UNIVERSIDADE FEDERAL FLUMINENSE

MARINA IVANOV PEREIRA JOSUÉ

Preparação de Objetos de Mídia e Efeitos Sensoriais para Formatação de Documentos Mulsemídia

NITERÓI

2021

UNIVERSIDADE FEDERAL FLUMINENSE

MARINA IVANOV PEREIRA JOSUÉ

Preparação de Objetos de Mídia e Efeitos Sensoriais para Formatação de Documentos Mulsemídia

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Doutor em Computação. Área de concentração: Ciência da Computação

Orientador:
DÉBORA CHRISTINA MUCHALUAT-SAADE

Co-orientador:
MARCELO FERREIRA MORENO

NITERÓI

2021

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

J83p Josué, Marina Ivanov Pereira
Preparação de objetos de mídia e efeitos sensoriais para formatação de documentos mulsemídia / Marina Ivanov Pereira Josué ; Débora Christina Muchaluat-Saade, orientadora ; Marcelo Ferreira Moreno, coorientador. Niterói, 2021.
137 f. : il.

Tese (doutorado)-Universidade Federal Fluminense, Niterói, 2021.

DOI: http://dx.doi.org/10.22409/PGC.2021.d.14030243754

1. Aplicação Mulsemídia. 2. Aplicação Multimídia. 3. Televisão Digital. 4. NCL. 5. Produção intelectual. I. Muchaluat-Saade, Débora Christina, orientadora. II. Ferreira Moreno, Marcelo, coorientador. III. Universidade Federal Fluminense. Instituto de Computação. IV. Título.

CDD -

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

MARINA IVANOV PEREIRA JOSUÉ

Preparação de Objetos de Mídia e Efeitos Sensoriais para Formatação de Documentos Mulsemídia

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Doutor em Computação. Área de concentração: Ciência da Computação

BANCA EXAMINADORA

Prof. DÉBORA CHRISTINA MUCHALUAT SAADE - Orientadora, UFF

Prof. MARCELO FERREIRA MORENO - Coorientador, UFJF

Prof. CÉLIO VINICIUS N. DE ALBUQUERQUE, UFF

Prof. FLÁVIA COIMBRA DELICATO, UFF

CARLOS DE SALLES SOARES NETO:78005728387

Assinado de forma digital por CARLOS DE SALLES SOARES NETO:78005728387
Dados: 2021.09.10 19:14:17 -03'00'

Prof. CARLOS DE SALLES SOARES NETO, UFMA

Prof. CELSO ALBERTO SAIBEL SANTOS, UFES

Niterói

2021

Agradecimentos

Agradeço em primeiro lugar a Deus, pelo dom da vida e por iluminar meu caminho ao longo de toda minha trajetória.

Aos meus pais, Luciana e Sebastião, que com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa da minha vida.

À minha irmã, Iollanda, pela motivação e apoio nos momentos difíceis.

Ao Washington, pelo companheirismo, incentivo e compreensão nos momentos em que não pude estar presente.

A meus familiares e amigos que tanto amo, pelo apoio e por me trazerem leveza e sorrisos em meio aos desafios enfrentados ao longo do doutorado. Em especial, Fábio, Eyre e Raphael.

Aos meus orientadores, professora Débora Saade e professor Marcelo Moreno, pela orientação, paciência, dedicação e pelo compartilhamento de seus conhecimentos ao longo desses anos. Obrigada por acreditarem em mim para o desenvolvimento deste trabalho.

Aos colegas do Laboratório MídiaCom, pelos bons momentos e ensinamentos compartilhados. Em especial, à nossa querida Marister.

A todos os professores do Programa de Pós-Graduação em Computação da UFF, que de alguma forma contribuíram para minha formação. E a CAPES, pelo suporte financeiro.

Resumo

Aplicações multimídia tradicionais podem ser enriquecidas com efeitos sensoriais para fornecer experiências imersivas com maior qualidade aos usuários. Essas aplicações são denominadas aplicações mulsemídia, e permitem estabelecer relações de sincronização temporal entre o conteúdo audiovisual e efeitos sensoriais, que estimulam outros sentidos como tato, olfato e paladar. O conteúdo das aplicações pode ser transmitido utilizando uma rede de comunicação, e por isso, a entrega dos dados pode sofrer com atrasos devido a problemas de congestionamento da rede. Além dos objetos de mídia tradicionais, a apresentação de efeitos sensoriais pode também sofrer com atrasos, devido à capacidade dos dispositivos responsáveis pela renderização dos efeitos, e ao tempo de comunicação entre o formatador e o atuador. A fim de evitar falhas de sincronização durante a apresentação de aplicações mulsemídia, esta tese propõe uma operação para preparação de objetos de mídia e efeitos sensoriais. A preparação de elementos que compõem uma aplicação mulsemídia pode ser utilizada em fase de autoria ou execução de um documento mulsemídia. Para a realização da preparação automática, a máquina de apresentação, também chamada formatador mulsemídia, constrói um plano de preparação de objetos de mídia baseado nas condições da rede e no comportamento temporal da aplicação obtido a partir da análise do documento multimídia. Já a preparação automática de efeitos sensoriais é dependente do tipo de efeito a ser renderizado. Em geral, aplicações mulsemídia são especificadas utilizando uma linguagem de autoria específica, como NCL - *Nested Context Language*. A fim de permitir a criação de aplicações NCL com efeitos sensoriais, esta tese propõe uma extensão à linguagem NCL e ao modelo NCM. Como prova de conceito, esta tese implementa a criação automática do plano de preparação no middleware Ginga-NCL. Além disso, são apresentados alguns casos de uso do evento de preparação aplicado tanto a conteúdo de mídia quanto a efeitos sensoriais. O primeiro caso de uso apresenta a preparação de objetos de mídia para dar suporte à aplicação de propaganda direcionada, e os casos de uso seguintes apresentam o evento de preparação aplicado a efeitos sensoriais de luz e aroma. Por fim, uma avaliação da proposta também é apresentada.

Palavras-chave: Aplicação Mulsemídia, Preparação de Objetos de Mídia, Preparação de Efeitos Sensoriais, Formatação Mulsemídia, Aplicações NCL, Eventos NCL, Ginga-NCL.

Abstract

Traditional multimedia applications can be enhanced with sensory effects to provide immersive experiences with more quality to users. These applications are named mulsemedia applications and synchronize sensory effects with audiovisual content to stimulate other human senses, such as touch, smell and taste. When the content that composes the application is delivered over a communication network, some delays may occur due to network congestion problems. Furthermore, delays may occur during the presentation of sensory effects due to the device's capabilities and the communication between the formatter and the actuator. In order to avoid synchronization faults during the presentation of the distributed mulsemedia applications, this thesis proposes the automatic preparation of media objects and sensory effects. The preparation mechanism proposed in this thesis can be used in the authoring or execution phase of a mulsemedia application. To perform the automatic preparation, the mulsemedia presentation machine, also called mulsemedia formatter, builds a preparation plan. Concerning media objects, the preparation plan orchestrates the content buffering and media player instantiation in advance. The automatic preparation of sensory effects is dependent on the type of effect to be rendered. In general, mulsemedia applications are defined using a specific authoring language, as NCL - Nested Context Language. In order to allow the specification of NCL applications with sensory effects, this thesis also proposes an extension to the NCL language and the NCM model. As proof of concept, we implemented the automatic creation of the preparation plan in the Ginga-NCL middleware. Furthermore, we present some use cases of the preparation mechanism applied to media content and sensory effects. Finally, an evaluation of our preparation event proposal is presented.

Keywords: Mulsemedia application, Media content preparation, Sensory effect preparation, Content preparation, Mulsemedia formatting, NCL application, NCL events, Ginga-NCL.

Lista de Figuras

2.1 Exemplo de modelagem temporal utilizando linha do tempo. 17

2.2 Exemplo de modelagem temporal baseada em eventos. 18

2.3 Visão estrutural da aplicação de recomendação de conteúdo. 19

2.4 Grafo temporal da aplicação de recomendação de conteúdo. 20

4.1 Entidades do modelo NCM para representação de nós de conteúdo e nós de efeitos sensoriais 33

4.2 Modelagem de posição com base em coordenadas esféricas. 38

4.3 Modelo de posicionamento do padrão MPEG-V. 39

5.1 Máquina de estados de eventos NCL. 44

5.2 Aplicação NCL com efeito sensorial de luz 50

5.3 Linha do tempo correspondente ao plano de apresentação da aplicação de exemplo. 53

5.4 Linha do tempo correspondente ao plano de preparação da aplicação de exemplo. As regiões em vermelho indicam os intervalos em que as mídias e efeitos estão sendo preparados. 56

6.1 Arquitetura Ginga retirada da recomendação ITU-T H.761 [45]. 58

6.2 Atualização da arquitetura Ginga para suporte à NCL 4.0. 58

6.3 Arquitetura da implementação de referência do Ginga.[1] 60

6.4 Diagrama de classes da implementação dos *players* de efeitos sensoriais. . . 61

6.5 Dispositivos utilizados nesta tese para renderização de efeitos de aroma e luz. 62

6.6 Dispositivo utilizado para renderização do efeito de vento. 63

6.7 Tela da aplicação NCL para calibragem do renderizador de aroma. 64

6.8 Vértices e arestas inseridas no HTG para a os links da Listagem 6.3. 70

6.9 Diagrama com as classes inseridas na implementação de referência do Ginga-NCL para preparação automática. 71

7.1 Grafo temporal da aplicação multimídia de propaganda direcionada. 78

7.2 Grafo temporal da aplicação de propaganda direcionada atualizado com o evento de preparação. 80

7.3 Grafo temporal da aplicação mulsemídia "AppLight" com efeito de luz. . . 85

7.4 Grafo temporal da aplicação mulsemídia "AppWind" com efeito de vento. . 85

7.5 Grafo temporal da aplicação mulsemídia "AppScent" com efeito de aroma. 85

7.6 Grafo temporal da aplicação mulsemídia com efeito de luz, atualizado com o evento de preparação. 88

7.7 Visão estrutural da aplicação mulsemídia de caso de uso. 89

7.8 Tela da ferramenta STEVE para construção da aplicação mulsemídia com efeito de vento e luz. 90

7.9 Tela da ferramenta STEVE para seleção da cor do efeito de luz. 91

7.10 Ambiente de execução da aplicação mulsemídia. 91

7.11 Tempo médio de chaveamento com e sem preparação automática, considerando diferentes taxas de transmissão, realizados no ambiente de teste A1. Os valores do eixo y estão representados em escala logarítmica. 93

7.12 Tempo médio de chaveamento com e sem preparação automática, considerando diferentes taxas de transmissão, realizados no ambiente de teste A2. Os valores do eixo y estão representados em escala logarítmica. 94

7.13 Atraso na apresentação do efeito de aroma, com e sem preparação automática, considerando diferentes usuários. 98

Lista de Tabelas

3.1 Comparação das propostas de trabalhos relacionados. 30

4.1 Áreas funcionais da linguagem NCL 3.0 retirada do livro Programando em NCL 3.0 [65] 35

4.2 Descrição dos atributos do elemento <*effect*> 36

5.1 Novos papéis de condição para conectores e elos NCL. 44

5.2 Novos papéis de ação para conectores e elos NCL. 44

5.3 Especificação temporal do plano de apresentação da aplicação de exemplo. 53

5.4 Especificação temporal do plano de preparação da aplicação de exemplo. . 56

7.1 Plano de apresentação inicial da aplicação de propaganda direcionada. . . . 78

7.2 Plano de preparação inicial da aplicação de propaganda direcionada. 79

7.3 Plano de apresentação inicial da aplicação mulsemídia “AppLight”. 86

7.4 Plano de apresentação inicial da aplicação mulsemídia “AppWind”. 86

7.5 Plano de apresentação inicial da aplicação mulsemídia “AppScent”. 86

7.6 Plano de preparação inicial da aplicação mulsemídia “AppLight”. 86

7.7 Plano de preparação inicial da aplicação mulsemídia “AppWind”. 87

7.8 Plano de preparação inicial da aplicação mulsemídia “AppScent”. 87

7.9 Duração média de preparação considerando preparações simultâneas de conteúdo de vídeo, com taxa de transmissão igual a 20Mbps 95

Lista de Abreviaturas e Siglas

ABNT	:	Associação Brasileira de Normas Técnicas;
BIFS	:	BInary Format for Scenes;
CoAP	:	Constrained Application Protocol
HTG	:	Hypermedia Temporal Graph;
HTML	:	HyperText Markup Language;
IPTV	:	Internet Protocol Television;
ITU	:	International Telecommunication Union;
MPEG	:	Moving Picture Experts Group;
MPEG-2 TS	:	MPEG-2 Transport Stream;
MQTT	:	Message Queuing Telemetry Transport
MSE	:	Media Source Extensions;
NCL	:	Nested Context Language;
NCM	:	Nested Context Model;
SBTVD-T	:	Sistema Brasileiro de TV Digital Terrestre;
SEDL	:	Sensory Effect Description Language;
SEM	:	Sensory Effect Metadata;
SMIL	:	Synchronized Multimedia Integration Language;
QoE	:	Quality of Experience;
QoS	:	Quality of Service;
UPnP	:	Universal Plug and Play
XML	:	eXtensible Markup Language;

Sumário

1 Introdução **1**

1.1 Objetivos 4

1.2 Contribuições da Tese 6

1.3 Estrutura do Texto 7

2 Fundamentação Teórica **9**

2.1 Formatação Multimídia 9

2.2 Formatação Mulsemídia 12

2.3 Grafo Temporal para Formatação de Aplicações Multimídia 16

3 Trabalhos Relacionados **22**

3.1 Preparação de Objetos de Mídia 23

3.2 Sincronização de Efeitos Sensoriais e Objetos de Mídia 26

4 Efeitos Sensoriais em NCL 4.0 **31**

4.1 Proposta de Extensão do Modelo NCM 31

4.2 Proposta de Extensão da Linguagem NCL 33

5 O Evento de Preparação **40**

5.1 Preparação Programada 42

5.2 Preparação Automática 47

5.2.1 Plano de Apresentação 48

5.2.2 Plano de Preparação 49

5.2.2.1 Cálculo do Tempo de Preparação para Objetos de Mídia . 53
5.2.2.2 Cálculo do Tempo de Preparação para Efeitos Sensoriais . 54
5.2.2.3 Criação do Plano de Preparação 55

6 Implementação no Ginga-NCL 57

6.1 Renderização de Efeitos Sensoriais 60
6.2 Calibragem de renderizadores de efeitos sensoriais 63
6.3 Implementação do Evento de Preparação 68

7 Casos de Uso e Avaliação da Proposta 73

7.1 Casos de Uso 73
7.1.1 Propaganda Direcionada 73
7.1.2 Preparação de efeitos sensoriais 80
7.1.3 Aplicação mulsemídia: autoria e execução 88
7.2 Avaliação 90
7.2.1 Preparação de conteúdo de mídia 92
7.2.2 Preparação de efeitos sensoriais 95
7.2.3 Considerações sobre os resultados 98

8 Conclusão 101

8.1 Publicações 103
8.2 Trabalhos Futuros 104

Referências 106

Apêndice A – Alteração em NCL 3.0 para especificação do evento de preparação 112

A.1 NCL30ConnectorCausalExpression.xsd 112
A.2 NCL30ConnectorAssessmentExpression.xsd 116

Apêndice B – Esquema NCL 4.0 **120**

B.1 NCL40Effect.xsd 120

B.2 Alteração no NCL30Descriptor.xsd 120

Capítulo 1

Introdução

O conteúdo dos objetos de mídia que compõem uma aplicação multimídia tradicional, como texto, imagem, áudio e vídeo, tem como foco estimular apenas dois dos sentidos humanos, visão e audição. Com o objetivo de tornar o consumo de conteúdo multimídia uma atividade mais imersiva para o usuário, alguns trabalhos da literatura propõem um novo tipo de aplicação, denominada por Ghinea et al. [34] de aplicação mulsemídia (*MULtiple SEnsorial MEDIA*). Nesta tese, considera-se que uma aplicação mulsemídia é composta por conteúdo audiovisual e um ou mais efeitos sensoriais - como vento, luz e calor, por exemplo - que podem ser apresentados de forma sincronizada.

Para fornecer uma experiência multissensorial ao usuário, inicialmente é preciso que o autor do conteúdo construa um documento mulsemídia que defina os objetos de mídia e efeitos sensoriais que irão compor a aplicação, e como estes elementos estarão sincronizados no tempo e no espaço, através do uso de uma linguagem de autoria. Na literatura, existem diferentes propostas de linguagens para especificação de objetos de mídia e dos relacionamentos entre eles, sendo que umas utilizam o paradigma declarativo, como NCL (*Nested Context Language*), [65], SMIL (*Synchronized Multimedia Integration Language*) [70], HTML (*HyperText Markup Language*) [72], e outras o paradigma procedural, por exemplo, JavaScript [33] e Lua [39].

Em relação à especificação de efeitos sensoriais, algumas propostas [77, 73, 76, 63] utilizam a linguagem SEDL (*Sensory Effect Description Language*) definida pelo padrão MPEG-V [44]. Através dessa linguagem é possível construir arquivos de metadados de efeitos sensoriais, utilizados para controlar os dispositivos atuadores. Apesar do arquivo de metadados de efeitos permitir a descrição dos efeitos a serem apresentados associados a um objeto multimídia, é necessária a criação de um outro documento com a especificação da aplicação multimídia, de tal forma que possa integrar e sincronizar diferentes objetos

de mídia.

Outra maneira de especificar efeitos sensoriais em documentos mulsemídia se dá por meio da identificação dos atuadores presentes na instalação física, como é proposto em [35]. Essa abordagem tem como vantagem possibilitar que o conteúdo multimídia e os efeitos sensoriais sejam especificados em um mesmo documento. Por outro lado, a definição de efeitos sensoriais com base nos atuadores faz com que a aplicação fique dependente do ambiente de execução, de modo que, se ocorrer alguma mudança no modo em que um efeito é implementado, a aplicação mulsemídia também deverá ser modificada. Essa mudança pode ser, por exemplo, a troca de protocolo de comunicação entre o sistema mulsemídia e o dispositivo atuador de efeito [62].

Com o objetivo de fornecer um nível de abstração mais alto na definição de efeitos em um modelo mulsemídia, é interessante representar um efeito sensorial como uma entidade de primeira classe. Desse modo, um efeito pode ser especificado independente dos dispositivos físicos utilizados para sua renderização. Esta é uma das motivações desta tese de doutorado.

Após a especificação do documento mulsemídia, este pode ser transmitido até o usuário final através de uma rede de comunicação, ou disponibilizado localmente em uma mídia de armazenamento. Aplicações mulsemídia podem ser transmitidas através da rede, utilizando o modo *push* [17, 53] ou *pull* [73, 76]. Na entrega em modo *push*, a transmissão dos dados que compõem a aplicação é iniciada pelo servidor, já no modo pull, a requisição para a transmissão é iniciada pelo receptor.

Por fim, o documento mulsemídia é interpretado e executado em um ambiente de apresentação que dê suporte às funcionalidades fornecidas pela linguagem de autoria. Durante a fase de execução, é importante que todas as relações de sincronização definidas pelo autor da aplicação na fase de autoria sejam mantidas. Em geral, nos sistemas multimídia, esta função é desempenhada por um componente chamado formatador (também conhecido como máquina de apresentação, ou do inglês, *presentation engine*)[68]. O formatador também é responsável por funções como carregamento e *parsing* do documento multimídia, acionamento de *players* de mídia e controle das interações do usuário com a aplicação. Já nos sistemas mulsemídia, o formatador também deve ser capaz de se comunicar com os dispositivos atuadores responsáveis pela renderização dos efeitos, e controlar a sincronização de efeitos sensoriais entre si, e entre efeito sensorial e objetos de mídia.

Em aplicações multimídia cujo conteúdo é transmitido através da rede, além das tarefas descritas anteriormente, o formatador deve também garantir que todos os conteúdos

de mídia estarão presentes no dispositivo receptor, no momento em que eles devem ser exibidos. Isto porque, caso ocorram atrasos na entrega do conteúdo dos objetos de mídia, devido a problemas de congestionamento na rede, poderão ocorrer falhas de sincronização, e, consequentemente, degradar a qualidade de experiência do usuário da aplicação.

A fim de reduzir faltas na disponibilidade dos objetos de mídia no momento da sua apresentação, alguns ambientes de execução de aplicações multimídia encontrados na literatura [28, 31, 27, 54, 4] propõem a utilização de mecanismos de pré-busca, também chamados de *pre-fetch*, nos quais o cliente requisita antecipadamente o conteúdo dos objetos de mídia. Ao requisitar um conteúdo antecipadamente, maior a probabilidade de que ele esteja disponível no receptor no momento de exibição, e, consequentemente, as relações de sincronização podem ser mantidas mais facilmente. É importante notar que tal mecanismo só é aplicável para os conteúdos da aplicação que são transmitidos em modo *pull*.

Em geral, as propostas que empregam o mecanismo de pré-busca têm como objetivo decidir qual conteúdo deve ser pré-buscado e o momento em que ele deve ser requisitado. Desse modo, essas propostas não consideram a capacidade de armazenamento necessária para realizar a pré-busca, que pode ser um recurso limitado em certos dispositivos, tais como receptores de TV digital. Uma vez que nas soluções de pré-busca todo o conteúdo do objeto de mídia deve ser transferido antecipadamente, sua implementação pode ser inviável em tais dispositivos com recursos limitados de armazenamento. Além disso, o mecanismo de pré-busca não considera um fator de considerável relevância em dispositivos de capacidade computacional restrita: o processamento necessário para a instanciação de um *player* para a reprodução de um conteúdo pode introduzir atrasos na reprodução e, novamente, degradação da sincronização entre as mídias. Esta é outra motivação desta tese de doutorado.

Assim como ocorre com os objetos de mídia tradicionais, os efeitos sensoriais devem ser apresentados no momento correto durante a execução da aplicação mulsemídia. Nesta tese, consideramos que um efeito sensorial não possui um conteúdo a ser buscado, e dessa forma, os mecanismos de pré-busca não podem ser aplicados sobre estes componentes, a fim de garantir sua disponibilidade de apresentação. Além disso, cada efeito sensorial possui fatores diferentes que influenciam no tempo necessário para ser renderizado e percebido pelo usuário. Por exemplo, para um efeito de vento, o tempo para ser renderizado irá depender das capacidades do atuador, e da intensidade do efeito especificada pelo autor. Já para o efeito de aroma, além das capacidades do atuador, deve-se levar em conta

a distância entre o usuário e o dispositivo, para obter o tempo necessário para o efeito ser percebido pelo usuário. Este é o terceiro problema investigado nesta tese.

1.1 Objetivos

Um dos objetivos desta tese é oferecer um nível mais alto de abstração na definição de efeitos sensoriais em modelos mulsemídia, e consequentemente em linguagens de autoria declarativas. Para isso, esta tese de doutorado propõe a representação de um efeito sensorial como entidade de primeira classe no modelo conceitual, o que permite transformar um modelo multimídia em mulsemídia [46]. Desse modo, o efeito passa a ser definido como um novo tipo de nó, denominado *sensory effect node*, possibilitando que seja especificado em uma aplicação mulsemídia, independente dos dispositivos físicos utilizados para sua renderização.

Na atual versão do modelo *Nested Context Model* (NCM) [67] é fornecido suporte apenas à especificação de aplicações com conteúdos de mídia tradicionais. A linguagem *Nested Context Language* (NCL), baseada em NCM, é a linguagem padrão para codificação de aplicações multimídia no Sistema Brasileiro de TV Digital Terrestre (SBTVD-T) [7] e padrão internacional da *International Telecommunication Union* (ITU-T) [45] para serviços IPTV (*Internet Protocol Television*). Em sua atual versão (NCL 3.0), a construção de aplicações com efeitos sensoriais pode ser feita apenas por meio de nós NCLua, contendo um *script* com os comandos para ativação e desativação de um efeito sensorial. Esta abordagem de implementação do efeito sensorial exige que o autor construa um documento utilizando a linguagem Lua para permitir que o formatador se comunique com os dispositivos atuadores.

Para permitir que efeitos sensoriais sejam especificados em aplicações NCL, esta tese propõe a extensão do modelo NCM com um novo tipo de nó para representar um efeito sensorial e, consequentemente, a inclusão de um novo módulo à linguagem NCL. O novo módulo, chamado de *Effect*, para especificação de efeitos sensoriais foi incluído na versão 4.0 de NCL, definindo um novo elemento de linguagem denominado *effect*. O elemento *effect* possui um conjunto de atributos que permitem configurar as características de apresentação de um efeito sensorial na aplicação mulsemídia NCL. A extensão do *middleware* Ginga-NCL para dar suporte à execução de documentos NCL com efeitos sensoriais também é objetivo desta tese. A proposta de NCL 4.0 está sendo levada ao Fórum SBTVD como proposta à futura geração da TV digital no Brasil, atendendo ao requisito de TV

imersiva[1].

Um formatador de aplicações multimídia ou mulsemídia necessita controlar a apresentação dos objetos de mídia e efeitos sensoriais durante a execução da aplicação. Quando a aplicação multimídia é transmitida em modo *pull*, o formatador pode carregar antecipadamente o conteúdo das mídias. Já em aplicações que utilizam efeitos sensoriais incorporados ao conteúdo audiovisual, é preciso que o formatador considere as características dos dispositivos responsáveis pela renderização dos efeitos para a apresentação correta do conteúdo mulsemídia.

Neste contexto, esta tese propõe um novo tipo de operação que pode ser aplicado tanto a objetos de mídia quanto a efeitos sensoriais, a fim de reduzir, ou até mesmo evitar falhas de sincronização durante a apresentação do conteúdo. Este novo tipo de operação pode ser representado como um novo tipo de evento em modelos conceituais mulsemídia, visando controlar o carregamento de objetos de mídia e ativação de efeitos sensoriais. Propõe-se um mecanismo de preparação de conteúdos, que pode ser explorado na fase de autoria, assim como na fase de execução, sempre que a aplicação necessitar que uma mídia ou efeito sensorial tenha seu conteúdo e processamento preparados para apresentação.

Quando aplicado a objetos de mídia, o evento de preparação inicia quando o *player* correspondente é instanciado e o conteúdo do objeto começa a ser buscado. O evento finaliza quando todo o conteúdo solicitado for carregado no dispositivo receptor, ou quando o *buffer* do *player* estiver completo (cheio), sendo considerada a situação que ocorrer primeiro. Durante a fase de autoria, o autor de aplicações mulsemídia pode definir explicitamente a preparação de um conteúdo de mídia ou efeito sensorial, além de especificar relações de sincronização com base na ocorrência da preparação. Em relação aos efeitos sensoriais, a preparação é definida de acordo com o tipo do efeito e depende também de sua implementação de acordo com o dispositivo atuador.

Além da preparação definida explicitamente pelo autor, esta tese propõe a preparação automática de objetos de mídia e efeitos sensoriais, realizada na fase de execução da aplicação mulsemídia. Este mecanismo envolve a criação de um plano de preparação baseado nos instantes de apresentação dos objetos de mídia e efeitos sensoriais que compõem a aplicação. Durante a fase de análise do documento mulsemídia, o formatador constrói um plano de apresentação, derivado de uma estrutura que representa o comportamento temporal da aplicação. Através do plano de apresentação e das informações da rede de transmissão, o formatador é capaz de prever o tempo de preparação dos objetos de mídia e

[1]https://forumsbtvd.org.br/tv3_0/

construir um plano de preparação automática. Para a preparação automática dos efeitos, o formatador deve consultar as informações dos dispositivos responsáveis pela renderização de efeitos. Essas informações podem estar disponíveis em um arquivo de configuração do ambiente de execução.

Com o objetivo de validar a proposta apresentada nesta tese, o evento de preparação foi adicionado à linguagem NCL 3.0, que utiliza o paradigma de sincronização baseado em eventos. O suporte ao evento de preparação na linguagem NCL, como proposto nesta tese, encontra-se já incorporado ao padrão ABNT (Associação Brasileira de Normas Técnicas) do Ginga-NCL para Perfil D de receptores, publicado em 2018 [7]. Para dar suporte à execução de novas aplicações NCL que empregam o evento de preparação, esta tese também modifica a implementação do *middleware* Ginga-NCL, que é a máquina de apresentação do SBTVD-T.

Diferente do mecanismo de pré-busca proposto na literatura, a preparação de conteúdo considera o espaço de *buffer* disponível no *player* para sua realização, ao invés de obter todo o conteúdo de cada objeto de mídia. Outra vantagem da operação de preparação é que ela é genérica o suficiente para ser aplicada tanto a objetos de mídia quanto a efeitos sensoriais, o que não é proposto por nenhum trabalho da literatura. Para demonstrar tal característica, alguns casos de uso do evento de preparação para formatação de documentos mulsemídia são apresentados no Capítulo 7.

Para avaliar as propostas, esta tese apresenta uma análise da efetividade da preparação de conteúdo de mídia e de efeitos sensoriais na redução de atrasos e falhas de sincronização em aplicações mulsemídia.

1.2 Contribuições da Tese

As contribuições desta tese de doutorado são as seguintes:

1. Extensão do modelo NCM e da linguagem de autoria NCL para dar suporte à especificação de efeitos sensoriais como entidades de primeira classe em documentos mulsemídia.

2. Extensão da máquina de apresentação do Sistema Brasileiro de TV Digital, o Ginga-NCL, para renderização de efeitos sensoriais através do elemento *effect*.

3. Proposta de um novo evento, chamado evento de preparação (*preparation*), empre-

gado em documentos mulsemídia para permitir a preparação antecipada de conteúdos de mídia e efeitos sensoriais a serem apresentados em uma aplicação.

4. Extensão da linguagem de autoria NCL para dar suporte ao evento de preparação em tempo de autoria. Esta proposta já está incorporada ao padrão ABNT[7] utilizado no SBTVD.

5. Proposta de um plano de preparação para dar suporte à preparação automática de objetos de mídia e efeitos sensoriais como parte do processo de formatação de documentos mulsemídia.

6. Implementação do evento de preparação na implementação de referência do Ginga-NCL, máquina de apresentação do Sistema Brasileiro de TV Digital.

7. Suporte à preparação de efeitos sensoriais no *middleware* Ginga-NCL, considerando as características intrínsecas a cada tipo de efeito.

1.3 Estrutura do Texto

Esta tese está estruturada da seguinte forma. O Capítulo 2 apresenta a fundamentação teórica e descreve conceitos base para o entendimento do trabalho. Estes conceitos englobam a formatação de aplicações multimídia e mulsemídia, e as estruturas de dados empregadas para representação do comportamento temporal de aplicações.

O Capítulo 3 discute os principais trabalhos relacionados ao gerenciamento da sincronização entre objetos de mídia, entre efeitos sensoriais e entre conteúdo audiovisual e efeitos. Os trabalhos apresentados neste capítulo empregam mecanismos para proporcionar que os objetos de mídia estarão disponíveis no receptor no momento de sua apresentação, e que os efeitos sensoriais serão apresentados no momento correto.

O Capítulo 4 apresenta as extensões propostas pelo presente trabalho para possibilitar que a linguagem NCL dê suporte à especificação de efeitos sensoriais. As extensões propostas foram incorporadas à nova versão da linguagem - NCL 4.0.

O Capítulo 5 apresenta o evento *preparation* proposto por esta tese com o objetivo de reduzir falhas de sincronização durante a apresentação de aplicações mulsemídia. O evento de preparação pode ser explorado tanto na fase de autoria (preparação programada), quando na fase de execução da aplicação mulsemídia (preparação automática).

O Capítulo 6 descreve a implementação da proposta de preparação e renderização de efeitos sensoriais no *middleware* Ginga-NCL, a fim de dar suporte à preparação automática em uma aplicação NCL.

O Capítulo 7 apresenta alguns casos de uso do evento de preparação e uma avaliação de desempenho da solução proposta. O primeiro caso de uso apresenta a preparação automática aplicada a conteúdos de mídia tradicionais, e os outros três casos de uso apresentados demonstram o uso de preparação aplicado a efeitos sensoriais, mais especificamente, preparação de efeitos de luz, vento e de aroma. Para avaliação da solução proposta, foi analisado o atraso de apresentação de conteúdo de mídia e de efeitos sensoriais para os cenários com e sem preparação automática.

Por fim, o Capítulo 8 apresenta as conclusões da tese, realçando suas contribuições e sugerindo trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Durante a fase de autoria de um documento multimídia, o autor pode especificar diferentes relações de sincronização espaço-temporal entre objetos de mídia, a fim de passar uma informação ao usuário que irá consumir o conteúdo. Estas relações definem o comportamento temporal da aplicação, e caso não sejam mantidas durante a fase de execução do documento, podem fazer com que a informação não seja entendida corretamente pelo usuário, ou até mesmo resultar em uma aplicação diferente da que foi definida pelo autor.

Além da sincronização entre objetos de mídia tradicionais, alguns trabalhos propõem incorporar efeitos sensoriais às aplicações multimídia para estimular outros sentidos humanos, além da visão e audição, nas chamadas aplicações mulsemídia [34]. Neste contexto, este capítulo apresenta os principais conceitos e desafios relacionados à execução de aplicações multimídia e mulsemídia, mantendo as relações de sincronização especificadas pelo autor do documento.

2.1 Formatação Multimídia

Durante a fase de execução de uma aplicação multimídia, é necessário realizar o controle de diferentes elementos que compõem o sistema multimídia, a fim de garantir que o conteúdo seja exibido conforme o que foi especificado pelo autor na fase de autoria. Para isso, alguns sistemas multimídia implementam um componente denominado formatador multimídia (máquina de apresentação multimídia), que é responsável por gerenciar a sincronização entre os objetos de mídia que compõem a aplicação, controlar os *players* de exibição e os eventos de interação, a partir de uma especificação do documento multimídia. Além disso, o formatador multimídia deve ser capaz de lidar com aplicações que permitem adaptação do conteúdo e apresentá-las ao usuário final com um certo nível de QoE (do inglês, *Quality*

of Experience).

Em geral, para realizar o controle da apresentação do documento multimídia, o formatador pode se basear na especificação da aplicação e nas características do ambiente de reprodução. Através da especificação da aplicação, é possível extrair informações do comportamento de apresentação dos objetos de mídia e as relações de sincronização espaço-temporal que existem entre eles. Já as características do ambiente de reprodução são importantes para que o formatador descubra, por exemplo, quais tipos de mídia podem ser reproduzidos, e quanto tempo o conteúdo de um objeto de mídia demora para ser obtido, quando este é transmitido através de uma rede de comunicação.

Alguns trabalhos na literatura [15, 22, 58, 21] derivam a especificação do documento multimídia em uma estrutura de dados que é utilizada para modelar o comportamento temporal da aplicação. Através desta estrutura de dados, o formatador é capaz de inferir os instantes em que os objetos de mídia são apresentados na aplicação multimídia e as informações de sincronização dos objetos.

Costa et al. [22] propõem a modelagem da aplicação multimídia através de um grafo de arestas dirigidas, chamado grafo temporal hipermídia (do inglês, *Hypermedia Temporal Graph* - HTG). O HTG é composto por vértices que representam transições de estado de um evento, e arestas dirigidas que representam os relacionamentos entre os eventos. Uma aresta no HTG pode ser rotulada com uma condição, que deve ser satisfeita, para que uma ação especificada no vértice de destino da aresta seja disparada. Além disso, o grafo temporal também permite definir condições e prioridades entre arestas com origem em um mesmo vértice.

O trabalho de Chung et al. [21] representa o comportamento temporal de aplicações multimídia especificadas em linguagem SMIL (*Synchronized Multimedia Integration Language*) [70], através de uma rede de Petri [55]. Na proposta de [21], a construção da estrutura de dados é realizada pelo gerenciador de entrega do conteúdo que realiza o *parsing* do documento SMIL contendo a especificação da aplicação. Desse modo, o gerenciador de entrega utiliza a rede de Petri para escalonar a entrega dos dados multimídia para o dispositivo cliente. A rede de Petri proposta por [21] modifica o modelo de rede de Petri convencional, especificando as durações e uso de recursos, e possui um conjunto de regras de disparo que rege a semântica do modelo. Diferente da estrutura proposta por [22], a rede de Petri de [21] não dá suporte à representação de indeterminismos como as interações do usuário, e não é expressiva o suficiente para especificar todas as relações semânticas e de tempo da linguagem SMIL.

Em [15], é apresentado um sistema hipermídia chamado Firefly, que contém um componente responsável (entidade *Scheduler*) por controlar as restrições temporais especificadas no documento hipermídia. No Firefly, o *Scheduler* utiliza uma estrutura para descrever o leiaute temporal do documento, que consiste basicamente de uma lista de comandos ordenados no tempo. Os comandos são divididos em dois grupos: *main schedule* e *auxiliary schedules*. O primeiro grupo contém os comandos responsáveis por controlar os eventos síncronos sobre os itens de mídias; e o segundo é composto por comandos para controlar as ações disparadas por eventos assíncronos.

