UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Marco Aurélio Freesz Júnior

STorM: Um Modelo de Autoria Hipermídia para *Out-of-Home Media*

Juiz de Fora

2017

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Marco Aurélio Freesz Júnior

STorM: Um Modelo de Autoria Hipermídia para *Out-of-Home Media*

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: Marcelo Ferreira Moreno

Juiz de Fora

2017

Marco Aurélio Freesz Júnior

"STorM: Um Modelo de Autoria Hipermídia para Out-of-Home Media"

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Mestre.

Aprovada em 25 de agosto de 2017.

BANCA EXAMINADORA

Prof. Dr. Marcelo Ferreira Moreno – Orientador
Universidade Federal de Juiz de Fora

Prof. Dr. Eduardo Barrére
Universidade Federal de Juiz de Fora

Prof. Dr. Carlos de Salles Soares Neto
Universidade Federal do Maranhão

Prof. Dr. Guilherme Augusto Ferreira Lima
Pontifícia Universidade Católica do Rio de Janeiro

Ficha catalográfica elaborada através do programa de geração
automática da Biblioteca Universitária da UFJF,
com os dados fornecidos pelo(a) autor(a)

Freesz Júnior, Marco Aurélio.
STorM: Um Modelo de Autoria Hipermídia para Out-of-Home Media / Marco Aurélio Freesz Júnior. -- 2017.
134 p.

Orientador: Marcelo Ferreira Moreno
Dissertação (mestrado acadêmico) - Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós Graduação em Ciência da Computação, 2017.

1. Representação Hipermídia. 2. Autoria de Conteúdo Hipermídia. 3. Modelo e Linguagem de Autoria Hipermídia. 4. Digital Out-of Home media. I. Moreno, Marcelo Ferreira, orient. II. Título.

Aos meus Pais.

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus por iluminar meu caminho ao longo de toda minha trajetória, além de presentear, a mim e à minha namorada, com um filho, Arthur, que trouxe muita luz e alegria nos momentos em que mais precisei.

Agradeço aos meus familiares, principalmente meus pais, Marco Aurélio e Maria José, por ensinarem os valores da vida e pelo apoio dado durante todo esse período. Agradeço à minha namorada Ana Paula pela paciência e suporte nos momentos difíceis.

Agradeço a todos os professores do Programa de Pós-Graduação em Ciência da Computação pelos ensinamentos transformadores. Em especial agradeço aos professores Eduardo Barrére e Alex Vieira por acreditarem em mim ao fazerem a recomendação para o programa e ao meu orientador Marcelo Moreno pela enorme paciência e dedicação.

Agradeço a todos os colegas do Mestrado e Graduação pela parceria e ajuda. Em especial, Laura Lima, Bruno Marques, Thiago Moratori e Ludmila Yung, pelo apoio e contribuição nessa jornada.

Agradeço ainda ao Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais pelo incentivo à minha qualificação. Agradeço também a todos os meus colegas de trabalho do Instituto Federal que me apoiaram e me ajudaram enquanto estive afastado para concluir esse trabalho.

"A mente que se abre a uma
nova ideia jamais voltará ao seu
tamanho original"
Albert Einstein

RESUMO

Em meio à diversidade de veículos de comunicação atuais, as mídias *Out-of-Home* vêm desempenhando um papel de destaque tanto em espaços públicos quanto privados, ao permitir a difusão de informações de cunho coletivo a um grande número de pessoas. No que diz respeito à autoria de conteúdos multimídia nesse meio, grande parte do ferramental existente ou faz uso de linguagens de domínios distintos, ou faz uso de abstrações gráficas que facilitam a autoria, porém, utilizam-se de linguagens para representação hiper/multimídia com conceitos distantes aos utilizados nessas abstrações. Portanto, a representação desses documentos se torna complexa, fazendo uso ainda de linguagens de *scripts* que são ilegíveis aos autores não-programadores e de difícil extração automatizada de informações. Com o foco nessa etapa, este trabalho elabora um modelo conceitual hipermídia com entidades de alto nível de abstração que busca se aproximar dos conceitos encontrados na indústria audiovisual, tais como cenas, trilhas e mídias. A partir desse modelo é criada uma linguagem declarativa em XML, utilizada para representar o conteúdo multimídia e, como consequência, permitir sua autoria textual. Essa linguagem procura tratar questões relativas às disposições espaciais, sincronização dos objetos de mídias, adaptação de conteúdos, além da interatividade e algumas de suas múltiplas modalidades de interação, como gestos e voz. Como resultado, alguns casos de uso são tratados por meio da criação de conteúdos DOOH utilizando-se da linguagem multimídia proposta e, em seguida, uma execução em um *player* de prova de conceito é realizada. Além disso, o trabalho propõe um pré-processamento específico, aplicado a cada instância do modelo, com o objetivo de otimizar a transmissão do conteúdo em arquiteturas *broadcast* e *multicast*, como IPTV. Esse pré-processamento aborda a renderização sucessiva de partes determinísticas da instância, de modo a facilitar o sincronismo do conteúdo entre terminais, reduzir a carga de processamento nestes e reduzir o uso de recursos da rede.

Palavras-chave: Representação Hipermídia. Autoria de Conteúdo Hipermídia. Modelo e Linguagem de Autoria. Interatividade Multimodal. Mídia de Sinalização Digital.

ABSTRACT

Among the variety of current social communication vehicles, Digital Out-of Home (DOOH) media have been playing a remarkable role in both public and private spaces, by allowing for the rapid dissemination of collective information to a large number of people. Regarding multimedia content authoring for this vehicle, much of the existing toolkits either makes use of languages from other domains or makes use of graphic abstractions that ease the authoring process, but making use of hyper/multimedia representation languages with distant concepts to those used in these abstractions. Therefore, document representation becomes complex, sometimes makes use of scripting languages, and therefore is illegible by authors and even difficult for automated information extraction. Focusing on this step, this paper develops a conceptual hypermedia model with high-level entities of abstraction which seek to approach the concepts found in the audiovisual industry, such as scenes, tracks and media. From that model a XML declarative language is created, used to represent the multimedia content and, as a result, allow its textual authorship. This language seeks to deal with matters relating to spatial arrangements, synchronizing media objects, content adaptation, in addition to the interactivity and some of its multiple modalities of interaction, gesture and voice. As a result, some use cases are handled by creating DOOH content using the proposed multimedia language, and then an execution in a proof-of-concept player is performed. In addition, this paper proposes an improvement applied to the STorML instance with the aim to optimize the transmission of content in *broadcast* and *multicast* architectures, such as IPTV. This improvement covers subsequent rendering of deterministic parts of the instance to facilitate content timing between terminals, reducing the processing in the terminals and decreasing the use of network resources.

Keywords: Hypermedia Representations. Hypermedia Content Authoring. Authoring Model and Language. Multimodal Interaction. Digital Out-of-Home media.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API Application Programming Interface

BIFS Binary Format for Scenes

CSS3 Cascading Style Sheets 3

DOOH Digital Out-Of-Home

DS Digital Signage

DSL Domain Specific Language

DTD Document Type Definition

FFmpeg Fast Forward Moving Pictures Expert Group

GUI Graphical User Interface

HTG Hypermedia Temporal Graph

HTML Hypertext Markup Language

HTML5 Hypertext Markup Language version 5

IPTV Internet Protocol Television

ITU-T Telecommunication Standardization Sector

JSON JavaScript Object Notation

MUI Multimodal User Interface

NCL Nested Context Language

NCM Nested Context Model

OA Objeto de Aprendizagem

QoS Quality of Service

QR Quick Response

RNP Rede Nacional de Ensino e Pesquisa

RSS Rich Site Summary

SMIL Synchronized Multimedia Integration Language

SSM SceneSync Model

STorM Scene- and Track-oriented hypermedia Model

SRT SubRip Text

TAL Template Authoring Language

TTML Timed Text Markup Language

UIT União Internacional de Telecomunicações

URI Uniform Resource Identifier

URL Uniform Resource Locator

VOP Video Object Plane

XML eXtensible Markup Language

XMT Extensible MPEG-4 Textual Format

XSD Xml Schema Definition

W3C World Wide Web Consortium

WebVTT Web Video Text Track

SUMÁRIO

1 INTRODUÇÃO

Nos dias de hoje é comum encontrar mídias digitais nos mais diversos locais, como por exemplo, estações de transportes públicos, centros comerciais, ambientes corporativistas ou até mesmo em ruas de grandes centros de algumas cidades. Tais mídias de exibição são de grande utilidade na difusão de informação (publicidade, avisos, informações, etc.) e são comumente conhecidas como mídias de Sinalização Digital (*Digital Signage*) (DUPIN, 2011) ou mídias do tipo *Digital Out-of-Home* (DOOH).

