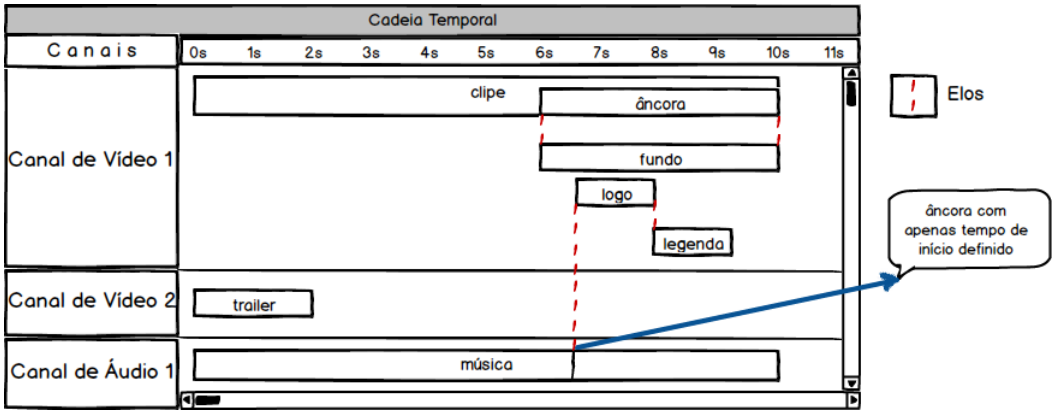


Figura 4.1: Representação gráfica da cadeia temporal

E a mídia "música" no "canal de áudio 1". Além destes canais, existe um outro canal de vídeo, ("canal de vídeo 2"), que exibe uma outra sequência de apresentação de mídias, que contém somente a mídia "trailer", para um outro dispositivo de vídeo de saída. Ainda nesta figura, as mídias "clipe", "trailer" e "música" correspondem ao início da apresentação que é dado pelo elemento *port* (como descrito na Seção 2.3) associado a cada uma destas mídias. A mídia "clipe" possui uma âncora que também está representada indicando o início e fim da apresentação da mídia "fundo" (sincronização dada por um elo do tipo *onBeginStart* e *onEndStop*). E a mídia "música" que possui uma âncora com apenas o instante de tempo de início, a qual está associada ao começo de apresentação da mídia "logo" (sincronização especificada por um elo do tipo "onBeginStart") cujo término faz com que a mídia "legenda" inicie sua apresentação, esta relação é obtida através de um *link* cuja semântica é definida por um conector do tipo *onEndStart*.

4.1.1 Representação da Interatividade

A representação da visão temporal de documentos multimídia se torna uma tarefa não trivial quando estes possuem interatividade. Visto que, os relacionamentos interativos, para disparar sua ação, dependem da interação do usuário. Ou seja, o instante de tempo em que ocorre a interação não pode ser determinado a priori, somente na execução do documento hipermídia. Esta dificuldade de representação da interatividade na cadeia temporal é tratada neste trabalho da seguinte maneira. Uma mídia que permite ser selecionada, isto é, o usuário é capaz de interagir com a mesma, é exibida na cadeia temporal com um ícone indicando a possibilidade de interatividade como pode ser observado na Figura 4.2.

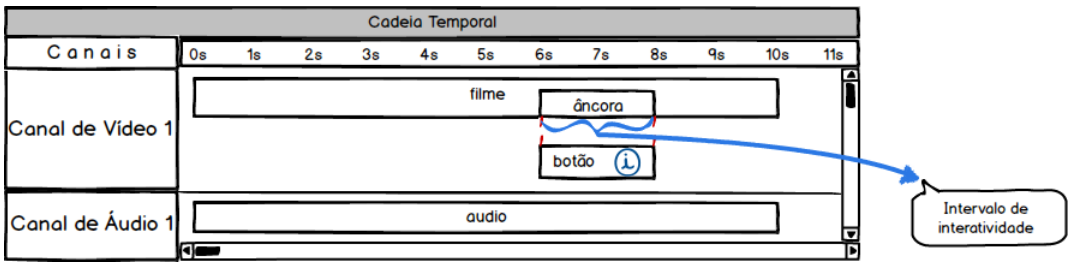
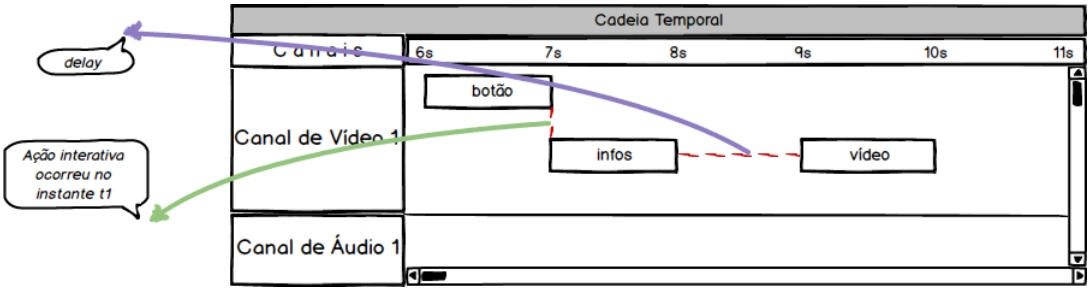
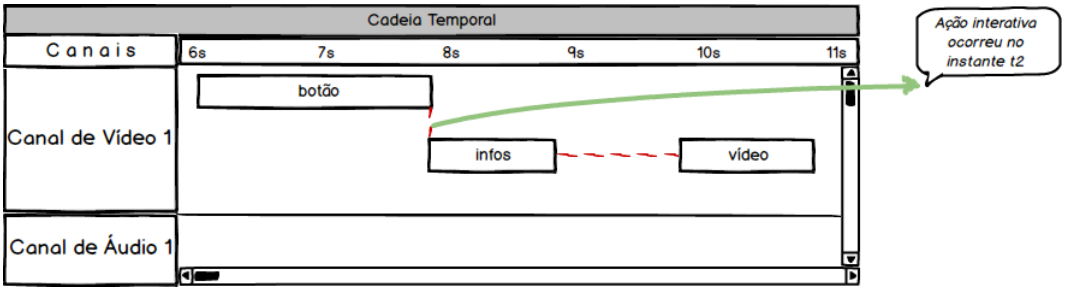


Figura 4.2: Indicação de interatividade na cadeia temporal

A mídia *botão*, no canal de vídeo, possui o ícone de interatividade "i", sendo assim, caso o usuário interaja com este objeto dentro do intervalo de interatividade, uma nova cadeia temporal, apresentando a sequência de apresentação das mídias, a partir do momento da interação, é exibida na visão temporal. Nesta sequência resultante da interação, são apresentadas novas mídias que são iniciadas devido a interação e outras são retiradas, pois elas têm sua apresentação terminada com a ocorrência da interatividade. Esta nova na cadeia para a visão temporal da Figura 4.2 devido a ação interativa é mostrada na Figura 4.3.



(a) Interatividade no instante t1 = 7s



(b) Interatividade no instante t2 = 7.8s

Figura 4.3: Representação de interatividade na cadeia temporal

Nota-se que a seleção do objeto *botão* resulta no início da mídia *infos* e no término da própria mídia *botão*. E com o fim da mídia *infos*, passados alguns instantes (*delay*), um vídeo é exibido. Como o instante de tempo, no qual a interação ocorre, é indeterminado, a cadeia temporal do editor permite que o usuário arraste a mídia interativa (*botão*) dentro do intervalo de tempo de interatividade, o qual é especificado pela âncora da mídia *filme*. Facilitando, assim, a visualização temporal do documento em diferentes instante de tempo que a interação pode ocorrer. Na Figura 4.3a e 4.3b, o objeto interativo é selecionado no instante "t1" e "t2" respectivamente.

No caso em que a ação interativa ocorre no instante "t2", é notado que o fim da apresentação do documento hipermídia se dá somente com o término da mídia "vídeo" como pode ser visto na Figura 4.4 onde as duas cadeias da visão temporal (cadeias temporais mostradas na Figura 4.2 e 4.3b) são unidas, diferentemente do caso em que a interação do usuário ocorre no instante "t1" onde as mídias "filme", "vídeo" e "audio" finalizam a apresentação.

Visto que a visão temporal de um documento NCL pode ser composta por várias cadeias temporais de acordo com a quantidade de relacionamentos que dependem da interação do usuário, a representação gráfica da visão temporal, além de exibir as diversas cadeias temporais, também mostra os

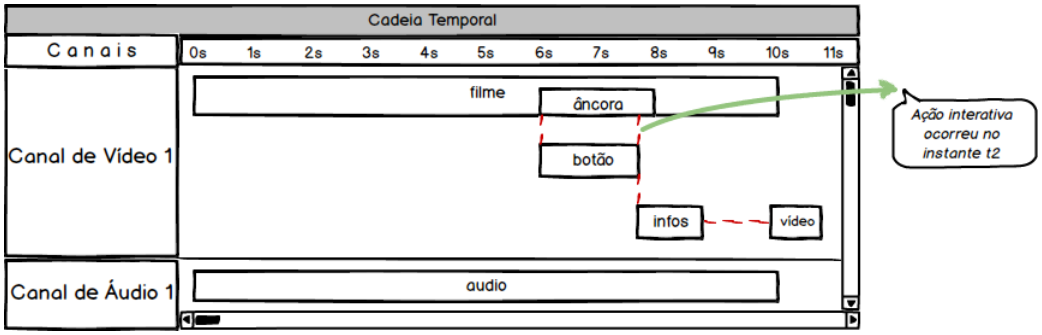


Figura 4.4: União das cadeias temporais

relacionamentos entre elas. A Figura 4.5 representa o relacionamento entre as duas cadeias temporais que foram unidas na Figura 4.4.

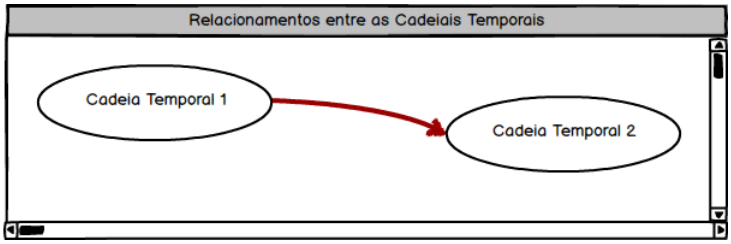


Figura 4.5: Relacionamento entre cadeias temporais

4.2 Edição da Visão Temporal

Como visto em alguns trabalhos relacionados, a visão temporal de documentos hipermídia pode ser editada graficamente. No caso de aplicações NCL, a modificação da ordem de apresentação das mídias por usuários que não são especialistas na linguagem pode se tornar complexa quando o documento possui muitos componentes e relacionamentos. Permitir que a sequência de apresentação das mídias de aplicações NCL seja alterada graficamente, possibilita estes usuários modificarem facilmente a ordem temporal do documento, sem se preocupar com as interfaces e elos necessários para realizar a sincronização temporal.

Assim, a edição gráfica da visão temporal, inicialmente, definida para a ferramenta proposta neste trabalho será realizada arrastando as mídias na cadeia temporal para os instantes de tempo em que se deseja posicioná-las. Também é possível arrastá-las entre os canais, conforme o tipo da mídia. Além disto, nesta estratégia de edição é possível criar novas mídias e posicioná-las na linha do tempo da aplicação. Como a alteração da ordem de uma mídia é realizada sem atentar aos elos e interfaces modificados e criados no documento NCL, é necessário que se mantenha a consistência temporal da aplicação. Um protótipo deste esquema da edição da cadeia temporal é ilustrado na Figura 4.6, mostrando a alteração do instante de apresentação da mídia *logo* arrastando-a para o início da cadeia temporal apresentada na Figura 4.1. Como a mídia "legenda" está associada por elo com o "logo", ela também é arrastada, desta forma mantendo a sincronização temporal entre estas duas mídias.

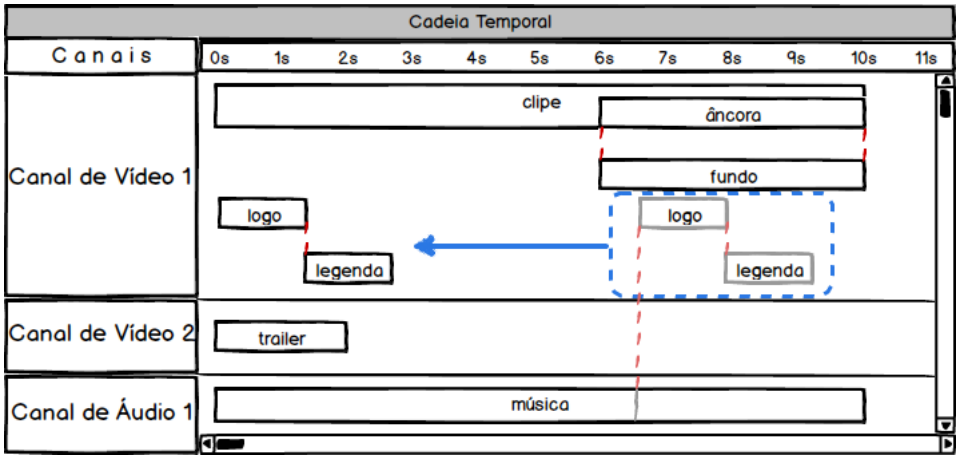


Figura 4.6: Edição da Cadeia Temporal

4.3 Modelo de Dados

Nesta seção, são apresentadas as estruturas de dados utilizadas para gerar a cadeia temporal de um documento NCL, descrevendo-as e detalhando seus processos de construção. A primeira delas é o *Hypermedia Temporal Graph* [15], um modelo de grafo para cadeias temporais hipermídia. A segunda é o plano de apresentação que coleta informações a partir da leitura do HTG necessárias para construir a visão temporal do documento NCL. E a última corresponde a própria cadeia temporal que é derivada do plano de apresentação. É importante enfatizar que as duas últimas estruturas são deduzidas do *HTG*.