A criação da estrutura para controle temporal da aplicação é realizada quando o usuário do sistema solicita a exibição de um documento hipermídia. Para que o documento hipermídia seja executado, o componente *Scheduler* do sistema Firefly define um instante de ocorrência para cada evento, combinando as restrições de duração fornecidas pelos itens de mídia com as restrições de sincronização temporal especificadas pelo autor do documento.

O formatador proposto em [68] é composto por quatro tipos de elementos - o pré-compilador, o compilador, o executor e os controladores. O pré-compilador é responsável por checar a consistência espacial e temporal do documento hipermídia. Através da especificação da aplicação, o compilador gera a estrutura de dados utilizada pelo formatador para a construção do plano de execução. O plano de execução utilizado no sistema Hyperprop apresentado em [68] é construído com base em um grafo similar a uma rede de Petri. Este grafo é derivado do documento hipermídia especificado através do modelo NCM (*Nested Context Model*), baseado em eventos.

O executor do sistema hipermídia apresentado em [68] é responsável por iniciar os *players* e enviar todas as informações necessárias para criar e controlar a apresentação de cada objeto de mídia que compõe a aplicação. O executor irá decidir qual controlador iniciar com base no descritor da mídia definido na especificação. Por fim, o controlador cria uma representação do objeto de mídia obtendo os dados do objeto através de um servidor, e obtém a lista de operações que determina as mudanças de comportamento que devem acontecer durante a apresentação do objeto na execução do documento.

A especificação temporal do documento multimídia pode ser feita de maneira flexível, com os instantes e durações de apresentação dos objetos podendo variar para preservar as relações entre mídias, ou de maneira fixa, com valores estáticos de duração e instante de apresentação. Quando a especificação temporal é definida de forma flexível, é possível fazer ajustes na apresentação do conteúdo "diminuindo" ou "esticando" o tempo de duração

de alguns objetos de mídia. Esta funcionalidade, que altera a duração dos objetos de mídia é denominada computação de tempo elástico [52], e pode ser implementada pelo formatador.

A especificação temporal flexível também é utilizada na proposta de [16], que define técnicas para construção de escalonadores de recuperação e de apresentação de objetos de mídia. O trabalho de [16] divide o processo de recuperação em duas fases: *(i)* fase que define o plano de apresentação que satisfaz a especificação temporal flexível contida no documento, e *(ii)* fase que gera um plano de recuperação que especifica os instantes em que o cliente deve fazer uma requisição ao servidor pelos objetos que compõem o documento multimídia.

Além das funcionalidades básicas relacionadas ao controle de execução da aplicação multimídia, que devem estar presentes em formatadores multimídia, existem outras funcionalidades características da linguagem de especificação do documento que podem também ser desempenhadas pelo formatador. Por exemplo, o formatador multimídia pode implementar técnicas para verificar a consistência do documento multimídia [26], mecanismos de adaptação de conteúdo ao usuário ou ambiente de reprodução [5], e interface com os *players* de mídia para obter informações sobre a reprodução da mídia [68]. Além disso, nas aplicações multimídia que dão suporte à apresentação distribuída em múltiplos dispositivos, o formatador deve ser capaz de gerenciar as conexões com os outros dispositivos para garantir a reprodução correta da aplicação [66].

Ao especificar uma aplicação multimídia, o autor pode definir pontos de adaptação da aplicação através de objetos de mídia alternativos ou alternativas de exibição de um mesmo conteúdo. Os mecanismos de adaptação são interessantes tanto para adequar o conteúdo ao perfil do usuário, quanto para adequar a aplicação às condições do ambiente de execução. Este tipo de aplicação multimídia exige que o formatador implemente um componente responsável pelo gerenciamento do contexto, que consiste na manutenção dos valores de parâmetros do usuário e do seu ambiente [5].

2.2 Formatação Mulsemídia

Atualmente, as aplicações multimídia têm sido empregadas em diferentes áreas como entretenimento, educação, serviços governamentais, saúde e comércio. Com o objetivo de aprimorar tais serviços e enriquecer a QoE dos usuários do conteúdo multimídia, alguns trabalhos na literatura [17, 74, 56, 57, 53, 63] propõem a incorporação de efeitos sensoriais

à multimídia tradicional. Este novo tipo de aplicação é denominado aplicação mulsemídia [34] e permite que o autor de aplicações especifique a sincronização de efeitos sensoriais com objetos de mídia, estimulando outros sentidos humanos, além da visão e audição, para transmitir uma informação.

Além de modificar a maneira como o autor de aplicações transmite uma informação ao usuário, a apresentação de documentos multimídia com múltiplos efeitos sensoriais introduz alterações no formatador multimídia tradicional. Isso porque, além de coordenar a sincronização entre os objetos de mídia, o formatador também deve ser capaz de manter as relações de sincronismo entre mídia e efeito sensorial e entre dois ou mais efeitos sensoriais. Além disso, o formatador deve ser capaz de coordenar o acionamento dos dispositivos atuadores responsáveis pela renderização de um efeito, além do acionamento dos *players* de mídia tradicionais.

Considerando a autoria de aplicações mulsemídia, várias das propostas presentes na literatura [74, 56, 63] descrevem as características dos efeitos sensoriais e da informação de sincronização dos efeitos através do padrão MPEG-V [44]. O padrão MPEG-V fornece alguns mecanismos como arquivos de descrição de metadados de efeitos sensoriais (SEM - *Sensory Effect Metadata*) e uma linguagem de descrição de efeito (SEDL - *Sensory Effect Description Language*), que especifica certos atributos como duração e intensidade de um efeito.

Um simulador de aplicações multimídia com múltiplos efeitos sensoriais é proposto por Waltl et al. [74], que é capaz de apresentar conteúdo multimídia e simular efeitos sensoriais sincronizados com a aplicação. A arquitetura do simulador é dividida em três camadas: camada de entrada de dados, camada núcleo e a camada de interface. A camada de entrada de dados recebe as informações sobre os arquivos de áudio e vídeo a serem exibidos e os metadados de efeitos sensoriais. Já a camada núcleo contém os módulos responsáveis por processar as informações fornecidas pela camada de entrada. Nessa camada, um *parser* XML extrai as características dos efeitos sensoriais descritos nos metadados e encaminha para o módulo simulador.

Além de enviar os efeitos para o temporizador que irá escalonar a entrega dos efeitos, o simulador carrega os arquivos de áudio e vídeo e os envia para o decodificador. O temporizador irá verificar então se, no tempo de reprodução atual, um efeito deve ser disparado e ativar o dispositivo atuador correspondente. Por fim, a camada de interface com o usuário contém módulos responsáveis pela apresentação do conteúdo, como os *players* de áudio e vídeo, e os dispositivos atuadores. O simulador dá suporte a efeitos de

luz, vento, umidade, vibração, temperatura, jatos de água e aromas.

Da mesma forma que ocorre com aplicações multimídia em um sistema de TV digital, uma aplicação mulsemídia também pode ser transmitida via *broadcast*. Em [17], é proposto um *framework* para transmissão em *broadcast* de aplicações multimídia sincronizadas com estímulos e efeitos sensoriais táteis para fornecer maior sensação de imersão ao usuário do conteúdo. O *framework* proposto em [17] possibilita a reprodução de aplicações com informações que estimulam receptores nervosos na pele (estímulos táteis) e informações sentidas através de força e movimento aplicados nos músculos (estímulos cinestésicos).

A especificação da aplicação mulsemídia, compatível com a proposta de [17], utiliza o padrão MPEG-4 BIFS [64] (*BInary Format for Scenes*), que permite sintetizar e sincronizar os objetos de mídia audiovisuais com os efeitos relacionados ao sentido do tato. Esse documento mulsemídia é processado pelo formatador, que obtém e analisa o grafo de cena BIFS para acionar os *players* responsáveis por reproduzir o conteúdo audiovisual e também os atuadores responsáveis pela renderização dos efeitos. Nos casos em que a aplicação possibilita interação do usuário por meio do tato, os sensores enviam a informação obtida do toque do usuário (força, posição, direção do movimento, por exemplo) para o formatador. O formatador então processa a informação de interação do usuário e realiza alguma ação na apresentação do conteúdo.

A transmissão de aplicações mulsemídia por *broadcast* também é explorada por [53], que analisa a viabilidade da inserção de efeitos sensoriais e interações multimodais no sistemas de TV atuais. De acordo com [53], para que o usuário sinta-se mais imerso no conteúdo fornecido via TV, é importante que diferentes sentidos do corpo humano sejam acionados pela experiência multimídia multissensorial. No estudo apresentado em [53], foi implementada a transmissão de uma partida de futebol onde diferentes dispositivos atuadores são disparados de maneira síncrona para entregar uma combinação de estímulos sensoriais.

A solução proposta por [53] dá suporte a utilização de alguns mecanismos definidos pelo MPEG-V para descrição dos efeitos sensoriais e a sincronização entre efeitos e objetos de mídia. O gerenciamento e a entrega dos diferentes tipos de dados do conteúdo multimídia multissensorial é realizado por um componente denominado *Receiver Gateway*, que pode utilizar os metadados de efeitos sensoriais propostos pelo MPEG-V para gerar os comandos para os dispositivos atuadores.

Outro tipo de efeito sensorial que pode ser sincronizado com o fluxo audiovisual é o

efeito de aroma, conforme apresentado em [56]. O trabalho de Murray et al. [56] analisa a percepção dos usuários sobre a sincronização entre os conteúdos audiovisuais e os conteúdos olfativos. Além disso, os autores analisam a capacidade dos usuários em detectarem falhas de sincronização e o impacto de tais distorções na QoE do usuário. Apesar de a proposta de [56] não especificar o ambiente de formatação do conteúdo, o trabalho destaca alguns desafios para implementação da máquina de execução de aplicações mulsemídia.

Na proposta de [56], a especificação dos efeitos de aromas no eixo temporal segue a recomendação definida em [57], que determina que o tempo mínimo de apresentação entre dois aromas consecutivos deve ser de 5 segundos. Esse requisito pode ser especificado na fase de autoria, e deve ser respeitado pelo formatador do conteúdo. Outro ponto interessante destacado no trabalho é o tempo que o usuário demora para detectar a presença de aromas, após estes serem emitidos pelos atuadores. Esse tempo está diretamente relacionado com a distância do usuário ao dispositivo dispersor do aroma.

Aplicações multimídia podem ser reproduzidas em diferentes plataformas, como TV digital, aparelhos celulares ou computadores. Nessas diferentes plataformas, a máquina de apresentação, responsável por coordenar a exibição do conteúdo, pode ser implementada através de softwares dedicados ou até mesmo como um *plugin* para o *browser* web. O *plugin* proposto em [76] sincroniza os objetos de mídia com os efeitos sensoriais através da análise do tempo de reprodução do conteúdo audiovisual. Se a marcação de tempo atual for igual ao instante de apresentação do efeito sensorial especificado pelos metadados, o *plugin* dispara a renderização do efeito. O instante de apresentação de efeito é especificado através de um intervalo de tempo, pois o *plugin* verifica se existe algum efeito a ser disparado, a cada 30 ms.

Os trabalhos mencionados anteriormente apresentam soluções que permitem simular ou apresentar efeitos sensoriais acoplados a um ambiente de execução específico. Com o objetivo de permitir que o mesmo renderizador de efeitos seja reutilizado por aplicações heterogêneas (videogames, aplicações de realidade virtual, etc.), Saleme et al. [63] propõem um *framework* mulsemídia, denominado PlaySEM SER 2. O *framework* PlaySEM SER 2 utiliza o protocolo MPEG-V apresentado anteriormente, e seu núcleo é composto por três componentes: *Communication Broker*, *Sensory Effects Processing* e *Connectivity Interface*.

O *Communication Broker* é responsável por transmitir os metadados de efeitos (arquivos SEM, do padrão MPEG-V) da aplicação mulsemídia para o PlaySEM SER 2 e prover serviços abstratos para iniciar, pausar, parar e sincronizar o conteúdo multimídia

com os renderizadores de efeitos sensoriais. No PlaySEM, a comunicação entre o *Communication Broker* e a aplicação mulsemídia pode ocorrer utilizando diferentes protocolos, como UPnP (Universal Plug and Play), CoAP (Constrained Application Protocol), MQTT (Message Queuing Telemetry Transport), e webSocket. A conversão dos metadados de efeitos sensoriais para mensagens de controle de dispositivos é realizada pelo *Sensory Effects Processing*. Por fim, o *Connectivity Interface* permite que o PlaySEM SER 2 estabeleça conexões com dispositivos heterogêneos. A conexão com os dispositivos pode se dar por uma porta USB, *Bluetooth* e Wi-Fi.

É importante destacar que a maioria dos trabalhos atuais, relacionados às aplicações multimídia com múltiplos efeitos sensoriais, focam principalmente na análise da percepção humana da qualidade de experiência no consumo deste tipo de conteúdo. Além disso, nenhum destes trabalhos propõe um mecanismo para garantir a apresentação dos efeitos sensoriais no instante esperado, conforme especificado pelo autor da aplicação mulsemídia. Neste contexto, esta tese tem o objetivo de propor um mecanismo para preparar a renderização de efeitos sensoriais, a fim de minimizar falhas de sincronização entre o conteúdo audiovisual e os efeitos sensoriais durante a formatação de documentos mulsemídia.

2.3 Grafo Temporal para Formatação de Aplicações Multimídia

Como dito anteriormente, aplicações multimídia podem ser especificadas através de linguagens declarativas, como NCL (*Nested Context Language*) [65], SMIL (*Synchronized Multimedia Integration Language*) [70] e HTML (*HyperText Markup Language*) [72], ou linguagens procedurais - JavaScript [33] e Lua [39], por exemplo. As linguagens para especificação de aplicações multimídia podem ser baseadas em modelos de sincronização temporal como sincronização baseada em eventos [15], e baseadas em uma linha do tempo (*timeline*)[12].

Em aplicações multimídia que empregam a sincronização temporal baseada em *timeline*, o alinhamento das apresentações de cada conteúdo é feito em um eixo temporal. Dessa forma, é possível posicionar, na linha do tempo, os instantes em que cada conteúdo da aplicação deve iniciar sua apresentação e também finalizá-la, conforme ilustrado na Figura 2.1. Na aplicação *App1* descrita na Figura 2.1, um conteúdo de áudio deve ser apresentado no início da aplicação, e tem duração de 16 segundos. Além disso, um conteúdo de imagem deve ter sua apresentação iniciada junto com a apresentação do con-

teúdo de áudio, com duração de 12 segundos. Por fim, um vídeo será apresentado aos 16 segundos da aplicação, com duração de 12 segundos.

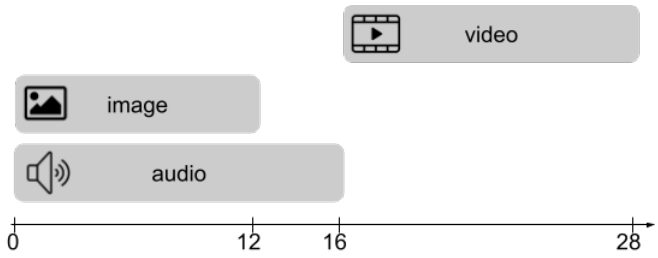

Figura 2.1: Exemplo de modelagem temporal utilizando linha do tempo.

É importante notar que, para especificar aplicações multimídia com sincronização temporal baseada em *timeline*, o autor deve saber o momento exato da exibição de cada conteúdo. Dessa forma, este modelo não pode ser utilizado para representar aplicações cuja apresentação de um conteúdo está condicionada a um evento imprevisível, como por exemplo, uma interação do usuário.

Além do modelo timeline, aplicações multimídia podem também especificar a sincronização temporal com base em eventos. Nesse modelo de sincronização, é possível associar um conjunto de ações a serem executadas em resposta à ocorrência de eventos. Um evento em uma aplicação multimídia pode ser caracterizado como determinístico ou não determinístico [22]. O evento é determinístico quando é possível determinar o instante exato no tempo de sua ocorrência, e não-determinístico caso contrário. Por exemplo, o início ou fim da apresentação de um conteúdo da aplicação podem ser considerados eventos determinísticos. Eventos não-determinísticos podem estar relacionados a ações de interação do usuário, adaptação de conteúdo, atribuição de valores a variáveis do ambiente, ou à existência ou não de múltiplos dispositivos conectados ao dispositivo base. Quando a sincronização baseia-se em eventos, a especificação de uma aplicação pode ser completamente definida de forma relativa, sem referências absolutas de tempo.

A aplicação *App1*, descrita anteriormente, poderia ser representada através da sincronização baseada em eventos, como ilustra a Figura 2.2. Nessa aplicação, o evento de início da apresentação do conteúdo de áudio (especificado por *onBegin* na Figura 2.2) dispara uma ação de início da apresentação do conteúdo de imagem (ação *start* na Figura 2.2). Além disso, o evento de fim da apresentação do conteúdo de áudio (evento *onEnd*) irá disparar uma ação de início da apresentação do conteúdo de vídeo (ação *stop*).

Visando permitir a modelagem temporal da formatação de aplicações multimídia cujo modelo de sincronização é baseado em eventos, alguns trabalhos da literatura [15, 60, 22]

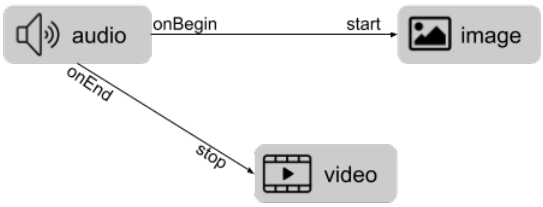

Figura 2.2: Exemplo de modelagem temporal baseada em eventos.

propõem o uso de uma estrutura de dados representada por grafos. Em geral, a modelagem temporal através de grafos utiliza os vértices para representar os eventos que podem ocorrer em uma aplicação, e as arestas representam as relações entre as ocorrências desses eventos.

Costa et al. [22] propõem um modelo de grafo temporal hipermídia (do inglês, *Hypermedia Temporal Graph* - HTG), através de um grafo de arestas dirigidas criado para representar as relações temporais entre os objetos que compõem uma aplicação multimídia. O grafo hipermídia é composto por vértices que representam transições de estado de um evento, e arestas dirigidas que representam os relacionamentos entre os eventos. Além disso, o HTG também permite definir condições e prioridades entre arestas com origem em um mesmo vértice, e todos esse elementos formam uma tupla (V, A, C, N), onde:

- V = $(v_0, v_1, v_2, ..., v_{n-1}, v_n)$ é um conjunto finito de vértices;
- A = $(a_0, a_1, a_2, ..., a_{n-1}, a_n)$ é um conjunto finito de arestas;
- C = c_{ij}, é um conjunto finito de condições de caminhamento associadas às arestas. A condição c_{ij} é associada à aresta (v_i, v_j), e deve ser satisfeita para que ocorra a execução do vértice destino da aresta;
- N = n_{ij} é um conjunto finito de prioridades associadas às arestas. A prioridade n_{ij} está associada à aresta (v_i, v_j).

A Figura 2.3 apresenta a visão estrutural de uma aplicação multimídia para recomendação de conteúdo, composta por três conteúdos de vídeo, e duas imagens. Nessa aplicação, ao fim do vídeo principal (video1) com duração de 60s, são apresentadas duas imagens (*imgOpt1* e *imgOpt2*) representando os vídeos recomendados. Caso o usuário selecione a imagem *imgOpt1*, o vídeo *video2* começará a ser exibido. Porém, se a imagem *imgOpt2* for selecionada, o vídeo *video3* deverá ser exibido. Além disso, quando uma

das imagens são selecionadas, as duas têm a sua apresentação finalizada. A Figura 2.4 apresenta um exemplo de HTG para a aplicação descrita.

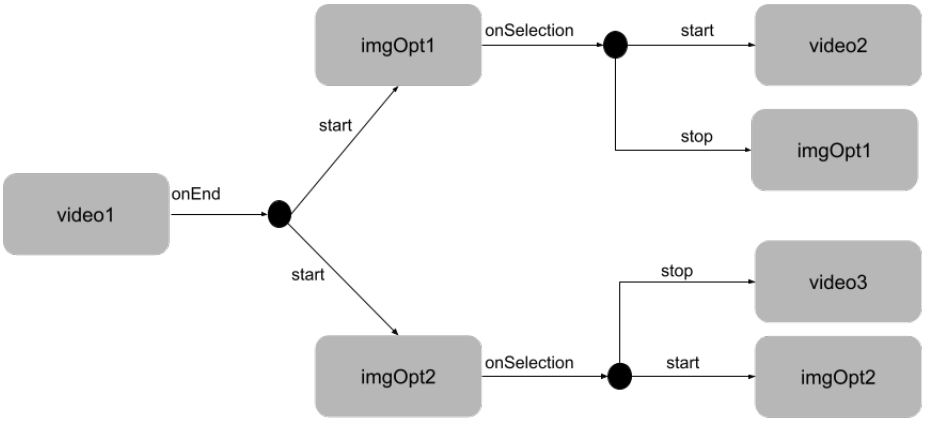

Figura 2.3: Visão estrutural da aplicação de recomendação de conteúdo.

No HTG da aplicação ilustrada na Figura 2.3, foram criados cinco vértices (v_1,v_2,v_3,v_6 e v_7) para representar o evento de início de apresentação das mídias *video1*, *imgOpt1*, *imgOpt2*, *video2* e *video3*, respectivamente. As relações de sincronização entre o vídeo principal e as duas imagens são definidas pelas arestas v_1v_2 e v_1v_3 do grafo temporal. Essas arestas possuem como condição a duração do vídeo principal *video1* (60 segundos). Os eventos de interação também são representados no HTG, como mostram os vértices v_4 e v_5 do grafo. As arestas v_2v_4 e v_3v_5 representam a relação entre a seleção de uma imagem e a apresentação do vídeo correspondente.Os vértices v_8 e v_9 representam o fim da apresentação das imagens com as opções (*imgOpt1* e *imgOpt2*) Por fim, os vértices v_{10}, v_{11} e v_{12} foram inseridos no grafo, derivados da duração implícita dos conteúdos de vídeo (todos os vídeos possuem a mesma duração de 60 segundos). As arestas que não possuem nenhuma condição de caminhamento são rotuladas com o valor "0". E as arestas que possuem como condição uma interação do usuário, foram rotuladas com um valor de tempo "t", pois esta informação não está disponível na fase de análise do documento.

Um grafo temporal hipermídia pode ser dividido em outros subgrafos, denominados cadeias temporais. Uma cadeia temporal do HTG é composta por um conjunto de vértices que representam sequências de transições de eventos previsíveis, que podem ser tanto determinísticas quanto não-determinísticas, a partir de um evento imprevisível. Desse modo, a subdivisão do HTG em cadeias temporais facilita o cálculo dos instantes temporais dos eventos nas aplicações multimídia e o controle da apresentação. Caso o HTG contenha apenas arestas que possuem o tempo como condição associada, o seu caminhamento pode definir uma única cadeia temporal. Por outro lado, em um grafo temporal contendo arestas onde as condições são formadas por ações externas (interação do usuário,

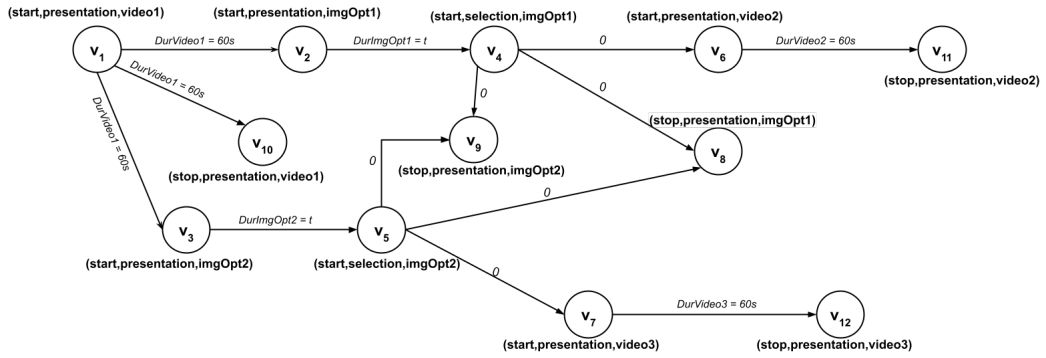

Figura 2.4: Grafo temporal da aplicação de recomendação de conteúdo.

avaliação de variáveis do ambiente ou do usuário, por exemplo), várias cadeias temporais poderão ser construídas. Desse modo, no grafo da Figura 2.4, é possível obter duas cadeias temporais, uma para cada opção de seleção do usuário. A cadeia 1 composta pelos vértices $\{v_1, v_2, v_3, v_4, v_6, v_8, v_9, v_{10}, v_{11}\}$ representa a execução da aplicação quando o usuário seleciona a imagem *imgOpt1*, e a cadeia 2 ($\{v_1, v_2, v_3, v_5, v_7, v_8, v_9, v_{10}, v_{12}\}$) equivale ao cenário em que o usuário seleciona a imagem *imgOpt2*.

O HTG é obtido através da análise do documento multimídia, e pode ser construído tanto pelo servidor do conteúdo quanto pela máquina de apresentação multimídia. Considerando a etapa de envio de uma aplicação multimídia, o servidor pode derivar o HTG em um plano de transmissão, que irá guiar como deve ser realizada a entrega dos objetos de mídia que compõem a aplicação, como é proposto em [22, 47]. Já no lado cliente, o HTG pode ser derivado em uma estrutura para controlar a apresentação da aplicação, chamada plano de apresentação [22].

Nesta tese, serão consideradas as aplicações mulsemídia nas quais a sincronização é baseada em eventos. Dessa forma, o HTG será empregado como estrutura para modelagem do comportamento temporal da aplicação. Para dar suporte à modelagem de aplicações mulsemídia, o HTG foi estendido para permitir a definição de vértices que representam a ocorrência de um evento sobre um efeito sensorial. Além disso, o evento de preparação foi adicionado como uma opção de evento para definição de vértices no grafo. O uso do grafo temporal, e a criação dos planos de apresentação e preparação implementados nesta tese são detalhados no Capítulo 5. Como o HTG pode ser utilizado para representar o comportamento da execução de documentos especificados em diferentes linguagens de autoria, a proposta de preparação automática desta tese pode ser aplicada também a diferentes formatadores hipermídia.

Neste capítulo foram apresentados os principais conceitos relacionados ao tema desta tese. A Seção 2.1 apresentou propostas que têm como objetivo possibilitar o controle da execução de aplicações multimídia. Os trabalhos que dão suporte à formatação de documentos mulsemídia que especificam a apresentação de efeitos sensoriais sincronizados com conteúdo audiovisual foram apresentados na Seção 2.2. Por fim, a Seção 2.3 especificou o grafo temporal hipermídia proposto por Costa et al. [22] para modelagem do comportamento temporal de aplicações multimídia.

Capítulo 3

Trabalhos Relacionados

Os objetos de mídia que compõem uma aplicação multimídia podem estar armazenados localmente no ambiente de execução ou podem ser transmitidos por um servidor localizado remotamente. Nos cenários de aplicações multimídia transmitidas através da rede, para evitar retardos na apresentação do conteúdo e perda de sincronização, faz-se necessária a definição de um mecanismo para garantir que os conteúdos das mídias estejam disponíveis no dispositivo de execução no momento em que devem ser exibidos. É importante destacar que este mecanismo está diretamente relacionado com a qualidade da apresentação da aplicação e qualidade de experiência do usuário.

A qualidade de experiência do usuário de um conteúdo multimídia é um fator importante a ser considerado durante a fase de autoria e execução da aplicação, e por isso é discutido em diferentes trabalhos da literatura. Por exemplo, os trabalhos de [63, 34, 51, 75] propõem adicionar efeitos sensoriais em aplicações multimídia a fim de tornar o conteúdo mais imersivo para o usuário, e consequentemente melhorar a QoE.

Do ponto de vista do formatador, este novo tipo de aplicação impõe certos desafios para a manutenção da sincronização espaço-temporal definida pelo autor do documento, uma vez que, além de garantir que os objetos de mídia encontram-se no dispositivo de exibição, o sistema multimídia também deverá garantir que os atuadores estejam prontos para renderizar os efeitos sensoriais desejados. Neste contexto, este capítulo apresenta alguns trabalhos relacionados à preparação de objetos de mídia para execução (Seção 3.1), além de algumas propostas que buscam garantir a qualidade de apresentação de efeitos sensoriais de maneira sincronizada com conteúdo audiovisual (Seção 3.2).

3.1 Preparação de Objetos de Mídia

Em sistemas de TV Digital que dão suporte à comunicação integrada de conteúdo *broadcast* com conteúdo transmitido via *broadband*, é possível oferecer aplicações multimídia interativas e também personalização de conteúdo multimídia. A fim de permitir a entrega e consumo sincronizado de conteúdos de mídia híbridos (*broadcast/broadband*), Boronat et al. [14] propõem uma plataforma fim-a-fim compatível com o o padrão *Hybrid Broadcast Broadband TV* (HbbTV) (versão 2.0.1) [36]. A plataforma proposta por Boronat et al. [14] dá suporte à entrega de conteúdo de mídia utilizando tecnologias *broadcast*, tal como *Digital Video Broadcasting* (DVB) [30], e via *broadband*, utilizando *Dynamic adaptive streaming over HTTP* (DASH) [40].

Boronat et al. [14] propõem a sincronização dos conteúdos *broadcast* e *broadband* utilizando um esquema Controlador/Agente. Esse esquema de sincronização utiliza um servidor de relógio compartilhado para inserir marcações de tempo em todos os *streams* de mídia e consiste de três etapas: (i) recuperação dos instantes de geração dos *frames* de vídeo (*Timed External Media Information* - TEMI); (ii) rastreamento dos *frames* de vídeo até os elementos renderizadores; e (iii) registrar as marcações de tempo absolutos de apresentação desses *frames* de vídeo. Após extrair a *timeline* TEMI do conteúdo processado e seu tempo estimado de apresentação, o servidor (*Controller*) envia frequentemente a informação para a aplicação multimídia executando no terminal híbrido (*Agent*). Após receber a informação, o receptor compara seu tempo de execução com o tempo do servidor de aplicação e realiza ajustes na reprodução do conteúdo, se for necessário. Diferente da proposta de Boronat et al. [14], que visa garantir a sincronização entre os fluxos *broadcast* e *broadband* através da inserção de marcas de tempo no fluxo, esta tese busca resolver este problema através da preparação do conteúdo *broadband*.

Aplicações multimídia desenvolvidas para a web, em geral, são especificadas através da linguagem HTML (*Hypertext Markup Language*) [72] e permitem a interação do usuário através da seleção de *hyperlinks* (elemento `<a>`). Os *hyperlinks* podem referenciar novas páginas HTML, que podem conter diferentes objetos de mídia a serem carregados e exibidos no navegador web, como outros *links* para novas páginas web. Desse modo, ao selecionar um *link*, o usuário deve esperar até que todas as mídias sejam obtidas a partir do servidor para que a página web de destino seja exibida por completo.

Com o objetivo de prover acesso rápido ao conteúdo de uma aplicação web, alguns trabalhos [69, 23] propõem uma busca antecipada de conteúdo de páginas HTML. Esta

técnica, chamada de *prefetching* consiste em tentar prever qual *hyperlink* tem mais probabilidade de ser selecionado pelo usuário, e armazenar antecipadamente os conteúdos referenciados por ele. Su et al. [69] propõem um esquema que utiliza informações de *logs* de acesso do cliente a um servidor web para calcular os documentos HTML mais prováveis de serem requisitados por um usuário. Após a análise dos *logs*, o servidor pode então alocar um *buffer* no lado cliente e pré-enviar o conteúdo para o usuário usando este *buffer*. Já a proposta de Davison [23] tenta prever a próxima ação do usuário com base na análise do conteúdo textual das páginas requisitadas recentemente pelo usuário. Para a implementação do *prefetching*, o autor Davison [23] constrói um *proxy* para monitorar e salvar o conteúdo HTML e todos os cabeçalhos de requisições e respostas do usuário. Essas informações são usadas para criar um modelo de interesses do usuário, classificar os *links* da página atualmente em exibição e determinar qual deles deve ser pré-buscado.

O mecanismo de pré-busca também é empregado por Meixner et al. [54], para diminuir o atraso e interrupções durante a exibição de hipervídeos. O termo hipervídeo se refere a um conteúdo de vídeo com metadados associados que estabelecem uma relação entre vídeos e informações adicionais. Estes metadados podem ser utilizados pelo *player*, antes da apresentação do vídeo, para obter os próximos conteúdos possíveis de serem exibidos e, assim, realizar a pré-busca. O trabalho de Meixner et al. [54] implementa estratégias de pré-busca em *players* web utilizando HTML5 [72] e *Media Source Extensions* (MSE) [71]. A proposta de Meixner et al. [54] é criar uma estrutura de grafo de cenas do vídeo, a partir da análise dos metadados de um hipervídeo. Cada cena do vídeo pode conter um conjunto de informações relacionadas, como imagens, texto, áudio ou vídeos adicionais.

Através do grafo de cenas proposto por Meixner et al. [54], é possível calcular quais elementos podem ser selecionados pelo usuário no próximo ponto de apresentação do hipervídeo. A pré-busca proposta por Meixner et al. [54] realiza o *download* antecipado apenas da parte da cena que é necessária para iniciar a reprodução do hipervídeo, juntamente com os dados da cena para evitar interrupções durante a apresentação. Como em um hipervídeo pode haver diferentes caminhos de reprodução, Meixner et al. [54] aplicam diferentes probabilidades para cada caminho possível, dependendo do comportamento anterior do usuário ou de outras configurações do hipervídeo definidas nos metadados, para a realização da pré-busca.

Diferente da proposta de Davison [23], que analisa a semântica do conteúdo multimídia, a solução de Figueroa [31] considera o comportamento temporal da aplicação para realizar a pré-busca. Figueroa [31] define a construção de um plano de pré-busca para

aplicações multimídia cuja sincronização é baseada em eventos com duração variável, ou até mesmo não-determinísticos. O plano de pré-busca de [31] é construído na fase de execução da aplicação e especifica os tempos de início para recuperação antecipada de todo o conteúdo dos objetos de mídia, e determina como calcular o tamanho máximo necessário do *buffer*.

As propostas apresentadas nos trabalhos de Davison [23], Su et al. [69] e Figureoa [31] tentam resolver o problema de atraso na exibição de objetos de mídia em aplicações multimídia, utilizando um mecanismo automatizado, sem que o autor da aplicação multimídia tenha controle sobre qual conteúdo deve ser pré-buscado. Entretanto, em alguns casos, o autor da aplicação pode desejar especificar o momento de pré-buscar cada conteúdo. Pensando nisso, o mecanismo proposto nesta tese permite que a preparação, que pode ser considerada uma pré-busca, ainda que parcial do conteúdo, seja controlada tanto pela máquina de apresentação, quanto pelo autor do conteúdo, na lógica da aplicação.

A pré-busca de conteúdo multimídia de aplicações web também é implementada por navegadores como o Mozilla Firefox [32], Google Chrome [20] e Internet Explorer [29]. Em alguns casos, a pré-busca é fornecida como uma característica da linguagem HTML5, e em outros, como uma funcionalidade exclusiva do navegador. Por exemplo, o Firefox utiliza seu tempo ocioso para realizar a pré-busca de *links* HTML5. A pré-busca de *links* consiste em ler as tags `<link>` especificadas no corpo do documento HTML, e que possuem o atributo `rel` com valor igual a *prefetch*. A partir da análise do código HTML, o navegador decide se e quando irá buscar os conteúdos referenciados pela tag `<link>` [27]. Desse modo, o documento contendo a especificação da aplicação pode fornecer sugestões de pré-busca ao navegador. Esta técnica de pré-busca fornecida pelos navegadores web através da marcação de tags `<link>` com o atributo *rel* definido como *prefetch* realiza a busca antecipada de todo o conteúdo referenciado pelo link, e portanto, demanda um consumo de memória maior do que a utilização do evento *preparation* proposto neste trabalho.

A linguagem HTML5 [72] define dois eventos, denominados *canplay* e *canplaythrough*, que são disparados durante o processo de carregamento de um objeto de áudio ou vídeo, permitindo que o autor da aplicação especifique funções a serem disparadas durante este processo. O evento *canplay* é disparado quando a exibição do áudio/vídeo especificado pode ser iniciada pelo browser. Já o evento *canplaythrough* é disparado quando o browser estima que foram carregados dados suficientes para iniciar a reprodução da mídia, e que esta reprodução poderá ocorrer até o fim da duração da mídia sem interrupções na repro-

dução do conteúdo. Os eventos *canplay* e *canplaythrough* definidos no HTML5 podem ser definidos apenas para o conteúdo inteiro de mídias de vídeo ou áudio, e não podem ser especificados sobre uma âncora representando parte do conteúdo. Já o evento *preparation* proposto nesta tese pode ser definido sobre qualquer tipo de mídia ou âncora de conteúdo, além de efeitos sensoriais, como será discutido com detalhes mais adiante.