Estudos mostram que o crescimento desses tipos de mídias é significativo (DUPIN, 2011) . O mercado global de tecnologias voltadas ao uso de DOOH cresceu de forma surpreendente nos últimos 5 anos. A estimativa mundial é de que em 2016 se gastou 4,5 bilhões de dólares em toda a infraestrutura responsável pela disseminação do conteúdo, desde a etapa de criação e exibição do conteúdo nos terminais, até o investimento em *players* e custos de manutenção. Em 2017, a previsão é que o valor chegue a 13,8 bilhões de dólares. Todo esse crescimento está diretamente relacionado ao fácil acesso aos equipamentos eletrônicos envolvidos, assim como os baixos custos comparados aos anos anteriores (DUPIN, 2011) .

Com base nesse uso cada vez mais frequente, avanços nos métodos de produção, distribuição e consumo de conteúdo DOOH se tornam cada vez mais importantes no cenário global. No entanto, várias soluções atuais para DOOH mostram-se excessivamente restritivas, ao tratar a autoria de conteúdo como uma simples criação de *playlists* de objetos de mídia. Nesses casos, os conteúdos usualmente veiculados acabam se limitando a uma apresentação de objetos de mídia predominantemente discreta, ainda que com suporte a vídeo/animações, mas como uma espécie de *slideshow* (DUPIN, 2011) .

Em outras soluções de autoria mais elaboradas, linguagens de domínio específico (*Domain-specific language* - DSL) voltadas para a representação de documentos hiper/-multimídia são oferecidas (A-SMIL, 2010) . Algumas destas linguagens, como HTML (aliada a *JavaScript*), agregam maior expressividade ao conteúdo, porém se mostram inadequadas para o perfil dos profissionais usualmente responsáveis pela autoria. Por isso, ferramentas de autoria gráfica são fundamentais neste contexto (4YOUSEE, 2014) . De fato, são notáveis as facilidades atuais oferecidas por essas ferramentas, principalmente

para a criação de conteúdo audiovisual e para o suporte à incorporação de diversos tipos de objetos de mídia nesse conteúdo. Porém, observa-se que existe uma grande distância entre as abstrações gráficas oferecidas por tais ferramentas de autoria em DOOH e a respectiva linguagem de representação dos documentos hiper/multimídia. A representação acaba por se tornar complexa, fazendo uso de abstrações não relacionadas ao domínio DOOH e até mesmo de linguagens baseadas em *script* de propósito mais geral. Dessa forma, a representação dos documentos vem a ser ilegível por seus criadores e até mesmo de difícil extração de informação e semântica, o que seria fundamental para a definição de *workflows* abrangentes que tratassem de maneira fim-a-fim o problema da produção e distribuição de mídias de DOOH.

Além disso, futuras modalidades de serviços DOOH devem não somente considerar os aspectos básicos da exibição de um conteúdo como uma sequência de objetos de mídia sincronizados, mas também os aspectos avançados que permitam o oferecimento de múltiplas modalidades de interação, leiautes cientes de arranjos de telas em *videowalls*, integração com dispositivos móveis dos usuários e adaptabilidade do conteúdo ao contexto da audiência. Porém, tais aspectos avançados encontram-se ainda incipientes nas ferramentas e sistemas DOOH. Nesse sentido, torna-se necessário buscar um modelo hipermídia especificamente voltado a DOOH, que não somente auxilie a autoria, mas também colabore com a difusão do conteúdo e com experiências engajadoras nos terminais.

1.1 PROPOSTA DA DISSERTAÇÃO

Como forma de contribuir com a minimização destes entraves citados, este trabalho propõe um modelo conceitual denominado STorM (**S**cene- and **T**rack-**or**iented hypermedia **M**odel), que busca reduzir a distância entre os conceitos utilizados nas abstrações gráficas de autoria e os elementos utilizados na representação de documentos hipermídia. Para isso, STorM e sua respectiva linguagem de representação, denominada STorML (STorM **L**anguage), apresentam entidades de mais alto nível, com conceitos próximos aos existentes no contexto da indústria do audiovisual, como cenas e trilhas, de modo a respeitar o perfil dos criadores de conteúdo. Além disso, STorM inclui suporte a formas avançadas de interação e de personalização do conteúdo, possibilitando seu uso na autoria textual de apresentações para DOOH.

O trabalho também aponta casos de uso do modelo, apresentando algumas instâncias

possíveis na utilização da linguagem e suas respectivas exibições em um *player* próprio simplificado como prova de conceito. Pensando na difusão do conteúdo, este trabalho demonstra uma maneira de se otimizar um conteúdo criado, buscando características determinísticas na instância do modelo, a fim de que se realize uma renderização parcial ou total do conteúdo em uma etapa prévia à sua transmissão.

1.2 ORGANIZAÇÃO DA DISSERTAÇÃO

A dissertação está organizada da seguinte forma: No Capítulo 2 é fornecido uma breve conceituação de mídias *Digital Out-of-Home* e, além disso, algumas ferramentas gráficas utilizadas na autoria de conteúdos DOOH são mostradas. Em seguida, alguns requisitos são levantados. O Capítulo 3 discute algumas linguagens de autoria de conteúdos para o domínio DOOH e também inclui linguagens de outros domínios que fazem uso de alguns conceitos semelhantes aos definidos nesta dissertação. O Capítulo 4 trata a definição do modelo conceitual STorM e descreve de forma detalhada uma linguagem de especificação de documentos baseada nesse modelo, denominada STorML. No Capítulo 5 um *player* STorML protótipo desenvolvido em *Java* é mostrado como prova de conceito. No Capítulo 6 são apresentados casos de uso da linguagem STorML aplicados a TV Corporativa (caso específico de DOOH), assim como a simulação de sua execução no *player* criado. Além disso, o capítulo aborda uma proposta de pré-processamento de instâncias STorM com o objetivo de prover certas otimizações na transmissão do conteúdo. Finalmente, o Capítulo 7 é dedicado às considerações finais do estudo e indicações de trabalhos futuros.

2 FUNDAMENTOS EM DOOH

Neste capítulo, são apresentados alguns fundamentos presentes em DOOH, assim como os conceitos envolvidos nessa tecnologia, a fim de melhor posicionar o leitor nas discussões ao longo do trabalho desenvolvido nesta dissertação.

2.1 CONCEITOS BÁSICOS

Com a evolução dos recursos tecnológicos, nos últimos anos, as agências de comunicação começaram a fazer a divulgação de diversos conteúdos em diversos lugares com a utilização de televisores, como em shoppings, rodovias, edifícios, dentre outros pontos de grande circulação de pessoas, se tornando meios importantes na aproximação de clientes e divulgação acelerada da informação. Além disso, essas mídias de apresentação são importantes maneiras de se economizar com quantidades excessivas de materiais impressos e, consequentemente, ajudar em questões ambientais no que diz respeito à sustentabilidade.

Diante desse cenário, os serviços de Sinalização Digital (*Digital Signage*) ganham força. De acordo com (DUPIN, 2011) , Sinalização Digital é uma rede de *displays* digitais que são gerenciados de forma centralizada e destinados à divulgação de informações relacionadas a entretenimento, *merchandising* e publicidade, em locais específicos. TV Corporativa se baseia no mesmo conceito, no entanto seu foco é especializado em ambientes menores e com um maior controle de gerenciamento, como empresas, Universidades e corporações em geral. Além do ambiente diferenciado, o público também é mais restrito, tendo como foco a comunicação e interação com funcionários, estudantes, fornecedores ou clientes. Vale ressaltar que alguns estudos, como em Want e Schilit (2012), tratam essas tecnologias como correspondentes e preferem não realizar diferenciações, de modo a categorizá-las como Sinalização Digital.

No presente trabalho, opta-se por utilizar o termo *Digital Out of Home Media* (mídias DOOH) (STALDER, 2011) , de significado equivalente a Sinalização Digital. A escolha se dá pela percepção de que as palavras de tal termo são mais abrangentes e talvez mais adequadas à potencial evolução de tais serviços, que mesmo atualmente já não mais simples ”sinalizações”.

Portanto, DOOH se define, hoje, não só como simples *displays* exibidores de conteúdo,

mas como uma arquitetura detalhada que aborda todo um fluxo na difusão da informação, desde a etapa de autoria do conteúdo até a sua exibição no receptor. Pensar em um modelo hipermídia que busca facilitar sua criação no lado do provedor, assim como definir formas de interação no lado do terminal, são requisitos essenciais.