4.3.1 Hypermedia Temporal Graph

O *Hypermedia Temporal Graph*, recomendado [33] para garantir a qualidade da apresentação de aplicações NCL, é um grafo dirigido (dígrafo) que armazena informações necessárias para o controle do sincronismo temporal entre os componentes de um documento NCL. Este dígrafo tem como base a máquina de estado de eventos apresentada na Seção 2.2.2. Ele é definido por um conjunto de vértices e arestas com condições.

Cada vértice é representado pela tripla: nome da ação que dispara a transição na máquina de estado, o identificador da âncora especificando o objeto ou um segmento de seu conteúdo, no caso de eventos de apresentação\seleção ou o identificador da propriedade de uma mídia e seu valor, para eventos de atribuição e o tipo de evento no qual a ação ocorre. Ou seja, o vértice corresponde a transição disparada na máquina de estado para uma determinada âncora sobre um evento.

As arestas do grafo representam os relacionamentos entre as transições, indicando que a transição no vértice de destino somente é disparada se a partir da origem a condição for satisfeita. Estas condições podem ser simples ou compostas. Uma condição simples pode corresponder a um intervalo de tempo que deve ser respeitado para que a transição destino da aresta seja disparada, a uma variável que é testada com um determinado valor que pode ser um intervalo de tempo inclusive e a uma interatividade que deve ocorrer para disparar o vértice destino (transição). Já as condições compostas são duas ou mais condições simples relacionadas com os operadores lógicos *OR*, *AND* e *NOT*.

Para exemplificar a estrutura do HTG, considere uma aplicação hipermídia que de início exibe

um vídeo cuja duração corresponde ao tempo total de apresentação da aplicação. Após alguns segundos de exibição do vídeo, um outro começa ser exibido em tamanho menor sobre o vídeo principal. Quando este último atinge um determinado instante de tempo de exibição, uma imagem que representa um botão é apresentada, indicando interatividade. Se a ação interativa ocorrer, o botão desaparece e é exibida uma nova imagem mostrando informações sobre o vídeo principal. Caso contrário, após um tempo, o botão desaparece, terminando o intervalo de interatividade. A partir disto, somente o vídeo é apresentado até seu término natural. Na Figura 4.7, é mostrada a visão espacial da aplicação.

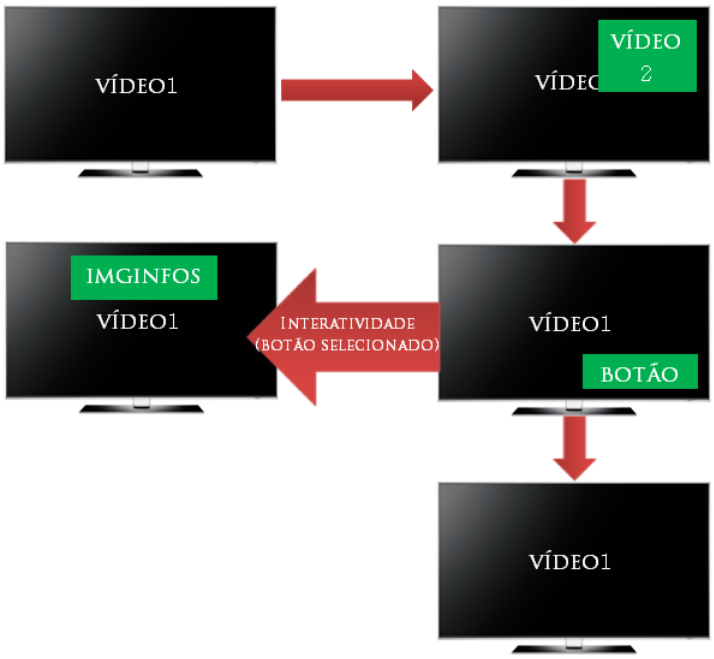


Figura 4.7: Visão espacial da aplicação exemplo

A Figura 4.8 apresenta o HTG correspondente à aplicação cuja visão espacial é mostrada na Figura 4.7, nela é omitido o tipo de evento *presentation* dos vértices para melhor organização da mesma. O vértice 1 representa o início da exibição do vídeo principal ("vídeo1"). O término do mesmo é indicado pelo vértice 2, sendo que a aresta entre os dois vértices é a duração do vídeo. O vértice 5 representa uma âncora do "video1", que tem somente o valor de tempo de início (5s), indicando o começo da âncora V ("âncoraV") que por sua vez determina o início do segundo vídeo ("vídeo2") representado pelo vértice 6. Já o vértice 7 simboliza o fim do "vídeo2" que possui duração de 10 segundos. Além desta âncora, os vértices 3 e 4 especificam outra com tempo de início (18s) e fim (24s). Ela define o começo e término da mídia "botão" representados pelos vértices 8 e 9 respectivamente. A condição da aresta entre os vértices 8 e 10 indica interatividade que deve ser realizada através do botão vermelho do controle remoto. Após o botão ser selecionado, é iniciada a imagem de informações do vídeo principal ("imgInfos") que é representada pelo vértice 11 e o término da mídia pelo 12. A mídia "botão", além de ser finalizada pelo término da "âncoraB", também é parada pelo fim da apresentação da imagem "imgInfos". É importante salientar que as arestas cuja condição possui o valor zero representa que a transição destino deve ser disparada imediatamente quando o vértice origem da condição é atingido.

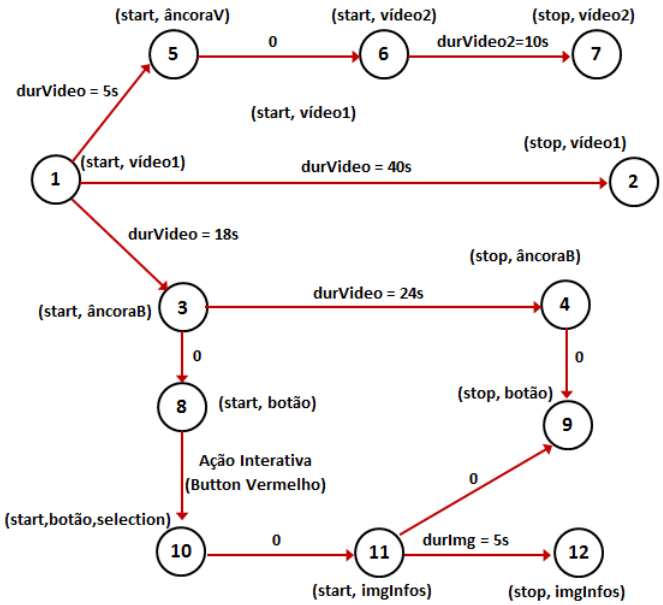


Figura 4.8: HTG para aplicação apresentada na Figura 4.7

4.3.2 Plano de Apresentação

A segunda estrutura de dados a ser apresentada, o plano de apresentação [33], é criada a partir do *Hypermedia Temporal Graph* apresentado. Ela é definida como um conjunto de transições de eventos disparados sobre os componentes da aplicação e cada uma destas transições possuem seus instantes de tempo de execução.

Construído o HTG, os valores para os instantes de tempo do disparo de suas transições de evento são calculados a partir do caminho pelas arestas do grafo. Porém, nem sempre esses valores são determinados. Ou seja, para os eventos previsíveis os instantes são completamente determinados antes da execução do documento. Já para os eventos imprevisíveis, isto é, interatividade, os instantes para o disparo das transições de evento de seleção são indeterminados *a priori*. Neste caso, as transições de evento que partem do vértice que representa a interatividade (vértice 10 na Figura 4.8) ou que derivam de vértices que dependem do vértice interativo têm seus tempos calculados relativamente à ocorrência do evento imprevisível. Por exemplo, na Figura 4.8, os vértices 11, 12 e 9 têm seus instantes de disparo calculados de forma relativa ao momento em que ocorre a ação interativa que dispara o evento de seleção representado pelo vértice 10. Note que neste caso, o vértice 9 é atingido por outro caminho que possui só eventos previsíveis, sendo assim, tem seu tempo determinado *a priori*. A Tabela 4.1 mostra o plano de apresentação para o HTG exibido na Figura 4.8 dividido em duas tabelas: a primeira 4.1a corresponde aos vértices cujas transições têm seu instante de tempo de disparo previsível e a segunda 4.1b apresenta as transições imprevisíveis.

Desta forma, uma parte do plano de apresentação possui todas as transições de eventos que têm seus instantes de tempo de disparo previsíveis e outra com as transições que possuem tempos calculados relativos ao tempo “X” em que ocorreu a ação interativa que selecionou, no exemplo, a mídia “botao”.

Transições	Instante de Tempo de Disparo
(start,vídeo1,presentation)	0s
(stop,vídeo1,presentation)	40s
(start,âncoraV,presentation)	5s
(start,vídeo2,presentation)	5s
(stop,vídeo2,presentation)	15s
(start,âncoraB,presentation)	18s
(stop,âncoraB,presentation)	24s
(start,botão,presentation)	18s
(stop,botão,presentation)	24s

(a) Transições Previsíveis

Transições	Instante de Tempo de Disparo
(start,botão,selection)	Xs
(stop,botão,presentation)	(X+0)s
(start,imgInfos,presentation)	(X+0)s
(stop,imgInfos,presentation)	(X+5)s

(b) Transições Imprevisíveis

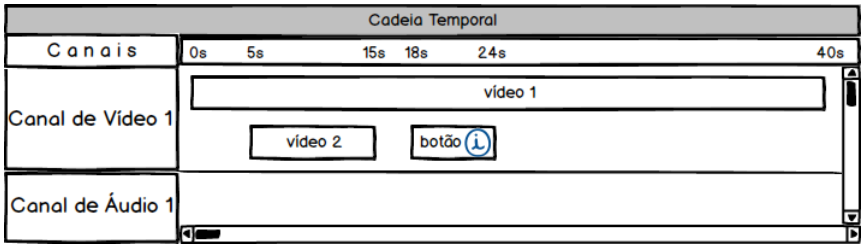
Tabela 4.1: Plano de apresentação para o HTG da Figura 4.8

4.3.3 Cadeia Temporal

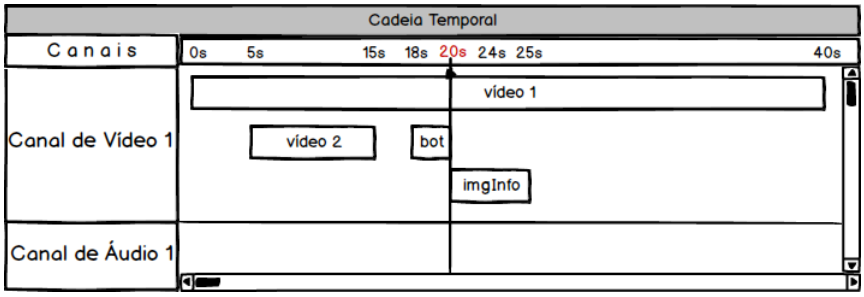
A partir do plano de apresentação especificado na Tabela 4.1, é possível determinar os instantes de tempo em que os componentes da aplicação são exibidos, ou seja, construir as cadeias temporais da aplicação. Uma cadeia temporal armazena, para cada mídia que pode ser apresentada na execução da aplicação, seu instante de início e término de apresentação. Para algumas mídias (discretas), o tempo de fim pode não ser determinado, pois, no desenvolvimento do documento hipermídia, o fim da mídia pode não ter sido especificado através de uma duração explícita (tempo definido pelo autor) e nem por um elo que a finaliza.

No caso de interatividade, algumas mídias podem ser finalizadas e outras novas apresentadas após a ação interativa ocorrer. Sendo assim, para manter a cadeia temporal consistente com a ordem de apresentação dos objetos de mídia da aplicação, além dos tempos de início e fim das novas mídias serem armazenados na cadeia temporal devido à interação, o tempo de parada dos objetos que são finalizados também devem ser guardados na cadeia. A Figura 4.9 mostra a representação gráfica da cadeia temporal para aplicação exemplo sem representar os elos entre as mídias.

Pela visão temporal do documento hipermídia exibida na Figura 4.9, a ordem de apresentação das mídias da aplicação pode ser verificada. A mídia “vídeo1” inicia sua apresentação no instante 0s e a finaliza em 40s. No tempo 5s, o segundo vídeo é iniciado e dura até quinze segundos. No intervalo de tempo 18s a 24s (intervalo de interatividade), a mídia “botão” é apresentada indicando possibilidade



(a) Cadeia temporal sem interação



(b) Cadeia temporal resultante da interação

Figura 4.9: Cadeia temporal para aplicação da Figura 4.7

de interação. Caso o "botao" não seja selecionado (não ocorreu interação), esta mídia é terminada no próprio instante de tempo 24s e somente o "vídeo1" é apresentado na tela até seu fim natural (40s) como mostrado na Figura 4.9a.

No contrário, se o botão vermelho do controle remoto for pressionado interagindo com a aplicação, uma nova representação da visão temporal, a partir da seleção do botão, é criada. No instante "X" da interação, que deve estar entre 18s a 24s, a mídia "imgInfos" é iniciada e a imagem do botão é terminada. Em seguida, a mídia "imgInfos" termina no tempo "X+5" segundos e novamente somente o vídeo principal é apresentado até os 40s. Como discutido na Seção 4.1.1, a parte interativa da cadeia temporal é representada para um dado instante de tempo dentro deste intervalo de interatividade, para demonstrar a cadeia temporal com a interação, o valor 20s foi dado ao "X" como pode ser visualizado na Figura 4.9b.