O padrão HbbTV (do inglês, *Hybrid Broadcast Broadband Television*) [28] também dá suporte à especificação de pré-busca de conteúdo pelo autor da aplicação, que pode definir quais mídias ele/ela deseja carregar previamente. Para isso, o padrão utiliza o atributo *preload* do elemento *media* da linguagem HTML5. No padrão HbbTV [28], quando o atributo *preload* de um elemento *media* é definido como "none", o terminal não deve realizar o pré-carregamento de seu conteúdo de áudio/vídeo. Quando o atributo *preload* é configurado como "metadata", o receptor deve fazer o *download* antecipado do conteúdo de áudio/vídeo para o elemento de mídia em uma taxa reduzida, por exemplo, com uma taxa de transmissão menor do que a de *download* em tempo real. Por fim, caso o atributo *preload* receba o valor "auto", o receptor pode escolher qualquer estratégia de *downloading* antecipado de conteúdo, inclusive, utilizar a largura de banda máxima disponível.

3.2 Sincronização de Efeitos Sensoriais e Objetos de Mídia

Durante a apresentação de uma aplicação mulsemídia, o formatador tem como desafio manter a sincronização entre efeitos sensoriais e objetos de mídia, conforme discutido em alguns trabalhos da literatura [78, 77, 18]. Yuan et al. [78] utilizam "regiões de sincronização", que representam intervalos temporais nos quais efeitos de dispersão de aroma e de sensação de toque devem ser disparados pelos dispositivos atuadores. Quando o efeito é disparado em um instante dentro desta região de sincronização, o usuário considera que o mesmo está sincronizado com o conteúdo visual. De acordo com a análise realizada por [78], o efeito de aroma pode ser renderizado 5 segundos antes do trecho de vídeo ao qual está relacionado, ou até no máximo 3 segundos após o trecho de vídeo, sem que os usuários percebam erros de sincronização. Já para o efeito de toque observou-se um limite de no máximo 1s de atraso na apresentação do efeito, em relação ao conteúdo de vídeo.

Apesar de apresentar os requisitos para sincronização de certos efeitos sensoriais com o conteúdo audiovisual, o trabalho de Yuan et al.[78] não propõe nenhum mecanismo para garantir que o efeito será apresentado no momento esperado. Entretanto, os estudos

apresentados em [78] podem ser utilizados pelo autor da aplicação mulsemídia ou pela máquina de apresentação para disparar o evento *preparation*, proposto neste trabalho, sobre um efeito sensorial específico.

O padrão MPEG-V [44] define uma linguagem de descrição de efeito, denominada SEDL (do inglês, *Sensory Effect Description Language*), que especifica certos atributos, possibilitando criar metadados de efeitos sensoriais (SEM - *Sensory Effect Metadata*). No *framework* proposto por [77], os metadados de efeitos são encapsulados em um fluxo MPEG-2 TS (*Transport Stream*) [43] e podem ser entregues de maneira síncrona ou assíncrona. O trabalho de [77] destaca que o cálculo do instante para envio dos metadados de efeitos deve considerar o tempo necessário para a preparação dos dispositivos sensoriais, além do tempo de transmissão dos metadados na rede e o instante de apresentação do efeito na aplicação.

Na proposta de Yoon [77], o controle do sincronismo é realizado no lado do servidor de conteúdo, que realiza o cálculo do instante correto para envio dos metadados de efeitos sensoriais. No lado do receptor do conteúdo, o ambiente de execução de uma aplicação mulsemídia pode ser composto por diferentes dispositivos atuadores, com características de renderização diferentes, como tempo para disparar um efeito ou posicionamento. Desse modo, o gerenciamento do sincronismo desempenhado no lado servidor, conforme proposto por Yoon [77] pode não atender corretamente a todos os ambientes de reprodução. Neste contexto, é interessante que o controle seja realizado pela máquina de apresentação do conteúdo ou pelo autor da aplicação, conforme proposto nesta tese.

Os mecanismos fornecidos pelo padrão MPEG-V também são utilizados no *framework* para *streaming* 4-D proposto por Choi et al. [18]. Choi et al. [18] propõem utilizar a Parte 2 do padrão MPEG-V, que fornece ferramentas para descrição de capacidade dos dispositivos atuadores e sensores presentes no ambiente de apresentação do *streaming* 4-D. De acordo com [18], essas informações podem ser utilizadas para prover o serviço de *streaming* de maneira apropriada, mantendo-se as relações de sincronismo entre objetos de mídia e efeitos sensoriais. Essas informações podem também ser utilizadas em conjunto com o evento *preparation* para a apresentação de aplicações mulsemídia, preservando as relações de sincronização entre objetos de mídia e efeitos sensoriais, ou entre dois ou mais efeitos sensoriais.

Aplicações mulsemídia podem dar suporte à interação do usuário, permitindo que o mesmo interaja tanto com o conteúdo audiovisual, quanto com um efeito sensorial que está sendo apresentado. Nessas aplicações, o tempo de resposta a uma interação é um

fator importante para a qualidade de experiência do usuário. Santos et al. [62] apresentam uma proposta para melhorar o tempo de resposta em aplicações mulsemídia baseadas em eventos, que executam em um sistema mulsemídia distribuído. Os autores implementaram a proposta na plataforma mulsemídia distribuída PlaySEM [61], que permite a renderização de efeitos sensoriais, especificados de acordo com o padrão MPEG-V [42]. A arquitetura do PlaySEM possui um componente central, denominado *Sensory Effects Renderer* (SER), que se comunica com os dispositivos renderizadores de efeitos.

Além do renderizador de efeitos SER, o PlaySEM possui um componente responsável pela apresentação de conteúdo de vídeo (*Sensory Effects Video Player*), e um módulo microcontrolador, responsável por receber os comandos do SER e enviá-los para os diferentes atuadores. Na proposta de Santos et al. [62], o processo de execução de aplicações mulsemídia interativas com reconhecimento de gesto, no PlaySEM, foi dividido em duas etapas. Na primeira etapa, a aplicação interativa e o PlaySEM SER se comunicam para transmitir e processar todos os *scripts* antes que o usuário comece de fato a interagir com a aplicação. Esses *scripts* contêm os metadados de efeitos sensoriais, do padrão MPEG-V, que especificam informações como o tipo de efeito a ser disparado, a intensidade do efeito, e características específicas do tipo de efeito (por exemplo, cor para o efeito de luz, e tipo de aroma para o efeito de aroma).

O envio dos metadados é feito por uma mensagem do tipo *SetSemEvent*, que transmite o *script* relacionado a um gesto específico (identificado por eventId), através da rede. Na segunda etapa, a aplicação interativa captura os gestos realizados pelo usuário, e caso o gesto esteja relacionado a alguma ação a ser disparada, a aplicação envia uma mensagem *SetPlayEvent* indicando que a execução do efeito pode iniciar. Por fim, o script é executado e os efeitos são renderizados no ambiente do usuário. A proposta de Santos et al. [62] tem como objetivo reduzir a troca de mensagens na rede entre os elementos que compõem a plataforma mulsemídia. Entretanto, o trabalho de Santos et al. [62] não considera o tempo de percepção do efeito para o envio antecipado do comando para os atuadores.

A Tabela 3.1 resume as características dos trabalhos relacionados que buscam garantir a manutenção da sincronização temporal em aplicações multimídia e mulsemídia. Após a análise dos trabalhos relacionados, foi possível observar que nenhum trabalho da literatura propõe uma técnica para garantia de sincronização que se aplica tanto a objetos de mídia quanto a efeitos sensoriais. Já o mecanismo de preparação proposto nesta tese pode ser utilizado tanto para preparação de conteúdo de mídia tradicional, quanto de efeito

sensorial. Em geral, uma aplicação mulsemídia é especificada através de uma linguagem de autoria, que permite especificar os elementos que compõem a aplicação e as relações de sincronização entre eles. A maioria dos trabalhos relacionados utilizam o HTML[72] como linguagem de autoria, e a SEDL [43] para a descrição de efeitos sensoriais. Nesta tese, foi utilizada a linguagem NCL para especificação das aplicações mulsemídia. Entretanto, é importante destacar que o mecanismo de preparação não está restrito a esta linguagem.

Em uma aplicação mulsemídia, o autor da aplicação pode desejar especificar alguma ação no documento em resposta à ocorrência da preparação de uma mídia ou efeito. Além disso, o autor pode desejar definir no documento a necessidade de iniciar a preparação de uma mídia ou efeito em um certo instante. Na literatura, esse suporte à preparação na fase de autoria é fornecido apenas pelos navegadores Mozilla [32], Chrome [20] e Explorer [29] e pelo padrão HbbTV [28]. Durante a fase de execução da aplicação mulsemídia, o formatador pode utilizar as informações do ambiente de execução para auxiliar na manutenção das relações de sincronização temporal.

A pré-busca de conteúdo, em fase de execução, é proposta pelos trabalhos de Davison [23], Meixner et. al. [54], Figueroa [31]. Já em relação à preparação de efeitos sensoriais, os trabalhos de Choi et al.[19] e Santos et al. [63] apresentam propostas para tentar reduzir a apresentação de efeitos sensoriais, também na fase de execução da aplicação. Por fim, os trabalhos de Su et al. [69] e Yoon [77] propõem mecanismos, a serem empregados na fase de transmissão da aplicação, para evitar falhas de sincronização entre efeito sensorial e conteúdo audiovisual. O mecanismo de preparação proposto nesta tese pode ser empregado tanto na fase de autoria da aplicação, quanto na fase de execução.

Neste capítulo foram apresentadas propostas da literatura que têm como objetivo o gerenciamento do sincronismo temporal em aplicações multimídia e mulsemídia. Em relação a aplicações multimídia, a maioria dos trabalhos analisados empregam algum mecanismo de busca antecipada de conteúdo para reduzir atrasos na apresentação da aplicação. Além disso, algumas propostas tentam evitar falhas de sincronização em aplicações mulsemídia, através do envio antecipado de comandos para os dispositivos renderizadores de efeitos. O próximo capítulo irá apresentar as modificações na linguagem NCL propostas por esta tese para dar suporte à especificação de efeitos sensoriais em documentos NCL.

Tabela 3.1: Comparação das propostas de trabalhos relacionados.

Proposta	**Abordagem**	**Linguagem**	**Fase da aplicação**	**Processo automático**	**Suporte a mídia e efeito**
Boronat et al. [14]	Inserção de marcas de tempo no fluxo	HTML	Transmissão e execução	Sim	Apenas mídia
Su et al. [69]	Pré-busca de todo conteúdo da página HTML com base em logs de acesso	HTML	Transmissão	Sim	Apenas mídia
Davison [23]	Pré-busca de páginas HTML	HTML	Execução	Sim	Apenas mídia
Meixner et al. [54]	Pré-busca de hipervídeos com base nos metadados de um hipervídeo específico	HTML	Execução	Sim	Apenas mídia
Figueroa [31]	Pré-busca de todo o conteúdo das mídias	NCL	Execução	Sim	Apenas mídia
Mozilla [32], Chrome [20] e Explorer [29]	Pré-busca dos conteúdos referenciados através de links	HTML	Autoria	Não	Apenas mídia
HTML [72]	Eventos canplay e canplaythrough	HTML	Autoria	Não	Apenas mídia
HbbTV [28]	Atributo preload para indicar se a mídia deve ser preparada	HTML	Autoria	Não	Apenas mídia
Yoon [77]	Envio antecipado de metadados de efeitos sensoriais pelo servidor da aplicação	SEDL	Transmissão	Sim	Apenas efeitos
Choi et al. [19]	Uso de informações de capacidade dos dispositivos para sincronização	SEDL	Execução	Não	Apenas efeitos
Santos et al. [62]	Envio de metadados de efeitos a serem processados, no início da aplicação	SEDL	Execução	Sim	Apenas efeitos
Proposta desta tese	**Preparação de mídia e efeitos sensoriais baseada na análise do comportamento temporal da aplicação mulsemídia**	**NCL**	**Autoria e Execução**	**Sim**	**Mídia e efeitos sensoriais**

Capítulo 4

Efeitos Sensoriais em NCL 4.0

Este capítulo apresenta as contribuições desta tese para a versão 4.0 da linguagem NCL, que engloba o novo módulo para dar suporte às aplicações multimídia com múltiplos efeitos sensoriais. Nessa nova versão, a linguagem NCL pode ainda utilizar componentes já padronizados pelo MPEG-V para a especificação de efeitos e suas características de renderização, através de uma especificação de alto nível, sem a necessidade de uso de *scripts* Lua na aplicação para comunicação com dispositivos atuadores.

A *Nested Context Language* (NCL)[65] é uma linguagem declarativa para especificação de aplicações multimídia baseada no modelo conceitual NCM (*Nested Context Model*) [67]. Dessa forma, para que o NCM também permita a representação de efeitos sensoriais, esta tese propõe uma extensão do modelo. A Seção 4.1 apresenta as extensões realizadas no modelo e a Seção 4.2 descreve os módulos inseridos na linguagem NCL 4.0.

4.1 Proposta de Extensão do Modelo NCM

O modelo NCM especifica um conjunto de entidades para representar um documento multimídia, como nós de mídia, nós de composição e elos para especificar relações entre os nós. Com base no modelo NCM, a linguagem NCL traz uma separação clara entre os conteúdos de mídia e a estrutura de uma aplicação, além de permitir a criação de relações de sincronização de mídia no tempo e no espaço.

Na atual versão do NCM [67], o modelo distingue duas classes básicas de nós, chamados de nós de conteúdo (*content nodes*) e nós de composição (*composite nodes*). Um nó de conteúdo é definido como um fragmento de informação e pode representar um vídeo, áudio, imagem, texto, um conteúdo de script ou até mesmo uma outra aplicação. Dessa

forma, a entidade nó do modelo visa abstrair o conteúdo dos itens de mídia, permitindo que aplicações multimídia sejam descritas de maneira abstrata. Um nó de conteúdo (nó de vídeo ou áudio, por exemplo) é composto de um conjunto de unidades de informação, que depende do tipo do item de mídia representado pelo nó. Por exemplo, um nó de vídeo tem como unidade de informação os *frames* que compõem o vídeo. As unidades de informação de um nó podem ser agrupadas em subconjuntos, definindo uma âncora do nó de conteúdo. A fim de permitir a modelagem de efeitos sensoriais no modelo NCM, esta tese estende o conceito da entidade Nó do modelo, conforme descrito a seguir:

"Um nó pode representar um conteúdo de mídia tradicional (composto por unidades de informação), uma composição de nós e elos ou efeitos sensoriais, que podem estimular os cinco sentidos dos usuários."

Efeitos sensoriais são caracterizados por um conjunto de atributos (propriedades) que podem especificar os estímulos sensoriais percebidos pelos usuários da aplicação mulsemídia. Diferente do conteúdo multimídia, como áudio, vídeo e imagem, nesta tese considera-se que os efeitos sensoriais não são compostos por unidades de informação, uma vez que eles não transmitem nenhum dado ao usuário. Baseado na definição de nó descrita acima, esta tese propõe a criação de um novo tipo de nó para representar um efeito sensorial, denominado *EffectNode*. Um nó de efeito sensorial é portanto, uma entidade de primeira classe que abstrai o tipo do efeito empregado e pode ser especializado em diferentes tipos, como efeito de vento (*WindEffectNode*), efeito de temperatura (*TemperatureEffectNode*), efeito de luz (*LightEffectNode*) e efeito de aroma (*ScentEffectNode*), etc. A Figura 4.1 ilustra a entidade Nó (*Node*) do modelo NCM, e suas especializações, incluindo os efeitos sensoriais propostos nesta tese.

É importante destacar que ao estender o conceito de um nó NCM para representar efeitos sensoriais, é possível agrupá-los em composições para criar efeitos compostos. Por exemplo, um efeito de explosão pode ser definido como um agrupamento dos seguintes tipos de efeitos: *WindEffectNode*, *TemperatureEffectNode*, *LightEffectNode* and *VibrationEffectNode* em um nó de composição. Além disso, como o modelo permite a definição de links para interconectar nós, estes podem ser utilizados para representar relações entre efeitos sensoriais, conteúdo multimídia e também com uma interação do usuário.

O modelo NCM também define uma entidade que agrupa informações das características de apresentação de um elemento, objetivando separar essas informações do conteúdo do documento e de sua estrutura. Essa entidade, chamada descritor genérico (*generic descriptor*), é especializada nas classes descritor (*descriptor*) e *switch* de descritores (*descrip-*

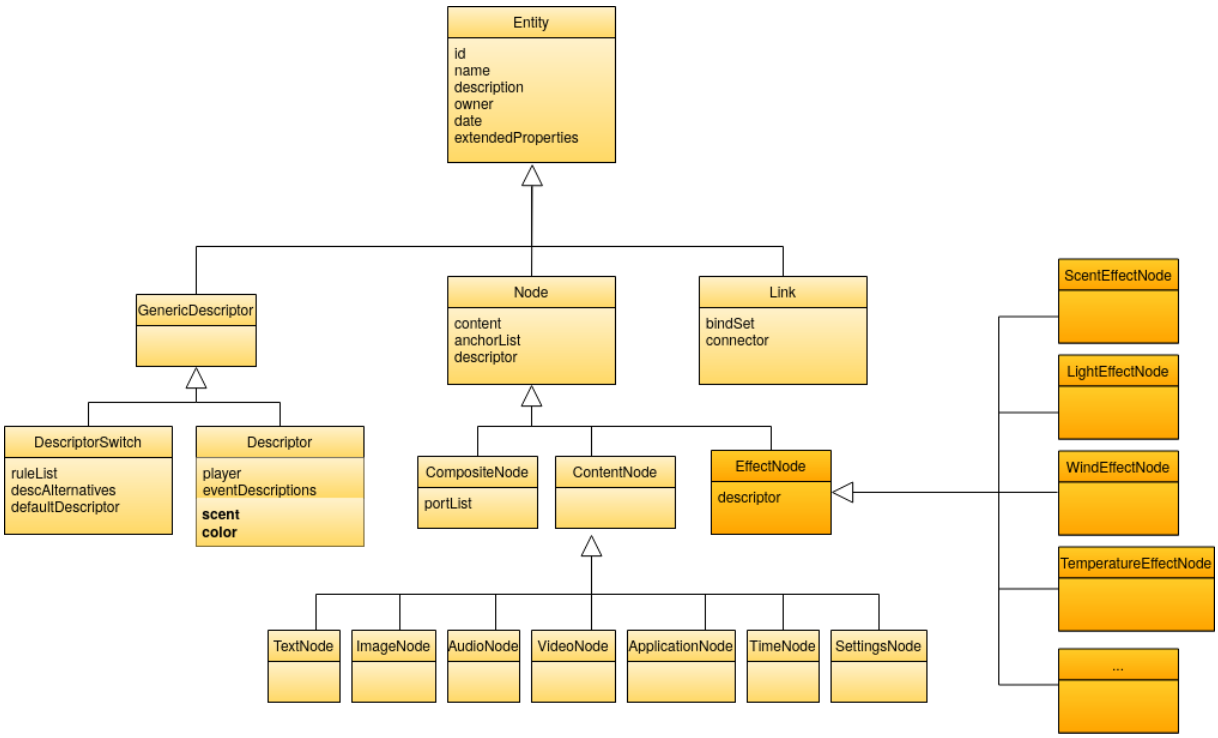

Figura 4.1: Entidades do modelo NCM para representação de nós de conteúdo e nós de efeitos sensoriais

tor switch). O descritor NCM é uma entidade que possui como propriedades adicionais: uma especificação de início de apresentação, uma especificação de fim de apresentação e uma coleção de descrições de eventos. Além desses atributos, esta tese propõe a criação de atributos para caracterizar efeitos sensoriais, como a cor do efeito de luz (propriedade *color*), por exemplo.

A representação de efeitos sensoriais como entidades de primeira classe tem como vantagem a abstração de como o efeito será renderizado no ambiente de execução. Dessa forma, é possível definir um efeito sensorial independente da instalação física na qual a aplicação será apresentada, e reutilizá-lo em diferentes aplicações mulsemídia.

4.2 Proposta de Extensão da Linguagem NCL

A linguagem NCL é especificada de forma modular, onde cada módulo representa uma coleção de elementos, atributos e valores de atributos semanticamente relacionados que representam uma unidade de funcionalidade. Até a escrita desta tese, a linguagem se encontra na versão 3.0 e é dividida em 15 áreas funcionais. A Tabela 4.1 apresenta um subconjunto das áreas funcionais de NCL e seus módulos correspondentes, definidos na versão 3.0 da linguagem.

A estrutura básica de uma aplicação NCL é formada pelo elemento *<ncl>* , e por seus elementos filhos *<head>* e *<body>*. Os objetos de mídia que compõem uma aplicação são apresentados em uma região espacial específica, definida pelo elemento *<region>*. O posicionamento dos objetos de mídia pode ser especificado por um conjunto de atributos do elemento *<region>* (width, height, top, bottom, right, left). A linguagem também dá suporte à especificação de relações espaço-temporais entre os objetos de mídia através dos elementos *<causalConnector>* e *<link>*. A configuração de apresentação de cada objeto de mídia que compõe a aplicação NCL é feita através do elemento *<descriptor>*, que permite especificar propriedades como duração explícita de apresentação, transparência e região de exibição.

A linguagem NCL utiliza o conceito de sincronização baseada em eventos, onde a ocorrência de um evento pode disparar um conjunto de ações dentro do documento. NCL dá suporte a eventos relacionados à apresentação de um objeto de mídia, como o início da apresentação de um vídeo, por exemplo, e também a eventos de interação do usuário através da seleção de teclas. Com o surgimento de sensores e dispositivos conectados à Internet das Coisas (do inglês, *Internet of Things* - IoT), é possível oferecer diferentes possibilidades de interação em plataformas multimídia. Entretanto, essas interações não são suportadas na atual versão da linguagem. Além disso, certas aplicações multimídia podem demandar a identificação do usuário que está consumindo o conteúdo, ou que está interagindo com a aplicação. Em NCL 3.0 não é possível criar este tipo de aplicação, pois não existe nenhum mecanismo de especificação de usuários.

Em aplicações NCL tradicionais (NCL versão 3.0), a informação é transmitida ao usuário utilizando apenas conteúdo audiovisual, (e.g., vídeo, áudio, texto e imagem) especificados através do elemento <media>. Com o surgimento de dispositivos capazes de renderizar efeitos sensoriais (efeito de aroma, de luz e de vibração, etc.), e de se conectar com a televisão através da Internet, as aplicações NCL podem ser enriquecidas para estimular outros sentidos dos usuários, além da audição e visão. Considerando as funcionalidades acima, foi criada uma nova versão da linguagem NCL (NCL 4.0), para dar suporte a aplicações mulsemídia, com multimodalidade e multiusuário [9]. A proposta desta tese engloba apenas os módulos responsáveis pela adição de efeitos sensoriais à linguagem. Os módulos adicionados à linguagem responsáveis pelo suporte à multimodalidade e identificação de usuários na aplicação estão fora do escopo desta tese e podem ser encontrados em [9, 10, 11].

Para que a linguagem dê suporte a especificação de efeitos sensoriais, esta tese propõe a

Tabela 4.1: Áreas funcionais da linguagem NCL 3.0 retirada do livro Programando em NCL 3.0 [65]

Áreas Funcionais	**Módulos**	**Elementos**
Structure	Structure	ncl
		head
		body
Layout	Layout	regionBase
		region
Components	Media	media
	Context	context
Interfaces	MediaContentAnchor	area
	CompositeNodeInterface	port
	PropertyAnchor	property
	SwitchInterface	switchPort
		mapping
Presentation Specification	Descriptor	descriptor
		descriptorParam
		descriptorBase
Linking	Linking	bind
		bindParam
		linkParam
		link
Connectors	CausalConnectorFunctionality	causalConnector
		connectorParam
		simpleCondition
		compoundCondition
		simpleAction
		compoundAction
		assessmentStatement
		attributeAssessment
		valueAssessment
		compoundStatement
	ConnectorBase	connectorBase
Presentation Control	TestRule	ruleBase
		rule
		compositeRule
	TestRuleUse	bindRule
	ContentControl	switch
		defaultComponent
	DescriptorControl	descriptorSwitch
		defaultDescriptor

criação de um novo módulo, denominado *Effect*, adicionado à área funcional *Components*. Dessa forma, os efeitos sensoriais são especificados pelo elemento <*effect*>, e são tratados como entidades de primeira classe em NCL [46]. Isto possibilita a manipulação de efeitos sensoriais de forma similar à manipulação de outros nós em NCL, como objetos de mídia, contextos e *switches*.

O elemento <*effect*> possui um conjunto de atributos, para identificar e caracterizar a ocorrência do efeito sensorial. A Tabela 4.2 descreve esses atributos e sua obrigatoriedade na especificação de um efeito na linguagem NCL 4.0.

Tabela 4.2: Descrição dos atributos do elemento <*effect*>

Atributo	**Descrição**	**Valor**	**Obrigatoriedade**
id	Identifica univocamente o elemento dentro do documento NCL.	Qualquer cadeia de caracteres que comece com uma letra ou um sublinhado ("_") e que contenha apenas letras, dígitos, "." e "_".	Obrigatório
type	Define o tipo do efeito sensorial	Um mimetype que define a classe do efeito. Pode seguir, por exemplo, os tipos de efeitos definidos pelo padrão MPEG-V.	Obrigatório
descriptor	Refere-se a um elemento <descriptor>	Id válido de um descritor do documento NCL.	Opcional

Um elemento de efeito sensorial é especializado pelo atributo *type*, que pode receber um dos valores especificados pelo padrão MPEG-V [44], como *LightType, ScentType, TemperatureType, WindType, VibrationType*, ou pode identificar outros tipos de efeitos disponíveis em uma implementação específica da máquina de apresentação Ginga-NCL. Um efeito sensorial pode ter um conjunto de propriedades, que definem seu comportamento de renderização. Essas propriedades podem ser comuns a todos os tipos de efeitos, ou específicas a um efeito sensorial. Exemplos de propriedades comuns a todos os efeitos sensoriais são a intensidade do efeito (propriedade *intensityValue*) e a faixa de valores de intensidade que o efeito pode atingir (propriedade *intensityRange*). Essas propriedades foram aproveitadas da linguagem para descrição de efeitos sensoriais (do inglês, *Sensory Effect Description Language*) definida pelo padrão MPEG-V [44].

Alguns tipos de efeitos sensoriais possuem propriedades específicas, como a cor de um efeito de luz (propriedade *color*), o aroma dispersado em um efeito do tipo *ScentType* (propriedade *scent*), ou a frequência de oscilação da luz em um efeito do tipo *FlashType*. Tanto as propriedades gerais de efeitos, quanto as propriedades específicas, podem ser

definidas em um documento NCL através do elemento *<property>*, filho de *<effect>*. A Listagem 4.1 apresenta um exemplo de como especificar um efeito sensorial de aroma de rosas. Os aromas podem ser descritos utilizando alguma padronização de descrição de aromas já existente, ou então através de um novo padrão futuro a ser definido para a linguagem NCL. No exemplo abaixo, foi utilizado o padrão de descrição de aromas, definido pelo padrão MPEG-V (anexo 2.4 da ISO/IEC 23005-6) [42].

```
1 <effect id="flowerScent" type="ScentType" descriptor="scentDesc">
2   <property name="scent" value="urn:mpeg:mpeg-v:01-SI-ScentCS-NS:rose"/>
3 </effect>
```

Listagem 4.1: Definição de efeito sensorial de aroma de rosas em NCL.

As propriedades de um efeito sensorial também podem ser especificadas em um elemento *<descriptor>* referenciado no atributo *descriptor* do elemento *<effect>*. A Listagem 4.2 especifica dois descritores, com propriedades relacionadas a efeitos, o primeiro (Linhas 1-3 na Listagem 4.2) contém propriedades de um efeito de aroma, e o segundo (Linha 5 na Listagem 4.2) especifica propriedades de um efeito de luz. Da mesma forma que ocorre com a especificação do aroma, as cores de um efeito de luz devem seguir um esquema pré-definido, ou serem especificadas em RGB. Atualmente, a linguagem NCL já especifica um esquema de cores para definir a fonte de elementos textuais, e também os valores dos atributos *selBorderColor* e *focusBorderColor*. Esses valores podem ser utilizados também para definir a cor do efeito de luz.

```
1 <descriptor id="scentDesc" region="flowerReg" explicitDur="4s" >
2  <descriptorParam name="intensityValue" value="0.9" />
3 </descriptor>
4 <descriptor id="lightDesc" region="lightReg" color="GREEN" />
```

Listagem 4.2: Definição de descritores para efeitos sensoriais.

A posição do efeito sensorial é definida através de novos atributos que foram adicionados ao elemento *<region>* de NCL. Este elemento pode ser usado tanto para especificar a região de exibição de mídias tradicionais ou de renderização de efeitos sensoriais. Para a localização de efeitos sensoriais, é proposto o uso do sistema de coordenadas esféricas, no qual a região começa no ponto indicado pelos atributos *azimuthal* e polar, e os atributos *width* e *height* indicam o tamanho da área a ser utilizada para renderização do efeito. Com esse sistema de coordenadas, o autor de uma aplicação não precisa se preocupar com cantos da instalação física, por exemplo, ou seu tamanho. Considere o exemplo a seguir,

que representa a posição P na Figura 4.2. Essa posição pode ser definida atribuindo os valores -45 e 45 para os atributos de posicionamento polar e azimuthal, respectivamente (Listagem 4.3). A Listagem 4.3 também especifica os atributos de largura (*width*) e altura (*height*), para fornecer uma região de renderização do efeito a partir do ponto P. Os esquemas XML correspondentes aos módulos inseridos na linguagem são apresentados no Apêndice B.

```
1 <region id="marReg"  polar="−45" azimuthal="45" width="5" height="5" />
```

Listagem 4.3: Definição de região com coordenadas polares.

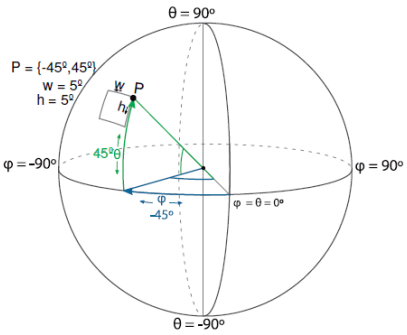

Figura 4.2: Modelagem de posição com base em coordenadas esféricas.

Além disso, o autor pode definir a direção do efeito utilizando o atributo *location*, que segue o modelo espacial utilizado no padrão MPEG-V. A propriedade *location* define a localização do efeito em um espaço 3D. O valor desta propriedade é uma concatenação de posições x, y e z definidas de acordo com o padrão MPEG-V. A Figura 4.3 ilustra todas as posições possíveis dentro do padrão. Por exemplo, para especificar a apresentação de um efeito sensorial, a esquerda no eixo x, no meio do eixo y e à frente no eixo z do ambiente, o autor pode utilizar o valor *left:middle:front* para o atributo *location*, como mostra a Listagem 4.4.

```
1 <region id="marReg"  location="left:middle:front"  />
```

Listagem 4.4: Definição de região com coordenadas esféricas.

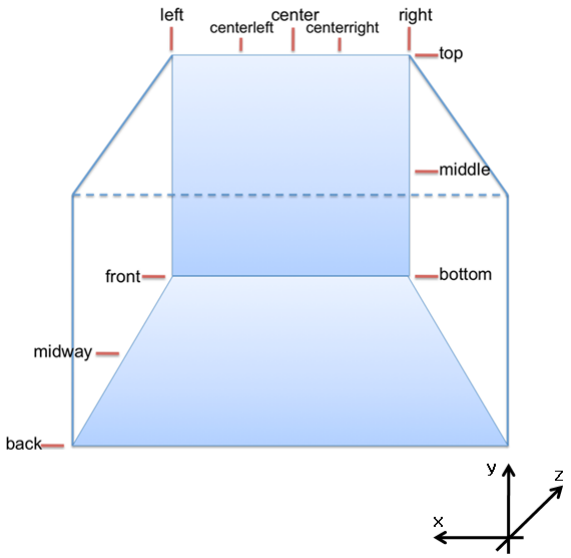

Figura 4.3: Modelo de posicionamento do padrão MPEG-V.

É importante destacar que apesar de NCL ser a linguagem padrão para codificação de aplicações no Sistema Brasileiro de TV Digital Terrestre (SBTVD-T) [7] e padrão internacional da *International Telecommunication Union* (ITU-T) [45] para serviços IPTV (*Internet Protocol Television*), ela pode ser empregada também em contextos diferentes, como em salas de terapia e ambientes multissensoriais [50].

O presente capítulo apresentou a versão 4.0 da linguagem NCL que incorpora um conjunto de novas funcionalidades. Uma das funcionalidades inseridas nessa nova versão é a especificação de efeitos sensoriais em aplicações NCL, que foi proposta por esta tese. Além disso, este trabalho adicionou novos elementos na linguagem para possibilitar a especificação de regiões para renderização de efeitos em uma aplicação através de coordenadas esféricas ou utilizando o modelo de posicionamento do padrão MPEG-V. Por fim, este trabalho também inseriu novas propriedades na linguagem NCL para descrever as características de apresentação de efeitos sensoriais.

Capítulo 5

O Evento de Preparação

A pré-busca de conteúdo é uma técnica interessante para garantir que a apresentação de aplicações multimídia ocorra de modo contínuo, reduzindo a possibilidade de falhas de sincronização. Entretanto, a pré-busca conforme discutida pelos trabalhos comentados no Capítulo 3 demanda uma alta capacidade de armazenamento, que é um recurso limitado em certos equipamentos como dispositivos móveis e receptores de TV digital. Além disso, de acordo com a definição de pré-busca, o mecanismo pode ser aplicado estritamente a objetos de mídia tradicionais, e por isso, não é adequado para reduzir a possibilidade de falhas de sincronização na renderização de efeitos sensoriais em aplicações mulsemídia.

Considerando tais limitações, esta tese propõe a definição de um novo tipo de evento a ser manipulado em aplicações multimídia e mulsemídia, denominado evento de preparação (*preparation*). O evento de preparação é genérico o suficiente para ser aplicado tanto a objetos de mídia tradicionais (áudio, vídeo, texto, por exemplo) quanto a efeitos sensoriais. O evento de preparação deve ser disparado quando um objeto de mídia tiver seu *player* instanciado e seu conteúdo preparado antecipadamente ao momento de sua apresentação na aplicação, ou quando os dispositivos responsáveis pela renderização de um efeito sensorial devem ser pré-ativados para uma melhor sincronização da percepção desse efeito pelo usuário.

Quando o evento de preparação é aplicado a um objeto de mídia, sua ocorrência chega ao fim quando (i) o buffer do *player* de mídia estiver completo ou (ii) o conteúdo da mídia a ser preparada for completamente carregado pelo *player*; o que acontecer primeiro. Esse comportamento é diferente do evento de pré-busca (*prefetch*), que só é finalizado quando todo o conteúdo do objeto de mídia for recuperado a partir de sua origem e não inclui a instanciação do *player* como parte de sua ocorrência.

Dessa maneira, o evento de preparação demonstra-se mais viável em certas aplicações multimídia e independente da capacidade de armazenamento do ambiente de reprodução. Por exemplo, em uma aplicação multimídia na qual o autor especifica ações a serem disparadas ao fim do evento de pré-busca, a ocorrência de tais ações fica condicionada ao espaço de memória disponível no dispositivo em que a aplicação está sendo executada. Entretanto, se ao invés do evento de pré-busca o autor utilizar o evento de preparação, esta situação não irá ocorrer.

Outra vantagem do evento de preparação é que sua definição pode ser facilmente aplicada para efeitos sensoriais, quando tais efeitos são tratados como entidades de primeira classe em uma aplicação mulsemídia [46]. A preparação de um efeito sensorial depende do seu tipo. Por exemplo, para o efeito de calor, seria necessário acionar o dispositivo aquecedor antecipadamente para que a temperatura do ambiente chegue ao valor desejado no instante em que deve ser sincronizado com uma cena em um deserto.

Como um efeito sensorial não contém um conteúdo a ser buscado remotamente, o evento de pré-busca não seria adequado para indicar a necessidade de acionar antecipadamente o dispositivo atuador. Portanto, se o autor da aplicação ou o formatador do conteúdo mulsemídia estiverem cientes do tempo necessário para que um efeito sensorial seja percebido pelo usuário, é possível disparar o evento de preparação sobre o efeito com antecedência, a fim de garantir o sincronismo entre a exibição de mídias e a renderização de efeitos.

O evento de preparação pode ser empregado tanto na fase de autoria quanto na fase de execução da aplicação multimídia. Na fase de autoria, o autor da aplicação multimídia pode especificar o disparo da preparação de um objeto de mídia ou de um efeito sensorial, caso julgue necessário, além de definir ações a serem executadas na aplicação em resposta à ocorrência do evento de preparação. Por exemplo, o autor da aplicação pode especificar que ao finalizar a preparação de um objeto de mídia, a apresentação do mesmo objeto deve ser iniciada. Para que o evento de preparação seja utilizado pelo autor, a linguagem de autoria deve ser estendida para incorporar esta nova funcionalidade. Esta tese propõe a extensão da linguagem NCL [49, 48] para incorporação do evento de preparação. A extensão proposta já foi incorporada à norma ABNT NBR 15606-2 [7], atualizada em 2018 e usada no Sistema Brasileiro de TV Digital.