Figura 2.1: Serviço Interativo em DOOH - Localização em espaço público.[1]

2.2 ARQUITETURA

Conforme citado nas seções anteriores a respeito da evolução dos sistemas de DOOH, muitos trabalhos vêm sendo desenvolvidos a fim de definir arquiteturas que demonstram os fluxos de trabalho (*workflows*) possíveis, desde a criação do conteúdo até a visualização nos terminais, como em Dayarathna, Withana e Sugiura (2011) e Freitas et al. (2013). No entanto, poucos abordam em detalhes a etapa de autoria e em grande parte das vezes tratam o conteúdo como um genérico *playlist* de mídias que serão entregues por meio de conexões *unicast* em modo *pull* e interpretados no terminal por um *player* específico.

Dentre as arquiteturas existentes, vale destacar os esforços que estão sendo feitos pela União Internacional de Telecomunicações (UIT). A recomendação UIT-T H.781 (ITU, 2015b) propõe, como pode ser visto na Figura 2.2, uma arquitetura que define as funções e suas entidades funcionais que compõem um sistema de Sinalização Digital. Observa-se

[1] <https://www.itu.int/dms_pub/itu-t/opb/tut/T-TUT-DS-2014-UCIS-PDF-E.pdf>

na figura que a criação de conteúdo está fora do escopo da recomendação.

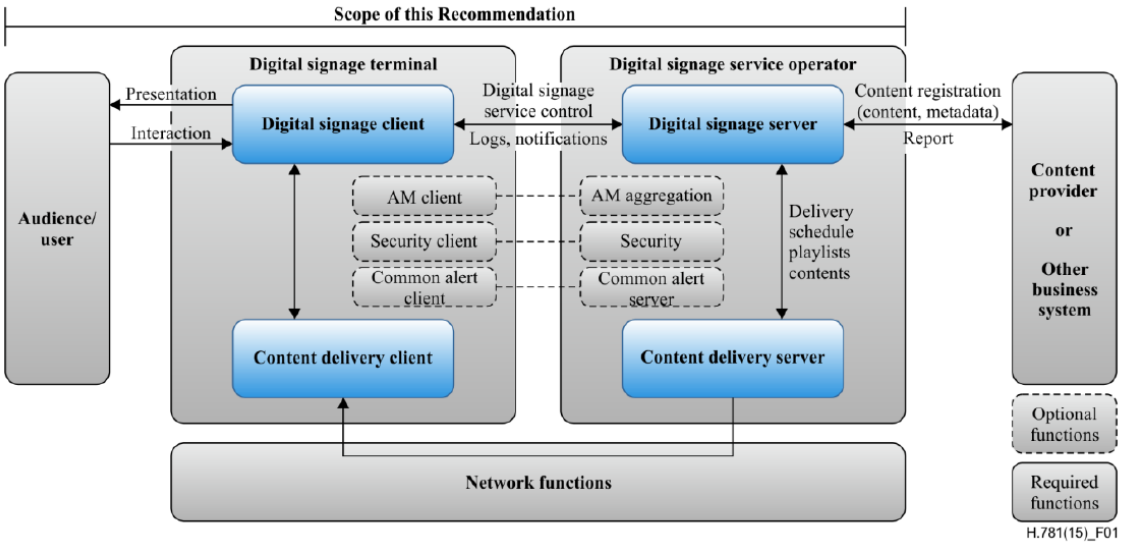


Figura 2.2: Visão Geral da Arquitetura para Sinalização Digital definida pela UIT.[2]

A arquitetura definida pela recomendação se baseia nas seguintes entidades globais:

- *Audience/User* - Usuários que interagem com o terminal, seja visualizando um conteúdo, seja através de aplicações;
- *DS Terminal* - Dispositivo que exibe o conteúdo recebido do provedor de serviço;
- *DS service operator* - Operadora que provê os serviços de Sinalização Digital. Gerencia os dispositivos terminais para que exibam o conteúdo recebido do provedor de conteúdo;
- *Content provider* - Entidade responsável por prover o conteúdo de acordo com seus interesses particulares, como anúncios, informações e alertas.

A partir dessa cadeia de valor, a recomendação detalha todas as entidades funcionais presentes, sendo que as funções de *DS Client*, *Content Delivery Client*, *DS Server* e *Content Delivery Server*, destacadas em azul na Figura 2.2, são tratadas como pilares de todo o sistema. O *DS Client* é responsável pela apresentação do conteúdo e interação com o público. O *Content Delivery Client* é responsável por receber o conteúdo através da rede. Já no lado do servidor, o *DS Server* é responsável por administrar todo o sistema, controlar a entrega de conteúdo e gerenciar os terminais. O *Content Delivery Server* é responsável pela entrega do conteúdo ao cliente (*Content Delivery Client*).

[2]<http://www.itu.int/itu-t/recommendations/rec.aspx?rec=12466ITU-T>

2.3 FERRAMENTAS DE AUTORIA

Atualmente, grande parte dos conteúdos para DOOH são criados por meio de ferramentas gráficas *Web*, algumas abertas, outras proprietárias (SAMSUNG, 1999) . Essas aplicações tratam toda a questão de leiaute da apresentação, como também a composição dos elementos de mídias de representação (e.g. imagem, vídeo, etc) e seus comportamentos. Baseado nisso, alguns padrões ou linguagens estão sendo utilizados para representar, persistir e intercambiar essas informações, como o SMIL (W3C, 2008) , JSON, XML e, principalmente, o HTML5 associado a *JavaScript* (CARNEIRO, 2013) .

No entanto, a maioria das ferramentas são destinadas à autoria de conteúdos meramente informativos, estáticos e destinados à publicidade. Portanto, a interatividade ainda é pouco explorada, dependendo, nesses casos, do conhecimento por parte do criador de conteúdos nas linguagens suportadas pelo sistema DOOH.

Com o propósito de buscar um modelo hipermídia que atenda o maior número possível de requisitos e que esteja alinhado às formas de como o conteúdo é organizado nas ferramentas gráficas atuais, algumas dessas ferramentas de autoria foram estudadas.

2.3.1 XIBO

Xibo (XIBO, 2010) é uma solução de DOOH fornecida de forma gratuita e *open source*. Como características principais, o sistema fornece toda a gestão do conteúdo por meio de uma interface *Web*, a qual permite um vasto suporte a mídias de representação e controle de programação. A Figura 2.3 mostra como é o preenchimento de uma região com mídias por meio de uma *timeline*.

Xibo se baseia em uma arquitetura cliente-servidor. O servidor é uma aplicação PHP/MySQL responsável por gerir o conteúdo e disponibilizar aos terminais. Estes fazem a recuperação do conteúdo em intervalos de tempo definidos, por meio de conexões *unicast*. De forma comparativa ao modelo proposto nesta dissertação, Xibo faz uso de um XML Schema próprio, responsável pela representação de como o conteúdo deve ser exibido nos terminais. O documento XML engloba a definição do leiaute, conteúdo e disposição temporal. Na Figura 2.4 pode-se visualizar como esse documento XML é estruturado.

De acordo com XIBO (2010), o documento é dividido entre estrutura e a mídia de representação. A estrutura, representada pela Figura 2.4, limita a definição de um único

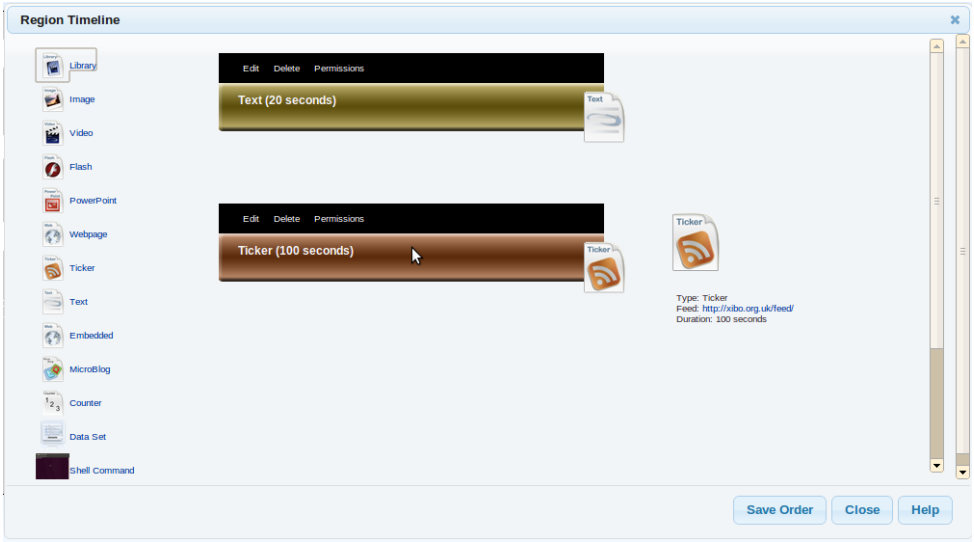


Figura 2.3: Preenchimento de *Timeline* em XIBO.

leiaute, que pode ser composto por diversas regiões. Em cada região poderá haver uma ou mais mídias, que serão exibidas de forma sequencial. Além das mídias, cada região apresenta um conjunto de opções que podem ser configuradas, como o número de repetições ou os tipos de transições.