4.4 Estudo de caso

Para exemplificar o uso das estruturas de dados utilizadas na construção da cadeia temporal de aplicações NCL e descrever o funcionamento da ferramenta mostrando o passo-a-passo da geração de cada uma das estruturas, considere uma aplicação NCL exemplo, chamada "musicAd.ncl" descrita a seguir. Ela, inicialmente, exibe um vídeo de introdução ("introduction") do comercial de músicas e uma imagem, que representa um logotipo ("logo") que para de ser exibido, quando a introdução atinge o seu fim. Após este vídeo, são anunciados dois produtos ("product1" e "product2") e seus respectivos preços ("price1" e "price2"), um de cada vez, sobre uma imagem de fundo sincronizados com o áudio ("audio") que também incia assim que o vídeo de introdução atinge seu término. Ao fim do áudio, um segundo fundo ("end") é exibido e um botão ("icon") é apresentado durante um tempo (intervalo de interatividade) indicando

possibilidade de interação com a aplicação. Se o telespectador interagir pressionando a tecla vermelha do controle remoto, é mostrada uma nova imagem ("productsInfos") contendo informação extra sobre os produtos e o último fundo continua sua apresentação, porém tem seu tamanho redimensionado. Após um tempo, este fundo termina e em seguida também as informações dos produtos anunciados, assim finalizando a aplicação. Caso contrário, a aplicação termina após o segundo fundo apresentado juntamente com o botão de interatividade terminar sua exibição. A Figura 4.10 apresenta a visão espacial da aplicação "musicAd".

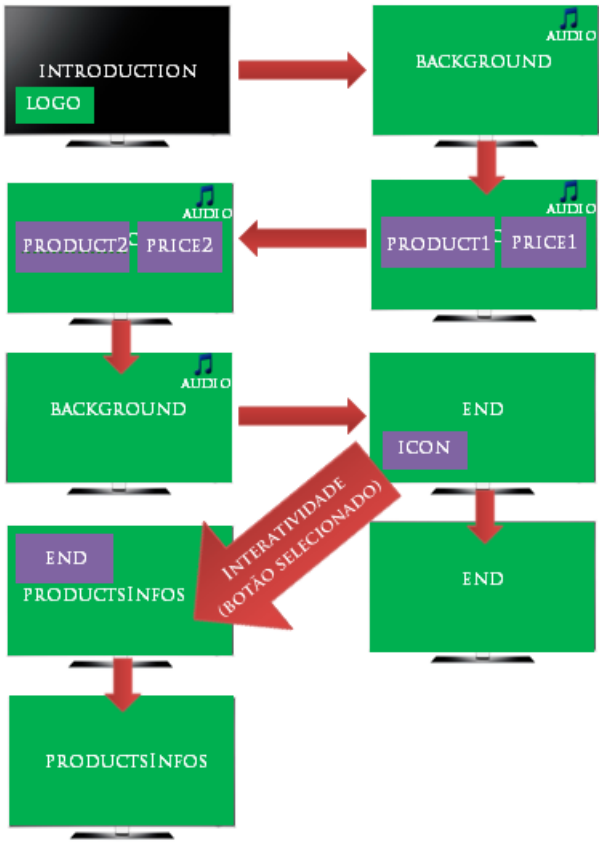


Figura 4.10: Visão espacial da aplicação NCL "musicAd"

Esta aplicação NCL possui características que ajudam na análise da construção do HTG e, consequentemente, do plano de apresentação e da cadeia temporal que será feita adiante. Tais características são descritas a seguir.

Nas próximas subseções, a construção do HTG e das suas estruturas derivadas é explicada em detalhes através de um passo-a-passo que analisa cada pedaço do documento NCL separadamente.

4.4.1 Geração do HTG

Adiante, a construção do HTG é descrita separando a geração dos vértices e arestas para as mídias e suas interfaces (âncoras de conteúdo) da geração do grafo dos relacionamentos do documento NCL exemplo.

Mídias e Interfaces

O trecho de código a seguir mostra as mídias do documento da aplicação NCL exemplo. Esses objetos de mídia são:

- vídeo de introdução (“introduction”);
- áudio do comercial (“audio”);
- a primeira imagem de fundo exibida na apresentação do documento (“background”);
- o botão que indica a interatividade para exibir as informações extras sobre os produtos anunciados (“icon”);
- o logotipo exibido sobre o vídeo de introdução (“logo”);
- o segundo fundo apresentado (“end”);
- o segundo fundo apresentado (“end”);
- uma imagem que mostras as informações extras dos produtos (“productsInfos”);
- a mídia que especifica se a aplicação permite interatividade (“globalvar”).

Além destas que estão no corpo do documento, elemento <*body*>, existem ainda as mídias que estão contidas dentro do elemento contexto “contextProduct” (<*context*>), as quais são: as imagens dos dois produtos que são anunciados (“product1” e “product2”) e seus preços (“price1” e “price2”).

```
(...)
<media id=''introduction'' src=''introduction.mpg''
descriptor=''screenDesc''>
        <area id=''anchorLogo'' begin=''2s''/>
</media>
<media id=''audio'' src=''audio.mp3'' descriptor=''audioDesc''>
        <area id=''anchorProduct1'' begin=''5s'' end=''9s''/>
        <area id=''anchorProduct2'' begin=''10s'' end=''14s''/>
</media>
<media id=''background'' src=''background.png'' descriptor=''screenDesc''/>
<media id=''icon'' src=''icon.png'' descriptor=''iconDesc''/>
<media id=''logo'' src=''logo.png'' descriptor=''logoDesc''/>
<media id=''end'' src=''end.png'' descriptor=''endDesc''>
        <property name=''bounds''/>
</media>
<media id=''productInfos'' src=''productInfos.png''
descriptor=''productInfosDesc''/>
<media id=''globalVar'' type=''application/x-ginga-settings''>
```

```
        <property name=''channel.interactivity'' value=''true''/>
</media>

<context id=''contextProduct''>
        <media id=''product1'' src=''product1.png''
        descriptor=''productDesc''/>
        <media id=''price1'' src=''price1.png'' descriptor=''priceDesc''/>
        <media id=''reusedAudio'' refer=''audio'' instance=''instSame''/>
(...)
        <media id=''product2'' src=''product2.png''
        descriptor=''productDesc''/>
        <media id=''price2'' src=''price2.png'' descriptor=''priceDesc''/>
(...)
        </context>
(...)
```

Para construir o HTG correspondente a estas mídias, para cada uma delas devem ser construídos dois vértices (início e fim) do evento de apresentação. Uma aresta ligando estes dois vértices somente é construída caso a mídia tenha duração associada. Este valor de tempo é então agregado a condição da aresta. É importante enfatizar que a duração da mídia pode ser implícita (caso das mídias contínuas) ou explícita (duração especificada no documento através do atributo *explicitDur* no elemento <*descriptor*> *da mídia*). Note que uma das mídias, "reusedAudio", faz referência a mídia "audio" com o atributo *instance* cujo valor é *instSame*. Neste caso, nada é feito no HTG se esta mídia que faz o reúso não contiver qualquer âncora de conteúdo definida.

Além disto, os objetos de mídia devem ter arestas que apontam para o início de suas âncoras (caso existam), tendo como condição da aresta o valor do início da mesma. Assim, devem ser construídos dois vértices representando o início e fim desta âncora com o valor de tempo de fim dela como condição. Caso ela somente tenha seu início especificado, apenas o vértice que representa o começo do evento de apresentação é necessário. A Figura 4.11 mostra parte do HTG referente a estas mídias do documento NCL exemplo gerado pela ferramenta.

Elos

Construídos os vértices HTG referentes as mídias e suas interfaces, devem ser analisados os elos entre os componentes do documento NCL. Os relacionamentos da aplicação podem possuir condições e ações simples ou compostas.

Relacionamentos simples são elos que possuem somente condições e ações simples. Tais *links* do documento NCL exemplo são descritos no código a seguir.

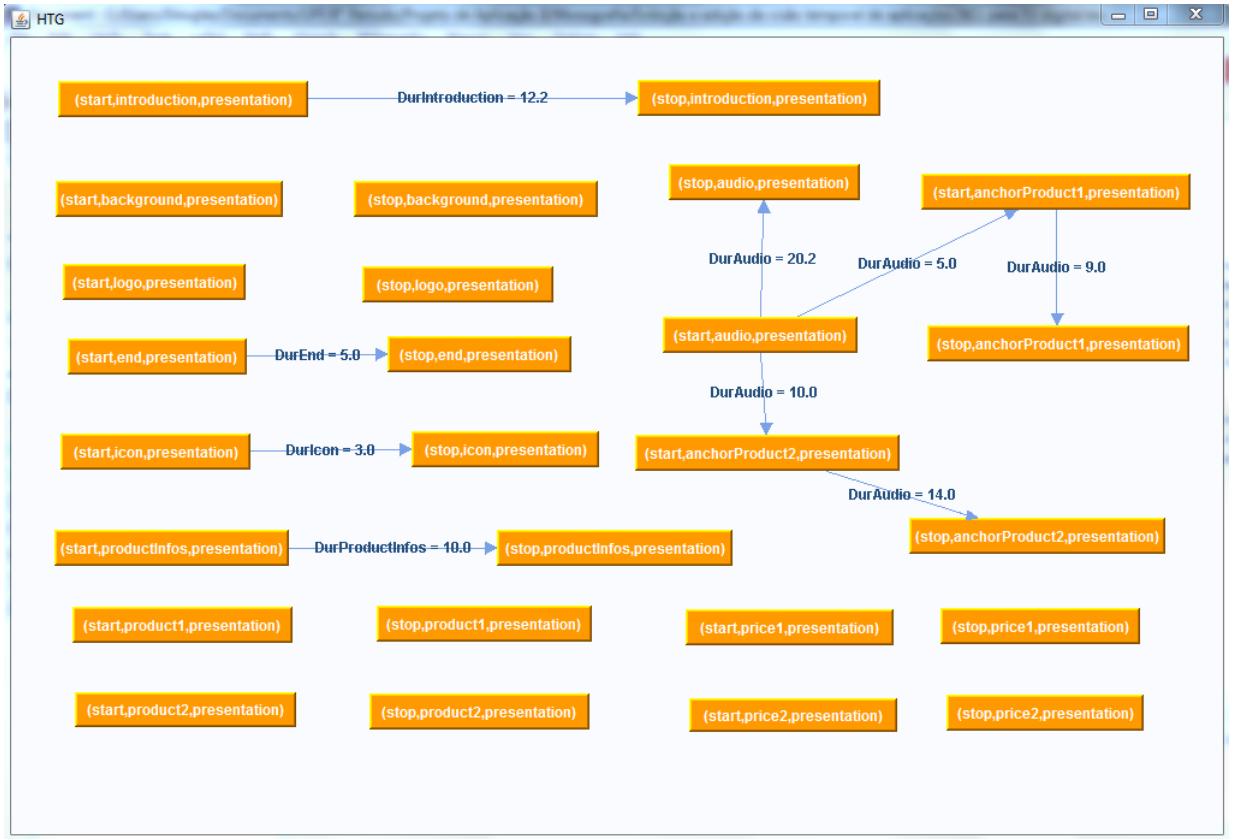


Figura 4.11: Parte do HTG que representa as mídias e suas âncoras

```
<context id=''contextProduct''>
(...)
        <link id=''lProduct1'' xconnector=''onBeginStartN''>
                <bind role=''onBegin'' component=''reusedAudio''
                interface=''anchorProduct1''/>
                <bind role=''start'' component=''product1''/>
                <bind role=''start'' component=''price1''/>
        </link>
        <link id=''lEndProduct1'' xconnector=''onEndStopN''>
                <bind role=''onEnd'' component=''reusedAudio''
                interface=''anchorProduct1''/>
                <bind role=''stop'' component=''product1''/>
                <bind role=''stop'' component=''price1''/>
        </link>
(...)
        <link id=''lProduct2'' xconnector=''onBeginStartN''>
                <bind role=''onBegin'' component=''reusedAudio''
                interface=''anchorProduct2''/>
                <bind role=''start'' component=''product2''/>
```

```
            <bind role=''start'' component=''price2''/>
        </link>
        <link id=''lEndProduct2'' xconnector=''onEndStopN''>
            <bind role=''onEnd'' component=''reusedAudio''
            interface=''anchorProduct2''/>
            <bind role=''stop'' component=''product2''/>
            <bind role=''stop'' component=''price2''/>
        </link>
</context>

        <link id=''lLogo'' xconnector=''onBeginStartDelay''>
            <bind role=''onBegin'' component=''introduction''/>
            <bind role=''start'' component=''logo''>
                <bindParam name=''delay'' value=''1s''/>
            </bind>
        </link>

        <link id=''lEnd'' xconnector=''onEndStop''>
            <bind role=''onEnd'' component=''introduction''/>
            <bind role=''stop'' component=''logo''/>
        </link>

        <link id=''lAudio'' xconnector=''onEndStartN''>
            <bind role=''onEnd'' component=''introduction''/>
            <bind role=''start'' component=''audio''/>
            <bind role=''start'' component=''background''/>
        </link>

        <link id=''lEndBackground'' xconnector=''onEndStartStop''>
            <bind role=''onEnd'' component=''audio''/>
            <bind role=''start'' component=''end''/>
            <bind role=''stop'' component=''background''/>
        </link>

(...)
```

O dois primeiros elos são responsáveis por apresentar a imagem do produto 1 e de seu preço sincronizadamente com o áudio do comercial e finalizar a apresentação dos mesmos utilizando a âncora "anchorProduct1" definida para a mídia "audio". Analogamente, os elos "lProduct2" e "lEndProduct2" sincronizam o produto 2 e seu preço com o áudio. O elo "lLogo" faz com que a mídia "logo" seja exibida

com um atraso de 1 segundo assim que o vídeo de introdução é iniciado. O relacionamento “lEnd” é responsável por terminar a apresentação desta mídia. Já para iniciar o áudio do comercial e a mídia “background” imediatamente após o término do vídeo “introduction”, o elo “lAudio” é definido. O *link* “lEndBackground” indica que ao fim do áudio a mídia “end” deve ser inciada e a mídia “background” parada.

