
**Romualdo Monteiro de Resende Costa**

# Controle do Sincronismo Temporal de Aplicações Hipermídia



**Tese de Doutorado**

Tese apresentada ao Programa de Pós-graduação em Informática da PUC-Rio como requisito parcial para obtenção do título de Doutor em Informática.

Orientador: Prof. Luiz Fernando Gomes Soares

Rio de Janeiro
Agosto de 2010


**Romualdo Monteiro de Resende Costa**

# Controle do Sincronismo Temporal de Aplicações Hipermídia

Tese apresentada como requisito parcial para obtenção do grau de Doutor pelo Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

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

**Prof. Sérgio Colcher**
Departamento de Informática – PUC-Rio

**Prof. Simone Diniz Junqueira Barbosa**
Departamento de Informática – PUC-Rio

**Prof. Cesar Augusto Camillo Teixeira**
Universidade Federal de São Carlos – UFSCar

**Prof. Guido Lemos de Souza Filho**
Universidade Federal da Paraíba – UFPB

**Prof. José Eugenio Leal**
Coordenador Setorial do Centro
Técnico Científico

Rio de Janeiro, 30 de agosto de 2010

**Romualdo Monteiro de Resende Costa**

Graduou-se em Ciência da Computação pela UFMG em 2000. Obteve em 2005 o título de Mestre em Informática pela PUC-Rio. Desde 2002 é Oficial do Exército, na especialidade de informática. É pesquisador do Laboratório Telemídia da PUC-Rio.



Ficha Catalográfica

Costa, Romualdo Monteiro de Resende

Controle do sincronismo temporal de aplicações hipermídia / Romualdo Monteiro de Resende Costa; orientador: Luiz Fernando Gomes Soares. - 2010.

160 f. : il.(color.) ; 30 cm

Tese (Doutorado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2010.

Inclui bibliografia

1. Informática – Teses. 2. HTG. 3. TV digital. 4. Hipermídia. 5. Grafo temporal. 6. NCL. 7. Sincronismo. 8. Interatividade. I. Soares, Luiz Fernando Gomes. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Este trabalho é dedicado

A Aline, pelo amor e dedicação.

A toda a minha família, pelo apoio e incentivos constantes.

E a Deus, por iluminar os nossos caminhos.

## Agradecimentos

Em especial, ao meu orientador, professor Luiz Fernando Gomes Soares, agradeço pela orientação, incentivo e disposição para acompanhar cada passo deste trabalho. Meus agradecimentos, no entanto, vão muito além dos ensinamentos necessários à elaboração deste trabalho. Obrigado pela amizade, pela confiança e pelo apoio em todos os momentos.

Aos amigos do Laboratório Telemídia, agradeço pela disposição de sempre cooperar e contribuir. Ao Moreno e ao Carlos, agradeço pelo exemplo que faz desse laboratório um ambiente de trabalho agradável e, ao mesmo tempo, muito produtivo. Ao Márcio, agradeço pelos trabalhos em conjunto e pelas contribuições necessárias à realização deste trabalho. Obrigado a todos os membros desse laboratório, incluindo aqueles que já integraram a nossa equipe, pela amizade e pelo convívio proporcionados ao longo desses anos.

Agradeço também ao Exército Brasileiro, representado pelos seus ilustres diretores: Gen Garcez, Cel Ferrari e TC Paulo, que compreenderam a importância deste trabalho e ofereceram o apoio necessário para que ele pudesse ser realizado.

Aos demais amigos agradeço pelo apoio, mesmo nas vezes em que me fiz ausente, tomado pelas atividades deste trabalho. Nesse período, tive a oportunidade de conferir que as verdadeiras amizades, ao contrário das aplicações hipermídia, são atemporais.

Agradeço a todos os professores e funcionários do Departamento de Informática, e à PUC-Rio, como instituição, pelo suporte financeiro.

## Resumo

Costa, Romualdo Monteiro de Resende; Soares, Luiz Fernando Gomes. **Controle do Sincronismo Temporal de Aplicações Hipermídia.** Rio de Janeiro, 2010. 160p. Tese de Doutorado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A preservação do sincronismo das aplicações hipermídia é um dos mais importantes requisitos de qualidade para uma apresentação. Para garantir essa qualidade, é necessário conhecer os instantes no tempo de ocorrência dos eventos existentes nas aplicações. Por meio dessa informação, é possível prever a ocorrência de atrasos e, mais do que isso, antecipar as ações necessárias à apresentação, com o objetivo de evitar a necessidade de ajustes provocados por esses atrasos. Esta tese discute como as ocorrências desses eventos podem ser previstas durante uma apresentação. Para auxiliar essa tarefa, um grafo temporal, construído a partir da especificação de uma aplicação, é proposto. Esse grafo, denominado HTG (*Hypermedia Temporal Graph*), pode ser utilizado no controle do sincronismo temporal de uma aplicação, desde o transporte das mídias até a sua apresentação nos clientes. Além da preservação da qualidade, esta tese explora outras vantagens que podem ser obtidas a partir do controle do sincronismo temporal das aplicações. Esse controle pode ser utilizado, por exemplo, para posicionar as apresentações em um instante qualquer no tempo, retrocedendo ou avançando até um ponto desejado. Outra vantagem relacionada a apresentação, também explorada nesta tese, é a distribuição de partes de uma aplicação para apresentação em diferentes dispositivos, sem que o sincronismo temporal de toda a aplicação seja prejudicado. Por fim, a atualização das aplicações, simultaneamente à sua apresentação, também é explorada nesta tese.

## Palavras-chave

HTG; TV Digital; Hipermídia; Grafo Temporal; NCL; Sincronismo; Interatividade.

## Abstract

Costa, Romualdo Monteiro de Resende; Soares, Luiz Fernando Gomes. **Temporal Synchronism Control of Hypermedia Applications.** Rio de Janeiro, 2010. 160p. DSc. Thesis - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Synchronization control in hypermedia applications is one of the most important requirements in the presentation of these applications. To assure high-quality presentations, it is necessary to know when events, produced during an application execution, occur in time. By means of this information, it is possible to predict undesirable delays and to forestall actions needed to avoid presentation adjustments. This thesis discusses how event occurrences can be predicted in the course of an application presentation. In order to assist this goal, a temporal graph, built up from the application specification, is proposed. This graph, called Hypermedia Temporal Graph – HTG, can be used in the temporal control of hypermedia applications, from the media transport system up to their presentation at client sides. Besides the quality of service control, this thesis elucidates other advantages that come from the synchronization management. Among them is allowing for presentations to be started at any moment in time of their life cycle, and allowing for presentations to be moved backward and forward up to a desirable presentation moment in time. Another advantage, also dealt with in this thesis, is the support to distributing parts of an application to different devices in charge of their presentations, without causing any hazard to the whole application temporal synchronism. Finally, this thesis also proposes how editions can be made over the HTG, and, therefore, how application control can be modified during runtime.

## Keywords

HTG; Digital TV; Hypermedia; Temporal Graph; NCL; Synchronism; Interactivity;

## Sumário

## Lista de figuras

## Lista de tabelas

# 1
# Introdução

Uma aplicação hipermídia é formada por um conjunto de informações distribuídas no tempo e espaço. Assim, cada aplicação, além do seu conteúdo (vídeo, áudio, texto, imagem etc.), contém a especificação da sincronização espaço-temporal das suas mídias. Um mesmo conjunto de mídias pode ser utilizado em diferentes aplicações, cada uma com sua própria especificação de sincronização espaço-temporal.

A especificação do sincronismo temporal entre objetos de mídia em uma aplicação pode ser realizada em relação a intervalos absolutos de um eixo temporal. No entanto, essa abordagem só é adequada quando os instantes temporais dos relacionamentos entre as mídias são deterministicamente conhecidos em tempo de especificação da aplicação e não se alteram durante a sua apresentação. Nas aplicações onde o sincronismo depende da ocorrência de eventos[1] com duração variável ou mesmo imprevisível no momento da especificação, é imperativo que essa especificação seja realizada de forma relativa à ocorrência desses eventos, independente do instante temporal em que eles ocorrem e se de fato eles irão ocorrer.

Nas aplicações especificadas de forma relativa, a apresentação pode ter de ser iniciada sem que todos os eventos que deverão ser processados sejam conhecidos a priori. No entanto, quando os momentos temporais absolutos de ocorrência dos eventos não podem ser ao menos previstos, pode ser difícil preservar a qualidade da apresentação, como será discutido na próxima seção.

Nesta tese, um conjunto de estruturas de dados é proposto com o objetivo de permitir que os momentos temporais para a ocorrência dos eventos possam ser calculados com precisão em tempo de exibição, e previstos (em todas as suas possibilidades) em tempo de carga da aplicação. A principal estrutura proposta, base de todas as outras, é formada por um grafo temporal de dependências,

[1] Qualquer ocorrência no tempo de duração finita ou, na maioria das vezes, infinitesimal (Pérez-Luque, 1996).

denominado HTG (*Hypermedia Temporal Graph*), baseado nos relacionamentos entre os eventos definidos na aplicação. Os momentos absolutos do sincronismo temporal são calculados a partir desse grafo. Quando a aplicação define eventos imprevisíveis e adaptações, como usualmente acontece nas aplicações hipermídia, esses momentos somente podem ser completamente calculados durante a apresentação, porém imediatamente antes que os momentos das ocorrências dos eventos considerados sejam alcançados.

Este capítulo prossegue descrevendo funcionalidades desejáveis em um sistema hipermídia, quer para o aumento da qualidade de uma apresentação, quer para uma melhor utilização de recursos (usualmente escassos na maioria dos casos), que dependem de informações sobre o sincronismo temporal entre os eventos definidos pelas aplicações. Essas funcionalidades servem como motivações para o trabalho desenvolvido nesta tese, bem como para evidenciar as contribuições alcançadas no trabalho. Após essa breve contextualização, os objetivos desta tese são estabelecidos e a estrutura do restante deste documento é apresentada.

## 1.1. Motivação

### 1.1.1. Controle do Tempo da Apresentação e da Transmissão

O controle do sincronismo temporal nas aplicações permite que os usuários possam posicionar as apresentações em um momento qualquer no tempo, retrocedendo a apresentação ou avançando até um ponto desejado. Esse controle também é necessário para que os usuários possam pausar uma aplicação e recomeçá-la algum tempo depois, próximo ao momento da pausa (por exemplo, na troca e retorno entre canais em um sistema de TV digital) ou mesmo dias ou semanas após essa interrupção (por exemplo, na gravação de uma aplicação de TV digital em um *Personal Video Recorder* - PVR para posterior continuação). Além de útil aos usuários assistentes da aplicação (por exemplo, os telespectadores de um sistema de TV digital) em geral, essas funcionalidades (pausa, recomeço, avanço etc.) são particularmente úteis para os desenvolvedores das aplicações, que constantemente precisam visualizar os resultados das modificações realizadas em

suas especificações, sem que seja necessário recomeçar, desde o início a apresentação, a cada modificação.

Como exemplo, nos Sistemas de TV Digital Interativa (STVDI) em geral (ABNT, 2009; ARIB, 2005; ATSC 2005; ETSI, 2005), telespectadores podem, ao sintonizar um canal, começar a assistir uma apresentação (um programa de TV) já iniciada. Nesse caso, para preservar a semântica da aplicação, pode ser necessário que a apresentação seja realizada a partir do ponto em que o vídeo principal dessa aplicação esteja sendo exibido. Mais ainda, como já mencionado, telespectadores podem sintonizar diversos canais sequencialmente, entrando e saindo em várias aplicações, em diferentes pontos de suas apresentações.

Para posicionar uma apresentação em um momento qualquer de sua duração, os instantes temporais para a ocorrência dos eventos devem ser conhecidos ou probabilisticamente preditos. Nesta tese, esses instantes temporais fazem parte de um plano de apresentação, construído passo a passo no tempo, a partir do HTG construído para a aplicação, mencionado na seção anterior.

Em relação à preservação da qualidade da apresentação, a exibição de cada conteúdo de mídia em uma aplicação exige o gerenciamento dos seus exibidores, que normalmente devem ser individualmente instanciados antes dos momentos previstos para cada exibição, evitando a ocorrência de atrasos e dessincronização. A partir do HTG, esta tese também propõe a construção de um plano de carregamento de exibidores, contendo os momentos apropriados para a instanciação de cada exibidor, durante uma apresentação.

Para preservar a qualidade da apresentação, não somente os exibidores devem ser instanciados em tempo hábil, mas os conteúdos das mídias também devem estar disponíveis para apresentação. Nas apresentações distribuídas não é necessário que todas as mídias definidas em uma aplicação tenham sido recebidas para que uma apresentação seja iniciada. Na verdade, esperar que todas as mídias sejam recebidas pode inserir um atraso desnecessário, além de exigir que a máquina de apresentação disponha de espaço suficiente para armazenar, ainda que temporariamente, todo o conteúdo de uma aplicação, o que não é o caso esperado em receptores de TV digital.

O recebimento do conteúdo das mídias simultaneamente à apresentação pode ser realizado através de solicitações aos servidores. Para realizar essas solicitações em tempo hábil, é necessário não só conhecer os instantes temporais

das ocorrências dos eventos nas aplicações, mas também conhecer as características do sistema de comunicação utilizado. Para realizar a solicitação dos conteúdos das mídias, esta tese propõe a construção de dois planos, também baseados no HTG, o plano de pré-busca, para canais onde não é oferecida nenhuma forma de alocação dos recursos no sistema de transporte e o plano de QoS, para sistemas que garantam a entrega do conteúdo solicitado mediante a negociação de parâmetros, como a taxa de transferência, a variação estatística do retardo etc.

Os conteúdos das mídias, no entanto, também podem ser enviados aos clientes sem que requisições individuais sejam realizadas (*pushed data*). Nesse caso, informações sobre o sincronismo temporal das aplicações também devem estar presentes, mas agora, nos servidores. Novamente, o HTG, neste caso construído no servidor, é proposto como a estrutura básica para a construção de outro plano, o plano de distribuição que, no caso particular dos STVDI, deve controlar os momentos em que os conteúdos das mídias devem estar presentes em uma estrutura cíclica de envio dos dados, o carrossel DSM-CC (ISO/IEC, 1998) no caso da maioria dos sistemas e em particular no sistema nipo-brasileiro (ISDB-$T_B$) (ABNT, 2009).

A construção desses vários planos é muito difícil, senão impossível, quando uma aplicação é descrita em uma linguagem imperativa. Essa é a razão pela qual nenhum STVDI hoje oferece as facilidades mencionadas acima, como será discutido no capítulo sobre trabalhos relacionados. NCL (*Nested Context Language*) (ABNT, 2009; Soares, 2006a), adotada no padrão ISDB-$T_B$ e pelo ITU-T[2] para IPTV (H.761) (ITU-T, 2009), é uma linguagem declarativa baseada em eventos e essas suas características viabilizaram a proposta desta tese.

### 1.1.2. Edição Simultânea à Apresentação

A autoria e a apresentação das aplicações hipermídia são usualmente tarefas independentes. No entanto, pode não ser possível que as aplicações sejam completamente especificadas (construídas) antes do início da apresentação. Esse é o caso, por exemplo, das aplicações com conteúdos produzidos ao vivo, onde,

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

além do conteúdo ser desconhecido antes do início da apresentação, também pode não ser possível especificar todos os relacionamentos espaciais e temporais entre as mídias a priori. Em outras palavras, a aplicação é projetada “ao vivo”. Como exemplo, em uma partida de futebol transmitida ao vivo, uma pausa inesperada para atendimento médico a um jogador poderia ser utilizada para adicionar uma propaganda (outra mídia) semanticamente relacionada com o atendimento em questão.

Em algumas propostas, mencionadas no próximo capítulo, as alterações de apresentações realizadas ao vivo não são registradas na especificação da aplicação, anteriormente enviada ao sistema de apresentação. Como consequência, uma posterior apresentação da aplicação não levará em conta tais alterações.

Para preservar a semântica das aplicações geradas ao vivo, esta tese propõe uma sintaxe de transferência formada por comandos que, quando são recebidos nos clientes, modificam não apenas a apresentação, mas também a especificação das aplicações. Na verdade, comandos de edição podem ser considerados como APIs para a construção do HTG em tempo de exibição.



### 1.1.3. Apresentação Distribuída através de Múltiplos Dispositivos

Aplicações hipermídia são muitas vezes apresentadas em um único dispositivo, no entanto, principalmente em apresentações com assistência coletiva, o uso de diversos dispositivos pode agregar vantagens à apresentação. Dispositivos individuais podem ser utilizados, por exemplo, para evitar que uma apresentação seja modificada para todos, em função de interações realizadas por um usuário particular. Nesse caso, apenas o dispositivo do usuário que realizou a ação interativa deve receber o resultado da interação.

A transmissão de parte da aplicação para apresentação em um dispositivo individual, além de não interromper a assistência coletiva, permite que vários usuários possam interagir simultaneamente, o que não seria possível com um único dispositivo. Além disso, uma vez iniciada a apresentação nos dispositivos individuais, interações ou adaptações realizadas nessa apresentação individual poderiam levar a uma apresentação completamente diferente da apresentação dos demais usuários.

O uso de múltiplos dispositivos em apresentações hipermídia não é uma funcionalidade nova (Cesar, 2006; Cesar, 2007; Cesar, 2008; Dini, 2007; Schroyen, 2008), principalmente quando esses dispositivos são utilizados para entrada de dados ou mesmo para exibição dos conteúdos das mídias. Também existem várias propostas para a comunicação entre esses dispositivos. O foco da contribuição desta tese, no entanto, diz respeito a como esses dispositivos podem ser orquestrados para que o sincronismo seja preservado em uma apresentação.

Em uma aplicação, a ocorrência de um evento interativo pode provocar a distribuição de parte da apresentação para um dispositivo específico. Em seguida, o mesmo evento interativo pode ser utilizado para iniciar a mesma apresentação em outro dispositivo. Nesse caso, se os dispositivos fazem parte de um mesmo grupo, pode ser desejável que os dispositivos estejam visualizando a mesma apresentação, independente do momento em que a segunda interação, realizada a partir de outro dispositivo, ocorre. O posicionamento de parte da apresentação em um momento temporal qualquer é uma especialização do controle do tempo apresentado na Seção 1.1.1.

Sendo mais específico, o controle de uma aplicação distribuída em múltiplos dispositivos se resume no controle da distribuição do HTG. Esse é o foco da proposta deste trabalho.

Como se pode depreender, o foco desta tese está no grafo HTG: sua construção antes do início de uma aplicação; sua construção passo a passo ao vivo, através de comandos de edição, sua distribuição e controle em vários dispositivos de exibição; e a construção de vários planos para o controle da apresentação de uma aplicação com qualidade. Isso resume formalmente a próxima seção.

## 1.2. Objetivos

Considerando a importância do controle temporal das aplicações hipermídia na preservação da qualidade das apresentações e de outras funcionalidades que dependam de informações sobre os instantes de ocorrência dos eventos nas apresentações, esta tese tem como primeiro objetivo facilitar o cálculo desses instantes temporais. Para cumprir esse objetivo, estruturas são propostas, a partir

da análise das características das aplicações hipermídia e dos modelos existentes para o controle das suas apresentações.

A principal estrutura proposta é o *Hypermedia Temporal Graph*. A partir desse grafo, outras estruturas, denominadas planos, também são propostas, contendo os instantes temporais para a execução dos eventos na apresentação das aplicações: plano de apresentação, plano de carregamento de exibidores, plano de pré-busca, plano de QoS e plano de distribuição (Costa, 2008).

Em relação à apresentação, esta tese pretende implementar funcionalidades associadas ao controle do tempo, permitindo que uma aplicação possa ser interrompida e iniciada em qualquer ponto de sua duração. Essa implementação inclui as estruturas de dados propostas e, portanto, pode servir de base para a implementação, em outros trabalhos, de outras funcionalidades que também dependam do controle do tempo das aplicações.

O suporte temporal, baseado no HTG, à edição ao vivo (Costa, 2006) e à exibição em múltiplos dispositivos (Costa, 2009; Soares, 2009a) também fazem parte dos objetivos desta tese. A implementação dessas funcionalidades, incluindo a interpretação dos comandos de edição ao vivo e do controle dos múltiplos dispositivos é proposta em conjunto com outro trabalho (Moreno, 2009), voltado à especificação do Ginga-NCL (Soares, 2007; Soares, 2010a).



## 1.3. Organização da Tese

Esta tese encontra-se organizada como a seguir. O Capítulo 2 discute alguns dos principais modelos para o controle da apresentação, analisando as suas características em relação à implementação do controle do tempo, à edição ao vivo e à realização de apresentações distribuídas através de múltiplos dispositivos. O Capítulo 3 apresenta a definição do HTG e de suas estruturas relacionadas. O Capítulo 4 apresenta a representação de uma sintaxe de autoria específica, aplicações NCL, através do HTG. O Capítulo 5 trata da especificação do controle do tempo, através do conjunto de planos propostos para oferecer um controle avançado do sincronismo temporal das aplicações. O Capítulo 6 discute a edição ao vivo nas aplicações e também a realização de apresentações distribuídas

através de múltiplos dispositivos. Finalmente, o Capítulo 7 tece as considerações finais desta tese, destacando sua contribuição e os trabalhos futuros.

# 2
# Trabalhos Relacionados

Para realizar a especificação das aplicações hipermídia é desejável que estruturas com alto nível de abstração estejam disponíveis para o autor. Para apresentação, por outro lado, é desejável que as estruturas existentes facilitem a preservação dos momentos temporais absolutos para a ocorrência dos eventos definidos nas aplicações. A mesma estrutura utilizada na autoria também poderia ser adotada para o controle da apresentação. No entanto, como a autoria e a apresentação têm diferentes objetivos, elas são normalmente suportadas por estruturas diferentes. Como consequência, todos os sistemas hipermídia têm, pelo menos, duas sintaxes, uma para autoria e outra para a apresentação.

A entrega das aplicações ao sistema de apresentação pode ser realizada através de uma outra estrutura, correspondente a uma sintaxe intermediária, construída com o objetivo de facilitar a transferência das aplicações do ambiente de autoria para o de apresentação. A sintaxe de transferência pode ser utilizada, por exemplo, para que a especificação de uma aplicação seja entregue em partes temporalmente relacionadas para apresentação, permitindo que uma apresentação seja iniciada sem que toda a sua especificação tenha sido recebida. Também é possível, através da sintaxe de transferência, realizar modificações na sintaxe de apresentação de uma aplicação simultaneamente à apresentação, com o objetivo de atualizar, na apresentação corrente, as modificações realizadas pelo autor.

Muitas vezes, a sintaxe de transferência é a própria sintaxe de autoria ou a sintaxe de apresentação. O MPEG-4 Sistemas (ISO/IEC, 2001), por exemplo, define um mesmo formato binário denominado BIFS (*Binary Format for Scenes*) (ISO/IEC, 2001) tanto para a apresentação quanto para o transporte das aplicações (sintaxe de apresentação = sintaxe de transferência). Nos principais STVDI (ABNT, 2009; ARIB, 2005; ATSC 2005; ETSI, 2005), por outro lado, normalmente a sintaxe de transferência é a mesma adotada na autoria (sintaxe de autoria = sintaxe de transferência). Esse é o caso, por exemplo, do ISDB-$T_B$ (ABNT, 2009), onde aplicações NCL (ABNT, 2009; Soares, 2006a) devem ser

recebidas pelos receptores antes do início da apresentação. Além disso, durante a apresentação, comandos, contendo especificações NCL, também podem ser distribuídos aos clientes para modificar a sintaxe de apresentação existente (Costa, 2006).

Considerando que o objetivo desta tese está relacionado ao controle do sincronismo na apresentação, modelos que possam oferecer suporte a esse objetivo são o foco deste capítulo. Assim, modelos voltados à autoria, como, por exemplo, o NCM (*Nested Context Model*) (Soares, 2005a), que serve como base para a especificação em NCL, não são considerados.

Nas próximas seções são analisadas as características de modelos voltados ao controle da apresentação em relação ao controle do tempo, à edição ao vivo e à realização de apresentações distribuídas através de múltiplos dispositivos, que são as principais contribuições deste trabalho. Estruturas utilizadas na sintaxe de transferência também serão abordadas quando tiverem relação direta com as contribuições mencionadas. Características individuais dos modelos não são discutidas em detalhes neste capítulo, porém, o leitor interessado pode encontrar essas informações no Apêndice A.



## 2.1. Controle do Tempo da Apresentação e Transmissão

*Timeline* é um modelo para a definição da sincronização nas aplicações baseado em um eixo do tempo (Bertino, 1998; Blakowski, 1996), onde os pontos de sincronização são associados a momentos temporais absolutos em relação a uma base, por exemplo, o início da apresentação. Esse modelo favorece o controle do tempo, tanto para a apresentação quanto para transmissão, uma vez que todos os momentos temporais das ocorrências dos eventos são conhecidos antes do início da apresentação da aplicação. No entanto, essa forma de sincronização é ineficaz quando aplicada às aplicações hipermídia, uma vez que ela não é capaz de representar eventos com duração variável ou mesmo imprevisível.

Como exemplo da sincronização *timeline*, pode ser mencionado o BIFS, mencionado no início deste capítulo. Para a autoria, o MPEG-4 Sistemas define uma outra linguagem (XMT-O (ISO/IEC, 2001)), onde os relacionamentos entre os eventos são especificados de forma relativa. No entanto, o uso de um modelo

na autoria capaz de representar eventos imprevisíveis não impede a perda da semântica dos relacionamentos entre os eventos, quando traduzidos para o BIFS (padronizado como forma de apresentação e formato de transferência), restando apenas os momentos previstos para a ocorrência dos eventos. Essa conversão se dá pela quebra da aplicação XMT-O em várias cenas BIFS, como se a aplicação fosse dividida em várias pequenas aplicações ligadas.

Na divisão de uma aplicação original, cada uma das aplicações gerada é usualmente formada por um conjunto de mídias associadas a um mesmo *timeline* temporal, completamente previsível. Essas aplicações podem ser iniciadas ou encerradas a partir de eventos externos à aplicação. No MPEG-4 Sistemas, por exemplo, eventos interativos podem ser utilizados para encerrar a apresentação atual e iniciar uma outra apresentação. Essa estratégia também é utilizada nos STVDI europeu (ETSI, 2005), americano (ATSC, 2005) e japonês (ARIB, 2005), onde várias aplicações baseadas em XHTML (W3C, 2002) podem ter suas apresentações encerradas ou iniciadas a partir de eventos interativos. Nesses STVDI, as apresentações também podem ser controladas através de eventos de sincronismo (ISO/IEC, 1998), enviados aos clientes em momentos específicos da exibição do conteúdo audiovisual principal. A solução desses sistemas é, no entanto, pior ainda do que a utilizada no MPEG-4, que pelo menos possui uma linguagem de auto nível (XMT-O) para a especificação da aplicação global (que reúne as várias aplicações do conjunto). No caso desses outros sistemas, cabe ao autor da aplicação realizar a quebra e o controle, ao passo que em MPEG-4 ela pode ser automatizada.

Dividir uma aplicação em várias pequenas aplicações é conveniente apenas para aplicações simples e, mesmo nesse caso, a semântica original da aplicação global pode ser completamente perdida. Em seu lugar tem-se um conjunto de apresentações isoladas executadas no tempo. A semântica da aplicação original permanece apenas na idéia do autor, que, normalmente, passa a ser também o controlador da apresentação, tarefa que, para aplicações com um grande número de mídias e eventos, pode ser difícil. Nos STVDI mencionados, outra possibilidade seria utilizar uma linguagem imperativa para a especificação das aplicações, porém, variáveis, estruturas condicionais e outras características inerentes a esse paradigma de programação dificultam o controle temporal das mesmas.

Na sincronização baseada em eventos, ao contrário da sincronização *timeline*, os relacionamentos em uma aplicação são definidos de forma relativa, sem nenhuma menção ao tempo absoluto de suas ocorrências. Aplicações hipermídia podem ser completamente especificadas e apresentadas através da sincronização baseada em eventos, no entanto, essa forma de sincronização não oferece facilidades para o controle do tempo da apresentação e da transmissão dos dados, uma vez que os momentos de ocorrência dos eventos nas aplicações são desconhecidos.

Como exemplo da sincronização baseada em eventos, pode ser citado o exibidor do sistema HyperProp (Soares, 2000), que realiza a apresentação através de um modelo de execução orientado a objetos. Nesse sistema, os conteúdos das mídias e os seus eventos associados são representados por objetos, que encapsulam suas propriedades e ações (Rodrigues, 2003a). Uma ação executada sobre um objeto pode gerar notificações em outros objetos, que, por sua vez, dão origem à execução de outras ações e, assim, sucessivamente. Esse modelo é suficiente e eficiente para realizar a apresentação das aplicações, porém, não oferece facilidades ao controle temporal, definidas como alvo desta tese no Capítulo 1.

Modelos que utilizam estruturas baseadas em grafos são bastante utilizados para o controle das apresentações (Buchanan, 1992; Chung, 2005; Jourdan, 1998; Little, 1990; vanRossum, 1993; Vuong, 1995). Alguns desses modelos são especializados em relação às suas entidades, seu formato ou em relação à semântica do sincronismo que eles representam. Nesse caso, usualmente, esses modelos possuem classificação própria, como é o caso, por exemplo, do modelo hierárquico (Baecker, 1996; Bulterman, 1991) e das redes de Petri (Chung, 2005; Little, 1990; Vuong, 1995; Woo, 1994; Yang, 2003).

A sincronização hierárquica apresenta limitações tanto para a especificação quanto para o controle da apresentação (Souza, 1997). A sincronização baseada em redes de Petri, por outro lado, possui a formalização necessária para a verificação de muitas propriedades temporais que podem ser utilizadas, tanto no controle do tempo da apresentação quanto da transmissão. Todavia, essa análise formal não é normalmente o foco do controle da apresentação e transmissão das aplicações hipermídia, e sua implementação em tempo de apresentação das aplicações pode ser complexa e resultar em atrasos inaceitáveis.

Diversos outros modelos baseados em grafos, genericamente denominados grafos temporais, podem ser utilizados para controlar as apresentações (Bertino, 1998; Buchanan, 1992; Jourdan, 1998; vanRossum, 1993). O exibidor CMIFed (vanRossum, 1993), por exemplo, utiliza uma estrutura denominada grafo de dependências temporais, construída a partir da estrutura hierárquica de uma aplicação CMIF (CWI *Multimedia Interchange Format*) (Bulterman, 1991), base da especificação da linguagem SMIL (*Synchronized Multimedia Integration Language*) (W3C, 2008b), que é a recomendação W3C[3] para a especificação de aplicações multimídia na Web.

No grafo de dependências temporais do CMIFed, os vértices representam o início e o fim dos eventos e as arestas as restrições temporais dos relacionamentos entre dois vértices. Durante a apresentação, os vértices vão sendo marcados. Um vértice especial, que representa o início da apresentação, é o primeiro a ser marcado. Ao ser marcado, as restrições temporais associadas as arestas de saída de um vértice geram atrasos (condições de sincronização temporal) que devem ser individualmente respeitados para cada aresta. Quando todas as arestas de entrada de um vértice têm seus tempos atingidos, esse vértice destino é marcado. A aplicação termina quando um outro vértice, que representa o fim da apresentação, é marcado.

O grafo do CMIFed oferece facilidades temporais como, por exemplo, a sinalização de que o término de uma apresentação foi alcançado (o vértice de fim da apresentação é marcado). No entanto, como todos os vértices folhas do grafo devem ser alcançados, essa estrutura não favorece a representação de aplicações adaptativas, que poderiam levar a apresentações diferentes, dependendo, por exemplo, das escolhas dos usuários ou das suas características, ou ainda das características do sistema de apresentação.

O exibidor Ambulant (Bulterman, 2004; Cesar, 2006) é uma ferramenta mais recente, evolução do CMIFed, proposta como referência para a apresentação de aplicações SMIL. Para controlar a apresentação, o Ambulant constrói um grafo temporal formado a partir dos relacionamentos hierárquicos e dos relacionamentos entre os eventos existentes na aplicação. Diferente do CMIFed, cada vértice do grafo do Ambulant possui uma máquina de estados que controla o estado do

[3] World Wide Web Consortium – http://www.w3.org

evento de apresentação de um mídia ou de uma composição (Bulterman, 2004). A execução dos eventos é realizada a partir dos relacionamentos hierárquicos ou a partir de transições ocorridas na máquina de estados de um nó. Assim, embora essa estrutura seja suficiente para controlar a apresentação, ela possui características muito próximas da sincronização baseada em relacionamentos entre eventos, identificadas anteriormente no formatador HyperProp e, portanto, não oferece facilidades para o controle temporal das apresentações.

Entre os diversos modelos baseados em grafos disponíveis, o modelo proposto no sistema Firefly (Buchanan, 1992; Buchanan, 1993; Buchanan, 2005) merece destaque em função do controle temporal que ele oferece. Nesse modelo, um conjunto de grafos é utilizado para representar os relacionamentos definidos em uma aplicação. A sequência de eventos previsíveis, a partir do evento de apresentação da mídia que inicia a aplicação, forma um grafo denominado principal. Para cada evento imprevisível, cujo tempo de execução não pode ser calculado antes do início da apresentação, um grafo auxiliar é construído.

Durante a apresentação, cada grafo auxiliar no Firefly é adicionado ao grafo principal tão logo o tempo de execução do evento inicialmente imprevisível que originou esse grafo auxiliar possa ser calculado. Dessa forma, quando o último evento imprevisível da aplicação é conhecido, o grafo temporal é completamente obtido, bem como o *timeline* da aplicação. Essa estrutura é adequada para o controle do tempo da apresentação e, portanto, sua proposta foi o ponto de partida desta tese. No entanto, essa estrutura é incompleta, uma vez que ela não preserva os relacionamentos entre os eventos. O grafo resultante representa apenas os momentos temporais associados à execução dos eventos em uma aplicação e não a semântica dos relacionamentos entre os eventos. Além disso, essa única estrutura não é capaz de representar, simultaneamente, todas as especificações temporais que merecem ser controladas, incluindo, por exemplo, a representação simultânea do controle da apresentação, dos momentos para solicitação dos conteúdos das mídias aos servidores, dos momentos para carregamento dos exibidores etc. O HTG pode ser considerado uma extensão dos grafos Firefly.

## 2.2. Edição Simultânea à Apresentação

Para atualizar nos clientes as modificações realizadas sobre as especificações das aplicações, uma opção é utilizar a mesma sintaxe de apresentação também para a transferência. Essa é, como mencionada, a estratégia do MPEG-4 Sistemas, onde a sincronização *timeline*, através do BIFS, é utilizada tanto na transferência quanto na apresentação das aplicações. Nesse sistema, durante a apresentação, é possível que os conteúdos das mídias sejam atualizados ou mesmo que objetos sejam inseridos, removidos ou que tenham os valores de suas propriedades modificados. No entanto, quando a sintaxe de transferência é a mesma utilizada na apresentação, as modificações realizadas não são registradas na especificação da aplicação, anteriormente enviada ao sistema de apresentação. Nesse caso, apresentações posteriores poderão não refletir as modificações realizadas. Isso pode acontecer com aplicações XMT-O, quanto a atualização é realizada através do BIFS.

Nos STVDI europeu, americano e japonês, aplicações baseadas em XHTML são transmitidas e apresentadas nos clientes. Durante a apresentação, outras dessas aplicações podem ser transmitidas, tanto para serem exibidas em paralelo quanto para substituir as aplicações existentes. Em ambos os casos, através das modificações nessas aplicações, a semântica da "aplicação original", construída a partir de várias dessas aplicações XHTML, pode ser modificada simultaneamente à sua apresentação. Eventos de sincronismo também podem ser utilizados na atualização das aplicações. Nesses STVDI, no entanto, para preservar a especificação da aplicação global (que reúne as várias aplicações do conjunto), o autor precisa manter o controle da versão apresentada, que inclui o histórico das aplicações XHTML e dos eventos de sincronismo executados. Essa tarefa, para aplicações com um grande número de mídias, eventos e atualizações, é, no mínimo, ineficiente e muito suscetível a erros. Além disso, esse controle não é transferido ao cliente que, ou perde a semântica global de uma aplicação, ou tem de inferi-la a partir dos comandos da sintaxe de transferência, o que em geral é, no mínimo, muito difícil.

Nos STVDI mencionados, linguagens imperativas também podem ser utilizadas na especificação das aplicações. Nesse caso, a sintaxe de transferência,

a mesma utilizada na autoria, seria formada por especificações de uma linguagem de programação. Linguagens interpretadas, como a linguagem Java[4] base da especificação imperativa nos STVDI, permitem que modificações sejam realizadas simultaneamente à apresentação. Particularmente, Java possui outras características favoráveis à edição simultânea à apresentação, como a alocação dinâmica de memória, incluindo a remoção de objetos que não são mais necessários (*garbage collector*). No entanto, independente dessas facilidades, pode ser difícil associar as modificações realizadas sobre a especificação de um programa interativo com a sua apresentação. Essa deve ser a razão pela qual os STVDI não permitem modificações simultâneas à apresentação, quando as aplicações são especificadas de forma imperativa.

Nesta tese, uma sintaxe de transferência é proposta com base na especificação utilizada na autoria, a fim de preservar a semântica das aplicações geradas ao vivo. No sistema de apresentação, as modificações recebidas, se relacionadas ao sincronismo temporal, irão provocar alterações na estrutura de apresentação, que corresponde ao HTG da aplicação.

A sintaxe de transferência proposta, que será apresentada no Capítulo 6, é muito próxima à de autoria. Na verdade, essa sintaxe espelha as modificações do autor através de comandos, não sendo adequada, portanto, ao controle da transmissão. Por esse motivo, o HTG também é calculado no ambiente de autoria (servidor), não como uma sintaxe a ser usada na autoria ou na transferência, mas para ser utilizada no controle da transmissão de dados sem solicitação (*pushed data*).

A edição simultânea à apresentação, embora seja necessária à especificação de aplicações hipermídia, como apresentado no Capítulo 1, ainda não é uma funcionalidade facilmente encontrada em ferramentas destinadas à apresentação dessas aplicações (Costa, 2006). No exibidor Ambulant, por exemplo, é necessário que uma aplicação SMIL seja completamente recebida antes do início da sua apresentação. Uma vez iniciada a apresentação nesse exibidor, qualquer modificação na especificação da aplicação não será refletida na sua apresentação.

[4] http://www.sun.com/java

## 2.3. Apresentação Distribuída através de Múltiplos Dispositivos

O uso de múltiplos dispositivos em apresentações hipermídia, como mencionado no Capítulo 1, não é uma funcionalidade nova (Cesar, 2006; Cesar, 2007; Cesar, 2008; Dini, 2007; Schroyen, 2008), principalmente quando o controle e a distribuição do conteúdo são realizados de forma centralizada. O exibidor Ambulant, por exemplo, pode atuar como um servidor em uma arquitetura onde é permitido aos dispositivos visualizar uma apresentação comum e selecionar uma parte específica para apresentação individual (Cesar, 2007; Cesar 2008). As ações executadas nos clientes (dispositivos) são transmitidas ao servidor, onde são processadas pelo Ambulant, que pode retornar uma aplicação, contendo a disposição espacial das mídias para apresentação no dispositivo onde a seleção foi realizada, em uma verdadeira arquitetura cliente-servidor (Soares, 2009a).

Quando a distribuição da apresentação aos dispositivos é realizada de forma centralizada, os requisitos para o controle da apresentação são praticamente os mesmos de quando a apresentação é realizada através de um único dispositivo. A principal diferença está na comunicação entre os dispositivos. Os aspectos dessa comunicação (controle da presença dos dispositivos, protocolos de comunicação entre os dispositivos etc.), no entanto, não são o foco desta tese que, embora considere a possibilidade do controle centralizado, está mais relacionada com a realização desse controle de forma distribuída, onde um cliente pode até mesmo tornar-se um servidor, em um verdadeiro controle distribuído hierárquico (Soares, 2009a).

No cenário distribuído hierárquico, faz-se necessário que a estrutura utilizada no controle da apresentação facilite a fragmentação da especificação de uma aplicação durante a sua apresentação. Essas partes, por sua vez, devem ser distribuídas, através de uma sintaxe de transferência, aos dispositivos desejados, cada parte a um dispositivo diferente. Posteriormente, pode ser necessário que essas partes obtidas possam ser novamente fragmentadas e distribuídas e, assim, sucessivamente.

Devido ao fato dos momentos temporais para a ocorrência dos eventos serem previamente conhecidos na sincronização *timeline*, ela pode favorecer a

fragmentação temporal das aplicações. Curiosamente, o MPEG-4 Sistemas não especifica uma forma direta de apresentação através de múltiplos dispositivos, apesar de favorecer a divisão de uma aplicação em várias outras aplicações, bem como a distribuição das aplicações obtidas a partir de uma aplicação original. Para apresentação através de dispositivos móveis, o MPEG-4, na sua parte 20 (ISO/IEC, 2006), sugere a utilização de uma outra linguagem, o LASeR (*Lightweight Application Scene Representation*) que possui construções baseadas, principalmente, no SVG (*Scalable Vector Graphics*) (W3C, 2003) e SMIL.

A sintaxe de apresentação para aplicações LASeR pode variar de acordo com a ferramenta utilizada para a sua apresentação.[5] É provável, no entanto, considerando as características do SVG e SMIL, que as ferramentas, em geral, utilizem um formato similar ao proposto pelo Ambulant, que não favorece o particionamento temporal das aplicações nem a transferência de partes específicas da aplicação original.

Em relação à apresentação de aplicações SVG, em (Concolato, 2007) os autores sugerem a fragmentação temporal dessas aplicações com o objetivo de otimizar a utilização da memória no sistema de apresentação. No entanto, nesse trabalho são apresentadas apenas aplicações simples e, mesmo nessas aplicações, a estratégia para fragmentação, que eventualmente também poderia ser utilizada na apresentação através de múltiplos dispositivos, não é discutida. Assim, (Concolato, 2007) é citado apenas como um exemplo das muitas possibilidades da fragmentação temporal das aplicações hipermídia, além, evidentemente, da apresentação através de múltiplos dispositivos.

Considerando o suporte limitado que, em geral, é oferecido para que as aplicações possam ser temporalmente fragmentadas durante a apresentação, alguns trabalhos propõem que a própria especificação das aplicações já seja realizada de forma fragmentada. Esse é o caso, por exemplo, do ambiente TERESA (Paterno, 2008), que permite a autoria multimodal de aplicações. Nesse ambiente, as aplicações são especificadas através de uma sintaxe abstrata que pode ser convertida para várias sintaxes de apresentação e de transferência padronizadas para diferentes plataformas (Paterno, 2008). Essa proposta, no

[5] O MPEG-4 especifica um conjunto de ferramentas que servem como software de referência (ISO/IEC, 2000a) desse padrão, porém, atualmente, não está disponível uma implementação de referência para a apresentação de aplicações LASeR.

entanto, exige que a escolha sobre a distribuição da apresentação seja realizada ainda na autoria e de forma estática, quando seria interessante que essa distribuição pudesse ser realizada de forma dinâmica, durante a apresentação. Realizada de forma dinâmica, a distribuição de uma apresentação pode ser realizada de forma adaptável, quer seja pelas escolhas realizadas pelos usuários ou mesmo em função das características do ambiente, incluindo se existem dispositivos apropriados à apresentação no momento atual.