A Figura 2.5 ilustra como os elementos de mídias de representação são organizados no documento XML. Dentre os atributos presentes, destaca-se o atributo responsável por definir a duração da mídia, seu tipo e qual a renderização. Esse último é utilizado para alertar o *player* qual o renderizador utilizar de acordo com o tipo de mídia, podendo ser um documento HTML ou não.

Além dos atributos, há a possibilidade de se definir opções de como a mídia de representação será exibida, podendo especificar tipos de transições, efeitos, velocidade, dentre

```
<layout schemaVersion="3" width="1920" height="1080" background="126.jpg" bgcolor="#FF3399">
    <region id="1" width="1920" height="1080" top="0" left="0" zindex="1">
        <media>...</media>
        <options>
            <loop>0</loop>
            <transitionType></transitionType>
            <transitionDuration></transitionDuration>
            <transitionDirection></transitionDirection>
        </options>
    </region>
    <tags>
        <tag>default</tag>
    </tags>
</layout>
```

Figura 2.4: Estrutura do XML Schema XIBO

```
<media id="" duration="" type="" render="">
    <options>
        <uri></uri>
        <transIn></transIn>
        <transInDuration></transInDuration>
        <transInDirection></transInDirection>
        <transOut></transOut>
        <transOutDuration></transOutDuration>
        <transOutDirection></transOutDirection>
    </options>
    <raw>

    </raw>
</media>
```

Figura 2.5: Elemento Mídia do XML Schema XIBO

outros. Uma instância completa pode ser vista no exemplo representado pela Figura 2.6.

```
<layout width="1920" height="1080" resolutionid="9" bgcolor="#ffffff" background="975.jpg">
   <tags>
      <tag>unittest</tag>
   </tags>
   <region id="1153" userId="1" width="1812" height="132" top="57.1875" left="54.7875">
      <media id="c962f98" type="text" render="native" duration="5" >
         <options>
            <xmds>1</xmds>
            <effect>none</effect>
            <speed>0</speed>
            <backgroundColor />
         </options>
      </media>
   </region>
   <region id="2033" width="1816.8" height="772.79" top="253.98" left="54.78">
      <media id="978" type="image" render="native" duration="10" lkid="2226">
         <options>
            <uri>978.png</uri>
         </options>
         <raw />
      </media>
      <media id="977" type="image" render="native" duration="10">
         <options>
            <uri>977.png</uri>
            <scaleType>center</scaleType>
            <align>left</align>
            <valign>middle</valign>
         </options>
         <raw />
      </media>
      <media id="976" type="image" render="native" duration="10">
         <options>
            <uri>976.png</uri>
            <scaleType>center</scaleType>
            <align>right</align>
            <valign>middle</valign>
         </options>
         <raw />
      </media>
   </region>
</layout>
```

Figura 2.6: Documento de representação XML Schema XIBO

2.3.2 4YOUSEE

A 4YouSee (4YOUSEE, 2014) é uma empresa dedicada ao desenvolvimento de soluções voltados a DOOH, de forma proprietária. Basicamente, sua plataforma é composta por 4 módulos: *Designer*, *Manager*, *Player* e *Analytics*. Como o foco desta dissertação é a autoria, apenas o módulo *Designer* e parte do módulo *Manager* serão detalhados.

O módulo de *Designer* é responsável pela criação e edição de conteúdos definindo o posicionamento de objetos de mídia de representação diversos. Além deste controle manual, o módulo permite fazer uso de arquétipos (*templates*) que facilitam e agilizam a criação de conteúdos por parte dos autores. Os objetos de mídia possíveis são imagens, vídeos, componentes responsáveis por informações climáticas, além de data e hora, normalmente tratados como *widgets*.

As criações destes conteúdos são realizadas por meio do elemento *canvas* do HTML5 (W3C, 2012b) . Esse elemento delimita uma área para renderização dinâmica de objetos gráficos ou desenhos, sendo o trabalho de criação e animação responsabilidade de uma linguagem dinâmica associada, como o *JavaScript*. Com base nisso, o módulo *Designer* permite a customização (SPYROU; IAKOVIDIS; MYLONAS, 2014) da apresentação por meio do posicionamento e tamanho dos objetos de mídias inseridos na área do conteúdo. Além da criação, o módulo *Designer* permite agrupar os conteúdos criados, de forma sequencial, como um *playlist*. Isso possibilita ao usuário elaborar uma apresentação com tempo de duração de cada conteúdo definido, além da definição de repetições. A troca de conteúdo pode acontecer com base no seu tempo de duração ou a partir da interação do usuário. Vale ressaltar que, no entanto, todos os conteúdos criados nesse módulo não tratam o sincronismo temporal entre os objetos de mídias de representação.

A Figura 2.7 ilustra o módulo 4YouSee *Designer*. No canto inferior da imagem, pode-se observar o conjunto de conteúdos organizados sequencialmente e a área responsável pela edição de um conteúdo pelo uso do *canvas* de *JavaScript*. A linguagem utilizada para persistir a apresentação criada é o *HTML5+JavaScript*. Essa apresentação pode ser exportada ou disponibilizada em um servidor, onde será acessada via *URL*. Controles como a duração da apresentação e o número de vezes que a apresentação será exibida também são abordados.

Dentre as ferramentas de autoria pesquisadas, essa era a única que permitia e facilitava a definição de pequenas interações por parte do criador de conteúdos sem que fosse

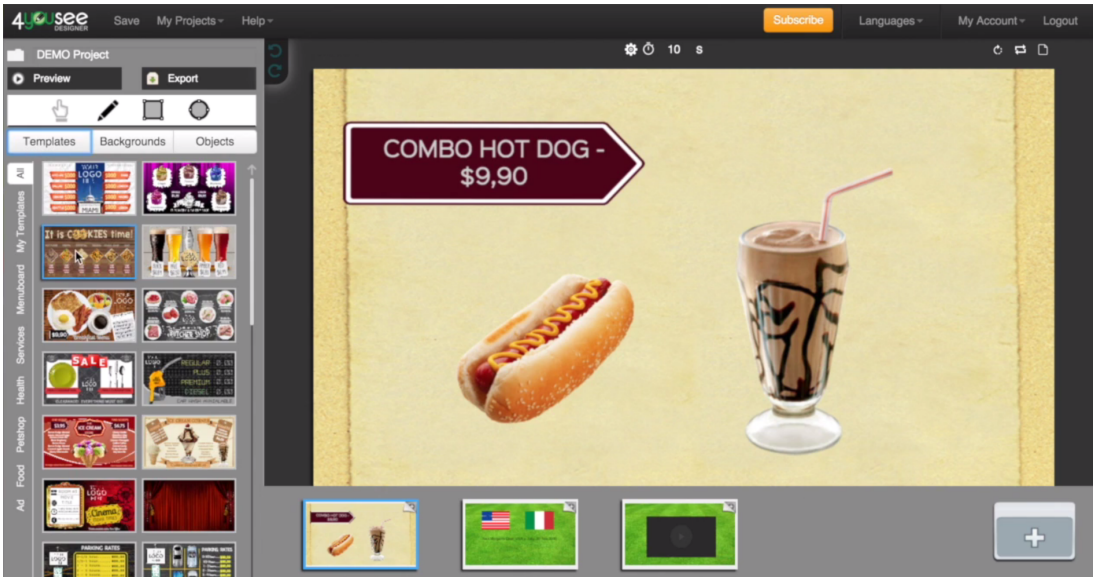


Figura 2.7: Módulo 4YouSee Designer

necessário o conhecimento em *scripts*. Como exemplo, os eventos de clique podem ser atribuídos a objetos de mídias de representação de modo a gerar ações, como exibir ou esconder uma imagem ou ir para um determinado instante da apresentação. Além disso, animações de movimentação de objetos de mídias também são tratadas.