Na construção do HTG, estes elos simples são representados apenas por uma aresta que liga o vértice que corresponde a transição de estado das mídias (ou âncoras de conteúdo) indicada pelo papel de condição e ao vértice que simboliza a ação do elo, tendo valor zero para a condição da aresta. Caso tenha mais de uma mídia associada a condição do relacionamento, esta é inserida na condição da aresta usando o operador “and” para fazer a composição destas condições. E para várias mídias associadas ao papel de ação do elo, deve construir uma aresta com origem no vértice que representa a condição do elo e destino no vértice que presenta a ação do elo para mídia ou interface associada a esta ação. A Figura 4.12 exibe o HTG construído para estes elos de condição e ação simples pelo editor gráfico da visão temporal.

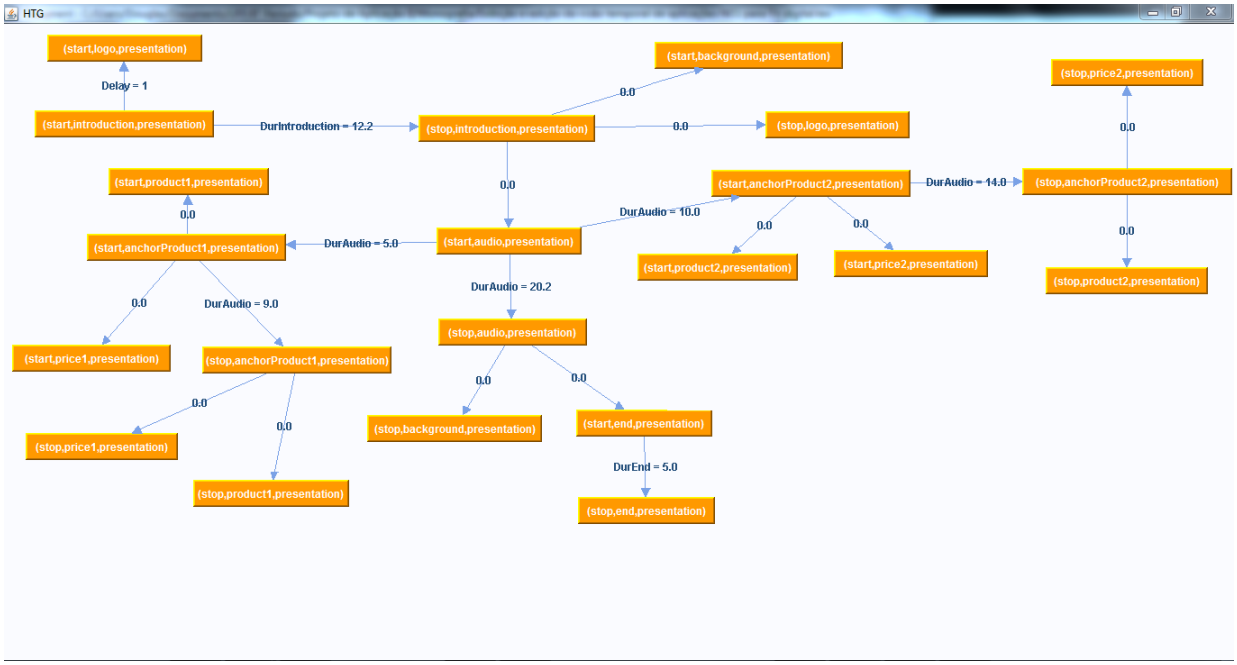


Figura 4.12: Parte do HTG que representa os elos de condição e ação simples

Relacionamentos compostos são elos que possuem pelo menos uma composição que pode ser na condição ou ação. A seguir, é mostrado o *link* da aplicação “musicAd.ncl” que representa um caso deste tipo de relacionamento.

```
(...)
<link id=''lIcon'' xconnector=''onBeginTestStart''>
        <bind role=''onBegin'' component=''end''/>
        <bind role=''testVar'' component=''globalVar''
        interface=''channel.interactivity''>
                <bindParam name=''var'' value=''true''/>
```

```
        </bind>
        <bind role=''start'' component=''icon''/>
</link>
(...)
```

Este elo é responsável por apresentar a mídia "botao" indicando a possibilidade de interação do telespectador com a aplicação. Ele é apresentado somente quando a mídia "end" atinge seu término e a aplicação permite interatividade (teste do valor da mídia "globalVar").

Para representar condições compostas no HTG, uma única aresta é construída. O vértice de origem é uma dessas condições. As demais são associadas à aresta. A composição das mesmas é realizada através do operador especificado no conector do elo ("and", "or" ou "not"). E o vértice destino é a transição de estado que representa a ação do elo. Para ação composta, mais de uma aresta é necessária para representá-la. Cada uma delas tem o destino nos vértices que representam cada uma destas ações. A Figura 4.13 exibe o HTG, gerado pela ferramenta, que corresponde ao elo "lIcon" que possui uma condição composta.

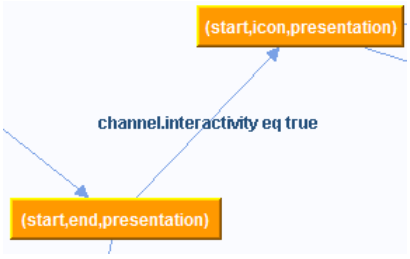


Figura 4.13: Parte do HTG que representa o elo de condição composta

Relacionamentos Interativos são elos que possuem a condição de seleção de alguma mídia (papel *onSelection*) representando interatividade. O trecho de código da aplicação exemplo que mostra este elo é exibido adiante.

```
(...)
<link id=''lProductInfos'' xconnector=''onKeySelectionStopSetStart''>
        <bind role=''onSelection'' component=''icon''>
                <bindParam name=''key'' value=''RED''/>
        </bind>
        <bind role=''stop'' component=''icon''/>
        <bind role=''set'' component=''end'' interface=''bounds''/>
        <bind role=''start'' component=''productInfos''/>
</link>
(...)
```

Este elo especifica que no momento da seleção da mídia "icon", o mesmo deve ser parado, o valor da propriedade "bounds" do componente "end" deve ser setado de uma valor especificado no conector

do elo e ainda deve ser iniciada a imagem “productsInfos”, que contém as informações extras sobre os produtos anunciados.

Este tipo de relacionamento é tratado diferentemente dos outros na construção do HTG. Deve ser feita uma aresta tendo como origem o vértice que representa o início da apresentação deste objeto que é selecionado para interação. E como destino, o vértice que indica o início do evento de seleção da mídia selecionada. A condição desta aresta possui qual é a tecla do controle remoto que deve ser pressionada para interagir com a aplicação e dispara as ações deste elo. A Figura 4.14 mostra os vértices e arestas construídos no HTG pelo editor para o elo de interatividade.

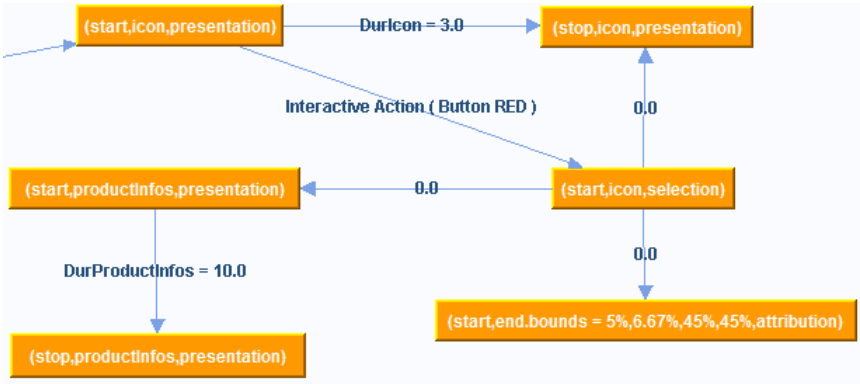


Figura 4.14: Parte do HTG que representa o elo de interatividade

O Apêndice A exibe o código completo da aplicação NCL “musicAd.ncl” mostrando além das mídias, âncoras e elos, as bases do documento, as quais são: a base de regiões, de descritores e de conectores.

O HTG completo gerado pela ferramenta para a aplicação exemplo é exibido na Figura 4.15.

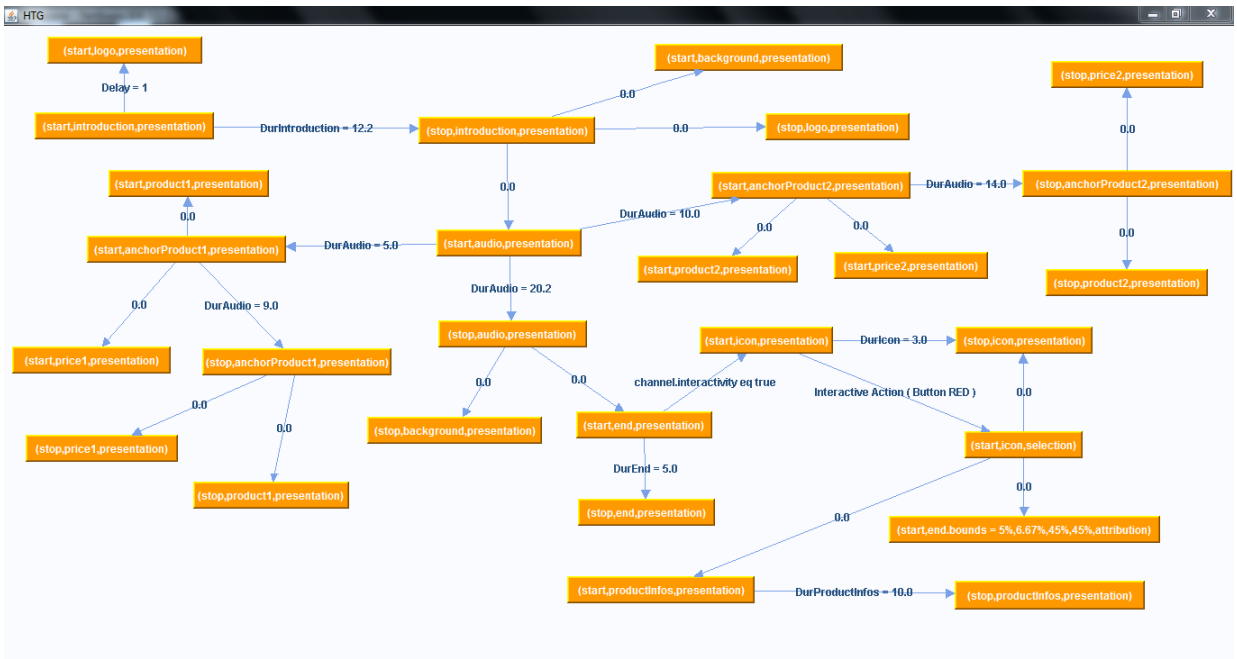


Figura 4.15: HTG completo da aplicação exemplo

4.4.2 Geração do Plano de Apresentação

O plano de apresentação é gerado a partir da leitura da HTG. Para isso, o vértice fonte do grafo deve ser encontrado, no caso do HTG da aplicação "musicAd.ncl", este vértice é definido pela tripla "(start,introduction,presentation)" que indica o início da apresentação da mídia "introduction". E o valor zero é atribuído ao instante de tempo de disparo desta transição de estado.

A leitura do grafo HTG é realizada através de uma busca em profundidade. Por cada aresta passada pela busca, o valor do intervalo de tempo que deve ser respeitado definido pela condição da aresta é somado com outros valores de tempo das arestas nas quais a busca já passou. Ou seja, cada transição de estado (vértice) possui um valor de tempo para seu disparo que é o resultado da soma dos instantes de tempo desde o vértice fonte (inicial) até esta transição.

Para o caso das arestas que possuem como condição uma ação interativa (seleção de alguma mídia ou interface), o valor de tempo para disparar o vértice destino é indeterminado. Sendo assim, neste momento a soma acumulativa dos intervalos de tempo calculados no caminho até o vértice interativo (destino da aresta que contém como condição a ação interativa), não é levado em consideração. Neste caso, este vértice que representa o início da seleção de uma âncora devido a interatividade, na aplicação exemplo este vértice é a tripla "(start,icon,selection)", recebe valor "X" para o instante de disparo de sua transição do evento de seleção indicando que este tempo é indeterminado. A partir daí, os valores de tempo que devem ser respeitados para o disparo dos vértices no caminho percorrido que parte desta transição de estado do evento de seleção são calculados relativamente ao valor "X", como é o caso dos vértices "(stop,icon,presentation)", "(start,end.bounds = 5%,6.67%,45%,45%,attribution)", "(start,productInfos,presentation)" e "(stop,productInfos,presentation)" no documento NCL exemplo. A Figura 4.16 mostra o plano de apresentação para o HTG da aplicação exemplo gerado pela ferramenta.

4.4.3 Geração da Cadeia Temporal

Com o plano de apresentação pronto, a ferramenta constrói a cadeia temporal da aplicação NCL. Para cada mídia que deve ser apresentada na execução da aplicação, ou seja, os vértices que são da forma "(start,anchorId,presentation)", são armazenadas na estrutura da cadeia temporal com seus instantes de tempo de início e fim de sua apresentação. No documento exemplo, as seguintes mídias com seus tempos do plano de apresentação devem ser armazenadas na cadeia temporal: "introduction", "audio", "background", "logo", "end", "product1", "price1", "product2", "price2", "icon", "productInfos".