Nos STVDI que suportam aplicações imperativas, o autor pode especificar que uma aplicação deve ser apresentada através de múltiplos dispositivos. Esse é o caso, por exemplo, do Ginga-J (Souza, 2007), a parte imperativa do *middleware* Ginga, onde classes e métodos baseados em Java (Souza, 2007) estão disponíveis para controlar os dispositivos existentes em um sistema de apresentação. Nesse caso, no entanto, o autor deve implementar todo o controle dos dispositivos e a especificação da fragmentação de uma aplicação, o que pode ser uma tarefa difícil, principalmente se o autor não for um especialista em programação Java com conhecimento da API Ginga-J, no caso, ou da API de outro STVDI específico. Além disso, a especificação do autor para a distribuição das aplicações torna-se parte da aplicação, devendo ser realizada, novamente, para cada aplicação desejada.

Na proposta desta tese, a distribuição e a possibilidade de distribuição são determinadas, declarativamente, no ambiente de autoria. No entanto, a distribuição é hierárquica e ciente de contexto, sendo realizada de forma adaptativa durante a apresentação de uma aplicação, levando em conta inclusive as interações no cliente receptor. Para tanto, a sintaxe de apresentação deve poder ser hierarquicamente particionada e distribuída do receptor ancestral a seus receptores filhos.

# 3
# Hypermedia Temporal Graph – HTG

Com o objetivo de atender aos requisitos desejáveis em uma estrutura voltada ao controle da apresentação, mencionados nos capítulos anteriores, um grafo temporal de dependências, denominado HTG (*Hypermedia Temporal Graph*), é proposto neste capítulo, como descrito a seguir.

## 3.1.
## Modelo de Grafos Temporais

### 3.1.1.
### Definição

Na sincronização baseada em grafos temporais, os relacionamentos são representados através de arestas dirigidas, enquanto os vértices representam os eventos que podem ocorrer sobre os conteúdos das mídias (Bertino, 1998). O grafo proposto nesta tese pode representar tanto eventos previsíveis quanto imprevisíveis, ambos normalmente presentes nas aplicações hipermídia e, portanto, esse grafo é chamado de Grafo Temporal Hipermídia (*Hypermedia Temporal Graph* - HTG).

No HTG, vértices representam transições de estado de um evento. Várias transições de estado de eventos podem ser representadas por esse grafo, particularmente, transições de estado de eventos de apresentação, seleção e de atribuição. Cada evento é controlado por uma máquina de estados, apresentada na Figura 3.1, que define as transições possíveis.

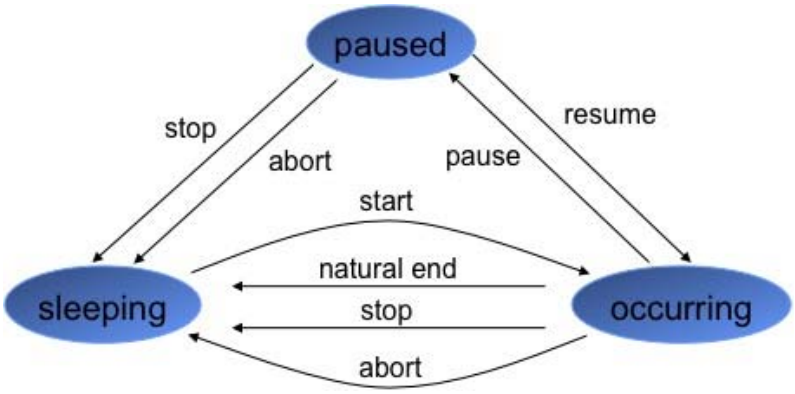


Figura 3.1 - Máquina de estado de um evento.

O controle formal dos estados de um evento é necessário para que seja possível determinar, com precisão, qual o comportamento esperado de um evento, considerando o seu estado atual e a transição realizada. Pode-se, assim, dizer que o HTG representa a ligação das máquinas de estado de eventos de uma apresentação hipermídia. Na Figura 3.1, os estados e as ações que executam as transições dos estados são apresentados em inglês apenas para corresponder à estrutura utilizada na linguagem NCL.

Como pode ser observado na Figura 3.1, um evento permanece dormindo (*sleeping*) até ser iniciado, através da transição de início (*start*). Uma vez iniciado, um evento está ocorrendo (*occurring*), podendo ser encerrado, quando retorna ao seu estado inicial, como consequência de uma transição de fim (*stop*, *abort* ou *natural end*) ou sofrer uma transição de pausa (*pause*), que modifica o estado do evento para pausado (*paused*). Nesse último estado, o evento pode ser encerrado, voltando ao seu estado inicial, ou ser recomeçado, voltando ao estado ocorrendo, dependendo da transição realizada (*stop*, *abort* ou *resume*).

Mais precisamente, no HTG, vértices são representados por triplas, cada qual formada pela transição da máquina de estado do evento, que pode ser identificada pela ação que dispara a transição (*start*, *stop*, *abort*, *natural end*, *pause* ou *resume*), pelo tipo de evento (apresentação, seleção, atribuição etc.) e pelo identificador da interface que define o evento:[6] uma âncora de conteúdo de um objeto, que normalmente representa o conteúdo de uma mídia, se o evento é do tipo apresentação ou seleção (ou outros eventos associados a conteúdos dos

[6] No HTG cada objeto define um conjunto de interfaces. Uma interface pode representar uma propriedade de um objeto de mídia, como sua posição, sua transparência etc., ou pode definir uma porção das unidades de informação que compõem o conteúdo do objeto, denominada uma âncora de conteúdo.

objetos), ou uma propriedade de um objeto. No caso de identificar uma propriedade, seu valor também é acoplado ao identificador da interface.

Os relacionamentos entre os eventos no HTG são representados através de arestas dirigidas entre as transições das máquinas de estado que representam cada evento em uma aplicação.

O HTG não define apenas o grafo (um conjunto de vértices e arestas dirigidas), mas também formas de caminhamento nesse grafo (condições e prioridades para a verificação das condições associadas às arestas com origem em um mesmo vértice). Todos esses elementos formam uma tupla (V, A, C, N), onde:

- $V = (v_0, v_1, v_2, v_3, ..., v_{n-1}, v_n)$ é um conjunto finito de vértices, onde cada vértice é formado por uma tripla representando uma transição na máquina de estado associada a um evento que atua sobre uma âncora de conteúdo ou sobre a propriedade de um objeto de mídia, alterando o seu valor;
- $A = (a_0, a_1, a_2, a_3, ..., a_{m-1}, a_m)$ é um conjunto finito de arestas que representam, individualmente, um relacionamento entre os elementos que compõem as triplas de dois vértices. Para cada aresta "a", v e w são os vértices de origem e destino, respectivamente ($(v, w) \in V \times V$);
- $C = \{c_{ij}\}$ é um conjunto finito de condições de caminhamento associadas às arestas. Uma condição "$c_{ij}$" é associada com a aresta ($v_i$, $w_j$) $\in$ A e deve ser satisfeita para que a execução do vértice (execução da transição) destino da aresta considerada seja realizada; e
- $N = \{n_{ij}\}$ é um conjunto finito de prioridades associadas às arestas. Uma prioridade "$n_{ij}$" é associada a cada aresta ($v_i$, $w_j$) $\in$ A, e corresponde à ordem na qual essa aresta deve ter sua condição "$c_{ij}$" verificada, comparada com outras arestas partindo do mesmo vértice. Em um conjunto de arestas com a mesma origem $v_i$, as arestas com o menor valor de "$n_{ij}$", para qualquer j, devem ter sua condição verificada antes das demais.

Condições de caminhamento associadas às arestas podem ser simples ou compostas. Uma condição simples pode ser formada por:

- Um tempo que deve ser obedecido a partir da ação que dispara a transição do vértice de origem;

- Uma variável que deve ser avaliada em relação a um valor desejado; e
- Ações externas, como as interações do usuário.

O tempo, definido como condição de uma aresta, deve ser especificado de forma explícita, por exemplo, “10 segundos”, e a sua progressão nunca é interrompida, após ser iniciada quando o vértice de origem da aresta associada a essa condição é alcançado.

Variáveis são utilizadas para representar propriedades existentes na sintaxe de autoria, para controlar a apresentação e também para avaliar propriedades dos eventos, incluindo:

- A duração de um evento: variável que corresponde ao tempo em que um evento sobre uma âncora de conteúdo ou objeto e sua propriedade permanece no estado ocorrendo (*occurring*). Caso o evento esteja no estado pausado (*paused*), o valor dessa variável permanece inalterado, e somente será novamente alterado quando o evento voltar ao estado ocorrendo ou for encerrado (*sleeping*), caso em que essa variável recebe o valor zero;
- As ocorrências de um evento: variável que contém o número de vezes que um evento sobre uma âncora de conteúdo ou objeto e sua propriedade alterna, seguidamente, entre os estados ocorrendo (*occurring*) e encerrado (*sleeping*); e
- O estado atual de um evento: todos os estados possíveis para um evento (*sleeping, occurring, paused*) sobre uma âncora de conteúdo ou objeto e sua propriedade podem ser representados por uma variável, que é verdadeira, se no momento da avaliação o evento encontra-se no estado em questão, ou falsa, caso contrário.

Qualquer condição pode ser negada (NOT). Operadores lógicos (OR, AND), além da negação, também podem ser utilizados para definir condições compostas, relacionando duas ou mais condições (simples ou compostas).

### 3.1.2. Exemplo

Para demonstrar a representação das aplicações através do HTG, foi escolhida uma aplicação onde um “conteúdo audiovisual” é exibido. Em um

instante temporal específico da exibição desse conteúdo, uma informação adicional, que pode ser uma propaganda, por exemplo, também é exibida. Essa informação adicional é formada por um vídeo, se o equipamento onde a aplicação está sendo apresentada for capaz de decodificar e exibir dois vídeos simultaneamente, ou por uma imagem, caso contrário.

A Figura 3.2, apresenta a visão espacial da aplicação mencionada em dois intervalos temporais diferentes. A parte “A” dessa figura corresponde à visão espacial da aplicação no intervalo temporal onde apenas o “conteúdo audiovisual” é exibido. A parte “B” representa a visão espacial dessa aplicação no intervalo temporal em que uma outra informação também é exibida, podendo ser um vídeo (parte superior da Figura 3.2) ou uma imagem (parte inferior da Figura 3.2), dependendo, como mencionado, das características do sistema de apresentação.

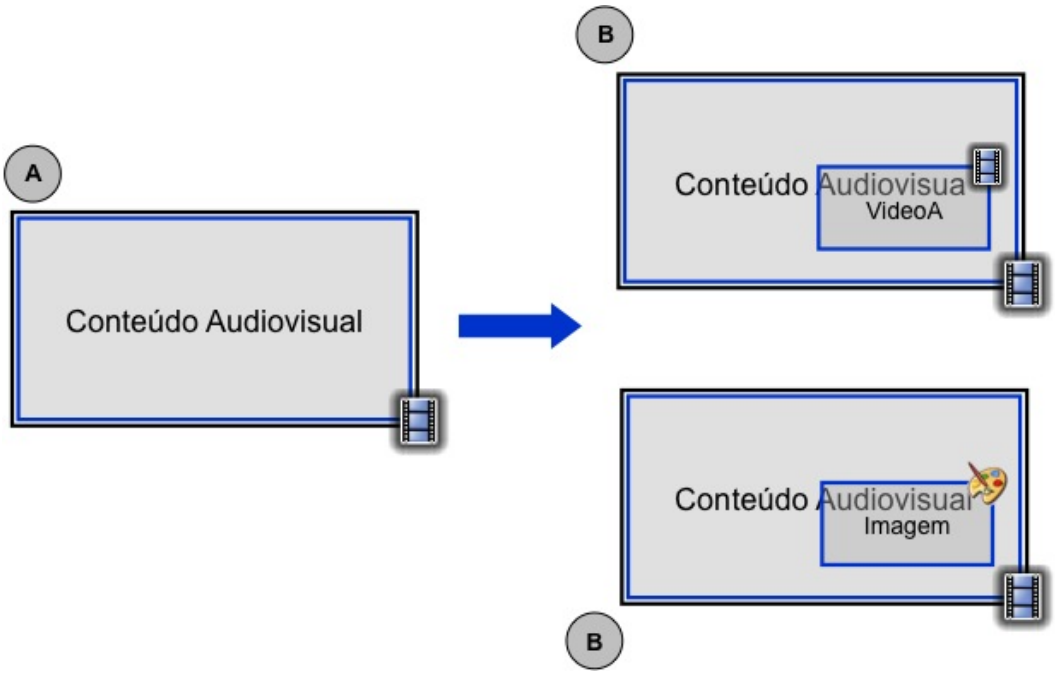


Figura 3.2 - Visões espaciais da aplicação exemplo (primeira parte).

Continuando com o exemplo, atingido o tempo total destinado à exibição da informação adicional (”vídeoA” ou “imagem”), apenas o “conteúdo audiovisual” permanece em exibição, e a visão espacial da apresentação volta a ser a mesma representada na parte “A” da Figura 3.2. Posteriormente, em um outro instante temporal da exibição do “conteúdo audiovisual”, um “ícone”, formado por um objeto de vídeo, é exibido, espacialmente sobre o “conteúdo audiovisual”, com o objetivo de informar ao usuário que existe uma possibilidade de interação. A visão

espacial correspondente ao intervalo temporal em que o “ícone” é exibido é representada na parte “C” da Figura 3.3.

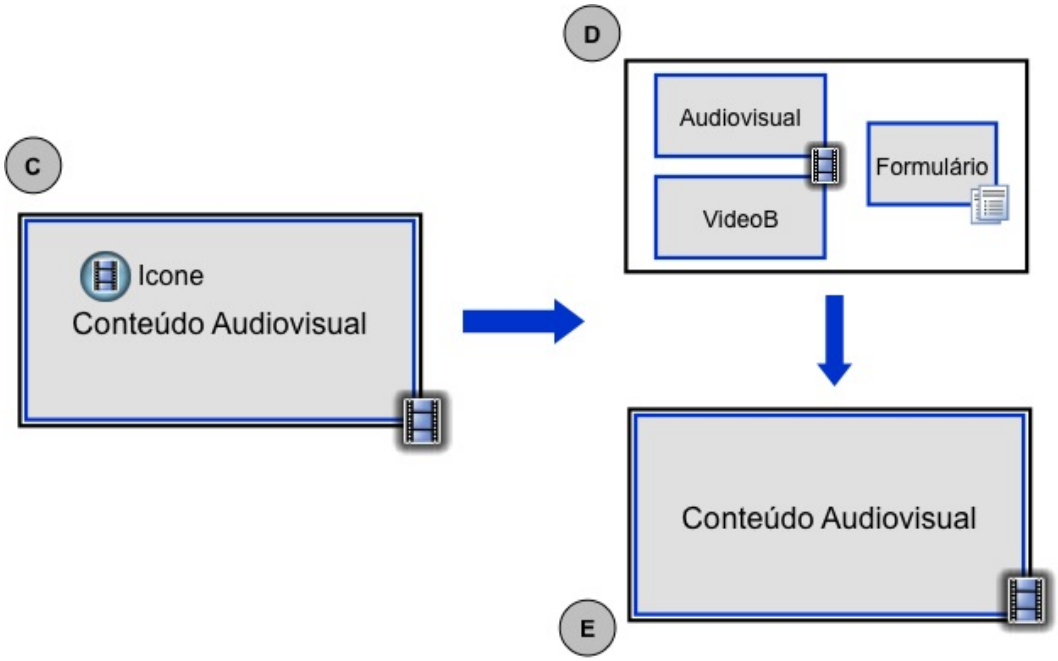


Figura 3.3 - Visões espaciais da aplicação exemplo (segunda parte).

Durante a exibição do objeto “ícone”, se o usuário realizar uma ação interativa, o “conteúdo audiovisual” é redimensionado, outro objeto de vídeo (“vídeoB”) é exibido, contendo outra propaganda, e um “formulário” também é exibido. Os efeitos dessa ação interativa são representados na visão espacial da parte “D” da Figura 3.3. Caso a ação interativa aconteça, quando a exibição do objeto de vídeo ”vídeoB” terminar, isto é, alcançar o seu fim natural, a exibição do “formulário” é encerrada e o “conteúdo audiovisual” é redimensionado para ocupar toda a área disponível para exibição, conforme a visão espacial da aplicação representada na parte “E” da Figura 3.3.

A Figura 3.4 apresenta o HTG correspondente à aplicação espacialmente apresentada nas Figuras 3.2 e 3.3. Para facilitar o entendimento da Figura 3.4, a descrição do evento de apresentação foi suprimida nas triplas (representadas entre parênteses) correspondentes aos vértices. Com o mesmo objetivo, esse evento também foi omitido nas variáveis que representam a duração desse evento, quando essa duração é condição de caminhamento nas arestas.

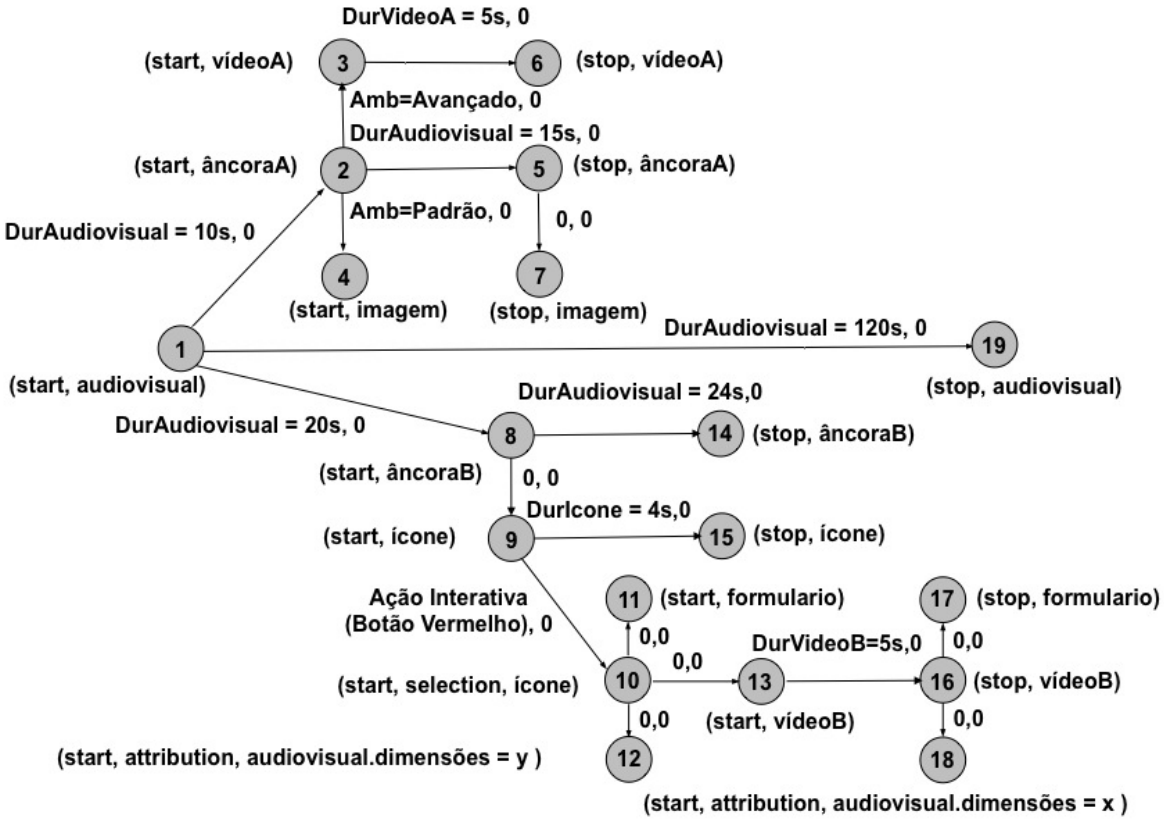


Figura 3.4 - HTG para a aplicação apresentada nas Figuras 3.2 e 3.3.

Neste ponto é importante mencionar alguns aspectos do HTG. Na sua representação gráfica, o primeiro parâmetro associado a uma aresta corresponde à condição para caminhamento no grafo através dessa aresta, e o segundo parâmetro corresponde à prioridade para avaliação da condição de uma aresta em relação às demais com a mesma origem. Na Figura 3.4, todas as arestas, independente da origem, têm a mesma prioridade (definida como zero).

Na Figura 3.4, o vértice 1 corresponde ao ponto de entrada da aplicação[7] e representa a transição de início da apresentação do conteúdo audiovisual ("audiovisual"). A condição associada à aresta entre os vértices 1 e 2 ("DurAudiovisual = 10s") avalia a duração do evento de apresentação do conteúdo audiovisual, tornando-se verdadeira quando essa duração atinge 10 segundos. Quando essa condição for verdadeira, deverá ser executada a transição de início da apresentação da âncora A ("anchorA"). O fim da apresentação dessa âncora (vértice 5) deve ser executado quando a duração da apresentação do

[7] O ponto de entrada é definido como a transição de evento que inicia a apresentação da aplicação. O HTG não exige que uma aplicação defina um único ponto de entrada, no entanto, as cadeias temporais, que serão discutidas na próxima seção, necessitam ser calculadas em relação a um ponto de entrada.

conteúdo audiovisual atinge 15 segundos, conforme descrito na condição associada à aresta entre os vértices 2 e 5 (“DurAudiovisual = 15s”). Assim, a duração total da apresentação dessa âncora temporal deverá ser 5 segundos, caso a apresentação do conteúdo audiovisual não seja interrompida no intervalo que vai de 10 até 15 segundos, a partir do início da sua apresentação.

O vértice que representa o início da âncora A (vértice 2) é a origem de duas arestas que têm como condições as características do sistema de apresentação. Como mencionado, caso o equipamento para apresentação ofereça recursos para que dois vídeos sejam simultaneamente exibidos, a condição associada à aresta entre os vértices 2 e 3 (“Amb = Avançado”) será verdadeira e a transição de início da apresentação do vídeo (“videoA”) deverá ser executada. Por outro lado, se não for possível exibir dois vídeos simultaneamente, essa condição será falsa e a condição associada à aresta entre os vértices 2 e 4 (“Amb = Padrão”) verdadeira. Nesse caso, a transição de início da apresentação da imagem (“imagem”) será executada, em vez da transição de início da apresentação do vídeo (“videoA”).

O resultado do teste das condições associadas às arestas com origem no vértice 2 produz a visão espacial ilustrada na parte “B” da Figura 3.2. A variável “Amb” (“Ambiente”) é utilizada no teste das condições do sistema de apresentação e faz parte da especificação da aplicação (sintaxe de autoria), mapeada no HTG para preservar a semântica definida pelo autor.

No HTG da Figura 3.4, se “videoA” for apresentado, essa apresentação deve ser finalizada quando a duração desse evento atingir 5 segundos, conforme especificado na condição associada à aresta entre os vértices 3 e 6 (“DurVideoA = 5s”). Por outro lado, se a imagem for apresentada, essa apresentação deverá ser encerrada após 5 segundos do seu início, pois o vértice 5, que representa o término da âncora A, provoca a imediata execução da transição de fim da apresentação do objeto imagem, representada pelo vértice 7.

Na Figura 3.4, algumas condições são especificadas como um valor no tempo, por exemplo, “0 segundo” na aresta entre os vértices 5 e 7, indicando, nesse caso, que a transição representada no vértice destino dessa aresta deve ser imediatamente executada. Outras condições, representadas na Figura 3.4, são especificadas através de variáveis como, por exemplo, a duração do evento de apresentação de uma âncora de conteúdo (“DurAudiovisual”, “DurVideoA”, “DurVideoB” e “DurIcon”). Essas durações pode ter seus valores definidos de

forma explícita, por exemplo, através da sintaxe de autoria, ou esses valores podem ser obtidos através da análise do conteúdo da mídia, para objetos como o áudio e o vídeo. Condições também podem ser derivadas de propriedades existentes na sintaxe de autoria, como mencionado, ou definidas como uma ação externa como, por exemplo, na condição associada à aresta entre os vértices 9 e 10, que será descrita logo a seguir.

Continuando com a descrição do grafo da Figura 3.4, com 20 segundos de duração da apresentação do conteúdo audiovisual, uma outra âncora desse conteúdo (“anchorB”) deverá ter sua apresentação iniciada (vértice 8). A transição de início dessa âncora, por sua vez, provoca a execução imediata da transição de início do evento de apresentação do objeto “icone“, representado pelo vértice 9, cuja duração prevista é de 4 segundos. A apresentação desse objeto é representada na visão espacial da parte “C” da Figura 3.3.

O vértice 10 corresponde à transição de início do evento de seleção. Esse vértice é o destino de uma aresta que tem como origem o vértice 9 (início da apresentação de “icone”) e como condição a ação de interatividade do usuário (“Ação Interativa – Botão Vermelho”). Caso essa ação interativa aconteça, a apresentação se desenvolve conforme a visão temporal representada na parte “D” da Figura 3.3.

Durante a apresentação de uma aplicação, partes podem se tornar desnecessárias e, eventualmente, podem ser removidas do grafo, quando os eventos imprevisíveis que ligam essas partes ao restante do grafo não puderem mais ser executados. No exemplo apresentado na Figura 3.4, caso o usuário não realize a ação interativa até o fim da apresentação do objeto de mídia “icone”, representado pelo vértice 14, os eventos especificados no grafo a partir do vértice 10 (início da seleção do usuário) nunca irão ocorrer.

Como consequência do início do evento de seleção (vértice 10), a disposição espacial do conteúdo audiovisual deverá ser alterada, através da imediata execução da transição representada pelo vértice 12, que inicia a atribuição da propriedade “dimensões” do objeto “audiovisual”. Também deverão ser imediatamente iniciadas as apresentações do vídeo da propaganda “videoB” (vértice 11) e do objeto “formulário” (vértice 13).

O fim da apresentação do objeto “videoB” (vértice 15) é atingido quando a duração da sua apresentação atingir 5 segundos (“DurVideoB = 5s”). A execução

desse vértice, por sua vez, provoca o término imediato da apresentação do "formulário" (vértice 16) e também o retorno da disposição espacial inicial do conteúdo audiovisual, através do início da execução do evento de atribuição sobre a sua propriedade "dimensões", agora com a atribuição do seu valor original (vértice 17).

### 3.1.3. Construção

O próximo capítulo descreve a construção do HTG para uma sintaxe específica: a definida pela linguagem NCL. Algumas regras, no entanto, devem ser respeitadas no processo de construção, independente da sintaxe de autoria.

Representar no grafo apenas as transições de eventos que provocam alterações na apresentação é a primeira dessas regras, que tem como objetivo preservar a simplicidade do grafo. Na Figura 3.4, os vértices 12 e 18 representam o início do evento de atribuição da propriedade "dimensões" do conteúdo audiovisual, com diferentes valores. A transição de fim desses eventos poderia ser representada no grafo por um outro vértice, destino dos vértices 12 e 18. No entanto, o fim desse evento de atribuição não altera a apresentação, não modifica o valor de nenhuma variável e também não é condição para a execução de nenhum outro evento e, portanto, não deve ser representado.

Outras regras também devem ser consideradas na contrução do HTG, independente da sintaxe utilizada para a autoria. A Tabela 3.1 resume essas regras, adicionais à primeira, já mencionada e listada como Regra 1.

| Regra | Descrição |
|---|---|
| 1 | Um vértice V do tipo (*transition*, *event*, *anchorId*) para eventos de apresentação ou seleção, qualquer que seja a transição (*transition*) e a âncora de conteúdo (*anchorId*), ou um vértice T do tipo (*transition*, *attribution*, *objectId.propertyId*) para eventos de atribuição, somente deve existir se V ou T for utilizado no controle da apresentação. |

| | |
|---|---|
| 2 | Um vértice V do tipo (*stop*, *event*, *anchorId*) no HTG pode ser equivalente a um vértice W do tipo (*natural end*, *event*, *anchorId*), para qualquer evento (*event*) e âncora de conteúdo (*anchorId*), caso a sintaxe de autoria não defina formas diferentes para o término de um evento (*natural end* e *stop*). Nesse caso, se necessários na apresentação (regra 1), um único vértice deve ser utilizado, contendo a transição *stop*. |
| 3 | Um vértice V do tipo (*abort*, *event*, *anchorId*) ou um vértice T do tipo (*abort*, *event*, *objectId.propertyId*) não pode ser vértice de origem de nenhum outro vértice, para qualquer evento (*event*) e âncora de conteúdo (*anchorId*) ou objeto (*objectId*) e sua propriedade (*propertyId*). |
| 4 | Um vértice T do tipo (*start*, *attribution*, *objectId.propertyId*) deve possuir, na própria tripla, o valor a ser atribuído à propriedade (*propertyId*). Se existirem diferentes valores a serem atribuídos a uma mesma propriedade, definida sobre um mesmo objeto (*objectId*), deve ser criado um vértice $T_n$ do tipo (*start*, *attribution*, *objectId.propertyId = value*) para cada valor (*value*) diferente. |
| 5 | Um vértice T do tipo (*natural end*, *attribution*, *objectId.propertyId*), se necessário no grafo (regra 1), deve ser o destino de todo vértice S do tipo (*start*, *attribution*, *objectId.propertyId = value*), para um mesmo *objectId e propertyId*, independente do valor a ser atribuído (*value*). |
| 6 | Um vértice T do tipo (*start*, *selection*, *anchorId*), se necessário no grafo (regra 1), deve ser o destino de um vértice origem S do tipo (*start*, *presentation*, *anchorId*), quando *anchorId* representa a mesma âncora de conteúdo para S e T. |
| 7 | Um vértice T do tipo (*natural end*, *presentation*, *anchorId*) deve ser o destino de um vértice origem S do tipo (*start*, *presentation*, *anchorId*), quando *anchorId* representa a mesma âncora de conteúdo para S e T, exceto quando a duração desse evento não tem relação com a sua transição de início. |

| | |
|---|---|
| 8 | Um vértice T do tipo (*start*, *presentation*, *anchorIdT*) deve ser o destino de um vértice origem S do tipo (*start*, *presentation*, *anchorIdS*), se *anchorIdT* é uma âncora equivalente a uma parte do conteúdo de *anchorIdS*. |
| 9 | Arestas do tipo ((*start*, *presentation*, *anchorId*), (*natural end*, *presentation*, *anchorId*)) para uma mesma *anchorId*, devem ter como condição a duração do evento de apresentação dessa âncora (*anchorId*) ou do objeto, caso *anchorId* seja uma parte do conteúdo desse objeto. |
| 10 | Arestas do tipo ((*start*, *presentation*, *anchorIdS*), (*start*, *presentation*, *anchorIdT*)) devem ter como condição parte da duração do evento de apresentação sobre *anchorIdS*, se *anchorIdT* é uma âncora equivalente a uma parte do conteúdo de *anchorIdS*. |

Tabela 3.1 - Regras para a construção do HTG.



Transições "*natural end*" e "*stop*" conduzem um evento qualquer ao mesmo estado "*sleeping*", como pode ser observado na máquina de estado da Figura 3.1. Essas transições, no entanto, possuem diferentes significados: a transição "*natural end*" representa que a duração prevista para um evento foi alcançada, enquanto a transição "*stop*" representa que a duração de um evento foi interrompida em razão da ocorrência de uma outra transição de evento. Essa diferença, no entanto, muitas vezes não é utilizada na sintaxe de autoria e, portanto, a segunda regra, define que, nesse caso, um único vértice contendo a transição "*stop*" deve ser construído, para um mesmo evento e âncora de conteúdo.

A transição "*abort*", embora também conduza um evento qualquer ao estado "*sleeping*", quando necessária em uma apresentação, deve ser representada por um vértice próprio, pois essa transição é utilizada para encerrar um evento sem que essa modificação altere a ocorrência de outros eventos. A terceira regra deixa claro que a transição "*abort*" não pode estar relacionada como causa da execução de nenhuma outra transição de evento, uma vez que o vértice que representa essa transição não pode ser o vértice de origem de nenhuma aresta. Caso a sintaxe de autoria não represente a semântica da transição "*abort*", não existirá nenhum vértice com essa transição nos grafos obtidos a partir de aplicações especificadas nessa sintaxe.

A quarta regra define que o valor a ser atribuído também faz parte dos vértices que representam o início de um evento de atribuição. Assim, se existirem diferentes valores para uma mesma atribuição (mesmo objeto e propriedade), vértices diferentes devem ser construídos, um para cada valor a ser atribuído.

Na Figura 3.4, por exemplo, foram especificados dois vértices (12 e 18) para a atribuição a um mesmo objeto de mídia e propriedade, mas com dois valores diferentes. Caso o valor a ser atribuído não fizesse parte do vértice, um número menor de vértices poderia ser utilizado para representar uma aplicação. No entanto, cada aresta precisaria ter um valor associado, que seria utilizado apenas quando o seu vértice destino representasse o início de um evento de atribuição.

A quinta regra especifica que, se for necessário representar a transição de fim do evento de atribuição para um objeto de mídia e sua propriedade, esse vértice deve ser o destino de todos os vértices relativos ao início da atribuição desse mesmo objeto e propriedade, para qualquer valor a ser atribuído.

Na Figura 3.4, por exemplo, caso o fim do evento de atribuição da propriedade “dimensões” fosse necessário à apresentação, esse novo vértice deveria ser o destino de arestas com origem nos vértices 12 e 18. Esse novo vértice seria necessário, caso o fim desse evento de atribuição estivesse relacionado como causa da execução de uma outra transição de evento, ou caso a atribuição em questão tivesse uma duração, que deveria estar especificada na aresta entre as transições de início e fim desse evento.

O vértice que representa o início de um evento de seleção, caso esse evento seja necessário em uma aplicação, deve ser o destino de uma aresta com origem no vértice de início do evento de apresentação da âncora de conteúdo sobre a qual essa seleção pode ocorrer. Essa deve ser a única origem do vértice que representa esse evento de seleção e essa restrição é especificada na regra seis. Na Figura 3.4, por exemplo, o vértice 10 (início da seleção do objeto “icone”) é o destino de uma aresta com origem no vértice que representa o início da apresentação do objeto “icone” (vértice 9) e essa é a única aresta que tem como destino o vértice 10.

As regras sete e oito definem a representação da apresentação dos objetos e de suas âncoras temporais. O evento de apresentação de cada objeto em uma apresentação, quando a duração desse evento é definida, deve ser representado por um par de vértices unidos através de uma aresta, onde o vértice de origem deve representar o início desse evento e o de destino deve representar o seu fim.

Apresentações de âncoras relativas a uma porção temporal de um objeto de mídia também devem ser representadas dessa forma, quando a sua duração é definida. Na Figura 3.4, os objetos, “videoA”, “icon” e “videoB” têm, cada um, o início e o fim das suas apresentações relacionadas através de arestas. O mesmo acontece com a âncora A (“anchorA”) do objeto “audiovisual”.

Em relação às âncoras que representam uma parte do conteúdo de um objeto, o vértice que representa o início da apresentação dessa âncora deve ser o destino de uma aresta com origem no vértice correspondente ao início da apresentação do objeto correspondente ao todo da porção relativa à âncora. Portanto, na Figura 3.4, o vértice que representa o início da apresentação do objeto “audiovisual” (vértice 1) é o vértice origem de arestas com destino nos vértices que representam o início da apresentação das âncoras correspondentes a cada porção temporal desse objeto (vértices 2 e 8). Cada uma dessas arestas tem como condição a variável correspondente à duração do evento de apresentação do objeto “audiovisual”, conforme especificado nas regras oito e nove da Tabela 3.1.

Os vértices que representam as transições de início (vértice 4) e fim (vértice 7) da apresentação do objeto “image” não são unidos através de uma aresta, pois não existe um relacionamento entre essas duas transições (a apresentação de uma imagem pode ter uma duração infinita em relação ao início desse evento).

Por fim, é importante mencionar que todos os vértices podem ser origem e destino de arestas construídas para representar relacionamentos especificados na aplicação. A exceção são os vértices que representam transições do tipo “*abort*”, conforme mencionado na regra quatro, e os vértices que representam o início do evento de seleção, conforme mencionado na regra cinco.

## 3.2. Cadeias Temporais

Para facilitar o cálculo dos instantes temporais para a ocorrência das transições dos eventos nas aplicações hipermídia, um HTG pode ser dividido em vários outros grafos. Antes de continuar a descrever como outros grafos podem ser construídos com base no HTG de uma aplicação, é importante diferenciar alguns conceitos que serão utilizados nesta seção:

1. Uma transição de evento é previsível a partir de uma transição de evento $T_{ev}$, se é possível determinar com certeza a sua ocorrência, a partir da ocorrência de $T_{ev}$;
2. Uma transição de evento é imprevisível a partir de uma transição de evento $T_{ev}$, se não é possível determinar com certeza a sua ocorrência, a partir da ocorrência de $T_{ev}$; e
3. Uma transição de evento, previsível ou imprevisível, pode ser:
    a. Determinística, se é possível determinar o instante no tempo exato de sua ocorrência. Supondo que a transição aconteça, no caso dela ser imprevisível.
    b. Não-determinística, se não é possível determinar o instante no tempo exato de sua ocorrência. Supondo que a transição aconteça, no caso dela ser imprevisível.

O uso de vários grafos na representação de aplicações hipermídia é uma estratégia utilizada em alguns sistemas, como no Firefly (Buchanan, 1992; Buchanan, 1993; Buchanan, 2005), precursor na idéia de uso de cadeias temporais. Nesse sistema, cada um dos seus grafos é formado por vértices representando sequências de eventos previsíveis e determinísticos, a partir de um evento imprevisível. Como mencionado no Capítulo 2, quando o evento imprevisível que inicializa uma dessas sequências (um grafo) acontece, é possível determinar o instante no tempo exato para a ocorrência dos demais eventos da sequência.

Diferente do Firefly, ao invés de eventos previsíveis e determinísticos, os grafos gerados a partir do HTG, embora também chamados de cadeias temporais,[8] são formados por vértices que representam sequências de transições de eventos previsíveis, que podem ser tanto determinísticas quanto não-determinísticas, a partir de um evento imprevisível. É permitido também nesses grafos que os vértices sorvedouros sejam eventos imprevisíveis. Essas sequências de transições de eventos, chamadas, resumidamente a partir deste ponto, de cadeias HTG, são propostas com o objetivo de facilitar tanto a construção dos grafos como também

[8] O termo cadeia temporal foi mantido para manter compatibilidade com seu entendimento usual na literatura de sistema hipermídia. No entanto, deve-se notar que a cadeia forma de fato um sub-grafo do HTG.

o controle da apresentação, como será descrito ao longo desta seção. A Tabela 3.2 resume as regras para a especificação das cadeias HTG.

| Regras | Descrição |
|---|---|
| 1 | Uma cadeia temporal é formada por vértices que representam uma sequência de transições de eventos previsíveis (determinísticos e não-determinísticos), a partir de uma transição de evento imprevisível, incluindo entre eles o vértice que representa a transição de evento correspondente ao ponto de entrada da aplicação. |
| 2 | Também são incluídos na cadeia temporal transições de eventos imprevisíveis alcançáveis por arestas a partir de um vértice incluído pela Regra 1. Esses vértices são sempre vértices sorvedouros da cadeia. |
| 3 | Vértices que representam transições de eventos imprevisíveis que também estão presentes nas cadeias secundárias devem ser nascedouros nessas cadeias secundárias. Vértices de transições de eventos previsíveis não-determinísticos que aparecem tanto na cadeia principal quanto em cadeias secundárias devem ser vértices sorvedouros nessas cadeias secundárias (vértices de transições de eventos previsíveis determinísticos só podem aparecer em uma única cadeia). |

Tabela 3.2 - Regras para a especificação de cadeias HTG.

Em relação à construção dos grafos, no Firefly, as sequências de eventos fazem parte da especificação, não sendo necessário obtê-las através de um caminhamento em um grafo global. Um HTG, por outro lado, precisa ser percorrido para que suas cadeias possam ser construídas. Nessa construção, para que se possa ter certeza de que uma sequência é determinística, pode ser necessário realizar um caminhamento através de vários percursos em um HTG, como será discutido ao longo desta seção (Figura 3.7). Além disso, seria necessário conhecer, *a priori*, todas as durações associadas às apresentações dos conteúdos das mídias.

Quando todas as arestas de um HTG têm apenas o tempo como condição associada, o seu caminhamento pode definir uma única cadeia. Nesse caso,

formada por uma sequência de vértices que representam transições de eventos previsíveis e determinísticos. Por outro lado, a partir de um HTG contendo arestas onde as condições são formadas por ações externas, como, por exemplo, a interação do usuário, ou por variáveis que não estão associadas ao tempo, como, por exemplo, aquelas utilizadas para avaliar as características do sistema de apresentação ou o perfil do usuário, várias cadeias temporais poderão ser construídas.

Independente da quantidade de cadeias temporais HTG, todas são obtidas através do caminhamento no HTG. Esse caminhamento, quando iniciado no vértice correspondente ao ponto de entrada da aplicação, produz uma cadeia denominada principal. O caminhamento para a construção de uma cadeia é interrompido quando uma aresta, que contém como condição uma ação externa ou uma variável não associada ao tempo, é encontrada, ou quando o vértice alcançado representa uma transição de evento imprevisível para o estado do seu evento.[9] Nesse caso, a partir do vértice destino dessa aresta, um novo caminhamento pode ser iniciado, para que uma nova cadeia temporal seja construída. Esse caminhamento somente não será iniciado caso a cadeia temporal a partir do vértice considerado já existir, isto é, caso essa cadeia já tenha sido previamente construída. Todas as demais cadeias temporais, além da principal, são denominadas secundárias.

A Figura 3.5 apresenta a cadeia temporal principal obtida através do caminhamento no HTG apresentado na Figura 3.4, partindo do vértice 1, definido como ponto de entrada. Assim como na Figura 3.4, a descrição do evento de apresentação foi suprimida nos vértices e também nas variáveis que representam a duração desse evento.

[9]A ocorrência de qualquer transição deve obedecer as especificações da máquina de estado da Figura 3.1. Durante um caminhamento também deve ser detectada a presença de ciclos (*loops*) que ocorrem quando um vértice em um caminhamento é novamente alcançado. O caminhamento também deve ser interrompido em relação ao vértice que indicou a presença desse ciclo.