Já na fase de execução, a máquina de apresentação pode disparar automaticamente a preparação de objetos de mídia e efeitos sensoriais, a fim de evitar retardos na apresentação da aplicação. A preparação, quando especificada pelo autor do conteúdo, é denominada

preparação programada, conforme será discutido na Seção 5.1. A preparação realizada pelo formatador multimídia ou mulsemídia é chamada de preparação automática, de forma transparente para o autor, como será discutido na Seção 5.2.

5.1 Preparação Programada

Como foi dito anteriormente, o evento de preparação pode ser empregado na fase de autoria pelo autor do conteúdo multimídia ou mulsemídia, a fim de programar o preparo de um conteúdo ou efeito sensorial a ser apresentado. Para dar suporte à preparação programada pelo autor da aplicação, esta tese propõe a incorporação do evento de preparação, denominado *preparation*, na linguagem NCL (do inglês, *Nested Context Language*) [45]. A linguagem NCL é uma linguagem declarativa para especificação de aplicações multimídia, e é baseada no modelo conceitual hipermídia NCM (*Nested Context Model*) [67], que permite definir relacionamentos multiponto entre componentes de um documento multimídia, com a semântica de causalidade ou restrição. Nos relacionamentos causais definidos por NCL, uma condição deve ser satisfeita para que uma ou mais ações sejam executadas.

Quando aplicado a objetos de mídia, o evento *preparation* consiste na recuperação de um subconjunto das unidades de informação da mídia, que pode ser especificado em NCL por um elemento `<area>`, ou pelo nó de mídia em si (elemento `<media>`). Já para os efeitos sensoriais, o evento *preparation* permite que o renderizador do efeito seja disparado previamente. Semelhante aos eventos já definidos na linguagem NCL 3.0 (*presentation*, *attribution*, *selection* e *composition*), o evento *preparation* possui os atributos *occurrences* e *repetition*, que podem ser utilizados pelo autor da aplicação ou pelo formatador NCL.

Além dos atributos mencionados acima, esta tese propõe a definição de um novo atributo para o evento de preparação. Este atributo é denominado *prepared*, e permite sinalizar se uma preparação foi bem sucedida ou não. Conforme dito anteriormente, aplicações multimídia podem ser executadas em dispositivos com pouca capacidade de armazenamento, como receptores de TV. Nesses dispositivos, caso a preparação seja feita com muita antecedência, e o dispositivo necessitar de espaço de armazenamento, os recursos alocados para a preparação de uma mídia podem ser liberados. Neste cenário, apesar do atributo *occurrences* do evento de preparação desta mídia estar com valor igual a 1, não significa que o conteúdo da mídia está preparado de fato.

O atributo *prepared* é inicializado com valor “false”, e modificado para “true” quando a

preparação do conteúdo ou efeito é finalizada com sucesso. Caso a preparação tenha sido finalizada com sucesso, e o dispositivo precisar liberar os recursos antes do objeto de mídia ser apresentado, o atributo *prepared* tem seu valor configurado para "false". Dessa forma, o autor de uma aplicação pode utilizar o atributo *prepared* para verificar se um objeto de mídia está preparado, antes de iniciar sua apresentação. A Listagem 5.1 apresenta um exemplo de relação, cujo conteúdo de mídia só será apresentado se sua preparação estiver sido concluída com sucesso.

```
1  <ncl>
2  <head>
3  ...
4  <connectorBase>
5   <causalConnector id="onBeginAndTestStart">
6    <compoundCondition operator="and">
7      <simpleCondition role="onBegin" />
8      <assessmentStatement comparator="eq">
9       <attributeAssessment role="test" eventType="preparation"
10       attributeType="prepared"/>
11      <valueAssessment value="true" />
12     </assessmentStatement>
13   </compoundCondition>
14   <simpleAction role="start" />
15  </causalConnector>
16  ...
17 </head>
18 <body>
19 ...
20  <link xconnector="onBeginAndTestStart">
21   <bind role="onBegin" component= "mainVideo" interface="a1"/>
22   <bind role="test" component="adVideo"/>
23   <bind role="start" component="adVideo"/>
24  </link>
25  ...
26 </body>
27 </ncl>
```

Listagem 5.1: Exemplo de código NCL consultando o atributo *prepared* do evento de

preparação.

Além disso, durante a reprodução da aplicação NCL, o estado do evento *preparation* sobre um nó de mídia, âncora ou efeito sensorial pode se comportar conforme apresentado na Figura 5.1, ou seja, de maneira similar a qualquer outro evento NCL.

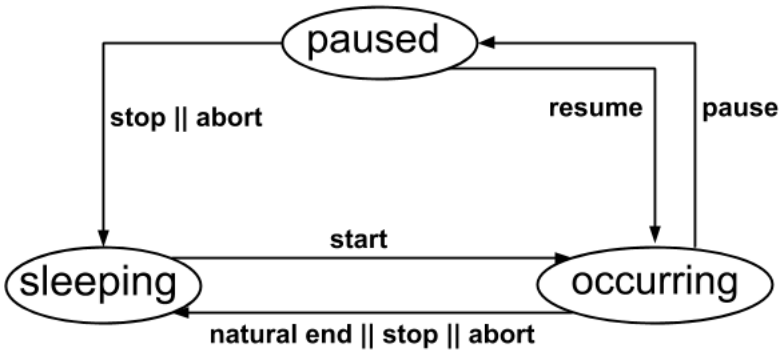

Figura 5.1: Máquina de estados de eventos NCL.

Ao incorporar o evento *preparation* à linguagem NCL, novos papéis predefinidos de condição e de ação são propostos para facilitar a autoria de aplicações, conforme apresentado nas Tabelas 5.1 e 5.2 respectivamente. Esses papéis são utilizados na definição de conectores (elemento `<causalConnector>`) que especificam a semântica das relações entre os objetos de mídia. Para a especificação dos elementos que irão participar das relações, é utilizado o elemento `<link>` da linguagem NCL.

Tabela 5.1: Novos papéis de condição para conectores e elos NCL.

Papéis de condição	**Transição**
onBeginPreparation	*starts*
onEndPreparation	*stops*
onAbortPreparation	*aborts*
onPausePreparation	*pauses*
onResumePreparation	*resumes*

Através dos novos papéis de condição e ação predefinidos para criação dos relacionamentos temporais entre os objetos que compõem a aplicação NCL, o autor pode especificar

Tabela 5.2: Novos papéis de ação para conectores e elos NCL.

Papéis de ação	**Transição**
startPreparation	*starts*
stopPreparation	*stops*
abortPreparation	*aborts*
pausePreparation	*pauses*
resumePreparation	*resumes*

ações em resposta às mudanças da máquina de estados do evento *preparation*. Por exemplo, em uma aplicação de vídeo sob demanda, o autor pode especificar que uma mídia com a mensagem "Loading" seja exibida durante a preparação do conteúdo de uma âncora do vídeo principal. Os conectores que definem a semântica do relacionamento descrito anteriormente são apresentados na Listagem 5.2, e os elos que associam os nós às relações definidas pelos conectores são descritos na Listagem 5.3. Além da relação entre o início do evento de preparação do vídeo principal, com o início do evento de apresentação do vídeo com a mensagem de "Loading" (linhas 11 a 15 da Listagem 5.3), a aplicação também define uma relação de interação do usuário, que ao selecionar a mídia "button" inicia a preparação do vídeo principal (linhas 7 a 10 da Listagem 5.3) .

```
1  <ncl>
2   <head>
3   ...
4      <connectorBase>
5       <causalConnector id="onSelectionStartPreparation">
6          <simpleCondition role="onSelection"/>
7          <simpleAction role="startPreparation"/>
8       </causalConnector>
9       <causalConnector id="onBeginPreparationStart">
10       <simpleCondition role="onBeginPreparation"/>
11       <simpleAction role="start"/>
12      </causalConnector>
13      <causalConnector id="onEndPreparationStartStop">
14       <simpleCondition role="onEndPreparation"/>
15       <simpleAction role="start"/>
16       <simpleAction role="stop"/>
17      </causalConnector>
18     </connectorBase>
19   </head>
20   ...
21 </ncl>
```

Listagem 5.2: Exemplo de código NCL com a especificação de conectores que utilizam o evento *preparation*.

```
1 <ncl>
2   <body>
```

```
3     ...
4     <media id="video" src="video.ogg" descriptor="dVideo"/>
5     <media id="loading" src="load.ogv" descriptor="dLoad"/>
6     <media id="button" src="play.png" descriptor="dButton"/>
7     <link xconnector="onSelectionStartPreparation">
8      <bind role="onSelection" component="button"/>
9      <bind role="startPreparation" component="video"/>
10    </link>
11    <link xconnector="onBeginPreparationStartStop">
12     <bind role="onBeginPreparation" component="video"/>
13     <bind role="start" component="loading"/>
14     <bind role="stop" component="button"/>
15    </link>
16    <link xconnector="onEndPreparationStartStop">
17     <bind role="onEndPreparation" component="video"/>
18     <bind role="start" component="video"/>
19     <bind role="stop" component="loading"/>
20    </link>
21    ...
22  </body>
23 </ncl>
```

Listagem 5.3: Exemplo de código NCL com a especificação de elos que utilizam o evento *preparation*.

A ação *startPreparation* gera a transição *starts* no evento *preparation*, indicada em um papel de condição com o nome *onBeginPreparation*. Ao fim do processo de preparação, é gerada a transição *stops* no evento *preparation*, indicada em um papel de condição com o nome *onEndPreparation*. É importante notar que, se o formatador não implementar nenhum mecanismo de preparação automática, caso a preparação de uma âncora ainda não tenha ocorrido, uma ação de *start* no evento *presentation* daquela mesma âncora deve desencadear uma ação *startPreparation* implícita e automática, disparada pelo formatador. Ou seja, em condições normais, a ação *start* no evento *presentation* sem preparação prévia da âncora gera uma ocorrência do evento *preparation*, ou seja, a sequência de transições *onBeginPreparation*, *onEndPreparation* e *onBegin* da mesma âncora.

5.2 Preparação Automática

Aplicações multimídia e mulsemídia podem ser executadas em diferentes ambientes, com dispositivos e condições de rede diversas. Por exemplo, em um ambiente de execução, pode-se ter um atuador de vento que leva 2 segundos para ser ativado e, em outro ambiente, um atuador que é ativado em menos tempo. Desse modo, torna-se difícil para o autor da aplicação saber, em fase de autoria, o tempo necessário para a preparação de um objeto de mídia, ou de um efeito sensorial, e portanto, uma preparação automática, considerando o ambiente de execução da aplicação faz-se necessária.

Neste contexto, esta tese propõe, além da preparação programada, a preparação automática de conteúdo de mídia e efeito sensorial, que é realizada durante a fase de execução de aplicações mulsemídia. Através da preparação automática é possível reduzir ou até mesmo evitar atrasos na obtenção de objetos de mídia e na renderização de efeitos, durante a apresentação de aplicações multimídia. Para que o formatador dê suporte à preparação automática, é preciso que este disponha das seguintes informações: (i) *"em que momento o objeto de mídia ou efeito sensorial deve ser apresentado na aplicação?"*, e, (ii) *"qual o tempo necessário para preparar um objeto de mídia ou efeito sensorial, dadas as características do ambiente de execução?"*.

Em aplicações multimídia, onde as relações de sincronização entre os objetos de mídia são definidas através do paradigma de *timeline*, os instantes de apresentação dos conteúdos são bem definidos através do posicionamento dos objetos de mídia ao longo do eixo temporal. Para estes casos, o formatador consegue extrair facilmente a informação referente ao momento de apresentação de cada componente da aplicação. Já nas aplicações onde a especificação da sincronização é baseada em eventos, o controle do tempo de apresentação torna-se mais difícil, uma vez que, neste modelo de sincronização, os momentos de ocorrência dos eventos nas aplicações não são explícitos. Portanto, em aplicações deste tipo, o formatador precisa construir uma estrutura com o propósito de modelar o comportamento temporal de aplicações hipermídia, e assim obter os instantes de apresentação dos componentes da aplicação. Os instantes de apresentação de cada objeto que compõe uma aplicação multimídia ou mulsemídia podem ser agrupados em uma estrutura de dados denominada plano de apresentação [22].

O tempo necessário para preparar um objeto de mídia depende das condições da rede, e do tamanho do *buffer* do *player* responsável pela execução da mídia. Diferente da preparação de um conteúdo de mídia, o cálculo do tempo de preparação de um efeito

sensorial irá depender do tipo de efeito. Por exemplo, para um efeito de luz, o tempo de preparação é composto pelo tempo gasto para enviar o comando de ativação até o atuador, mais o tempo gasto por este para processar a mensagem de ativação e o efetivo acionamento do dispositivo. Já a preparação de um efeito de aroma, deve considerar, além dos fatores descritos para o efeito de luz, o tempo para que o aroma chegue até o usuário, de acordo com a distância em que este se encontra do atuador.

Uma vez que o formatador é capaz de computar o tempo para preparar cada componente, e seu tempo de apresentação na aplicação, é possível obter o instante no tempo em que cada componente deve ser preparado. Desse modo, esta tese propõe a criação de um plano de preparação por parte do formatador, que irá conter os instantes em que cada componente da aplicação multimídia ou mulsemídia deve ser preparado para apresentação. Através deste plano, o formatador é capaz de realizar a preparação automática de conteúdo de mídia e efeitos sensoriais, e evitar que ocorram falhas de sincronização durante a apresentação da aplicação. As estruturas de dados empregadas pelo formatador para a construção do plano de preparação são apresentadas nas seções a seguir.

5.2.1 Plano de Apresentação

Durante a fase de autoria de uma aplicação multimídia, o autor pode especificar como os objetos de mídia devem se comportar ao longo tempo, seu posicionamento no espaço, e as relações de sincronização espaço-temporal entre eles. Durante a fase de execução, para garantir uma apresentação de qualidade, é importante que o formatador mantenha essas relações, e para isso, pode ser utilizado um plano de apresentação.

O plano de apresentação é uma estrutura derivada do grafo temporal hipermídia. Ele contém os instantes de ocorrência previsíveis das ações sobre o evento de apresentação de cada objeto de mídia que compõe a aplicação. Desse modo, ao observar um plano de apresentação, é possível verificar o instante no tempo de apresentação em que um objeto inicia (transição *start* no evento *presentation*) e finaliza sua exibição (transição *stop* no evento *presentation*). De acordo com Costa et al. [22], o plano de apresentação possibilita posicionar uma apresentação em um instante qualquer no tempo, avançando, retrocedendo ou retomando uma aplicação em um ponto específico de sua duração.

O grafo temporal hipermídia possui uma cadeia principal, representada pelos vértices que podem ser atingidos pelo ponto de entrada da aplicação, a partir de arestas que têm apenas o tempo como condição associada. Uma cadeia secundária é construída quando uma aresta que possui como condição associada, ações externas ou variáveis não associ-

adas ao tempo, é encontrada. Inicialmente, o plano de apresentação é formado apenas pelas especificações de tempo associadas à cadeia principal. Isto porque, em aplicações multimídia que possuem eventos imprevisíveis, conforme o nome já diz, não é possível definir a priori o momento de ocorrência do evento, e se ele irá ou não ocorrer. Desse modo, o plano de apresentação de aplicações que contenham eventos imprevisíveis deve ser construído de maneira progressiva, durante uma apresentação da aplicação. Por exemplo, em uma aplicação onde a seleção de um botão, por parte do usuário, inicia a apresentação de um conjunto de objetos de mídia, os instantes no tempo para apresentação de cada objeto só poderão ser obtidos no momento em que o usuário selecionar o botão.

Como a preparação de um objeto é uma tarefa a ser realizada antes da apresentação do mesmo, não é possível saber a priori se um evento imprevisível irá ocorrer ou não. Por esta razão, esta tese cria um plano de apresentação inicial considerando que todos os eventos imprevisíveis irão ocorrer. Essa abordagem garante que os objetos de mídia, cujos disparos estejam relacionados a eventos imprevisíveis, sejam apresentados corretamente, sem sofrer atrasos. Entretanto, em alguns casos, esse método pode levar a desperdício de memória, e exige que um mecanismo de liberação de recursos seja aplicado nos casos em que uma preparação antecipada torna-se desnecessária na aplicação.

5.2.2 Plano de Preparação

Como foi dito anteriormente, para que o formatador seja capaz de executar a preparação automática de conteúdo de mídia e efeitos sensoriais, é necessária a criação de um plano de preparação. Em relação à preparação de conteúdo de mídia, o plano de preparação proposto nesta tese leva em consideração o espaço disponível no *buffer* do *player* de mídia e as condições da rede de transmissão para prever quando iniciar o evento de preparação automaticamente, evitando que o autor precise definir a preparação explicitamente em momento de autoria.

A partir do plano de apresentação de uma aplicação multimídia ou mulsemídia, é possível inferir os instantes em que cada componente da aplicação deve ser exibido. Por exemplo, considere uma aplicação mulsemídia composta de três conteúdos de vídeo, efeito de luz e efeito de aroma. Inicialmente, é apresentado um efeito de luz laranja, refletindo a cor predominante no conteúdo audiovisual sendo exibido na tela. Durante a execução da aplicação, o efeito de luz tem a sua propriedade "color" alterada, sincronizada com a troca entre os conteúdos de vídeo que são apresentados na tela. A Figura 5.2 apresenta três momentos distintos na aplicação NCL proposta, que é especificada na Listagem 5.4. Além

(a) Efeito de luz laranja

(b) Efeito de luz verde

(c) Efeito de luz azul

Figura 5.2: Aplicação NCL com efeito sensorial de luz

do efeito de luz, a aplicação apresenta um efeito de aroma sincronizado com o terceiro vídeo. O efeito de aroma é renderizado pelo dispositivo, no momento em que o terceiro vídeo começa a ser exibido.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <ncl id="aplTesteEffect" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
3  <head>
4     <regionBase>
5       <region id="rg1" width="100%" height="100%"/>
6       <region id="rg2" location="center:top:front" />
7       <region id="rg3" location="center:top:front" />
8        </regionBase>
9     <descriptorBase>
10        <descriptor id="desc1"  region="rg1" explicitDur="12s"/>
11      <descriptor id="descL"  region="rg2" intensityValue="100"
12      intensityRange="[0,10]"/>
13      <descriptor id="descS" region="rg3"/>
14    </descriptorBase>
15   <connectorBase>
16      <causalConnector id="onBeginStart">
17        <simpleCondition role="onBegin" />
```

```
18        <simpleAction role="start" max="unbounded"/>
19      </causalConnector>
20      <causalConnector id="onBeginSet">
21      <connectorParam name="var"/>
22        <simpleCondition role="onBegin" />
23        <simpleAction role="set" max="unbounded" value="$var"/>
24      </causalConnector>
25      <causalConnector id="onEndStop">
26        <simpleCondition role="onEnd" />
27        <simpleAction role="stop" max="unbounded"/>
28      </causalConnector>
29      <causalConnector id="onEndStart">
30        <simpleCondition role="onEnd" />
31        <simpleAction role="start" max="unbounded"/>
32      </causalConnector>
33      </connectorBase>
34  </head>
35  <body>
36    <port id="pVideo" component="video1" />
37    <media id="video1" src="media/autumn.mp4"
38        descriptor="desc1"/>
39    <media id="video2" src="media/forest.mp4"
40        descriptor="desc1"/>
41    <media id="video3" src="media/sea.mp4"
42        descriptor="desc1"/>
43    <effect id="lightEffect" type="LightType"
44        descriptor="descL">
45      <property name="color" value="orange"/>
46    </effect>
47    <effect id="scentEffect" type="ScentType"
48        descriptor="descS">
49        <property name="scent"
50        value="urn:mpeg:mpeg-v:01-SI-ScentCS-NS:sea_breeze"/>
51    </effect>
52
53    <link xconnector="onBeginStart">
54      <bind role="onBegin" component="video1"  />
55      <bind role="start" component="lightEffect" />
```