No caso dos vértices cujos valores de tempo para o disparo de suas transições foram calculados em relação ao vértice interativo, a ferramenta deve ter como entrada o valor para este tempo relativo "X" criado na geração do plano de apresentação. Assim, é possível representar na cadeia temporal a ordem das mídias que são apresentadas devido a interação do telespectador com a aplicação supondo que o mesmo irá interagir neste valor de tempo atribuído ao "X". Tal valor deve estar dentro do intervalo de interatividade, o qual é especificado pelo tempo de início e fim da apresentação do objeto de mídia que indica a possibilidade de interação. Além disso, também deve ser armazenado pela cadeia temporal as mídias que foram paradas devido a interação permitindo, assim, representar corretamente as mídias que

Presentation Plan

Transições	Instantes de Execução
(start,introduction,presentation)	0.0
(stop,introduction,presentation)	12.2
(start,audio,presentation)	12.2
(stop,audio,presentation)	32.4
(start,background,presentation)	12.2
(stop,background,presentation)	32.4
(start,icon,presentation)	32.4
(stop,icon,presentation)	35.4
(start,logo,presentation)	1.0
(stop,logo,presentation)	12.2
(start,end,presentation)	32.4
(stop,end,presentation)	37.4
(start,product1,presentation)	17.2
(stop,product1,presentation)	21.2
(start,price1,presentation)	17.2
(stop,price1,presentation)	21.2
(start,product2,presentation)	22.2
(stop,product2,presentation)	26.2
(start,price2,presentation)	22.2
(stop,price2,presentation)	26.2
(start,anchorProduct1,presentation)	17.2
(stop,anchorProduct1,presentation)	21.2
(start,anchorProduct2,presentation)	22.2
(stop,anchorProduct2,presentation)	26.2
(start,icon,selection)	X
(stop,icon,presentation)	X +0.0
(start,end.bounds = 5%,6.67%,45%,45%,attribution)	X +0.0
(start,productInfos,presentation)	X +0.0
(stop,productInfos,presentation)	X +10.0

Figura 4.16: Plano de apresentação para o HTG da Figura 4.15

são apresentadas ou terminadas após a interação.

Para a aplicação “musicAd.ncl” analisada ao longo da seção, os vértices “(start,icon,presentation)” e “(stop,icon,presentation)” indicam o valor de início e fim do intervalo de interatividade respectivamente. O vértice interativo é representado pela tripla “(start,icon,selection)” e a aresta entre este vértice é o a tripla que indica o início da apresentação da mídia “icon” tem como condição que a tecla vermelha do controle remoto seja apertada (ação interativa) pelo telespectador. O vértice “(stop,icon,presentation)” da aplicação exemplo possui como valor do instante de disparo “X+0.0” indicando que a mídia “icon” é parada imediatamente quando a interação ocorre. Desta forma, é possível representar a parte interativa da visão temporal para diversos instantes de tempo que a interação pode ocorrer. No exemplo estudado, as mídias que são apresentadas ou terminadas na parte interativa da cadeia são: “icon” e “productInfos”. A Figura 4.17 apresenta as cadeias temporais geradas pela ferramenta para aplicação NCL estudada. Nota-se que existem duas cadeias temporais devido à ação interativa.

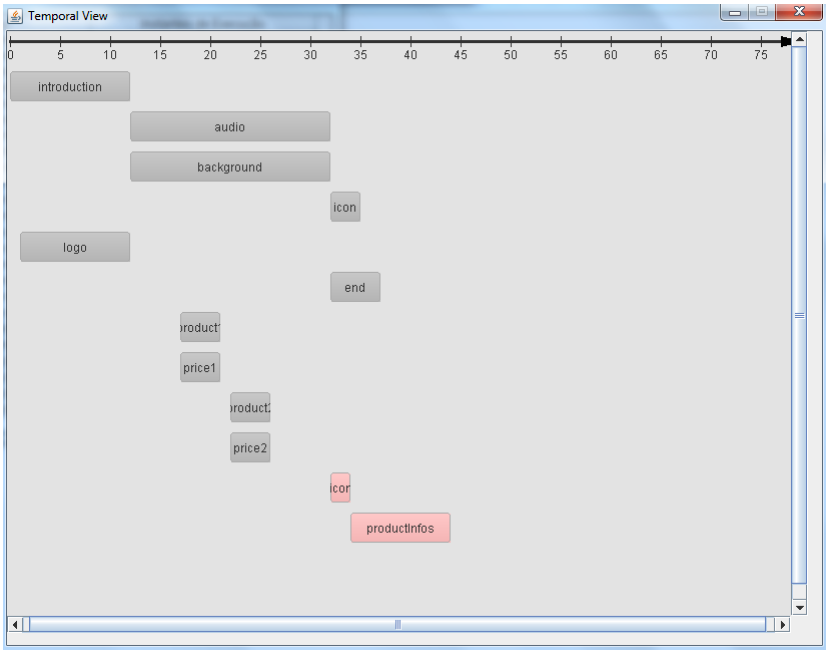


Figura 4.17: Cadeia temporal para o documento NCL “musicAd.ncl”

4.5 Interface Gráfica do Usuário

Este trabalho teve a implementação do modelo de dados para armazenar as informações necessárias do documento NCL como foco principal. Assim, a interface gráfica do usuário apresentada nesta seção é um protótipo de interface para ilustrar como o editor da visão temporal irá oferecer a visualização e edição da cadeia temporal. Esta interface é mostrada na Figura 4.18.

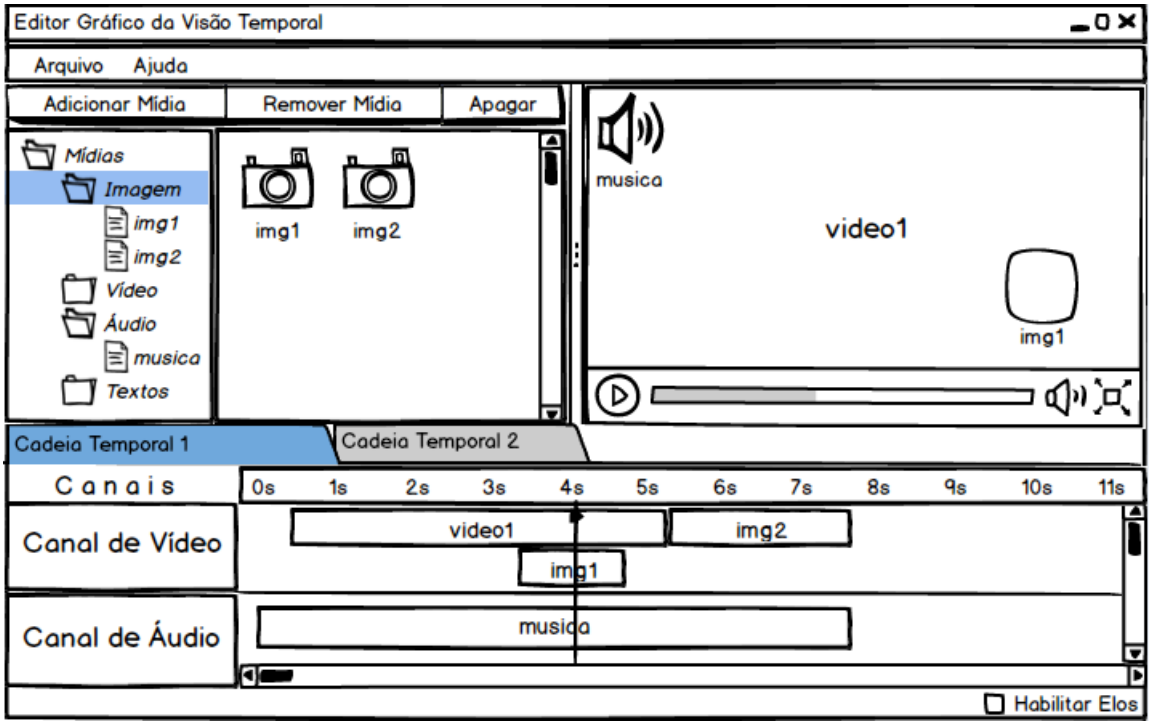


Figura 4.18: Protótipo da interface gráfica do usuário da ferramenta

Nesta interface, a cadeia temporal divide as mídias em dois canais, um canal de vídeo e outro de áudio como discutido na Seção 4.1. As mídias são representadas como retângulos com tamanho proporcional a sua duração na cadeia temporal e caso tenha âncora, esta é representada por outro retângulo menor dentro da mídia, delimitando o intervalo do trecho do conteúdo definido por esta âncora (caso esta defina o início e fim do segmento de conteúdo). As relações causais entre as mídias e âncoras são representadas com linhas tracejadas ligando os dois pontos que se relacionam. Esta cadeia temporal encontra-se na parte inferior da janela do editor como mostrado na Figura 4.18. Nela, é possível notar que existe uma barra vertical na cadeia temporal indicando o instante em que se encontra a execução do documento que é apresentado no exibidor discutido mais adiante. E ainda, no caso de documentos com interatividade, mais de uma cadeia temporal é apresentada indicando que a ordem das mídias é alterada devido a ação interativa. Para exibir estas diversas cadeias temporal para um documento interativo, abas são usadas permitindo que o usuário navegue por elas como as abas "Cadeia Temporal 1" e "Cadeia Temporal 2" mostradas na Figura 4.18.

No caso da edição da cadeia temporal, os retângulos são arrastados pela linha do tempo para posicionar as mídias no tempo de apresentação desejado. Para facilitar a escolha das mídias utilizadas na edição, um repositório de mídias é exibido na parte esquerda da janela do editor. O repositório de mídias permite o armazenamento e visualização das mídias a serem utilizadas na edição da cadeia temporal. As mídias são representadas graficamente no repositório como ícones para tornar mais fácil o processo de identificação do conteúdo desejado. Ou seja, para a mídia do tipo imagem, o ícone é representado pelo *thumbnail* (versão reduzida da imagem) da mídia; para mídia vídeo, o primeiro quadro do vídeo compõe o ícone; para os tipos de mídia áudio, texto e outros, são usados símbolos para representá-los. Além disso, para facilitar a organização do conteúdo do repositório, é utilizada uma estrutura em árvore. A árvore de mídias possui cinco nós sendo que cada um armazena um tipo de mídia, ou seja, imagem, vídeo, áudio, texto e outros. Quando um dos nós é selecionado na árvore, são apresentadas as mídias que correspondem ao tipo desse nó. Também é possível exibir todas as mídias simultaneamente, selecionando-se o nó raiz (Mídias).

Na parte direita da janela da ferramenta, é mostrada a simulação do documento NCL representado pela cadeia temporal. Este exibidor de documentos NCL será discutido com mais detalhes na conclusão quando serão analisados os trabalhos futuros. Ele permitirá que o usuário visualize facilmente como sua aplicação apresenta seus conteúdos que foram ajustados no tempo pela cadeia temporal. Desta maneira, a ferramenta oferece um ambiente, onde o usuário possa visualizar e editar a cadeia temporal de documentos NCL e ainda simular a execução da aplicação sem recorrer a um formatador NCL (*player* de documentos NCL).

Neste protótipo, existe um menu com as opções "Arquivo" e "Ajuda" responsáveis por abrir um documento NCL e salvá-lo caso a cadeia temporal tenha sofrido alguma modificação e exibir ajuda respectivamente. E os seguintes botões: um do tipo *checkbox* para habilitar e desabilitar a visualização das linhas tracejadas que simbolizam os relacionamentos entre as mídias e âncoras. Para executar o documento e controlar sua exibição, um botão de *player*, *audio* e *tela cheia* são exibidos juntamente com a barra de progresso da execução do documento. Além destes, existem outros botões que correspondem

à inclusão de mídias no repositório, exclusão da mídia selecionada e para apagar todo o conteúdo do repositório.

Este capítulo apresentou o editor gráfico da visão temporal proposto neste trabalho mostrando como ele representa as cadeias temporais de um documento NCL interativo. E ainda discutiu como poderá ser realizada a edição na visão temporal. Além disto, foi estudada cada uma das estruturas de dados utilizadas para armazenar as informações necessárias para a implementação da visão temporal de documentos NCL. A primeira destas estruturas é o *Hypermedia Temporal Graph* que a partir dela são geradas as outras, o plano de apresentação e a cadeia temporal. Apresentado o modelo de dados, foi realizado um estudo de caso com uma aplicação NCL exemplo para demonstrar em detalhes a geração de cada uma das estruturas de dados pela ferramenta para produzir a visão temporal deste documento. Ao final, é mostrado um protótipo de interface gráfica do usuário para a ferramenta proposta no trabalho.

Capítulo 5

Implementação

Neste capítulo, é discutida a modelagem realizada para a implementação do editor gráfico para exibição da visão temporal analisando os diagramas de pacotes e de classes. Além disto, os pacotes são estudados separadamente de acordo com a função de cada um, mostrando suas classes e alguns métodos importantes. O estudo da implementação, então, é dividido nos seguintes assuntos: adaptação da ferramenta ao NEXT, leitura do documento NCL, construção do HTG, geração do plano de apresentação, criação da cadeia temporal, interface gráfica do usuário da ferramenta e pacotes utilitários. É importante salientar que a classe *StartPluging* presente no pacote *myPlugin* contém o controle do fluxo da aplicação fazendo chamadas aos construtores e métodos das outras classes presentes nos diversos pacotes da implementação da ferramenta. O construtor da classe *StartPlugin* contém os principais métodos para ler e abrir o documento NCL, coletar e armazenar os dados necessários, construir o HTG e o plano de apresentação, criar a cadeia temporal e a interface gráfica do usuário. Assim esta classe é discutida em partes ao longo deste capítulo. A arquitetura da ferramenta é mostrada através do diagrama de pacotes na Figura 5.1.