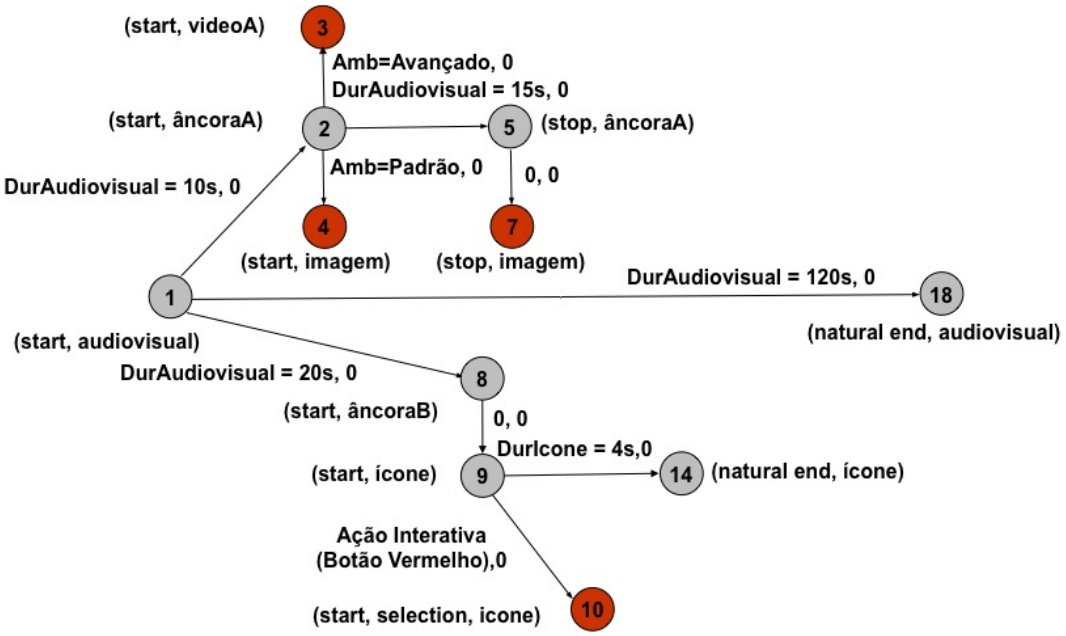


Figura 3.5 - Cadeia temporal principal do HTG apresentado na Figura 3.4.

Na Figura 3.5, os vértices sorvedouros, que representam transições de eventos imprevisíveis, estão destacados em vermelho. A cadeia principal representada nessa figura contém três transições de eventos imprevisíveis, representadas pelos vértices 3, 4, 7 e 10. A execução das transições dos eventos representadas pelos vértices 3 e 4 dependem das características do equipamento onde a aplicação é apresentada. A execução da transição representada pelo vértice 7 depende da apresentação da “imagem” estar ocorrendo (transição representada no vértice 4 ter sido executada). A execução da transição de início do evento de seleção, representada no vértice 10, depende da ocorrência de uma ação interativa do usuário, nesse caso, pressionar um botão vermelho. Dessa forma, os vértices 3, 4, 7 e 10 da cadeia temporal principal apresentada na Figura 3.5, são origem, cada um, de uma outra cadeia temporal (secundária), apresentadas na Figura 3.6.

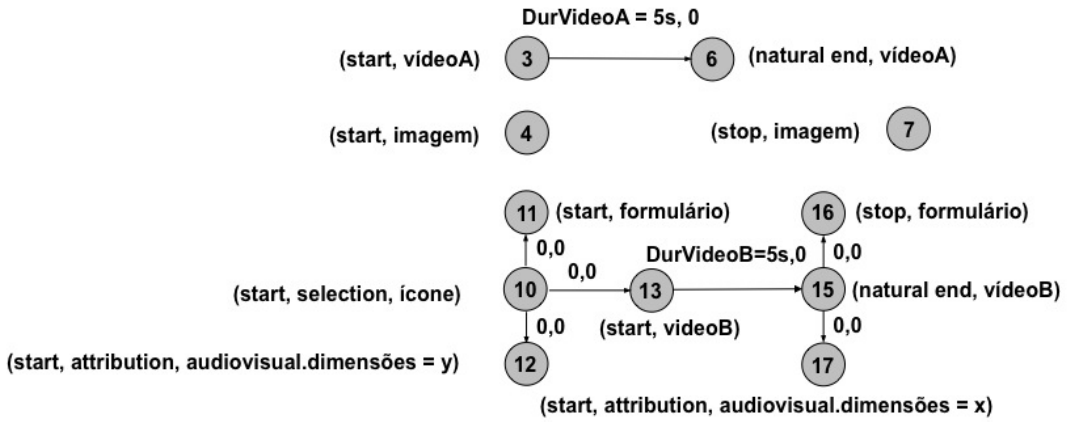


Figura 3.6 - Cadeias temporais secundárias do HTG da Figura 3.4.

Na Figura 3.6, os vértices 3 e 6 representam, respectivamente, o início e o fim da apresentação de “videoA”. Como anteriormente mencionado, as transições que os vértices dessa cadeia representam serão executadas caso o sistema de apresentação tenha a capacidade de exibir dois vídeos simultaneamente. Por outro lado, o vértice 4 representa o início da apresentação de “imagem”, que será executada caso o sistema de apresentação somente posssa exibir um vídeo por vez. O vértice 7, indiretamente, também depende dessa mesma condição. Caso a apresentação de “imagem” não tenha sido iniciada, a transição de fim desse evento de apresentação também não irá acontecer, como definido na máquina de estado de eventos, apresentada na Figura 3.1. Nessa mesma figura, os vértices 10, 11, 12, 13, 15, 16 e 17 representam as transições dos eventos que devem ser executadas caso a ação interativa seja realizada (aresta entre os vértices 9 e 10 da Figura 3.5). As transições dos eventos existentes nessa cadeia alteram as dimensões do conteúdo audiovisual (“audiovisual”) e controlam a apresentação do “formulário” e do “videoB”.

Neste ponto pode-se justificar o uso de cadeias e não diretamente o HTG. Note que nas cadeias temporais, mesmos os eventos previsíveis não determinísticos ganham um tempo determinístico de ocorrência (tempo estimado de ocorrência). Esse tempo é aquele que será verdadeiro, caso a cadeia seja percorrida sem interrupção até o fim. No próximo exemplo, um caso de ocorrência de interrupção será tratado. Essa é uma grande diferença das cadeias temporais do Firefly, onde os eventos previsíveis não determinísticos não têm seus tempos seguer estimados.

Construir apenas sequências de transições de eventos determinísticas, como no Firefly, pode tornar o controle da apresentação totalmente dependente da ocorrência de eventos e com pouca utilidade na prevenção do descasamento do sincronismo de uma apresentação e no controle da distribuição dos conteúdos de mídia (ou seja, do processo de gerência de transporte). A Figura 3.7 exemplifica essa situação. Na figura, “N vídeos” podem ser sequencialmente apresentados, porém, durante a apresentação de cada vídeo, uma ação interativa pode interromper a apresentação do vídeo atual e iniciar a apresentação do próximo vídeo. Nesse exemplo, assim como acontece nas aplicações NCL, que serão abordadas no próximo capítulo, não existem diferenças entre as transições que correspondem ao fim de um evento (Regra 2 da Tabela 3.1).

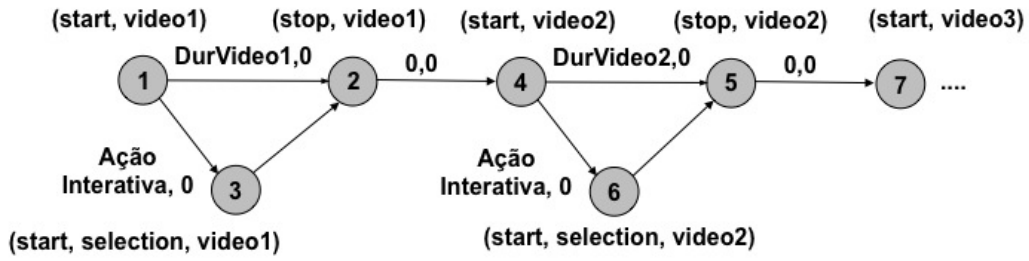


Figura 3.7 - HTG com vários eventos imprevisíveis.

No controle pelas cadeias temporais Firefly, a apresentação da Figura 3.7 seria iniciada pela transição de início da apresentação do objeto “video1”, único vértice da sua cadeia principal. Apenas utilizando essa cadeia, não haveria qualquer informação sobre possíveis ocorrências, e, portanto, possíveis ajustes e envio de dados, caso não ocorresse a interação do usuário. Após a execução da transição de início do “vídeo1”, a apresentação da aplicação permaneceria aguardando pelo fim da apresentação desse vídeo ou por uma ação interativa. Só após a ocorrência de um desses dois eventos, a cadeia secundária iniciada por um deles seria acoplada à cadeia principal e o controle da apresentação continuaria. Esse comportamento se repetiria ao longo da apresentação, em um verdadeiro paradigma orientado a (transições de) eventos, com ações sendo tomadas apenas quando das ocorrências desses eventos.

Através do HTG da Figura 3.7, a cadeia temporal principal proposta nesta tese, ao contrário da cadeia Firefly anterior, apresenta importantes informações adicionais, como apresentado na Figura 3.8.

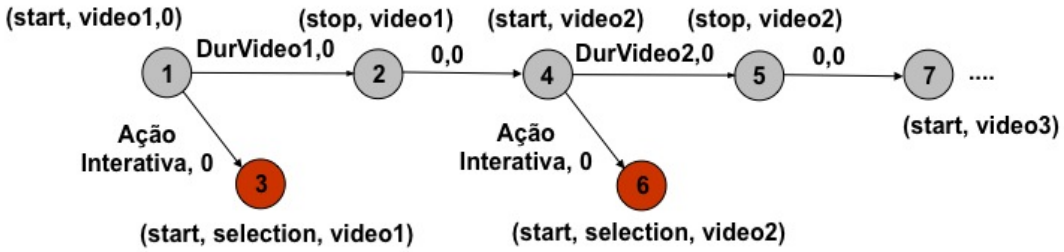


Figura 3.8 - Cadeia temporal principal do HTG da Figura 3.7.

A cadeia da Figura 3.8 oferece ao escalonador da apresentação uma previsão do comportamento da aplicação. Caso nenhuma ação interativa aconteça, o comportamento da apresentação será realizado conforme descrito na cadeia principal. Por outro lado, caso alguma das ações interativas possíveis ocorra, os eventos de uma cadeia secundária, com origem na transição de evento imprevisível (vértices 3 e 6) deverão ser adicionados à cadeia principal, que

poderá ter os tempos dos eventos previsíveis não-determinísticos devidamente corrigidos a partir de então. A Figura 3.9 apresenta duas das N cadeias secundárias da aplicação representada pelo HTG da Figura 3.7. Durante o caminhamento para construção de uma cadeia temporal, podem ser encontrados vértices já definidos em uma outra cadeia. Esse é o caso, por exemplo, dos vértices 2 e 5 da Figura 3.9. Como esses vértices já foram definidos em outra cadeia, nesse caso, na cadeia principal, eles são apenas referenciados nas cadeias secundárias.

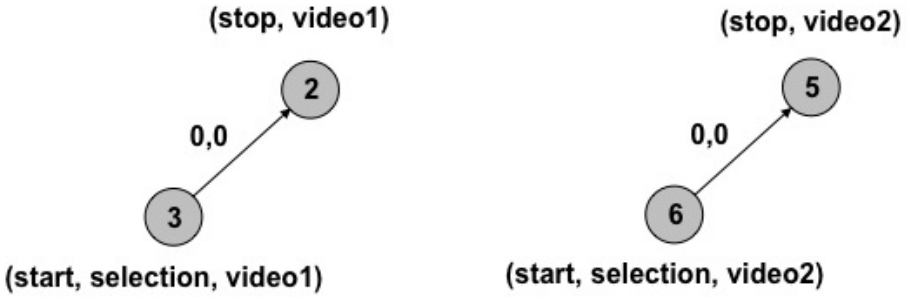


Figura 3.9 - Duas cadeias temporais secundárias do HTG da Figura 3.7.

Ainda na Figura 3.9, a primeira cadeia secundária é iniciada pelo vértice correspondente ao início do evento de seleção do objeto “video1” (vértice 3). Essa transição, quando executada como consequência de uma ação interativa representada na cadeia principal, provoca o fim da apresentação do objeto “video1” (vértice 2). Com consequência dessa ocorrência, o tempo previsto para o início da apresentação do objeto “video2” (vértice 4) será modificado. A Figura 3.10 mostra a cadeia resultante da junção dessa cadeia secundária com a cadeia principal, no momento “x segundos” que ocorre a interação do usuário.

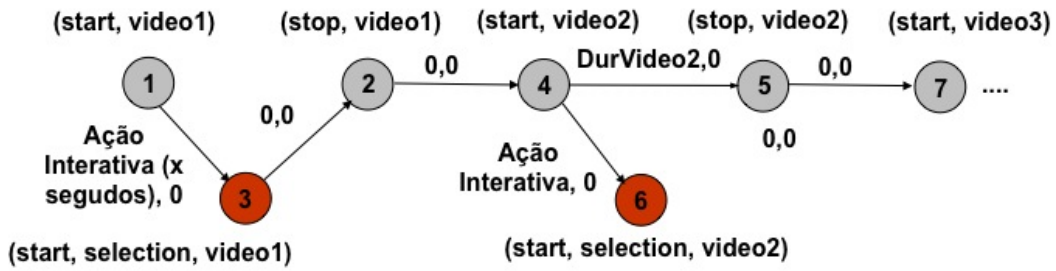


Figura 3.10 - Cadeia HTG resultante da junção da cadeia principal com a primeira cadeia secundária.

A outra cadeia secundária da Figura 3.9 possui vértices que representam as mesmas transições de eventos, porém, para o objeto “video” (vértices 6 e 5). As

demais cadeias secundárias desse exemplo serão construídas de forma similar, para os demais vídeos da aplicação.

Cadeias temporais podem ser interpretadas de diferentes formas. Cada uma dessas interpretações define um plano temporal, alguns dos quais serão propostos no Capítulo 5, com o objetivo de controlar a qualidade das apresentações e otimizar a utilização de recursos.

Em relação à apresentação, durante a sua ocorrência, cadeias vão sendo adicionadas ou descartadas, dependendo da ocorrência das transições de eventos imprevisíveis, conforme exemplificado na Figura 3.10. Ao final de uma apresentação, um outro HTG, na maioria das vezes diferente do HTG original concebido na autoria (um HTG original filtrado), será construído. Esse novo HTG representa o histórico da apresentação que pode ser salvo a qualquer momento, permitindo que a mesma apresentação seja novamente realizada, com todas as adaptações e opções escolhidas pelo usuário.

Como exemplo, considerando a aplicação da Figura 3.4, supondo que nesse exemplo, o sistema de apresentação não seja capaz de apresentar dois vídeos simultaneamente e que o usuário não realize a ação interativa permitida, no final da apresentação o seu histórico é representado pelo HTG ilustrado na Figura 3.11.

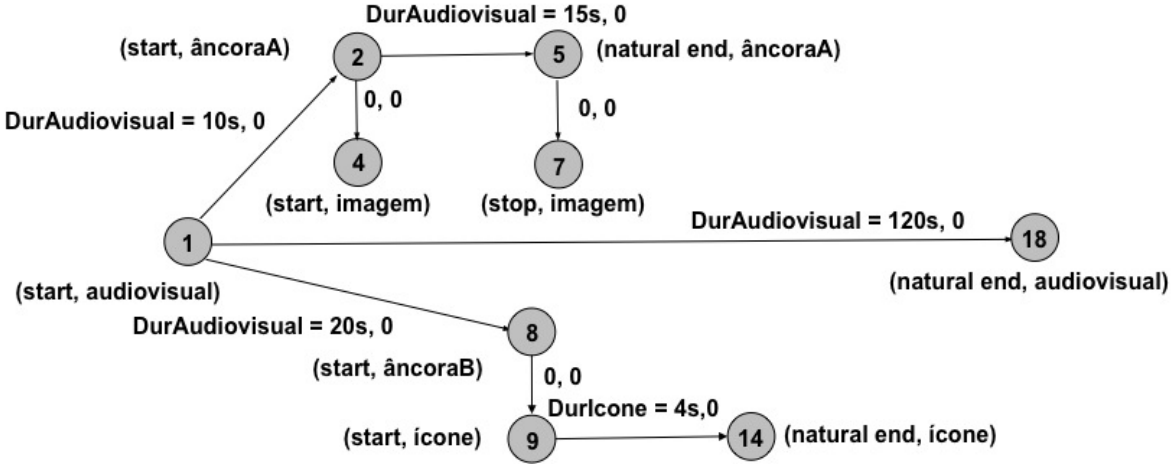


Figura 3.11 - Um possível HTG final para a apresentação da Figura 3.4.

# 4
# Representação de Aplicações Hipermídia Declarativas – Aplicações NCL

Este capítulo trata da tradução das especificações do autor para o HTG, a partir de uma sintaxe de autoria específica, a sintaxe da linguagem NCL. O controle do sincronismo temporal das aplicações NCL é necessário, tanto para garantir a qualidade das apresentações para TV quanto para uma melhor utilização dos recursos existentes nos sistemas de TV digital interativa.

Voltada para suporte à autoria declarativa, NCL oferece várias abstrações com o objetivo de facilitar essa tarefa (Neto, 2009; Muchaluat-Saade, 2002; Soares, 2010b). Essas facilidades, contudo, podem tornar difícil o controle da apresentação, uma vez que os instantes temporais para a ocorrência dos eventos existentes nas aplicações precisam ser calculados a partir de estruturas com alto nível de abstração. Através do uso do HTG, as especificações temporais das aplicações NCL podem ser simplificadas, sob o ponto de vista do controle da apresentação, como será discutido ao longo deste capítulo.

## 4.1.
## Módulos, Perfis e Aplicações NCL

NCL 3.0 segue uma abordagem modular, onde cada módulo agrupa elementos e atributos XML (W3C, 2008a) semanticamente relacionados. Esses módulos, por sua vez, podem ser agrupados em diferentes perfis de linguagem, construídos para atender a uma demanda específica. Este capítulo explora aplicações do perfil avançado para TV (perfil EDTV) (Soares, 2006a), padronizado para a especificação de aplicações declarativas no ISDB-$T_B$ e para IPTV segundo o ITU-T.[10]

O perfil avançado contém todas as funcionalidades da NCL para a especificação de aplicações nos STVDI, que podem ser, em grande parte,

[10]A definição completa dos módulos de NCL 3.0 e dos seus principais perfis é realizada utilizando XML Schema (XMLSchema, 2004a; XMLSchema, 2004b; XMLSchema, 2004c). Essas definições estão disponíveis no site web da NCL (www.ncl.org.br/NCL3.0).

descritas através de uma aplicação proposta como exemplo. Nessa aplicação, um filme, uma animação sobre um jogador de futebol, é recebida por difusão nos receptores de TV. Também por difusão, diversos outros objetos de mídias são recebidos e devem ser exibidos, sincronamente, de acordo com a especificação da aplicação em NCL que é, na verdade, mais um dos conteúdos recebidos por difusão.

Durante toda a apresentação da animação (filme), uma música de fundo é executada. Após 12 segundos do início dessa animação, uma foto real do jogador (imagem), ator principal retratado na animação, é apresentada na tela. Dependendo do tamanho da tela, informação que em NCL pode ser obtida através de uma variável, a duração da apresentação dessa imagem irá variar. Serão 20 segundos de apresentação se essa tela tiver um tamanho mínimo e 10 segundos caso contrário, evitando que detalhes importantes da animação permaneçam encobertos.

Em outro momento da apresentação da animação, uma publicidade sobre equipamentos esportivos, formada pela imagem de uma chuteira, pode ser apresentada, mas apenas se ações interativas forem permitidas. Novamente, essa possibilidade é definida a partir de uma outra variável da aplicação.

Caso a imagem da chuteira seja apresentada, esse evento será iniciado aos 45 segundos do início da animação e será encerrado quando essa duração atingir 51 segundos. Essa apresentação indica aos usuários a possibilidade da ocorrência de outros eventos, disparados em função da ação interativa. Para dar maior destaque a essa possibilidade, a posição da imagem da chuteira irá variar na tela durante o seu período de exibição.

Nessa aplicação, o usuário pode selecionar a imagem da chuteira pressionando o botão vermelho do controle remoto (mesma cor da chuteira) e, como resultado, sua apresentação será encerrada, o vídeo da animação redimensionado (através da manipulação de outras variáveis da aplicação), uma propaganda e um formulário serão exibidos na tela, todos sobre uma imagem de fundo. Para ilustrar esse comportamento, a Figura 4.1 mostra a aplicação em dois momentos: primeiro, quando a imagem da chuteira é apresentada e, depois, caso essa imagem seja selecionada.

Figura 4.1 - Apresentação de uma aplicação NCL com interatividade.

A propaganda, apresentada como resultado do evento de seleção da imagem da chuteira, é formada por um vídeo e por um formulário. Esse formulário pode ser uma aplicação XHTML em português ou outra aplicação em inglês, dependendo, novamente, do valor de outra variável que contém o idioma do usuário. O final da apresentação do vídeo da propaganda irá provocar o término da apresentação do formulário e da imagem de fundo, além de redimencionar a animação ao seu tamanho original.

A aplicação descrita nos parágrafos anteriores contém várias especificações associadas ao controle do sincronismo e à realização de adaptações. A Tabela 4.1 resume essas especificações, com pelo menos um exemplo da aplicação.

| Características | Exemplo (na aplicação NCL) |
|---|---|
| Sincronismo no tempo (sem interação) | Apresentação de uma imagem (foto do jogador) é iniciada, quando atingido os 12 segundos do início da apresentação do vídeo da animação. |
| Adaptação da apresentação | Apresentação de uma imagem (foto do jogador) pode durar 10 ou 20 segundos, dependendo das características do ambiente (tamanho da tela). |

| | |
|---|---|
| Sincronismo no tempo (com condição) | Apresentação de uma imagem (chuteira) é iniciada, quando a animação atinge 45 segundos a partir do seu início; e finalizada no máximo aos 51 segundos da duração da animação. Essa imagem somente será apresentada caso a interatividade seja permitida (outra característica do ambiente de apresentação). |
| Animação | Durante a apresentação de uma imagem (chuteira) sua posição na tela irá variar. |
| Sincronismo no tempo (com interação) | Seleção de uma imagem (chuteira) encerra a apresentação desse objeto e inicia outros eventos. |
| Adaptação do conteúdo | Uma aplicação (formulário) será apresentada, após escolha entre duas alternativas. Essa escolha depende das características do ambiente de apresentação (idioma do usuário). |

Tabela 4.1 - Principais características relacionadas ao sincronismo e à adaptação encontradas na aplicação NCL exemplo.

Além das funcionalidades da linguagem NCL listadas na Tabela 4.1, outras funcionalidades, como a estruturação da aplicação pelo uso de contextos e a possibilidade de reúso das suas entidades (ABNT, 2009), devem ser consideradas na construção de um HTG. Todas essas características da linguagem serão discutidas nas próximas seções. A especificação completa da aplicação descrita nesta seção é apresentada no final deste capítulo, na Seção 4.6.

Nas próximas seções, a construção do HTG do exemplo citado é realizada paulatinamente.

## 4.2. Objetos de Mídia e Interfaces

Na NCL, os tipos básicos de objetos de mídia são definidos no módulo *Media*. Para a especificação dos objetos, esse módulo define o elemento <*media*>, contendo um conjunto de atributos para, entre outras informações (ABNT, 2009), identificar cada objeto (atributo *id*) e o seu conteúdo (atributo *src*).

Também na NCL, sobre cada objeto de mídia podem ser especificadas âncoras de conteúdo, correspondentes a uma porção das unidades de informação que compõem o conteúdo de um objeto. Essas âncoras são definidas no módulo *MediaContentAnchor*, através do elemento <*area*>. Diferentes propriedades também podem ser especificadas sobre um objeto de mídia. A especificação de cada propriedade é realizada através do elemento <*property*>, definido no módulo *PropertyAnchor*.

A Figura 4.2 apresenta os principais objetos de mídia da aplicação NCL proposta como exemplo. Esses objetos são: imagem de fundo (“background” – linha 35), animação (“animation” – linhas 36 a 39), música de fundo (“music” – linha 40), foto do jogador (“photo” – linha 41), imagem da chuteira (“icon” – linhas 46 a 48), vídeo da propaganda (“shoes” – linha 49) e as propriedades globais da aplicação (“globalVar” – linhas 50 a 52).

```
...
35.    <media id="background" src="background.png" descriptor="backgroundDesc"/>
36.    <media id="animation" src="animGar.mp4" descriptor="screenDesc">
37.       <area id="segPhoto" begin="12s"/>
38.       <area id="segIcon" begin="45s" end="51s"/>
39.    </media>
40.    <media id="music" src="choro.mp3" descriptor="audioDesc"/>
41.    <media id="photo" src="photo.png" descriptor="photoDesc">
...
46.    <media id="icon" src="icon.png" descriptor="iconDesc">
47.        <property name="left"/>
48.    </media>
49.    <media id="shoes" src="shoes.mp4" descriptor="shoesDesc"/>
50.    <media id="globalVar" type="application/x-ginga-settings">
51.        <property name="channel.interactivity" value="true"/>
52.    </media>
...
```

Figura 4.2 - Objetos de mídia e suas interfaces na aplicação NCL exemplo.

Na construção do HTG, para cada objeto devem ser construídos dois vértices, correspondentes ao início e ao fim do seu evento de apresentação.[11] Esses vértices devem estar unidos por arestas, caso o objeto tenha uma duração associada (implícita ou explícita). A duração implícita é obtida pelo próprio conteúdo da mídia, enquanto a explícita é definida na especificação da aplicação. Tão logo essa duração seja conhecida, ela deve ser especificada como condição de

[11] Na conversão de uma aplicação NCL, toda transição de fim de um evento deve ser representada pela ação “*stop*”, uma vez que essa linguagem não define formas diferentes para o término de um evento (“*natural end*” e “*stop*”) (Soares, 2009a).

uma aresta. Uma parte do HTG que representa as transições dos eventos de apresentação para os objetos e as interfaces descritas na Figura 4.2 é apresentado na Figura 4.3.[12]

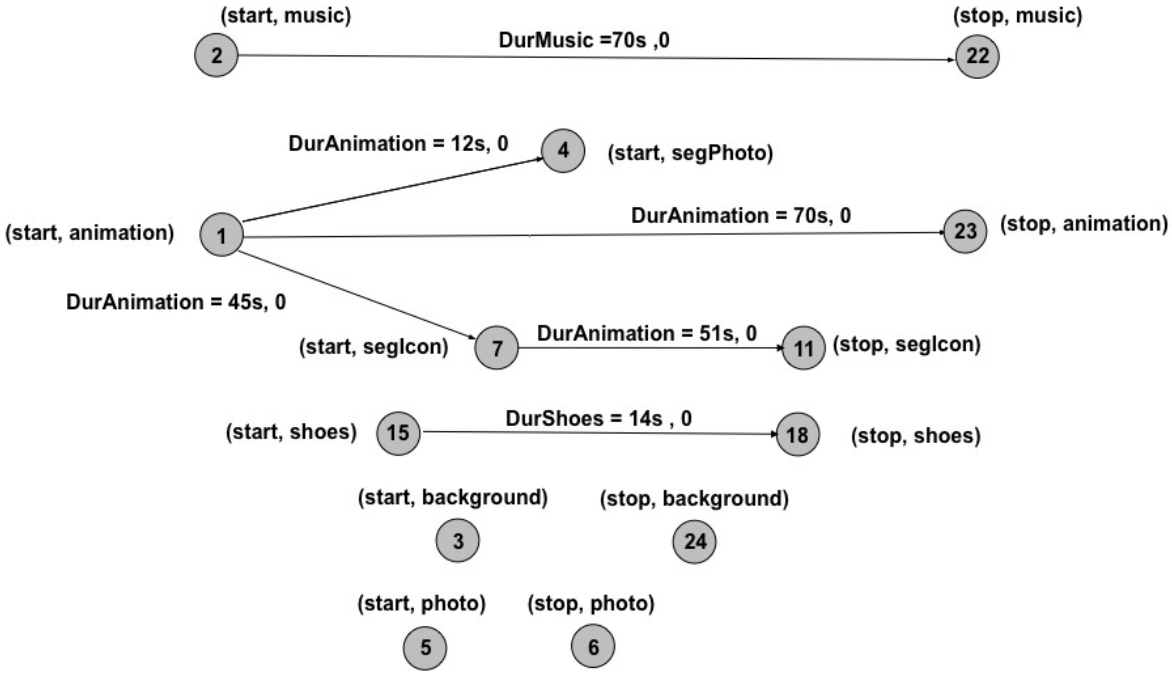


Figura 4.3 - Parte do HTG que representa os objetos e suas interfaces descritos na Figura 4.2.

As âncoras de conteúdo de uma aplicação NCL, que têm uma duração explícita, também devem ser representadas, cada uma, por dois vértices no HTG. Esses vértices correspondem ao início e ao fim do evento de apresentação sobre cada âncora. Esses pares de vértices também devem estar unidos, cada par, por uma aresta que tem como condição a duração desse evento sobre à âncora. Esse é o caso da aresta entre os vértices 7 e 11 da Figura 4.3 (“segIcon”).

Os relacionamentos de um objeto de mídia com as suas âncoras de conteúdo também devem estar especificados no HTG. Esse é o caso da aresta entre os vértices 1 e 4 e também da aresta entre os vértices 1 e 7 na Figura 4.3. Essas arestas têm como condição uma porção temporal da duração do conteúdo do objeto de mídia “animation”.

Os demais vértices da Figura 4.3 representam transições de início e de fim do evento de apresentação que podem ocorrer sobre os demais objetos de mídia especificados na Figura 4.2 (imagem de fundo, foto do jogador, imagem da

[12] Para facilitar o entendimento das figuras que representam parte de um HTG, a descrição do evento de apresentação será sempre suprimida neste capítulo. Isso acontece nas triplas correspondentes aos vértices e também nas variáveis que representam a duração desse evento.

chuteira). As ocorrências dessas transições são especificadas em outras partes da aplicação NCL, como é o caso, por exemplo, da apresentação do objeto correspondente à foto do jogador, que possui uma duração explícita, discutida na próxima seção.

## 4.3. Adaptação da Apresentação e Adaptação do Conteúdo

O autor pode definir, através da NCL, formas alternativas para a apresentação de um mesmo conteúdo (adaptação da apresentação). Também é possível ao autor definir conteúdos alternativos para apresentação (adaptação do conteúdo). Tanto a forma de apresentação quando o conteúdo a ser apresentado são escolhidos em tempo de apresentação da aplicação.

Nas aplicações NCL, a escolha entre conteúdos alternativos ou formas de apresentação alternativas é realizada através da avaliação de regras que, quando satisfeitas, selecionam uma opção. Para a especificação dessas regras, o módulo *TestRule* define o elemento <*ruleBase*>, que agrupa um conjunto de regras.

As informações espaço-temporais necessárias para a iniciação da apresentação dos componentes em uma aplicação NCL são especificadas através de descritores (ABNT, 2009). O elemento <*descriptor*>, definido no módulo *Presentation Specification*, é utilizado nessa especificação. Assim, a adaptação da apresentação é realizada através da escolha de um dos possíveis descritores para um conteúdo. Cada conjunto de descritores alternativos são agrupados através do elemento <*descriptorSwitch*>, também definido no módulo *Presentation Specification*.

Na aplicação NCL escolhida como exemplo, a duração da apresentação de uma imagem, correspondente à foto do jogador de futebol, é especificada pelo autor da aplicação de forma explícita. Na verdade, mais de uma duração é especificada, através de um conjunto de descritores alternativos. A Figura 4.4 apresenta esse conjunto de descritores (linhas 18 a 23) e o teste utilizado para a escolha de um desses descritores (linha 5).

```
...
03.      <ruleBase>
...
05.      <rule id="screenSize" var="system.screenGraphicSize" value="800,600"
                                                    comparator="lt"/>
06.      </ruleBase>
...
18.      <descriptorSwitch id="photoDesc">
19.         <bindRule constituent="withoutImgDur" rule="screenSize"/>
20.         <defaultDescriptor descriptor="withImgDur"/>
21.         <descriptor id="withoutImgDur" region="frameReg" explicitDur="10s"/>
22.         <descriptor id="withImgDur" region="frameReg" explicitDur="20s"/>
23.      </descriptorSwitch>
...
```

Figura 4.4 - Adaptação da apresentação na aplicação NCL exemplo.

Na construção do HTG, cada duração explícita deve ser representada como uma condição associada a uma aresta. Na especificação da Figura 4.4, caso a resolução gráfica do dispositivo de apresentação seja menor que 800x600 *pixels*, a duração da apresentação da foto do jogador será de 10 segundos. Caso contrário, essa duração será de 20 segundos (linha 22).

A Figura 4.5 apresenta a parte do HTG correspondente à adaptação da apresentação especificada na Figura 4.4. Nessa parte do grafo, cada uma das duas possíveis durações e o teste correspondente para a escolha de uma dessas durações são representados através de uma condição composta. Cada uma dessas condições, por sua vez, é associada a uma aresta construída entre as transições de início e de fim do evento de apresentação do objeto correspondente à foto do jogador.

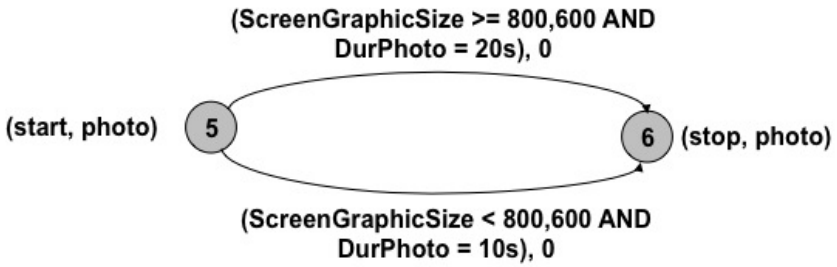


Figura 4.5 - Parte do HTG que representa a adaptação da apresentação descrita na Figura 4.4.

Em relação à adaptação do conteúdo, um conjunto de possíveis conteúdos para apresentação devem estar agrupados nas aplicações NCL através do elemento

<*switch*>, definido no módulo *ContentControl*.[13] Na aplicação NCL definida como exemplo, um *switch* agrupa dois objetos de mídia, mais especificamente, dois formulários XHTML, um em português e outro em inglês. A Figura 4.6 apresenta esses objetos (linhas 60 e 61), especificados como componentes de um elemento <*switch*> (linhas 53 a 62). Dependendo do idioma escolhido no ambiente de apresentação, cujo valor é armazenado na variável "*system.language*", um desses objetos de mídia será escolhido (linha 4).

```
...
03.      <ruleBase>
04.        <rule id="en" var="system.language" value="en" comparator="eq"/>
...
06.      </ruleBase>
...
53.       <switch id="form">
54.            <switchPort id="spForm">
55.               <mapping component="enForm"/>
56.               <mapping component="ptForm"/>
57.            </switchPort>
58.            <bindRule constituent="enForm" rule="en"/>
59.            <defaultComponent component="ptForm"/>
60.            <media id="ptForm" src="ptForm.htm" descriptor="formDesc"/>
61.            <media id="enForm" src="enForm.htm" descriptor="formDesc"/>
62.        </switch>
...
```

Figura 4.6 - Adaptação do conteúdo na aplicação NCL exemplo.

No HTG que representa a aplicação NCL, para cada elemento filho de um *switch*, devem ser construídos vértices representando as transições dos eventos que podem ocorrer sobre esse *switch*. Essas possíveis transições são definidas através dos relacionamentos com a participação do *switch* (relacionamentos serão vistos na Seção 4.5). A Figura 4.7 apresenta uma parte do HTG, correspondente a representação do *switch* especificado na Figura 4.6.

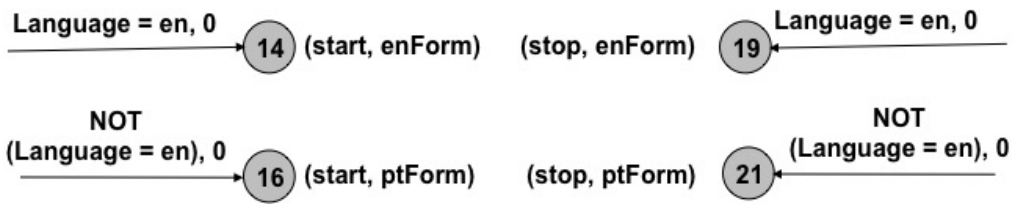


Figura 4.7 - Parte do HTG que representa a adaptação do conteúdo descrita na Figura 4.6.

[13] Um elemento <*switch*> agrupa um conjunto de nós alternativos, que podem ser objetos de mídia, outros elementos <*switch*> ou ainda contextos, que serão vistos ainda neste capítulo.

Na Figura 4.7, os vértices representam as transições de início (vértices 14 e 16) e de fim (vértices 19 e 21) da apresentação de cada um dos objetos de mídia existentes no *switch* (“enForm” e “ptForm”). Quando um relacionamento tem como consequência o disparo de uma transição de evento sobre um *switch*, uma aresta no HTG deve ser construída para cada um dos elementos filhos desse *switch*. A condição associada a cada uma dessas arestas deve ser a mesma definida nas regras para a escolha de um dos componentes do *switch*. Esse é o caso das arestas com destino nos vértices 14 e 19 (início e fim da apresentação de “ptForm”) que têm como condição que o idioma do sistema de apresentação seja inglês (“Language = en”). Esse também é o caso das arestas com destino nos vértices 16 e 21 (início e fim da apresentação de “enForm”), que têm como condição que esse idioma não seja inglês (“NOT Language = en”).

Quando um relacionamento na NCL tem como causa o disparo de uma transição de evento sobre um *switch*, uma aresta também deve ser construída para cada um dos elementos filhos desse *switch*. Essas arestas, no entanto, não têm como condição regras para a escolha de um dos elementos filhos do *switch*. Apenas as condições existentes no próprio relacionamento devem ser condições dessas arestas (relacionamentos serão vistos na Seção 4.5).

Além de objetos de mídia, um *switch* pode conter outros elementos <*switch*> e também contextos como filhos. No caso do elemento filho também ser um *switch*, na construção do HTG, as regras dos elementos <*switch*> filhos devem ser adicionadas às regras do elemento pai, formando regras compostas. No caso dos contextos, os elementos filhos desse contexto devem ser mapeados, para a construção dos vértices e arestas. A representação dos contextos da NCL no HTG será discutida na Seção 4.6.

## 4.4. Reúso

Uma das principais facilidades oferecidas pela NCL é a possibilidade de reusar os seus elementos (Neto, 2009), em particular, os objetos de mídia e os contextos. O reúso deve ser especificado através do atributo *refer*, definido no módulo *Reuse*. O valor desse atributo deve ser o identificador (valor do atributo *id*) do elemento que será reusado.

O reúso em um contexto (elemento <*context*>) faz com que uma cópia do contexto referenciado (valor do atributo *refer*), incluindo todos os seus elementos filhos, seja realizada.

O reúso em um objeto de mídia (<media>) também gera uma cópia do objeto referenciado (indicado no atributo *refer*). No entanto, diferente do reuso de contextos, o objeto que realiza a referência pode definir novos elementos filhos (<*area*> e <*property*>). Todos os atributos de um elemento <*media*>, que faz referência a outro objeto, devem ser ignorados e substituídos pelos atributos do elemento referenciado. Exceto o atributo *id* e, em alguns casos, o atributo *descriptor*, dependendo do valor do atributo *instance*, presente no elemento que faz a referência. Esse atributo pode ter os valores "*instSame*", "*new*" ou "*gradSame*". Cada um desses valores define uma variação na forma com que os objetos podem ser reusados, gerando arestas e vértices diferentes no HTG, como será descrito a seguir.

Quando o atributo *instance* do objeto que faz a referência tem o valor "*instSame*", o objeto que referencia é o mesmo objeto referenciado, desde a sua instanciação, incluindo todas as suas propriedades e âncoras de conteúdo. Esse é o caso do objeto "*reusedAnimation*", apresentado na Figura 4.8, que reusa as especificações do objeto "*animation*".

```
...
43.       <media id="reusedAnimation" refer="animation" instance="instSame">
44.           <property name="bounds"/>
45.       </media>
...
```

Figura 4.8 - Reúso na aplicação NCL exemplo.

A especificação da Figura 4.8 foi retirada de parte da aplicação NCL utilizada como exemplo ao longo deste capítulo. Nessa aplicação, um contexto é utilizado para agrupar semanticamente os objetos e os relacionamentos associados a uma publicidade (a propaganda da chuteira). Nesse contexto, existem relacionamentos com a participação do objeto que representa a animação, tanto para iniciar a apresentação da imagem da chuteira, quanto para redimensionar a animação, caso essa imagem seja selecionada. Assim, o objeto "animation" é reusado no contexto para facilitar a especificação dos relacionamentos.

Na construção do HTG referente ao objeto "reusedAnimation", são utilizados os mesmos vértices que representam as transições do evento de

apresentação do objeto “animation” e de suas âncoras de conteúdo. A parte do HTG que representa a especificação da Figura 4.8 é a mesma apresentada na Figura 4.3, para o objeto “animation” e suas âncoras de conteúdo (vértices 1, 4, 7, 11 e 13). Assim, quando o reúso “*instSame*” é utilizado, nenhum vértice ou aresta precisa ser adicionado ao HTG, exceto se novas âncoras de conteúdo forem especificadas.

O atributo *instance* de um objeto também pode ter o valor “*new*”. Nesse caso, uma cópia do objeto referenciado é realizada, isto é, um objeto totalmente independente, cópia de outro, é construído. Somente nesse caso o atributo *descriptor* do objeto que faz a referência pode ser redefinido. O valor “*new*” é o padrão (*default*) para o atributo *instance*, caso esse atributo não seja especificado.

Quando o reúso na aplicação NCL envolve uma nova instância de um objeto, uma cópia dos vértices e das arestas associadas ao objeto referenciado deve ser realizada no HTG que representa a aplicação. Assim, um novo objeto é representado no grafo, e novos vértices e arestas são construídos para representar as transições dos eventos sobre esse novo objeto.

O outro valor possível para o atributo *instance* é “*gradSame*”. Nesse outro caso, o objeto que faz a referência e o objeto referenciado são o mesmo, porém, as incorporações no objeto que faz a referência serão realizadas à medida que suas propriedades e âncoras de conteúdo tiverem suas apresentações iniciadas.