```
56    </link>
57    <link xconnector="onEndStart">
58      <bind role="onEnd" component="video1"  />
59      <bind role="start" component="video2" />
60    </link>
61    <link xconnector="onBeginSet">
62      <bind role="onBegin" component="video2"/>
63      <bind role="set" component="lightEffect" interface="color" >
64        <bindParam name="var" value="green"/>
65      </bind>
66    </link>
67    <link xconnector="onEndStart">
68      <bind role="onEnd" component="video2"  />
69      <bind role="start" component="video3" />
70    </link>
71    <link xconnector="onBeginStart">
72      <bind role="onBegin" component="video3"  />
73      <bind role="start" component="scentEffect" />
74    </link>
75    <link xconnector="onBeginSet">
76      <bind role="onBegin" component="video3"/>
77      <bind role="set" component="lightEffect" interface="color" >
78        <bindParam name="var" value="blue"/>
79      </bind>
80    </link>
81    <link xconnector="onEndStop">
82      <bind role="onEnd" component="video3" />
83      <bind role="stop" component="lightEffect" />
84      <bind role="stop" component="scentEffect" />
85    </link>
86  </body>
87  </ncl>
```

Listagem 5.4: Código da aplicação NCL com efeitos sensoriais de luz e aroma.

A partir da análise do documento, é possível construir o HTG da aplicação mulsemídia descrita anteriormente, para derivar o plano de apresentação apresentado na Tabela 5.3. Este plano é composto pelo instante no tempo de ocorrência de cada transição no evento de

apresentação dos objetos que compõem a aplicação e o objeto que receberá tal transição. Como a aplicação de exemplo não possui nenhum evento imprevisível, este plano de apresentação se manterá inalterado ao longo de toda a execução da aplicação mulsemídia. Uma representação do plano de apresentação na linha do tempo é apresentada na Figura 5.3.

Tabela 5.3: Especificação temporal do plano de apresentação da aplicação de exemplo.

Instante	**Transição**	**Evento**	**Objeto**
0s	start	*presentation*	video1
0s	start	*presentation*	lightEffect
12s	stop	*presentation*	video1
12s	start	*presentation*	video2
24s	stop	*presentation*	video2
24s	start	*presentation*	video3
24s	start	*presentation*	scentEffect
36s	stop	*presentation*	video3
36s	stop	*presentation*	scentEffect

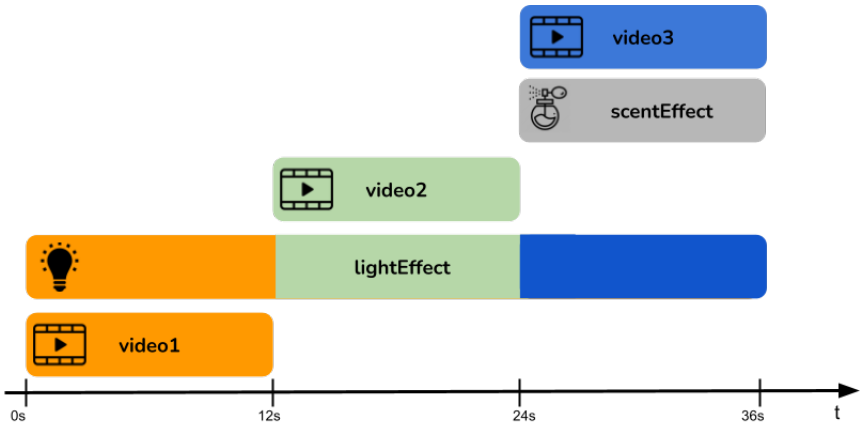

Figura 5.3: Linha do tempo correspondente ao plano de apresentação da aplicação de exemplo.

As próximas seções apresentam como é feito o cálculo do tempo de preparação automática para objetos de mídia e efeitos sensoriais.

5.2.2.1 Cálculo do Tempo de Preparação para Objetos de Mídia

A interface definida entre o formatador e cada *player* de mídia normalmente permite que o formatador obtenha informações do *player*, como o estado de reprodução da mídia (isto é, se a mídia está sendo reproduzida, está pausada ou parada), e o espaço disponível no *buffer*, por exemplo. Além da comunicação com os *players* de mídia, o formatador pode

se comunicar com outros componentes do ambiente de execução da aplicação, como a interface de rede, a fim de obter as condições de transmissão de dados – vazão, *jitter*, *delay*, por exemplo.

Segundo a definição de preparação proposta nesta tese, o tempo para preparar um objeto de mídia é dependente do tamanho do *buffer* do *player* de mídia. Então, a duração da preparação de um objeto de mídia pode ser obtida através da Equação 5.1, onde o parâmetro **throughput** representa a taxa de transferência de dados na rede, e **buffer_size** o tamanho do *buffer* do *player* de mídia. Além disso, o tempo de preparação depende do tempo para instanciar o *player* responsável pela apresentação do conteúdo de mídia (**player_inst_time**).

$$dur_preparation_{media} = player_inst_time + \frac{buffer_size}{throughput} \tag{5.1}$$

Desse modo, uma vez que o formatador consegue obter o instante em que um objeto de mídia será apresentado na aplicação através do plano de apresentação, o tamanho do *buffer* do *player* de mídia, e o tempo para obter parte desse conteúdo através da rede, torna-se possível a criação automática do plano de preparação dos objetos de mídia que compõem a aplicação. O instante de preparação de um objeto de mídia i ($t_{preparation_i}$) é obtido através da Equação 5.2. O instante de apresentação do conteúdo de mídia é representado pelo elemento $t_{presentation_i}$, e o elemento $dur_preparation_{media_i}$ representa a duração de preparação do conteúdo de mídia.

$$t_{preparation_i} = t_{presentation_i} - dur_preparation_{media_i} \tag{5.2}$$

É importante notar que apesar do evento de preparação poder ser aplicado tanto para objetos de mídia tradicionais, quanto para efeitos sensoriais, a preparação para cada um deles é realizada de modo bastante diferente, como visto na próxima seção.

5.2.2.2 Cálculo do Tempo de Preparação para Efeitos Sensoriais

Um efeito sensorial não possui um conteúdo a ser buscado na rede, como um objeto de mídia tradicional. Além disso, alguns efeitos sensoriais, como o efeito de aroma, têm sua preparação dependente do posicionamento do usuário em relação ao atuador. Desse modo, cada tipo de efeito sensorial possui uma equação diferente para o cálculo de seu tempo de preparação.

Considerando a aplicação de exemplo, para o efeito de luz, o valor da variável *dur_preparation* é obtido através da Equação 5.3, onde $t_{message}$ é o tempo gasto para que uma mensagem de ativação seja enviada do formatador para o atuador, e $t_{actuation}$, o tempo para o efetivo acionamento do dispositivo. Esses tempos podem ser obtidos pelo formatador, através da interface de comunicação com o dispositivo atuador, ou através de um arquivo que descreve as capacidades dos dispositivos presentes no ambiente de apresentação.

$$dur_preparation_{lightEffect} = t_{message} + t_{actuation} \tag{5.3}$$

Para o efeito de aroma, a preparação deve considerar, além dos parâmetros $t_{message}$ e $t_{actuation}$, o tempo necessário para que o usuário perceba o efeito (t_{sense}). Nesta tese, o tempo necessário para percepção do efeito de aroma por um usuário é obtido através de um processo de calibragem do dispositivo, conforme será descrito na Seção 6.2. É importante destacar que esse tempo de percepção irá variar para cada usuário da aplicação, e por isso, a calibragem deve ser feita individualmente por usuário. A Equação 5.4 obtém o tempo de preparação para o efeito de aroma.

$$dur_preparation_{scentEffect} = t_{message} + t_{actuation} + t_{sense} \tag{5.4}$$

De posse dos tempos necessários para preparar cada objeto de mídia e efeito sensorial em uma dada aplicação, pode-se criar o seu plano de preparação, como apresentado na próxima seção.

5.2.2.3 Criação do Plano de Preparação

Considerando-se, por exemplo, que o tempo para preparar os vídeos *video1*, *video2* e *video3* da aplicação ilustrada na Figura 5.3 seja igual a 3 segundos cada; o tempo para preparação do efeito sensorial de luz seja igual a 1 segundo, e o tempo para preparação do efeito de aroma seja igual a 5 segundos, é possível construir um plano de preparação derivado do plano de apresentação da Tabela 5.3, conforme apresentado na Tabela 5.4. É importante notar que, caso o tempo para preparar um componente da aplicação mulsemídia (*dur_preparation*) seja maior que o instante de início da apresentação do mesmo componente ($t_{presentation}$), o instante de preparação ($t_{preparation}$) resultante será um valor negativo. E como não existe tempo negativo, o formatador atribui o valor zero para o instante de preparação nestes casos. O mesmo ocorre com as mídias e efeitos que devem

ser iniciados no momento em que a aplicação inicia. A Figura 5.4 apresenta a linha do tempo referente ao plano de preparação, onde as regiões em vermelho representam o início do evento de preparação sobre cada componente da aplicação mulsemídia.

Tabela 5.4: Especificação temporal do plano de preparação da aplicação de exemplo.

Instante	**Transição**	**Evento**	**Objeto**
0s	start	*preparation*	video1
0s	start	*preparation*	lightEffect
9s	start	*preparation*	video2
19s	start	*preparation*	scentEffect
21s	start	*preparation*	video3

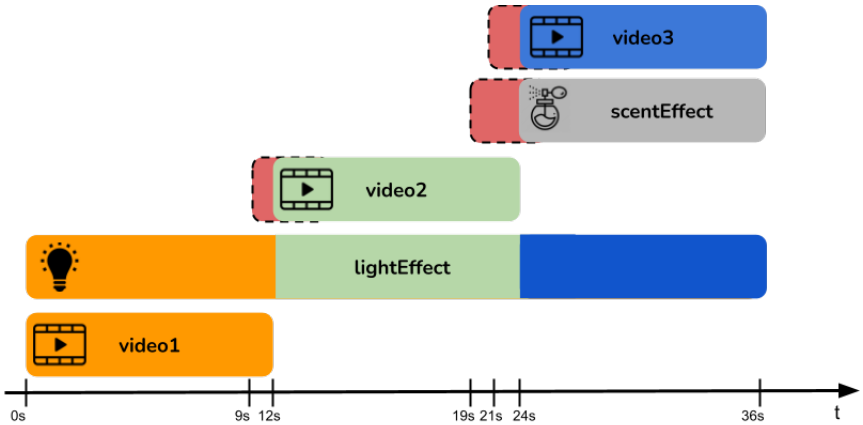

Figura 5.4: Linha do tempo correspondente ao plano de preparação da aplicação de exemplo. As regiões em vermelho indicam os intervalos em que as mídias e efeitos estão sendo preparados.

Este capítulo apresentou a proposta desta tese para preparação de conteúdo de mídia e efeitos sensoriais em documentos mulsemídia. A Seção 5.1 especificou a preparação utilizada na fase de autoria de aplicações, também conhecida como preparação programada. Por fim, a Seção 5.2 descreveu as etapas empregadas para realização da preparação de documentos mulsemídia na fase de execução da aplicação - a preparação automática. O próximo capítulo discute como foi realizada a implementação das propostas desta tese no *middleware* Ginga-NCL.

Capítulo 6

Implementação no Ginga-NCL

Este capítulo apresenta a implementação da preparação de conteúdo de objetos de mídia e da preparação e renderização de efeitos sensoriais no Ginga-NCL [7], que é o subsistema do *middleware* Ginga responsável por prover uma infraestrutura de apresentação para aplicações declarativas especificadas na linguagem NCL. É importante notar que, apesar da proposta ter sido implementada no *middleware* Ginga, para aplicações multimídia especificadas na linguagem NCL, a preparação automática proposta nesta tese não está obrigatoriamente relacionada ao uso desta linguagem.

Conforme apresentado na recomendação ITU-T H.761 [45], que especifica o ambiente de apresentação NCL para prover interoperabilidade em sistemas IPTV, a arquitetura do *middleware* Ginga contém o subsistema *Ginga-NCL Presentation Environment*, o *Ginga Common Core*, a pilha de protocolos e a camada de aplicações e serviços IPTV, conforme mostra a Figura 6.1. O componente núcleo do subsistema *Ginga-NCL Presentation Environment* é o *NCL Formatter*, responsável por controlar a execução da aplicação multimídia. Nesta tese, são propostas modificações nas camadas *Ginga-NCL Presentation Environment*, o *Ginga Common Core* visando permitir que o *middleware* Ginga dê suporte à execução de aplicações especificadas em NCL 4.0.

Inicialmente, o *NCL Formatter* foi modificado, através da inserção de novos componentes (*Presentation Orchestrator*, *Preparation Orchestrator*, *Sensory Device Calibrator*, *Sensory Effect Renderer Manager*, *Interaction Manager*), e alteração de componentes já existentes na arquitetura (*Player Manager* e *NCL Context Manager*), conforme ilustrado na Figura 6.2. Para simplificação, apenas os elementos relacionados às modificações propostas são apresentados na Figura 6.2.

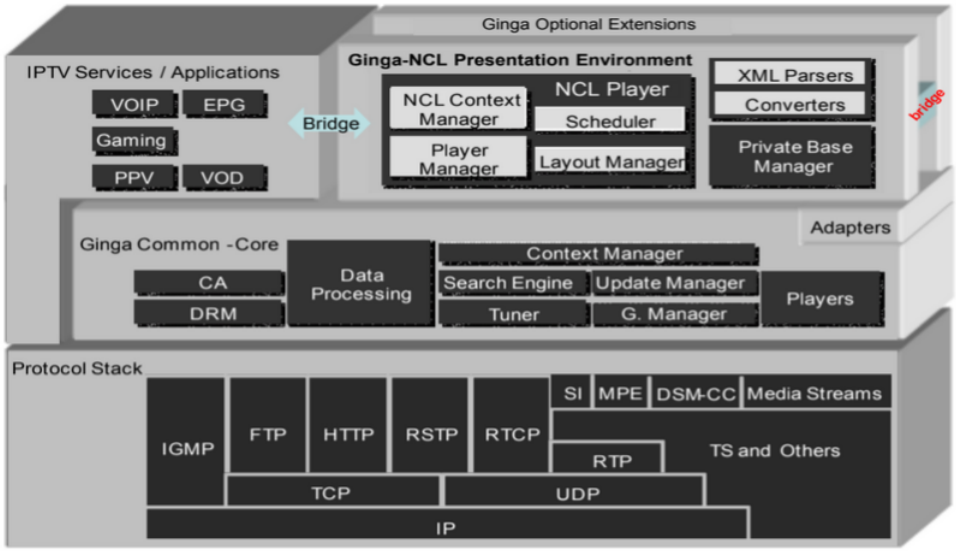

Figura 6.1: Arquitetura Ginga retirada da recomendação ITU-T H.761 [45].

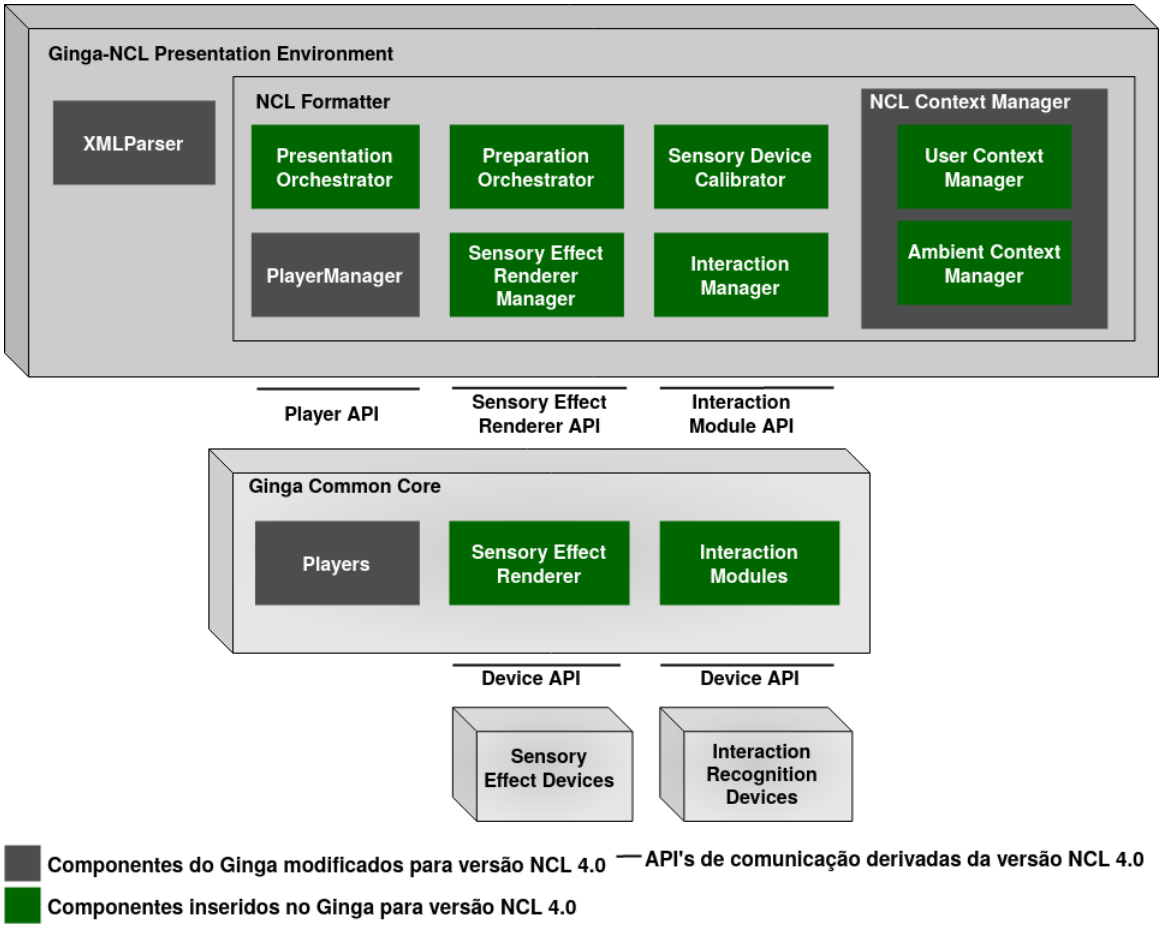

Figura 6.2: Atualização da arquitetura Ginga para suporte à NCL 4.0.

Os componentes *Presentation Orchestrator*, *Preparation Orchestrator*, *Sensory Device Calibrator* e *Sensory Effect Renderer Manager* foram inseridos por esta tese e serão

detalhados a seguir. Já os componentes *Interaction Manager*, *User Context Manager* e *Ambient Context Manager* foram propostos por Barreto [9]. O componente *Interaction Manager* permite a manipulação de diferentes modalidades de interação como interação por voz, gestos ou reconhecimento de expressão facial. Além disso, a proposta de Barreto [9] permite o gerecimento de contexto do usuário e do ambiente de execução através dos componentes *User Context Manager* e *Ambient Context Manager*, respectivamente.

Aplicações NCL são entregues para *Ginga-NCL Presentation Environment* pelo subsistema *Ginga Common Core*. Neste subsistema são implementados os *players* de mídia responsáveis pela reprodução de conteúdo de mídia tradicional (vídeo, áudio, texto e imagem), além de outras funcionalidades como obtenção de metadados da aplicação. Para a execução dos efeitos sensoriais no *middleware* Ginga, esta tese propõe a inserção do componente *Sensory Effect Renderer* ao *Ginga Common Core*, que pode ser especializado para cada tipo de efeito sensorial. No Ginga-NCL, a comunicação entre os *players* de mídia e a máquina de apresentação é realizada através de uma API genérica (Player API). Esta API possibilita que o *Ginga-NCL Presentation Environment* e o *Ginga Common Core* sejam subsistemas independentes entre si. Dessa forma, para manter a independência entre os subsistemas, esta tese propõe a inserção de uma API genérica para comunicação com os renderizadores de efeitos sensoriais (*Sensory Effect Renderer API*). Além disso, para que a implementação do Ginga-NCL seja independente do dispositivo utilizado para renderização de efeito, é preciso implementar uma *API Sensory Effect Renderer*.

Para a validação da solução proposta nesta tese, a implementação de referência da máquina de apresentação Ginga-NCL[1] foi modificada, permitindo que o Ginga dê suporte à apresentação de efeitos sensoriais em aplicações NCL 4.0, e à preparação automática de conteúdo e efeitos sensoriais. Além disso, na nova versão implementada nesta tese, o *middleware* Ginga é capaz de reconhecer o evento *preparation*, especificado pelo autor de uma aplicação NCL para preparações programadas.

A API interna da implementação de referência do Ginga-NCL é composta por um componente `Formatter` e outros dois componentes principais, o parser (componente `Parser`) e uma representação do documento multimídia (componente `Document`), conforme ilustrado na Figura 6.3. O formatador controla todo o ciclo de vida da aplicação NCL, que tem início na análise do documento multimídia, realizada pelo `Parser`. A Seção 6.1 apresenta os componentes que foram inseridos e modificados no Ginga para dar suporte à renderização de efeitos. A Seção 6.2 discute o processo de calibragem de renderizadores

[1] https://github.com/TeleMidia/ginga

de efeitos sensoriais. E a Seção 6.3 descreve as alterações implementadas no Ginga para suporte à preparação automática de mídia e de efeitos sensoriais.

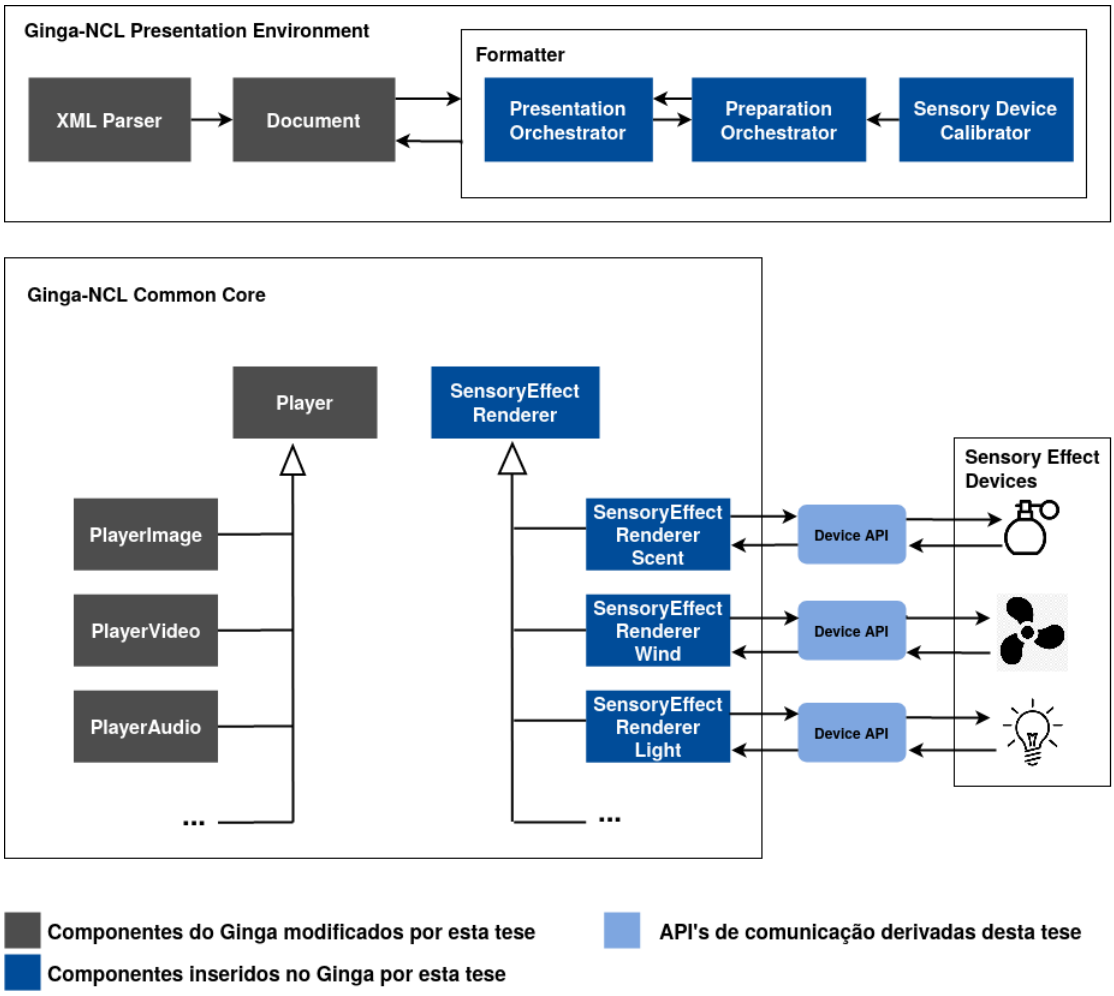

Figura 6.3: Arquitetura da implementação de referência do Ginga.[2]

6.1 Renderização de Efeitos Sensoriais

Conforme mencionado anteriormente, NCL 4.0 possibilita a representação de efeitos sensoriais em um documento mulsemídia, através do elemento `<effect>` inserido na linguagem. Dessa forma, o componente `Parser` foi modificado para identificar o novo tipo de elemento, e as novas propriedades que podem ser empregadas para caracterizar a renderização de um efeito sensorial.

Em sua versão atual, o *middleware* Ginga possibilita a execução de diferentes tipos de mídia (áudio, vídeo, imagem e texto) através de *players* de mídia, que são uma especialização do componente `Player`, apresentado na Figura 6.3. Para dar suporte à execução de efeitos sensoriais, foi criado um novo componente, denominado `EffectPlayer`, que realiza

a comunicação entre o formatador e os dispositivos físicos responsáveis pela renderização do efeito. O *player* de efeito é criado quando um efeito tem sua preparação iniciada, e é destruído assim que o efeito é finalizado - através de uma ação de *stop* sobre o efeito, ou quando o mesmo atinge uma duração, definida de forma explícita pelo autor da aplicação.

Cada *player* de efeito está associado a apenas um efeito sensorial, e vice-versa. Entretanto, pode-se ter várias instâncias de *player* de efeito, que se comunicam com um mesmo dispositivo físico. O *player* de efeito pode ser especializado de acordo com o tipo de efeito sensorial que ele manipula. A Figura 6.4 apresenta os *players* de efeitos sensoriais que foram implementados nesta tese, porém a arquitetura proposta é extensível e permite a implementação de outros tipos de *players* de efeito, como efeitos de temperatura, vibração, etc.

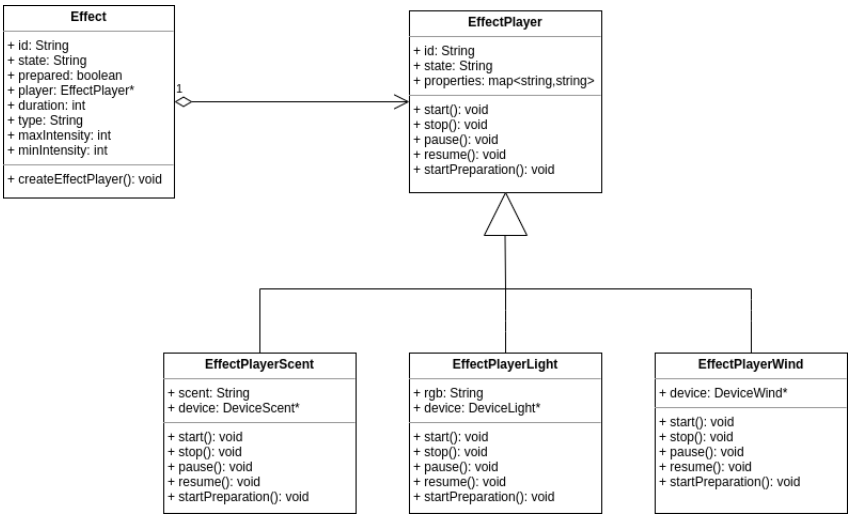

Figura 6.4: Diagrama de classes da implementação dos *players* de efeitos sensoriais.

Os dispositivos físicos responsáveis pela renderização de efeitos podem ser de diferentes fabricantes, e na maioria dos casos, implementam diferentes protocolos de comunicação (por exemplo, API REST [59], MQTT [37], CoAP [13], etc). Essa comunicação deve ser implementada pelo componente `Device API`, que é específico para cada dispositivo renderizador. A API (do inglês, *Application Interface Programming*) deve implementar um conjunto de funções, tais como conexão com o dispositivo físico, ativação, desativação e comandos para alterar a intensidade de renderização do efeito. Além disso, podem ser implementadas funcionalidades específicas do efeito, como por exemplo, comandos para alterar a cor de um efeito de luz.

Nesta tese, foram implementadas três APIs de dispositivo - `DeviceScent`, `DeviceWind` e `DeviceLight`. O componente `DeviceScent` se comunica com o dispositivo Moodo[3]

[3]https://moodo.co/

(a) Dispositivo Moodo. (b) Dispositivo Yeelight.

Figura 6.5: Dispositivos utilizados nesta tese para renderização de efeitos de aroma e luz.

(item (a) na Figura 6.5). O Moodo é um dispositivo desenvolvido para casas inteligentes e funciona com um sistema de cápsulas com cristais de aromas, suportando até quatro cápsulas acopladas. As cápsulas são acionadas por meio de seus respectivos ventiladores com intensidades reguláveis. Para se comunicar com o Moodo, o `DeviceScent` utiliza uma API RESTful[4] que usa o formato JSON (*JavaScript Object Notation*) para entrada e saída de informações.

A renderização do efeito de luz foi implementada através de uma lâmpada inteligente Yeelight[5] (item (b) na Figura 6.5), que aceita comandos de acionamento e alterações na cor da luz emitida através de comando via rede. O componente `DeviceLight` implementa o protocolo de controle especificado pelo fabricante Yeelight, que define três tipos de mensagem: *COMMAND*, *RESULT* e *NOTIFICATION*. Todas as mensagens devem ser entregues em um formato JSON específico em uma conexão TCP. A mensagem do tipo *COMMAND* permite enviar os comandos para ligar a lâmpada e desligá-la, e também alterar a cor e o brilho da luz renderizada. A mensagem *RESULT* é gerada pela lâmpada quando ela recebe uma mensagem do tipo *COMMAND*. Sempre que ocorre uma mudança de estado da lâmpada inteligente, ela enviará uma mensagem *NOTIFICATION* a todos os dispositivos de terceiros conectados à ela.

Para a renderização do efeito de vento, utilizou-se um mini ventilador controlado por um microcontrolador Arduino Uno R3. A conexão do ventilador com o Arduino é ilustrada na Figura 6.6. O ventilador utilizado suporta apenas comandos para ativação e desativação, não sendo possível alterar sua intensidade. A comunicação do formatador e o ventilador é feita através do protocolo de comunicação MQTT, implementado pelo

[4]https://rest.moodo.co/
[5]https://www.yeelight.com/

componente `DeviceWind`. O MQTT (*Message Queue Telemetry Transport*) é um protocolo de mensagens que utiliza o paradigma *publish-subscribe* para troca de informações e o conceito de tópicos para processá-las. Desta forma, é necessário um *broker* que será o responsável por receber dados dos *publishers*, enfileirar e enviar os dados recebidos para os *subscribers*. Um *publisher* não envia mensagens diretamente a um *subscriber*: ele deve se conectar ao *broker* e publicar a mensagem em um tópico. O *subscriber* alvo, deve se conectar ao *broker*, e receber as mensagens em cujos tópicos está inscrito [38]. O componente `DeviceWind` publica os comandos de ativar e desativar no tópico do broker hospedado na rede local. O Arduino então se inscreve no mesmo tópico, e ao ler a mensagem *"start"*, ativa o ventilador, e o desliga ao receber ler uma mensagem *"stop"* no tópico inscrito.

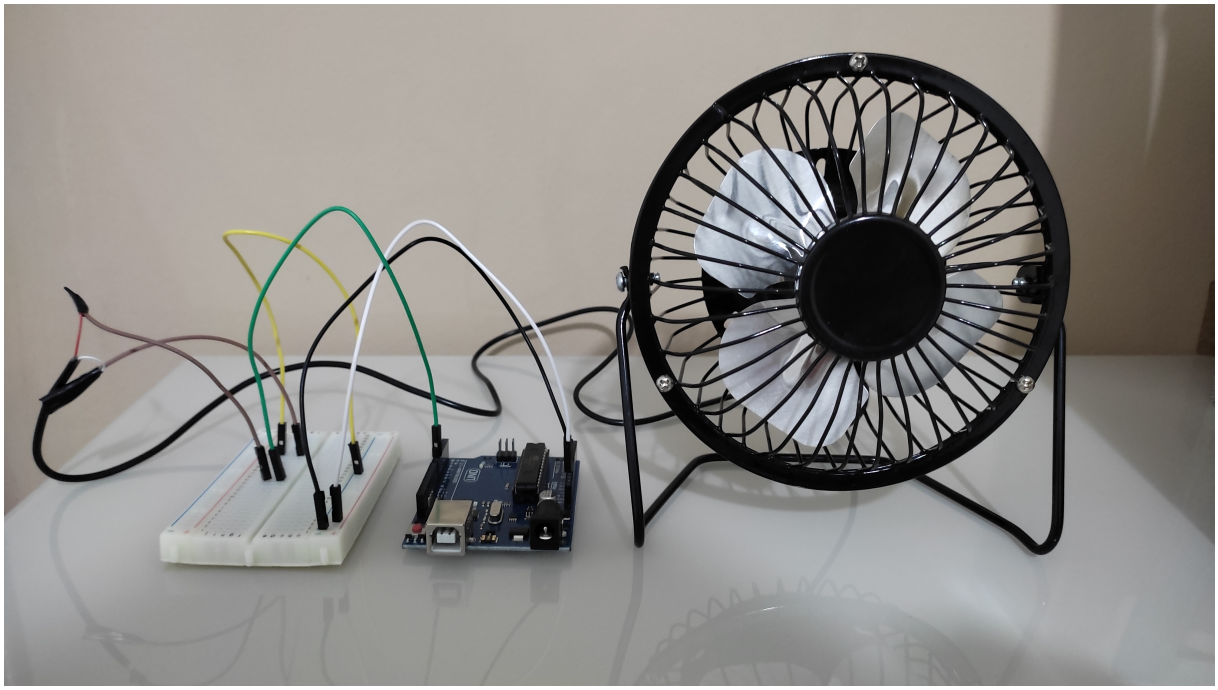

Figura 6.6: Dispositivo utilizado para renderização do efeito de vento.

6.2 Calibragem de renderizadores de efeitos sensoriais

Em geral, o efeito sensorial de aroma leva um tempo maior para ser percebido pelo usuário, após sua liberação pelo dispositivo atuador. O tempo para o efeito ser percebido pelo usuário depende do ambiente físico no qual a aplicação mulsemídia será apresentada, da distância do usuário para o emissor de aroma e do próprio usuário em si. Como esse tempo de percepção do aroma influencia na preparação do efeito, foi implementado um mecanismo de calibragem do renderizador de efeito de aroma nesta tese. A calibragem dos renderizadores de efeitos de luz e de vento não foi necessária, visto que o atraso entre o

disparo do efeito e a percepção do mesmo é da ordem de milissegundos, conforme descrito na Seção 7.2.2.

Nesta tese, o processo de calibragem do renderizador de aroma foi feito através de uma aplicação NCL 4.0, composta por duas imagens e um efeito de aroma. A imagem exibida na Figura 6.7 informa ao usuário que ele deve selecionar o botão "ENTER" do controle remoto, quando perceber o aroma. Quando o usuário seleciona a tecla "ENTER", uma imagem de confirmação da seleção é apresentada na tela. A aplicação de calibragem é apresentada na Listagem 6.1.

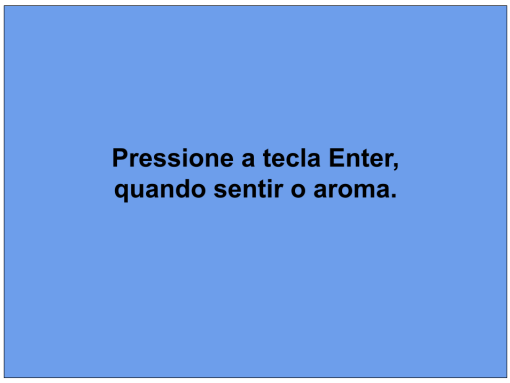

Figura 6.7: Tela da aplicação NCL para calibragem do renderizador de aroma.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <ncl id="appCalibragem">
3   <head>
4    <regionBase>
5      <region id="rg1" width="100%" height="100%" zIndex = "0" />
6      <region id="rg2" location="center:top:front" />
7      <region id="rg3" width="25%" height="25%" top="55%" left="40%"/>
8      </regionBase>
9    <descriptorBase>
10       <descriptor id="d1"  region="rg1" />
11     <descriptor id="d2"  region="rg2" intensityValue="100" scent="urn:
     mpeg:mpeg-v:01-SI-ScentCS-NS:rose"/>
12     <descriptor id="d3"  region="rg3" explicitDur="5s"/>
13   </descriptorBase>
14   <connectorBase>
15       <causalConnector id="onBeginStart">
16         <simpleCondition role="onBegin" />
17         <simpleAction role="start" max="unbounded"/>
```

```
18      </causalConnector>
19      <causalConnector id="onEndStop">
20        <simpleCondition role="onEnd" />
21        <simpleAction role="stop" max="unbounded"/>
22      </causalConnector>
23      <causalConnector id="onSelectionStart">
24            <simpleCondition role="onSelection" key="ENTER"/>
25            <simpleAction role="start"/>
26      </causalConnector>
27      <causalConnector id="onBeginStopDelay">
28        <simpleCondition role="onBegin" />
29        <simpleAction role="stop" max="unbounded" delay="5s"/>
30      </causalConnector>
31    </connectorBase>
32  </head>
33  <body>
34  <port id="pBackground" component="background" />
35  <media id="background" src="media/img-scent.png" descriptor="d1">
36    <area id="a1" begin="5s"/>
37  </media>
38  <media id="okImage" src="media/ok.png" descriptor="d3"/>
39  <effect id="scentEffect" type="ScentType" descriptor="d2"/>
40  <link xconnector="onBeginStart">
41    <bind role="onBegin" component="background" interface="a1" />
42    <bind role="start" component="scentEffect" />
43  </link>
44  <link xconnector="onSelectionStart">
45    <bind role="onSelection" component="background"/>
46    <bind role="start" component="okImage" />
47  </link>
48  <link xconnector="onEndStop">
49    <bind role="onEnd" component="okImage" />
50    <bind role="stop" component="background" />
51  </link>
52  <link xconnector="onEndStop">
```

```
53     <bind role="onEnd" component="background" />
54     <bind role="stop" component="scentEffect" />
55   </link>
56 </body>
57 </ncl>
```

Listagem 6.1: Código da aplicação NCL para calibragem do renderizador de aroma.

Ao executar o middleware Ginga em modo calibragem, o mecanismo de preparação automática é desativado, e o formatador coleta os instantes de tempo no qual o efeito é disparado na aplicação, e o tempo em que o usuário seleciona a tecla "ENTER", indicando a percepção do efeito. A aplicação NCL de calibragem é finalizada 5 segundos após a interação do usuário. Ao finalizar a aplicação de calibragem, o formatador calcula o atraso entre o disparo do efeito e a interação do usuário, e configura esse valor como tempo de preparação do efeito de aroma no arquivo de configuração de dispositivos renderizadores. O arquivo de configuração é baseado na parte 2 do padrão MPEG-V [41] e na proposta de Saleme [61] que permite especificar a configuração do ambiente de execução, através de um arquivo chamado *"SERenderer.xml"*. O arquivo utilizado nesta tese especifica as capacidades (intensidade máxima e mínima, por exemplo) e configuração de conexão dos dispositivos físicos (e.g, endereço IP). A Listagem 6.2 apresenta um exemplo de arquivo de configuração que especifica um dispositivo para renderização de efeito de aroma, e um dispositivo renderizador de luz.

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <devices>
3   <deviceDescription>
4     <id>lightDevice</id>
5     <effectType>LightType</effectType>
6     <minIntensity>0</minIntensity>
7     <maxIntensity>5</maxIntensity>
8     <locator>center:middle:left</locator>
9     <unit>lumens</unit>
10    <ipAddress>192.168.1.3</ipAddress>
11    <supportedValues>
12      <value>yellow</value>
13      <value>blue</value>
14      <value>green</value>
```

```
15        <value>orange</value>
16        <value>red</value>
17      </supportedValues>
18    </deviceDescription>
19    <deviceDescription>
20      <id>moodoDevice</id>
21      <effectType>ScentType</effectType>
22      <minIntensity>0</minIntensity>
23      <maxIntensity>5</maxIntensity>
24      <locator>center:middle:left</locator>
25      <unit>ml</unit>
26      <supportedValues>
27        <value>Sea</value>
28        <value>Coconuts</value>
29        <value>Amber Marine</value>
30        <value>Sweet</value>
31      </supportedValues>
32      <preparationTime>13</preparationTime>
33      <login>gingapreparation@gmail.com</login>
34      <password>12345</password>
35    </deviceDescription>
36  </devices>
```

Listagem 6.2: Exemplo de arquivo de configuração de efeitos sensoriais.

Cada dispositivo deve ser identificado de forma única no arquivo de dispositivos, através do elemento `<id>`. O tipo de efeito suportado por um dispositivo é especificado pela tag `<effectType>`. A localização do renderizador de efeito, dentro do ambiente de execução da aplicação mulsemídia é especificada pelo elemento `<locator>`, e pode ser definido tanto por coordenadas esféricas, quanto pelo formato de localização definido no padrão MPEG-V [41], como explicado na Seção 4.2. Um dispositivo pode possuir um tempo de preparação específico, ou definido pelo mecanismo de calibragem. Essa informação é especificada pelo elemento `<preparationTime>` do arquivo de configuração. Por fim, dispositivos podem suportar um conjunto de valores, de acordo com seu tipo. Por exemplo, o renderizador de aroma pode suportar apenas um conjunto de tipos de aroma, e o renderizador de luz pode ser capaz de apresentar apenas algumas cores. Esses

valores são especificados através do elemento `<value>`, filho de `<supportedValues>`.

6.3 Implementação do Evento de Preparação

O mecanismo de preparação automática de objetos de mídia e efeitos sensoriais tem como ponto inicial a construção do grafo temporal da aplicação. Para isso, nesta tese, o componente `Parser` foi alterado de modo a gerar um grafo temporal a partir do processamento do documento mulsemídia. Nesta etapa, o grafo temporal é construído com base nos nós que compõem o documento, e as arestas são criadas com base nas relações definidas pelos elos NCL (elemento `<link>`). Por exemplo, para a relação de sincronização temporal entre a mídia *video1* e o efeito de luz, apresentada na Listagem 6.3 (linhas 3 a 6) da aplicação descrita na Seção 5.2.2, os vértices 1 e 2 e a aresta $(v_1 v_2)$ que conecta esses vértices devem ser adicionados ao grafo temporal da aplicação, conforme ilustra a Figura 6.8. Os eventos de atribuição para alterar a propriedade "color" do efeito de luz também são representados no grafo (vértices 5 e 9 da Figura 6.8), apesar de não serem representados no plano de apresentação da aplicação.

```
1  <body>
2  ...
3      <link xconnector="onBeginStart">
4      <bind role="onBegin" component="video1"  />
5      <bind role="start" component="lightEffect" />
6    </link>
7    <link xconnector="onEndStart">
8      <bind role="onEnd" component="video1"  />
9      <bind role="start" component="video2" />
10   </link>
11   <link xconnector="onBeginSet">
12     <bind role="onBegin" component="video2"/>
13     <bind role="set" component="lightEffect" interface="color" >
14       <bindParam name="var" value="green"/>
15     </bind>
16   </link>
17   <link xconnector="onEndStart">
18     <bind role="onEnd" component="video2"  />
19     <bind role="start" component="video3" />
```

```
20   </link>
21   <link xconnector="onBeginStart">
22     <bind role="onBegin" component="video3"  />
23     <bind role="start" component="scentEffect" />
24   </link>
25   <link xconnector="onBeginSet">
26     <bind role="onBegin" component="video3"/>
27     <bind role="set" component="lightEffect" interface="color" >
28       <bindParam name="var" value="blue"/>
29     </bind>
30   </link>
31   <link xconnector="onEndStop">
32     <bind role="onEnd" component="video3" />
33     <bind role="stop" component="lightEffect" />
34     <bind role="stop" component="scentEffect" />
35   </link>
36 </body>
37 ...
```

Listagem 6.3: Trecho de código NCL para especificar as relações temporais entre os objetos de mídia que compõem a aplicação mulsemídia.

Além das arestas derivadas dos elos (elemento `<link>`) do documento, o `Parser` cria arestas que representam a transição de fim da apresentação de um objeto de mídia, tendo como condição a duração explícita da mídia – arestas (v_1v_3), (v_4v_6) e (v_7v_{10}) da Figura 6.8. A duração explícita de uma mídia ou de um efeito sensorial é definida em NCL por meio da propriedade `explicitDur`. Além disso, para objetos de mídia do tipo contínua (áudio e vídeo, por exemplo), caso não exista nenhuma duração explícita definida para eles, o formatador obtém sua duração implícita através da análise dos metadados dos mesmos.

A representação do grafo temporal é definida pela classe `TemporalGraph` inserida na implementação de referência do Ginga-NCL. Esta classe especifica a estrutura de grafo dirigido e as principais funções para manipulação do mesmo, tal como o método de percurso no grafo para a obtenção das cadeias temporais. A construção do plano de apresentação e sua definição é especificada pelo componente `PresentationOrchestrator`. Este componente possui uma estrutura denominada `PresentationPlan` que contém os instantes de ocorrência de transições sobre o evento de apresentação de cada objeto de mídia

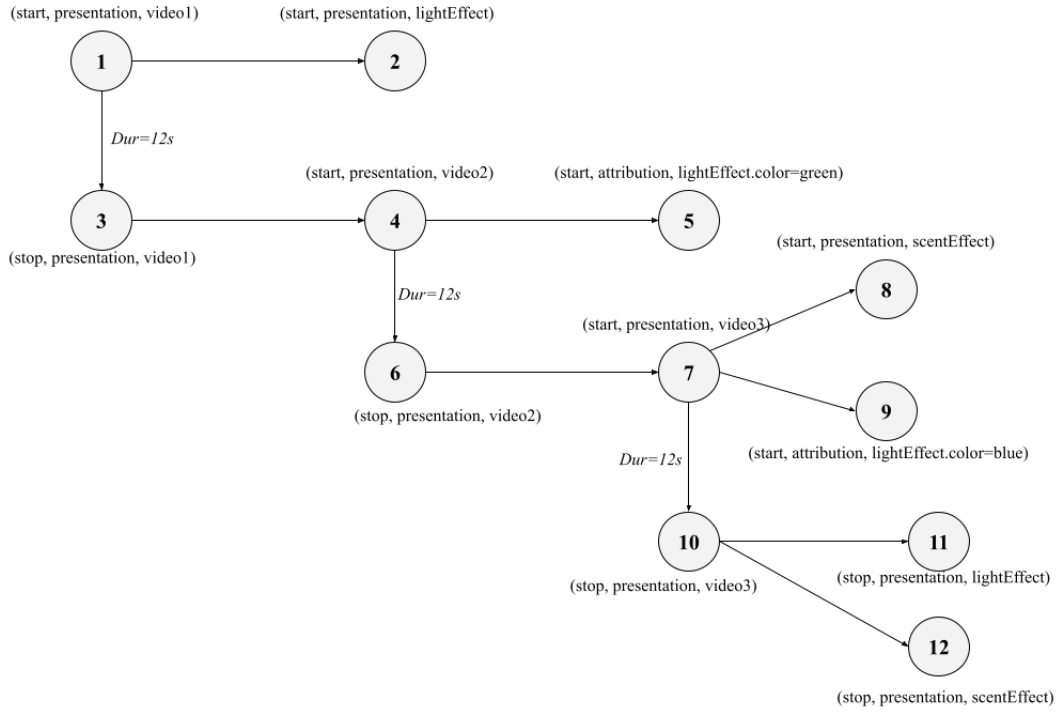

Figura 6.8: Vértices e arestas inseridas no HTG para a os links da Listagem 6.3.

ou efeito sensorial que compõe a aplicação. O objeto do tipo `TemporalGraph` obtido da especificação da aplicação é utilizado como parâmetro para a construção do plano de apresentação.

Por fim, o componente `PreparationOrchestrator` implementa os métodos necessários à construção do plano de preparação. Inicialmente, as condições da rede são analisadas para obter o tempo necessário para preparar um objeto de mídia com base no tamanho do *buffer*. Já para os efeitos sensoriais, o `PreparationOrchestrator` considera as informações extraídas do arquivo de configuração dos dispositivos renderizadores. Nesta etapa, as equações apresentadas na Seção 5.2.2 e o plano de apresentação (`PresentationPlan`) são utilizados para a criação do plano de preparação. O plano de preparação resultante é uma estrutura contendo os instantes de preparação de cada objeto de mídia e efeito sensorial que compõe a aplicação. O diagrama de classes com as relações entre os novos componentes é apresentado na Figura 6.9.

Assim como ocorre com os outros tipos de evento definidos em NCL (*presentation, selection, attribution*), o evento *preparation* é inicializado no estado *"dormindo"*. Para objetos de mídia tradicionais, o evento *preparation* permanece no estado "ocorrendo" enquanto a âncora (subconjunto de unidades de informação de um objeto de mídia) a ser preparada é progressivamente carregada no *buffer* do *player* de mídia. Já na preparação de

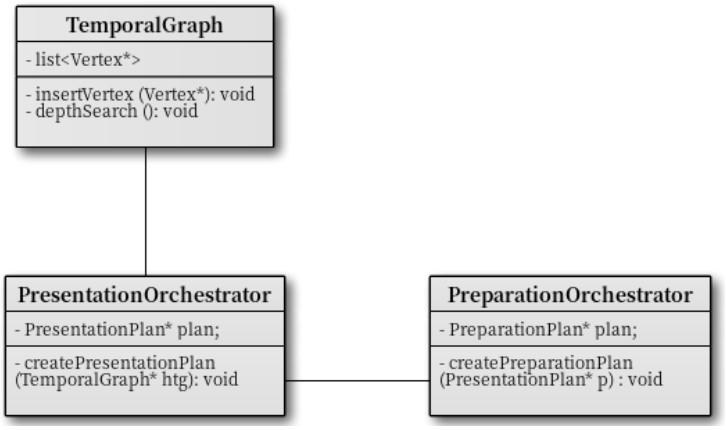

Figura 6.9: Diagrama com as classes inseridas na implementação de referência do Ginga-NCL para preparação automática.

efeitos sensoriais, o evento permanece no estado "ocorrendo" até que o formatador receba uma notificação do *player* de efeito. Além dessas transições, um evento de preparação pode passar do estado "ocorrendo" para "dormindo" em resposta ao fim do carregamento do conteúdo, ou devido a uma ação *stops* ou *aborts* sobre o evento *preparation*. A máquina de estados de um evento é ilustrada na Figura 5.1.

O evento *preparation* pode ser empregado pelo autor através da definição de papéis em *links* e conectores, responsáveis pela especificação das relações entre os objetos que compõem a aplicação. Portanto, o elemento `Parser` da API interna do Ginga foi modificado para reconhecer os novos papéis de condição e ação relacionados ao evento *preparation*. Caso um elo NCL especifique a ação *startPreparation*, o componente `Formatter` instancia um *player* para a mídia a ser preparada e solicita o carregamento do conteúdo da âncora correspondente no *buffer* do *player*. Ao fim do processo de preparação, o *player* envia uma mensagem ao `Formatter`, que gera a transição de *stops* no evento *preparation*, representado no papel de condição denominado *onEndPreparation*.

É importante destacar que, caso a preparação de um objeto de mídia seja definida explicitamente pelo autor da aplicação, o instante de início da preparação do objeto no plano de preparação será o tempo definido pelo autor, e não o instante calculado com base no plano de apresentação. Portanto, a preparação automática só será aplicada aos objetos de mídia que não possuem preparação especificada no documento multimídia.

O formatador se comunica com cada *player* de mídia para disparar ações como iniciar a apresentação da mídia, pausar uma mídia, ou então finalizar sua apresentação. Além disso, o formatador pode configurar propriedades de um *player* de mídia, como, por exemplo, o volume de reprodução de uma mídia. O componente `Player`, e consequentemente, os

players de mídia que o estendem (exceto o *player* de texto e o *player* NCLua), foram alterados, nesta tese, para permitir que o formatador inicie a preparação do mesmo. Essa funcionalidade também foi implementada nos players de efeitos sensoriais, através do método *startPreparation()*.

A preparação de um *player* de mídia consiste em sua instanciação, e carregamento do conteúdo, com base no espaço disponível do *buffer*. Após finalizar o carregamento do conteúdo, o componente `Player` notifica ao formatador o fim da preparação do objeto de mídia. Já a preparação do *player* de efeito inclui o estabelecimento de conexão com a API do dispositivo, e também o envio antecipado do comando ao dispositivo. As APIs dos dispositivos físicos enviam uma mensagem de confirmação, a cada comando enviado pelo *player* de efeito. Dessa forma, quando o *player* de efeito recebe a confirmação do dispositivo, ele considera que a preparação do efeito sensorial foi finalizada, e notifica ao formatador.

Este capítulo apresentou a implementação da solução proposta no middleware Ginga-NCL. Para que o middleware dê suporte à execução de aplicações NCL 4.0, esta tese implementou alguns componentes para renderização de efeitos sensoriais, conforme descrito na Seção 6.1. A Seção 6.2 descreveu a implementação de um mecanismo de calibragem para auxiliar no processo de preparação automática de efeitos sensoriais. Além disso, as modificações realizadas no middleware para dar suporte ao evento de preparação foram apresentadas na Seção 6.3.

Capítulo 7

Casos de Uso e Avaliação da Proposta

Este capítulo apresenta alguns casos de uso do evento de preparação aplicado a objetos de mídia e a efeitos sensoriais, e avaliações da solução proposta. Os casos de uso representam uma aplicação multimídia que sincroniza conteúdo transmitido por *broadcast* com conteúdo *broadband*, uma aplicação mulsemídia com efeito de luz e uma aplicação mulsemídia com efeito de aroma, respectivamente. Essas aplicações demandam a utilização do mecanismo de preparação proposto nesta tese, para garantir o sincronismo temporal durante sua execução. Os casos de uso foram executados na implementação do *middleware* do Ginga-NCL apresentada no Capítulo 6.

7.1 Casos de Uso

7.1.1 Propaganda Direcionada

Esta seção apresenta uma aplicação multimídia utilizada como prova de conceito, a fim de demonstrar a utilização da preparação de objetos de mídia na fase de execução de aplicações multimídia. No serviço de propaganda direcionada, é fornecido ao usuário um conteúdo considerando seu perfil, ao invés do comercial geral distribuído via sinal *broadcast*. Para que este tipo de aplicação possa ser fornecido ao usuário, é necessário que o receptor seja capaz de receber tanto conteúdo enviado por *broadcast* quanto por *broadband*. Os sistemas multimídia com este tipo de suporte são denominados sistemas IBB (do inglês, *Integrated Broadcast Broadband*), e estão disponíveis nos perfis C [6] e D [7] de receptores do Sistema Brasileiro de TV Digital.

O caso de uso de propaganda direcionada foi especificado através de um documento NCL, e é composto por três objetos de mídia do tipo vídeo. Na aplicação proposta, um vídeo (*"mainVideo"*) representando a transmissão de programação de um canal *broadcast* é transmitido como conteúdo principal, e outros conteúdos de vídeo transmitidos via interface *broadband* são sincronizados com este conteúdo. No instante de 23 segundos da transmissão do conteúdo principal, um vídeo contendo um comercial com duração de 30 segundos é apresentado. O conteúdo do comercial é adaptado ao perfil do usuário, que pode ser feminino ou masculino. Caso o perfil de usuário definido no receptor seja feminino, uma propaganda de perfume feminino (*"adWoman"*) será apresentada; e uma propaganda de perfume masculino (*"adMan"*) será apresentada caso contrário.

A preparação de conteúdo de mídia tem como objetivo evitar falhas de sincronização na reprodução de conteúdo transmitido via *broadband*. Entretanto, em alguns cenários pode ocorrer da preparação não ser bem-sucedida, e o conteúdo de mídia não estará preparado no momento em que deve ser apresentado na aplicação. Dessa forma, a aplicação de caso de uso contém uma relação que especifica que o conteúdo direcionado só será apresentado se estiver preparado. Para isso, foi utilizado o atributo *prepared* do evento *preparation*. O código da aplicação NCL de propaganda direcionada é apresentado na Listagem 7.1.