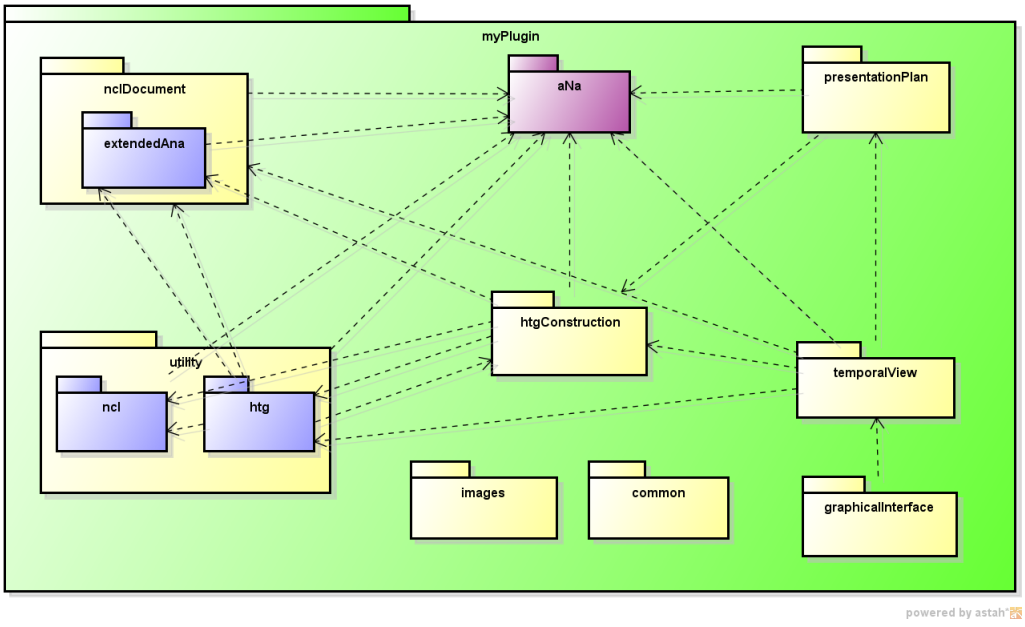


Figura 5.1: Arquitetura do Editor Gráfico da visão Temporal

5.1 Adaptação do Editor ao NEXT

Nesta seção é discutido como a ferramenta é adaptada ao NEXT para funcionar como um *plugin*, assim dando mais recursos e facilidades de edição e criação de aplicações NCL ao editor gráfico para autoria de documentos NCL com suporte a templates. A Figura 5.2 mostra as duas classes que são responsáveis por tornar a ferramenta um *plugin* do NEXT.

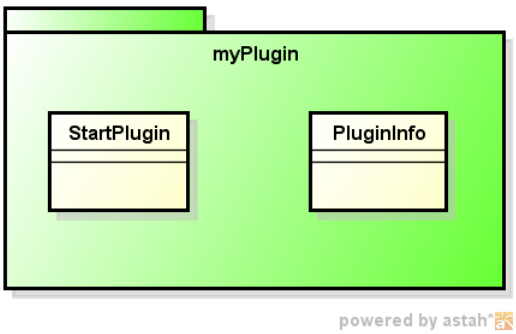


Figura 5.2: Diagrama de Classe do pacote *myPlugin*

5.1.1 Pacote *myPlugin*

Este pacote possui as classes e os métodos obrigatórios que devem ser implementados para a ferramenta se tornar um *plugin* do NEXT. Para existir a comunicação entre o *plugin* e o núcleo do NEXT, o editor gráfico da visão temporal utiliza um pacote chamado *myPlugin*, o qual implementa as classes *StartPlugin* e *PluginInfo*.

A classe *StartPlugin* estende a *JInternalFrame* do pacote *Java Swing* [9], que inicia o *plugin* quando o usuário do NEXT deseja utilizá-lo. Já a classe *PluginInfo* fornece ao núcleo informações sobre o *plugin* da visão temporal. Ela possui os métodos *getAlias* e *getNCLInterest*. O primeiro método informa o nome do *plugin* que é exibido pelo NEXT para identificar o editor gráfico da visão temporal. O método *getNCLInterest* retorna uma lista com os elementos NCL de interesse do *plugin* para que o mesmo receba notificações de modificações do documento NCL do NEXT. Assim, a ferramenta consegue manter o documento NCL consistente, o qual pode ser usado por outro *plugin* do NEXT.

5.1.2 Pacote *myPlugin.common*

Este pacote contém três classes que são comuns a todos os *plugins* desenvolvidos para o NEXT, porém estas não são obrigatórias. A primeira classe é a *Constants* que armazena constantes utilizadas no desenvolvimento de *plugins*. A segunda corresponde a *languages* que identifica o idioma escolhido pelo usuário no NEXT e guarda diversas enumerações que especificam os textos exibidos pelo *plugins* nos idiomas que são oferecidos pelo NEXT. Assim, quando o NEXT é aberto e o usuário seleciona o idioma que deseja, o *plugin* através desta classe recebe esta língua escolhida e faz com que toda frase exibida pelo *plugin* (buscada nas enumerações da classe *languages*) seja mostrada neste idioma. A última é a

classe *MyButton* que estende a classe *JButton* do *Java* e é responsável por criar os botões do *plugin* com a mesma aparência com que são exibidos na interface gráfica do NEXT.

5.2 Leitura do Documento NCL

Para realizar a leitura do documento NCL de entrada e coletar as informações necessárias para a construção do HTG, foi usada a API *aNa - API for NCL Authoring* [27] que será discutida ainda nesta seção. Além disso, para coletar informações específicas para a geração do grafo temporal hipermídia, algumas classes desta API foram estendidas para acrescentar novos métodos que atendessem a coleta de dados essenciais ao desenvolvimento da ferramenta.

As classes que foram estendidas estão presentes no pacote *myPlugin.nclDocument.extendedAna*. Para armazenar as informações lidas do documento NCL pela *aNa*, foram implementadas algumas classes que estão no pacote *myPlugin.nclDocument*. Ambos os pacotes são discutidos adiante. A Figura 5.3 mostra as classes destes pacotes omitindo as relações que indicam a extensão das classes da aNa e a implementação da interface *Node* para simplificar o diagrama.

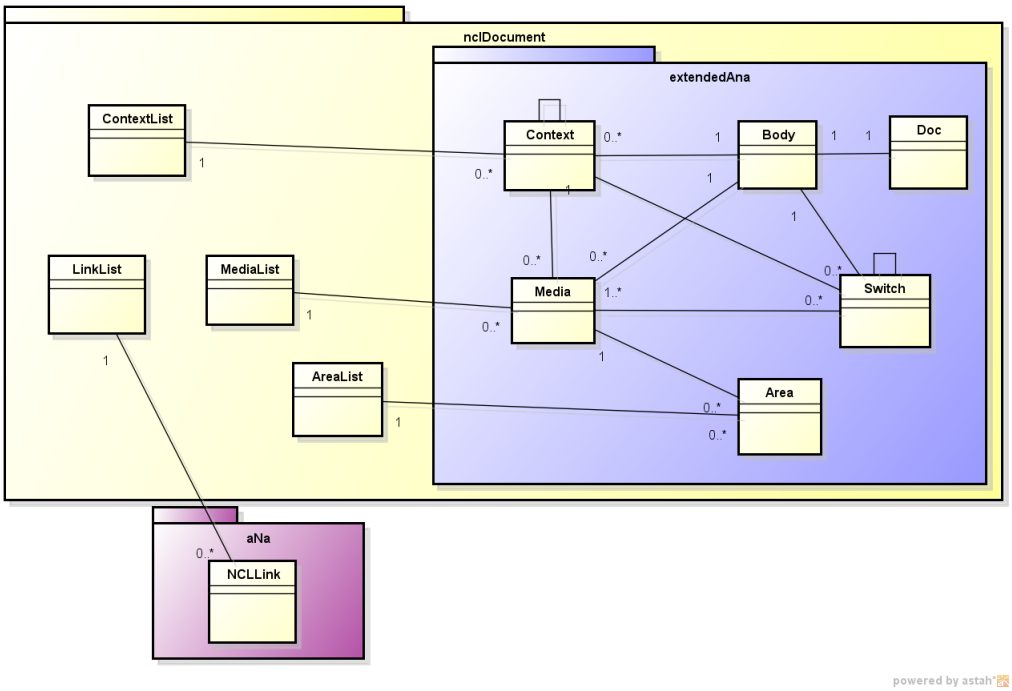


Figura 5.3: Diagrama de Classe do pacote *nclDocument*

5.2.1 API aNa

A API *aNa* realiza a leitura e criação de documentos NCL. Ela cria objetos *Java* que representam cada um dos elementos NCL que possuem diversos métodos que permitem colher informações destes elementos como também editá-los e criá-los. Para representar o documento NCL, a API cria um objeto do tipo *NCLDoc* e a partir do método *loadXML*, ela permite a leitura do documento. Tal método é usado pelo *openNCLDocument* da classe *StartPlugin* que abre o documento NCL. O objeto *NCLDoc*

contém os objetos *NCLHead* e *NCLBody* que representam o cabeçalho e o corpo do documento NCL respectivamente. O *NCLHead* possui os objetos que representam as bases de regiões (*NCLRegionBase*), descritores (*NCLDescriptorBase*) e conectores (*NCLConnectorBase*). E o objeto *NCLBody* contém as portas, propriedades, mídias, contextos, *switches* e elos do documento que são representados pelas classes *NCLPort*, *NCLProperty*, *NCLMedia*, *NCLContext*, *NCLSwitch* e *NCLLinkLPort* respectivamente. Assim, a API permite a navegação pelos elementos NCL e seus atributos buscando as informações desejadas. Para maiores informações sobre a API *aNa*, consulte [27].

5.2.2 Pacote *myPlugin.nclDocument.extendedAna*

Este pacote contém algumas classes que foram estendidas para melhor atender a busca de informações específicas necessárias para construir o HTG. Tais classes criadas para estender os objetos da API *aNa* são discutidas a seguir. As classes *Doc*, *Body*, *Context* e *Node* estendem *NCLDoc*, *NCLBody*, *NCLContext* e *NCLNode* da API respectivamente, acrescentando novos métodos necessários pra obter novas informações dos elementos.

A classe *Media* estende a classe *NCLMedia* que representa o nó de mídia no documento NCL, acrescentando o método *getDuration* que obtém a duração de mídias contínuas (duração implícita) e discretas, caso esta tenha uma duração explícita indicada pelo atributo *explicitDur* do elemento *descriptor* associado a mídia, através dos métodos privados *getImplicitDur* e *getExplicitDur* respectivamente. Além destes, tem os métodos *approximateDouble* e *convertToSeconds* para adequar os valores da duração conforme desejado. E outros métodos que auxiliam na implementação. E a última classe estendida é a *NCLSwitch* pela *Switch* que representa o elemento que faz adaptação de conteúdo numa apresentação NCL.

5.2.3 Pacote *myPlugin.nclDocument*

O pacote *myPlugin.nclDocument* contém as classes que são responsáveis por guardar todas as informações lidas do documento NCL que são essenciais na implementação da cadeia temporal de documentos NCL. A classe *AreaList* permite armazenar todas as âncoras de conteúdo das mídias do documento NCL. Já a classe *ContextList* guarda os contextos. A *LinkList* armazena todos os elos do corpo como também dos contextos do documento. E a classe *MediaList* guarda todas mídias. Elas são usadas pelo método *collectDataForHTG* da classe *StartPlugin* que insere os valores nestas estruturas.

5.3 Construção do HTG

A modelagem da estrutura do *Hypermedia Temporal Graph* é dada por quatro classes do pacote *myPlugin.htgConstruction*. O HTG é construído pelo método *constructHTG()* presente no construtor da classe *StartPlugin*. O diagrama de classe deste pacote é mostrado na Figura 5.4.

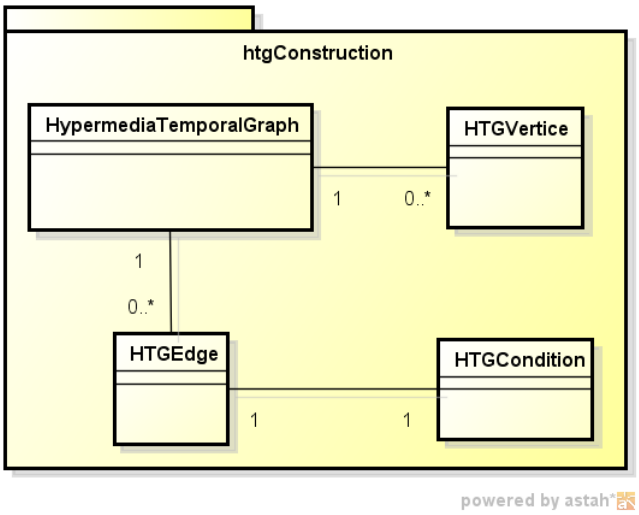


Figura 5.4: Diagrama de Classe do pacote *htgConstruction*

5.3.1 myPlugin.htgConstruction

Este pacote contém as classes *HypermediaTemporalGraph*, *HTGVertice*, *HTGEdge* e *HTGCondition*. A primeira corresponde ao grafo, tendo como atributos uma lista de vértices e arestas. Ela apresenta, como principais, os métodos *addVertice* que adiciona o vértice passado como parâmetro e o *addEdge* que inclui a aresta também passada como prâmetro de entrada no método. Ao adicionar uma aresta, o método também adiciona no vértice de origem da aresta o destino na sua lista de adjacências. Desta maneira, o HTG foi implementado utilizando o conceito de lista de adjacências, uma estrutura de dados frequentemente empregada em teoria dos grafos.

A classe *HTGVertice* tem como principais atributos para representar um vértice no HTG, as variáveis *eventAction*, *eventType* e *anchorId* para representar a tripla: ação, nome da âncora e tipo de evento da transição de estado. E tendo como método principal o *addAdj* utilizado pelo método *addEdge* da classe *HypermediaTemporalGraphxt*. Já a classe *HTGEdges* possui como atributos dois objetos *HTGVertice* representando os vértices de entrada e saída da aresta e ainda um atributo do tipo *HTGCondition* para a condição. A classe *HTGCondition* apresenta diversos atributos para representar as várias condições, estas que podem ser simples ou compostas, que uma aresta pode possuir. Dentre os métodos que ela possui, os principais são: *addDefaultCondition* que adiciona uma condição de intervalo de tempo ou uma variável com valor de tempo através de um operador lógico, *addAssessmentStatement* que adiciona uma assertiva de avaliação a condição, *addInteractivity* que inclui uma condição interativa indicando a tecla do controle remoto que deve ser pressionada para ocorrer a interação e ainda o método *addDelay* que adiciona um valor de tempo de atraso para o disparo do vértice destino (transição).