No reúso “*gradSame*”, as transições de início para o evento de apresentação dos objetos (que faz a referência e o referenciado) podem ocorrer em instantes temporais distintos. Nesse caso, no HTG que representa uma aplicação NCL, devem ser construídos vértices correspondentes ao início e ao fim do evento de apresentação para um objeto que utiliza o reúso “*gradSame*”.

Para exemplificar a representação do HTG relativo a aplicações NCL com objetos que utilizam o reúso “*gradSame*”, uma aplicação contendo três vídeos (“vídeoA”, “vídeoB” e “vídeoC”) é proposta. A apresentação dessa aplicação é iniciada através da apresentação do objeto “videoA”. O início da apresentação desse objeto inicia também a apresentação do objeto “videoB”. O objeto “videoC”, que reusa o objeto “videoB” e que especifica o atributo *instance* com valor “*gradSame*”, somente é iniciado caso ocorra a seleção do objeto “videoA”. A Figura 4.9 apresenta o HTG dessa aplicação.

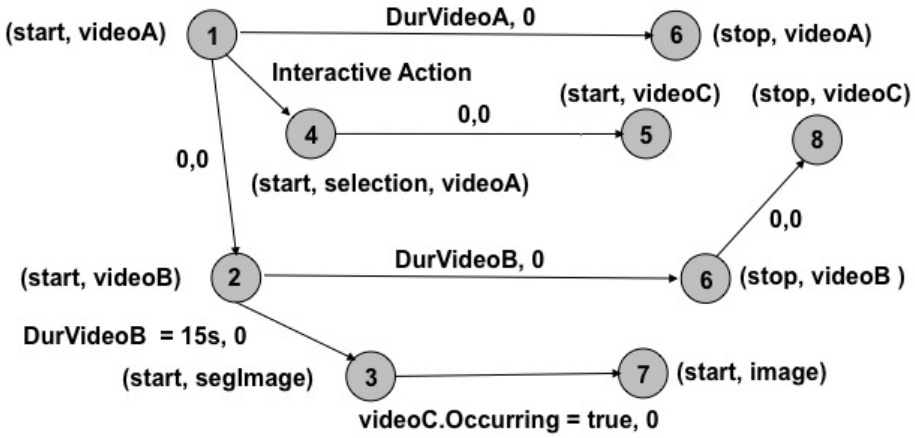


Figura 4.9 - HTG para o exemplo de reuso "*gradSame*".

Continuando a descrição do exemplo apresentado na Figura 4.9, uma imagem ("image") deve ser apresentada em um instante temporal específico da apresentação do objeto "videoC". Esse instante é definido através de uma âncora de conteúdo ("segImage"), originalmente especificada no objeto "videoB" e herdada pelo objeto "videoC".

Como pode ser observado na Figura 4.9, no reúso "*gradSame*", as mesmas âncoras de conteúdo do objeto referenciado são utilizadas nos relacionamentos com a participação das âncoras herdadas pelo objeto que faz a referência. Porém, para representar a semântica desse reúso, uma variável, referente à ocorrência (*occurring*) da apresentação do objeto que faz a referência, é necessária como condição nos relacionamentos com origem nessas âncoras. Nesse exemplo, a apresentação do objeto "image" será iniciada apenas se o objeto "videoC" estiver sendo apresentado quando o instante temporal previsto para o início da apresentação da âncora "segImage" for alcançado (15 segundos da duração de "videoB").

O controle da ocorrência da apresentação dos objetos somente precisa ser realizado quando a transição de início da apresentação de um objeto que especifica o reúso "*gradSame*" e as transições das suas âncoras de conteúdo estão em cadeias temporais diferentes. Esse é o caso do HTG da Figura 4.9, onde os vértices 4 e 5 são parte de uma cadeia secundária. Por outro lado, quando todas as transições de eventos associadas a um mesmo objeto fazem parte da mesma cadeia, essas transições são previsíveis em relação a um mesmo ponto de origem, o início da cadeia temporal. Nesse outro caso, relacionamentos que dependem da

ocorrência de um objeto podem ser avaliados antes da apresentação, simplificando o controle da apresentação em relação à especificação da NCL.

## 4.5. Relacionamentos

Nas aplicações NCL, os relacionamentos são especificados através do elemento <*link*>, definido no módulo *Linking*. Esse elemento agrupa elementos <*bind*>, também definidos no módulo *Linking*, que especificam os participantes de uma relação. É o conector, definido no módulo *ConnectorBase*, que especifica a semântica da relação de um elo. Por exemplo, em uma relação causal, o conector especifica condições que devem ser satisfeitas para que ações possam ser disparadas.

Conectores podem ser especificados na própria aplicação NCL, através do elemento <*causalConnector*>. Cada elemento <*causalConnector*> é especificado como um filho de um elemento <*connectorBase*>, que agrupa (define uma base de) conectores. Uma outra opção consiste em importar os conectores, através do elemento <*importBase*>. A Figura 4.10 contém uma parte da aplicação NCL proposta como exemplo, onde uma base de conectores é especificada. Nessa base são importados conectores especificados em outro documento (causalConnBase.ncl).

```
...
29.      <connectorBase>
30.        <importBase documentURI="../causalConnBase.ncl" alias="conEx"/>
31.      </connectorBase>
...
```

Figura 4.10 - Base de conectores utilizados na aplicação NCL exemplo.

Neste ponto, é importante mencionar que todas as relações exploradas nesta tese são especificadas por conectores causais, uma vez que elas são as únicas permitidas no perfil da linguagem para TV digital. Além disso, as condições dessas relações podem simples ou compostas, isto é, podem ser formadas por mais de uma condição, mas têm somente uma única condição que deve ser satisfeita em um instante de tempo infinitesimal.

Nos conectores, condições e ações definem um papel (atributo *role*), que deve ser único na especificação do conector. São esses papéis que serão assumidos pelos participantes de um relacionamento, associados através dos

<*binds*> de um elo. Para facilitar a especificação das aplicações, dois conjuntos de papéis são pré-definidos na NCL. Esses papéis associam transições das máquinas de estado dos eventos, que podem representar tanto condições (primeiro conjunto, descrito na Tabela 4.2) quanto ações (segundo conjunto, descrito na Tabela 4.3 em um relacionamento.

| Papel (role) | Ação que executa a transição | Evento |
|---|---|---|
| *onBegin* | *start* | *presentation* |
| *onEnd* | *stop* | *presentation* |
| *onAbort* | *abort* | *presentation* |
| *onPause* | *pause* | *presentation* |
| *onResume* | *resume* | *presentation* |
| *onSelection* | *start* | *selection* |
| *onBeginAttribution* | *start* | *attribution* |
| *onEndAttribution* | *start* | *attribution* |

Tabela 4.2 - Valores reservados para papéis utilizados como condição.

| Papel (role) | Ação que executa a transição | Evento |
|---|---|---|
| *start* | *start* | *presentation* |
| *stop* | *stop* | *presentation* |
| *abort* | *abort* | *presentation* |
| *pause* | *pause* | *presentation* |
| *resume* | *resume* | *presentation* |
| *set* | *start* | *attribution* |

Tabela 4.3 - Valores reservados para papéis utilizados como ação.

Voltando ao exemplo inicial sobre a animação introduzido na Seção 4.1, como vários elos são especificados na aplicação NCL, a construção de partes do HTG correspondentes aos relacionamentos será realizada paulatinamente. A Figura 4.11 apresenta alguns dos elos especificados nessa aplicação NCL. Nessa figura estão os elos utilizados na especificação do sincronismo temporal da aplicação NCL utilizada como exemplo que têm uma condição e uma ação simples.

```
...
94.      <link  id="lMusic" xconnector="conEx#onBeginStart">
95.        <bind role="onBegin" component="animation"/>
96.        <bind role="start" component="music"/>
97.      </link>
98.      <link id="lBackground" xconnector="conEx#onBeginStart">
99.        <bind role="onBegin" component="animation"/>
100.       <bind role="start" component="background"/>
101.     </link>
102.     <link id="lPhoto" xconnector="conEx#onBeginStart">
103.       <bind role="onBegin" component="animation" interface="segPhoto"/>
104.       <bind role="start" component="photo"/>
105.     </link>
106.     <link id="lEnd" xconnector="conEx#onEndStop">
107.       <bind role="onEnd" component="animation"/>
108.       <bind role="stop" component="background"/>

109.     </link>
110.     <link id="lEndIcon" xconnector="conEx#onEndStop">
111.       <bind role="onEnd" component="animation" interface="segIcon"/>
112.       <bind role="stop" component="icon"/>
113.     </link>
...
```

Figura 4.11 - Relacionamentos associados ao sincronismo no tempo (sem interação) na aplicação NCL exemplo.

Na Figura 4.11, três elos têm como condição que o evento de apresentação sobre um objeto seja iniciado (“*onBegin*”) para que a apresentação de um outro objeto também seja iniciada (“*start*”). Outros dois elos têm como condição que o evento de apresentação sobre um objeto seja encerrado (“*onEnd*”) para que a apresentação, de um outro objeto, também seja encerrada (“*stop*”).

Na construção do HTG, elos da aplicação NCL que possuem uma condição e uma ação simples são representados por uma única aresta, unindo os vértices correspondentes à condição e à ação desse relacionamento. A Figura 4.12 apresenta as arestas construídas em parte do HTG para os elos especificados na Figura 4.11. Os vértices e arestas correspondentes a esses elos são destacados em vermelho nessa figura.

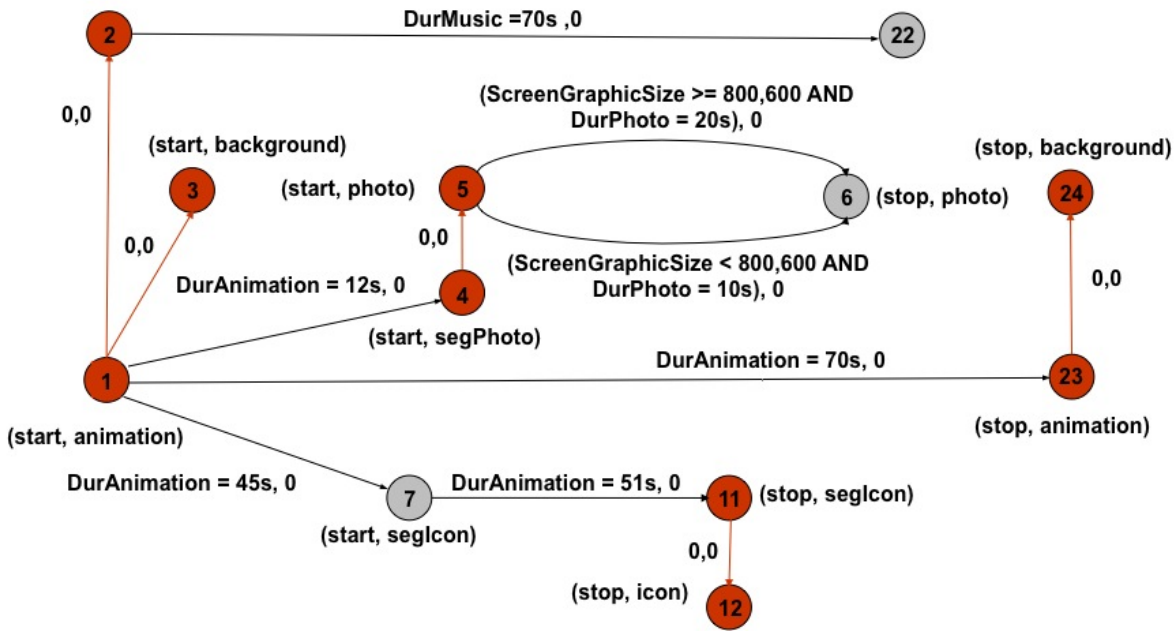


Figura 4.12 - Parte do HTG que representa o sincronismo no tempo especificado na Figura 4.11.

Na Figura 4.12, as arestas entre os vértices 1 e 2, 1 e 3, 4 e 5, 11 e 12, 23 e 24, representam, respectivamente, a semântica dos elos "*lMusic*", "*lBackground*", "*lPhoto*", "*lEndIcon*" e "*lEnd*".

Na mesma aplicação NCL encontram-se também relacionamentos com condições e ações compostas. Uma condição composta, como mencionado, pode ter outras condições, simples ou compostas, incluindo itens a serem avaliados. Uma ação composta pode ser formada por outras ações, simples ou compostas, que devem ser executadas em paralelo ou sequencialmente, uma após a outra (não é necessário aguardar o término de uma ação para o disparo da próxima).

O elo "*lIcon*", definido na aplicação NCL e especificado na Figura 4.13, tem como condições o início da apresentação de uma âncora ("*segIcon*") do objeto "*reusedAnimation*" (linha 64) e também que a interatividade esteja habilitada (linhas 65 a 67). Atendidas essas condições, a apresentação da imagem da chuteira deve ser iniciada (linha 68) e também iniciada a atribuição de valores propriedade "*left*" dessa imagem (posição espacial) (linhas 69 a 73). Os valores atribuídos a essa variável variam no tempo, em um intervalo de 5 segundos (animação).

```
...
63.        <link  id="lIcon" xconnector="conEx#onBeginTestStartSetDelay">
64.          <bind role="onBegin" component="reusedAnimation"
                                                    interface="segIcon"/>
65.           <bind role="testVar" component="globalVar"
                                              interface="channel.interactivity">
66.              <bindParam name="var" value="true"/>
67.          </bind>
68.          <bind role="start" component="icon"/>
69.          <bind role="set" component="icon" interface="left">
70.              <bindParam name="var" value="87.5%"/>
71.              <bindParam name="delay" value="1s"/>
72.              <bindParam name="duration" value="5s"/>
73           </bind>
74.       </link>
...
```

Figura 4.13 - Relacionamento associado ao sincronismo no tempo (com condição) na aplicação NCL exemplo.

Para representar condições compostas, uma única aresta é construída no HTG. O vértice de origem dessa aresta é uma dessas condições. As demais condições são associadas à aresta como condição (do HTG) para a ocorrência do seu vértice de destino. Mais de uma aresta pode ser necessária para representar uma ação composta. O vértice que representa cada ação simples de uma ação composta é o destino de uma aresta construída. A Figura 4.14 representa os vértices e arestas (em vermelho) construídos para representar a semântica do elo da Figura 4.13 (“*lIcon*”).

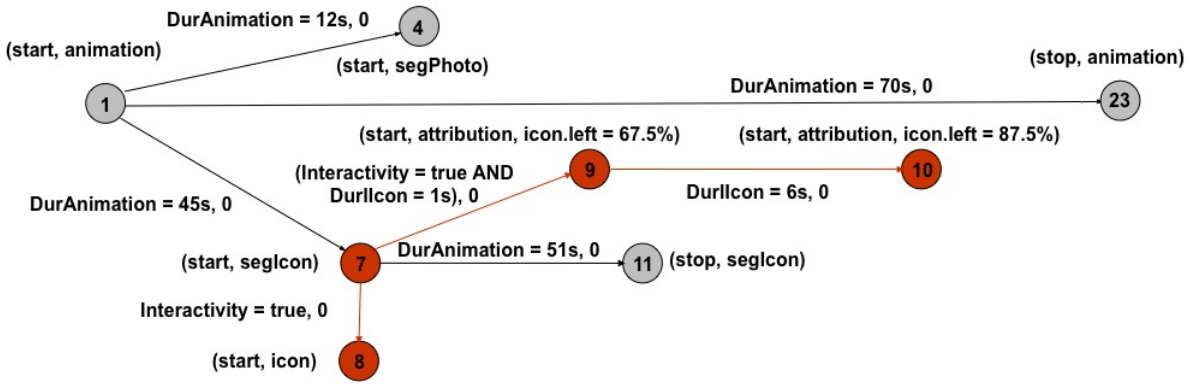


Figura 4.14 - Parte do HTG que representa o sincronismo no tempo (com condição) especificado na Figura 4.13.

Na Figura 4.14, as condições do elo "*lIcon*" são representadas pelo vértice 7, correspondente à transição de início do evento de apresentação de "segIcon",[14] e pela condição associada às arestas ("*interactivity = true*"). Como esse elo especifica duas ações, é criada uma aresta para cada uma dessas ações, com origem no vértice 7 (condição). Uma dessas arestas tem como destino o vértice 8, que representa o início da apresentação do objeto "*icon*" (imagem da chuteira). O destino da outra aresta é o vértice 9, que representa o início da atribuição da propriedade "*left*" do objeto "*icon*".

No elo "*lIcon*", um atraso deve ser obedecido antes do início da atribuição da propriedade "*left*". Esse atraso é representado no HTG como mais uma condição da aresta entre os vértices 7 e 9. O valor final da atribuição da propriedade "*left*" é representado pelo vértice 10. A aresta que une os vértices 9 e 10 tem como condição uma duração (5 segundos) que deve ser obedecida para que o valor final da atribuição da propriedade "*left*" seja alcançado.

Assim como os objetos, os elos da NCL também podem ter uma duração. Essa duração é uma variável identificada pelo nome do elo. O seu valor é iniciado quando o vértice que corresponde à condição do elo é alcançado. Como será discutido na Seção 4.6, o valor dessa variável pode permanecer inalterado, caso o contexto onde o elo foi definido seja pausado.

Nas aplicações NCL, a condição de um relacionamento também pode ser uma ação interativa. Esse é o caso do elo especificado na Figura 4.15, que tem como condição o início da seleção da imagem da chuteira. Para iniciar essa seleção, uma ação interativa deve acontecer, através do pressionamento do botão vermelho do controle remoto pelo usuário ("*keyCode = RED*"). Uma vez atendida essa condição, um conjunto de transições serão disparadas: o início da apresentação do vídeo da propaganda (linha 79), o início da apresentação do elemento <*switch*> (linha 80), o início da atribuição da animação, através do objeto "*reusedAnimation*" (linhas 81 a 83) e a interrupção da apresentação da imagem da chuteira (linha 84).

[14] Como pode ser observado na Figura 4.14, a condição do elo é um vértice que representa um evento sobre uma âncora de conteúdo do objeto "*animation*". Essa associação deve-se ao reúso do objeto "*animation*", especificado no objeto "*reusedAnimation*" (Seção 4.4).

```
...
75.       <link  id="lBeginShoes" xconnector="conEx#onKeySelectionStopSetStart">
76.         <bind role="onSelection" component="icon">
77.            <bindParam name="keyCode" value="RED"/>
78.         </bind>
79.         <bind role="start" component="shoes"/>
80.         <bind role="start" component="form" interface="spForm"/>
81.         <bind role="set" component="reusedAnimation" interface="bounds">
82.           <bindParam name="varSet" value="5%,6.67%,45%,45%"/>
83.         </bind>
84.         <bind role="stop" component="icon"/>
85.       </link>
...
```

Figura 4.15 - Relacionamento associado ao sincronismo no tempo (com interação) na aplicação NCL exemplo.

A Figura 4.16 representa os vértices e arestas construídas para representar o elo da Figura 4.15 (“*lBeginShoes*”). Nessa figura, o vértice 13 representa a transição de início da seleção da imagem da chuteira. As arestas com origem nesse vértice tem como destino vértices que representam cada uma das ações do elo “*lBeginShoes*”. Essas ações são o fim da apresentação da imagem da chuteira (vértice 12), o início da apresentação do vídeo da propaganda (vértice 15), o início da atribuição do filme da animação e o início da apresentação do elemento <*switch*>. Como mencionado na Seção 4.3, transições de um evento sobre um *switch* são realizadas sobre os seus elementos filhos, no caso, o início da apresentação do formulário em inglês (vértice 14) e do formulário em português (vértice 16).

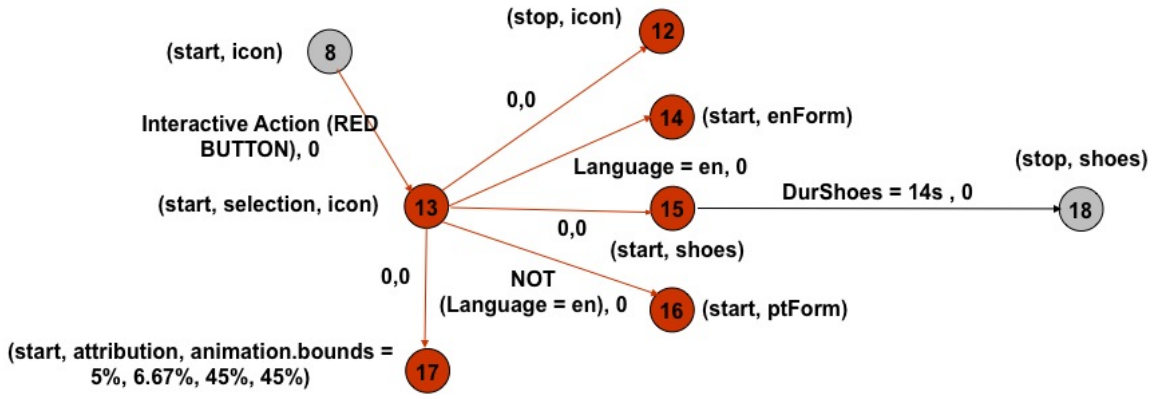


Figura 4.16 - Parte do HTG que representa o sincronismo no tempo (com interação) especificado na Figura 4.15.

A Figura 4.17 apresenta o último elo da aplicação NCL exemplo. Esse elo tem como condição o fim da apresentação do vídeo da propaganda (linha 87).

Como consequência dessa condição, duas ações devem ser executadas: o início da atribuição do filme da animação (linhas 88 a 90), restaurando as dimensões originais desse objeto e o fim da apresentação do elemento <*switch*> (linha 91).

```
...
86.       <link  id="lEndShoes" xconnector="conEx#onEndSetStop">
87.         <bind role="onEnd" component="shoes"/>
88.         <bind role="set" component="reusedAnimation" interface="bounds">
89.           <bindParam name="varSet" value="0,0,222.22%,222.22%"/>
90.         </bind>
91.         <bind role="stop" component="form"/>
92.       </link>
...
```

Figura 4.17 - Relacionamento associado a um evento de atribuição na aplicação NCL exemplo.

A Figura 4.18 apresenta os vértices e arestas construídas para representar o elo da Figura 4.17. O vértice 18 representa a transição de evento que é a condição desse elo (fim da apresentação do vídeo da propaganda). Esse vértice é a origem de três arestas. Uma das arestas, entre os vértices 18 e 20, associa o final do evento de apresentação do vídeo da propaganda ao início da atribuição do objeto "*animation*". As outras duas arestas, entre os vértices 18 e 19 e entre os vértices 18 e 21, associam o final do evento de apresentação do vídeo da propaganda ao final desse evento para cada um dos objetos filhos do *switch* ("*enForm*" e "*ptForm*").

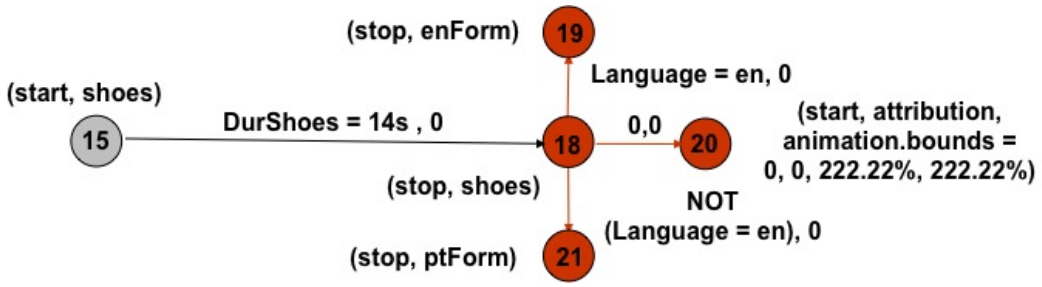


Figura 4.18 - Parte do HTG que representa o evento de atribuição especificado na Figura 4.17.

## 4.6. Contextos

NCL é uma linguagem estruturada, que permite agrupar objetos de mídia e seus relacionamentos (elos) em diferentes contextos, definidos no módulo

*Context*. Cada contexto pode ter um conjunto de portas (elemento <*port*>), definidas no módulo *CompositeNodeInterface*. Uma porta especifica um mapeamento para um dos componentes de um contexto ou para uma outra interface de um desses componentes.[15]

Contextos não têm semântica temporal implícita, eles representam a estrutura lógica de uma aplicação. Dessa forma, contextos não interferem no sincronismo temporal de uma aplicação NCL, exceto nos casos em que os relacionamentos são especificados diretamente sobre um contexto, quer seja como condição ou como ação de uma relação causal.[16]

A Tabela 4.4 resume o comportamento dos componentes de um contexto, quando uma ação, relativa ao evento de apresentação, é executada diretamente sobre esse contexto, sem que nenhuma das suas portas seja mencionada.

| Ações | Comportamento |
|---|---|
| *start* | Aplicada a todos os componentes mapeados pelas portas existentes no contexto. |
| *stop* | Aplicada a todos os componentes existentes no contexto. Caso o contexto contenha elos sendo avaliados ou com a avaliação pausada, essas avaliações serão suspensas. |
| *abort* | Aplicada a todos os componentes existentes no contexto. Caso o contexto contenha elos sendo avaliados ou com a avaliação pausada, essas avaliações serão suspensas. |
| *pause* | Aplicada a todos os componentes existentes no contexto. Caso o contexto contenha elos sendo avaliados, essas avaliações serão pausadas. |



[15] Um contexto também pode ter propriedades (elemento <*property*>), definidas no módulo *PropertyAnchor*. O HTG não diferencia propriedades pertencentes a um objeto de mídia ou a um contexto. Assim, a representação no HTG de ambos os casos é a mesma.

[16] Esse caso é diferente da especificação de relacionamentos diretamente sobre os componentes de um contexto, acessados através de suas portas. Nesse outro caso, o HTG é construído considerando apenas os elementos filhos do contexto.

| | |
|---|---|
| *resume* | Aplicada a todos os componentes existentes no contexto. Retoma a apresentação dos elementos anteriormente pausados pela ação de pause no contexto (os que estavam pausados antes da ação de pause no contexto permanecem pausados) e a avaliação dos elos existentes em um contexto que foram pausados por uma ação de pause aplicada ao contexto. |

Tabela 4.4 - Comportamento dos componentes de um contexto de acordo com as ações realizadas sobre esse contexto.

Para ilustrar a construção de um HTG para cada possível ação executada diretamente sobre um contexto, a Figura 4.19 apresenta, como exemplo, a visão estrutural de um contexto que agrupa três objetos de mídia (A, B e C). Nesse contexto são especificadas duas portas, que definem, cada uma, um mapeamento para os objetos “A” e “B”.

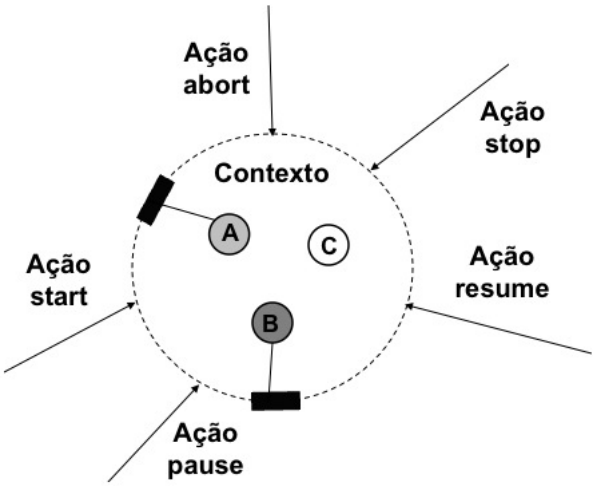


Figura 4.19 - Visão estrutural de um contexto com ações.

Um vértice no HTG, para cada componente de um contexto, deve ser construído para representar as ações do tipo “*stop*” ou “*abort*” executadas diretamente sobre esse contexto. No caso das ações do tipo “*start*”, devem ser construídos vértices apenas para os componentes mapeados através das suas portas. A Figura 4.20 apresenta os vértices e as arestas de parte de um HTG que representa a visão estrutural proposta na Figura 4.19, para as ações “*start*”, “*stop*” e “*abort*”.

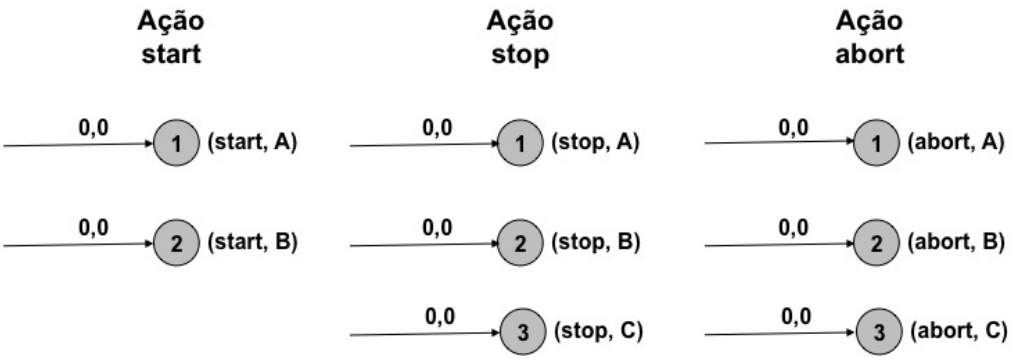


Figura 4.20 - Parte de um HTG que representa as ações de "*start*", "*stop*" e "*abort*" sobre o contexto da Figura 4.19.

Ações do tipo "*pause*" para a apresentação de um contexto devem ser representadas no HTG por dois vértices, para cada componente desse contexto. Um desses vértices corresponde à transição de pausa ("*pause*") da apresentação de cada componente do contexto. O outro vértice corresponde à transição de início do evento de atribuição de uma variável (*paused*). Essa variável contém a informação de que um componente teve sua apresentação pausada por uma ação executada sobre o seu contexto. Essa informação é importante, uma vez que, ações do tipo "*resume*" sobre um contexto, discutidas a seguir, só terão efeito sobre os componentes desse contexto se a apresentação desses componentes foi pausada por uma ação do tipo "*pause*", previamente executada sobre esse mesmo contexto.

A Figura 4.21 apresenta os vértices de parte de um HTG construídos para representar uma ação do tipo "*pause*" executada sobre o contexto da visão estrutural da Figura 4.19. Para que as transições representadas pelos vértices da Figura 4.21 sejam executadas, é necessário que o objeto de mídia correspondente ao vértice esteja ocorrendo (*occurring*). Essa condição é representada nas arestas que têm como destino os vértices construídos.

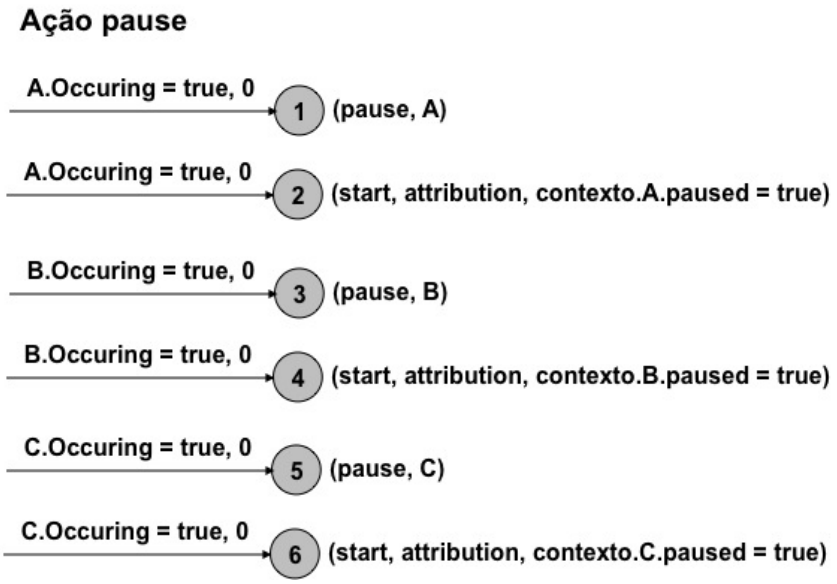


Figura 4.21 - Parte de um HTG que representa a ação de "*pause*" sobre o contexto da Figura 4.19.

Um vértice no HTG, para cada componente de um contexto, deve ser construído para representar as ações do tipo "*resume*" na apresentação desse contexto. Para que as transições representadas em cada vértice construído sejam executadas, é necessário que a apresentação do componente tenha sido anteriormente pausada por uma ação do tipo "*pause*", executada sobre o seu contexto. Essa condição é associada à aresta que tem como destino cada vértice construído, conforme apresentado em parte do HTG da Figura 4.22.

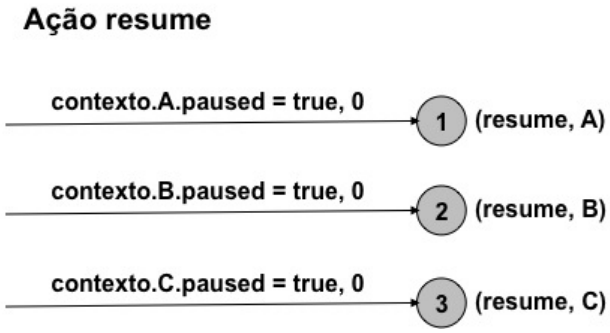


Figura 4.22 - Parte de um HTG que representa a ação de "*resume*" sobre o contexto da Figura 4.19.

Na aplicação NCL, o estado do evento de apresentação de um contexto pode ser especificado como condição em um relacionamento. A Tabela 4.5 resume os possíveis estados da apresentação de um contexto. Esses estados dependem dos eventos de apresentação dos componentes (elementos filhos) de um contexto.

| Estados | Estados dos eventos de apresentação dos elementos filhos |
|---|---|
| *occurring* | Se ao menos um componente do contexto tiver o seu evento de apresentação no estado *occurring* |
| *sleeping* | Se todos os componentes do contexto tiverem os seus eventos de apresentação no estado *sleeping*. |
| *paused* | Se todos os componentes do contexto tiverem os seus eventos de apresentação em um estado diferente de *occurring* e pelo menos um desses componentes tiver o seu evento de apresentação no estado *paused*. |

Tabela 4.5 - Estados da apresentação de um contexto.

Para ilustrar a construção de um HTG para cada possível condição associada aos estados de um contexto, a Figura 4.23 apresenta, como exemplo, a visão estrutural do mesmo contexto da Figura 4.19. A diferença da Figura 4.23 são as palavras reservadas ("*onBegin*", "*onPause*", "*onEnd*") utilizadas para destacar as possíveis condições que podem estar associadas a um contexto.

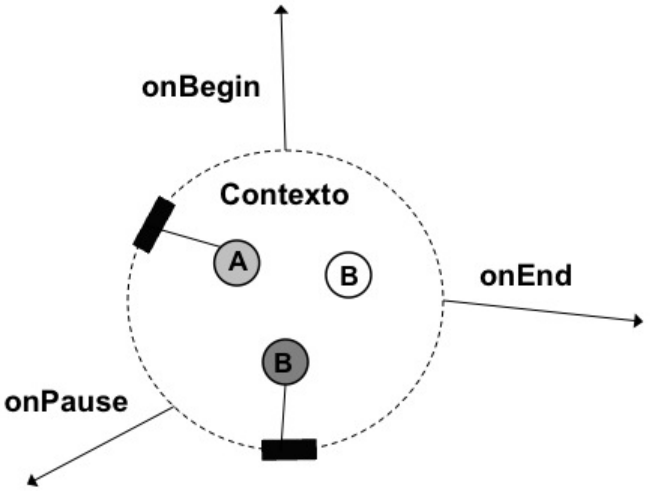


Figura 4.23 - Visão estrutural de um contexto com condições associadas.

Uma aresta no HTG, para cada componente de um contexto, deve ser construída para representar as condições do tipo "*onBegin*", "*onEnd*" e "*onPause*" especificadas sobre esse contexto. No caso das condições do tipo "*onBegin*" nenhuma condição associada às arestas é necessária. Essas arestas têm como origem o vértice que representa o início da apresentação de cada componente (elemento filho) desse contexto. Para que essa condição seja atendida, basta que qualquer uma das apresentações de um dos componentes do contexto seja

iniciada. A Figura 4.24 apresenta os vértices e as arestas de parte de um HTG que representa a visão estrutural proposta na Figura 4.23.

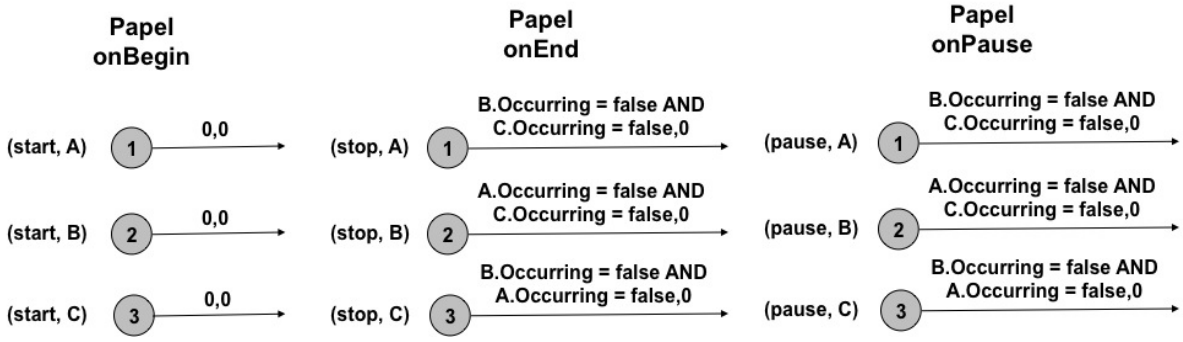


Figura 4.24 - Parte do HTG que representa as condições "*onBegin*", "*onPause*" e "*onEnd*" sobre o contexto da Figura 4.23.

Para representar as demais condições que podem estar associadas a um contexto em uma aplicação NCL, as arestas do HTG devem ter como condição que a apresentação dos demais componentes do contexto não estejam ocorrendo. Esse é o caso quando o contexto participa em um relacionamento associado a um papel "*onEnd*". O mesmo acontece quando o contexto participa em um relacionamento associado a um papel "*onPause*". A diferença entre esses casos é que, no primeiro, as arestas têm origem na transição de fim do evento de apresentação de cada componente do contexto e, no segundo, o vértice origem de cada aresta corresponde à transição de pausa do evento de apresentação de cada componente do contexto.

As variáveis utilizadas para representar a semântica dos contextos de uma aplicação NCL através do HTG (*occurring*, *paused*) somente são necessárias no grafo quando os seus valores podem ser alterados em cadeias temporais diferentes daquela onde os valores dessas variáveis são avaliados. Essa simplificação para o controle da apresentação em relação à especificação da NCL é a mesma utilizada na representação do reúso "*gradSame*", abordada na Seção 4.4.