O módulo *Manager* é responsável por organizar as diversas apresentações criadas em *playlists*, além de gerenciar a configuração e o cadastramento dos *players*. Uma apresentação desenvolvida no módulo *Designer* deve ser importada ou inserida via *URL* no módulo *Manager* para que o agendamento de exibição possa ser configurado, de acordo com uma data e hora em que deve ser exibida. Além disso, a ferramenta permite segmentar apresentações a determinados grupos de *players*.

2.3.3 DIGITALSIGNAGE.COM

DigitalSignage.com (INC, 2011) disponibiliza um serviço gratuito e *open source* que oferece as ferramentas de software necessárias para a criação e exibição de conteúdos em DOOH. Através de um software escalável e com suporte a tecnologia *Web* 2.0, denominado *StudioLite*, o usuário pode criar seus conteúdos de maneira bem intuitiva. Nessa ferramenta, pode-se controlar as divisões de tela, inserir canais e gerenciar suas mídias de representação e recursos.

A criação de conteúdos no *StudioLite* está associada aos conceitos de **campanha**, ***timelines*** e **canais**. Uma campanha é toda uma apresentação realizada pelo usuário,

composta por diversas ***timelines***. Cada ***timeline*** está associada a um *layout*, que define as regiões da tela onde serão exibidas as mídias de representação. Para o controle temporal dessas mídias, cada região do *layout* está associada a um **canal**, que é responsável pelo agendamento das mídias de forma sequencial e com durações customizáveis. A Figura 2.8 ilustra essas definições.

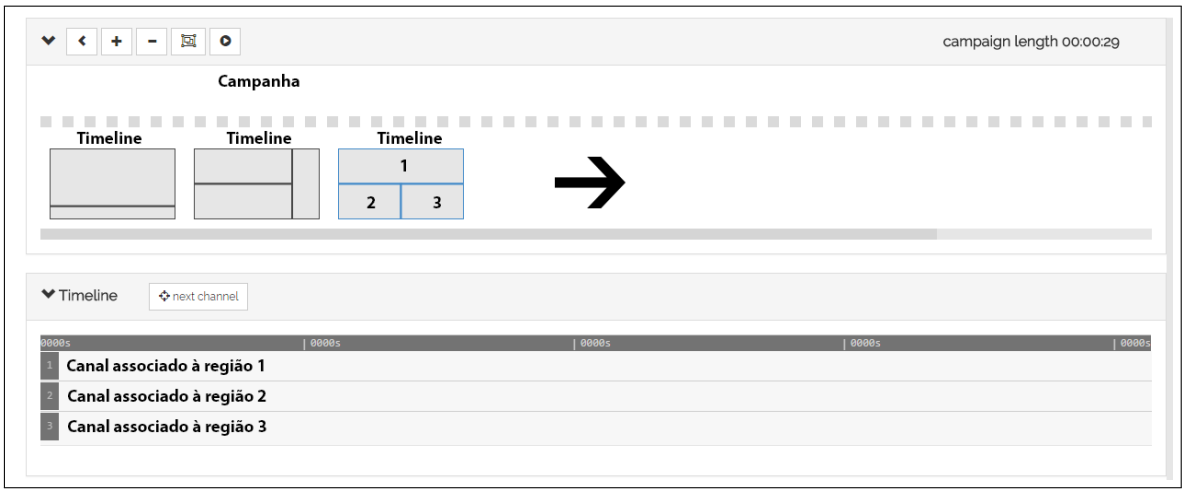


Figura 2.8: Componentes da Ferramenta de Autoria StudioLite

Com base nesses conceitos, o usuário pode criar sua apresentação (**campanha**), que irá apresentar de forma sequencial suas ***timelines***, inserindo objetos de mídias nos canais desejados, como vídeos, animações Adobe Flash, documentos HTML, dentre outros. Criada a apresentação, o usuário poderá disponibilizá-la para as estações configuradas na ferramenta. Essas estações são terminais equipados com o *player Web* apropriado capaz de interpretar os dados criados pelo usuário. Nesse sistema, o documento utilizado para representar a apresentação é processado em Flash, porém sua estrutura não é disponibilizada pelos autores e sua transmissão ocorre por meio de conexões *unicast* em modo *pull*.

2.4 REQUISITOS

Com base nas ferramentas gráficas de autoria mostradas na Seção 2.3 e no documento técnico da UIT (União Internacional de Telecomunicações) sobre casos de uso de interatividade em sinalização digital (ITU-GROUP16, 2014) , alguns requisitos foram levantados em apresentações para DOOH. Além disso, levou-se em conta a possibilidade de se desenvolver um modelo alinhado ao meio de comunicação social, além de possibilitar sua

utilização em infra-estruturas como a Rede Nacional de Ensino e Pesquisa (RNP), a qual pode prover um nível de Qualidade de Serviço (QoS) satisfatório para a utilização de TV sobre IP, conforme implementado em GT-IpêTeVê (2014).

Além disso, devido à possibilidade de se utilizar documentos multimídia para treinamento de corporações, esse estudo levou em conta alguns requisitos levantados pelo trabalho SceneSync (BUSSON et al., 2016) , o qual será visto com mais detalhes na Seção 3.2.

Portanto, cita-se como requisitos de autoria em DOOH:

- Suporte a implantação de conteúdo e aplicativos independentes da plataforma (RQ1);
- Representação por meio de documento multimídia com componentes reusáveis (RQ2);
- Divisão da apresentação em contextos semânticos diferentes, cada qual com seu momento de apresentação (RQ3);
- Navegação linear e não linear entre estes contextos semânticos (RQ4);
- Fácil sincronização das mídias de representação pertencentes a um contexto semântico (RQ5);
- Metadados responsáveis por identificar os componentes da apresentação para futura busca e recuperação (RQ6);
- Efeitos de transição entre objetos de mídias de representação (RQ7);
- Nível de abstração compatível com perfil dos criadores de conteúdo, de modo a facilitar a criação por usuários menos especializados (RQ8);
- Apresentações com conteúdos adaptativos, baseados em restrições de local geográfico, grupos de terminais, dentre outras condições personalizáveis (RQ9);
- Objetos de textos deslizantes (RQ10);
- Apresentação de conteúdo alternativo em casos de indisponibilidade da rede (RQ11);
- Suporte a diferentes orientações e resoluções de conteúdo (RQ12);
- Leiautes bem definidos e reusáveis (RQ13);

- Suporte a aplicações autocontidas baseadas em linguagens distintas (p.ex, NCL, HTML5, dentre outras) geralmente denominadas de *widgets* (RQ14);
- Suporte a diferentes tipos de instalações com uso de múltiplas telas(*videowall*) (RQ15);
- Necessidade de conteúdos imersivos com suporte a dispositivos móveis (RQ16);
- Suporte a interatividade do usuário e suas múltiplas modalidades (RQ17);
- Suporte a geração dinâmica de QR Codes (RQ18);
- Controle de ruído para conteúdos destinados a ambientes sensíveis (p.ex: Hospitais) (RQ19);

De forma compilada, a tabela 2.1 mostra uma relação entre os requisitos levantados nessa seção e o seu suporte nas ferramentas gráficas estudas na seção 2.3. O símbolo de "+" especifica que o requisito é suportado, o símbolo "-" especifica que o requisito não é suportado e o símbolo "+/-" especifica que o requisito é suportado parcialmente;

	RQ1	RQ2	RQ3	RQ4	RQ5	RQ6	RQ7	RQ8	RQ9	RQ10	RQ11	RQ12	RQ13	RQ14	RQ15	RQ16	RQ17	RQ18	RQ19
Xibo	-	+	+	-	+	-	-	+/-	+/-	+	-	+	+	+	-	-	-	-	-
4YouSee	+	-	+	+	-	-	-	+/-	+/-	-	-	+	+	+	-	-	+/-	-	-
DigitalSignage.com	-	-	+	-	+	-	+	+/-	+/-	+/-	+	+	+	+	-	-	-	-	+

Tabela 2.1: Suporte a Requisitos nas Ferramentas Gráficas de Autoria DOOH

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados alguns trabalhos relacionados a esse estudo. Esses trabalhos foram levantados a fim de se ter um panorama geral dos modelos e linguagens hipermídias utilizadas atualmente. Além dos trabalhos que abordam domínios semelhantes, foram observados trabalhos com aplicações em domínios distintos. Alguns destes contribuíram na especificação do modelo apresentado nesse estudo.