5.3.2 O método *constructHTG*

Na classe *StartPlugin*, há o método *constructHTG* que gera o HTG seguindo o passo-a-passo discutido na Seção 4.4.1. Para isso, ele possui os métodos *constructMediaVertices*, *constructAnchorVertices* e *constructLinkEdges*. O primeiro deles constrói os vértices e arestas que correspondem às mídias lidas do

documento que estão armazenadas na variável *mediaList* do tipo *MediaList*. Como descrito na Seção 4.4.1, o algoritmo deste método descreve como deve ser criada a parte do HTG para as mídias do documento NCL. O método *constructAnchorVertices* constrói a parte do grafo que corresponde às âncoras das mídias, as quais estão armazenadas na variável *areaList* do tipo *AreaList*. E o último cria as arestas necessárias para representar os elos da aplicação no grafo. Como tais *links* podem ser simples ou compostos, os métodos *constructSimpleLinkEdges* e *constructCompoundLinkEdges* são responsáveis por criar as arestas de elos simples e compostos respectivamente. Ambos os métodos seguem o passo-a-passo da criação do HTG cuja descrição foi apresentada no Capítulo 4.

5.4 Construção do Plano de Apresentação

O plano de apresentação é estruturado através da classe *PresentationPlan*, do pacote *myPlugin.presentationPlan*, que recebe o HTG construído como descrito na seção anterior para gerar os instantes de tempo de início e término, se possível, das transições de estado do grafo através do método *generatePresentationPlan* da classe *StartPlugin*.

5.4.1 myPlugin.presentationPlan

Este pacote contém somente a classe *PresentationPlan* que define uma estrutura de dados necessária para armazenar as transições de estado (vértices do HTG) e seus respectivos instantes de tempo de ocorrência. Além disto, apresenta métodos para a leitura do grafo e e obtenção dos valores de tempo como é discutido na subseção seguinte.

5.4.2 O método *generatePresentationPlan*

Este método instancia um objeto do tipo *PresentationPlan* para a criação do plano de apresentação. Após isto, os métodos *insertVertices* e *insertTimes* são executados a partir do construtor da classe *PresentationPlan*. O primeiro é responsável por ler o HTG, obtendo todos os seus vértices e armazenando-os na matriz oferecida pela classe *PresentationPlan*. O segundo faz o caminho pelos vértices e arestas do grafo para obter os valores dos instantes de tempo da ocorrência de cada transição de estado (vértice do HTG). Estes valores são calculados como descrito na Seção 4.4.2. Para isto, os métodos *findInitialVertice* e *dephFirstSearchHTG* são responsáveis por encontrar o vértice fonte do grafo e fazer a busca em profundidade necessária para inserir os valores de tempo na tabela do plano de apresentação para as transições.

5.5 Construção da Visão Temporal

A construção da cadeia temporal é realizada pelo método *createTemporalView* presente no construtor da classe *StartPlugin*. A cadeia temporal é estruturada pela classe *TemporalView* do pacote *myPlugin.temporalView* cujo diagrama de classe é exibido na Figura 5.5.

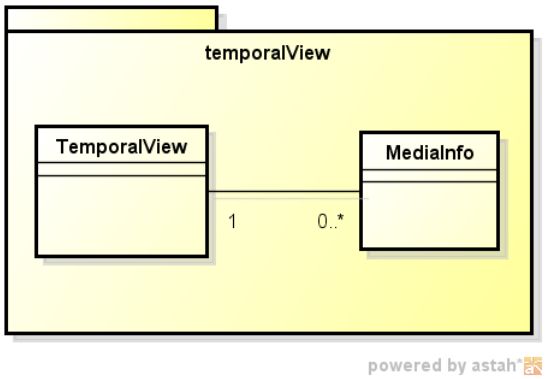


Figura 5.5: Diagrama de Classe do pacote *temporalView*

5.5.1 myPlugin.temporalView

Este pacote contém as classes *TemporalView* e *MediaInfo*. A primeira é responsável por armazenar, através de listas, as mídias que são apresentadas na cadeia temporal e seus respectivos instantes de tempo. Caso o documento NCL contenha interatividade, uma outra lista de mídias (novas ou terminadas devido a ação interativa) é criada para representar a parte da cadeia temporal, caso ocorra interatividade. A segunda representa a estrutura de cada mídia armazenada nas listas da classe *TemporalView*. Para cada mídia da cadeia temporal, é guardado seu identificador, tempo de início e parada (caso tenha) de apresentação na cadeia temporal, os quais são representados pelos atributos *id*, *startTime* e *stopTime* da classe *MediaInfo*.

5.5.2 O método *createTemporalView*

Com o plano de apresentação gerado pelo método *generatePresentationPlan*, o método *createTemporalView* cria um objeto do tipo *TemporalView*, recebendo como parâmetro de entrada este plano de apresentação, para a criação da cadeia temporal. Assim, as listas de mídias são preenchidas pela leitura da tabela do plano de apresentação conforme descrito na Seção 4.4.3.

5.6 Interface Gráfica do Usuário

O desenvolvimento da interface gráfica do usuário da ferramenta é realizado pelo método *createGUI* da classe *StartPlugin*. A representação gráfica da cadeia temporal e da janela do editor são construídos pelas classes do pacote *myPlugin.graphicalInterface*.

5.6.1 myPlugin.graphicalInterface

Este pacote contém as classes *TemporalViewPanel*, *Media* e *TimeAxis*. A primeira delas corresponde a representação gráfica da cadeia temporal proposta no Capítulo 4. Esta representação possui um eixo do tempo cuja implementação é dada pela classe *TimeAxis*, além disso a cadeia temporal pode possuir várias mídias que correspondem a objetos do tipo da classe *Media*. Como este trabalho focou na implementação

das estruturas de dados essenciais na geração da cadeia temporal, este pacote irá possuir futuramente outras classes para compor, completamente, a interface gráfica da ferramenta proposta na Seção 4.5. As imagens que fazem parte desta interface são armazenadas no pacote *myPlugin.images* para facilitar a busca das mesmas no desenvolvimento da interface.

5.6.2 O método *createGUI*

Visto que a implementação da interface gráfica da ferramenta não foi o foco deste trabalho, este método apenas instancia um objeto do tipo *TemporalViewPanel* que cria a representação gráfica da cadeia temporal do documento NCL. O construtor desta classe *TemporalViewPanel* recebe como parâmetro de entrada a estrutura de dados criada para a cadeia temporal (*TemporalView*) que foi derivada das outras estruturas desenvolvidas na implementação da ferramenta (o *Hypermedia Temporal Graph* e o plano de apresentação).

5.7 Pacotes Utilitários

Os pacotes utilitários são: *myPlugin.utilty.htg* e *myPlugin.utilty.ncl*. O primeiro possui as classes *HtgUtil* e *VerticeTriple*. A primeira classe apresenta métodos cuja função é auxiliar na obtenção e alteração de informações do HTG. Já a segunda facilita a criação das triplas dos vértices do grafo. O pacote *myPlugin.utilty.ncl* possui as classes *BindUtil*, *ConnectorUtil* e *LinkUtil* que ajudam a obter dados do elementos *bind*, *connector* e *link*, respectivamente, do documento NCL.

Neste capítulo, a implementação do editor gráfico da visão temporal foi analisada através da discussão de cada um dos pacotes criados para armazenar as classes de acordo com a função de cada uma delas. Tais funções foram: adaptação da ferramenta ao NEXT, leitura do documento NCL, construção do HTG, geração do plano de apresentação, criação da cadeia temporal, construção da interface gráfica do usuário e oferecer métodos auxiliares às demais classes. Primeiramente, a arquitetura do editor foi mostrada através do diagrama de pacotes mostrando as relações de dependência entre os mesmos. Em seguida, cada pacote foi discutido de acordo com sua função, apresentando suas classes e seus métodos principais. E para alguns dos pacotes também foi mostrado seu diagrama de classe.

Capítulo 6

Conclusão

Este capítulo final discute a contribuição do trabalho, apresentando a funcionalidade da ferramenta proposta. Além disto, uma análise com os trabalhos relacionados é realizada com o editor gráfico da visão temporal. E ainda, são discutidos os trabalhos futuros que implementarão algumas das estratégias e conceitos somente apresentados neste trabalho que tornarão o editor gráfico da visão temporal uma ferramenta de autoria de aplicações NCL através de uma linha do tempo.

6.1 Contribuição

Como a visualização e edição gráfica da cadeia temporal de aplicações hipermídia, inclusive na área de TV digital, possui grande relevância na autoria de documentos multimídia interativos, o trabalho apresentou uma ferramenta que oferece a visão temporal de documentos NCL, permitindo o fácil entendimento da ordem de apresentação das mídias deste documento. A interatividade nas aplicações NCL foi tratada pela ferramenta, propondo uma estratégia para representá-la na cadeia temporal de forma bem intuitiva, onde é possível visualizar como a parte interativa da cadeia temporal se apresenta a partir de um instante de tempo determinado, desde que este valor de tempo esteja dentro do intervalo de interatividade. Deste modo, o editor gráfico da visão temporal contribui para o processo de desenvolvimento de aplicações interativas para o sistema brasileiro de televisão digital permitindo que os autores visualizem, rapidamente, como os componentes de seus documentos se comportam no tempo. É importante enfatizar que nenhuma ferramenta atual de autoria de aplicações NCL para a TV digital brasileira oferece a visualização da cadeia temporal destas aplicações.

6.2 Análise Comparativa com Trabalhos Relacionados

Dentre os trabalhos relacionados apresentados, somente o *Composer* é uma ferramenta que auxilia a autoria de documentos NCL para o sistema brasileiro de televisão digital interativa. Este permite, como a ferramenta proposta neste trabalho, a visualização da cadeia temporal de documentos NCL. Porém, o *Composer* em sua última versão não oferece a visão temporal, fazendo com que o editor gráfico deste

trabalho seja a única ferramenta que permite a visualização das mídias de um documento NCL dispostas numa linha do tempo.

As ferramentas *CMIFed* e *GRiNs* utilizam um modelo de sincronização baseado em composições hierárquicas. Este tipo de representação especifica como as mídias dos documentos estão relacionadas no tempo através de composições paralelas e sequenciais. A visão temporal, apesar de ser simples e prática nestas ferramentas, não se adequa a documentos NCL que definem a sincronização temporal baseada em eventos. Já o sistema *FireFly* apresenta uma visão em grafo para ilustrar a ordem temporal das mídias de documentos hipermídia, sendo esta representação bastante expressiva, mas pouco intuitiva para o usuário.

Além destas ferramentas, foram analisados softwares comercias que permitem a criação de apresentações multimídia. Eles usam o paradigma de sincronização baseada em *timeline*, isto é, a construção da apresentação é realizada colocando as mídias diretamente sobre a linha do tempo. Esta representação é bastante intuitiva e fácil de ser editada, porém apresenta sérias limitações para a especificação do documento multimídia. A ferramenta proposta neste trabalho apresenta as mídias do documento NCL no tempo, porém respeita o modelo temporal baseado em eventos, de forma que a visão temporal possibilite ao usuário identificar facilmente e rapidamente a ordem das mídias de aplicações NCL, sem perder a expressividade do modelo.

O NEXT, editor gráfico de autoria de documentos NCL com suporte a templates de composição, oferece diversas visões de aplicações NCL, porém não possui a visão temporal. Sendo assim, a ferramenta deste trabalho irá enriquecer este ambiente de autoria, tornando-o mais completo para os usuários que necessitam criar suas aplicações de forma ágil e fácil para a TV digital brasileira.

Dentre os trabalhos relacionados discutidos, apenas o *Composer* trata a interatividade nas aplicações multimídia, assim como o editor gráfico da visão temporal proposto neste trabalho. Porém, como tal ferramenta não possui mais a visão temporal em sua última versão, apenas o editor deste trabalho analisa a interatividade nos documentos NCL atualmente.

6.3 Trabalhos Futuros

Visto que a implementação da ferramenta proposta focou-se no modelo de dados, ou seja, nas estruturas de dados necessárias para gerar a cadeia temporal dos documentos NCL, foi apresentada somente a estratégia para edição da cadeia temporal. A sua implementação será realizada em trabalhos futuros.

Além disto, a simulação de aplicações NCL, como mencionada na Seção 4.5, será também implementada futuramente. Visto que a ferramenta proposta será integrada no editor gráfico NEXT, ela ajudará ainda mais a autoria de aplicações NCL para TV digital brasileira, oferecendo novos recursos ao editor NEXT. Desta forma, a interface gráfica do usuário da ferramenta proposta deste trabalho apresentada como um protótipo na Seção 4.5 será implementada como trabalho futuro.