A Figura 4.25 apresenta a especificação da aplicação NCL descrita neste capítulo. O HTG construído para essa aplicação ao longo deste capítulo pode ser visualizado na Figura 4.26.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Exemplo de Aplicação NCL -->
01. <ncl id="htg" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
02.   <head>
03.     <ruleBase>
04.       <rule id="en" var="system.language" value="en" comparator="eq"/>
05.       <rule id="screenSize" var="system.screenGraphicSize" value="800,600"
                                                          comparator="lt"/>
06.     </ruleBase>
07.     <regionBase>
08.      <region id="backgroundReg" width="100%" height="100%" zIndex="1"/>
09.      <region id="screenReg" width="100%" height="100%" zIndex="2">
10.      <region id="frameReg" left="5%" top="6.7%" width="18.5%"
                                          height="18.5%" zIndex="3"/>
11.      <region id="iconReg" left="67.5%" top="11.7%" width="8.45%"
                                        height="6.7%" zIndex="3"/>
12.      <region id="shoesReg" left="15%" top="60%" width="25%" height="25%"
                                                    zIndex="3"/>
13.      <region id="formReg" left="56.25%" top="8.33%" width="38.75%"
                                      height="71.7%" zIndex="3"/>
14.     </regionBase>
15.     <descriptorBase>
16.      <descriptor id="backgroundDesc" region="backgroundReg"/>
17.      <descriptor id="screenDesc" region="screenReg"/>
18.      <descriptorSwitch id="photoDesc">
19.         <bindRule constituent="withoutImgDur" rule="screenSize"/>
20.         <defaultDescriptor descriptor="withImgDur"/>
21.         <descriptor id="withoutImgDur" region="frameReg"
                                                    explicitDur="10s"/>
22.         <descriptor id="withImgDur" region="frameReg" explicitDur="20s"/>
23.      </descriptorSwitch>
24.      <descriptor id="audioDesc"/>
25.      <descriptor id="iconDesc" region="iconReg"/>
26.      <descriptor id="shoesDesc" region="shoesReg"/>
27.      <descriptor id="formDesc" region="formReg"/>
28.     </descriptorBase>
29.     <connectorBase>
30.      <importBase documentURI="../causalConnBase.ncl" alias="conEx"/>
31.     </connectorBase>
32.   </head>
33.   <body>
34.    <port id="entry" component="animation"/>
35.    <media id="background" src="background.png"
                                          descriptor="backgroundDesc"/>
36.    <media id="animation" src="animGar.mp4" descriptor="screenDesc">
37.      <area id="segPhoto" begin="12s"/>
38.      <area id="segIcon" begin="45s" end="51s"/>
39.     </media>
40.     <media id="music" src="choro.mp3" descriptor="audioDesc"/>
41.     <media id="photo" src="photo.png" descriptor="photoDesc">
42.     <context id="advert">
```

```
43.       <media id="reusedAnimation" refer="animation" instance="instSame">
44.           <property name="bounds"/>
45.       </media>
46.       <media id="icon" src="icon.png" descriptor="iconDesc">
47.              <property name="left"/>
48.       </media>
49.       <media id="shoes" src="shoes.mp4" descriptor="shoesDesc"/>
50.       <media id="globalVar" type="application/x-ginga-settings">
51.           <property name="channel.interactivity" value="true"/>
52.       </media>
53.       <switch id="form">
54.           <switchPort id="spForm">
55.              <mapping component="enForm"/>
56.              <mapping component="ptForm"/>
57.           </switchPort>
58.           <bindRule constituent="enForm" rule="en"/>
59.           <defaultComponent component="ptForm"/>
60.           <media id="ptForm" src="ptForm.htm" descriptor="formDesc"/>
61.           <media id="enForm" src="enForm.htm" descriptor="formDesc"/>
62.        </switch>
63.        <link  id="lIcon" xconnector="conEx#onBeginTestStartSetDelay">
64.          <bind role="onBegin" component="reusedAnimation"
                                                interface="segIcon"/>
65.           <bind role="testVar" component="globalVar"
                                      interface="channel.interactivity">
66.              <bindParam name="var" value="true"/>
67.          </bind>
68.          <bind role="start" component="icon"/>
69.          <bind role="set" component="icon" interface="left">
70.              <bindParam name="var" value="87.5%"/>
71.              <bindParam name="delay" value="1s"/>
72.              <bindParam name="duration" value="5s"/>
73           </bind>
74.       </link>
75.       <link  id="lBeginShoes" xconnector="conEx#onKeySelectionStopSetStart">
76.          <bind role="onSelection" component="icon">
77.             <bindParam name="keyCode" value="RED"/>
78.          </bind>
79.          <bind role="start" component="shoes"/>
80.          <bind role="start" component="form" interface="spForm"/>
81.          <bind role="set" component="reusedAnimation" interface="bounds">
82.             <bindParam name="varSet" value="5%,6.67%,45%,45%"/>
83.          </bind>
84.          <bind role="stop" component="icon"/>
85.       </link>
86.       <link  id="lEndShoes" xconnector="conEx#onEndSetStop">
87.          <bind role="onEnd" component="shoes"/>
88.          <bind role="set" component="reusedAnimation" interface="bounds">
89.             <bindParam name="varSet" value="0,0,222.22%,222.22%"/>
90.          </bind>
91.         <bind role="stop" component="form"/>
```

```
92.       </link>
93.     </context>
94.     <link  id="lMusic" xconnector="conEx#onBeginStart">
95.       <bind role="onBegin" component="animation"/>
96.       <bind role="start" component="music"/>
97.     </link>
98.     <link id="lBackground" xconnector="conEx#onBeginStart">
99.       <bind role="onBegin" component="animation"/>
100.      <bind role="start" component="background"/>
101.    </link>
102.    <link id="lPhoto" xconnector="conEx#onBeginStart">
103.      <bind role="onBegin" component="animation" interface="segPhoto"/>
104.      <bind role="start" component="photo"/>
105.    </link>
106.    <link id="lEnd" xconnector="conEx#onEndStop">
107.      <bind role="onEnd" component="animation"/>
108.      <bind role="stop" component="background"/>
109.    </link>
110.    <link id="lEndIcon" xconnector="conEx#onEndStop">
111.      <bind role="onEnd" component="animation" interface="segIcon"/>
112.      <bind role="stop" component="icon"/>
113.    </link>
114.  </body>
115. </ncl>
```

Figura 4.25 - Aplicação NCL exemplo.

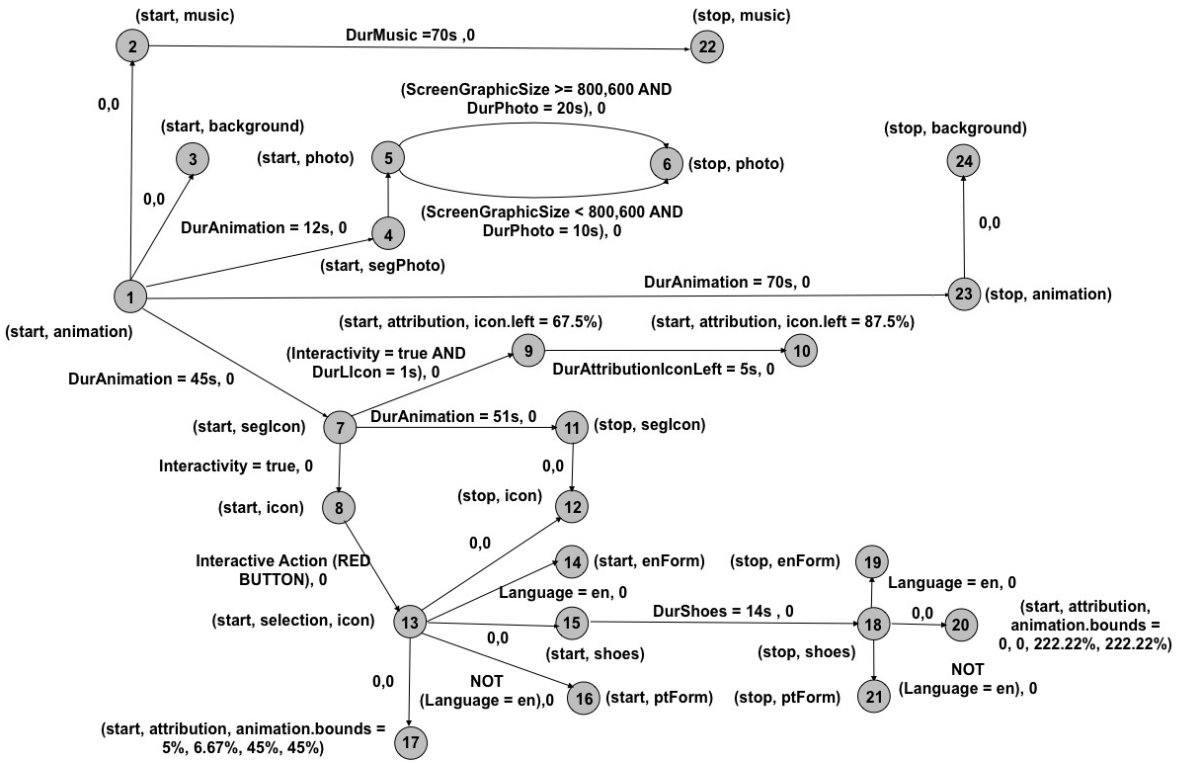


Figura 4.26 - HTG correspondente à aplicação NCL da Figura 4.25.

# 5
# Planos para o Controle do Sincronismo Temporal

Neste capítulo são apresentadas as estruturas de dados, construídas com base no HTG, que permitem calcular as especificações temporais necessárias ao controle do sincronismo das apresentações hipermídia. Com base nas cadeias temporais do HTG, as seguintes estruturas são propostas neste capítulo:

- Plano de apresentação;
- Plano de carregamento de exibidores;
- Plano de distribuição;
- Plano de pré-busca; e
- Plano de QoS.

A construção de cada plano e o auxílio que cada um pode oferecer à preservação da qualidade e à otimização dos recursos para a apresentação será discutida nas seções a seguir.

## 5.1.
## Planos para o Controle da Execução de uma Aplicação

São dois os planos para controle da execução de uma aplicação:

O *plano de apresentação* é proposto para orquestrar (escalonar) a apresentação de uma aplicação hipermídia. Através desse plano deve ser possível posicionar uma apresentação em um instante qualquer no tempo, retrocedendo a apresentação ou avançando até o ponto desejado de sua duração.

Como mencionado no Capítulo 1, em um STVDI pode ser possível que um usuário telespectador comece a assistir uma apresentação já iniciada. Nesse caso, o plano de apresentação pode ser necessário para posicionar a apresentação da aplicação no ponto atual da exibição do vídeo principal, preservando a semântica dessa aplicação.

Quando as aplicações de TV são posicionadas no instante atual da exibição do vídeo principal, eventos interativos que poderiam ter ocorrido no passado, como as interações dos telespectadores, devem ser considerados. Uma solução é

ignorar todas as interações que poderiam ter ocorrido antes do início da apresentação de uma aplicação. Essa solução pode ser adequada apenas para uma aplicação que está sendo apresentada pela primeira vez. Caso contrário, é desejável que as interações e as transições de eventos, já executadas em função dessas interações, sejam preservadas.

O plano de apresentação deve permitir um eficiente (com o mínimo de atraso possível) posicionamento inicial da apresentação, em qualquer momento da sua duração no tempo. O plano de apresentação deve preservar o histórico das transições que ocorreram durante uma apresentação. Dessa forma, quando uma apresentação for reiniciada, todas as opções escolhidas e as adaptações realizadas serão preservadas. Também como consequência, uma mesma apresentação poderia ser novamente realizada, de forma idêntica a primeira.

A apresentação das aplicações envolve a gerência dos exibidores. Usualmente, em um ambiente de apresentação estão disponíveis um conjunto de exibidores, cada um apropriado a um tipo específico de mídia. É importante que cada exibidor seja instanciado e esteja pronto para o uso antes que o instante exato no tempo em que um conteúdo precisa ser apresentado seja alcançado.

No ambiente de apresentação pode não ser possível que todos os exibidores permaneçam instanciados, em razão, principalmente, da limitação de memória. Nesse caso, pode ser necessário instanciar os exibidores específicos a cada início de um evento de apresentação. No entanto, as mesmas limitações que impedem os exibidores de permanecerem instanciados podem introduzir atrasos indesejáveis na instanciação de um determinado exibidor. Esses atrasos podem levar à perda da sincronização entre as mídias de uma apresentação, caso um *plano para o carregamento dos exibidores* não seja construído para prever essa situação.

Nas aplicações que contêm apenas transições de eventos previsíveis, as informações necessárias para construção dos planos de apresentação e de carregamento de exibidores podem ser completamente computadas, antes do início da apresentação. Por outro lado, nas aplicações contendo transições de eventos imprevisíveis, a construção desses planos somente pode ser realizada de forma progressiva, simultaneamente à apresentação. Em ambos os casos, o HTG, através das suas cadeias temporais, é necessário na construção desses planos, como será apresentado no exemplo a seguir.

No capítulo anterior, foi apresentada uma aplicação NCL (Figura 4.25) onde uma animação (“*animation*”) e uma música (“*music*”) eram executadas em paralelo. Após 12 segundos do início dessa animação, uma foto real do jogador (“*photo*”), ator principal retratado na animação, era apresentada na tela. A duração da apresentação dessa imagem variava de acordo com a configuração da tela. Eram 20 segundos de apresentação caso essa tela tivesse uma resolução mínima e 10 segundos caso contrário. Após 45 segundos do início dessa animação, a imagem de uma chuteira (“*icon*”) poderia ser apresentada, mas apenas se ações interativas fossem permitidas. A apresentação dessa imagem era encerrada quando a duração da animação atingia 51 segundos. Durante o seu período de exibição, a posição dessa imagem variava na tela.

Na aplicação proposta, o usuário poderia selecionar a imagem da chuteira pressionando o botão vermelho do controle remoto e, como resultado, sua apresentação seria encerrada, o vídeo da animação redimensionado, uma propaganda (“*shoes*”) e um formulário (“*form*”) seriam exibidos na tela, todos sobre uma imagem de fundo (“*background*”). Esse formulário, na verdade, poderia ser uma aplicação em português (“*ptForm*”) ou outra aplicação em inglês (“*enForm*”), dependendo do idioma do usuário. O final da apresentação do vídeo da propaganda provocava o término da apresentação do formulário e da imagem de fundo, além de redimencionar a animação ao seu tamanho original.

Também no capítulo anterior, foi construído o HTG (Figura 4.26) que representa a aplicação descrita nos parágrafos anteriores. As cadeias temporais desse HTG são apresentadas nas Figuras 5.1 e 5.2.

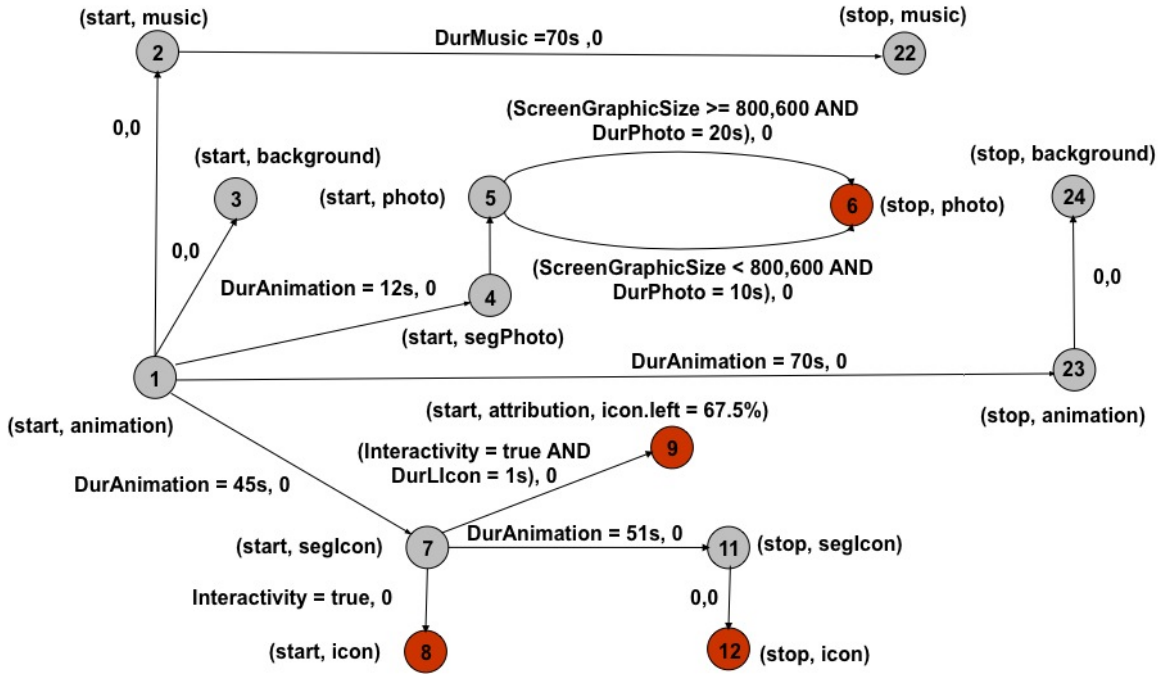


Figura 5.1 - Cadeia temporal principal do HTG construído no Capítulo 4.

A cadeia principal, representada na Figura 5.1, contém quatro transições de eventos imprevisíveis, representadas pelos vértices 6, 8, 9 e 12. A execução da transição representada pelo vértice 6 depende das características do ambiente de apresentação (resolução gráfica).[17] A execução das transições representadas pelos vértices 8 e 9 dependem da interatividade ser permitida, outra característica do ambiente de apresentação. A execução da transição representada pelo vértice 12 somente irá ocorrer caso uma transição de início, relativa a esse mesmo evento e objeto, aconteça. Como essa transição de início é imprevisível na cadeia principal (vértice 8), a transição representada pelo vértice 12 também será. Nas Figuras 5.1 e 5.2, os vértices sorvedouros, que iniciam outras cadeias, estão destacados em vermelho.

[17]O vértice 6 pode ser alcançado por duas arestas distintas que, nesse caso, têm condições complementares: quando uma for falsa a outra, necessariamente, será verdadeira. As propriedades do ambiente de apresentação, no entanto, devem ser avaliadas nesse ambiente.

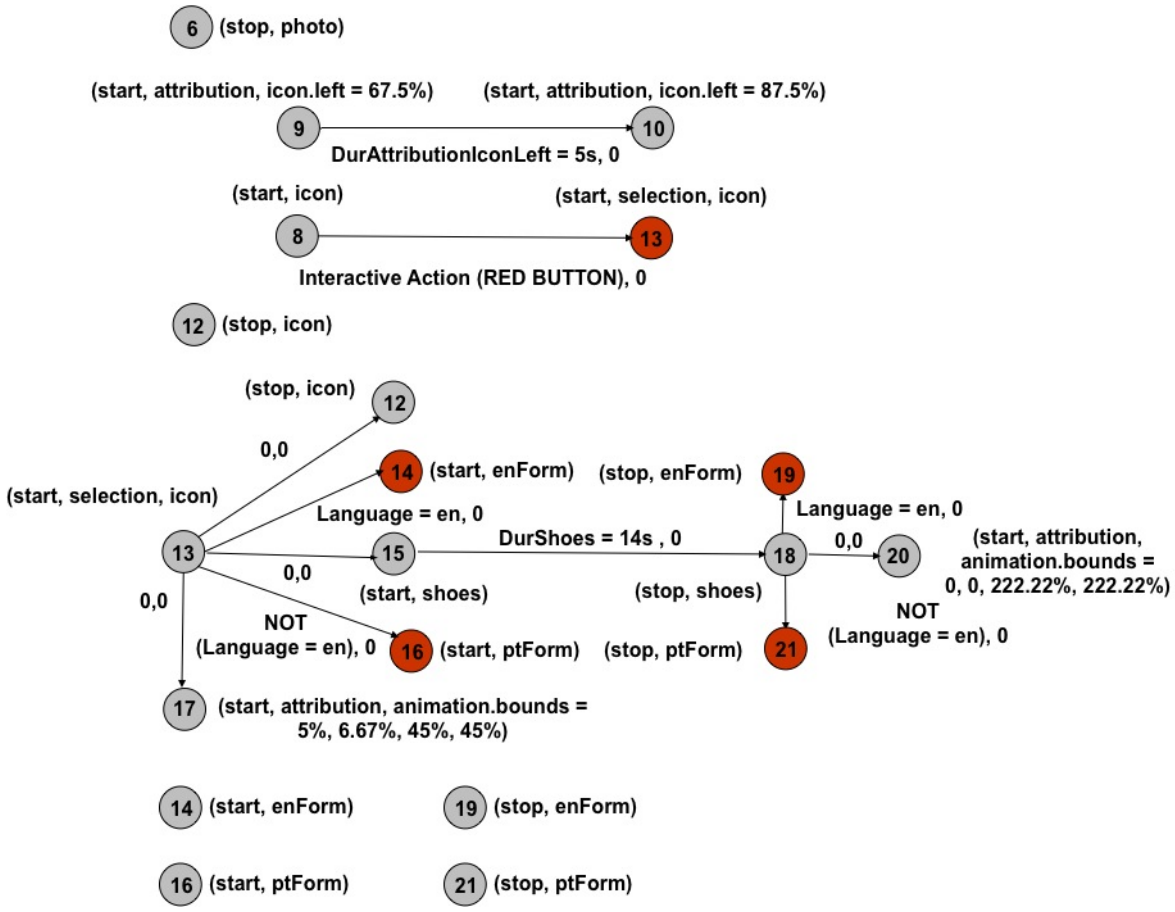


Figura 5.2 - Cadeias temporais secundárias do HTG construído no Capítulo 4.

Na Figura 5.2, as cadeias secundárias iniciadas a partir dos vértices 6 (início da apresentação da foto do jogador), 8 (início da apresentação da imagem da chuteira), 9 (início da atribuição da propriedade "*left*" da imagem da chuteira) e 12 (fim da apresentação da imagem da chuteira) são apresentadas. O vértice 13 representa o início da seleção da imagem da chuteira e é origem de outra cadeia secundária. Os vértices 14 e 16 correspondem, cada um, ao início da apresentação de cada objeto componente de um *switch*. O fim da apresentação de cada um desses objetos é representado através dos vértices 19 e 21.

Antes do início da apresentação, para cada cadeia, os instantes temporais para a execução das transições representadas por cada um dos seus vértices devem ser computados. Esses instantes são obtidos através do caminhamento nessas cadeias, a partir do vértice definido como ponto de início da aplicação, no caso da cadeia principal, ou, no caso das cadeias secundárias, a partir do vértice que representa a transição de evento imprevisível origem dessa cadeia.

A Tabela 5.1 contém as especificações temporais obtidas no caminhamento na cadeia principal, apresentada na Figura 5.1.

| Vértice | Transição | Evento | Objeto | Valor | Instante |
|---|---|---|---|---|---|
| 1 | *start* | *presentation* | *animation* | - | 0s |
| 2 | *start* | *presentation* | *music* | - | 0s |
| 3 | *start* | *presentation* | *background* | - | 0s |
| 4 | *start* | *presentation* | *segPhoto* | - | 12s |
| 5 | *start* | *presentation* | *photo* | - | 12s |
| 7 | *start* | *presentation* | *segIcon* | - | 45s |
| 11 | *stop* | *presentation* | *segIcon* | - | 51s |
| 22 | *stop* | *presentation* | *music* | - | 70s |
| 23 | *stop* | *presentation* | *animation* | - | 70s |
| 24 | *stop* | *presentation* | *background* | - | 70s |

Tabela 5.1 - Especificação temporal da cadeia principal.

As Tabelas 5.2 a 5.10 contêm as especificações temporais obtidas no caminhamento nas cadeias secundárias, apresentadas na Figura 5.2.

| Vértice | Transição | Evento | Objeto | Valor | Instante |
|---|---|---|---|---|---|
| 6 | *stop* | *presentation* | *photo* | - | 0s |

Tabela 5.2 - Especificação temporal da cadeia iniciada no Vértice 6.

| Vértice | Transição | Evento | Objeto | Valor | Instante |
|---|---|---|---|---|---|
| 8 | *start* | *presentation* | *icon* | - | 0s |

Tabela 5.3 - Especificação temporal da cadeia iniciada no Vértice 8.

| Vértice | Transição | Evento | Objeto | Valor | Instante |
|---|---|---|---|---|---|
| 9 | *start* | *attribution* | *icon.left* | 67.5% | 0s |
| 10 | *start* | *attribution* | *icon.left* | 87.5% | 5s |

Tabela 5.4 - Especificação temporal da cadeia iniciada no Vértice 9.

| Vértice | Transição | Evento | Objeto | Valor | Instante |
|---|---|---|---|---|---|
| 12 | *stop* | *presentation* | *icon* | - | 0s |

Tabela 5.5 - Especificação temporal da cadeia iniciada no Vértice 12.

| Vértice | Transição | Evento | Objeto | Valor | Instante |
|---|---|---|---|---|---|
| 13 | *start* | *selection* | *icon* | - | 0s |
| 12 | *stop* | *presentation* | *icon* | - | 0s |
| 15 | *start* | *presentation* | *shoes* | - | 0s |
| 16 | *start* | *presentation* | *ptForm* | - | 0s |
| 17 | *start* | *attribution* | *animation. bounds* | 5%, 6.67%, 45%, 45% | 0s |
| 18 | *stop* | *presentation* | *shoes* | - | 14s |
| 20 | *start* | *attribution* | *animation. bounds* | 0, 0, 222.22%, 222.22% | 14s |
| 21 | *stop* | *presentation* | *ptForm* | - | 14s |

Tabela 5.6 - Especificação temporal da cadeia iniciada no Vértice 13.

| Vértice | Transição | Evento | Objeto | Valor | Instante |
|---|---|---|---|---|---|
| 14 | *start* | *presentation* | *enForm* | - | 0s |

Tabela 5.7 - Especificação temporal da cadeia iniciada no Vértice 14.

| Vértice | Transição | Evento | Objeto | Valor | Instante |
|---|---|---|---|---|---|
| 16 | *start* | *presentation* | *ptForm* | - | 0s |

Tabela 5.8 - Especificação temporal da cadeia iniciada no vértice 16.

| Vértice | Transição | Evento | Objeto | Valor | Instante |
|---|---|---|---|---|---|
| 19 | *stop* | *presentation* | *enForm* | - | 0s |

Tabela 5.9 - Especificação temporal da cadeia iniciada no vértice 19.

| Vértice | Transição | Evento | Objeto | Valor | Instante |
|---|---|---|---|---|---|
| 21 | *stop* | *presentation* | *ptForm* | - | 0s |

Tabela 5.10 - Especificação temporal da cadeia iniciada no vértice 21.

O plano de apresentação é construído através de cópias dos vértices e arestas das cadeias temporais e dos instantes previstos para execução das transições de eventos representadas por esses vértices. Antes do início da apresentação, esse plano é formado apenas pelas especificações associadas à cadeia principal. A Tabela 5.11 apresenta a especificação temporal do plano de apresentação, imediatamente antes do início da apresentação.

| Vértice | Transição | Evento | Objeto | Valor | Instante |
|---|---|---|---|---|---|
| 1 | *start* | *presentation* | *animation* | - | 0s |
| 2 | *start* | *presentation* | *music* | - | 0s |
| 3 | *start* | *presentation* | *background* | - | 0s |
| 4 | *start* | *presentation* | *segPhoto* | - | 12s |
| 5 | *start* | *presentation* | *photo* | - | 12s |
| 7 | *start* | *presentation* | *segIcon* | - | 45s |
| 11 | *stop* | *presentation* | *segIcon* | - | 51s |
| 22 | *stop* | *presentation* | *music* | - | 70s |
| 23 | *stop* | *presentation* | *animation* | - | 70s |
| 24 | *stop* | *presentation* | *background* | - | 70s |

Tabela 5.11 - Especificação temporal inicial do plano de apresentação.

Durante uma apresentação, o plano de apresentação deve ser atualizado, caso uma transição de evento imprevisível, origem de uma cadeia, ocorra. Antes, porém, as especificações temporais computadas para essa cadeia devem ser atualizadas. Nessa atualização deve ser adicionado, aos instantes temporais

representados pelos seus vértices, o instante temporal de ocorrência da transição de evento, inicialmente imprevisível.

Quando o instante de 12 segundos do início da apresentação da animação é alcançado na aplicação exemplo, a resolução gráfica do ambiente de apresentação deve ser avaliada ("*ScreenGraphicSize*"). Essa avaliação é condição das arestas entre os vértices 5 e 6 da cadeia principal. Nesse exemplo foi considerado que a resolução gráfica tem o tamanho mínimo desejado e, portanto, a duração da apresentação da foto do jogador será de 20 segundos. Assim, o plano de apresentação deve ser atualizado através da inclusão do vértice 6 (cadeia secundária) com o seu instante temporal, previsto para a sua ocorrência, atualizado. A Tabela 5.12 apresenta o plano de apresentação com essa atualização. Nesse plano, a modificação realizada foi destacada em cinza.

| Vértice | Transição | Evento | Objeto | Valor | Instante |
|---|---|---|---|---|---|
| 1 | *start* | *presentation* | *animation* | - | 0s |
| 2 | *start* | *presentation* | *music* | - | 0s |
| 3 | *start* | *presentation* | *background* | - | 0s |
| 4 | *start* | *presentation* | *segPhoto* | - | 12s |
| 5 | *start* | *presentation* | *photo* | - | 12s |
| 6 | *stop* | *presentation* | *photo* | - | 32s |
| 7 | *start* | *presentation* | *segIcon* | - | 45s |
| 11 | *stop* | *presentation* | *segIcon* | - | 51s |
| 22 | *stop* | *presentation* | *music* | - | 70s |
| 23 | *stop* | *presentation* | *animation* | - | 70s |
| 24 | *stop* | *presentation* | *background* | - | 70s |

Tabela 5.12 - Especificação temporal do plano de apresentação modificado (1).

Continuando a descrição da construção do plano de apresentação, quando o instante de 45 segundos do início da apresentação da animação é atingido, o plano de apresentação poderá ser novamente atualizado. Nesse instante, a permissão para a ocorrência de ações interativas deve ser avaliada no ambiente de apresentação ("*Interactivity*"). Essa variável faz parte da condição associada às arestas entre os vértices 7 e 8 e entre os vértices 7 e 9 da cadeia principal. Nesse exemplo foi considerado que a interatividade é permitida. Portanto, os vértices das duas cadeias secundárias com origem nos vértices 8 e 9 serão adicionados ao plano de apresentação. A transição representada pelo vértice 8, único da sua cadeia, terá o seu instante previsto para ocorrência adicionado de 45 segundos. Nos vértices da cadeia secundária com origem no vértice 9, 46 segundos serão

adicionados às especificações temporais das suas transições. Nesse caso, existe uma condição temporal especificada na aresta entre os vértices 7 e 9 (1 segundo). A Tabela 5.13 apresenta novamente o plano de apresentação, com as novas atualizações destacadas.

| Vértice | Transição | Evento | Objeto | Valor | Instantes Temporais |
|---|---|---|---|---|---|
| 1 | *start* | *presentation* | *animation* | - | 0s |
| 2 | *start* | *presentation* | *music* | - | 0s |
| 3 | *start* | *presentation* | *background* | - | 0s |
| 4 | *start* | *presentation* | *segPhoto* | - | 12s |
| 5 | *start* | *presentation* | *photo* | - | 12s |
| 6 | *stop* | *presentation* | *photo* | - | 32s |
| 7 | *start* | *presentation* | *segIcon* | - | 45s |
| 8 | *start* | *presentation* | *icon* | - | 45s |
| 9 | *start* | *attribution* | *icon.left* | 67.5% | 46s |
| 10 | *start* | *attribution* | *icon.left* | 87.5% | 51s |
| 11 | *stop* | *presentation* | *segIcon* | - | 51s |
| 12 | *stop* | *presentation* | *icon* | - | 51s |
| 22 | *stop* | *presentation* | *music* | - | 70s |
| 23 | *stop* | *presentation* | *animation* | - | 70s |
| 24 | *stop* | *presentation* | *background* | - | 70s |

Tabela 5.13 - Especificação temporal do plano de apresentação modificado (2).

Continuando com a apresentação da aplicação utilizada como exemplo, considerando que a ação interativa prevista nessa aplicação aconteça no instante de 51 segundos, a partir do início da apresentação da animação, os vértices da cadeia iniciada no vértice 13 devem ser adicionados ao plano de apresentação. Novamente, os vértices dessa cadeia precisam ter os seus instantes atualizados, acrescidos de 51 segundos. A Tabela 5.14 apresenta o plano de apresentação atualizado, com destaque para as últimas atualizações.

| Vértice | Transição | Evento | Objeto | Valor | Instantes Temporais |
|---|---|---|---|---|---|
| 1 | *start* | *presentation* | *animation* | - | 0s |
| 2 | *start* | *presentation* | *music* | - | 0s |
| 3 | *start* | *presentation* | *background* | - | 0s |
| 4 | *start* | *presentation* | *segPhoto* | - | 12s |
| 5 | *start* | *presentation* | *photo* | - | 12s |
| 6 | *stop* | *presentation* | *photo* | - | 32s |
| 7 | *start* | *presentation* | *segIcon* | - | 45s |
| 8 | *start* | *presentation* | *icon* | - | 45s |
| 9 | *start* | *attribution* | *icon.left* | 67.5% | 46s |

| | | | | | |
|---|---|---|---|---|---|
| 13 | *start* | *selection* | *icon* | - | 51s |
| 12 | *stop* | *presentation* | *icon* | - | 51s |
| 15 | *start* | *presentation* | *shoes* | - | 51s |
| 17 | *start* | *attribution* | *animation. bounds* | 5%, 6.67%, 45%, 45% | 51s |
| 10 | *start* | *attribution* | *icon.left* | 87.5% | 51s |
| 11 | *stop* | *presentation* | *segIcon* | - | 51s |
| 18 | *stop* | *presentation* | *shoes* | - | 65s |
| 20 | *start* | *attribution* | *animation. bounds* | 0, 0, 222.22%, 222.22% | 65s |
| 22 | *stop* | *presentation* | *music* | - | 70s |
| 23 | *stop* | *presentation* | *animation* | - | 70s |
| 24 | *stop* | *presentation* | *background* | - | 70s |

Tabela 5.14 - Especificação temporal do plano de apresentação modificado (3).

Imediatamente após a inserção do vértice 13 no plano de apresentação, o idioma do usuário deve ser avaliado no ambiente de apresentação ("*Language*"). Essa avaliação é condição das arestas entre os vértices 13 e 14 e entre os vértices 13 e 16 da cadeia secundária, que foi inserida no plano de apresentação. Considerando que o idioma definido nesse exemplo é o português, o plano de apresentação deve ser novamente atualizado, através da inclusão do vértice 16. Antes, porém, o instante temporal de ocorrência da transição desse evento deve ser atualizado (51 segundos). A Tabela 5.15 apresenta o plano de apresentação com essa atualização em destaque.

| **Vértice** | **Transição** | **Evento** | **Objeto** | **Valor** | **Instantes Temporais** |
|---|---|---|---|---|---|
| 1 | *start* | *presentation* | *animation* | - | 0s |
| 2 | *start* | *presentation* | *music* | - | 0s |
| 3 | *start* | *presentation* | *background* | - | 0s |
| 4 | *start* | *presentation* | *segPhoto* | - | 12s |
| 5 | *start* | *presentation* | *photo* | - | 12s |
| 6 | *stop* | *presentation* | *photo* | - | 32s |
| 7 | *start* | *presentation* | *segIcon* | - | 45s |
| 8 | *start* | *presentation* | *icon* | - | 45s |
| 9 | *start* | *attribution* | *icon.left* | 67.5% | 46s |
| 13 | *start* | *selection* | *icon* | - | 51s |
| 12 | *stop* | *presentation* | *icon* | - | 51s |
| 15 | *start* | *presentation* | *shoes* | - | 51s |
| 16 | *start* | *presentation* | *ptForm* | - | 51s |
| 17 | *start* | *attribution* | *animation. bounds* | 5%, 6.67%, 45%, 45% | 51s |
| 10 | *start* | *attribution* | *icon.left* | 87.5% | 51s |

| | | | | | |
|---|---|---|---|---|---|
| 11 | *stop* | *presentation* | *segIcon* | - | 51s |
| 18 | *stop* | *presentation* | *shoes* | - | 65s |
| 20 | *start* | *attribution* | *animation. bounds* | 0, 0, 222.22%, 222.22% | 65s |
| 22 | *stop* | *presentation* | *music* | - | 70s |
| 23 | *stop* | *presentation* | *animation* | - | 70s |
| 24 | *stop* | *presentation* | *background* | - | 70s |

Tabela 5.15 - Especificação temporal do plano de apresentação modificado (4).

Quando a apresentação atinge 51 segundos do seu início, a transição de evento imprevisível, representada pelo vértice 12, deve ser avaliada. Essa transição somente ocorreria caso, no instante da sua avaliação, a imagem da chuteira estivesse ocorrendo. A Tabela 5.16 resume as transições que podem ocorrer para um determinado estado (atual) de evento. As especificações dessa tabela foram obtidas da máquina de estado de evento da Figura 3.1 (Capítulo 3).