3.1 NCL - NESTED CONTEXT LANGUAGE

NCL (Nested Context Language) é uma linguagem declarativa definida em XML para autoria de documentos hipermídia baseados no modelo conceitual NCM (SOARES; RODRIGUES, 2005) . Inicialmente, a linguagem foi desenvolvida para o ambiente Web, no entanto, com o advento da TV Digital no Brasil e a necessidade de se desenvolver aplicações interativas para esse ambiente, se tornou a linguagem declarativa do padrão do Sistema Brasileiro de TV Digital Terrestre (ABNT, 2016) . Além disso, em 2009 foi definida como recomendação da União Internacional de Telecomunicações para serviços IPTV (ITU, 2014) . As entidades básicas do modelo NCM são mostradas na Figura 3.1.

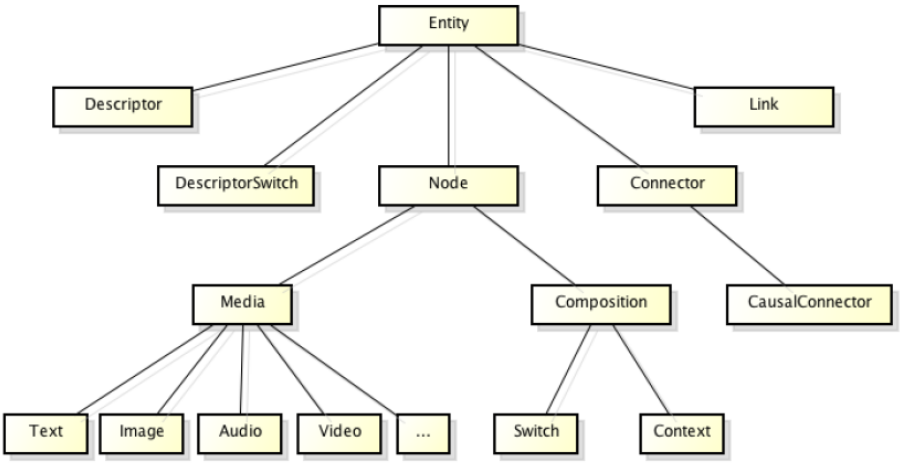


Figura 3.1: Entidades Básicas do NCM

Atualmente, NCL está em sua versão 3.0 (SOARES, 2009) e, a partir da versão 2.0, foi projetada de forma modular, permitindo a combinação de módulos para a obtenção de perfis mais apropriados para determinados usos da linguagem. O Enhanced Digital

TV Profile (EDTV) de NCL 3.0 é o perfil utilizado no Sistema Brasileiro de TV Digital. Já o *Raw Profile* (SOARES et al., 2010) , é um perfil voltado à simplificação da implementação de máquinas de apresentação, buscando o mapeamento dos elementos presentes em um documento NCL em elementos mais concisos, eliminando redundâncias, açucares sintáticos e estruturas de reuso. Devido à simplicidade e, ao mesmo tempo, poder de expressão, Raw Profile é usado como referência neste trabalho, em parte da definição de STorML.

Assim como outras linguagens de propósito específico em multimídia, NCL é utilizada para definir como os objetos de mídias são estruturados e relacionados, com a utilização de relacionamentos temporais e espaciais. No entanto, diferente de (W3C, 2008) , no qual os relacionamentos são definidos pela utilização de composições com semânticas incorporadas de sequenciação e paralelismo, em NCL as relações são baseadas no paradigma de causalidade (também chamado de orientado a evento) (SOARES; BARBOSA, 2012) .

NCL é organizada basicamente em contextos. Essa entidade tem como função a estruturação do documento de modo a organizar logicamente a apresentação, da mesma maneira que um livro está estruturado em capítulos (ANTONACCI et al., 2000) . Cada elemento deste é composto por diversos objetos de mídias os quais podem ser relacionados entre si com a utilização de elos causais (<*link*>) que definem, através do tratamento de eventos, quando (condições) uma mídia deve, por exemplo, iniciar, parar, pausar ou retomar sua exibição (ações).

A estrutura de um documento NCL é dividida entre o cabeçalho e corpo do documento. No cabeçalho estão definidas as bases de componentes que serão usados no corpo do modelo. Essas bases são estruturas responsáveis por organizar, controlar e favorecer o reúso dos elementos, conforme é abordado em Antonacci et al. (2000). Na base de regiões estão contidos os elementos <*region*> que determinam onde os objetos de mídias serão exibidos. A base de descritores define características específicas de como esses objetos serão exibidos, como, por exemplo, controlar o volume que um objeto de áudio deve ter durante sua exibição. Na base de conectores estão contidos os elementos <*causalConnector*> responsáveis por definir a semântica associada a elos (<*link*>). Respectivamente, essas bases respondem as seguintes questões: Onde exibir? Como exibir? e Quando exibir?. Além destas, há um base responsável por definir as regras que serão usadas no corpo do documento para adaptar conteúdos de acordo com análise prévia da regra em

elementos <*switch*> e <*descriptorSwitch*>. A Figura 3.2 ilustra a estrutura básica de um documento NCL.

```
<ncl id="exemplo" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
    <head>
        <ruleBase>
            <!-- regras para seleção de objetos e descritores -->
        </ruleBase>
        <regionBase>
            <!-- regiões onde as mídias serão apresentadas -->
        </regionBase>
        <descriptorBase>
            <!-- definição de como as mídias serão apresentadas -->
        </descriptorBase>
        <connectorBase>
            <!-- definição de como as mídias serão apresentadas -->
        </connectorBase>
    </head>
    <body>
        <port id="nomeDaPorta" component="umNoDesteContexto" />
        <!-- contextos, nós de mídia e suas âncoras, elos e outros elementos -->
    </body>
</ncl>
```

Figura 3.2: Estrutura básica de uma aplicação NCL

No corpo são especificados os nós de conteúdo, links, portas e nós de contexto. Os nós de conteúdos são os objetos de mídias especificados pelo elemento <*media*>, que serão usados na aplicação e apresentam um conjunto de propriedades (<*property*>). Essas propriedades facilitam a customização dos objetos de mídia, permitindo não só especificar mídias comuns, como áudio, vídeo e imagem, como também tratar documentos declarativos e baseados em *scripts*. Além disso, objetos de mídias possuem âncoras definidas pelos elementos <*area*> que especificam trechos temporais e espaciais.

Os elos (<*link*>), são elementos criados com o objetivo de relacionar os objetos de mídia, como, por exemplo, definir que um vídeo será iniciado após o término de um áudio. Para isso, o elo faz uso da semântica especificada em um conector <*causalConnector*>, definido na base de conectores no cabeçalho. Na especificação de um elo, os objetos de mídia participantes do relacionamento são associados (<*bind*>) aos papéis definidos no conector usado. A Figura 3.3 ilustra essa definição.

As portas (<*port*>) definem quais os objetos de mídia dentro do corpo do documento ou contexto serão exibidos inicialmente. Quanto ao contexto (<*context*>), conforme já mencionado, são elementos de composição responsáveis por estruturar logicamente o documento, dentro destes elementos são encontradas as mesmas entidades presentes no corpo

```
<link id="lMusic" xconnector="onEndStart">
    <bind role="onEnd" component="animation"/>
    <bind role="start" component="background">
        <bindParam name="delay" value="5s"/>
    </bind>
</link>
```

Figura 3.3: Estrutura de um elemento *link*

do documento, como portas, mídias, links e até outros contextos de forma aninhada. De fato, o corpo de um documento NCL é considerado um contexto.

Um outro recurso presente em NCL é a adaptação de conteúdos, tratada por meio de elementos <*switch*> e <*descriptorSwitch*>. Sua utilização é similar a outras linguagens como em W3C (2008). No entanto, NCL faz uso de regras <*rule*>, compostas por uma propriedade de mídia a ser avaliada, o operador de comparação e um valor ao qual a variável será comparada. As composições de nós alternativos <*switch*>, podem conter objetos de mídias, contextos ou outros *switches*. Portanto, de acordo com a avaliação das regras associadas por meio de elementos <*bindRule*>, o conteúdo correspondente será selecionado. A Figura 3.4 exemplifica tal situação, na qual é possível identificar uma base de regras relacionadas ao idioma do sistema e um elemento <*switch*>. Nesse elemento, estão definidas as mídias que poderão ser exibidas de acordo com a avaliação da regra. A primeira mídia que tiver sua regra validada será exibida.

```
<head>
    <ruleBase>
        <rule id="rEn" var="system.language" comparator="eq" value="en"/>
        <rule id="rPt" var="system.language" comparator="eq" value="pt"/>
    </ruleBase>
</head>
<body>
    <media id="noSettings" type="application/x-ncl-settings">
        <property name="system.language">
    </media>
    <switch id="switchIdioma">
        <bindRule rule="rEn" constituent="audioEn"/>
        <bindRule rule="rPt" constituent="audioPt"/>
        <media id="audioEn" src="media/audioEn.mp3" descriptor="dAudio"/>
        <media id="audioPt" src="media/audioPt.mp3" descriptor="dAudio"/>
    </switch>>
</body>
```