```
1  <ncl id="App_Targeted_Advertising">
2  <head>
3   <ruleBase>
4    <rule id="rWoman" var="profile" comparator="eq" value="woman"/>
5    <rule id="rMan" var="profile" comparator="eq" value="man"/>
6   </ruleBase>
7   <regionBase>
8    <region id="rgVideo" width="100%" height="100%"/>
9   </regionBase>
10  <descriptorBase>
11   <descriptor id="dVideo" region="rgVideo"/>
12   <descriptor id="dAdv" region="rgVideo"/>
13  </descriptorBase>
14  <connectorBase>
15   <causalConnector id="onBeginAndTestStart">
16    <compoundCondition operator="and">
17     <simpleCondition role="onBegin" />
18     <assessmentStatement comparator="eq">
```

```
19        <attributeAssessment role="test"  eventType="preparation"  attributeType="
    prepared"/>
20          <valueAssessment value="true" />
21        </assessmentStatement>
22      </compoundCondition>
23      <simpleAction role="start" />
24    </causalConnector>
25    <causalConnector id="onEndStop">
26      <simpleCondition role="onEnd" />
27      <simpleAction role="stop" />
28    </causalConnector>
29    <causalConnector id="onBeginSet">
30      <connectorParam name="var"/>
31      <simpleCondition role="onBegin" />
32      <simpleAction role="set" value="$var"/>
33    </causalConnector>
34    <causalConnector id="onEndSet">
35      <connectorParam name="var"/>
36      <simpleCondition role="onEnd" />
37      <simpleAction role="set" value="$var"/>
38    </causalConnector>
39  </connectorBase>
40 </head>
41 <body>
42  <port id="pStart" component="mainVideo"/>
43  <media id="mainVideo" src="stream.mp4" descriptor="dVideo">
44    <area id="a1" begin="23s" end="53s"/>
45    <property name="soundLevel" value="1"/>
46    <property name="zIndex" value="0"/>
47  </media>
48  <media id="settings" type="application/x-ginga-settings">
49      <property name="profile" value="woman"/>
50    </media>
51  <switch id="switchAdv">
52    <switchPort id="pWoman">
53      <mapping component="adWoman"/>
54    </switchPort>
55    <switchPort id="pMan">
```

```
56      <mapping component="adMan"/>
57    </switchPort>
58    <bindRule constituent="adWoman" rule="rWoman"/>
59    <bindRule constituent="adMan" rule="rMan"/>
60    <media id="adWoman" src="http://.../media/adv1.mp4" descriptor="dAdv">
61      <property name="zIndex" value="1"/>
62    </media>
63    <media id="adMan" src="http://.../media/adv2.mp4" descriptor="dAdv">
64      <property name="zIndex" value="1"/>
65    </media>
66  </switch>
67
68  <link xconnector="onBeginAndTestStart">
69    <bind role="onBegin" component= "mainVideo" interface="a1"/>
70    <bind role="test" component="switchAdv" interface="pWoman"/>
71    <bind role="start" component="switchAdv"/>
72  </link>
73  <link xconnector="onBeginAndTestStart">
74    <bind role="onBegin" component= "mainVideo" interface="a1"/>
75    <bind role="test" component="switchAdv" interface="pMan"/>
76    <bind role="start" component="switchAdv"/>
77  </link>
78  <link xconnector="onEndStop">
79    <bind role="onEnd" component= "mainVideo" interface="a1"/>
80    <bind role="stop" component="switchAdv"/>
81  </link>
82  <link xconnector="onBeginSet">
83    <bind role="onBegin" component= "switchAdv"/>
84    <bind role="set" component="mainVideo" interface="soundLevel">
85      <bindParam name="var" value="0"/>
86    </bind>
87  </link>
88  <link xconnector="onEndSet">
89    <bind role="onEnd" component= "switchAdv"/>
90    <bind role="set" component="mainVideo"  interface="soundLevel">
91      <bindParam name="val" value="1"/>
92    </bind>
93  </link>
```

```
94 </body>
95 </ncl>
```

Listagem 7.1: Aplicação NCL de propaganda direcionada.

A linguagem NCL dá suporte à adaptação de conteúdo através do elemento `<switch>` (linhas 55 a 70 da Listagem 7.1), que é uma composição com nós alternativos, onde apenas um deles será selecionado. A decisão sobre qual nó será selecionado é dada por regras de mapeamento, definidas por elementos `<bindRule>` (linhas 62 e 63 na Listagem 7.1). O `switch` foi utilizado na aplicação de caso de uso, a fim de permitir a escolha do conteúdo de propaganda que será apresentado ao usuário, com base na propriedade "profile" do nó do tipo *application/x-ncl-settings* (linhas 52 a 54 da Listagem 7.1) que mantém as variáveis globais do ambiente de execução.

Em sistemas de TV Digital com suporte à transmissão integrada de conteúdo *broadcast* e *broadband*, durante a exibição da propaganda direcionada, o conteúdo transmitido via *broadband* deve sobrepor o conteúdo *broadcast*. Além disso, em alguns casos, não é possível parar a recepção e reprodução do conteúdo *broadcast*. Nesses cenários, o conteúdo *broadcast* deve então ter seu volume de áudio configurado para zero quando a apresentação do conteúdo *broadband* inicia (linhas 86 a 91 da Listagem 7.1). Para que o conteúdo de vídeo da propaganda sobreponha o conteúdo principal, foi utilizada a propriedade `zIndex` da linguagem NCL (linhas 50, 65 e 68 da Listagem 7.1). A propriedade `zIndex` especifica como fica a sobreposição de regiões, de forma que uma região de maior valor para `zIndex` se sobrepõe àquela de menor valor.

Por fim, para garantir que a propaganda direcionada irá finalizar no momento correto, ou seja, quando a transmissão do programa principal iniciar novamente, foi criada uma relação de sincronização especificada pelo elo nas linhas 82-85 da Listagem 7.1. O elo criado irá parar a apresentação do conteúdo de propaganda direcionada no fim do período do intervalo comercial.

Como foi apresentado na Seção 2.3, o comportamento temporal de uma aplicação multimídia pode ser modelado através da utilização de um grafo com arestas dirigidas, denominado HTG. De acordo com a proposta de [22], um vértice no HTG é definido como uma tripla 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, preparação) e pelo identificador da interface que define o evento: uma âncora de conteúdo de um objeto, se

o evento é do tipo apresentação ou seleção, ou uma propriedade de um objeto.

O HTG para a aplicação de caso de uso é apresentado na Figura 7.1, onde as triplas que definem os vértices são especificadas entre parênteses. Os rótulos ao lado de cada aresta representam as condições simples ou compostas de percurso no grafo. Arestas que não possuem nenhuma condição associada são disparadas imediatamente após o vértice de origem ser alcançado. 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*.

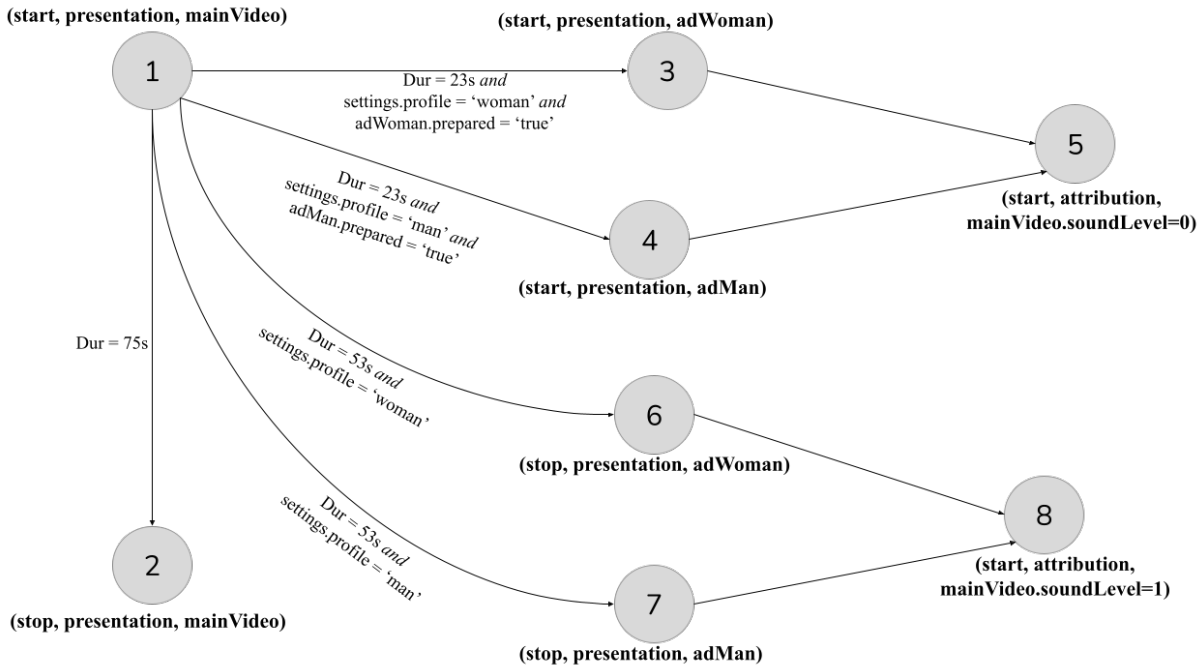

Figura 7.1: Grafo temporal da aplicação multimídia de propaganda direcionada.

Durante a fase de análise do documento multimídia, após obter o grafo temporal da aplicação, o formatador é capaz de calcular os instantes de apresentação de cada objeto de mídia que compõe a aplicação, e construir o plano de apresentação inicial da mesma, conforme apresentado na Tabela 7.1.

Tabela 7.1: Plano de apresentação inicial da aplicação de propaganda direcionada.

Instante	**Transição**	**Evento**	**Objeto**
0s	*start*	*presentation*	*mainVideo*
23s	*start*	*presentation*	*adWoman*
23s	*start*	*presentation*	*adMan*
53s	*stop*	*presentation*	*adWoman*
53s	*stop*	*presentation*	*adMan*
75s	*stop*	*presentation*	*button*

Como foi dito anteriormente, a preparação de um objeto de mídia tem seu fim quando o *buffer* do *player* de mídia é totalmente preenchido ou quando o conteúdo da âncora é totalmente carregado no *buffer*, sendo considerada a condição que ocorrer primeiro. Desse modo, o tempo máximo de preparação pode ser obtido utilizando como parâmetro, o espaço disponível em *buffer*. A partir dos testes realizados, foi possível observar que o tempo médio de preparação de conteúdo de vídeo, em uma rede com taxa de transmissão igual a 25Mbps, é em média 6 segundos. Considerando este valor, o plano de preparação automático criado pelo formatador é apresentado na Tabela 7.2. Este plano utiliza a Equação 5.2, apresentada na Seção 5.2.2, para o cálculo do instante para início de preparação de cada objeto de mídia.

Tabela 7.2: Plano de preparação inicial da aplicação de propaganda direcionada.

Instante	**Transição**	**Evento**	**Objeto**
0s	*start*	*preparation*	*mainVideo*
17s	*start*	*preparation*	*adWoman*
17s	*start*	*preparation*	*adMan*

Durante a construção do plano de preparação, o HTG é atualizado a fim de modelar também os eventos de preparação dos objetos de mídia. Desse modo, o grafo temporal apresentado anteriormente deve ser modificado, conforme ilustrado na Figura 7.2. Os nós em vermelho (9 e 10) representam os eventos de preparação adicionados ao HTG.

É importante notar que em alguns casos, o fim da preparação do objeto de mídia não implica necessariamente o início de sua apresentação. Por exemplo, caso a preparação do objeto *adWoman* finalize em menos de 6 segundos, a mídia estaria pronta para ser apresentada antes do instante de 23 segundos. Porém, de acordo com a relação especificada pelo autor, o vídeo deve iniciar aos 23 segundos. Desse modo, ao relacionar o fim da preparação com o início da apresentação poderia levar à falha de sincronização. Por esse motivo, nenhuma aresta foi inserida no grafo, relacionando o fim do evento de preparação de um objeto de mídia com o início do evento de apresentação do mesmo.

Como foi dito anteriormente, o plano de apresentação e o plano de preparação podem ser atualizados durante a execução da aplicação. Por exemplo, caso a aplicação seja reproduzida em um ambiente cujo perfil do usuário é feminino, os eventos de início de apresentação da propaganda (*"adMan"*), relativos ao perfil masculino serão removidos do plano de apresentação.

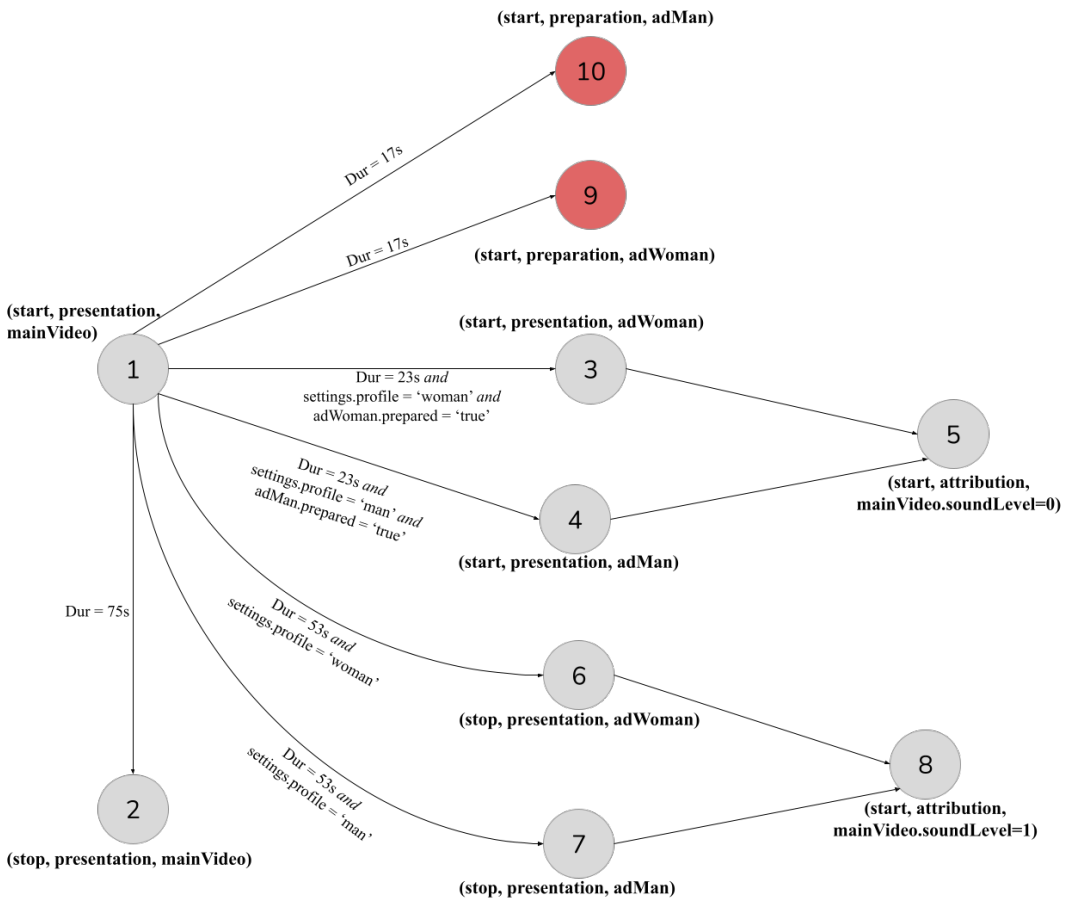

Figura 7.2: Grafo temporal da aplicação de propaganda direcionada atualizado com o evento de preparação.

7.1.2 Preparação de efeitos sensoriais

Esta seção apresenta três aplicações mulsemídia, uma aplicação que sincroniza um efeito de luz com o conteúdo audiovisual ("AppLight"), uma que emprega efeito de aroma ("AppScent"), e outra aplicação que sincroniza efeito de vento com um vídeo ("AppWind"), a fim de demonstrar o evento de preparação aplicado a efeitos sensoriais. Foram criadas duas aplicações diferentes para possibilitar avaliar de forma separada a preparação para cada tipo específico de efeito. As aplicações mulsemídia foram especificadas utilizando a versão 4.0 da linguagem NCL, proposta nesta tese, que permite a especificação de efeitos através do elemento `<effect>`.

A aplicação mulsemídia "AppLight" inicia com um vídeo representando uma sala de estar ("living Room"), e no instante 18s um efeito de luz é apresentado, para representar a abertura das cortinas da sala. O início da apresentação do efeito sensorial é sincronizado com uma âncora do vídeo principal, denominada "a1". Esta âncora tem início no instante 18s e possui duração de 5s, correspondente ao intervalo em que a cortina se mantém

aberta. O código da aplicação “AppLight” é apresentado na Listagem 7.2. A duração do efeito sensorial é definida através da propriedade *explicitDur* especificada no descritor utilizado pelo elemento <effect> referente ao efeito de luz (linha 9 da Listagem 7.2).

```
1  <ncl id="AppLight">
2  <head>
3   <regionBase>
4    <region id="rgVideo" width="100%" height="100%"/>
5    <region id="rgLight" location="center:top:front"/>
6   </regionBase>
7   <descriptorBase>
8    <descriptor id="dVideo" region="rgVideo"/>
9    <descriptor id="dLight" region="rgLight" explicitDur="5s"/>
10  </descriptorBase>
11  <connectorBase>
12   <causalConnector id="onBeginStart">
13    <simpleCondition role="onBegin"/>
14    <simpleAction role="start"/>
15   </causalConnector>
16  </connectorBase>
17 </head>
18 <body>
19  <port id="pStart" component="livingRoom"/>
20  <media id="livingRoom" src="media/livingroom.mp4" descriptor="dVideo">
21   <area id="a1" begin="18s"/>
22  </media>
23  <effect id="lightEffect" type="LightType" descriptor="descLight">
24    <property name="color" value="yellow"/>
25   </effect>
26  <link xconnector="onBeginStart">
27   <bind role="onBegin" component= "livingRoom" interface="a1"/>
28   <bind role="start" component="lightEffect"/>
29  </link>
30 </body>
31 </ncl>
```

Listagem 7.2: Aplicação mulsemídia NCL com efeito sensorial de luz.

O vídeo da aplicação acima também é utilizado na aplicação “AppWind”, que sincro-

niza um efeito de vento com o vídeo. Nesta aplicação, o efeito de vento é disparado após 30 segundos do vídeo principal, sincronizando com o momento em que a janela apresentada no vídeo se abre. A apresentação do efeito de vento é finalizada 5 segundos após o seu início. Essa relação de sincronização é especificada através de dois elos NCL, conforme apresentado na Listagem 7.3.

```
1  <ncl id="AppWind">
2  <head>
3   <regionBase>
4    <region id="rgVideo" width="100%" height="100%"/>
5    <region id="rgWind" location="center:top:front"/>
6   </regionBase>
7   <descriptorBase>
8    <descriptor id="dVideo" region="rgVideo" explicitDur="55s"/>
9    <descriptor id="dWind" region="rgWind"/>
10  </descriptorBase>
11  <connectorBase>
12   <causalConnector id="onBeginStart">
13    <simpleCondition role="onBegin"/>
14    <simpleAction role="start"/>
15   </causalConnector>
16   <causalConnector id="onEndStop">
17    <simpleCondition role="onEnd"/>
18    <simpleAction role="stop"/>
19   </causalConnector>
20  </connectorBase>
21 </head>
22 <body>
23  <port id="pStart" component="mainVideo"/>
24  <media id="mainVideo" src="media/livingroom.mp4" descriptor="dVideo">
25     <area id="a1" begin="30s" end="35s"/>
26  </media>
27  <effect id="windEffect" type="WindType" descriptor="descWind"/>
28  <link xconnector="onBeginStart">
29   <bind role="onBegin" component="mainVideo" interface="a1"/>
30   <bind role="start" component="windEffect"/>
31  </link>
32  <link xconnector="onEndStop">
```

```
33    <bind role="onEnd" component= "mainVideo" interface="a1"/>
34    <bind role="stop" component="windEffect"/>
35   </link>
36 </body>
37 </ncl>
```

Listagem 7.3: Aplicação mulsemídia NCL com efeito sensorial de vento.

A aplicação "AppScent" sincroniza um efeito de aroma de mar com o conteúdo de vídeo apresentado na tela. O vídeo apresenta diversas paisagens, e no momento em que aparece um cenário de mar (27 segundos depois do início do vídeo), o aroma é liberado pelo renderizador de efeito. O efeito de aroma na aplicação "AppScent" tem duração de 15 segundos, correspondendo à duração da apresentação do vídeo de mar exibido na tela. A sincronização do efeito de aroma com o vídeo é dado por dois elos temporais, conforme descrito na Listagem 7.4. Para a definição do tipo de aroma, utilizou-se o padrão de descrição de aromas, definido pelo padrão MPEG-V (anexo 2.4 da ISO/IEC 23005-6) [42].

```
1  <ncl id="AppScent">
2  <head>
3   <regionBase>
4    <region id="rgVideo" width="100%" height="100%"/>
5    <region id="rgScent" location="center:top:front"/>
6   </regionBase>
7   <descriptorBase>
8    <descriptor id="dVideo" region="rgVideo" explicitDur="42s"/>
9    <descriptor id="dScent" region="rgScent"/>
10  </descriptorBase>
11  <connectorBase>
12   <causalConnector id="onBeginStart">
13    <simpleCondition role="onBegin"/>
14    <simpleAction role="start"/>
15   </causalConnector>
16   <causalConnector id="onEndStop">
17    <simpleCondition role="onEnd"/>
18    <simpleAction role="stop"/>
19   </causalConnector>
20  </connectorBase>
21 </head>
```