| **Estado (atual)** | **Transição (ação que executa)** | **Estado (final)** |
|---|---|---|
| *sleeping* | *start* | *occurring* |
| *occurring* | *stop* | *sleeping* |
| | *natural end* | |
| | *abort* | |
| | *pause* | *paused* |
| *paused* | *stop* | *sleeping* |
| | *abort* | |
| | *resume* | *occurring* |

Tabela 5.11 - Estados alcançáveis através da máquina de estado de eventos.

Quando o fim da apresentação da propaganda (“*shoes*”) é alcançado, o idioma do usuário deve ser novamente avaliado. Essa avaliação é condição das arestas entre os vértices 18 e 19 e entre os vértices 18 e 21 da cadeia secundária com origem no vértice 13, inserida no plano de apresentação. Nesse caso, o plano de apresentação deve ser atualizado através da inclusão da cadeia secundária que contém o vértice 21. Antes, porém, o instante temporal de ocorrência da transição desse evento deve ser atualizado (65 segundos). A Tabela 5.16 apresenta o plano de apresentação final para essa aplicação, construído paulatinamente nesta seção. A última atualização nesse plano é destacada nessa tabela.

| Vértice | Transição | Evento | Objeto | Valor | Instantes Temporais |
|---|---|---|---|---|---|
| 1 | *start* | *presentation* | *animation* | - | 0s |
| 2 | *start* | *presentation* | *music* | - | 0s |
| 3 | *start* | *presentation* | *background* | - | 0s |
| 4 | *start* | *presentation* | *segPhoto* | - | 12s |
| 5 | *start* | *presentation* | *photo* | - | 12s |
| 6 | *stop* | *presentation* | *photo* | - | 32s |
| 7 | *start* | *presentation* | *segIcon* | - | 45s |
| 8 | *start* | *presentation* | *icon* | - | 45s |
| 9 | *start* | *attribution* | *icon.left* | 67.5% | 46s |
| 13 | *start* | *selection* | *icon* | - | 51s |
| 12 | *stop* | *presentation* | *icon* | - | 51s |
| 15 | *start* | *presentation* | *shoes* | - | 51s |
| 16 | *start* | *presentation* | *ptForm* | - | 51s |
| 17 | *start* | *attribution* | *animation. bounds* | 5%, 6.67%, 45%, 45% | 51s |
| 10 | *start* | *attribution* | *icon.left* | 87.5% | 51s |
| 11 | *stop* | *presentation* | *segIcon* | - | 51s |
| 18 | *stop* | *presentation* | *shoes* | - | 65s |
| 20 | *start* | *attribution* | *animation. bounds* | 0, 0, 222.22%, 222.22% | 65s |
| 21 | *stop* | *presentation* | *ptForm* | - | 65s |
| 22 | *stop* | *presentation* | *music* | - | 70s |
| 23 | *stop* | *presentation* | *animation* | - | 70s |
| 24 | *stop* | *presentation* | *background* | - | 70s |

Tabela 5.16: Especificação temporal final do plano de apresentação.

Durante uma apresentação, transições de eventos imprevisíveis podem não ocorrer no instante de tempo em que suas ocorrências são possíveis. Nesse caso, as especificações temporais calculadas para uma cadeia, que dependem da ocorrência dessa transição, podem ser descartadas. É esse o caso da cadeia com início no vértice 12. Esse caso também poderia ocorrer, se a ação interativa especificada na aplicação NCL não ocorresse entre 45 e 51 segundos, período em que o evento de seleção sobre a imagem da chuteira poderia ser executado.

Como mencionado, além das especificações temporais, o plano de apresentação é formado pelos vértices e arestas das cadeias inseridas nesse plano. Assim, ao final de uma apresentação, esse plano corresponde ao HTG final dessa apresentação, de fato, sua cadeia temporal final. Esse HTG é diferente do HTG inicial, pois partes do HTG que não foram apresentadas devem ser removidas e partes que ocorreram mais de uma vez devem ser repetidas no HTG final. A

Figura 5.3 apresenta o plano de apresentação final para a aplicação NCL utilizada como exemplo.

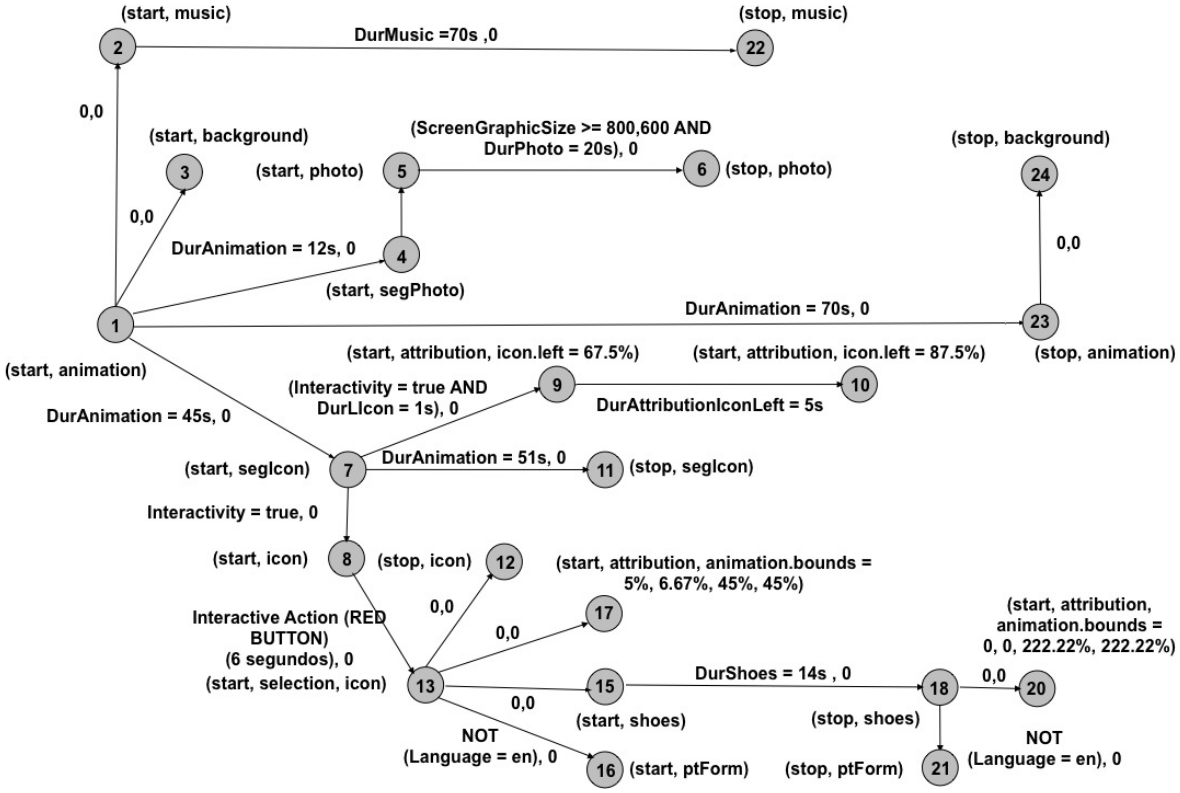


Figura 5.3 - HTG final para a especificação temporal descrita na Tabela 5.16.

Durante uma apresentação, os instantes temporais computados para os vértices do plano de apresentação podem sofrer alterações quando uma transição de evento, inicialmente imprevisível, ocorre. Esse é o caso, por exemplo, quando uma transição do tipo "*pause*", para o evento de apresentação, é adicionada a esse plano, em razão da ocorrência de uma transição de evento imprevisível. Nesse caso, é necessário modificar, no plano de apresentação, os instantes temporais previstos para a ocorrência de transições associadas ao evento de apresentação do objeto que teve sua apresentação pausada (transições de eventos não-determinísticas).

Analizando as cadeias construídas para representar uma aplicação, é possível identificar, *a priori*, se os vértices de uma cadeia, quando adicionados ao plano de apresentação, provocarão alterações sobre os instantes temporais computados para esse plano. Caso as cadeias contenham vértices que representam apenas transições de eventos executadas sobre objetos distintos ou que somente possam ser executadas em instantes temporais distintos, a única alteração no plano de apresentação, provocada por uma cadeia, é a inclusão de novos vértices.

Todos os esforços para preservar o plano de apresentação são aproveitados na construção do plano de carregamento de exibidores, que é formado apenas por especificações temporais calculadas sobre o plano de apresentação.

A partir do plano de apresentação, para a construção do plano de carregamento de exibidores devem ser selecionadas as transições relativas ao início ("*start*") e ao fim ("*stop*") do evento de apresentação sobre os objetos. As demais transições podem ser desconsideradas. A Tabela 5.17 apresenta um plano de carregamento de exibidores, calculado a partir do plano de apresentação da Tabela 5.16.

| Vértice | Transição | Evento | Objeto | Valor | Instantes Temporais |
|---|---|---|---|---|---|
| 1 | *start* | *presentation* | *animation* | - | 0s |
| 2 | *start* | *presentation* | *music* | - | 0s |
| 3 | *start* | *presentation* | *background* | - | 0s |
| 5 | *start* | *presentation* | *photo* | - | 12s |
| 6 | *stop* | *presentation* | *photo* | - | 32s |
| 8 | *start* | *presentation* | *icon* | - | 45s |
| 12 | *stop* | *presentation* | *icon* | - | 51s |
| 15 | *start* | *presentation* | *shoes* | - | 51s |
| 16 | *start* | *presentation* | *ptForm* | - | 51s |
| 18 | *stop* | *presentation* | *shoes* | - | 65s |
| 21 | *stop* | *presentation* | *ptForm* | - | 65s |
| 22 | *stop* | *presentation* | *music* | - | 70s |
| 23 | *stop* | *presentation* | *animation* | - | 70s |
| 24 | *stop* | *presentation* | *background* | - | 70s |

Tabela 5.17 - Especificação temporal final do plano de carregamento de exibidores para as cadeias das Figuras 5.1 e 5.2.

O plano de carregamento de exibidores deve ser construído para cada ambiente de apresentação específico, considerando os atrasos existentes em uma determinada plataforma. Assim, os instantes temporais para a ocorrência das transições de eventos da Tabela 5.17 devem ser modificados por ajustes, dependentes de uma plataforma específica.

Além das informações sobre o instante no tempo em que um exibidor deve ser instanciado, através do plano de carregamento de exibidores é possível estimar quantas vezes um determinado exibidor deve ser instanciado durante uma apresentação. Através dessa informação é possível, por exemplo, optar por manter esse exibidor carregado na memória durante toda a apresentação.

O plano de carregamento de exibidores também torna possível estimar quais exibidores serão necessários em um instante temporal específico de uma apresentação. Essa informação é importante para avaliar se os recursos disponíveis para a apresentação serão suficientes.

Dependendo da disponibilidade de recursos, o plano de carregamento de exibidores também pode ser construído considerando que as transições de eventos imprevisíveis sempre irão ocorrer. Essa estratégia permitiria instanciar, com antecedência, os exibidores responsáveis por objetos que deverão ser apresentados somente se um determinado evento imprevisível ocorrer. Na próxima seção, a construção de planos considerando a ocorrência de eventos imprevisíveis é discutida.

## 5.2. Planos para o Transporte e Carregamento do Conteúdo



Para preservar a qualidade de uma apresentação, é necessário que os conteúdos das mídias estejam disponíveis no ambiente de apresentação antes que os seus instantes previstos para apresentação sejam atingidos.

Apresentações distribuídas podem ter de ser iniciadas sem que todas as mídias especificadas nas aplicações estejam disponíveis no ambiente de apresentação. Como mencionado no Capítulo 1, esperar que todas as mídias sejam recebidas pode inserir um atraso desnecessário para o início de uma apresentação, além de exigir que a máquina de apresentação disponha de espaço suficiente para armazenar todo o conteúdo de uma aplicação.

As mídias necessárias a uma apresentação podem ser recebidas nos clientes sem que requisições individuais tenham sido realizadas (*pushed data*). Também é possível o recebimento das mídias mediante solicitações aos servidores (*pulled data*). A seguir serão discutidas as características de cada uma dessas opções, relacionadas ao controle do sincronismo das apresentações.

### 5.2.1. Controle da Distribuição do Conteúdo

Nos STVDI algumas mídias são enviadas através de fluxos sincronizados utilizando *timestamps* (sincronização *timeline*) (ISO/IEC, 2000b). Esse é o caso

do áudio e do vídeo principais, continuamente transmitidos nesses sistemas. Além do conteúdo transmitido de forma sincronizada, outros conteúdos (outros vídeos, outros áudios, imagens, textos etc.) também podem ser transmitidos de forma assíncrona, através de outros fluxos. Apesar da transmissão desses outros conteúdos ser assíncrona, é possível que eles sejam sincronizados entre si e também com o conteúdo audiovisual principal. A especificação desse sincronismo é normalmente definida através de uma aplicação que é, na verdade, mais um dos conteúdos assincronamente transmitidos.

No envio dos conteúdos, todos os fluxos são multiplexados e transportados até os receptores através de um canal específico (uma faixa de frequência no caso da TVD terrestre). Considerando que os receptores podem sintonizar um canal a qualquer momento, conteúdos transmitidos de forma assíncrona precisam ser enviados repetidamente, para garantir a recepção independente do momento de sintonização do canal. Para implementar essa forma de transmissão, o padrão MPEG-2 define uma estrutura cíclica de envio dos dados, o carrossel DSM-CC (ISO/IEC, 1998).

Um carrossel é uma estrutura de dados onde os conteúdos devem ser inseridos para serem repetidamente transmitidos. Um conteúdo qualquer pode ser inserido mais de uma vez em qualquer parte de um carrossel. O tamanho do carrossel, a sua taxa de transmissão e o espaço entre instâncias de um mesmo conteúdo colocados nessa estrutura são fatores que devem ser considerados no cálculo do atraso para a entrega de um conteúdo aos clientes.

Na transmissão do conteúdo de TV digital terrestre, a capacidade de um canal, limitada por uma faixa de frequência, é compartilhada por todos os fluxos, incluindo o conteúdo audiovisual principal e o conteúdo transmitido através do carrossel. Dessa forma, para aumentar a capacidade de transmissão destinada ao carrossel, seria preciso diminuir a capacidade de transmissão do áudio e vídeo principais. Essa opção não é usualmente considerada, pois pode requerer a diminuição da qualidade do conteúdo audiovisual principal. Em sistemas IPTV, a banda passante do canal de transmissão também limita a taxa de dados de envio do carrossel.

No ISDB-$T_B$, por exemplo, a largura de banda concedida às emissoras de TV é de 6MHz. A taxa da transmissão para essa largura de banda pode variar de acordo com parâmetros da modulação (ABNT, 2008), podendo atingir cerca de 19

Mbps (ABNT, 2008; Yamada, 2008). Grande parte dessa taxa é dedicada ao conteúdo audiovisual principal. Em (Pessoa, 2008), após a análise de fluxos de transporte (ISO/IEC, 2000b) de várias emissoras, foi constatado que, no mínimo, 14 Mbps são utilizados para esse conteúdo, sobrando, no máximo, da ordem de 5 Mbps para o transporte do carrossel.

Como exemplo, considerando a taxa para o transporte do carrossel como 5 Mbps, um carrossel com tamanho de 6 MB demoraria quase 10 segundos para ser completamente transmitido (6MB / 625KB/s). A taxa destinada ao carrossel, no entanto, pode ser menor, aumentado o tempo para a entrega do seu conteúdo.

Para minimizar o atraso do conteúdo transmitido de forma assíncrona, é ideal que o tamanho do carrossel seja o menor possível. No entanto, a construção de um carrossel é um problema que envolve muitas condições a serem otimizadas. Se o tamanho do carrossel for definido, esse problema pode ser reduzido ao problema da Mochila Booleano (0/1) (Garey, 1979). Nesse problema, devem ser colocados uma quantidade de itens (conteúdos), que podem ser inseridos mais de uma vez, em uma mochila com um peso máximo (tamanho do carrossel), de forma a maximizar o valor da mochila (rapidez na entrega). Apesar da complexidade desse problema, algumas heurísticas podem ser empregadas na sua resolução (Garey, 1979; Pessoa, 2008). Para a implementação dessas heurísticas, os instantes no tempo em que os conteúdos são necessários às apresentações precisam ser conhecidos, considerando que o valor da mochila corresponde ao menor atraso possível para a entrega desses conteúdos.

Mesmo que a distribuição dos conteúdos em um carrossel seja manualmente realizada, é necessário conhecer os instantes no tempo em que os conteúdos são necessários às apresentações. Como discutido no Capítulo 2, em alguns STVDI, é passado ao autor/produtora da aplicação a responsabilidade da gerência do carrossel. O autor/produtora, que conhece toda a aplicação, incluindo os instantes em os conteúdos são necessários, constrói um conjunto de pequenas aplicações que juntas equivalem à aplicação “global”. Cada uma dessas pequenas aplicações é transmitida no carrossel. Nos clientes, essas aplicações são iniciadas através de eventos de sincronismo (ISO/IEC, 1998), cujo disparo também precisa ser controlado pelos autores/produtores.

O objetivo desta tese, em relação ao controle da distribuição do conteúdo, é permitir que os instantes no tempo em que os conteúdos são necessários às

apresentações possam ser calculados. Para alcançar esse objetivo, um plano de distribuição é proposto. Esse plano é construído nos servidores considerando que todas as transições de eventos imprevisíveis irão ocorrer no menor instante no tempo em que suas ocorrências são possíveis. Todas as condições no HTG que dependem das características o ambiente de apresentação ou dos usuários também são consideradas como verdadeiras.

Somente os instantes temporais relativos às transições de início ("*start*") de um evento de apresentação fazem parte do plano de distribuição. São essas transições que indicam que um conteúdo deve estar presente nos clientes. A Tabela 5.18 contém as especificações temporais do plano de distribuição, construído a partir das cadeias temporais apresentadas nas Figura 5.1 e 5.2.

| Vértice | Transição | Evento | Objeto | Valor | Instantes Temporais |
|---|---|---|---|---|---|
| 1 | *start* | *presentation* | *animation* | - | 0s |
| 2 | *start* | *presentation* | *music* | - | 0s |
| 3 | *start* | *presentation* | *background* | - | 0s |
| 5 | *start* | *presentation* | *photo* | - | 12s |
| 8 | *start* | *presentation* | *icon* | - | 45s |
| 14 | *start* | *presentation* | *enForm* | - | 45s |
| 15 | *start* | *presentation* | *shoes* | - | 45s |
| 16 | *start* | *presentation* | *ptForm* | - | 45s |

Tabela 5.18 - Especificação do plano de distribuição para as cadeias das Figuras 5.1 e 5.2.

Com base nas informações do plano de distribuição, o servidor de conteúdo pode estimar quais objetos devem ser colocados no carrossel, quantas vezes, e em quais posições, bem como determinar quais objetos podem ser removidos do carrossel. Como mencionado, embora esse seja um problema de difícil otimização, as informações básicas para a sua resolução estão disponíveis no plano de distribuição. A partir das informações da Tabela 5.18, por exemplo, um carrossel pode ser formado por várias repetições da especificação da aplicação (NCL) e das mídias necessárias ao início da sua apresentação. Nesse exemplo, a distância entre cada especificação da NCL e da imagem da foto do jogador ("*photo*") não deve possuir um atraso para o recebimento nos clientes superior a 12 segundos. Essa mesma observação deve ser seguida em relação às demais mídias da apresentação, considerando um atraso de 45 segundos.

### 5.2.2.
### Controle do Carregamento do Conteúdo

Para que os conteúdos especificados em uma aplicação possam ser recebidos simultaneamente à sua apresentação, devem ser conhecidos não apenas os instantes previstos para a apresentação de cada conteúdo, mas também as características do canal de comunicação utilizado para o transporte.

Os instantes previstos para a apresentação de cada conteúdo podem ser calculados através do plano de apresentação. Como cada tipo de canal de comunicação usualmente possui diferentes características (vazão, latência, variação do atraso etc.), um plano para o controle do recebimento dos conteúdos precisa ser construído para cada tipo de canal. Entre todas as características de um canal, a possibilidade de negociação de QoS pode influenciar decisivamente na construção dos planos, como será discutido a seguir.

Nos canais de comunicação com provisão de QoS, requisitos como a vazão, o retardo máximo, entre outros, podem ser garantidos através de protocolos empregados na negociação e reserva dos recursos desejados. Esses recursos devem ser negociados para cada conteúdo necessário em uma apresentação (QoS intramídia) (Rodrigues, 2003b). Para armazenar os parâmetros necessários a essa negociação, um plano específico deve ser construído, chamado plano de QoS.

Na construção do plano de QoS, todas as transições de eventos imprevisíveis de uma aplicação devem ser tratadas como se fossem previsíveis. Apenas os instantes temporais relativos às transições de início (“*start*”) do evento de apresentação devem fazer parte desse plano. Na especificação temporal desse plano, devem ser considerados, para cada objeto que tem sua apresentação prevista, o atraso e a variação estatística do atraso a serem negociados no canal de comunicação.

Os procedimentos para a negociação da QoS devem ser implementados para um canal específico. Caso essa negociação seja bem sucedida, é garantido que um objeto de mídia será recebido no instante temporal necessário a sua apresentação. Caso contrário, se esses parâmetros não puderem ser atendidos, eles precisarão ser relaxados. Mesmo nesse outro caso, a possibilidade de negociação de QoS aumenta as chances do sincronismo temporal das aplicações ser obedecido.

Nos canais que não oferecem provisão de QoS, um outro tipo de plano deve ser construído. Esse outro plano, baseado em estimativas e cenários, é chamado de plano de pré-busca. Na sua construção devem ser considerados apenas os instantes temporais relativos às transições de início ("*start*") do evento de apresentação e, sobre esses instantes, devem ser considerados o atraso e a variação estatística do atraso, estimados no canal de comunicação específico.

Como o plano de pré-busca é baseado em estimativas, ele apenas diminui a possibilidade da ocorrência de falhas no recebimento do conteúdo pelos clientes. Para tentar minimizar essas falhas, um cenário conservador pode ser adotado. Nesse cenário, além de considerar o pior caso possível para o atraso e a sua variação, é considerado que todas as transições de eventos imprevisíveis irão ocorrer no menor instante no tempo em que suas ocorrências são possíveis.

Adotar um cenário conservador pode evitar a ocorrência de falhas temporais no recebimento dos conteúdos, mas pode ter um custo para os receptores, principalmente em relação ao armazenamento prévio dos conteúdos. O caso extremo desse cenário consiste em aguardar o recebimento de todos os conteúdos para que uma apresentação seja iniciada. Assim, a implementação da pré-busca dos conteúdos não é uma tarefa fácil, mesmo a partir das informações do plano proposto nesta tese.



## 5.3. Controle do Sincronismo Temporal no Ginga-NCL para TV Terrestre

A parte declarativa do *middleware* Ginga oferece suporte à execução de aplicações NCL e a aplicações especificadas na linguagem LUA,[18] através de uma implementação que segue uma abordagem modular (Moreno, 2009). Alguns desses módulos, diretamente relacionados ao controle do sincronismo temporal, são apresentados na Figura 5.4. A especificação e construção desses módulos fazem parte do trabalho proposto em (Moreno, 2010). Assim, para uma descrição completa desses módulos, a autor interessado deve consultar essa referência.

[18] http://www.lua.org


Figura 5.4 - Arquitetura Ginga: módulos relacionados ao controle do sincronismo no Ginga-NCL.

Como pode ser observado na descrição arquitetural da Figura 5.4, o Ginga é dividido em dois subsistemas. O Núcleo Ginga, um desses subsistemas, é responsável por oferecer serviços à Máquina de Apresentação Ginga-NCL, que é o outro subsistema existente, responsável por iniciar e controlar as aplicações NCL.

No Núcleo Ginga, o módulo Sintonizador é responsável por receber o conteúdo de TV transmitido pelos provedores de conteúdo. Esse conteúdo é um fluxo de transporte MPEG (ISO/IEC, 2000b), que é demultiplexado pelo módulo Processador de Dados. No fluxo de transporte, além do conteúdo audiovisual principal, estão os conteúdos assíncronos, incluindo os eventos de sincronismo (comandos de edição) e o fluxo NPT (*Normal Play Time*) (ISO/IEC, 1998).

Um fluxo NPT é formado por estruturas de dados denominadas descritores de referência NPT (*NPT Reference Descriptor*) (ISO/IEC, 1998), que implementam valores temporais contextualizados. Um descritor NPT contém um campo *contentId*, identificando a qual conteúdo de um fluxo esse descritor se refere, e um outro campo, em que pode ser deduzido um valor no tempo para o fluxo identificado.

Os fluxos elementares de mídia contínua têm no NPT a sua referência temporal. Assim, o NPT é utilizado para o controle do sincronismo das apresentações. Em uma aplicação NCL, por exemplo, se um elemento <*media*> faz referência ao conteúdo de vídeo principal, as especificações temporais sobre esse objeto, como as suas âncoras temporais, devem ter como referência os valores do NPT.

Na recepção do conteúdo, os descritores NPT são constantemente monitorados pelo módulo Processador de Dados. A mudança do valor do campo *contentId* de um descritor NPT para um outro consecutivo indica a interrupção da apresentação de um conteúdo para a exibição de outro. Essa interrupção pode ser temporária ou permanente. Em ambos os casos, o módulo Escalonador, da Máquina de Apresentação do Ginga-NCL, deve ser notificado para que a apresentação de todos os objetos de mídia associados ao *contentId* sejam pausados (Moreno, 2008).

No primeiro caso (interrupção temporária), quando a base temporal é retomada (quando voltar a receber o descritor NPT com o mesmo *contentId*), a apresentação da aplicação também é retomada. Pode ser necessário, no entanto, sincronizar a apresentação no momento atual do descritor NPT. Esse caso também pode acontecer quando o usuário começa a assistir uma apresentação já iniciada (sintoniza um canal).

No segundo caso (interrupção permanente), quando a base temporal não é retomada em um determinado intervalo máximo de tempo (Moreno, 2008), o ciclo de vida da aplicação chega ao fim. Nesse caso, todos os eventos relacionados ao término da apresentação devem ser disparados pelo Escalonador.

O Núcleo Ginga possui outros módulos relacionados ao controle do sincronismo, apresentados na Figura 5.4. O módulo Transporte é responsável por controlar protocolos e interfaces de rede, atendendo a demanda por conteúdos. O módulo Gerente de Contexto preserva as informações sobre as características da máquina de apresentação e sobre o perfil do usuário telespectador. O módulo Exibidores contém um conjunto de exibidores, cada um adequado a apresentação de um conteúdo específico.

Na Máquina de Apresentação NCL, a orquestração da apresentação é realizada pelo módulo Escalonador. Para que uma apresentação possa ser iniciada, esse módulo precisa receber uma notificação, enviada pelo módulo Gerenciador de Bases Privadas. A função do Gerenciador de Bases Privadas é controlar o local de armazenamento (base privada) das aplicações. Além disso, é esse componente que recebe os comandos de edição (Costa, 2006), que serão abordados no próximo capítulo. Apenas para exemplificar, quando o comando "*startDocument*" é recebido, a apresentação é iniciada.

Durante uma apresentação, quando um conteúdo precisa ser exibido, o Escalonador solicita ao Gerente de Adaptadores que instancie o exibidor apropriado ao tipo de conteúdo a ser exibido. Quando a exibição do conteúdo é concluída, o Escalonador é notificado. Nesse momento, o Escalonador decide se o adaptador deve permanecer instanciado ou se a instância do exibidor (adaptador) deve ser destruída.

Os módulos do Ginga-NCL são implementados através de componentes, que podem ser individualmente carregados (Moreno, 2010). Além das vantagens relacionadas à performance (Moreno, 2010), essa abordagem facilita a incorporação de novas funcionalidades ao *middleware*. Entre essas funcionalidades estão os planos temporais propostos neste Capítulo para o uso nos clientes (apresentação, carregamento de exibidores, pré-busca e QoS). Os componentes associados à construção dos planos temporais são representados na Figura 5.5.

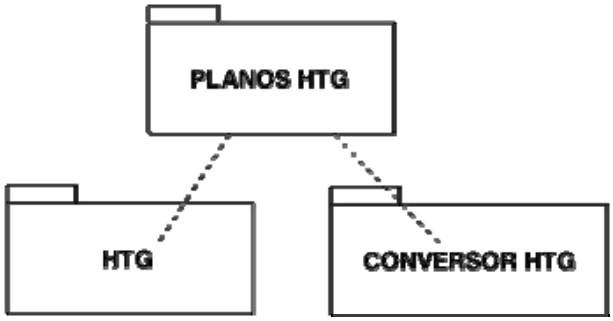


Figura 5.5 - Diagrama de pacotes referente aos componentes do HTG.

O componente Planos HTG define a classe que constrói (classe *PlanManager*) cada um dos planos (classe *Plan*) para o controle do sincronismo. A estrutura do HTG e de suas cadeias é definida no componente HTG. Quem constrói um HTG, a partir de uma especificação NCL, são as classes existentes no componente Conversor HTG.

A estrutura do HTG e das suas cadeias é implementada através de uma lista de adjacências. Nessa representação, “n” listas encadeadas são construídas, uma para cada vértice do grafo. Dado um vértice “i” qualquer, os elementos da sua lista representam os seus vértices adjacentes. Os elementos dessas listas, além das informações do vértice, contêm as condições associadas às arestas e também as suas prioridades.

As classes para a tradução das especificações NCL para o HTG foram construídas através do XML Schema do perfil EDTV da NCL 3.0. Essa construção foi realizada através do *framework* proposto em (Soares, 2006b), que facilita a tradução das especificações NCL para uma outra sintaxe qualquer (Soares, 2006b). As classes obtidas através desse *framework* contêm pontos de flexibilização, que devem ser tratados de acordo com a sintaxe de destino. Esses pontos de flexibilização são formados por métodos abstratos, que foram sobrecarregados por outros métodos, onde a tradução específica para o HTG foi implementada. Essa implementação foi realizada obedecendo às especificações para a construção do HTG apresentadas no Capítulo 4.

Cada componente apresentado na Figura 5.5 poderia ser individualmente carregado no Ginga-NCL. No entanto, como o componente Planos HTG depende tanto das estruturas do componente HTG quando do conversor implementado no componente Conversor HTG, todos esses componentes precisam estar carregados para que os planos sejam construídos. Essa relação de dependência é especificada na Figura 5.6. Essa figura contém a descrição do componente Planos HTG, através da sintaxe padronizada para o carregamento de componentes no Ginga-NCL (Moreno, 2010).