Apêndice A

Documento NCL “musicAd.ncl”

```
<?xml version=''1.0'' encoding=''ISO-8859-1''?>
<ncl id=''musicAd'' xmlns=''http://www.ncl.org.br/NCL3.0/EDTVProfile''>

<head>

        <regionBase>
                <region id=''tvReg''>
                        <region id=''screenReg'' zIndex=''1''/>
                        <region id=''endReg'' zIndex=''2''/>
                        <region id=''logoReg'' left=''5%'' bottom=''5%''
                        width=''200'' height=''55'' zIndex=''3''/>
                        <region id=''iconReg'' left=''5%'' bottom=''5%''
                        width=''200'' height=''55'' zIndex=''3''/>
                        <region id=''productReg'' left=''25%'' top=''20%''
                        width=''250'' height=''250'' zIndex=''3''/>
                        <region id=''priceReg'' left=''60%'' top=''40%''
                        width=''150'' height=''150'' zIndex=''3''/>
                </region>
        </regionBase>

        <descriptorBase>
                <descriptor id=''screenDesc'' region=''screenReg''/>
                <descriptor id=''audioDesc''/>
                <descriptor id=''logoDesc'' region=''logoReg''/>
                <descriptor id=''iconDesc'' region=''iconReg''
                explicitDur=''3s''/>
                <descriptor id=''endDesc'' region=''endReg''
```

```
        explicitDur=''5s''/>
        <descriptor id=''productDesc'' region=''productReg''/>
        <descriptor id=''priceDesc'' region=''priceReg''/>
        <descriptor id=''productInfosDesc'' region=''screenReg''
        explicitDur=''10s''/>
</descriptorBase>

<connectorBase>
        <causalConnector id=''onBeginStartN''>
                <simpleCondition role=''onBegin''/>
                <simpleAction role=''start'' max=''unbounded''
                qualifier=''par''/>
        </causalConnector>
        <causalConnector id=''onEndStopN''>
                <simpleCondition role=''onEnd''/>
                <simpleAction role=''stop'' max=''unbounded''
                qualifier=''par''/>
        </causalConnector>
        <causalConnector id=''onBeginStartDelay''>
                <connectorParam name=''delay''/>
                <simpleCondition role=''onBegin''/>
                <simpleAction role=''start'' delay=''$delay''/>
        </causalConnector>
        <causalConnector id=''onEndStop''>
                <simpleCondition role=''onEnd''/>
                <simpleAction role=''stop''/>
        </causalConnector>
        <causalConnector id=''onEndStartN''>
                <simpleCondition role=''onEnd''/>
                <simpleAction role=''start'' max=''unbounded''
                qualifier=''par''/>
        </causalConnector>
        <causalConnector id=''onEndStartStop''>
                <simpleCondition role=''onEnd''/>
                <compoundAction operator=''seq''>
                        <simpleAction role=''start''/>
                        <simpleAction role=''stop''/>
                </compoundAction>
        </causalConnector>
```

```
            <causalConnector id=''onBeginTestStart''>
                <connectorParam name=''var''/>
                <compoundCondition operator=''and''>
                    <simpleCondition role=''onBegin''/>
                    <assessmentStatement comparator=''eq''>
                        <attributeAssessment
                        role=''testVar''
                        attributeType=''nodeProperty''
                        eventType=''attribution''/>
                        <valueAssessment value=''$var''/>
                    </assessmentStatement>
                </compoundCondition>
                <simpleAction role=''start''/>
            </causalConnector>
            <causalConnector id=''onKeySelectionStopSetStart''>
                <connectorParam name=''key''/>
                <simpleCondition role=''onSelection''
                key=''$key''/>
                <compoundAction operator=''seq''>
                    <simpleAction role=''stop''/>
                    <simpleAction role=''set''
                    value=''5%,6.67%,45%,45%''/>
                    <simpleAction role=''start''/>
                </compoundAction>
            </causalConnector>
        </connectorBase>

</head>

<body>

        <port id=''entry'' component=''introduction''/>

        <media id=''introduction'' src=''introduction.mpg''
        descriptor=''screenDesc''>
                <area id=''anchorLogo'' begin=''2s''/>
        </media>
        <media id=''audio'' src=''audio.mp3'' descriptor=''audioDesc''>
                <area id=''anchorProduct1'' begin=''5s''
```

```
                end=''9s''/>
                <area id=''anchorProduct2'' begin=''10s''
                end=''14s''/>
</media>
<media id=''background'' src=''background.png''
descriptor=''screenDesc''/>
<media id=''icon'' src=''icon.png'' descriptor=''iconDesc''/>
<media id=''logo'' src=''logo.png'' descriptor=''logoDesc''/>
<media id=''end'' src=''end.png'' descriptor=''endDesc''>
        <property name=''bounds''/>
</media>
<media id=''productInfos'' src=''productInfos.png''
descriptor=''productInfosDesc''/>
<media id=''globalVar'' type=''application/x-ginga-settings''>
        <property name=''channel.interactivity'' value=''true''/>
</media>

<context id=''contextProduct''>
        <media id=''product1'' src=''product1.png''
        descriptor=''productDesc''/>
        <media id=''price1'' src=''price1.png''
        descriptor=''priceDesc''/>
        <media id=''reusedAudio'' refer=''audio''
        instance=''instSame''/>
        <link id=''lProduct1'' xconnector=''onBeginStartN''>
                <bind role=''onBegin'' component=''reusedAudio''
                interface=''anchorProduct1''/>
                <bind role=''start'' component=''product1''/>
                <bind role=''start'' component=''price1''/>
        </link>
        <link id=''lEndProduct1'' xconnector=''onEndStopN''>
                <bind role=''onEnd'' component=''reusedAudio''
                interface=''anchorProduct1''/>
                <bind role=''stop'' component=''product1''/>
                <bind role=''stop'' component=''price1''/>
        </link>
```

```
        <media id=''product2'' src=''product2.png''
        descriptor=''productDesc''/>
        <media id=''price2'' src=''price2.png''
        descriptor=''priceDesc''/>
        <link id=''lProduct2'' xconnector=''onBeginStartN''>
                <bind role=''onBegin'' component=''reusedAudio''
                interface=''anchorProduct2''/>
                <bind role=''start'' component=''product2''/>
                <bind role=''start'' component=''price2''/>
        </link>
        <link id=''lEndProduct2'' xconnector=''onEndStopN''>
                <bind role=''onEnd'' component=''reusedAudio''
                interface=''anchorProduct2''/>
                <bind role=''stop'' component=''product2''/>
                <bind role=''stop'' component=''price2''/>
        </link>

</context>

<link id=''lLogo'' xconnector=''onBeginStartDelay''>
        <bind role=''onBegin'' component=''introduction''/>
        <bind role=''start'' component=''logo''>
                <bindParam name=''delay'' value=''1s''/>
        </bind>
</link>

<link id=''lEnd'' xconnector=''onEndStop''>
        <bind role=''onEnd'' component=''introduction''/>
        <bind role=''stop'' component=''logo''/>
</link>

<link id=''lAudio'' xconnector=''onEndStartN''>
        <bind role=''onEnd'' component=''introduction''/>
        <bind role=''start'' component=''audio''/>
        <bind role=''start'' component=''background''/>
</link>

<link id=''lEndBackground'' xconnector=''onEndStartStop''>
        <bind role=''onEnd'' component=''audio''/>
```

```
            <bind role=''start'' component=''end''/>
            <bind role=''stop'' component=''background''/>
        </link>

        <link id=''lIcon'' xconnector=''onBeginTestStart''>
            <bind role=''onBegin'' component=''end''/>
            <bind role=''testVar'' component=''globalVar''
            interface=''channel.interactivity''>
                <bindParam name=''var'' value=''true''/>
            </bind>
            <bind role=''start'' component=''icon''/>
        </link>

        <link id=''lProductInfos''
        xconnector=''onKeySelectionStopSetStart''>
            <bind role=''onSelection'' component=''icon''>
                <bindParam name=''key'' value=''RED''/>
            </bind>
            <bind role=''stop'' component=''icon''/>
            <bind role=''set'' component=''end'' interface=''bounds''/>
            <bind role=''start'' component=''productInfos''/>
        </link>

</body>

</ncl>
```

Referências Bibliográficas

[1] Macromedia authorware. http://www.adobe.com/products/authorware/, 2003.

[2] *Digital terrestrial television - Data coding and transmission specification for digital broadcasting Part 2: Ginga-NCL for fixed and mobile receivers - XML application language for application coding.* ABNT 15606-2, 2007.

[3] Synchronized Multimedia Integration Language - SMIL. http://www.w3.org/TR/SMIL/, 2007.

[4] *Nested Context Language (NCL) and Ginga-NCL for IPTV.* ITU H.761 recommendation, 2009.

[5] Adobe Flash. http://www.adobe.com/devnet/actionscript/references.html., 2010.

[6] iMovie '11. http://www.apple.com/br/ilife/imovie/, 2010.

[7] Nero Vision xtra. http://www.nero.com, 2010.

[8] Adobe Prelude CS6. http://www.adobe.com/br/products/prelude.html, 2012.

[9] Api java swing. http://docs.oracle.com/javase/6/docs/technotes/guides/swing/, 2012.

[10] Final Cut Pro X. http://www.apple.com/br/finalcutpro/, 2012.

[11] J. F. ALLEN. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[12] M.C. Buchanan and P.T. Zellweger. Specifying Temporal Behavior in Hypermedia Documents. In *Proceedings of European Conference on Hypertext*, 1992.

[13] D.C.A. Bulterman, L. Hardmen, J. Jasen, K.S. Mullender, and L. Rutledge. GRiNS: A GRaphical INterface for creating and playing SMIL documents. *WWW7 Conference, Computer Networks and ISDN Systems*, 30(1-7):519–529, 1998.

[14] R.M. Coelho. Integração de Ferramentas Gráficas e Declarativas na Autoria de Arquiteturas Modeladas através de Grafos Compostos. Dissertação de mestrado, Pontifícia Universidade Católica do Rio de Janeiro, 2004.

[15] R. M. R. Costa. *Controle do Sincronismo Temporal de Aplicações Hipermídia.* Tese de doutorado, Pontifícia Universidade Católica do Rio de Janeiro, 2010.

[16] R. M. R. Costa and L.F.G. Soares. Modelos Temporais para Especificação de Sincronismo em Documentos Hipermídia.

[17] R. L. Guimarães. Composer um ambiente de autoria de documentos NCL para TV digital interativa. Dissertação de mestrado, Pontifícia Universidade Católica do Rio de Janeiro, 2007.

[18] R. L. Guimarães, R.M. Costa, and L. F. Soares. Composer: Authoring Tool for iTV Programs. *European Interactive TV Conference - EuroITV*, 2008.

[19] F. Halasz and M. Schwartz. The Dexter Hypertext Reference Model. *NIST Hypertext Standardization Workshop*, 1990.

[20] L. Hardman, G. van Rossum, and D.C.A. Bulterman. The Amsterdam Hypermedia Model: extending hypertext to support real multimedia. *Hypermedia Journal*, 5(1):47–69, 1993.

[21] R Ierusalimschy, L. H. Figueiredo, and W Celes. *Lua 5.1 Reference Manual*. Lua.org, 2006.

[22] B.S. Lima, R.G. Azevedo, M.F. Moreno, and L.F.G. Soares. Composer 3: Ambiente de Autoria Extensível, Adaptável e Multiplataforma. *WebMedia - Workshop de TV Digital Interativa (WTVDI)*, 2010.

[23] D.P. Mattos, J.V. Silva, and D.C. Muchaluat-Saade. Wizard para Autoria Gráfica de Documentos NCL com Templates. *WebMedia - Workshop de Iniciação Científica*, 2012.

[24] D. C. Muchaluat-Saade and L. F. G. Soares. XConnector & XTemplate: Improving the Expressiveness and Reuse in Web Authoring Languages. *New Review of Hypermedia and Multimedia*, 8:139–169, 2002.

[25] M.J. Pérez-Luque and T.D.C. Little. A Temporal Reference Framework for Multimedia Synchronization. *IEEE Journal on Selected Areas in Communications*, 14(1):36–51, 1996.

[26] R.F. Rodrigues. *Formatação e Controle de Apresentações Hipermídia com Mecanismos de Adaptação Temporal*. Tese de doutorado, Pontifícia Universidade Católica do Rio de Janeiro, 2003.

[27] J. A. F. Santos. Multimedia and Hypermedia Document Validation and Verification Using a Model Driven Approach. Dissertação de mestrado, Universidade Federal Fluminense, 2012.

[28] J. A. F. Santos and D. C. Muchaluat-Saade. XTemplate 3.0: Spatio-temporal Semantics and Structure Reuse for Hypermedia Compositions. *Multimedia Tools and Applications*, 61(3):645–673, 2012.

[29] J. A. F. Santos, J.V. Silva, R. Vasconcelos, W. Schau, C. Werner, and D.C. Muchaluat-Saade. aNa: API for NCL Authoring. *WebMedia - Workshop de Ferramentas e Aplicações*, 2012.

[30] J. V. Silva. NEXT - Editor Gráfico para Programas NCL com Suporte a Templates. Dissertação de mestrado, Universidade Federal Fluminense, 2012.

[31] J.V. Silva and D.C. Muchaluat-Saade. NEXT - Editor Gráfico para Autoria de Documentos NCL com Suporte a Templates de Composição. *WebMedia - Trilha de TV Digital*, 2012.

[32] L.F.G Soares and S.D.J. Barbosa. *Programando em NCL 3.0.* TeleMídia, 2 edition, 2012.

[33] L.F.G Soares and S.D.J. Barbosa. *Programando em NCL 3.0, Apêndice G - HTG.* TeleMídia, 2 edition, 2012.

[34] L.F.G. Soares and R. Rodrigues. Nested Context Model 3.0 Part 1 - NCM Core.

[35] L.F.G. Soares, R.F. Rodrigues, and D. C. Muchaluat-Saade. Modeling, Authoring and Formatting Hypermedia Documents in the HyperProp System. *Multimedia Systems archive*, 8:118 – 134, 2000.

[36] G. van Rossum, J. Jansen, K. S. Mullender, and D.C.A. Bulterman. CMIFed: A Presentation Environment for Portable Hypermedia Documents. In *ACM Multimedia*, 1993.

[37] R. Willrich, P. Saqui-Sannes, P. S'nac, and M. Diaz. *Design and management of multimedia information systems.* In: RAHMAN, S. M. (Ed.) [S.l.]: IGI Publishing.