```
22 <body>
23  <port id="pStart" component="mainVideo"/>
24  <media id="mainVideo" src="video.mp4" descriptor="dVideo">
25   <area id="a1" begin="27s" end="42s"/>
26  </media>
27  <effect id="scentEffect" type="ScentType" descriptor="descScent">
28     <property name="scent" value="urn:mpeg:mpeg-v:01-SI-ScentCS-NS:sea_breeze"
    />
29  </effect>
30  <link xconnector="onBeginStart">
31   <bind role="onBegin" component="mainVideo" interface="a1"/>
32   <bind role="start" component="scentEffect"/>
33  </link>
34  <link xconnector="onEndStop">
35   <bind role="onEnd" component= "mainVideo" interface="a1"/>
36   <bind role="stop" component="scentEffect"/>
37  </link>
38 </body>
39 </ncl>
```

Listagem 7.4: Aplicação mulsemídia NCL com efeito sensorial de aroma.

Assim como ocorre com as aplicações multimídia, durante a análise do documento NCL que especifica a aplicação mulsemídia, o formatador constrói um grafo temporal hipermídia para modelar o comportamento temporal de cada aplicação (Figura 7.3 para o efeito de luz, Figura 7.4 para o efeito de vento e Figura 7.5) para o efeito de aroma. Como o efeito sensorial é modelado como um nó no documento mulsemídia, o autor da aplicação pode especificar a ocorrência de eventos para controlar a exibição de tais efeitos, de forma similar ao que é feito com nós de conteúdo. Desse modo, o HTG foi estendido para permitir que um vértice represente também a ocorrência de uma transição sobre um evento aplicado a um efeito sensorial.

Com base no grafo temporal da aplicação, o formatador é capaz de construir seu plano de apresentação, que contém os instantes em que os objetos de mídia e efeitos sensoriais devem ser apresentados em cada aplicação. O plano de apresentação da aplicação "AppLight" proposta é apresentado na Tabela 7.3, a Tabela 7.4 apresenta o plano de apresentação da aplicação "AppWind", por fim, a Tabela 7.5 apresenta o plano de apresentação da aplicação "AppScent".

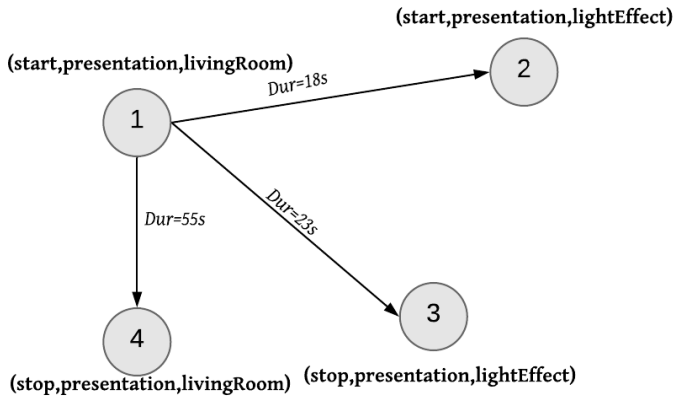

Figura 7.3: Grafo temporal da aplicação mulsemídia “AppLight” com efeito de luz.

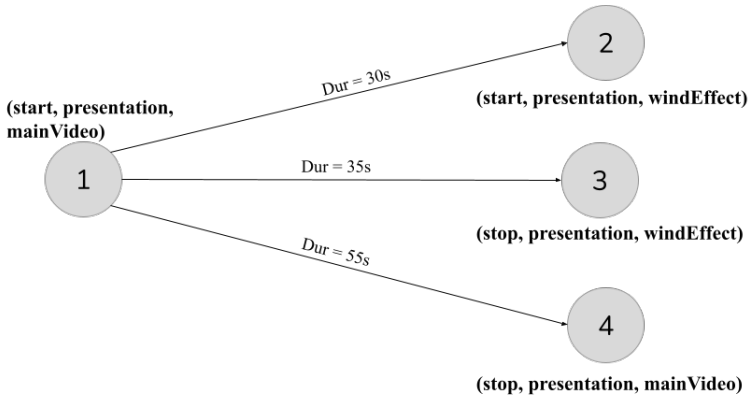

Figura 7.4: Grafo temporal da aplicação mulsemídia “AppWind” com efeito de vento.

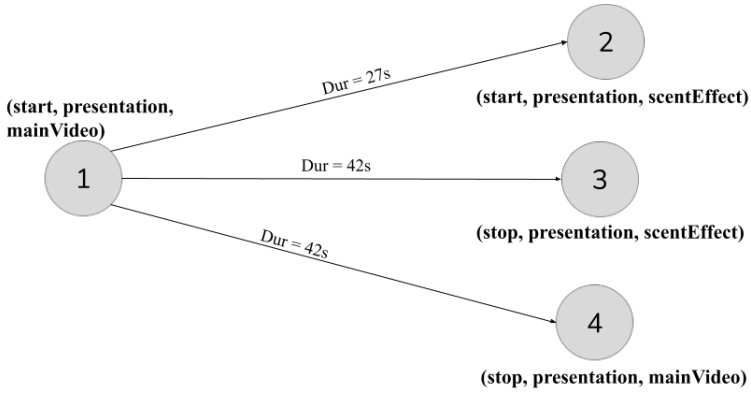

Figura 7.5: Grafo temporal da aplicação mulsemídia “AppScent” com efeito de aroma.

Como foi dito anteriormente, a preparação de um efeito sensorial, depende do tipo de efeito. Neste trabalho, considerou-se que o tempo de preparação do efeito de luz consiste no tempo gasto entre o envio da mensagem para o atuador, e a efetiva ativação do mesmo. Para a renderização do efeito de luz, utilizou-se a lâmpada inteligente Yeelight

Tabela 7.3: Plano de apresentação inicial da aplicação mulsemídia "AppLight".

Instante	**Transição**	**Evento**	**Objeto**
0s	*start*	*presentation*	*livingRoom*
18s	*start*	*presentation*	*lightEffect*
23s	*stop*	*presentation*	*lightEffect*
55s	*stop*	*presentation*	*livingRoom*

Tabela 7.4: Plano de apresentação inicial da aplicação mulsemídia "AppWind".

Instante	**Transição**	**Evento**	**Objeto**
0s	*start*	*presentation*	*mainVideo*
30s	*start*	*presentation*	*windEffect*
25s	*stop*	*presentation*	*windEffect*
55s	*stop*	*presentation*	*livingRoom*

Tabela 7.5: Plano de apresentação inicial da aplicação mulsemídia "AppScent".

Instante	**Transição**	**Evento**	**Objeto**
0s	*start*	*presentation*	*mainVideo*
27s	*start*	*presentation*	*scentEffect*
42s	*stop*	*presentation*	*scentEffect*
42s	*stop*	*presentation*	*mainVideo*

[1]. A lâmpada Yeelight dá suporte a comandos de acionamento e configuração de cor do efeito de luz através de um protocolo próprio que envia mensagens utilizando uma conexão TCP. Como o dispositivo que executa a aplicação mulsemídia e a lâmpada estão na mesma rede local, o tempo para transmissão da mensagem do formatador para a lâmpada levou em média 0.9 segundos durante os testes. Dessa forma, nesta tese, optou-se por não preparar o efeito de luz, visto que seu tempo de ativação é inferior a 1 segundo. O plano de preparação para a aplicação "AppLight" pode ser visto na Tabela 7.6.

Tabela 7.6: Plano de preparação inicial da aplicação mulsemídia "AppLight".

Instante	**Transição**	**Evento**	**Objeto**
0s	start	preparation	livingRoom

O efeito de vento foi renderizado utilizando um ventilador controlado por um Arduino, que recebe comandos via protocolo MQTT. O *broker* MQTT pode ser implementado tanto na rede local, quanto em servidores remotos. Quando implementado em servidores remotos, pode ocorrer atraso na publicação e leitura de tópicos. Dessa forma, nesta tese, foi implementado um *broker* local para comunicação entre o formatador e o ventilador. Após a realização dos testes, foi possível observar que o atraso entre o formatador enviar

[1]https://www.yeelight.com/

o comando de início de apresentação para o *player* de vento e a renderização do efeito foi em média 297 milissegundos. Entretanto, é importante notar que para o Arduino receber o comando, este deve estar conectado e inscrito no tópico de interesse. Portanto, foi definido como 1 segundo o tempo de preparação do efeito de vento, para permitir a conexão antecipada do dispositivo com o *broker* MQTT, conforme apresentado na Tabela 7.7.

Tabela 7.7: Plano de preparação inicial da aplicação mulsemídia "AppWind".

Instante	**Transição**	**Evento**	**Objeto**
0s	start	preparation	mainVideo
29s	start	preparation	windEffect

Já a preparação do efeito de aroma depende tanto do tempo de percepção do usuário, quanto do tempo para enviar o comando de ativação ao dispositivo. Nesta tese, considerou-se que o tempo de preparação do efeito de aroma é algo dependente do usuário. Dessa forma, antes da execução da aplicação, o usuário deve realizar a calibragem do renderizador de aroma, conforme apresentado na Seção 6.2. Após a execução da calibragem do dispositivo, o tempo de preparação é armazenado em um arquivo de configuração, que é lido pelo formatador no momento de execução da aplicação mulsemídia. Os testes realizados com efeito de aroma demonstraram que o usuário leva em média 12 segundos para perceber o efeito, após o formatador executar uma ação de *start* na apresentação do efeito em questão. Portanto, para a construção do plano de preparação da aplicação "AppScent" (Tabela 7.8), considerou-se o tempo de preparação do efeito de aroma igual a 12s, assim o instante de início da preparação do efeito deve ser 15s (27s - 12s).

Tabela 7.8: Plano de preparação inicial da aplicação mulsemídia "AppScent".

Instante	**Transição**	**Evento**	**Objeto**
0s	start	preparation	mainVideo
15s	start	preparation	scentEffect

Após a construção do plano de preparação da aplicação "AppScent", o HTG é modificado, através da inclusão dos vértices que representam o evento de preparação aplicado ao efeito de aroma. Conforme ilustrado na Figura 7.6, o vértice 5 (em vermelho) representa o evento de preparação inserido no HTG. Diferente do que ocorre com a preparação de objetos de mídia tradicionais, o fim do evento de preparação de um efeito sensorial, implica necessariamente o início do evento de apresentação do mesmo. Esta característica é representada pela aresta que liga os vértices 5 e 2 na Figura 7.6.

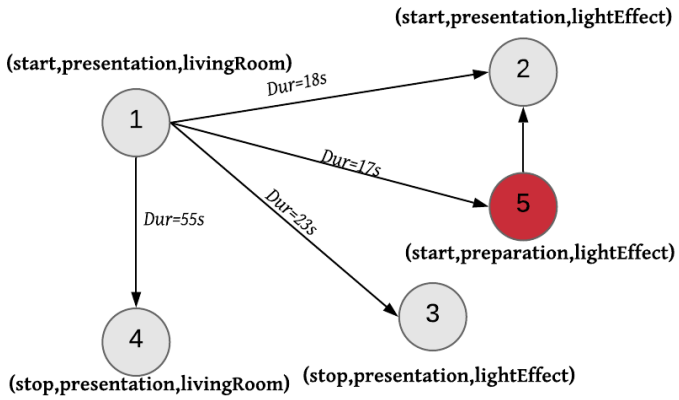

Figura 7.6: Grafo temporal da aplicação mulsemídia com efeito de luz, atualizado com o evento de preparação.

7.1.3 Aplicação mulsemídia: autoria e execução

O ciclo de vida de uma aplicação mulsemídia passa por três fases: autoria, transmissão e execução. Na fase de autoria, o autor da aplicação especifica as mídias e efeitos sensoriais que a compõem, e também as relações de sincronização temporal e espacial entre os componentes da aplicação. O autor pode especificar o documento mulsemídia utilizando uma linguagem de autoria, ou então utilizando uma ferramenta de autoria [74, 19, 25].

STEVE [24] é uma ferramenta de autoria de aplicações mulsemídia, que se baseia no modelo denominado MultiSEM (do inglês, *Multimedia Sensory Effect Model*). O MultiSEM é um modelo conceitual para representar efeitos sensoriais em aplicações mulsemídia interativas, com sincronização temporal baseada em eventos. O editor STEVE tem como objetivo possibilitar que usuários sem conhecimento em linguagens de autoria multimídia e modelos criem aplicações mulsemídia para sistemas de TV Digital e web. Ao utilizar o STEVE, o autor da aplicação pode gerar um documento NCL 4.0, compatível com a versão do Ginga apresentado nesta tese.

Na fase de execução da aplicação, o documento mulsemídia é processado pelo formatador mulsemídia, e apresentado ao usuário final. Nesta fase, o formatador se comunica com *players* de mídia e renderizadores de efeitos sensoriais, para permitir a apresentação da aplicação, conforme especificado pelo autor.

O caso de uso desta seção apresenta as fases de autoria e execução de uma aplicação mulsemídia especificada em NCL 4.0. A aplicação proposta foi desenvolvida utilizando a ferramenta STEVE, e executada na implementação do *middleware* Ginga com suporte a

efeitos sensoriais, proposta nesta tese.

Para demonstrar a utilização integrada do STEVE com o Ginga, foi criada uma aplicação mulsemídia que apresenta efeitos de luz e vento sincronizados com um conteúdo audiovisual. Na aplicação proposta inicialmente é apresentado um vídeo de fundo representando uma sala de estar. Caso o usuário interaja com a aplicação pressionando o botão vermelho, um efeito de luz da cor vermelha é apresentado, juntamente com um vídeo de animação representando a lareira acesa, e um áudio. Além disso, após 19 segundos do início do vídeo inicial, um efeito de vento é disparado juntamente com um conteúdo de áudio, sincronizados com a abertura da janela no vídeo principal. A Figura 7.7 apresenta a visão estrutural da aplicação descrita.

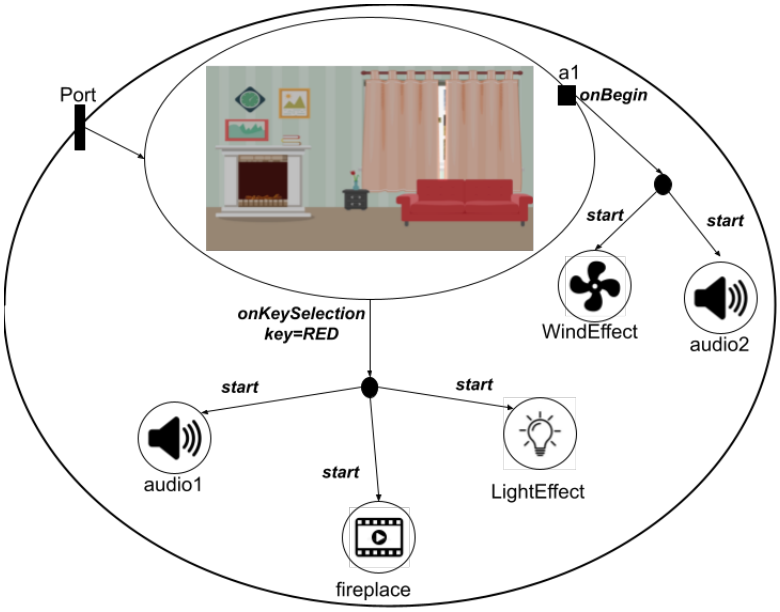

Figura 7.7: Visão estrutural da aplicação mulsemídia de caso de uso.

Utilizando a ferramente STEVE, o autor pode importar objetos de mídia para compor uma aplicação, e especificar as relações temporais entre eles. Ao adicionar uma mídia da aplicação, o autor pode posicioná-la em uma linha temporal, e especificar relações como "ao iniciar uma mídia inicia outra" ou "iniciar um efeito ao finalizar a apresentação de uma mídia". A Figura 7.8 apresenta o STEVE sendo utilizado para criar as relações temporais entre o vídeo principal, o efeito de vento e um elemento de áudio. Além disso, a ferramenta possibilita a especificação de interatividade na aplicação, identificando mídias com as quais o usuário da aplicação pode interagir.

A ferramenta permite especificar também as propriedades gerais de apresentação do efeito como posição e intensidade. Para o efeito de luz, o STEVE permite especificar também a cor da luz que deve ser apresentada (Figura 7.9). A ferramenta também

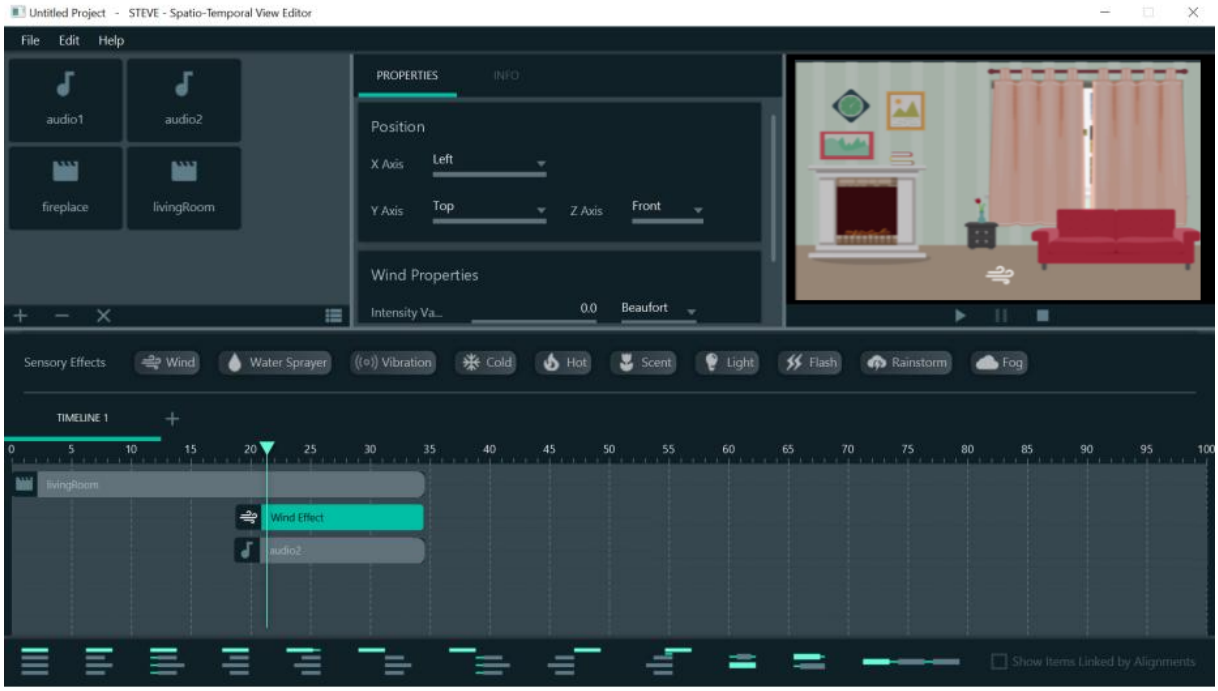

Figura 7.8: Tela da ferramenta STEVE para construção da aplicação mulsemídia com efeito de vento e luz.

apresenta uma pré-visualização da aplicação em construção, indicando tanto o conteúdo de vídeo quanto o efeito apresentado, de acordo com o instante de tempo selecionado na linha temporal. Ao gerar o documento NCL 4.0, a cor especificada pelo autor é convertida em um código RGB.

O ambiente de execução da aplicação é composto por um computador executando o Ginga com suporte a efeitos sensoriais, uma lâmpada inteligente *Yeelight*, e um ventilador controlado por um Arduino. Neste caso de uso, o conteúdo de vídeo estava armazenado localmente, e desta forma, não foi necessária a preparação do conteúdo de vídeo, apenas a preparação do efeito de vento. A Figura 7.10 apresenta a configuração do ambiente de execução utilizado para reproduzir a aplicação. A execução da aplicação de caso de uso pode ser vista neste link[2].

7.2 Avaliação

Esta seção apresenta as avaliações da utilização de preparação automática aplicada a conteúdo de mídia tradicional e a efeitos sensoriais. Para a realização dos testes, foi utilizada a versão da implementação de referência do Ginga-NCL modificada por esta tese. Os testes de preparação de conteúdo foram realizados em dois ambientes de execução

[2] `https://youtu.be/zm0ghb9eito`

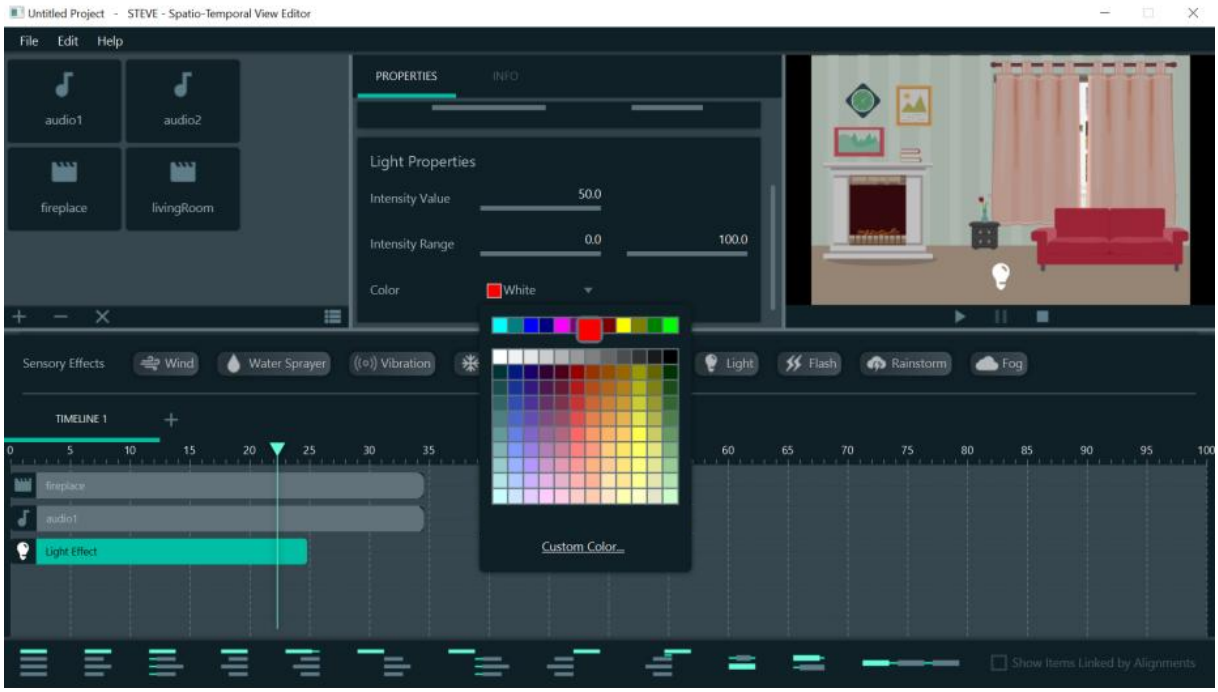

Figura 7.9: Tela da ferramenta STEVE para seleção da cor do efeito de luz.

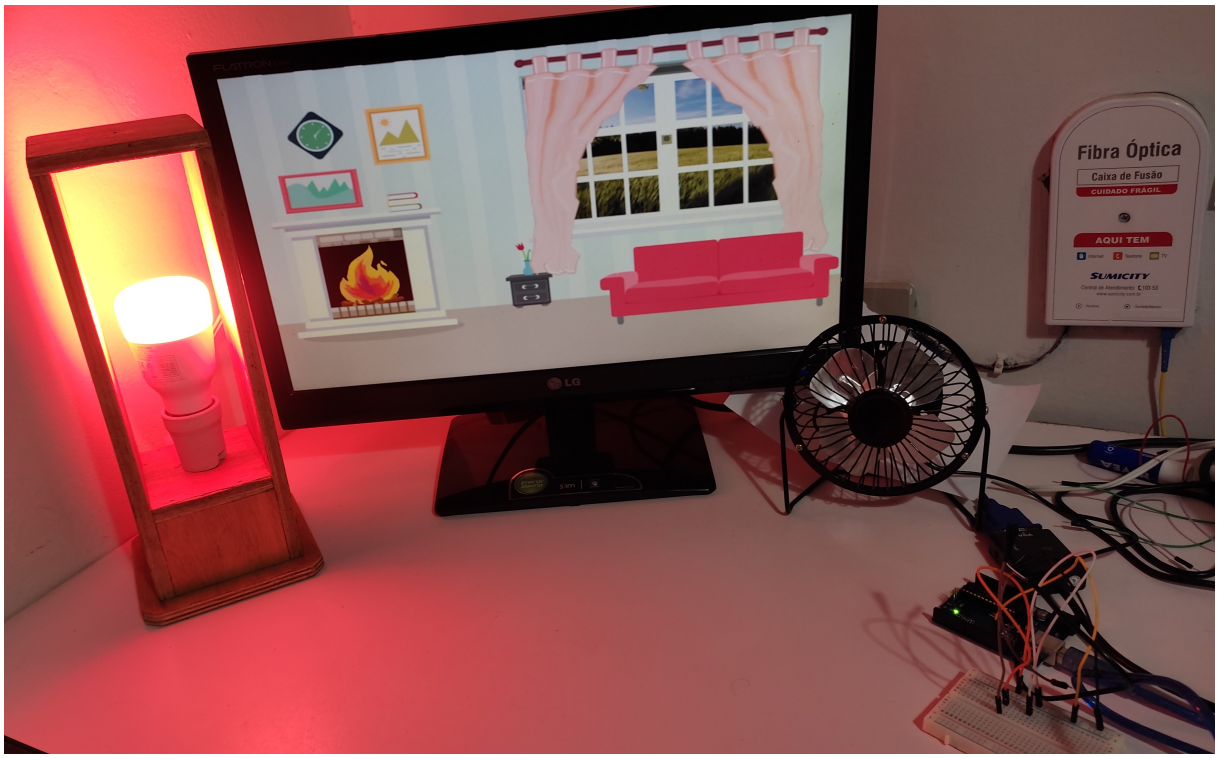

Figura 7.10: Ambiente de execução da aplicação mulsemídia.

diferentes, que simulam o dispositivo receptor da aplicação. O primeiro ambiente (A1) é um computador Dell Intel Core i7-8550U CPU 1.80GHz, com HD de 1TB e 16 GB de RAM. Já o segundo ambiente de execução (A2), é um contêiner LXC rodando sobre um Raspberry Pi 4 Modelo B, 4 GB de RAM e processador quadcore Cortex-A72. E as

aplicações mulsemídia foram executadas apenas no Ambiente A1.

Aplicações multimídia, como a de propaganda direcionada podem ser executadas em set-top boxes de TV Digital, ou televisores com receptor integrado. Alguns desses televisores possuem como característica um processador com arquitetura ARM (Acorn RISC Machine), 32 ou 64-bit, CPU dual ou quad-core, e memória RAM de 2GB [2, 3, 1]. Nos televisores, apenas parte da CPU fica disponível para as aplicações, uma vez que estes também precisam fazer o processamento do conteúdo recebido pelo canal de radiodifusão. Dessa forma, a configuração do contêiner utilizado nos testes foi de 1 GB de RAM e CPU de um núcleo.

A renderização do efeito de luz foi feita utilizando o dispositivo Yeelight, o Moodo foi empregado para a renderização do efeito de aroma, e o efeito de vento foi renderizado por um ventilador. Os testes foram separados em testes de preparação de mídia (Seção 7.2.1) e teste de preparação de efeito (Seção 7.2.1). A execução dos testes foi realizada em duas etapas distintas, na primeira, as aplicações descritas na Seção 7.1 foram executadas no Ginga-NCL com o mecanismo de preparação automática desativado. Na segunda etapa, as aplicações foram executadas com o mecanismo de preparação automática de objetos de mídia e efeitos sensoriais.

7.2.1 Preparação de conteúdo de mídia

Visando avaliar o impacto da preparação em aplicações multimídia que sincronizam conteúdo *broadband* com a transmissão *broadcast*, foram executados testes utilizando a aplicação de propaganda direcionada descrita na Seção 7.1.1. Na primeira fase de testes, a aplicação de propagando direcionada foi executada em uma versão do Ginga-NCL que não implementa preparação automática de conteúdo. Esta versão também não dá suporte à utilização do atributo *prepared*, para verificar se o conteúdo está de fato preparado. Na segunda fase, a aplicação foi executada na implementação do Ginga, modificada por esta tese, que utiliza o plano de preparação. Cada fase de teste consiste em 50 execuções da aplicação para obter o tempo médio de chaveamento entre o conteúdo *broadcast* e o *broadband*. Esse tempo de chaveamento está diretamente relacionado ao atraso na apresentação do conteúdo direcionado.

De acordo com o padrão HbbTV [36], o tempo de chaveamento em uma aplicação IBB, é o período entre o instante no qual a apresentação do vídeo *broadcast* deixa de ser apresentado e o instante em que o conteúdo de áudio ou de vídeo transmitido via *broadband* começa a ser exibido. Durante a execução do conteúdo *broadband*, o dispositivo receptor

permanece recebendo e processando o conteúdo *broadcast*, da mesma forma que acontece nos set-top boxes de TV Digital.

Os testes de preparação de conteúdo de mídia foram executados com diferentes taxas de transmissão de dados, com base na última pesquisa da Agência Nacional de Telecomunicações (ANATEL) sobre o serviço de comunicação multimídia no Brasil [8]. De acordo com a pesquisa, a taxa de transmissão média no país é de 24.62 Mbps. Além disso, no estado brasileiro onde a taxa de transmissão é a menor, esse valor chega a aproximadamente 8Mbps.

Considerando o ambiente de testes A1, o tempo médio de chaveamento no cenário com taxa de transmissão igual a 8 Mbps, foi de 0,02 segundos para as execuções da aplicação com a utilização do mecanismo de preparação automática. Enquanto isso, nas execuções sem a preparação automática, esse tempo chegou a 9,9 segundos. Nos testes com taxa de transmissão de 15 Mbps, o uso da preparação automática reduziu em 4,5 segundos o tempo de chaveamento, comparado às execuções sem preparação automática. Com o aumento da taxa de transmissão para 25 Mbps, o tempo de chaveamento sem preparação reduziu para 2,46 segundos, porém manteve-se 98 vezes maior do que o tempo nos casos com preparação automática, nessa mesma taxa. A Figura 7.11 apresenta a comparação dos tempos médios de chaveamento entre o conteúdo *broadcast* e o conteúdo de propaganda direcionada para cada taxa de transmissão descrita acima com intervalo de confiança de 95%. Como os valores da média sem preparação e com preparação possuem escalas muito diferentes, utilizou-se escala logarítmica no eixo y para representar o tempo médio de chaveamento em milissegundos.

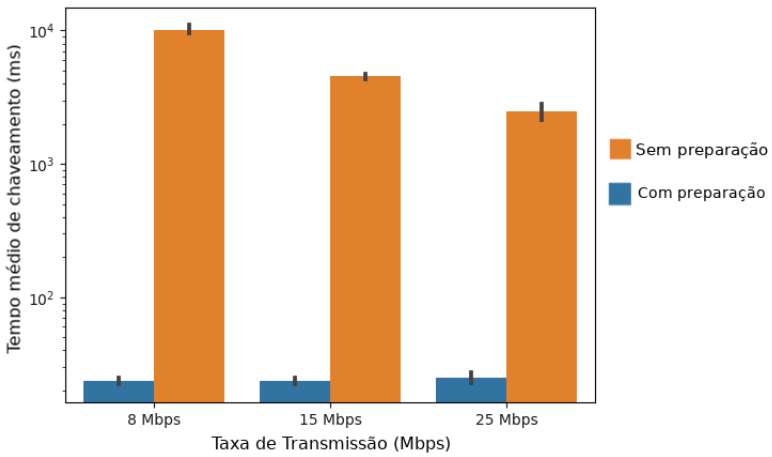

Figura 7.11: Tempo médio de chaveamento com e sem preparação automática, considerando diferentes taxas de transmissão, realizados no ambiente de teste A1. Os valores do eixo y estão representados em escala logarítmica.

Os testes de execução das aplicações no ambiente de teste A2 (contêiner Ginga executando no Raspberry) demonstraram que no cenário onde a taxa de transmissão era igual a 8 Mbps, a utilização de preparação automática reduziu em média 10 segundos o tempo médio de chaveamento comparado aos testes sem preparação automática. Já para uma taxa de transmissão de 15 Mbps, o tempo médio de chaveamento foi igual a 5,47 segundos pros casos sem preparação automática, e 0,04 segundos para os testes com a solução proposta nesta tese. Por fim, no cenário onde a taxa de transmissão era igual a 25 Mbps, o tempo médio de chaveamento foi igual a 2,21 segundos para as execuções sem preparação automática, e 0,039 segundos nos testes que empregam o plano de preparação proposto nesta tese. A comparação do tempo médio de chaveamento para cada taxa de transmissão é apresentada na Figura 7.12. Optou-se por representar o eixo y do gráfico na Figura 7.12 em escala logarítmica pelo mesmo motivo descrito anteriormente.

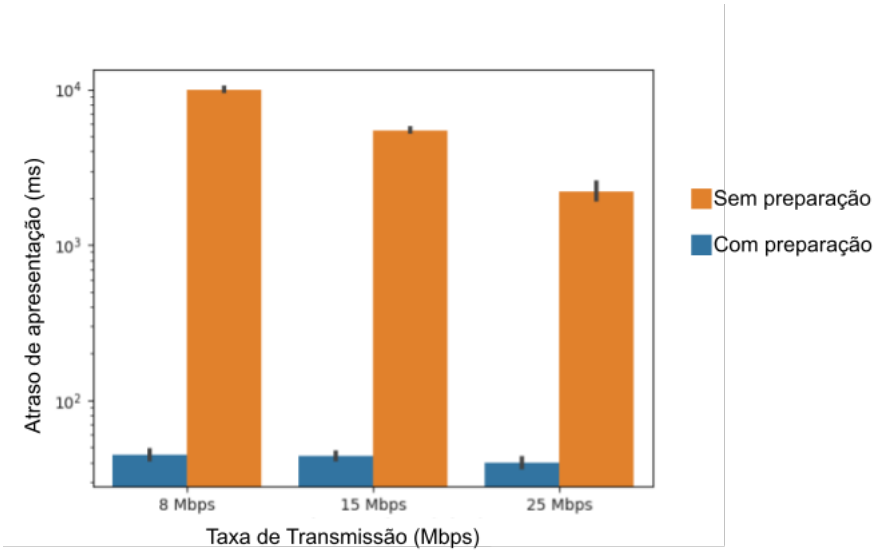

Figura 7.12: Tempo médio de chaveamento com e sem preparação automática, considerando diferentes taxas de transmissão, realizados no ambiente de teste A2. Os valores do eixo y estão representados em escala logarítmica.

Dessa forma, é possível concluir que o mecanismo de preparação proposto nesta tese foi capaz de reduzir o atraso na entrega de conteúdos de mídia, auxiliando a manter as relações de sincronização durante a execução de uma aplicação multimídia. Além disso, verificou-se que o desempenho do mecanismo de preparação automática mostrou-se eficiente tanto em dispositivos com maior capacidade computacional, quanto em dispositivos mais limitados, como os utilizados no ambiente de TV Digital.

Em NCL, é possível especificar aplicações multimídia onde o usuário pode selecionar um conteúdo a ser exibido através de eventos de interação. Além disso, a linguagem dá suporte à adaptação de conteúdo, através de elementos <switch>. Para esses tipos de

aplicação, onde o conteúdo a ser exibido é definido em tempo de execução, o formatador irá preparar cada conteúdo ligado a cada opção. Dessa forma, uma terceira fase de testes foi realizada para verificar o impacto de múltiplas preparações de conteúdo ocorrendo simultaneamente. No ambiente de testes A1, foram realizados testes de preparação simultânea preparando 10 vídeos simultaneamente, 20 vídeos, 30 vídeos e por fim, 50 vídeos simultâneos. Já no ambiente de teste A2, representado pelo contêiner LXC executando no Raspiberry, foi possível realizar no máximo 30 preparações simultâneas. Isso porque, ao aumentar a taxa, o dispositivo começava a apresentar baixo desempenho, inclusive travando diversas vezes. A Tabela 7.9 apresenta o tempo médio de preparação para cada situação, considerando cada ambiente de teste. Para esses testes foi utilizada uma rede cabeada com taxa de transmissão igual a 20 Mbps. As aplicações com preparação simultânea fora executadas 50 vezes cada uma.

Tabela 7.9: Duração média de preparação considerando preparações simultâneas de conteúdo de vídeo, com taxa de transmissão igual a 20Mbps

Nº de preparações simultâneas	**Ambiente A1**		**Ambiente A2**	
	Duração	Desvio Padrão	Duração	Desvio Padrão
10	2,6 s	0,27	7,1 s	1,27
20	4,8 s	0,62	10,4 s	4,6
30	9,4 s	2,2	23,6 s	1,2
50	16,5 s	2,14	-	-

7.2.2 Preparação de efeitos sensoriais

Em relação à utilização do evento de preparação para efeitos sensoriais, foram realizados alguns testes com as aplicações mulsemídia descritas nas Seção 7.1.2. Os testes de preparação de efeitos foram realizados em duas etapas, sendo a primeira etapa composta por execuções das aplicações mulsemídia sem a preparação automática; e a segunda etapa, com os testes de execução da aplicação mulsemídia empregando a preparação automática de efeitos.

Em cada rodada de teste, foram medidos os atrasos de apresentação de cada efeito sensorial. Conforme dito anteriormente, o efeito de aroma leva um tempo para ser percebido pelo usuário, após o atuador liberar o efeito. Dessa forma, o atraso de apresentação do efeito de aroma foi calculado considerando o intervalo de tempo entre o formatador iniciar a apresentação do efeito, e o instante de tempo em que o usuário percebeu o efeito de aroma. Já para o efeito de luz, o atraso de apresentação foi calculado considerando o

tempo entre o formatador mulsemídia iniciar a apresentação do efeito e o tempo em que atuador recebe o comando de ativação.

Para os testes do efeito de luz, a aplicação mulsemídia *AppLight*, descrita na seção 7.1.2, foi executada cinquenta vezes para cada etapa (com e sem preparação automática). Após a execução dos testes foi possível observar que o atraso médio de apresentação do efeito de luz sem preparação automática foi igual a 0.92 segundos. Já com a preparação automática, o atraso médio foi igual a 0.8 segundos. Dessa forma, nesta tese optou-se por não preparar o efeito de luz, uma vez que o ganho da preparação automática foi de apenas 0,1 segundo, comparado ao cenário sem preparação.

É importante destacar que o dispositivo renderizador de efeito de luz utilizado nos testes e o formatador mulsemídia se encontram na mesma rede local, e a comunicação entre eles se dá por um protocolo de comunicação próprio utilizando uma conexão TCP. E esse tempo para comunicação pode sofrer alterações, de acordo com o protocolo de comunicação implementado pelo renderizador de efeitos. Portanto, dependendo do atraso para envio de comandos de ativação imposto pelo protocolo utilizado, a preparação automática do efeito pode ou não ser realizada.

Os testes do efeito de aroma foram executados com dois usuários distintos, e a aplicação *AppScent* foi executada cinco vezes para cada usuário. Como o aroma leva um tempo para se dispersar no ambiente, cada execução da aplicação foi feita com um intervalo de tempo entre elas. Isto foi feito para evitar que o aroma de uma execução anterior da aplicação influenciasse na rodada de teste posterior. Na aplicação mulsemídia com efeito de aroma, o usuário era instruído a apertar um botão do controle remoto quando identificasse que o aroma foi liberado. Com isso, foi possível calcular a diferença entre o instante que o formatador inicia o efeito, e o instante que o usuário o percebe.

A fim de extrair apenas o tempo de percepção do usuário, e excluir o tempo entre o usuário perceber o aroma e selecionar o botão, foram realizados alguns testes com os mesmos usuários. Nestes testes, foi utilizada uma aplicação NCL simples, composta por duas imagens. A primeira imagem contém uma mensagem informativa para o usuário, solicitando que o mesmo pressione a tecla "Enter" do controle quando um círculo vermelho for exibido na tela, e a segunda imagem é um círculo vermelho. Essa aplicação foi executada cinco vezes para cada usuário, e em cada execução foi escolhido um tempo diferente para a apresentação da imagem do círculo. Desse modo, foi possível coletar o atraso entre o usuário visualizar a imagem, e pressionar o botão do controle. A aplicação NCL descrita acima é especificada na Listagem 7.5.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <ncl id="aplSelection" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
3   <head>
4    <regionBase>
5     <region id="rg1" width="100%" height="100%" zIndex = "0" />
6     <region id="rg2" width="35%" height="45%" top="35%" left="50%"/>
7    </regionBase>
8    <descriptorBase>
9      <descriptor id="desc1"  region="rg1"/>
10    <descriptor id="desc2"  region="rg2"/>
11   </descriptorBase>
12   <connectorBase>
13     <causalConnector id="onBeginStart">
14       <simpleCondition role="onBegin" />
15       <simpleAction role="start" max="unbounded"/>
16     </causalConnector>
17     <causalConnector id="onEndStop">
18       <simpleCondition role="onEnd" />
19       <simpleAction role="stop" max="unbounded"/>
20     </causalConnector>
21     <causalConnector id="onSelectionStop">
22         <simpleCondition role="onSelection" key="ENTER"/>
23         <simpleAction role="stop" delay="5s"/>
24     </causalConnector>
25   </connectorBase>
26   </head>
27   <body>
28    <port id="pStart" component="background" />
29    <media id="background" src="media/message.png" descriptor="desc1">
30      <area id="a1" begin="7s"/>
31    </media>
32    <media id="circle" src="media/circle.png" descriptor="desc2"/>
33    <link xconnector="onBeginStart">
34      <bind role="onBegin" component="background" interface="a1"/>
35      <bind role="start" component="circle" />
36    </link>
37    <link xconnector="onSelectionStop">
38      <bind role="onSelection" component="circle"/>
```

```
39      <bind role="stop" component="circle" />
40    </link>
41    <link xconnector="onEndStop">
42      <bind role="onEnd" component="circle"/>
43      <bind role="stop" component="background" />
44    </link>
45  </body>
46  </ncl>
```

Listagem 7.5: Aplicação NCL para cálculo do atraso de seleção pelo usuário.

Após a execução dos testes, foi possível observar que os usuários levaram em média 0,77 segundos para selecionar a tecla "Enter", após a apresentação da imagem do círculo vermelho. Nos testes da aplicação com efeito de aroma sem a preparação automática, observou-se um atraso médio de 11,9s para apresentação do efeito. Já nos testes onde a aplicação mulsemídia foi executada utilizando a preparação automática, este atraso médio foi reduzido para 3,84s. A Figura 7.13 apresenta os resultados dos testes separados por usuário, e também o tempo médio geral, com intervalo de confiança de 95%.

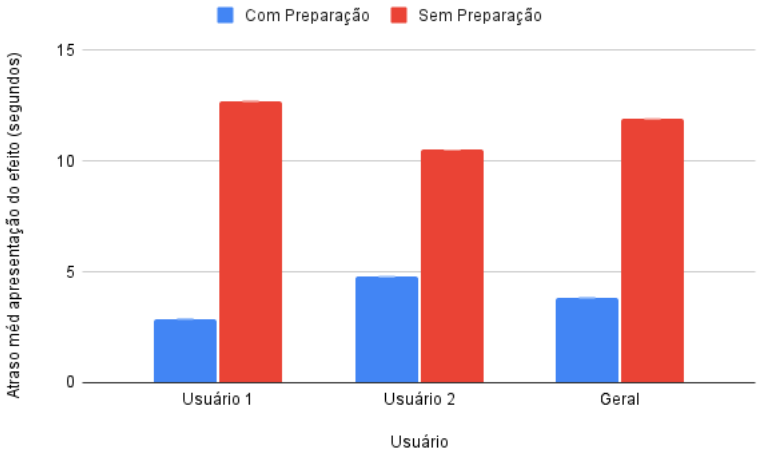

Figura 7.13: Atraso na apresentação do efeito de aroma, com e sem preparação automática, considerando diferentes usuários.

7.2.3 Considerações sobre os resultados

Os testes mostraram que a preparação automática de conteúdo de mídia e efeitos sensoriais reduzem de forma significativa o atraso na apresentação de aplicações mulsemídia. Além disso, foi possível observar que, para efeitos sensoriais que tem sua percepção instantânea

por parte do usuário (efeito de vento e de luz, por exemplo), o que impacta no atraso de apresentação é apenas o protocolo de comunicação utilizado pelo atuador.

A calibragem do efeito de aroma permitiu que o tempo de preparação se adequasse a cada usuário, e também ao ambiente de execução da aplicação mulsemídia. Essa calibragem permite que o formatador considere tanto o tempo para envio do comando de ativação do dispositivo, quanto o tempo de percepção do usuário. O gerenciamento das relações de sincronização de uma aplicação mulsemídia, deve tentar garantir que tanto o início quanto o fim da apresentação de um efeito ocorram no momento especificado pelo autor da aplicação.

O mecanismo de preparar o fim do efeito, através do envio antecipado do comando para desativação do dispositivo, permite reduzir o atraso no fim da apresentação do efeito. Porém, para o efeito de aroma, o fim completo do efeito, não pode ser garantido pelo formatador. Isto porque, depende de condições físicas do ambiente de execução. Por exemplo, em um ambiente aberto, o tempo para dissipação do efeito de aroma é menor, do que num ambiente pequeno e completamente fechado. Uma opção para tentar garantir que o fim da apresentação do efeito ocorra no momento correto, seria enviar de forma antecipada o comando para o dispositivo renderizador. Durante a fase de testes, foram coletados os atrasos de finalização dos efeitos de vento, luz e aroma. Para o efeito de aroma, considerou-se o fim do efeito, como o momento em que o renderizador do efeito enviou a confirmação de desativação para o formatador. Em todos os casos, o atraso médio de finalização dos efeitos foi inferior a 1 segundo. Dessa forma, nesta tese, optou-se por não preparar a finalização destes efeitos.

Aplicações mulsemídia podem ser compostas por diferentes tipos de efeitos sensoriais. O formatador mulsemídia proposto nesta tese dá suporte apenas à execução e preparação de efeitos de luz, aroma e vento. Caso o formatador receba um documento mulsemídia que especifique outro tipo de efeito diferente dos citados anteriormente, a aplicação será executada sem nenhum erro, porém não apresentará o efeito em questão.

Em relação à preparação de conteúdo de mídia, os testes demonstraram que é possível manter as relações de sincronização temporal de aplicações que sincronizam conteúdo *broadcast* com conteúdo *broadband* ao aplicar o mecanismo de preparação automática. O mecanismo proposto obteve um bom desempenho, mesmo quando executado em um ambiente com menor capacidade de processamento e armazenamento. Ao comparar os resultados dos testes nos dois ambientes diferentes (A1 e A2), também foi possível observar que o fator que impacta mais no atraso de apresentação, é a taxa de transmissão. As

capacidades de computação (quantidade de núcleos de CPU) e armazenamento (memória RAM) não influenciaram tanto no desempenho da preparação.

O mecanismo de preparação automática de mídia proposto nesta tese considera as condições atuais da rede responsável pela transmissão do conteúdo e a capacidade de armazenamento do dispositivo receptor. Nos testes realizados, considerou-se que apenas um usuário requisita o conteúdo direcionado por vez. Por isso, o tempo para acessar o servidor não foi considerado no cálculo do tempo de preparação. Para considerar esse valor, é necessário acrescentá-lo no cálculo do tempo de preparação da Equação 5.1 apresentada no capítulo 5. Esta atualização é deixada como trabalho futuro. Caso o autor da aplicação opte por não apresentar o conteúdo caso não seja possível garantir sua apresentação no momento correto, ele pode utilizar o atributo *prepared* proposto nesta tese. Neste cenário, se o tempo de preparação obtido pelo mecanismo de preparação automática não refletir o atraso real da rede, o conteúdo não será apresentado.

Este capítulo apresentou casos de uso do evento de preparação e algumas avaliações da solução proposta nesta tese. A Seção 7.1.1 apresentou um caso de uso do evento de preparação aplicado a conteúdo de mídia transmitido via *broadband*. Na Seção 7.1.2, foram apresentados três casos de uso de preparação de efeitos sensoriais. Por fim, a Seção 7.1.3 descreveu um caso de uso de autoria e execução de uma aplicação mulsemídia, utilizando a ferramenta de autoria STEVE e o *middleware* Ginga-NCL para execução da aplicação. A avaliação da solução proposta foi apresentada na Seção 7.2.

Capítulo 8

Conclusão

A manutenção das relações de sincronização espaço-temporal em aplicações mulsemídia é um fator importante para a preservação da qualidade de apresentação do conteúdo. Deste modo, faz-se necessária a utilização de mecanismos para garantir que os objetos de mídia e efeitos sensoriais serão apresentados no instante correto, conforme definido pelo autor da aplicação. Estes mecanismos estão relacionados ao carregamento dos *players* de mídia e obtenção do conteúdo das mídias, e acionamento de renderizadores de efeitos sensoriais.

Como certas aplicações são reproduzidas em dispositivos com recursos de rede ou memória limitados, é importante que estes mecanismos considerem tais requisitos em sua implementação. Neste contexto, esta tese propôs a definição de um novo tipo de operação para dar suporte à preparação de conteúdo, levando em conta a quantidade de memória disponível no dispositivo em que a aplicação é executada.

O evento de preparação proposto nesta tese é genérico o suficiente para ser aplicado tanto a objetos de mídia quanto a efeitos sensoriais. Esta característica do evento é importante, uma vez que a incorporação de efeitos sensoriais em aplicações multimídia tem sido empregada atualmente em diferentes propostas na literatura, com o objetivo de proporcionar uma melhor qualidade de experiência ao usuário. A sincronização de efeitos sensoriais com conteúdo audiovisual pode ser especificada através de uma linguagem de autoria, como a linguagem SEM do padrão MPEG-V. Visando permitir a especificação de aplicações mulsemídia utilizando a linguagem NCL, que é a linguagem de codificação padrão de aplicações no Sistema Brasileiro de TV Digital, esta tese apresentou uma extensão à linguagem, através da criação de um novo elemento. A extensão proposta nesta tese faz parte da nova versão da linguagem, denominada NCL 4.0. O elemento proposto permite a modelagem de efeitos sensoriais como entidade de primeira classe dentro do documento mulsemídia. Essa representação permite a especificação de um efeito sensorial

independente do dispositivo físico utilizado para renderização do efeito. Além da extensão na linguagem NCL, esta tese também estendeu o modelo NCM, que é o modelo conceitual utilizado como base para a definição da linguagem NCL.