```
01. <?xml version="1.0" encoding="UTF-8"?>
02. <component package="htg-plans" name="libhtgplans.so" version="0.10.1">
03.    <dependency name="libtelemidiautil.so" version="0.10.1"/>
04.    <dependency name="libhtg.so" version="0.10.1"/>
05.    <dependency name="libhtgconverter.so" version="0.10.1"/>
06.    <symbol object="PlanManager" creator="createPlanManager"
                    destroyer="destroyPlanManager" interface="IPlanManager"/>
07.    <location type="local" uri="/usr/local/lib/ginga/" />
08. </component>
...
```

Figura 5.6 - Descrição do componente Planos HTG.

Antes do componente Escalonador receber a notificação para iniciar uma apresentação, uma notificação é realizada pelo Gerenciador de Bases Privadas para o Conversor, outro módulo da Máquina de Apresentação Ginga-NCL, indicando que a especificação de uma aplicação (NCL) foi recebida. É o componente Conversor que constrói a estrutura interna do controle da apresentação, definida originalmente em (Rodrigues, 2003b).

A mesma notificação enviada ao módulo Conversor também é recebida pelo componente Planos HTG. Essa notificação é repassada ao componente Conversor HTG, para que o HTG, correspondente à aplicação NCL, seja construído. Após a construção do HTG, o componente Planos HTG realiza a construção das cadeias temporais e dos planos de apresentação, de carregamento de exibidores e de pré-busca (QoS ainda não é oferecido). Nesse processo, se for necessário, o componente Gerente de Contexto é consultado sobre os valores de variáveis associadas ao ambiente de apresentação ou ao contexto do usuário.

A estrutura construída pelo módulo Conversor e os planos construídos pelo componente Planos HTG são ambos necessários ao controle da apresentação. A estrutura construída pelo módulo Conversor contém informações espaciais que não são representadas através dos planos temporais. Na verdade, através da estrutura construída pelo módulo Conversor, uma apresentação poderia ser completamente realizada. No entanto, o controle da ocorrência das transições dos eventos, da disponibilidade dos conteúdos e dos exibidores somente podem ser realizados através dos planos temporais, como mencionado diversas vezes nesta tese.

Para a construção dos planos, um processamento adicional é necessário no Ginga-NCL, que pode impor um atraso ao início da apresentação. Para ilustrar essa situação, a Figura 5.7 apresenta um gráfico comparativo do tempo necessário (em milisegundos no eixo vertical) para o carregamento de um conjunto de diferentes aplicações NCL, com geração do HTG e dos planos correspondentes às aplicações.

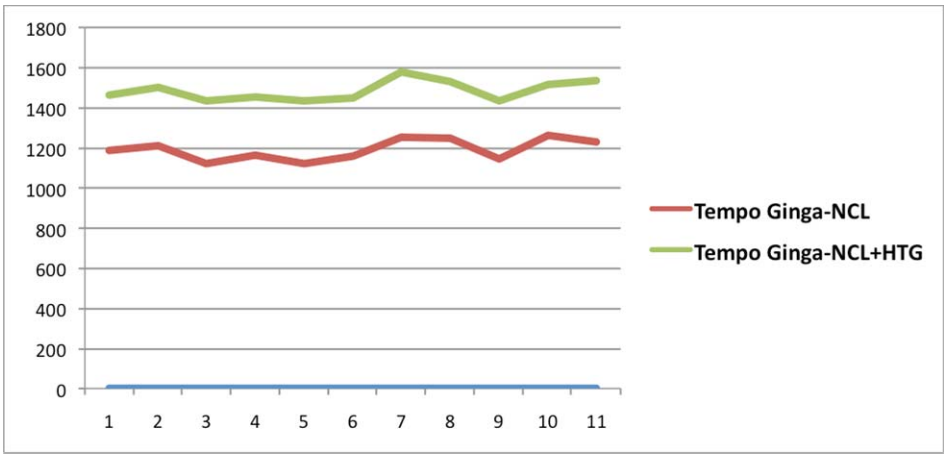


Figura 5.7 - Tempos para o carregamento do Ginga-NCL para o conjunto de aplicações da Tabela 5.19.

Cada linha da Figura 5.7 une os tempos necessários para o carregamento das aplicações listadas na Tabela 5.19 (eixo horizontal). Esse tempo inclui o tempo necessário ao carregamento dos componentes do Ginga-NCL e também as configurações necessárias à apresentação de uma aplicação, até o ponto imediatamente antes do seu início. Os tempos adicionais, necessários ao carregamento dos componentes para a construção dos planos temporais (Figura 5.5) e para a construção desses planos, para cada aplicação, são unidos através da linha verde do gráfico da Figura 5.7.

As medições do gráfico da Figura 5.7 foram realizadas com a versão 0.10.1 do Ginga-NCL, através de uma máquina virtual[19] limitada a 500MB de RAM.

| Índice | Aplicação | Vértices | Cadeias | Vértices na Cadeia Principal |
|---|---|---|---|---|
| 1 | Exemplo – Figura 4.25 | 34 | 10 | 14 |
| 2 | Programando NCL (01) | 14 | 1 | 14 |
| 3 | Programando NCL (02) | 24 | 2 | 19 |
| 4 | Programando NCL (03) | 25 | 2 | 19 |
| 5 | Programando NCL (04) | 27 | 2 | 21 |
| 6 | Programando NCL (05) | 29 | 2 | 21 |
| 7 | Programando NCL (06) | 38 | 4 | 21 |
| 8 | Programando NCL (07) | 61 | 9 | 29 |
| 9 | Programando NCL (08) | 61 | 9 | 29 |
| 10 | Programando NCL (09) | 62 | 9 | 30 |
| 11 | Programando NCL (10) | 134 | 14 | 51 |

Tabela 5.19 - Informações do HTG para um conjunto de aplicações.

A primeira aplicação utilizada na construção do gráfico da Figura 5.7 (Tabela 5.19) foi especificada no Capítulo 4 e pode ser visualizada na Figura 4.25. As demais aplicações foram retiradas da referência (Soares, 2009b). Para as aplicações da Tabela 5.19, o custo para o carregamento dos componentes adicionais e para a construção dos planos manteve-se praticamente constante. O maior custo observado foi de 303 milissegundos (exemplo 11).

Como contrapartida ao custo de processamento, os planos temporais oferecem o controle temporal das apresentações. É possível, por exemplo, posicionar uma apresentação em um instante qualquer no tempo de sua duração. Esse ajuste pode ser realizado de forma automática, de acordo com o valor do NPT. Nesse caso, o módulo Processador de Dados informa o instante no tempo

[19] Uma máquina virtual similar pode ser encontrada em http://www.softwarepublico.gov.br

em que a apresentação deve ser posicionada. A apresentação é posicionada no instante apropriado através de transições de evento executadas pelo módulo Escalonador. Cabe ao componente Planos HTG informar ao módulo Escalonador as transições de eventos que devem ser executadas para o instante de tempo considerado.

Também é possível que uma apresentação seja posicionada em um instante qualquer de sua duração de forma interativa: um recurso bastante útil durante a edição e a realização de testes de uma aplicação. A mesma interface que o componente Planos HTG oferece para que um instante no tempo da aplicação seja informado pelo módulo Processador de Dados, pode ser utilizada pelo usuário através de uma aplicação LUA residente. A Figura 5.8 ilustra essa possibilidade.


Figura 5.8 - Menu LUA do Ginga-NCL para acesso ao plano de apresentação.

No exemplo, a aplicação LUA construída para acessar as interfaces do componente Planos HTG tem seu início e fim associados ao evento de pausa da aplicação NCL. Assim, quando uma aplicação NCL é pausada, a aplicação LUA é iniciada na parte inferior da aplicação NCL, como pode ser observado na Figura 5.8. Essa aplicação LUA informa o instante no tempo atual da apresentação e permite ao usuário, através do controle remoto, escolher um novo instante no tempo para posicionar essa apresentação. Nesse caso, as mídias necessárias à apresentação, no instante temporal desejado, devem estar disponíveis no receptor.

Durante a apresentação, existe uma comunicação constante entre o módulo Escalonador e o componente Planos HTG. Toda ação imprevisível que acontece durante uma apresentação é informada pelo Escalonador ao componente Planos HTG, para que os planos de apresentação e carregamento de exibidores possam ser atualizados. Em contrapartida, o componente Planos HTG informa ao componente Escalonador as transições de evento que devem ser executadas para sincronizar uma apresentação em um instante no tempo qualquer e também disponibiliza a esse componente as especificações temporais para o carregamento de exibidores e para operações de pré-busca que forem necessárias a uma apresentação.

Outra característica interessante dos planos temporais que merece ser destacada é a sua representação visual. Para visualizar a estrutura do HTG, das cadeias e até mesmo do plano de apresentação, uma ferramenta gráfica foi implementada, a partir de uma ferramenta para desenho de grafos.[20] Essa ferramenta permite o desenho de vértices e de arestas com descrições. Ela foi estendida para permitir o desenho de vários grafos, com cores diferentes, para que as cadeias pudessem ser representadas.

O componente Planos HTG foi implementado com vários métodos para depuração. Esses métodos, quando habilitados, enviam informações relacionadas ao HTG para a ferramenta gráfica, onde essas informações são apresentadas. O componente Planos HTG deve solicitar uma conexão à ferramenta gráfica (servidor) e, a partir dessa conexão, enviar as informações associadas ao HTG. A Figura 5.8 apresenta, como exemplo, o HTG construído pelo componente Planos HTG (Conversor HTG) para uma aplicação NCL (Aplicação 3) da Tabela 5.19.

[20] http://abugraph.sourceforge.net

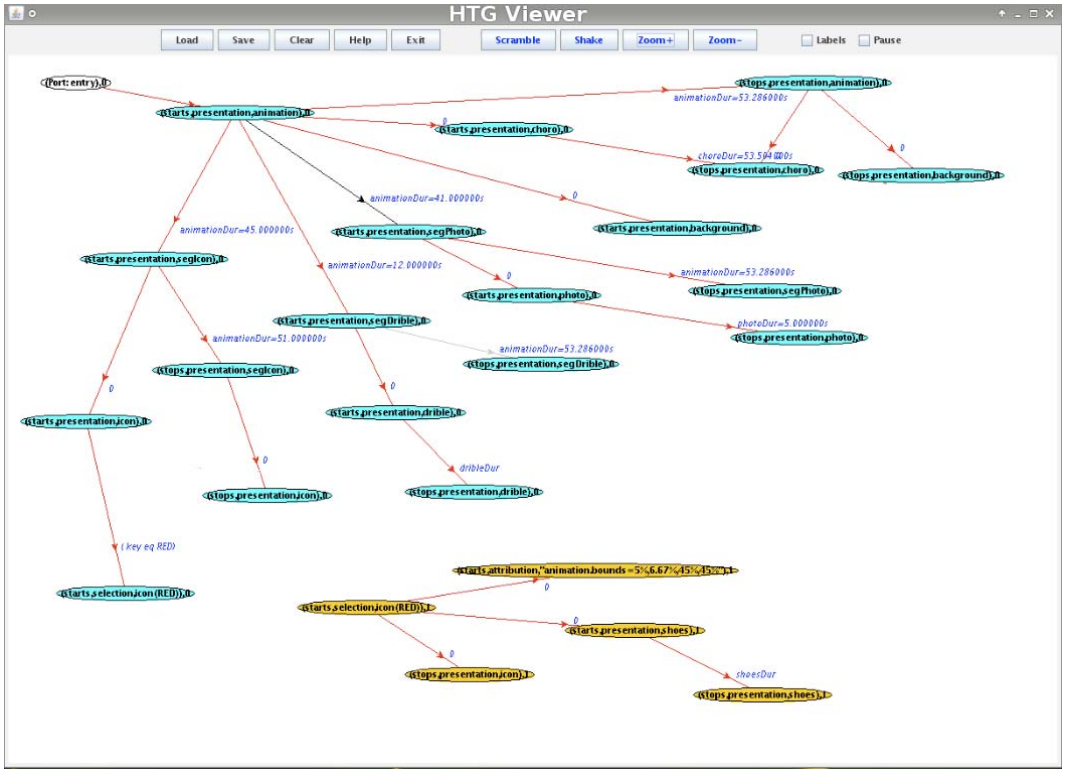


Figura 5.9 - Cadeias HTG construídas pelo componente Planos HTG para uma aplicação NCL (Aplicação 3 da Tabela 5.19).

# 6
# Edição e Distribuição de Apresentações Hipermídia

Para que uma apresentação possa ser atualizada durante a sua execução, uma sintaxe de transferência é proposta neste capítulo. Essa sintaxe é formada por comandos que, quando recebidos nos clientes, preservam tanto a especificação da aplicação quanto o HTG que representa o seu sincronismo.

O HTG, durante uma apresentação, também pode ter suas partes distribuídas para diferentes dispositivos, que passam a controlar, cada um, uma parte da apresentação original. Dessa forma, torna-se possível que vários usuários, de forma escalável, possam interagir através dos seus dispositivos individuais, sem interromper a apresentação coletiva. Na proposta desta tese, cada dispositivo pode redistribuir a apresentação que ele controla, definindo um verdadeiro modelo distribuído hierárquico, que será apresentado neste capítulo.

## 6.1.
## Edição Simultânea à Apresentação

Nos casos em que os conteúdos das aplicações não são conhecidos *a priori*, não é possível ao autor especificar completamente uma aplicação antes que a sua apresentação seja iniciada. Esse é o caso, por exemplo, das aplicações contendo conteúdos produzidos “ao vivo”, como foi exemplificado no Capítulo 1.

Nesta tese, uma sintaxe de transferência é proposta, formada por comandos de edição que podem ser enviados do servidor (ambiente de autoria da aplicação) aos clientes receptores e exibidores da aplicação, com o objetivo de modificar tanto a especificação dessa aplicação quanto o fluxo da sua apresentação. De fato, os comandos atuam sobre o HTG, modificando sua estrutura em tempo de exibição (Costa, 2006). Caso as modificações realizadas ao vivo sobre as apresentações não fossem registradas simultaneamente nas especificações das aplicações, anteriormente enviadas ao sistema de apresentação, uma posterior apresentação dessas aplicações não levaria em conta tais alterações.

### 6.1.1.
### Edição ao Vivo nas Aplicações NCL

Comandos para edição permitem inserir, remover ou editar as especificações de uma aplicação NCL. Em geral, esses comandos têm a seguinte sintaxe:

- add*Entity*(*baseId*, *documentId*, *xml*); e
- remove*Entity*(*baseId*, *documentId*, *entityId*).

Nesses comandos, a palavra "*Entity*" deve ser substituída pelo nome da entidade NCL (*Connector, Rule, Descriptor, Link* etc.) que o comando pretende incluir, remover ou editar. A aplicação NCL que será modificada é especificada no lugar do parâmetro "*documentId*". Além de especificar a aplicação, o comando de edição também precisa identificar a base privada onde essa aplicação está armazenada. Essa informação deve estar especificada no lugar do parâmetro "*baseId*".[21]

Caso um comando de edição seja utilizado para inserir ou editar uma entidade NCL, a especificação dessa entidade deve estar no lugar do parâmetro "*xml*". A especificação das entidades (sintaxe) deve obedecer ao XML Schema do perfil da NCL para comandos de edição (*NCL 3.0 Editing Commands Profile*) (ABNT, 2009). Um comando para adicionar uma entidade, que já existe na especificação da aplicação é utilizado, de fato, para atualizar (editar) essa entidade.

No caso dos comandos utilizados para remover entidades em uma aplicação NCL, o identificador único dessa entidade deve estar especificado no lugar do parâmetro "*entityId*".

A ordem de recepção dos comandos de edição nos clientes é importante para que a consistência de uma aplicação seja preservada. Como exemplo, caso um elo fosse adicionado a uma aplicação (*addLink*), antes da construção dos nós e das interfaces referenciadas por esse elo, um especificação inválida seria obtida. Para evitar esse problema, quando as entidades referenciadas por um comando de edição não estiverem presentes na aplicação, uma definição padrão para essas entidades deve ser construída. Posteriormente, essas entidades, construídas de forma padrão, poderão ser substituídas, através de outros comandos de edição (Costa, 2006).

[21] Como mencionado no capítulo anterior, as aplicações NCL recebidas nos clientes são preservadas em uma base privada (Moreno, 2009).

Também em relação à recepção dos comandos, se o autor quiser construir âncoras temporais que fazem menção ao instante no tempo em que o comando é recebido, o valor dessa âncora temporal deve fazer referência ao valor "*now*". Nos clientes, esse valor é substituído pelo instante no tempo em que o comando é recebido.

A especificação dos comandos de edição NCL pode ser encontrada no Apêndice B. Nesse apêndice encontra-se a definição completa desses comandos, incluindo o trabalho proposto por (Moreno, 2009). Nesse outro trabalho, esses comandos são estendidos para poderem identificar também os recursos (as estruturas de dados), especificados como parâmetros dos comandos, transmitidos externamente às estruturas de dados que carregam esses comandos. Nesse mesmo trabalho também é definido como o ciclo de vida das aplicações NCL pode ser controlado com base nos comandos de edição.



### 6.1.2. Edição ao Vivo no HTG

Um HTG pode ser completamente construído através dos comandos de edição. Também é possível, através desses comandos, remover ou editar uma parte de um HTG. Como mencionado no início deste capítulo, os comandos de edição são especificados em relação às entidades da NCL para que, além do HTG, a especificação de uma aplicação também permaneça atualizada.

Para inserir, remover ou editar vértices e arestas que representam um evento de apresentação em uma parte de um HTG, os seguintes comandos, descritos na Tabela 6.1, são propostos.

| Comandos | Descrição |
|---|---|
| addNode(baseId, documentId, compositeId, {uri, id}+) | Adiciona um nó NCL (<media>, <context> ou <switch>) a um nó de composição (*compositeId*) em uma aplicação NCL (*documentId*) localizada uma base privada (*baseId*) |
| removeNode(baseId, documentId, compositeId, Id) | Remove um nó NCL (<media>, <context> ou <switch>) (*Id*) de um nó de composição (*compositeId*) de uma aplicação NCL (*documentId*) em uma base privada (*baseId*) |

| | |
|---|---|
| addInterface(baseId, documentId, nodeId, xml) | Adiciona uma interface (<port>, <area>, <property> ou <switchPort>)(*xml*) a um nó (<media>, <body>, <context> ou <switch>) (*nodeId*) de uma aplicação NCL (*documentId*) em uma base privada (*baseId*) |
| removeInterface(baseId, documentId, nodeId, Id) | Remove uma interface (<port>, <area>, <property> ou <switchPort>)(*Id*) de um nó (<media>, <body>, <context> ou <switch>) (*nodeId*) de uma aplicação NCL (*documentId*) em uma base privada (*baseId*) |

Tabela 6.1 - Comandos para inserir, remover ou editar vértices e arestas que representam um evento de apresentação ou a adaptação do conteúdo.

Um comando *addNode*, quando especificado sobre um nó de mídia (<media>), pode provocar a inserção de dois vértices em um HTG, correspondentes ao início e ao fim do evento de apresentação desse nó. Caso essa mídia tenha uma duração, esses vértices devem estar unidos por uma aresta, que deve ter como condição essa duração, tão logo ela seja conhecida. Caso esse comando seja especificado sobre um nó de mídia já existente na aplicação, a condição da aresta que representa a sua duração pode ser modificada (editada), recebendo um novo valor. Também é possível que essa aresta seja removida, caso a duração da apresentação dessa mídia deixe de ser especificada. O comando *removeNode* é utilizado para remover esses vértices e a aresta entre eles.

Um comando *addInterface*, quando especificado sobre uma âncora de conteúdo (<area>) de um nó de mídia (<media>), pode provocar a inserção de dois vértices em um HTG, correspondentes ao início e ao fim do evento de apresentação dessa âncora. O vértice que representa essa transição de fim somente será construído caso exista uma duração explícita nessa âncora (especificada no parâmetro *xml*). Nesse caso, uma aresta entre esses vértices será construída, tendo como condição a duração da apresentação dessa âncora. Caso esse comando seja especificado sobre uma âncora já existente na aplicação, a condição da aresta que representa a sua duração pode ser modificada (editada) por um novo valor. Também é possível que essa aresta e o seu vértice destino sejam removidos, caso essa âncora temporal seja modificada, deixando de ter uma duração explícita. A remoção desses vértices e da aresta entre eles é realizada através do comando *removeInterface*.

O comando *addInterface*, quando especificado sobre uma âncora de conteúdo (<area>), também provoca no HTG a construção de uma aresta que representa o relacionamento entre o objeto de mídia e a sua âncora de conteúdo. O vértice que representa o início da apresentação desse objeto é a origem dessa aresta, que tem como destino o vértice que representa a transição de início da apresentação sobre a âncora especificada. Essa aresta é removida através do comando *removeInterface*, que remove também os vértices e arestas do HTG descritos no parágrafo anterior.

Os comandos *addNode*, *removeNode*, *addInterface* e *removeInterface* também podem ser utilizados para inserir, remover ou editar vértices e arestas de parte de um HTG que representam uma adaptação do conteúdo em uma aplicação. Nesse caso, um elemento <switch> é especificado como parâmetro dos comandos *addNode* ou *removeNode.* Através desses comandos são inseridos (*addNode*) ou removidos (*removeNode*) vértices no HTG correspondentes às transições de início e de fim da apresentação de cada componente filho do elemento <switch>. Se forem definidas interfaces específicas nesses componentes (*addInterface* e *removeInterface*), arestas representando os relacionamentos com a participação dessas interfaces são inseridas (*addInterface*) ou removidas (*removeInterface*) no HTG. A essas arestas devem estar associadas condições para a escolha do conteúdo entre as opções para adaptação. Os comandos de edição que permitem inserir, remover ou editar essas condições sobre as arestas são apresentados na Tabela 6.2.



| Comandos | Descrição |
|---|---|
| addRule<br>(baseId, documentId, xml) | Adiciona uma regra (<rule>)(*xml*) a uma aplicação NCL (*documentId*) em uma base privada (*baseId*) |
| removeRule<br>(baseId, documentId, Id) | Remove uma regra (<rule>)(*Id*) de uma aplicação NCL (*documentId*) em uma base privada (*baseId*) |
| addRuleBase<br>(baseId, documentId, xml) | Adiciona uma base de regras (<ruleBase>)(*xml*) a uma aplicação NCL (*documentId*) em uma base privada (*baseId*) |

| | |
|---|---|
| removeRuleBase (baseId, documentId, Id) | Remove uma base de regras (<ruleBase>)(*xml*) de uma aplicação NCL (*documentId*) em uma base privada (*baseId*) |

Tabela 6.2 - Comandos para inserir, remover ou editar condições nas arestas que representam a adaptação do conteúdo ou da apresentação.

Uma condição, associada a uma aresta do HTG, pode ser inserida ou modificada por um comando para adicionar uma regra (ou uma base de regras) (*addRule, addRuleBase*). Condições também pode ser removidas (*removeRule*, *removeRuleBase*). Além da adaptação do conteúdo, regras também são necessárias para inserir, remover ou editar condições nas arestas do HTG que representam a adaptação da apresentação de um objeto. Quando a apresentação de um objeto pode ser realizada com diferentes durações (explícitas), uma aresta deve ser construída para cada uma dessas durações. É a condição associada a essas arestas que define, em tempo de apresentação, qual será a duração da apresentação desse objeto.

Para inserir, remover ou editar cada uma das arestas que tem como condição diferentes durações explícitas, os comandos *addDescriptor* (inserir ou editar) e *removeDescriptor* (remover), especificados na Tabela 6.3, são propostos.

| Comandos | Descrição |
|---|---|
| addDescriptor (baseId, documentId, xml) | Adiciona um descritor (<descriptor>)(*xml*) a uma aplicação NCL (*documentId*) em uma base privada (*baseId*) |
| removeDescriptor (baseId, documentId, Id) | Remove um descritor (<descriptor>)(*Id*) de uma aplicação NCL (*documentId*) em uma base privada (*baseId*) |
| addDescriptorSwitch (baseId, documentId, xml) | Adiciona um *switch* de descritores (<descriptorSwitch>)(*xml*) a uma aplicação NCL (*documentId*) em uma base privada (*baseId*) |
| removeDescriptorSwitch (baseId, documentId, Id) | Remove um *switch* de descritores (<descriptorSwitch>)(*Id*) de uma aplicação NCL (*documentId*) em uma base privada (*baseId*) |
| addDescriptorBase(baseId, documentId, xml) | Adiciona uma base de descritores (<descriptorBase>)(*xml*) a uma aplicação NCL (*documentId*) em uma base privada (*baseId*) |

| | |
|---|---|
| removeDescriptorBase(baseId, documentId, Id) | Remove uma base de descritores (<descriptorBase>)(*Id*) de uma aplicação NCL (*documentId*) em uma base privada (*baseId*) |

Tabela 6.3 - Comandos para inserir, remover ou editar arestas que representam a adaptação da apresentação.

Descritores devem estar especificados em uma base (*addDescriptorBase*, *removeDescriptorBase*). Quando diferentes alternativas para apresentação são oferecidas (adaptação da apresentação), os descritores de cada uma dessas alternativas devem estar agrupados em um *switch* de descritores (*addDescriptorSwitch*, *removeDescriptorSwitch*).

As arestas do HTG que representam relacionamentos causais entre as transições de eventos (vértices) podem ser inseridas, removidas ou modificadas, através dos comandos da Tabela 6.4. Quando a ação de um relacionamento corresponde a uma transição de um evento de atribuição, um vértice, destino da aresta que representa esse relacionamento, também pode ser inserido, removido ou modificado, através dos comandos da Tabela 6.4.

| Comandos | Descrição |
|---|---|
| addConnector (baseId, documentId, xml) | Adiciona um conector (<connector>)(*xml*) a base de conectores de uma aplicação NCL (*documentId*) em uma base privada (*baseId*) |
| removeConnector (baseId, documentId, Id) | Remove um conector (<connector>)(*Id*) da base de conectores de uma aplicação NCL (*documentId*) em uma base privada (*baseId*) |
| addConnectorBase (baseId, documentId, xml) | Adiciona uma base de conectores (<connectorBase>)(*xml*) a uma aplicação NCL (*documentId*) em uma base privada (*baseId*) |
| removeConnectorBase (baseId, documentId, Id) | Remove uma base de conectores (<connectorBase>)(*Id*) de uma aplicação NCL (*documentId*) em uma base privada (*baseId*) |
| addLink(baseId, documentId, compositeId, xml) | Adiciona um elo (<link>)(*xml*) a um nó de composição NCL (<body>, <context> ou <switch>) em uma aplicação NCL (*documentId*) localizada em uma base privada (*baseId*) |

| | |
|---|---|
| removeLink(baseId, documentId, compositeId, Id) | Remove um elo (<link>)(*Id*) de um nó de composição NCL (<body>, <context> ou <switch>) de uma aplicação NCL (*documentId*) em uma base privada (*baseId*) |

Tabela 6.4 - Comandos para inserir, remover ou editar vértices e arestas que representam aos relacionamentos de uma apresentação.

Os comandos para adicionar ou modificar um conector (*addConnector* e *addConnectorBase*) e para adicionar ou modificar um elo (*addLink*) atuam em conjunto sobre o HTG. Quando esses comandos definem um relacionamento que tem uma condição e uma ação simples, uma única aresta é construída no HTG, unindo os vértices correspondentes à condição e à ação desse relacionamento. Caso o relacionamento especificado nos comandos de edição defina ações compostas, várias arestas são construídas no HTG, cada qual tendo como destino o vértice correspondente a cada ação desse relacionamento. Caso esse relacionamento defina condições compostas, condições do HTG serão associadas a cada aresta construída para representar esse relacionamento. A remoção dos vértices e arestas do HTG, que representam os relacionamentos mencionados, é ser realizada através dos comandos *removeConnector*, *removeConnectorBase* e, principalmente, *removeLink*.

## 6.2. Apresentação Distribuída através de Múltiplos Dispositivos

O uso de múltiplos dispositivos pode agregar várias vantagens à apresentação das aplicações hipermídia (Costa, 2009; Soares, 2009b). Em cenários com assistência coletiva, por exemplo, dispositivos individuais podem ser utilizados para evitar que uma apresentação seja interrompida em função das interações realizadas individualmente pelos usuários.

Além de evitar que a assistência coletiva seja prejudicada, o uso de múltiplos dispositivos pode permitir diversas navegações individuais. Nos STVDI, por exemplo, vários usuários telespectadores poderiam interagir simultaneamente, o que seria impossível utilizando apenas a TV e o controle remoto tradicional. Dependendo da aplicação, cada telespectador poderia escolher uma opção diferente, e visualizar, individualmente, o resultado da sua escolha.

As escolhas realizadas pelos usuários, através dos seus dispositivos, podem resultar em diferentes apresentações. Caso o controle dessas apresentações seja realizado de forma centralizada, o dispositivo que contém a especificação da aplicação, chamado, a partir deste ponto, de dispositivo base, precisa controlar cada interação realizada através de cada dispositivo individual. Nessa forma de controle, também pode ser necessário que o dispositivo base conheça as características de cada dispositivo, para que as adaptações especificadas nas aplicações possam ser realizadas. Assim, o controle centralizado das apresentações distribuídas pode exigir recursos (armazenamento, processamento, comunicação etc.) no dispositivo base que, em muitos cenários, incluindo o cenário formado por receptores em um STVDI, podem não estar disponíveis.

Caso os dispositivos individuais tenham a capacidade de executar uma apresentação, uma solução mais interessante, proposta nesta tese, consiste em distribuir, para cada dispositivo, o controle da sua própria apresentação (parte da apresentação "global"). Nesse caso, cabe ao dispositivo base controlar a distribuição da apresentação aos dispositivos individuais, como será discutido no exemplo a seguir.

A aplicação NCL especificada na Figura 4.25 (Capítulo 4) poderia ter sua apresentação realizada de forma distribuída. Nessa aplicação, uma animação ("*animation*") e uma música ("*music*") são executadas em paralelo, desde o ínicio da sua apresentação. Também nessa aplicação, após 45 segundos do início da apresentação da animação, a imagem de uma chuteira ("*icon*") pode ser apresentada, mas apenas se ações interativas forem permitidas. Caso o usuário selecione essa imagem, uma propaganda ("*shoes*") e um formulário ("*form*") são apresentados. Para que essas outras mídias possam ser apresentadas na tela, único dispositivo de apresentação no exemplo original, a animação precisa ser redimensionada, como resultado da seleção da imagem da chuteira.

Caso cada usuário tivesse seu próprio dispositivo, a apresentação descrita no parágrafo anterior poderia ser executada sem que a assistência coletiva fosse prejudicada, como acontece, por exemplo, no redimensionamento do filme da animação. Nesse caso, o filme da propaganda e o formulário para aquisição do produto poderiam ser apresentados somente nos dispositivos dos usuários que realizaram a ação interativa. A Figura 6.1 ilustra esse cenário, onde a apresentação da aplicação NCL, escolhida como exemplo, é realizada de forma distribuída.

Figura 6.1 - Múltiplos dispositivos para exibição.

No cenário proposto na Figura 6.1, a imagem da chuteira é apresentada em todos os dispositivos individuais. Se essa imagem for selecionada em um dispositivo específico, terá início à apresentação, nesse mesmo dispositivo, do vídeo relativo à propaganda da chuteira e do formulário. Na Figura 6.1, dois usuários realizaram essa seleção. Através dos seus dispositivos, esses usuários podem fazer escolhas individuais em paralelo, o que seria impossível com apenas um único dispositivo. Nesse exemplo, um dos usuários já finalizou sua compra, enquanto o outro ainda está avaliando o produto.

O controle da apresentação ilustrada na Figura 6.1 pode ser realizado tanto de forma centralizada, no dispositivo base, quanto de forma distribuída, em cada dispositivo individual. No controle centralizado, no entanto, além dos requisitos já mencionados nesta seção, para a apresentação do exemplo ilustrado na Figura 6.1, o dispositivo base precisaria controlar todas as informações individuais dos usuários necessárias para a compra de cada produto (modelo, tamanho, forma de pagamento etc.).

Independente da forma como a apresentação é controlada (centralizada ou distribuída), para que ela possa ser realizada de forma distribuída, na especificação da aplicação devem ser definidos quais objetos de mídia deverão ser apresentados nos dispositivos individuais. Essa especificação pode ser realizada pelo autor ou ser definida de forma automática, como proposto em (Cesar, 2007). Em ambos os casos, é desejável que essa especificação possa ser realizada de forma adaptável. Dessa forma, uma apresentação pode ser realizada de forma distribuída apenas se dispositivos individuais estiverem disponíveis no ambiente de apresentação (Costa, 2009; Soares, 2009b).

Distribuir para cada dispositivo, o controle da sua própria apresentação, requer a fragmentação da especificação de uma aplicação durante à sua apresentação. Nesta tese, o HTG é proposto como a estrutura de dados que oferece suporte a essa fragmentação, que é realizada selecionando, no HTG da aplicação, os vértices que representam transições de eventos sobre objetos que devem ser apresentados nos dispositivos individuais. As arestas com origem e destino sobre os vértices selecionados também devem ser selecionadas para a construção desse novo HTG, parte do "HTG original".

Voltando ao exemplo ilustrado na Figura 6.1, considerando que nessa aplicação foi especificado que a imagem da chuteira ("*icon*"), o filme da propaganda ("*shoes*") e o formulário ("*enForm*" e "*ptForm*") deverão ser apresentados através dos dispositivos individuais disponíveis, o HTG, apresentado na Figura 6.2, é construído para o controle dessa apresentação nos dispositivos. Esse HTG foi obtido com base no "HTG original" dessa aplicação, apresentado no Capítulo 4 (Figura 4.26).

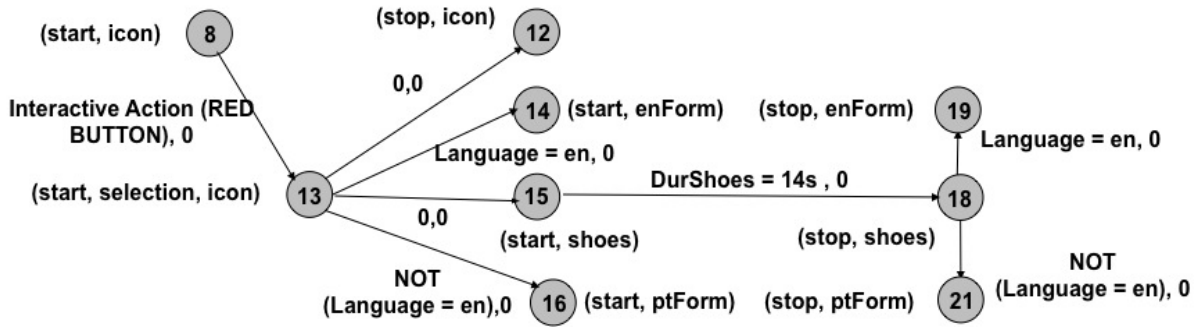


Figura 6.2 - HTG para controle da apresentação nos dispositivos individuais.

No HTG da Figura 6.2 são representadas todas as transições de eventos e os seus relacionamentos necessários à execução da apresentação nos dispositivos individuais da Figura 6.1. Na Figura 6.2, o ponto de início do HTG é o vértice 8 (início da apresentação da imagem da chuteira). Seguindo as especificações desse HTG, a imagem da chuteira é apresentada e pode ser selecionada. Essa seleção, por sua vez, finaliza a apresentação dessa imagem e inicia a apresentação do formulário e da propaganda. Em cada dispositivo, um formulário, em um idioma diferente (português ou inglês), pode ser apresentado, dependendo das características do usuário.

Durante a apresentação, o dispositivo base deve enviar, aos dispositivos individuais, todos os conteúdos das mídias que podem ser apresentados nesses

dispositivos. Também devem ser enviados o HTG para o controle da apresentação (Figura 6.2) e as especificações espaciais necessárias à apresentação dos conteúdos (Costa, 2009).

Caso existam relacionamentos entre os objetos apresentados nos dispositivos individuais, e os demais objetos, apresentados no dispositivo base, eventos devem ser gerados para essa comunicação (Costa, 2009). No exemplo da Figura 6.1, um evento para iniciar a apresentação da imagem da chuteira (vértice 8) deve ser enviado pelo dispositivo base aos dispositivos individuais. O mesmo acontece para terminar a apresentação dessa imagem (vértice 12).[22] Esses relacionamentos podem ser identificados no HTG da aplicação, através das arestas com origem e destino em vértices sobre objetos apresentados em dispositivos de diferentes classes, que são definidas a seguir.

Um classe de dispositivos define um conjunto de dispositivos que sempre executam uma mesma parte de uma apresentação. Esse é o caso, por exemplo, dos dispositivos individuais da Figura 6.1. Classes onde os dispositivos controlam suas próprias apresentações são definidas como “ativas”.[23] Diversos dispositivos podem fazer parte de uma mesma classe e inúmeras classes podem ser definidas, dependendo da distribuição desejada para uma apresentação.

Dispositivos em uma mesma classe “ativa” visualizam, normalmente, a mesma apresentação apenas no seu início. Caso transições de eventos interativos e adaptações estejam presentes em uma apresentação, esses pontos de imprevisibilidade poderão levar a apresentações completamente diferentes em cada dispositivo. Esse é o caso, por exemplo, das apresentações realizadas nos dispositivos individuais da Figura 6.1, onde um usuário já finalizou a sua compra, enquanto o outro, através do seu dispositivo, ainda está analisando o produto.

Em alguns casos, no entanto, pode ser desejável sincronizar os dispositivos que fazem parte de uma mesma classe, para que eles estejam visualizando a mesma apresentação em um determinado instante da duração dessa apresentação. Esse caso pode acontecer quando, por exemplo, usuários compartilham o mesmo dispositivo e, a partir de um determinado instante, cada um desses usuários deseja

[22] Nesse exemplo, a apresentação da imagem da chuteira pode ser interrompida caso o usuário realize uma ação interativa. Assim, um evento de termino da apresentação da imagem da chuteira recebido posteriormente seria desprezado, considerando os estados possíveis para um evento (Figura 3.1).

[23] Esse termo é proposto para diferenciar os dispositivos que apenas reproduzem informações (“passivos”), sem controlar, de fato, a apresentação.

usar o seu próprio dispositivo. Nesse caso, um dispositivo, ao se registrar em uma classe, pode receber o plano de apresentação construído em um outro dispositivo, dessa mesma classe. Dessa forma, ambos os dispositivos terão suas apresentações posicionadas no mesmo instante no tempo da apresentação. A partir desse ponto, os usuários desses dispositivos podem interagir de forma independente, podendo obter, como resultado, diferentes apresentações individuais.

Cada dispositivo em uma classe "ativa" pode redistribuir a apresentação que ele controla. Na aplicação ilustrada na Figura 6.1, por exemplo, um grupo de usuários poderia desejar usar um outro dispositivo, com uma tela maior, para visualizar o vídeo da propaganda. Nesse exemplo, ilustrado na Figura 6.3, o preenchimento do formulário para a compra do produto continuaria sendo realizado através dos dispositivos individuais.

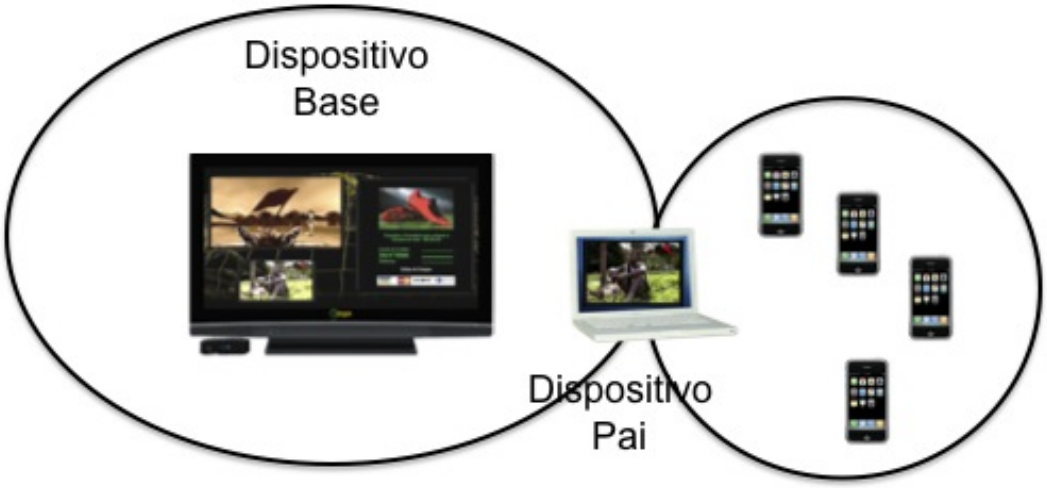


Figura 6.3 - Distribuição de dispositivos em domínios.

Na Figura 6.3, dois domínios são representados. Cada domínio define uma organização hierárquica, onde um dispositivo controla a distribuição da apresentação para os demais integrantes desse domínio. O dispositivo que realiza esse controle é chamado de dispositivo pai. O dispositivo base é um dispositivo pai que controla a distribuição da apresentação completa.

No exemplo da Figura 6.3, parte da apresentação é distribuída para ser executada em um computador. Essa parte, é a mesma apresentada nos dispositivos individuais da Figura 6.1. Porém, nesse exemplo, o formulário será apresentado em outros dispositivos. No computador será apresentada a imagem da chuteira. Caso essa imagem seja selecionada, o vídeo da propaganda será apresentado nesse mesmo dispositivo. O formulário para aquisição desse produto, porém, será

redistribuído, a partir do computador, para ser apresentado em outros dispositivos individuais.

O computador faz parte de um domínio, onde a TV é o dispositivo base. O computador também faz parte de um outro domínio, onde ele é o dispositivo pai. Nesse outro domínio, é o computador que faz a distribuição da apresentação aos demais dispositivos. Assim, nesse exemplo, pode ser observada a hierarquia para a distribuição das apresentações. Essa distribuição é iniciada na TV (dispositivo base) que passa para o computador (dispositivo pai em um domínio) parte da apresentação que, por fim, é entregue aos dispositivos individuais.

# 7
# Conclusões

Esta tese apresentou aspectos relacionados ao controle do sincronismo das aplicações hipermídia. Entre esses aspectos, foi destacada a importância de serem conhecidos os instantes no tempo para a ocorrência dos eventos existentes nas aplicações. Foi discutido que, sem essa informação, uma apresentação pode ser completamente realizada, mas torna-se difícil garantir a sua qualidade, bem como realizar a otimização dos recursos necessários para essa apresentação. Procurou-se destacar também algumas funcionalidades que, para serem realizadas, dependem das informações sobre a especificação temporal das aplicações, como a atualização das aplicações simultaneamente à sua apresentação e a realização de uma apresentação através de múltiplos dispositivos.

## 7.1.
## Contribuições da Tese

Ao longo dessa tese, foi proposta uma nova estrutura de dados para o controle do sincronismo temporal das aplicações. Antes, porém, foram analisadas diferentes sintaxes e estratégias para o controle desse sincronismo. Entre todas as possibilidades analisadas, foi destacada a estrutura de grafos proposta pelo sistema Firefly. Com base na análise realizada, foi então proposto um grafo temporal de dependências, denominado HTG (*Hypermedia Temporal Graph*).

No HTG, diferente de outros modelos de grafos, os relacionamentos entre as transições de eventos existentes em uma aplicação são preservados. Os instantes temporais associados à execução dos eventos em uma aplicação são calculados dividindo e, posteriormente, combinando partes de um HTG. Essa é uma importante contribuição proposta nesta tese. A partir do HTG foram definidas as cadeias temporais. Cada cadeia temporal define sequências de transições de eventos, que podem ser combinadas de diferentes formas, para controlar requisitos específicos do sincronismo temporal. Cada uma dessas possíveis combinações foi definida como um plano temporal. Um importante conjunto de planos temporais

foi proposto nesta tese, com o objetivo de orquestrar serviços voltados ao controle do sincronismo temporal de uma aplicação, desde o transporte das mídias até a sua apresentação nos clientes.

A representatividade do HTG foi detalhadamente exposta nesta tese. Para isso, as especificações da NCL, linguagem que apresenta um alto nível de abstração, foram traduzidas para as entidades do HTG. Foram destacadas as várias simplificações que podem ser obtidas na representação temporal de uma sintaxe de autoria através do HTG.

Uma sintaxe de transferência, destinada a atualizar as especificações do HTG nos clientes, simultaneamente à apresentação, também foi proposta nesta tese. Esse processo de atualização é outra contribuição importante desta tese.

Esta tese também discutiu como a execução de uma apresentação pode ser distribuída em múltiplos dispositivos. Entre essas formas, merece ser destacado o controle hierárquico proposto e o suporte que o HTG oferece a essa forma de distribuição.



## 7.2. Trabalhos Futuros

Os trabalhos realizados nesta tese abrem várias possibilidades para o desenvolvimento de outros trabalhos. Em relação aos planos temporais, um aspecto que merece maior atenção diz respeito às suas implementações. Nos clientes, o cálculo dos momentos para a construção do plano de pré-busca foi realizado de forma muito simples, baseado no pior caso. O mesmo acontece nos servidores, para o cálculo do plano de distribuição. Uma extensão natural desse trabalho é o aprimoramento de cada um dos planos propostos, seja em relação a uma plataforma específica, a um algoritmo, a uma heurística etc.

Na versão atual do Ginga-NCL, o uso dos planos temporais só é possível em conjunto com a estrutura utilizada pelo módulo Escalonador desse ambiente declarativo. Essa abordagem é necessária, pois a estrutura desse módulo controla aspectos da apresentação que não estão relacionados ao tempo. No entanto, o Escalonador replica muitas das funcionalidades fornecidas pelo HTG, pois foi pensado e especificado quando esta estrutura de dados ainda não existia. Assim, uma nova implementação do Escalonador é desejável, eliminando todas as

redundâncias com o HTG, o que o tornará muito mais simples. Através dessa simplificação, além de diminuir o *overhead* entre as implementações, é esperado que uma implementação mais robusta do módulo Escalonador seja obtida, e assim do Ginga-NCL, uma vez que esse módulo é seu núcleo central. Atualmente, esse módulo pode instanciar um conjunto ilimitado de processos leves (*threads*) para controlar a apresentação, tornando muito difícil a verificação do código que o implementa.

Os trabalhos desta tese não incluem a realização de adaptações para compensar atrasos maiores do que os valores previstos na construção dos planos. O objetivo dos planos é justamente tentar evitar que atrasos aconteçam, ao possibilitar que o estado futuro de uma apresentação possa ser previsto. No entanto, considerando que não é possível impedir a ocorrência de atrasos, outro trabalho futuro é a realização de adaptações de tempo elástico (Bachelet, 2007) sobre as apresentações.

Em relação à apresentação através de múltiplos dispositivos, a versão atual incorpora facilidades para controle das apresentações. A edição das aplicações apresentadas através de múltiplos dispositivos, no entanto, atualmente não é considerada. Essa edição é um importante trabalho futuro, que envolve também a atualização das aplicações simultaneamente à apresentação.

Por fim, é importante mencionar que o HTG, apesar de ser originalmente proposto para o controle da apresentação, também pode ser utilizado como apoio para ferramentas de autoria das aplicações. Uma possibilidade é utilizar, na autoria, o mesmo cálculo dos instantes no tempo para apresentação, permitindo a visualização temporal das aplicações.

# 8
# Referências


ABNT – Associação Brasileira de Normas e Técnicas. **Televisão digital terrestre – Sistema de Transmissão.** ABNT, NBR 15601:2007. Versão corrigida. 2008.

ABNT – Associação Brasileira de Normas e Técnicas. **Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 2: Ginga-NCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações.** ABNT, NBR 15606-2:2007. Versão corrigida 3. 2009.

ALLEN, J.F.; **Maintaining Knowledge about Temporal Intervals**, Comunication of the ACM, 26. Novembro, 1983.

ARIB – Association of Radio Industries and Businesses. **Data Coding and Transmission Specifications for Digital Broadcasting**. ARIB, STD-B24, Volume 2: XML-Based Multimedia Coding Schema. 2005.

ATSC - Advanced Television Systems Committee. **ATSC Standard: Advanced Common Application Platform (ACAP)**. Padrão A/101. 2005.

BACHELET, B.; MAHEY, P.; RODRIGUES, R.F.; SOARES, L.F.G. **Elastic Time Computation in QoS-driven Hypermedia Presentation**. ACM Multimedia Systems. Maio, 2007.

BAECKER, R.; ROSENTHAL, A.J.; FRIEDLANDER, N.; SMITH, E.; COHEN, A. **Multimedia System for Authoring Motion Pictures**. ACM Multimedia. Boston, EUA. 1996.

BERTINO, E.; FERRARI, E. **Temporal Synchronization Models for Multimedia Data**. IEEE Transactions on Knowledge and Data Engineering. Agosto, 1998.

BLAKOWSKI, G.; STEINMETZ, R. **A Media Synchronization Survey: Reference Model, Specification, and Case Studies**. IEEE Journal on Selected Areas in Communications, n. 14. Janeiro, 1996.

BUCHANAN, M.C.; ZELLWEGER, P.T. **Specifying Temporal Behavior in Hypermedia Documents**. European Conference on Hypertext. Milão, Itália. Dezembro, 1992.

BUCHANAN, M.C.; ZELLWEGER, P.T. **Automatic Temporal Layout Mechanisms**. ACM Multimedia. California, EUA. Agosto, 1993.

BULTERMAN, D.C.A.; VAN ROSSUM, G.; VAN LIERE, R. **A Structure for Transportable Dynamic Multimedia Documents**. USENIX Conference. 1991.

BULTERMAN, D.C.A.; JANSEN, J.; KLEANTHOUS, K.; BLOM, K.; BENDEN, D. **AMBULANT: A Fast, Multi-Platform Open Source SMIL Player**. ACM International Conference on Multimedia. Nova York, EUA. 2004.

BULTERMAN, D.C.A.; HARDMAN, L. **Structured Multimedia Authoring**. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP). ISSN 1551-6857. Nova York, EUA. 2005.

CESAR, P.; BULTERMAN, D.C.A.; JANSEN, J. **An Architecture for End-User TV Content Enrichment**. European Interactive TV Conference. Atenas, Grécia. 2006.

CESAR, P.; BULTERMAN, D.C.A.; OBRENOVIC, Z.; DUCRET, J.; CRUZ-LARA, S. **An Architecture for Non-Intrusive User Interfaces for Interactive Digital Television Experiences**. European Interactive TV Conference. Amsterdam, Holanda. 2007.

CESAR, P.; BULTERMAN, D.C.A.; GEERTS, D.; JANSEN, J.; KNOCHE, H.; SEAGER, W. **Enhancing Social Sharing of Videos: Fragment, Annotate, Enrich, and Share**. ACM International Conference on Multimedia. Vancouver, Canadá. Outubro, 2008.

CHUNG, S.M.; PEREIRA, A.L. **Timed Petri Net Representation of SMIL**. IEEE Multimedia, v. 12, n. 1. Março, 2005.

COSTA, R.M.R; MORENO, M.F.; RODRIGUES, R.F.; Soares L.F.G. **Live Editing of Hypermedia Documents.** ACM Symposium on Document Engineering. Amsterdam, Holanda. 2006.

COSTA R.M.R; MORENO M.F.; SOARES L.F.G. **Intermedia Synchronization Management in DTV Systems.** ACM Symposium on Document Engineering. São Paulo, Brasil. 2008.

COSTA, R.M.R; MORENO, M.F.; SOARES, L.F.G. **Ginga-NCL: Suporte a Múltiplos Dispositivos.** Simpósio Brasileiro de Sistemas Multimedia e Web – WebMedia. Natal, Brasil. 2009.

CONCOLATO, C.; LE FEUVRE, J.; MOISSINAC, J.C. **Timed-Fragmentation of SVG Documents to Control the Playback Memory Usage**. ACM Symposium on Document Engineering. Nova York, EUA. 2007.

DINI, R.; PATERNO, F.; SANTORO, C. **An Environment to Support Multi-User Interaction and Cooperation for Improving Museum Visits through Games**. Mobile Human-Computer Interaction Conference. Cingapura. Setembro, 2007.

ETSI – European Telecommunications Standards Institute. **Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.1.1.** Especificação Técnica ETSI TS 102 B12. 2005.

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

GAREY, M.R.; JOHNSON, D.S. **Computers and Intractablitiy: A Guide to the Theory of NP-Completeness**. ISSN 978-0716710455. W. H. Freeman. Nova York, EUA. 1979.

ISO/IEC - International Organization for Standardization 13818-6. **Information technology – Generic coding of moving pictures and associated audio information - Part 6: Extensions for DSM-CC**. 1998.

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

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

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

ISO/IEC - International Organization for Standardization. **14496-20:2006. Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format.** 2006.

ITU-T – International Telecommunication Union - **Recommendation H.761 - Nested Context Language (NCL) and Ginga-NCL for IPTV Services**. Geneva, Suição. Abril, 2009.

JOURDAN, M.; ROISIN, C.; TARDIF, L. **Madeus, an Authoring Environment for Interactive Multimedia Documents**. ACM Multimedia Conference. Inglaterra. Setembro, 1998.

KIM, M.; WOOD, S. **MPEG-4 Flexible Timing Standard (FlexTime)**. Overview of FlexTime. IBM Research. 2002. Disponível em http://www.research.ibm.com/mpeg4/Projects/Flextime.htm. Acesso em: 05 jul. 2010.

LITTLE, T.; GHAFOOR, A. **Synchronization and Storage Models for Multimedia Objects**, IEEE Journal on Selected Areas in Communications, v. 8, n. 3. ISSN 0733-8716. Abril, 1990.

MORENO, M.F.; COSTA, R.M.R; SOARES, L.F.G. **Sincronismo entre Fluxos de Mídia Contínua e Aplicações Multimídia em Redes por Difusão.** Simpósio Brasileiro de Sistemas Multimedia e Web – WebMedia. Vilha Velha, Brasil. 2008.

MORENO, M.F. **Ginga-NCL: Relating Imperative, Declarative and Media Objects.** Workshop of Thesis and Dissertations. European Interactive TV Conference. Leven, Bélgica. 2009.

MORENO, M.F.; SOARES, L.F.G; CERQUEIRA, R. **Uma Arquitetura Orientada a Componentes para o Middleware Ginga-NCL.** Simpósio Brasileiro de Sistemas Multimedia e Web – WebMedia. Belo Horizonte, Brasil, 2010.

MUCHALUAT-SAADE, D.C.; RODRIGUES, R.F.; SOARES, L.F.G. **XConnector: Extending XLink to Provide Multimedia Synchronization.** ACM Symposium on Document Engineering. McLean, EUA. 2002.

NETO, C.S.S; SOARES, L.F.G. **Reuso e Importação em Nested Context Language**. Simpósio Brasileiro de Sistemas Multimedia e Web – WebMedia. Fortaleza, Brasil. 2009.

PATERNO, F.; SANTORO, C.; MANTYJARVI, J.; MORI, G.; SANSONE, S. **Authoring Pervasive Multimodal User Interfaces.** International Journal of Web Engineering and Technology, v. 4. ISSN: 1476-1289. 2008.

PÉREZ-LUQUE, M.J.; LITTLE, T.D.C. **A Temporal Reference Framework for Multimedia Synchronization**. IEEE Journal on Selected Areas in Communications. ISSN 0733-8716. Janeiro, 1996.

PESSOA, B.J.S.; SOUZA, G.L.S.; CABRAL, L.A.F. **Metaheurísticas Aplicadas à Geração de Carrossel no Sistema Brasileiro de TV Digital.** Simpósio Brasileiro de Sistemas Multimedia e Web – WebMedia. Vilha Velha, Brasil. 2008.

RODRIGUES, R. F. **Formatação e Controle de Apresentações Hipermídia com Mecanismos de Adaptação Temporal**. 2003. 170 p. Tese (Doutorado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, 2003.

RODRIGUES, R. F.; SOAREZ, L.F.G. **Inter and Intra Media-Object QoS Provisioning in Adaptive Formatters**. ACM Symposium on Document Engineering. Grenoble, França, 2003.

SCHROYEN, J.; GABRIELS, K.; LUYTEN, K.; TEUNKENS, D.; et al. **Training Social Learning Skills by Collaborative Mobile Gaming in Museums**. International Conference on Advances in Computer Entertainment Technology. Yokohama, Japão. Dezembro, 2008.

SOARES, L.F.G; RODRIGUES, R.F.; MUCHALUAT-SAADE, D.C. **Modeling, Authoring and Formatting Hypermedia Documents in the HyperProp System**. Multimedia Systems, v. 8, n. 2. Alemanha. 2000.

SOARES, L.F.G.; RODRIGUES R.F. **Nested Context Model 3.0 Part 1 – NCM Core.** Relatório Técnico do Departamento de Informática da PUC-RIO, MCC 18/05. ISSN 0103-9741. Rio de Janeiro, Brasil. Maio, 2005.