Figura 3.4: Adaptação de conteúdo NCL por meio do elemento *switch*

Vale ressaltar que NCL também permite exibição de mídias em múltiplos dispositivos, classificados em dois tipos de grupos: Dispositivos ativos e dispositivos passivos. O

primeiro tipo trata dos dispositivos capazes de interpretar um documento NCL e que, portanto, possuem capacidade de exibição de mídias, incluindo documentos NCL como mídia. Dispositivos passivos são os que apenas recebem via *streaming* o conteúdo resultante dos sincronismos, que são, de fato processados em um receptor ativo pai. Essa especificação é feita no documento para cada base de regiões, através de seu atributo *device*. A omissão do atributo define a exibição no dispositivo principal (SOARES; BARBOSA, 2012) . A Figura 3.5 ilustra esse conceito.

```
<head> <!-- uma base para cada dispositivo -->
    <regionBase id="rbTV"> <!-- dispositivo: TV (default) -->
        <region id="rgTVtelaInteira"> <!-- tela da TV -->
            <region id="rgTVmenu" > <!-- menu na TV -->
                <region id="rgTVmenu1" />
                <region id="rgTVmenu2" />
                <region id="rgTVmenu3" />
            </region>
        </region>
    </regionBase>
    <regionBase id="rbPDA" device="systemScreen(1)"> <!-- disp:PDA-->
        <region id="rgPDAtelaInteira"> <!-- tela do PDA -->
            <region id="rgPDAmenu" > <!-- menu no PDA -->
                <region id="rgPDAmenu1" />
                <region id="rgPDAmenu2" />
                <region id="rgPDAmenu3" />
            </region>
        </region>
    </regionBase> <!-- continuação do cabeçalho da aplicação -->
</head>
```

Figura 3.5: Definição das bases de regiões para dois dispositivos e suas regiões filhas.

3.1.1 ELOS CAUSAIS NCL EM TEXTO CORRIDO

No SBTVD, a linguagem NCL está padronizada em sua versão 3.0. Já em IPTV, a linguagem encontra-se especificada em sua versão 3.1 (ITU, 2014) . No desenvolvimento da versão 3.1, houve propostas publicadas na literatura que exercitaram alternativas interessantes para a simplificação da linguagem, como o trabalho Soares G.F. Lima (2010). Tal proposta mantinha todas as funcionalidades da versão anterior, porém buscava facilitar a autoria com a eliminação de alguns elementos destinados ao reúso, além de mudanças na definição de elos pelo elemento <*link*>.

Os elos, até então, eram definidos com a associação das mídias a papéis de conectores, de tal modo que a expressividade e a reusabilidade eram recursos valorizados na produção de diversos documentos. No entanto, a dificuldade e os problemas em se criar esses

relacionamentos definidos nos conectores levantaram a questão de se tratar a semântica de relações diretamente nos elos, sem a utilização da base de conectores. Dessa forma, foi introduzido, de acordo com Soares G.F. Lima (2010), um açúcar sintático na linguagem com objetivo de facilitar a autoria de relacionamentos entre os objetos de mídias.

A abordagem permite a definição dos relacionamentos em texto corrido, uma espécie de pseudo código em linguagem quase natural, que compõe o conteúdo do <*link*>. A imagem 3.6 ilustra essa facilidade mostrando a forma realizada na versão 3.0 e a nova proposta definida em (SOARES G.F. LIMA, 2010) .

No entanto, a especificação de elos em texto corrido não chegou a ser incorporada às especificações de NCL, seja para IPTV ou TV Digital. Como forma de retomar as discussões em torno de tal forma de declaração de elos, a linguagem definida nesta dissertação faz uso da mesma abordagem, tornando a abstração de elo um pouco mais acessível aos produtores de conteúdo, ao mesmo tempo em que se mantém a expressividade dos relacionamentos. No Capítulo 4 esse formalismo será descrito em mais detalhes.

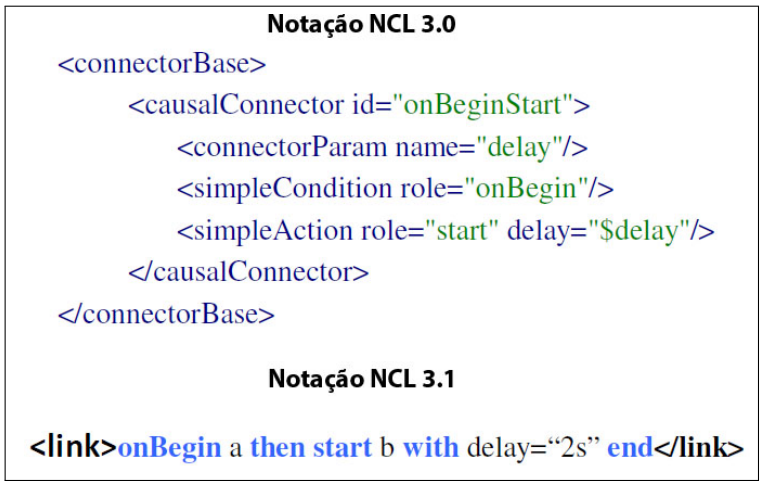


Figura 3.6: Comparativo entre as definições de elos NCL.

3.2 SCENESYNC

SceneSync (BUSSON et al., 2016) é uma linguagem desenvolvida com o propósito de autoria hipermídia para conteúdos compostos por objetos de aprendizagem. Baseada em um modelo conceitual simplificado, a linguagem aborda a sincronização de OAs em estruturas que, conceitualmente, se assemelham ao modelo proposto neste estudo. No entanto, aplicado a um domínio de aplicação distinto.

Essa semelhança está principalmente relacionada ao conceito de cena, utilizado por SceneSync. Com o mesmo propósito de elevar o nível de abstração ao criador de conteúdos, SceneSync define essa entidade como um conjunto de mídias relacionadas que estão organizadas temporalmente, baseadas em um cronômetro interno. Portanto, seu sincronismo é baseado em linha do tempo, diferente do estruturado utilizado por Soares e Barbosa (2012). Além disso, o trabalho destaca a interoperabilidade da linguagem, abordando conversões para HTML5 e NCL.

Conforme ilustrado na Figura 3.7, seu modelo define um número menor de entidades quando comparado ao NCL. Basicamente, sua estrutura é composta por cenas, mídias e âncoras. O modelo faz a analogia de que, assim como uma peça teatral, a apresentação é composta por um conjunto de cenas ligadas de acordo com o tema tratado. Portanto, metaforicamente, uma cena é constituída de mídias(atores), as quais apresentam propriedades (figurinos) específicas ao tema, e âncoras que representam as ações disparadas durante uma cena. Por definição, uma cena não pode conter outros objetos de cena.

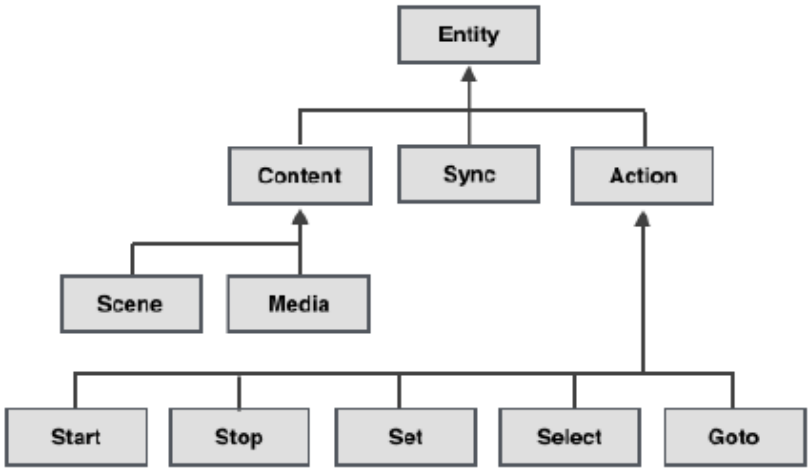


Figura 3.7: Entidades do Modelo usado por SceneSync.

O documento SceneSync segue a estrutura inicial de NCL e SMIL (W3C, 2008) . Seu elemento raiz <*scenesync*> é composto pelo cabeçalho <*head*> e o corpo do documento <*body*>. O cabeçalho contém apenas o elemento <*meta*>, responsável por descrever o documento. No corpo é definida a semântica de apresentação, contendo os objetos de cena <*scene*>, suas mídias *(*<*image*>,<*video*>,<*text*>,<*audio*>*)* e os objetos de sincronismo definidos por meio do elemento <*sync*>. A imagem 3.8 ilustra essa hierarquia do documento.