O mecanismo de preparação proposto nesta tese pode ser aplicado tanto em fase de autoria de um documento mulsemídia, quanto em fase de execução. Para validar a utilização da preparação em fase de autoria, esta tese propõe a incorporação de um novo evento à linguagem NCL. Este novo evento é denominado *preparation* e pode ser caracterizado por três atributos – *occurrences*, *repetition* e *prepared*. Os atributos *occurrences* e *repetition* já existiam na linguagem NCL para os eventos da linguagem – apresentação, seleção e atribuição. Estes atributos informam quantas vezes um evento já ocorreu e quantas vezes ele ainda deverá ocorrer imediatamente após o final da ocorrência atual, respectivamente. Já o atributo *prepared*, inserido nesta tese, informa se um conteúdo de mídia ou âncora de conteúdo está preparado ou não. O atributo *prepared* pode receber o valor *true* caso o conteúdo esteja preparado para apresentação, e *false* caso contrário.

Para garantir que um conteúdo de mídia ou efeito sensorial esteja preparado para ser apresentado, o formatador pode manipular automaticamente o evento *preparation*, com base em um plano de preparação. O plano de preparação é construído com base em grafo temporal, e especifica os instantes em que cada conteúdo de mídia ou efeito sensorial deve ser preparado para ser apresentado. A utilização da operação de preparação em fase de execução foi implementada no *middleware* Ginga-NCL, a fim de inserir o suporte à preparação automática de conteúdo e efeitos com base em um plano de preparação.

O *middleware* Ginga-NCL também foi modificado para permitir a execução de aplicações mulsemídia especificadas utilizando a linguagem NCL 4.0. Foram incorporados dois novos *players* de efeitos, um para efeito de aroma (EffectPlayerScent) e outro para efeito de vento (EffectPlayerWind). Os *players* de efeito implementados nesta tese, se comunicam com os dispositivos físicos, por meio de APIs específica, também implementadas por esta tese. O efeito sensorial de aroma, pode levar um tempo para ser percebido pelo usuário após sua emissão pelo dispositivo atuador. Esse tempo pode variar de usuário para usuário, e também em relação à distância do usuário para o dispositivo emissor de aroma. Considerando esta questão, esta tese implementou um mecanismo de calibragem do efeito de aroma, para obter o tempo necessário de preparação deste efeito. A calibragem é realizada pelo próprio *middleware* Ginga, através de uma aplicação especificada na linguagem NCL 4.0.

Por fim, esta tese apresentou três casos de uso do evento de preparação, aplicado

na formatação de documentos multimídia e mulsemídia. Além disso, foi apresentada uma avaliação da solução através de testes comparando execução de aplicações com e sem a preparação de mídias e efeitos sensoriais. Inicialmente verificou-se o impacto da preparação automática em aplicações que integram transmissão *broadcast* com *broadband*. A preparação simultânea de diferentes conteúdos também foi testada nesta tese. Esta tese também avaliou o uso do mecanismo de preparação aplicado ao efeito de aroma e de luz. A partir desta avaliação foi possível observar que a proposta desta tese realmente reduz os atrasos na entrega do conteúdo de aplicações multimídia transmitidas através da rede, e consequentemente evita as falhas de sincronização durante a execução da aplicação. A preparação automática também possibilitou a redução no atraso de apresentação de efeitos sensoriais.

8.1 Publicações

Esta tese gerou as seguintes publicações:

- Josué, M., Abreu, R., Barreto, F., Mattos, D., Amorim, G., dos Santos, J., and Muchaluat-Saade, D. (2018, June). **Modeling sensory effects as first-class entities in multimedia applications**. In Proceedings of the 9th ACM Multimedia Systems Conference (pp. 225-236). ACM,2018.

- Josué, Marina Ivanov P.; Moreno, Marcelo Ferreira; Muchaluat-Saade, D. C. **Preparation of Media Object Presentation and Sensory Effect Rendering in Mulsemedia Applications**. In: 24rd Brazillian Symposium on Multimedia and the Web, 2018, Salvador. Proceedings of the 24th Brazilian Symposium on Multimedia and the Web - WebMedia '18, 2018.

- Josué, Marina Ivanov P.; Moreno, Marcelo Ferreira; Muchaluat-Saade, D. C. **Mulsemedia preparation: a new event type for preparing media object presentation and sensory effect rendering**. In: Proceedings of the 10th ACM Multimedia Systems Conference. ACM, 2019. p. 110-120.

- Josué, Marina Ivanov P.; Moreno, Marcelo Ferreira; Muchaluat-Saade, D. C. **Automatic Preparation of Media Objects in Multimedia Applications**. Proceedings of the 25th Brazilian Symposium on Multimedia and the Web - WebMedia '19, 2019.

- IVANOV, MARINA; Montevecchi, Eyre ; Abreu, Raphael ; Barreto, Fabio ; Santos, Joel ; MUCHALUAT-SAADE, D. C. **Ambientes multissensoriais aplicados à saúde: desenvolvimento de aplicações e tendências futuras**. Minicursos SBCAS 2020. 1ed.: SBC, 2020, v. , p. 48-89.

8.2 Trabalhos Futuros

Aplicações multimídia podem dar suporte à interação do usuário ou adaptação de conteúdo, por exemplo. Neste tipo de aplicação, a reprodução do conteúdo pode seguir diferentes caminhos e durações de exibição, para diferentes usuários, ou para um mesmo usuário, em outros momentos de exibição do conteúdo. Durante a fase de análise da aplicação e construção do plano de preparação não é possível definir *a priori* qual caminho o usuário irá seguir. Portanto, todas as mídias relacionadas a todos os caminhos possíveis devem ser preparadas para exibição. A prática descrita acima pode levar à presença de "lixo" na memória do receptor, e alto consumo de memória. Neste contexto, faz-se necessária a utilização de um mecanismo semelhante ao *garbage collector*. Este mecanismo é deixado como trabalho futuro.

Uma aplicação mulsemídia pode estimular um ou mais dos cinco sentidos dos seres humanos (audição, visão, tato, olfato e paladar) através de diferentes efeitos sensoriais. Um trabalho futuro importante é a implementação da preparação automática de novos tipos de efeitos sensoriais, além dos efeitos de cheiro, luz e vento, implementados nesta tese.

Aplicações multimídia e mulsemídia podem ser especificadas em diferentes linguagens (NCL, SMIL e HTML, por exemplo). Desse modo, para que o formatador dê suporte à preparação automática de conteúdo em aplicações definidas por diferentes linguagens, é necessária a implementação de um módulo responsável por converter cada uma dessas linguagens em um grafo temporal hipermídia, a fim de permitir a construção do plano de preparação. A implementação desse módulo, para as linguagens, diferentes da NCL, que foi apresentada como prova de conceito nesta tese, é deixado como trabalho futuro.

Do mesmo modo que ocorre com o início da apresentação de certos efeitos sensoriais, a finalização da renderização de um efeito sensorial, pode não ser imediata. Por exemplo, um efeito de aroma leva um certo tempo até ser dispersado no ambiente, após sua renderização ser finalizada pelo formatador. Como trabalho futuro, torna-se necessária a implementação de um mecanismo que garanta que o efeito irá realmente acabar, no

momento definido pelo autor da aplicação mulsemídia.

Referências

[1] Lg intelligent processor a9 vs a7: a9 vs a7 smart processors, specifications and comparison. https://en.tab-tv.com/?p=14681. acessado em 20/08/2021.

[2] Mediatek s900. https://www.mediatek.com/products/digitalTv/s900-mt9950/. acessado em 20/08/2021.

[3] What are the advantages of quad core processor in samsung 7/8/9 f series smart tv? https://www.samsung.com/hk_en/support/tv-audio-video/what-are-the-advantages-of-quad-core-processor-in-samsung-7-8-9-f-series-smart acessado em 20/08/2021.

[4] Abdelli, A.; Badache, N. Context-aware adaptation of multimedia documents for consistent presentations. *Multimedia systems 17*, 6 (2011), 465–486.

[5] Abdelli, A.; Badache, N. Context-aware adaptation of multimedia documents for consistent presentations. *Multimedia systems 17*, 6 (2011), 465–486.

[6] ABNT. Nbr 15606-1: Televisão digital terrestre - codificação de dados e especificações de transmissão de dados e especificações de transmissão para radiodifusão digital. Tech. rep., 2013. ABNT, NBR 15606-1:2013.

[7] ABNT. *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*, feb 2021.

[8] ANATEL. *Relatório de monitoramento do departamento de telecomunicações*, 2018.

[9] Barreto, F. *Uma Proposta de Extensão do Middleware Ginga-NCL para Interação Multimodal e Suporte Multiusuário em Ambientes Hipermídia.* Tese de Doutorado, Programa de Pós-graduação em Computação da Universidade Federal Fluminense, Setembro 2021.

[10] Barreto, F.; de Abreu, R. S.; Montevecchi, E. B. B.; Josué, M. I.; Valentim, P. A.; Muchaluat-Saade, D. C. Extending ginga-ncl to specify multimodal interactions with multiple users. In *Proceedings of the Brazilian Symposium on Multimedia and the Web* (2020), pp. 281–288.

[11] Barreto, F.; Montevecchi, E. B. B.; Abreu, R.; dos Santos, J. A.; Muchaluat-Saade, D. C. Providing multi-user in ncl with useragent and user-profile. In *Anais Estendidos do XXV Simpósio Brasileiro de Sistemas Multimídia e Web* (2019), SBC, pp. 205–206.

[12] Blakowski, G.; Steinmetz, R. A media synchronization survey: Reference model, specification and case studies. *Journal on Selected Areas in Communications 14*, 1 (January 1996), 5–35.

[13] Bormann, C.; Castellani, A. P.; Shelby, Z. Coap: An application protocol for billions of tiny internet nodes. *IEEE Internet Computing 16*, 2 (2012), 62–67.

[14] Boronat, F.; Marfil, D.; Montagud, M.; Pastor, J. Hbbtv-compliant platform for hybrid media delivery and synchronization on single-and multi-device scenarios. *IEEE Transactions on Broadcasting 64*, 3 (2017), 721–746.

[15] Buchanan, M. C.; Zellweger, P. T. Automatic temporal layout mechanisms revisited. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 1*, 1 (2005), 60–88.

[16] Candan, K. S.; Prabhakaran, B.; Subrahmanian, V. Retrieval schedules based on resource availability and flexible presentation specifications. *Multimedia Systems 6*, 4 (1998), 232–250.

[17] Cha, J.; Ho, Y.-S.; Kim, Y.; Ryu, J.; Oakley, I. A framework for haptic broadcasting. *IEEE MultiMedia 16*, 3 (2009), 16–27.

[18] Choi, B.; Lee, E.; Yoon, K. Streaming media with sensory effect. In *Information Science and Applications (ICISA), 2011 International Conference on* (2011), IEEE, pp. 1–6.

[19] Choi, B.; Lee, E.-S.; Yoon, K. Streaming media with sensory effect. In *Information Science and Applications (ICISA), 2011 International Conference on* (2011), IEEE, pp. 1–6.

[20] Chrome, G. https://www.google.com.br/chrome/, 2018.

[21] Chung, S. M.; Pereira, A. L. Timed petri net representation of smil. *IEEE MultiMedia 12*, 1 (2005), 64–72.

[22] Costa, R. M. d. R.; Moreno, M. F.; Gomes Soares, L. F. Intermedia synchronization management in dtv systems. In *Proceedings of the eighth ACM symposium on Document engineering* (2008), ACM, pp. 289–297.

[23] Davison, B. D. Predicting web actions from html content. In *Proceedings of the thirteenth ACM conference on Hypertext and hypermedia* (2002), ACM, pp. 159–168.

[24] de Mattos, D. P. *An Approach for Authoring Mulsemedia Documents Based on Events.* Tese de Doutorado, Programa de Pós-graduação em Computação da Universidade Federal Fluminense, Setembro 2021.

[25] de Mattos, D. P.; Muchaluat-Saade, D. C.; Ghinea, G. An approach for authoring mulsemedia documents based on events. In *2020 International Conference on Computing, Networking and Communications (ICNC)* (2020), IEEE, pp. 273–277.

[26] Dos Santos, J. A.; Muchaluat-Saade, D. C.; Roisin, C.; Layaïda, N. A hybrid approach for spatio-temporal validation of declarative multimedia documents. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 14*, 4 (2018), 86.

[27] ET AL., D. F. Link prefetching faq, [online], 2014. Disponível em: https://developer.mozilla.org/en-US/docs/Web/HTTP/Link_prefetching_FAQ.

[28] ETSI. *ETSI TS 102 796 V1.4.1: Hybrid Broadcast Broadband TV*, 2016. European Telecommunications Standards Institute.

[29] EXPLORER, I. https://www.microsoft.com/pt-br/download/internet-explorer.aspx, 2018.

[30] FARIA, G.; HENRIKSSON, J. A.; STARE, E.; TALMOLA, P. Dvb-h: Digital broadcast services to handheld devices. *Proceedings of the IEEE 94*, 1 (2006), 194–209.

[31] FIGUEROA, A. M. Pré-busca de conteúdo em apresentações multimídia, Março 2014.

[32] FIREFOX, M. https://www.mozilla.org/pt-br/firefox, 2018.

[33] FLANAGAN, D. *JavaScript: the definitive guide.* "O'Reilly Media, Inc.", 2006.

[34] GHINEA, G.; TIMMERER, C.; LIN, W.; GULLIVER, S. R. Mulsemedia: State of the art, perspectives, and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 11* (2014), 17.

[35] GUEDES, Á. L. V.; AZEVEDO, R. G. D. A.; BARBOSA, S. D. J. Extending multimedia languages to support multimodal user interactions. *Multimedia Tools and Applications 76*, 4 (Feb 2017), 5691–5720.

[36] HBBTV. Hbbtv 2.0. 1 specification. *HbbTV Association, available in https://www. hbbtv. org/pages/about_hbbtv/HbbTV_specification_ 2_0. pdf 19* (2015).

[37] HUNKELER, U.; TRUONG, H. L.; STANFORD-CLARK, A. Mqtt-s—a publish/subscribe protocol for wireless sensor networks. In *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)* (2008), IEEE, pp. 791–798.

[38] HUNKELER, U.; TRUONG, H. L.; STANFORD-CLARK, A. Mqtt-s—a publish/subscribe protocol for wireless sensor networks. In *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)* (2008), IEEE, pp. 791–798.

[39] IERUSALIMSCHY, R. *Programming in lua*, 2nd ed. Roberto Ierusalimschy, March 2006.

[40] ISO/IEC. *Dynamic Adaptive Streaming Over HTTP (DASH). Part 1: Media Presentation Description and Segment Formats*, 2012. ISO/IEC 23009-1: 2012.

[41] ISO/IEC. *Information technology — Media context and control — Part 2: Control Information*, 2013. ISO/IEC 23005-2:2019.

[42] ISO/IEC. *Information technology — Media context and control — Part 6: Common types and tools*, 2019. ISO/IEC 23005-6:2019.

[43] ISO/IEC. *ISO/IEC 13818-1:2019 Information technology — Generic coding of moving pictures and associated audio information — Part 1: Systems*, june 2019.

[44] Information technology – Media context and control – Part 3: Sensory information, 2011.

[45] ITU-Recommendation. Nested context language (ncl) and ginga-ncl for iptv services.

[46] Josue, M.; Abreu, R.; Barreto, F.; Mattos, D. P.; Amorim, G.; Santos, J. A. F. D.; Muchaluat-Saade, D. C. Modeling sensory effects as first-class entities in multimedia applications. In *ACM Multimedia Systems Conference, 2018, Amsterdam. MMSys 2018* (2018), ACM, pp. 1–12.

[47] Josué, M.; Moreno, M.; Costa, R. An adaptable transmission management framework for push-mode hypermedia content. In *Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web* (2017), ACM, pp. 349–356.

[48] Josué, M.; Moreno, M.; Muchaluat-Saade, D. Mulsemedia preparation: a new event type for preparing media object presentation and sensory effect rendering. In *Proceedings of the 10th ACM Multimedia Systems Conference* (2019), ACM, pp. 110–120.

[49] Josué, M.; Muchaluat-Saade, D.; Moreno, M. Preparation of media object presentation and sensory effect rendering in mulsemedia applications. In *Proceedings of the 24th Brazilian Symposium on Multimedia and the Web* (2018), ACM, pp. 45–52.

[50] Josué, M.; Montevecchi, E.; Abreu, R.; Barreto, F.; Santos, J.; Muchaluat-Saade, D. C. *Livro de Minicursos do SBCAS 2020.* Sociedade Brasileira de Computação, 2020, ch. 2, pp. 48–90.

[51] Kim, J.; Lee, C.; Kim, Y.; Ryu, J. Construction of a haptic-enabled broadcasting system based on the mpeg-v standard. *Signal Processing: Image Communication 28*, 2 (2013), 151–161.

[52] Kim, M. Y.; Song, J. Multimedia documents with elastic time. In *Proceedings of the third ACM international conference on Multimedia* (1995), ACM, pp. 143–154.

[53] Luque, F. P.; Galloso, I.; Feijoo, C.; Martín, C. A.; Cisneros, G. Integration of multisensorial stimuli and multimodal interaction in a hybrid 3dtv system. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 11*, 1s (2014), 16.

[54] Meixner, B.; Einsiedler, C. Download and cache management for html5 hypervideo players. In *Proceedings of the 27th ACM Conference on Hypertext and Social Media* (2016), ACM, pp. 125–136.

[55] Murata, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE 77*, 4 (1989), 541–580.

[56] Murray, N.; Lee, B.; Qiao, Y.; Muntean, G.-M. Multiple-scent enhanced multimedia synchronization. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 11*, 1s (2014), 12.

[57] Nakamoto, T.; Yoshikawa, K. Movie with scents generated by olfactory display using solenoid valves. *IEICE transactions on fundamentals of electronics, communications and computer sciences 89*, 11 (2006), 3327–3332.

[58] Rodrigues, R. F.; Soares, L. F. G. A framework for prefetching mechanisms in hypermedia presentations. In *Fourth International Symposium on Multimedia Software Engineering, 2002. Proceedings.* (2002), IEEE, pp. 278–285.

[59] Rodriguez, A. Restful web services: The basics. *IBM developerWorks 33* (2008), 18.

[60] Rossum, G. V.; Jansen, J.; Mullender, K. S.; Bulterman, D. C. Cmifed: a presentation environment for portable hypermedia documents. In *Proceedings of the first ACM international conference on Multimedia* (1993), ACM, pp. 183–188.

[61] Saleme, E. B. *An Architectural Model and a Framework for Dealing with Variability in Mulsemedia Systems.* Tese de Doutorado, Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, Maio 2019.

[62] Saleme, E. B.; Santos, C. A.; Ghinea, G. Improving response time interval in networked event-based mulsemedia systems. In *Proceedings of the 9th ACM Multimedia Systems Conference* (2018), pp. 216–224.

[63] Saleme, E. B.; Santos, C. A.; Ghinea, G. A mulsemedia framework for delivering sensory effects to heterogeneous systems. *Multimedia Systems 25*, 4 (2019), 421–447.

[64] Signes, J.; Fisher, Y.; Eleftheriadis, A. Mpeg-4's binary format for scene description. *Signal Processing: Image Communication 15*, 4 (2000), 321–345.

[65] Soares, L. F. G.; Barbosa, S. D. Programando em ncl 3.0: Desenvolvimento de aplicações para o middleware ginga. *Campus, Rio de Janeiro, RJ* (2009).

[66] Soares, L. F. G.; Costa, R. M.; Moreno, M. F.; Moreno, M. F. Multiple exhibition devices in dtv systems. In *Proceedings of the 17th ACM international conference on Multimedia* (2009), ACM, pp. 281–290.

[67] Soares, L. F. G.; Rodrigues, R. F. Nested context model 3.0 part 1 - ncm core. Tech. rep., Informatics Department, PUC-Rio, Rio de Janeiro, May 2005.

[68] Soares, L. F. G.; Rodrigues, R. F.; Saade, D. C. M. Modeling, authoring and formatting hypermedia documents in the hyperprop system. *Multimedia Systems 8*, 2 (2000), 118–134.

[69] Su, Z.; Yang, Q.; Zhang, H.-J. A prediction system for multimedia pre-fetching in internet. In *Proceedings of the eighth ACM international conference on Multimedia* (2000), ACM, pp. 3–11.

[70] W3C. Synchronized multimedia integration language - smil 3.0 specification. http://www.w3c.org/TR/SMIL3, 2008. World-Wide Web Consortium Recommendation.

[71] W3C. Media source extensions. w3c recommendation 17 november 2016, 2016. https://www.w3.org/TR/media-source/.

[72] W3C. Html5, 2018. https://www.w3schools.com/.

[73] Waltl, M.; Rainer, B.; Timmerer, C.; Hellwagner, H. A toolset for the authoring, simulation, and rendering of sensory experiences. In *Proceedings of the 20th ACM international conference on Multimedia* (2012), ACM, pp. 1469–1472.

[74] Waltl, M.; Rainer, B.; Timmerer, C.; Hellwagner, H. An end-to-end tool chain for sensory experience based on mpeg-v. *Signal Processing: Image Communication 28*, 2 (2013), 136–150.

[75] Waltl, M.; Timmerer, C.; Hellwagner, H. Improving the quality of multimedia experience through sensory effects. In *Quality of Multimedia Experience (QoMEX), 2010 Second International Workshop on* (2010), IEEE, pp. 124–129.

[76] Waltl, M.; Timmerer, C.; Rainer, B.; Hellwagner, H. Sensory effects for ambient experiences in the world wide web. *Multimedia tools and applications 70*, 2 (2014), 1141–1160.

[77] Yoon, K. End-to-end framework for 4-d broadcasting based on mpeg-v standard. *Signal Processing: Image Communication 28*, 2 (2013), 127–135.

[78] Yuan, Z.; Bi, T.; Muntean, G.; Ghinea, G. Perceived synchronization of mulsemedia services. *IEEE Transactions on Multimedia 17*, 7 (2015), 957–966.

APÊNDICE A – Alteração em NCL 3.0 para especificação do evento de preparação

A linguagem NCL 3.0 permite a especificação de elos e conectores para definir o sincronismo entre os objetos de uma aplicação NCL. Os conectores definem relações sem especificar os elementos participantes, enquanto os elos definem relacionamentos referindo-se a um conector e definindo os participantes da relação. A linguagem permite a especificação de conectores causais, onde uma ou mais condições pode disparar uma ou mais ações. A fim de facilitar a autoria de aplicações, NCL 3.0 define um conjunto de valores pré-definidos para especificação de condições e de ações para serem utilizadas em conectores. Estes valores são especificados no módulo *ConnectorCausalExpression* da linguagem NCL 3.0, como mostra a Listagem A.1.

Visando possibilitar a especificação do evento de preparação em aplicações NCL, esta tese inseriu novos papéis de condição e ação na linguagem. Os novos papéis de condição são definidos no tipo *conditionRolePrototype* (linhas 23 a 25 da Listagem A.1). As ações relacionadas ao evento de preparação são definidas no tipo *actionNamePrototype* (linhas 67 a 69 da Listagem A.1).

NCL também permite construir condições que testam o estado de apresentação de um evento, atributos associados a eventos ou valores de propriedades. Essa característica é especificada pelo módulo *ConnectorAssessmentExpression* da linguagem NCL 3.0. Para verificar se um componente da aplicação está preparado ou não, o autor da aplicação NCL pode utilizar o atributo *prepared* proposto nesta tese. Este atributo foi inserido no tipo *attributePrototype* (linha 29 da Listagem A.1).

A.1 NCL30ConnectorCausalExpression.xsd

```

<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:connectorCausalExpression="http://www.ncl.org.br/NCL3.0/
    ConnectorCausalExpression"
     xmlns:connectorCommonPart="http://www.ncl.org.br/NCL3.0/
    ConnectorCommonPart"
  targetNamespace="http://www.ncl.org.br/NCL3.0/ConnectorCausalExpression
    "
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

<!-- import the definitions in the modules namespaces -->
<import namespace="http://www.ncl.org.br/NCL3.0/ConnectorCommonPart"
     schemaLocation="http://www.ncl.org.br/NCL3.0/modules/
    NCL30ConnectorCommonPart.xsd"/>

<simpleType name="conditionRoleUnion">
  <union memberTypes="string connectorCausalExpression:
    conditionRolePrototype"/>
</simpleType>

<simpleType name="conditionRolePrototype">
  <restriction base="string">
    <enumeration value="onBegin" />
    <enumeration value="onEnd" />
    <enumeration value="onPause" />
    <enumeration value="onResume" />
    <enumeration value="onAbort" />
    <enumeration value="onBeginPreparation" />
    <enumeration value="onEndPreparation" />
    <enumeration value="onAbortPreparation" />
  </restriction>
</simpleType>

<simpleType name="maxUnion">
```

```
30   <union memberTypes="positiveInteger connectorCausalExpression:
      unboundedString"/>
31 </simpleType>
32
33 <simpleType name="unboundedString">
34   <restriction base="string">
35     <pattern value="unbounded"/>
36   </restriction>
37 </simpleType>
38
39 <complexType name="simpleConditionPrototype">
40   <attribute name="role" type="connectorCausalExpression:
      conditionRoleUnion" use="required"/>
41   <attribute name="eventType" type="connectorCommonPart:eventPrototype"
      use="optional"/>
42   <attribute name="key" type="string" use="optional"/>
43   <attribute name="transition" type="connectorCommonPart:
      transitionPrototype" use="optional"/>
44   <attribute name="delay" type="string" use="optional"/>
45   <attribute name="min" type="positiveInteger" use="optional"/>
46   <attribute name="max" type="connectorCausalExpression:maxUnion" use="
      optional"/>
47   <attribute name="qualifier" type="connectorCommonPart:
      logicalOperatorPrototype" use="optional"/>
48 </complexType>
49
50 <complexType name="compoundConditionPrototype">
51   <attribute name="operator" type="connectorCommonPart:
      logicalOperatorPrototype" use="required"/>
52   <attribute name="delay" type="string" use="optional"/>
53 </complexType>
54
55 <simpleType name="actionRoleUnion">
56   <union memberTypes="string connectorCausalExpression:
      actionNamePrototype"/>
```

```
57 </simpleType>
58
59 <simpleType name="actionNamePrototype">
60   <restriction base="string">
61     <enumeration value="start" />
62     <enumeration value="stop" />
63     <enumeration value="pause" />
64     <enumeration value="resume" />
65     <enumeration value="abort" />
66     <enumeration value="set" />
67     <enumeration value="startPreparation" />
68     <enumeration value="stopPreparation" />
69     <enumeration value="abortPreparation" />
70   </restriction>
71 </simpleType>
72
73 <simpleType name="actionOperatorPrototype">
74   <restriction base="string">
75     <enumeration value="par" />
76     <enumeration value="seq" />
77   </restriction>
78 </simpleType>
79
80 <complexType name="simpleActionPrototype">
81   <attribute name="role" type="connectorCausalExpression:actionRoleUnion"
       use="required"/>
82   <attribute name="eventType" type="connectorCommonPart:eventPrototype"
      use="optional"/>
83   <attribute name="actionType" type="connectorCausalExpression:
      actionNamePrototype" use="optional"/>
84   <attribute name="delay" type="string" use="optional"/>
85   <attribute name="value" type="string" use="optional"/>
86   <attribute name="repeat" type="positiveInteger" use="optional"/>
87   <attribute name="repeatDelay" type="string" use="optional"/>
88   <attribute name="min" type="positiveInteger" use="optional"/>
```

```
89   <attribute name="max" type="connectorCausalExpression:maxUnion" use="
      optional"/>
90   <attribute name="qualifier" type="connectorCausalExpression:
      actionOperatorPrototype" use="optional"/>
91 </complexType>
92
93 <complexType name="compoundActionPrototype">
94   <choice minOccurs="2" maxOccurs="unbounded">
95     <element ref="connectorCausalExpression:simpleAction" />
96     <element ref="connectorCausalExpression:compoundAction" />
97   </choice>
98   <attribute name="operator" type="connectorCausalExpression:
      actionOperatorPrototype" use="required"/>
99   <attribute name="delay" type="string" use="optional"/>
100 </complexType>
101
102   <!-- declare global elements in this module -->
103 <element name="simpleCondition" type="connectorCausalExpression:
      simpleConditionPrototype" />
104 <element name="compoundCondition" type="connectorCausalExpression:
      compoundConditionPrototype" />
105 <element name="simpleAction" type="connectorCausalExpression:
      simpleActionPrototype" />
106 <element name="compoundAction" type="connectorCausalExpression:
      compoundActionPrototype" />
107
108 </schema>
```

Listagem A.1: Esquema do conector causal contendo os novos papéis propostos nesta tese

A.2 NCL30ConnectorAssessmentExpression.xsd

```
1
2 <schema xmlns="http://www.w3.org/2001/XMLSchema"
3   xmlns:connectorAssessmentExpression="http://www.ncl.org.br/NCL3.0/
      ConnectorAssessmentExpression"
```

```
4   xmlns:connectorCommonPart="http://www.ncl.org.br/NCL3.0/
     ConnectorCommonPart"
5   targetNamespace="http://www.ncl.org.br/NCL3.0/
     ConnectorAssessmentExpression"
6   elementFormDefault="qualified" attributeFormDefault="unqualified" >
7
8 <!-- import the definitions in the modules namespaces -->
9 <import namespace="http://www.ncl.org.br/NCL3.0/ConnectorCommonPart"
10     schemaLocation="http://www.ncl.org.br/NCL3.0/modules/
     NCL30ConnectorCommonPart.xsd"/>
11
12 <simpleType name="comparatorPrototype">
13   <restriction base="string">
14     <enumeration value="eq" />
15     <enumeration value="ne" />
16     <enumeration value="gt" />
17     <enumeration value="lt" />
18     <enumeration value="gte" />
19     <enumeration value="lte" />
20   </restriction>
21 </simpleType>
22
23 <simpleType name="attributePrototype">
24   <restriction base="string">
25     <enumeration value="repeat" />
26     <enumeration value="occurrences" />
27     <enumeration value="state" />
28     <enumeration value="nodeProperty" />
29     <enumeration value="prepared" />
30   </restriction>
31 </simpleType>
32
33 <simpleType name="statePrototype">
34   <restriction base="string">
35     <enumeration value="sleeping" />
```

```
36      <enumeration value="occurring" />
37      <enumeration value="paused" />
38    </restriction>
39  </simpleType>
40
41  <simpleType name="valueUnion">
42    <union memberTypes="string connectorAssessmentExpression:statePrototype
      "/>
43  </simpleType>
44
45  <complexType name="assessmentStatementPrototype" >
46    <sequence>
47      <element ref="connectorAssessmentExpression:attributeAssessment"/>
48      <choice>
49        <element ref="connectorAssessmentExpression:attributeAssessment"/>
50        <element ref="connectorAssessmentExpression:valueAssessment"/>
51      </choice>
52    </sequence>
53    <attribute name="comparator" type="connectorAssessmentExpression:
      comparatorPrototype" use="required"/>
54  </complexType>
55
56  <complexType name="attributeAssessmentPrototype">
57    <attribute name="role" type="string" use="required"/>
58    <attribute name="eventType" type="connectorCommonPart:eventPrototype"
      use="required"/>
59    <attribute name="key" type="string" use="optional"/>
60    <attribute name="attributeType" type="connectorAssessmentExpression:
      attributePrototype" use="optional"/>
61    <attribute name="offset" type="string" use="optional"/>
62  </complexType>
63
64  <complexType name="valueAssessmentPrototype">
65    <attribute name="value" type="connectorAssessmentExpression:valueUnion"
        use="required"/>
```

```
66 </complexType>
67
68 <complexType name="compoundStatementPrototype">
69   <choice minOccurs="1" maxOccurs="unbounded">
70     <element ref="connectorAssessmentExpression:assessmentStatement" />
71     <element ref="connectorAssessmentExpression:compoundStatement" />
72   </choice>
73   <attribute name="operator" type="connectorCommonPart:
     logicalOperatorPrototype" use="required"/>
74   <attribute name="isNegated" type="boolean" use="optional"/>
75 </complexType>
76
77   <!-- declare global elements in this module -->
78 <element name="assessmentStatement" type="connectorAssessmentExpression:
     assessmentStatementPrototype" />
79 <element name="attributeAssessment" type="connectorAssessmentExpression:
     attributeAssessmentPrototype" />
80 <element name="valueAssessment" type="connectorAssessmentExpression:
     valueAssessmentPrototype" />
81 <element name="compoundStatement" type="connectorAssessmentExpression:
     compoundStatementPrototype" />
82
83 </schema>
```

Listagem A.2: Esquema do *ConnectorAssessment* contendo o atributo *prepared* proposto nesta tese

APÊNDICE B – Esquema NCL 4.0

B.1 NCL40Effect.xsd

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:effect="http://www.ncl.org.br/NCL4.0/Effect"
  targetNamespace="http://www.ncl.org.br/NCL4.0/Effect"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

 <complexType name="effectPrototype">
    <attribute name="id" type="ID" use="required"/>
    <attribute name="type" type="string" use="required"/>
 </complexType>

  <!-- declare global elements in this module -->
  <element name="effect" type="effect:effectPrototype"/>

</schema>
```

Listagem B.1: Esquema do módulo Effect

B.2 Alteração no NCL30Descriptor.xsd

```

<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:descriptor="http://www.ncl.org.br/NCL3.0/Descriptor"
  targetNamespace="http://www.ncl.org.br/NCL3.0/Descriptor"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

```

```
7   <complexType name="descriptorParamPrototype">
8     <attribute name="name" type="string" use="required" />
9     <attribute name="value" type="string" use="required"/>
10  </complexType>
11
12  <complexType name="descriptorPrototype">
13    <sequence minOccurs="0" maxOccurs="unbounded">
14      <element ref="descriptor:descriptorParam"/>
15    </sequence>
16    <attribute name="id" type="ID" use="required"/>
17    <attribute name="player" type="string" use="optional"/>
18    <attribute name="activate" type="boolean" use="optional"/>
19    <attribute name="duration" type="string" use="optional"/>
20    <attribute name="fade" type="string" use="optional"/>
21    <attribute name="intensityValue" type="string" use="optional"/>
22    <attribute name="intensityRange" type="string" use="optional"/>
23    <attribute name="scent" type="string" use="optional"/>
24    <attribute name="color" type="string" use="optional"/>
25  </complexType>
26
27 <!--
28 Formatters should support the following descriptorParam names.
29 * For audio players: soundLevel; balanceLevel; trebleLevel; bassLevel.
30 * For text players: style, which refers to a style sheet with information
       for text
31 presentation; textAlign; fontColor; FontFamily; fontStyle; fontSize;
       fontVariant;
32 fontWeight.
33 * For visual media (any NCL media object, represented by a <media>
       element, whose
34 content produces a visual presentation when the object is started)
       players : background,
35 specifying the background color used to fill the area of a region
       displaying media;
```

```
36 scroll, which allows the specification of how an author would like to
    configure the
37 scroll in a region; fit, indicating how an object will be presented (
    hidden, fill, meet,
38 meetBest, slice); transparency, indicating the degree of transparency of
    an object
39 presentation (the value shall be between 0 and 1, or a real value in the
    range [0,100]
40 ending with the character "%" (e.g. 30%)); visible, indicating if the
    presentation is to
41 be seen or hidden; rgbChromakey; the object positioning parameters: top,
    left, bottom,
42 right, width, height, zIndex, plan, location, size and bounds; the focus
    movement
43 parameters: moveLeft, moveRight, moveUp, moveDown, focusIndex; the other
    related focus
44 parameters: focusBorderColor, selBorderColor, focusBorderWidth,
    focusBorderTransparency,
45 focusSrc, and focusSelSrc; the transition parameters: transIn and
    transOut; the timing
46 parameters: explicitDur and freeze; and the multiple device parameters:
    baseDeviceRegion
47 and deviceClass.
48 * For players in general: player; reusePlayer, which determines if a new
    player shall be
49 instantiated or if a player already instantiated shall be used; and
    playerLife, which
50 specifies what will happen to the player instance at the end of the
    presentation.
51 -->
52
53   <complexType name="descriptorBasePrototype">
54     <attribute name="id" type="ID" use="optional"/>
55   </complexType>
56
```

```
57  <!-- declare global elements in this module -->
58  <element name="descriptorParam" type="descriptor:
     descriptorParamPrototype"/>
59  <element name="descriptor" type="descriptor:descriptorPrototype"/>
60  <element name="descriptorBase" type="descriptor:descriptorBasePrototype
     "/>
61
62  <!-- declare global attributes in this module -->
63  <attributeGroup name="descriptorAttrs">
64    <attribute name="descriptor" type="string" use="optional"/>
65  </attributeGroup>
66
67 </schema>
```

Listagem B.2: Esquema do descritor contendo as novas propriedades propostas nesta tese para efeitos sensoriais.