SOARES, L.F.G.; et al. **Sistema Brasileiro de Televisão Digital – Recomendações para o Modelo de Referência – Sincronismo de Mídias.** Relatório Técnico do Departamento de Informática da PUC-RIO, MCC 41/05. ISSN 0103-9741. Rio de Janeiro, Brasil. Dezembro, 2005.

SOARES, L.F.G; RODRIGUES, R.F. **Nested Context Language 3.0 Part 8 – NCL Digital TV Profiles**. Relatório Técnico do Departamento de Informática da PUC-RIO, MCC 35/06. ISSN 0103-9741. Rio de Janeiro, Brasil. Outubro, 2006.

SOARES, L.F.G; RODRIGUES, R.F.; COSTA, R.M.R. **Geração Automática de Frameworks para Processamento de Documentos XML.** Simpósio Brasileiro de Sistemas Multimedia e Web – WebMedia. Natal, Rio Grande do Norte. 2006.

SOARES, L.F.G.; RODRIGUES, R.F.; MORENO, M.F. **Ginga-NCL: the Declarative Environment of the Brazilian Digital TV System**. Journal of the Brazilian Computer Society. ISSN 0104-6500. 2007.

SOARES, L.F.G.; COSTA, R.M.R; MORENO, M.F.; MORENO, M.F. **Multiple Exhibition Devices in DTV Systems.** ACM International Conference on Multimedia. Beijing, China. 2009.

SOARES, L.F.G.; BARBOSA, S.D.J. **Programando em NCL 3.0 – Desenvolvimento de Aplicações para o Middleware Ginga, TV Digital e Web**. ISBN 978-85-352-3457-2. Elsevier. 2009.

SOARES, L.F.G.; MORENO, M.F.; SOARES NETO, C.S.; MORENO, M.F. **Ginga-NCL: Declarative Middleware for Multimedia IPTV Services**. IEEE Communications Magazine, v. 48, n. 6. ISSN: 0163-6804. Junho, 2010.

SOARES, L.F.G.; RODRIGUES, R.F; CERQUEIRA, R.; BARBOSA, S.D.J. **Variable and State Handling in NCL**. Multimedia Tools and Applications. ISSN: 1380-7501. Março, 2010.

SOUZA, G.L. **Sincronismo na Modelagem e Execução de Apresentações de Documentos Multimídia**. 2003. 231 p. Tese (Doutorado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro. 1997.

SOUZA, G.L.; Leite, L.E.C; Batista, C.E.C.F. **Ginga-J: The Procedural Middleware for the Brazilian Digital TV System**. Journal of the Brazilian Computer Society, v. 13, n. 4. ISSN: 0104-6500. 2007.

VAN ROSSUM, G.; JANSEN, J.; MULLENDER, K.S.; BULTERMAN, D.C.A. **CMIFed: A Presentation Environment for Portable Hypermedia Documents**. ACM Multimedia. Anaheim, EUA. Agosto, 1993.

VUONG, S.; COOPER, K.; ITO, M. **Petri Net Models for Describing Multimedia Synchronization Requirements**, International Conference on Network Protocols. Japão. Novembro, 1995.

W3C - World-Wide Web Consortium. **XHTML 1.0 The Extensible HyperText Markup Language (Second Edition)**. W3C Recommendation. Agosto, 2002.

W3C - World-Wide Web Consortium. **Scalable Vector Graphics – SVG 1.1 Specification**. W3C Recommendation. Janeiro, 2003.

W3C - World-Wide Web Consortium. **XML Schema Part 0: Primer (Second Edition)**. W3C Recommendation. Outubro, 2004.

W3C - World-Wide Web Consortium. **XML Schema Part 1: Structures (Second Edition)**. W3C Recommendation. Outubro, 2004.

W3C - World-Wide Web Consortium. **XML Schema Part 2: Datatypes (Second Edition)**. W3C Recommendation. Outubro, 2004.

W3C - World-Wide Web Consortium. **Extensible Markup Language (XML) 1.0 (Fifth Edition)**. W3C Recommendation. Novembro, 2008.

W3C - World-Wide Web Consortium. **Synchronized Multimedia Integration Language (SMIL 3.0) Specification**. W3C Recommendation. Dezembro, 2008.

WOO, M.; QAZI, N.; GHAFOOR, A. **A Synchronization Framework for Communication of Pre-Orchestrated Multimedia Information**. IEEE Network. Fevereiro, 1994.

YAMADA, F.; BEDICKS, G.; CASTRO, P.H. **Modulação do Sistema Brasileiro de TV Digital**. Revista Produção Profissional – A Modulação do SBTVD. Edição 74. Fevereiro, 2008.

YANG, C.C.; HUANG, J.H. **A Multimedia Synchronization Model and Its Implementation in Transport Protocols**, IEEE Journal on Selected Areas in Communications, v. 14, n.1. Janeiro, 1996.

YANG, C.C. **Detection of the Time Conflicts for SMIL-Based Multimedia Presentations**. Workshop on Computer Networks, Internet and Multimedia. International Computer Symposium. Taiwan, 2000.

YANG, C.C.; TIEN, C.W.; WANG, Y.C. **Modeling of the Non-Deterministic Synchronization Behaviors in SMIL 2.0 Documents**, International Conference on Multimedia and Expo, v. 3. Julho, 2003.

# Apêndice A
# Sincronismo Temporal de Aplicações Hipermídia

Nas aplicações hipermídia, a especificação de quando os eventos associados às mídias deverão ocorrer corresponde ao seu sincronismo temporal. Essas aplicações podem conter mídias com conteúdo dinâmico, como vídeos, áudios e animações e também mídias com eventos associados entre si, como na exibição simultânea de um áudio e um vídeo com conteúdos semanticamente relacionados. Esse último caso, que constitui o foco deste capítulo, é usualmente denominado sincronismo intermídia, sendo diferente do primeiro caso, usualmente denominado sincronismo intramídia, que trata da sincronização entre as várias unidades que compõem um mesmo objeto.[24]

Diferentes modelos podem ser utilizados para controlar o sincronismo temporal das aplicações. Em (Blakowski, 1996) esses modelos são classificados em eixo do tempo, baseados em intervalos, hierárquicos, scripts, baseados em redes de Petri, baseados em pontos de referência e baseados em eventos. Essa, no entanto, não é a única classificação existente. Em (Bertino, 1998), por exemplo, os modelos são classificados em grafos, redes de Petri, orientados a objetos e baseados em linguagens.

Nas classificações propostas em (Blakowski, 1996) e (Bertino, 1998), os modelos em comum são descritos, basicamente, com as mesmas características. Diferentes classificações não implicam divergências em relação às características dos modelos considerados, elas normalmente são o resultado de diferentes objetivos de avaliação. Em (Bulterman, 2005), por exemplo, onde a análise dos modelos para o sincronismo das aplicações é voltada à autoria, os principais modelos são classificados em eixo do tempo, estruturados, scripts e grafos.

Neste capítulo não é adotada uma classificação específica. O foco são os modelos e, assim, foram eles os escolhidos e não suas possíveis classificações. Porém, a fim de organizar as comparações entre os modelos, quando necessário,

[24] Embora distintos, os tipos de sincronização devem funcionar de forma colaborativa. Para manter a sincronização intermídia, por exemplo, é importante preservar e controlar a sincronização intramídia dos objetos relacionados (Rodrigues, 2003b).

as classificações propostas em (Blakowski, 1996) e (Bertino, 1998) foram utilizadas. Por fim, cabe mencionar que este capítulo não tem por objetivo esgotar a descrição de todos os modelos existentes e sim priorizar aqueles utilizados em padrões e ferramentas voltados à apresentação das aplicações.

## A.1. Modelos de Sincronização Temporal

### A.1.1. Sincronização Baseada em um Eixo do Tempo (*Timeline*)

Existem várias alternativas para a construção de estruturas baseadas em eixo do tempo, das quais podem ser destacadas a multiplexação síncrona, o selo de tempo (*timestamping*) e a linha de tempo (*timeline*) (Soares, 2005b); sendo essa última utilizada tanto para a especificação quanto para a apresentação das aplicações (Bulterman, 2005), o que contribuiu para que a expressão *timeline* fosse utilizada em equivalência à de eixo do tempo.

A sincronização *timeline* tem por base a associação de um tempo absoluto a cada evento. Essa associação é realizada em relação a uma base, por exemplo, o início da apresentação de uma aplicação, assim, os eventos são posicionados em um eixo abstrato do tempo, com origem, por exemplo, no início da apresentação.

Na autoria, a sincronização *timeline* oferece uma abstração do comportamento temporal de um documento (Bulterman, 2005), através da qual o autor pode conhecer a distribuição das mídias no tempo sem conhecer detalhes sobre a implementação do seu sincronismo. No entanto, vários são os problemas conhecidos desse modelo (Bulterman, 2005; Soares, 2005b). Através dele não é possível especificar eventos cujo início ou duração não podem ser previstos antes do início da apresentação, como, por exemplo, as interações dos usuários. Esse modelo também pode tornar difícil a manutenção das aplicações, uma vez que modificações no posicionamento dos eventos associados a uma mídia podem exigir modificações nos momentos temporais associados à execução de outros eventos, a fim de preservar a semântica original da aplicação.

Para superar as limitações da sincronização *timeline*, uma estratégia possível é utilizar construções baseadas nesse modelo apenas para a apresentação e transmissão das aplicações, utilizando um outro modelo na autoria. Essa opção é,

por exemplo, oferecida pelo MPEG-4 Sistemas (ISO/IEC, 2001), que define um formato binário denominado BIFS (*Binary Format for Scenes*) (ISO/IEC, 2001) para a apresentação e o transporte das aplicações através da sincronização *timeline*.[25]

Especificações BIFS são projetadas com o objetivo de serem multiplexadas de forma síncrona em fluxos que possam ser apresentados tão logo sejam recebidos pelos clientes. BIFS permite que as modificações realizadas nas aplicações sejam atualizadas nos clientes simultaneamente à apresentação. Cada apresentação BIFS é organizada em cenas, cada uma estruturada através de uma hierarquia de objetos que representam os conteúdos das mídias. Os conteúdos das mídias em uma cena podem ser atualizados durante a apresentação e as próprias cenas também podem ser atualizadas através da inserção ou exclusão de objetos ou ainda através de mudanças nas propriedades dos objetos. Também é possível inserir ou excluir eventos relacionados aos objetos em cena, como eventos de animação e interação. É possível ainda substituir uma cena inteiramente por outra (Herpel, 1999).

Apesar de adotar a sincronização timeline, BIFS permite que sejam especificadas aplicações com alguma adaptação e interação. Múltiplos conteúdos relacionados a uma mesma mídia da aplicação podem ser simultaneamente transmitidos, para que um desses conteúdos seja escolhido durante a apresentação. No entanto, caso os conteúdos a serem adaptados tenham durações distintas, não será possível associar o evento de apresentação do conteúdo escolhido a outras mídias da aplicação.

Em relação à interatividade, BIFS define dois casos particulares. Um deles consiste na possibilidade de alterar propriedades relacionadas à apresentação das mídias como, por exemplo, definir a exibição de uma mídia visível (verdadeiro) ou não (falso), como resultado de uma ação interativa. Para mídias como imagens e textos, essa possibilidade permite simular que as mídias são apresentadas em função da ocorrência de um evento interativo, quando, na verdade, as mídias já estavam sendo executadas, porém de forma invisível. No entanto, para mídias que possuem durações associadas, como áudios e vídeos, essa simulação não é

[25] BIFS opcionalmente oferece suporte a algumas construções temporais usando um modelo chamado *FlexTime* (Kim, 2002) que implementa um conjunto limitado das relações de Allen (Allen, 1983). Essa opção, no entanto, não faz parte da implementação de referência para apresentações desse formato (ISO/IEC, 2000a).

possível. Outra possibilidade, relacionada à interatividade, consiste em substituir uma cena inteiramente por outra, a partir da ocorrência de um evento interativo, o que corresponde a encerrar a apresentação atual e iniciar a apresentação de uma outra aplicação.

Uma aplicação original pode ser dividida em um conjunto de pequenas aplicações, cada uma formada por um conjunto de mídias que possuem eventos associados a um mesmo momento temporal. Através dessa abordagem, eventos interativos podem ser utilizados para encerrar a apresentação atual e iniciar a apresentação de uma outra aplicação, ambas obtidas a partir da aplicação original. Essa estratégia, além de definida no MPEG-4, também pode ser utilizada nos sistemas de TV digital europeu (ETSI, 2005), americano (ATSC, 2005) ou japonês (ARIB, 2005), onde linguagens baseadas em XHTML (W3C, 2002) são oferecidas para autoria declarativa. Nesse caso, várias aplicações XHTML, cada uma com a especificação do leiaute espacial de um conjunto de mídias, podem ter suas apresentações iniciadas a partir de eventos interativos disparados por outras aplicações ou ainda através de eventos de sincronismo (ISO/IEC, 1998), transmitidos para os clientes em momentos específicos da apresentação do conteúdo audiovisual principal.

Dividir uma aplicação em várias pequenas aplicações é conveniente apenas para aplicações simples e, mesmo nesse caso, a semântica original pode ser completamente perdida. Em seu lugar tem-se um conjunto de apresentações isoladas executadas no tempo. A semântica da aplicação original permanece apenas na idéia do autor, que passa a ser também o controlador da apresentação, tarefa que para aplicações com um grande número de mídias e eventos pode ser difícil.

Quando utilizado na apresentação e transporte das aplicações, o sincronismo *timeline* não é capaz de representar os relacionamentos entre os eventos especificados na autoria. Mesmo as facilidades propostas pelo MPEG-4 para o formato BIFS não impedem que a semântica dos relacionamentos entre os eventos especificados de forma relativa na autoria seja perdida na apresentação, restando apenas os momentos previstos para o disparo dos eventos.

### A.1.2.
### Sincronização Hierárquica

Na sincronização hierárquica a especificação do sincronismo é realizada baseada em duas operações principais: a sincronização paralela e serial de ações (Baecker, 1996). Uma ação pode ser atômica ou composta. Ações atômicas estão relacionadas com a apresentação de uma única mídia ou com uma interação do usuário. Ações compostas são formadas por um conjunto de ações atômicas e operações de sincronização (paralela e sequencial). Graficamente, essa forma de sincronização pode ser definida como uma árvore, onde os nós folhas (terminais) representam as ações atômicas e os demais as operações de sincronização.

É possível que existam variações na estrutura proposta para que, por exemplo, seja possível especificar atrasos associados às ações. Um atraso pode ser especificado, por exemplo, como uma ação atômica, sendo aplicado as demais ações dentro de uma mesma ação composta (vanRossum, 1993).

Na Figura 9.1, uma árvore representando a sincronização hierárquica de uma aplicação formada por dois vídeos, dois áudios e dois conteúdos textuais é apresentada. Nessa aplicação, um vídeo, um áudio e um texto semanticamente relacionados são apresentados em paralelo, em sequência são apresentados, novamente em paralelo, as demais mídias.

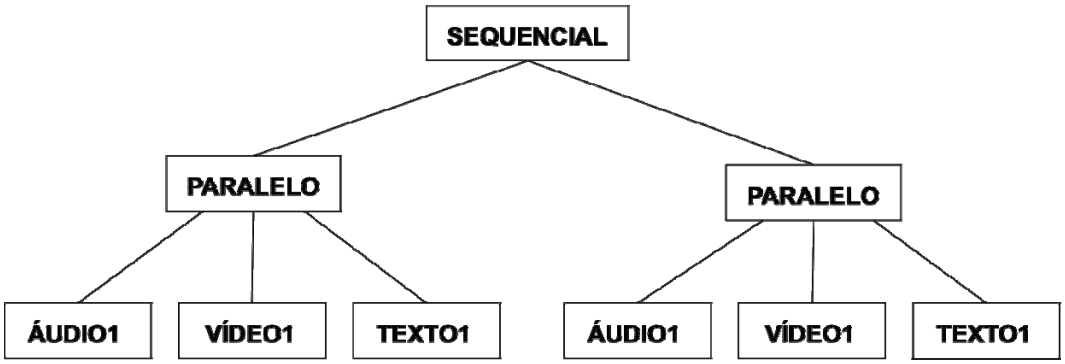


Figura A.1 - Exemplo de sincronização hierárquica.

Na sincronização hierárquica, relacionamentos entre os eventos são especificados de forma simples, através de combinações com semântica paralela e sequencial. No entanto, essa sincronização também limita a especificação dos relacionamentos apenas ao início ou ao final dos eventos de apresentação das mídias. Outros pontos de sincronização, relacionados a momentos da duração da

apresentação das mídias também podem ocorrer na sincronização hierárquica, porém de forma implícita, através da especificação de atrasos nas aplicações.

Como não é possível especificar explicitamente relacionamentos de sincronização entre os eventos que ocorrem durante a exibição de uma determinada mídia, o autor pode ser obrigado a dividir as mídias originais em várias outras. No exemplo apresentado na Figura 2, o áudio e o vídeo que são apresentados em paralelo foram divididos em duas partes, para que legendas, formadas por comentários textuais, pudessem estar sincronamente relacionadas a partes específicas (momentos temporais da duração) do vídeo. Uma outra opção seria a especificação de atrasos, mas, nesse caso, a especificação do sincronismo seria definida de forma implícita (Souza, 1997).

O formato CMIF (*CWI Multimedia Interchange Format*) (Bulterman, 1991), que corresponde a base da especificação da linguagem SMIL (*Synchronized Multimedia Integration Language*) (W3C, 2008b), recomendação W3C para a descrição de aplicações multimídia na Web, oferece construções baseadas no modelo hierárquico de sincronização. As aplicações CMIF são especificadas através de nós representando operações de sincronização paralela ou sequencial, que por sua vez podem conter ações atômicas associadas à apresentação das mídias.

O formato CMIF, além de possuir as características da sincronização hierárquica, permite que os atrasos em uma aplicação sejam especificados de forma explítica, através de relacionamentos entre ações atômicas definidas em uma mesma ação composta. Esses relacionamentos, denominados arcos de sincronização (Bulterman, 1991; vanRossum, 1993), são representados na árvore hierárquica através de arestas dirigidas entre os nós das ações atômicas envolvidas.

Na especificação CMIF do exemplo apresentado na Figura 9.1, o áudio e o vídeo são representados cada um por uma única ação, ambos sincronizados em uma mesma ação composta com semântica paralela, contendo também os dois textos correspondentes às legendas. Dois arcos de sincronização são necessários nessa aplicação, relacionando o início da exibição do vídeo com o início da exibição de cada legenda, cada um com a especificação do atraso necessário para posicionar a exibição da legenda no momento correto de exibição do vídeo.

Arcos de sincronização estendem as facilidades da sincronização hierárquica para a especificação das aplicações (vanRossum, 1993), mas não trazem, assim como os nós correspondentes às ações atômicas e operações de sincronização, facilidades à apresentação. Outros modelos de sincronização são normalmente utilizados para controlar a apresentação das aplicações especificadas de forma hierárquica. Esse é o caso, por exemplo, do exibidor do ambiente CMIFed (vanRossum, 1993), proposto para executar a apresentação de aplicações CMIF, que projeta, durante a compilação das aplicações, um grafo de dependências temporais a partir da sua especificação de sincronização hierárquica.

No grafo de dependências temporais do CMIFed, nós correspondem aos pontos de início e fim das ações compostas e atômicas e as arestas representam as relações temporais entre dois desses nós. A estrutura adotada pelo CMIFed, embora calculada a partir de uma estrutura hierárquica, possui características distintas desse modelo de sincronização, que serão discutidas ainda neste capítulo.



### A.1.3. Sincronização Baseada em Redes de Petri

A estrutura de uma rede de Petri (Chung, 2005; Little, 1990; Vuong, 1995; Woo, 1994; Yang, 2003) pode ser utilizada para representar o sincronismo das aplicações e permitir que os momentos temporais relacionados à execução dos eventos sejam calculados (Yang, 1996; Yang, 2000; Yang 2003).

Os elementos que formam uma rede de Petri podem ser graficamente representados por um grafo dirigido, com dois tipos de nós: círculos representando os lugares (*places*) e barras representando as transições (*transitions*). No grafo, arestas dirigidas unem os círculos às barras e vice-versa. Uma aresta com origem em um círculo e destino em uma barra define o círculo de origem como uma função de entrada da transição. De maneira similar, uma função de saída é definida quando existe uma aresta com origem em uma transição e destino em um círculo. Além dessas entidades, fichas (*tokens*) podem estar associadas aos lugares existentes em uma rede. No grafo, fichas são representadas por pontos preenchidos desenhados nos círculos. As fichas definem o estado de execução de uma rede de Petri (Vuong, 1995).

As entidades tradicionais de uma rede de Petri podem ter suas definições estendidas, ou novas entidades, além daquelas citadas no parágrafo anterior,

podem ser adicionadas. Esse é o caso das *Timed Petri Nets* – TPN (Chung, 2005), onde estruturas temporais de duração (*clock*) são adicionadas aos elementos das redes de Petri. A OCPN (*Object Composition Petri Net*) (Little, 1990) é um exemplo de uma TPN onde, após receber uma ficha, um lugar passa para o estado ativo e permanece nesse estado durante o intervalo especificado pela sua duração (tempo de exibição da mídia). Uma vez no estado ativo, o lugar permanece com a ficha, que somente será liberada quando a sua duração for atingida. As transições na OCPN representam pontos de sincronização, disparados quando todos os seus lugares de origem contêm fichas liberadas. Quando uma transição é executada, as fichas passam dos lugares de origem para os lugares de destino de uma determinada transição.

Em (Vuong, 1995) são comparadas diferentes propostas de redes de Petri com enfoque temporal, que incluem a OCPN, a XOCPN (*Extended Object Composition Petri Net*) (Woo, 1994), entre outras. Infelizmente, mesmo nas propostas destinadas ao controle temporal, como nas variações das TPN, não é possível a especificação da sincronização em relação a eventos que ocorrem durante a exibição das mídias. É possível, de forma similar à sincronização hierárquica, realizar a divisão das mídias originais em partes temporalmente relacionadas. Essa, no entanto, é uma solução que torna a representação das redes de Petri mais complexa, tanto para a especificação quanto para o controle da apresentação das aplicações.

Considerando a importância das redes de Petri na modelagem de sistemas concorrentes, principalmente em relação à análise de propriedades que incluem a simulação do comportamento de uma aplicação, alguns trabalhos, que têm como objetivo facilitar a especificação de aplicações multimídia e hipermídia através de redes de Petri, merecem ser considerados.

Em (Chung, 2005) é proposta uma abordagem onde os lugares (*places*), além de obedecerem a duração de exibição das mídias, como normalmente ocorre nas TPN, podem representar intervalos temporais quaisquer ou até mesmo eventos, que podem ser, inclusive, disparados por ações interativas. Nessa proposta, cada lugar pode ter até duas fichas, de tal forma que a combinação dessas fichas define se o lugar encontra-se no estado ativo, inativo e também se a transição de destino desse lugar deve ser executada (Chung, 2005). Quando essa proposta é utilizada para o controle da apresentação, um lugar ativo representa o

início da exibição de uma mídia associada. Quando um lugar torna-se inativo, ocorre uma pausa na exibição dessa mídia, que terá sua exibição resumida somente quando o lugar em questão se tornar novamente ativo. A exibição da mídia termina quando a transição de destino do lugar considerado for executada.

A ERTSM (*Extended Real-Time Synchronization Model*) (Yang, 2003) é outro exemplo de uma rede de Petri proposta para a modelagem de aplicações hipermídia. Nessa estrutura, os lugares são classificados em lugares regulares (*regular*) ou lugares a serem executados (*enforced*). Lugares regulares têm a mesma semântica definida na OCPN, enquanto os lugares a serem executados provocam a execução imediata das transições de destino quando a ficha desses lugares encontra-se desbloqueada. Lugares a serem executados, além da exibição das mídias, podem estar associados a intervalos temporais ou a diferentes tipos de eventos, incluindo os imprevisíveis, como aqueles executados por ações interativas. O ERTSM especifica também um conjunto de novas entidades, denominadas controladores, para modelar o reinício, a repetição e o tempo mínimo dos eventos associados aos lugares. Para auxiliar a apresentação das aplicações, em (Yang, 2003) é proposto um algoritmo para o cálculo do tempo de apresentação de aplicações SMIL modeladas através do ERTSM.

Propostas como a ERTSM e a descrita em (Chung, 2005) adicionam novas entidades e opções semânticas às redes de Petri com o objetivo de facilitar a especificação das aplicações hipermídia. Em relação à apresentação, essas propostas oferecem facilidades como, por exemplo, o cálculo do tempo de apresentação das aplicações (Yang, 2003) e a detecção de conflitos temporais (Yang, 2000), incluindo a localização de relacionamentos que, embora estejam especificados, nunca irão ocorrer durante a apresentação. Por outro lado, embora estruturas baseadas em redes de Petri possuam a formalização necessária para a verificação de muitas propriedades temporais, essa análise formal não é normalmente o foco do controle da apresentação das aplicações hipermídia, e sua implementação em tempo de apresentação das aplicações pode ser difícil. Isso é particularmente verdade considerando os diversos novos elementos e as novas opções semânticas que fazem parte de muitas das variações de redes de Petri apresentadas, incluindo a própria ERTSM, a proposta em (Chung, 2005) e diversas outras (Little, 1990; Vuong, 1995; Woo, 1994; Yang, 1996).

### A.1.4.
### Sincronização Baseada em Causalidade/Restrição de Eventos

A sincronização baseada em eventos é aquela onde as ações associadas à apresentação, como o seu início e fim, são executadas a partir da ocorrência de eventos de sincronização, como a exibição de uma parte temporal do conteúdo de uma mídia. Nesse tipo de sincronização, a especificação de uma aplicação pode ser completamente definida de forma relativa, sem qualquer menção explícita ao tempo (Soares, 2005b).

Relacionamentos entre os eventos podem envolver causalidade ou restrição. Relacionamentos causais envolvem ações que devem ser executadas quando condições estabelecidas forem satisfeitas. Por exemplo, em relação a duas mídias simultaneamente exibidas, o fim da exibição de uma dessas mídias pode ser definido como condição para o término da outra. Por outro lado, relacionamentos de restrição estabelecem regras que devem ser obedecidas, como exemplo, duas mídias, quando simultaneamente exibidas devem terminar ao mesmo tempo. Diferente do primeiro exemplo, em que uma das mídias pode ter sua exibição encerrada antes do seu fim natural, para que a restrição do segundo exemplo seja obedecida, as últimas amostras de ambas as mídias devem ser simultaneamente exibidas.

Em relação aos eventos, além da apresentação, outras ocorrências relativas às aplicações também merecem ser consideradas, como, por exemplo, a seleção de uma mídia ou de uma parte do seu conteúdo, a atribuição de valores às propriedades das mídias ou da aplicação, entre outras. Todas essas ocorrências, independente do fato de terem uma duração ou serem atômicas, definem eventos que podem ser qualificados como condição ou ação em um relacionamento causal ou ainda fazer parte de uma restrição associada a um relacionamento.

A especificação do sincronismo baseada em relacionamentos entre eventos, com semântica causal ou de restrição, é útil para aplicações onde o sincronismo depende da ocorrência de eventos com duração variável ou mesmo imprevisível no momento da especificação. Por outro lado, uma vez que os momentos temporais absolutos para a execução dos eventos não são conhecidos, a apresentação das aplicações pode se tornar uma tarefa difícil.

É possível realizar a apresentação das aplicações sem que os momentos temporais absolutos para a execução dos eventos sejam conhecidos. Essa é, por exemplo, a proposta do sistema HyperProp (Soares, 2000), cujo modelo de execução implementa, através do paradigma de orientação a objetos (Rodrigues, 2003a), a sincronização baseada em causalidade e restrição de eventos. Nesse sistema, os conteúdos das mídias e os seus eventos associados são representados por objetos, que encapsulam suas propriedades e ações. A visibilidade entre os objetos é limitada. Dependendo dos relacionamentos existentes na aplicação, um objeto deve se registrar em outro para receber notificações a respeito de alterações nas propriedades ou ações desse outro objeto. Quando uma notificação é recebida, ações podem ser executadas no objeto registrado, gerando notificações em outros objetos e, consequentemente, executando novas ações, sucessivamente.

Quando o controle da apresentação é realizado obedecendo diretamente a especificação relativa entre os eventos, a semântica dos relacionamentos especificados na autoria é preservada na apresentação. Por outro lado, nenhuma facilidade para o controle temporal das aplicações é oferecida e, nesse caso, pode ser difícil oferecer funcionalidades associadas à apresentação, mesmo as mais simples, como, por exemplo, controlar se o final de uma apresentação foi atingido.



### A.1.5. Sincronização Baseada em Grafos Temporais

Diferentes modelos de sincronização temporal baseados em grafos podem ser encontrados (Bertion, 1998; Buchanan, 1992; Jourdan, 1998, vanRossum, 1993). Alguns desses modelos são bastante especializados em relação às suas entidades, seu formato ou mesmo em relação a semântica do sincronismo que eles representam. Nesse caso, usualmente, esses modelos possuem sua própria classificação, como é o caso, por exemplo, das árvores utilizadas no modelo hierárquico e das redes de Petri.

O CMIFed utiliza para o controle da apresentação das aplicações uma estrutura denominada grafo de dependências temporais (vanRossum, 1993), contruída a partir da estrutura hierárquica de uma aplicação CMIF. Nesse grafo, os nós representam o início e o fim dos eventos e as arestas as restrições temporais dos relacionamentos entre dois nós. As arestas são obtidas a partir da estrutura de

sincronização hierárquica. Por exemplo, em uma operação de sincronização sequencial, o final de cada filho deve preceder o início do próximo filho existindo, portanto, uma aresta entre essas ações. Arestas também são utilizadas para representar a duração dos eventos e os arcos de sincronização. Quando partes específicas da apresentação das mídias são relacionadas através dos arcos de sincronização, nós específicos são construídos. Esses nós, por sua vez, são temporalmente relacionados, através de outras arestas, a outras partes da mesma mídia.

Dois nós especiais são construídos no grafo de dependências temporais, um representando o início e outro o fim da apresentação de uma aplicação. Uma aresta sem restrições temporais é utilizada para unir o nó de início da apresentação ao nó que representa a raiz da estrutura hierárquica. Os nós que representam as folhas da estrutura hierárquica são unidos, também sem restrições temporais, ao nó de fim da aplicação.

Durante a apresentação, os nós e as arestas do grafo vão sendo marcados. O nó de início da apresentação é o primeiro a ser marcado. Ao ser marcado, as restrições temporais associadas as arestas de saída de um nó geram atrasos que devem ser individualmente respeitados para cada aresta. Quando o tempo relativo a um desses atrasos é atingido, sua aresta é marcada. Quando todas as arestas de entrada de um nó são marcadas, o próprio nó é marcado. A aplicação termina quando o nó de fim da apresentação é marcado.

No grafo de dependências temporais todos os nós correspondentes as folhas da estrutura hierárquica devem ser alcançados, isto é, não existem nós que representam possibilidades que podem ser ou não executadas dependendo, por exemplo, de ações interativas ou adaptativas. Essa estrutura oferece facilidades temporais como a representação do final da apresentação e a indicação de referências circulares (vanRossum, 1993). No entanto, não é possível utilizar diretamente esse grafo no controle de aplicações adaptativas e interativas como, por exemplo, aplicações que podem ser especificadas através de SMIL 3.0 (W3C, 2008b).

Na linguagem SMIL, a estrutura lógica de uma aplicação define a sua estrutura de apresentação. Elementos dessa linguagem representam composições com semântica temporal embutida (paralela, sequencial ou exclusiva) que podem agrupar elementos representando as mídias ou elementos representando outras

composições, recursivamente. Além dessas composições, a linguagem define uma composição voltada à adaptação do conteúdo (*switch*), onde apenas um dos seus elementos constituintes será apresentado (W3C, 2008b). O comportamento temporal das aplicações SMIL também pode ser definido de forma relativa à ocorrência de eventos, sendo possível definir relacionamentos causais entre elementos, de forma complementar a semântica embutida pelas composições. Os eventos associados aos elementos podem ser executados, inclusive, a partir de ações interativas (W3C, 2008b).

Para controlar a apresentação de aplicações SMIL, o exibidor Ambulant (Bulterman, 2004; Cesar, 2006) propõe a construção de uma árvore cujos nós representam todos os elementos da aplicação. As arestas dessa árvore preservam os relacionamentos temporais entre os nós que representam as composições e seus nós filhos, que podem representar as mídias ou outras composições, recursivamente. Outras arestas, representando os relacionamentos existentes entre os eventos da aplicação, também podem existir e, nesse caso, a árvore passa a ser definida como um grafo temporal.

No grafo temporal proposto pelo Ambulant, cada nó possui uma máquina de estados que controla o estado do evento de apresentação da mídia ou da composição associada ao nó. No início da apresentação, o nó correspondente à raiz da árvore tem sua máquina de estados colocada no estado ocorrendo. Quando um nó tem seu estado alterado, essa modificação é notificada aos demais nós relacionados através das arestas. Caso uma aresta tenha alguma condição associada, essa condição deve ser verdadeira para que a notificação aconteça. Uma condição pode ser um atraso temporal, regras em uma adaptação ou ainda um evento interativo.

A estrutura proposta pelo Ambulant é suficiente e eficiente para controlar a apresentação, porém, ela possui características mais próximas da especificação do sincronismo baseada em relacionamentos entre eventos do que da sincronização baseada em grafos. Para calcular os momentos absolutos no tempo relativos à execução dos eventos seria necessário realizar uma simulação da apresentação utilizando essa estrutura, o que não seria viável em tempo de execução da aplicação.

Uma estrutura mais próxima da sincronização baseada em grafos temporais é proposta no sistema Firefly (Buchanan, 1992; Buchanan, 1993; Buchanan,

2005) onde, na verdade, um conjunto de grafos é utilizado para representar os relacionamentos existentes em uma aplicação. A sequência de eventos previsíveis, a partir do evento de apresentação da mídia que inicia a aplicação, forma um grafo nesse sistema denominado principal. Para cada evento imprevisível, cujo tempo de execução não pode ser calculado antes do início da apresentação, um grafo auxiliar é construído. Obviamente, todos os eventos representados em um grafo auxiliar devem ser previsíveis em relação aos demais eventos presentes nesse mesmo grafo.

Durante a apresentação, cada grafo auxiliar pode ser adicionado ao grafo principal tão logo o tempo de execução do evento inicialmente imprevisível que originou esse grafo auxiliar possa ser calculado. Dessa forma, quando o último evento imprevisível da aplicação é conhecido, o grafo temporal é completamente obtido. Esse grafo obtido representa os momentos temporais associados à execução dos eventos em uma aplicação e não à semântica dos relacionamentos entre os eventos. Como exemplo, em aplicações onde eventos em um grafo auxiliar estão relacionados a eventos definidos em outros grafos auxiliares ou ao grafo principal, o grafo obtido pode ser formado por vários eventos com a mesma semântica, porém dispostos em diferentes momentos temporais.

# Apêndice B
# Comandos de Edição

| Comandos | Descrição |
|---|---|
| openBase (baseId, location) | Abre uma base privada existente localizada pelo parâmetro *location*. Se a base privada não existir ou se o parâmetro location não for informado, uma nova base é criada com o identificador *baseId*. O parâmetro *location* deve obrigatoriamente especificar o dispositivo e o caminho onde está a base a ser aberta |
| activateBase (baseId) | Ativa uma base privada aberta |
| deactivateBase (baseId) | Desativa uma base privada aberta |
| saveBase (baseId, location) | Salva todo o conteúdo da base privada em um dispositivo de armazenamento persistente (se disponível). O parâmetro *location* deve obrigatoriamente especificar o dispositivo e caminho para salvar a base |
| closeBase (baseId) | Fecha a base privada e descarta todo o seu conteúdo |
| addDocument (baseId, {uri, id}+) | Adiciona uma aplicação NCL a uma base privada. Os arquivos da aplicação NCL podem ser:<br>a) enviados sem solicitação pela rede de difusão de dados; nesse caso, o par {uri, id} é usado para relacionar um conjunto de caminhos de arquivos especificados na aplicação NCL com suas respectivas localizações no sistema de transporte<br>NOTA: Os conjuntos de pares de referência devem obrigatoriamente ser suficientes para que o middleware possa mapear qualquer referência a arquivos presentes na especificação da aplicação NCL na sua localização concreta na memória do dispositivo receptor.<br>b) recebidos pelo canal de interatividade sob demanda, ou residentes no receptor; para esses arquivos, nenhum par {uri, id} necessita ser enviado, exceto o par {uri, “null”} associado à aplicação NCL, que deverá ser adicionado na base baseId, se a aplicação NCL não for recebida sem solicitação (*pushed file*) |
| removeDocument (baseId, documentId) | Remove uma aplicação NCL de uma base privada |

| | |
|---|---|
| saveDocument (baseId, documented, location) | Salva uma aplicação NCL em um dispositivo de armazenamento persistente (se disponível). O parâmetro *location* deve especificar o dispositivo e o caminho no dispositivo onde a aplicação será salva. Se a aplicação NCL estiver sendo exibida, ele deve primeiro ser parada (todos os eventos no estado *occurring* devem ser parados) |
| startDocument (baseId, documentId, interfaceId, offset, refDocumentId, refNodeId)<br><br>NOTA: O parâmetro offset especifica um valor de tempo. | Inicia a reprodução de uma aplicação NCL em uma base privada, iniciando a apresentação a partir de uma interface específica da aplicação. A referência do tempo transportada no campo *eventNPT* estabelece o ponto de início da aplicação, com respeito à base de tempo NPT do conteúdo *refNodeId* da aplicação *refDocumentId* sendo recebida. Três casos podem ocorrer:<br>1) Se *eventNPT* for maior ou igual ao valor de NPT da base temporal do conteúdo *refNodeId* sendo recebido, espera-se até que NPT atinja o valor dado em *eventNPT* e começa a exibição da aplicação do seu ponto inícial no tempo+offset;<br>2) Se *eventNPT* for menor que o valor de NPT da base temporal do conteúdo *refNodeId* sendo recebido, o início da exibição da aplicação é imediato e deslocado no tempo de seu ponto inicial do valor "*offset*+(NPT–*eventNPT*) segundos";<br>NOTA: Somente nesse caso, o parâmetro offset pode receber um valor negativo, mas *offset*+(NPT–*eventNPT*) segundos deve obrigatoriamente ser um valor positivo<br>3) Se *eventNPT* for igual a 0, a exibição da aplicação é imediata e a partir de seu ponto inicial no tempo+offset |
| stopDocument (baseId, documentId) | Pára a apresentação de uma aplicação NCL em uma base privada. Todos os eventos da aplicação que estão em andamento devem obrigatoriamente ser parados |
| pauseDocument (baseId, documentId) | Pausa a apresentação de uma aplicação NCL em uma base privada. Todos os eventos da aplicação que estão em andamento devem obrigatoriamente ser pausados |
| resumeDocument (baseId, documentId) | Retoma a apresentação de uma aplicação NCL em uma base privada. Todos os eventos da aplicação que foram previamente pausados pelo o comando de edição *pauseDocument* devem obrigatoriamente ser retomados. |
| addRegion (baseId, documentId, regionBaseId, regionId, xmlRegion) | Adiciona um elemento <region> como filho de outro <region> no <regionBase>, ou como filho do <regionBase> (*regionId* = "null") de uma aplicação NCL em uma base privada. |

| | |
|---|---|
| removeRegion (baseId, documentId, regionId) | Remove um elemento <region> de um <regionBase> de uma aplicação NCL em uma base privada. |
| addRegionBase (baseId, documentId, xmlRegionBase) | Adiciona um elemento <regionBase> ao elemento <head> de uma aplicação NCL em uma base privada. Se a especificação XML do regionBase for enviada em um sistema de transporte como um sistema de arquivo, o parâmetro *xmlRegionBase* é apenas uma referência para esse conteúdo. |
| removeRegionBase (baseId, documentId, regionBaseId) | Remove um elemento <regionBase> do elemento <head> de uma aplicação NCL em uma base privada. |
| addRule (baseId, documentId, xmlRule | Adiciona um elemento <rule> ao <ruleBase> de uma aplicação NCL em uma base privada. |
| removeRule (baseId, documentId, ruleId) | Remove um elemento <rule> do <ruleBase> de uma aplicação NCL em uma base privada. |
| addRuleBase (baseId, documentId, xmlRuleBase) | Adiciona um elemento <ruleBase> ao elemento <head> de uma aplicação NCL em uma base privada. Se a especificação XML do ruleBase for enviada em um sistema de transporte como um sistema de arquivo, o parâmetro *xmlRuleBase* é apenas uma referência para esse conteúdo. |
| removeRuleBase (baseId, documentId, ruleBaseId) | Remove um elemento <ruleBase> do elemento <head> de uma aplicação NCL em uma base privada. |
| addConnector (baseId, documentId, xmlConnector) | Adiciona um elemento <connector> ao <connectorBase> de um aplicação NCL em uma base privada. |
| removeConnector (baseId, documentId, connectorId) | Remove um elemento <connector> do <connectorBase> de uma aplicação NCL em uma base privada. |
| addConnectorBase (baseId, documentId, xmlConnectorBase) | Adiciona um elemento <connectorBase> ao elemento <head> de uma aplicação NCL em uma base privada. Se a especificação XML do <connectorBase> for enviada em um sistema de transporte como um sistema de arquivo, o parâmetro *xmlConnectorBase* é apenas uma referência para esse conteúdo. |
| removeConnectorBase (baseId, documentId, connectorBaseId) | Remove um elemento <connectorBase> do elemento <head> de uma aplicação NCL em uma base privada. |
| addDescriptor (baseId, documentId, xmlDescriptor) | Adiciona um elemento <descriptor> ao <descriptorBase> de uma aplicação NCL em uma base privada. |
| removeDescriptor (baseId, documentId, descriptorId) | Remove um elemento <descriptor> do <descriptorBase> de uma aplicação NCL em uma base privada. |

| | |
|---|---|
| addDescriptorSwitch (baseId, documentId, xmlDescriptorSwitch) | Adiciona um elemento <descriptorSwitch> ao <descriptorBase> de uma aplicação NCL em uma base privada. Se a especificação XML do <descriptorSwitch> for enviada em um sistema de transporte como um sistema de arquivo, o parâmetro *xmlDescriptorSwitch* é apenas uma referência para esse conteúdo. |
| removeDescriptorSwitch (baseId, documentId, descriptorSwitchId) | Remove um elemento <descriptorSwitch> do <descriptorBase> de uma aplicação NCL em uma base privada. |
| addDescriptorBase (baseId, documentId, xmlDescriptorBase) | Adiciona um elemento <descriptorBase> ao elemento <head> de uma aplicação NCL em uma base privada. Se a especificação XML do <descriptorBase> for enviada em um sistema de transporte como um sistema de arquivo, o parâmetro *xmlDescriptorBase* é apenas uma referência para esse conteúdo. |
| removeDescriptorBase (baseId, documentId, descriptorBaseId) | Remove um elemento <descriptorBase> do elemento <head> de uma aplicação NCL em uma base privada. |
| addTransition (baseId, documentId, xmlTransition) | Adiciona um elemento <transition> ao <transitionBase> de uma aplicação NCL em uma base privada. |
| removeTransition (baseId, documentId, transitionId) | Remove um elemento <transition> do <transitionBase> de uma aplicação NCL em uma base privada. |
| addTransitionBase (baseId, documentId, xmlTransitionBase) | Adiciona um elemento <transitionBase> ao elemento <head> de uma aplicação NCL em uma base privada. Se a especificação XML do <transitionBase> for enviada em um sistema de transporte como um sistema de arquivo, o parâmetro *xmlTransitionBase* é apenas uma referência para esse conteúdo. |
| removeTransitionBase (baseId, documentId, transitionBaseId) | Remove um elemento <transitionBase> do elemento <head> de uma aplicação NCL em uma base privada. |
| addImportBase (baseId, documentId, docBaseId, xmlImportBase) | Adiciona um elemento <importBase> à base (elemento <regionBase>, <descriptorBase>, <ruleBase>, <transitionBase>, ou <connectorBase>) de uma aplicação NCL em uma base privada. |
| removeImportBase (baseId, documentId, docBaseId, documentURI) | Remove um elemento <importBase>, cujo atributo documentURI é identificado pelo parâmetro documentURI, a partir da base (elemento <regionBase>, <descriptorBase>, <ruleBase>, <transitionBase>, or <connectorBase>) de uma aplicação NCL em uma base privada. |

| | |
|---|---|
| addImportedDocumentBase (baseId, documentId, xmlImportedDocumentBase) | Adiciona um elemento <importedDocumentBase> ao elemento <head> de uma aplicação NCL em uma base privada. |
| removeImportedDocumentBase (baseId, documentId, importedDocumentBaseId) | Remove um elemento <importedDocumentBase> do elemento <head> de uma aplicação NCL em uma base privada. |
| addImportNCL (baseId, documentId, xmlImportNCL) | Adiciona um elemento <importNCL> ao elemento <importedDocumentBase> de uma aplicação NCL em uma base privada. |
| removeImportNCL (baseId, documentId, documentURI) | Remove um elemento <importNCL>, cujo atributo documentURI é identificado pelo parâmetro documentURI, a partir do <importedDocumentBase> de uma aplicação NCL em uma base privada. |
| addNode (baseId, documentId, compositeId, {uri, id}+) | Adiciona um nó (elemento <media>, <context> ou <switch>) a um nó *de composição* (elemento <body>, <context> ou <switch>) de uma aplicação NCL em uma base privada. A especificação XML do nó e seu conteúdo de mídia podem:<br>a) ser enviados sem solicitação pela rede de difusão de dados; nesse caso, o par {uri, id} é usado para relacionar um conjunto de caminhos de arquivos, definidos no XML da especificação do nó, com suas respectivas localizações no sistema de transporte;<br>NOTA: Os conjuntos de pares de referência devem obrigatoriamente ser suficientes para que o *middleware* possa mapear qualquer referência a arquivos, presentes na especificação do nó, na sua localização concreta na memória do dispositivo receptor<br>b) recebidos pelo canal de interatividade sob demanda, ou residentes no receptor; para esses arquivos, nenhum par {uri, id} necessita ser enviado, exceto o par {uri, “null”} associado à especificação XML do nó NCL que deverá ser adicionado em compositeId, caso o XML não seja recebido sem solicitação (*pushed file*). |
| removeNode(baseId, documentId, compositeId, nodeId) | Remove um nó (elemento <media>, <context> ou <switch>) de um nó de composição (elemento <body>, <context> ou <switch>) de uma aplicação NCL em uma base privada. |
| addInterface (baseId, documentId, nodeId, xmlInterface) | Adiciona uma interface (<port>, <area>, <property> ou <switchPort>) a um nó (elemento <media>, <body>, <context> ou <switch>) de uma aplicação NCL em uma base privada. |

| | |
|---|---|
| removeInterface (baseId, documentId, nodeId, interfaceId) | Remove uma interface (<port>, <area>, <property> ou <switchPort>) de um nó (elemento <media>, <body>, <context> ou <switch>) de uma aplicação NCL em uma base privada. A *interfaceId* deve obrigatoriamente identificar um atributo name de um elemento <property> ou um atributo id de um elemento <port>, <area> ou <switchPort>. |
| addLink (baseId, documentId, compositeId, xmlLink) | Adiciona um elemento <link> a um nó de composição (elemento <body>, <context> ou <switch>) de uma aplicação NCL em uma base privada. |
| removeLink (baseId, documentId, compositeId, linkId) | Remove um elemento <link> de um nó de composição (elemento <body>, <context> ou <switch>) de uma aplicação NCL em uma base privada. |
| setPropertyValue(baseId, documentId, nodeId, propertyId, value) | Atribui o valor a uma propriedade. A *propertyId* deve obrigatoriamente identificar um atributo *name* de um elemento <property> ou um atributo *id* de elemento <switchPort>. O <property> ou <switchPort> deve obrigatoriamente pertencer a um nó (elemento <body>, <context>, <switch> ou <media>) de uma aplicação NCL em uma base privada identificada pelos parâmetros. |

Tabela B.1 – Comandos de edição.

| **Identificadores usados nos comandos** | **Descrição** |
|---|---|
| baseId | Identificador de uma base privada. Usualmente, em um sistema de TV digital, uma base privada é associada a um canal de TV |
| documentId | Atributo *id* de um <ncl> de uma aplicação NCL |
| refDocumentId | Atributo *id* de um <ncl> de uma aplicação NCL |
| refNodeId | Atributo *id* de um <media> de uma aplicação NCL |
| regionId | Atributo *id* de um <region> de uma aplicação NCL |
| ruleId | Atributo *id* de um <rule> de uma aplicação NCL |
| connectorId | Atributo *id* de um <connector> de uma aplicação NCL |
| descriptorId | Atributo *id* de um <descriptor> de uma aplicação NCL |
| descriptorSwitchId | Atributo *id* de um <descriptorSwitch> de uma aplicação NCL |
| transitionId | Atributo *id* de um <transition> de uma aplicação NCL |

| | |
|---|---|
| regionBaseId | Atributo *id* de um <regionBase> de uma aplicação NCL |
| ruleBaseId | Atributo *id* de um <ruleBase> de uma aplicação NCL |
| connectorBaseId | Atributo *id* de um <connectorBase> de uma aplicação NCL |
| descritptorBaseId | Atributo *id* de um <descriptorBase> de uma aplicação NCL |
| transitionBaseId | Atributo *id* de um elemento <transitionBase> de uma aplicação NCL |
| docBaseId | Atributo *id* de um <regionBase>, <ruleBase>, <connectorBase>, <descriptorBase>, ou <transitionBase> de uma aplicação NCL |
| documentURI | Atributo documentURI de um <importBase> ou um elemento <importNCL> de uma aplicação NCL |
| importedDocumentBaseId | Atributo *id* de um elemento <importedDocumentBase> de uma aplicação NCL |
| compositeId | Atributo *id* de um <body>, <context> ou <switch> de uma aplicação NCL |
| nodeId | Atributo *id* de um <body>, <context>, <switch> ou <media> de uma aplicação NCL |
| interfaceId | Atributo *id* de um <port>, <area>, <property> ou <switchPort> de uma aplicação NCL |
| linkId | Atributo *id* de um <link> de uma aplicação NCL |
| propertyId | Atributo *id* de um <property>, ou <switchPort> de uma aplicação NCL |

Tabela B.2 – Identificadores utilizados nos comandos de edição.