Para exemplificar seu uso, a Figura 3.9 ilustra uma instância do modelo representando

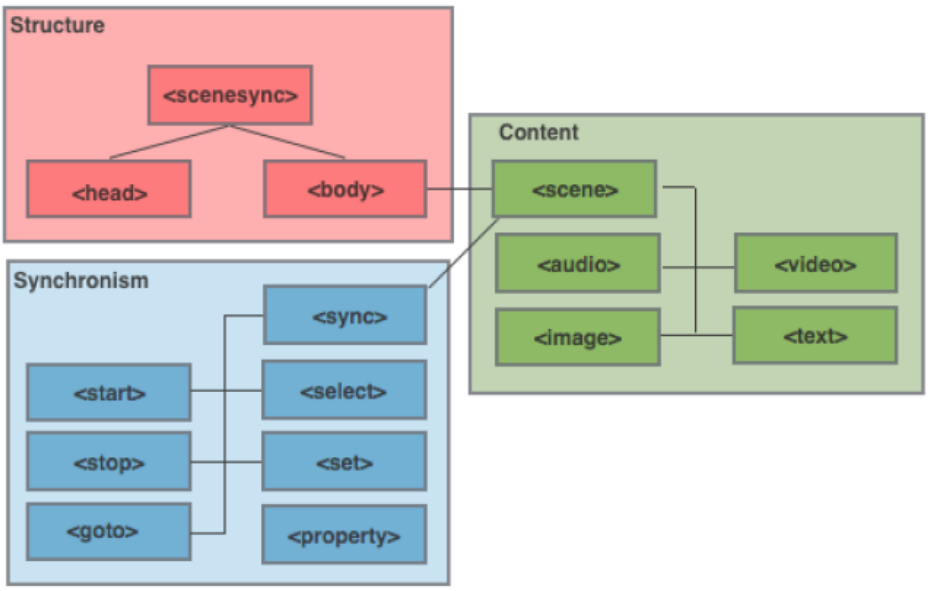


Figura 3.8: Módulos e elementos da linguagem SceneSync

uma cena composta por duas âncoras que iniciam, em tempos distintos, duas mídias da cena em locais e tamanhos especificados pelas propriedades dos objetos de mídia *width*, *height*, *left* e *top*. Além destes atributos, a mídia contém o atributo *src* que especifica a *URI* do conteúdo do objeto. Todos os elementos de um documento são identificados e referenciados pelos seus respectivos atributos *id*.

```
<scenesync>
    <head>
        <meta/>
    </head>
    <body>
        <scene id="scene1">
            <img id="image1" src="files/fungo.png" left="60%" top="20%"  width="20%" height="50%"/>
            <video id="video1" src="files/briofitas.mp4" left="5%" top="5%" width="40%" height="30%"/>
            <sync id="sync1" time="0s">
                <start id="startvideo" target="video1"/>
            </sync>
            <sync id="sync2" time="10s">
                <start id="startimage" target="image1"/>
            </sync>
            <sync id="sync2" time="20s">
                <stop id="stopImage" target="image1"/>
            </sync>
        </scene>
        ...
        <scene>
        </scene>
    </body>
</scenesync>
```

Figura 3.9: Instância do Modelo sobre aula de biologia

A instância ilustrada na Figura 3.9 define que a cena identificada como **scene1** pelo atributo *id*, exibirá (<*start*>) o vídeo de *id* **video1** (*target*) no instante **0s** da cena através

do elemento (<*sync*>), conforme identificado por "A"na Figura 3.10. Após a exibição do primeiro objeto, aos **10s** da cena, a imagem de *id* **image1** (*target*) será exibida (<*start*>), conforme identificado por "B"na Figura 3.10. Aos **20s**, a imagem **image1** (*target*) iniciada aos **10s**, terá sua exibição finalizada (<*stop*>), conforme identificado por "C"na Figura 3.10.

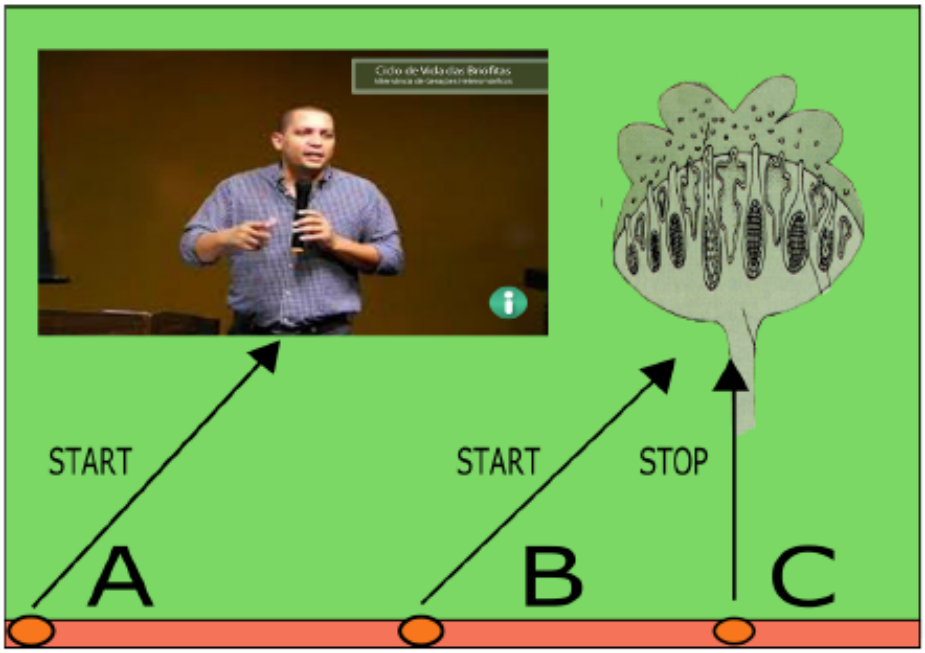


Figura 3.10: Visão espaço-temporal da cena sobre aula de biologia

Além de tais relacionamentos temporais existentes dentro de uma cena, o modelo propõe ações que tratam os elos entre cenas. Para isso, são definidas ações no tempo (<*goto*>) por meio de seleção do usuário (<*select*>) que possibilitam a navegação entre cenas, além de permitir a especificação de um instante de tempo na cena de destino.

Portanto, as ações presentes no SceneSync, executada internamente a um elemento <*sync*>, são representadas pelos seguintes elementos:

- <*start*> - Inicia a apresentação de um objeto de conteúdo interno à cena.
- <*stop*> - Termina a apresentação de um objeto de conteúdo interno à cena.
- <*set*> - Altera os valores das propriedades de um objeto de conteúdo interno à cena.
- <*goto*> - Redireciona a apresentação para outra cena ou instante de tempo.
- <*select*> - Redireciona a apresentação para outra cena ou instante de tempo ao disparo de um evento de entrada (pressionar tecla, clique do mouse, etc)..

Dessa forma, pode-se observar que os relacionamentos abordados por SceneSync são sempre da ordem de 1:N. Portanto, apenas uma condição simples será avaliada, a qual poderá acionar N ações no caso de ser validada. Isso limita um pouco as possibilidades existentes de interatividade e relacionamentos, diferente do que é tratado em Soares e Barbosa (2012), o qual aborda relacionamentos causais N:M. Além disso, devido à simplicidade da linguagem, os documentos SceneSync são normalmente extensos.

Em comparação ao modelo definido nesta dissertação, vale ressaltar o domínio específico para objetos de aprendizagem, não contemplando, portanto, alguns dos requisitos relevantes levantados para DOOH. Outra questão a ser destacada é seu pouco foco no reúso, requisito importante em DOOH, devido ao uso repetitivo de descrições, como as de leiautes comuns, com disposições espaciais bem definidas.

3.3 SMIL

Em meados da década de 90, com o aumento da utilização da *Web* como um ambiente multimídia, surgiu a necessidade de se tratar a criação de apresentações com conteúdos de áudio, vídeo, texto e imagens. Baseado nisso, foi criada a linguagem SMIL(*Synchronized Multimedia Integration Language*), que atualmente se encontra em sua versão 3.0 (W3C, 2008) .

SMIL é uma linguagem declarativa baseada em XML e recomendada pelo W3C para descrever apresentações multimídia destinadas à *Web*, com o objetivo de complementar o HTML. Trata-se de uma linguagem responsável pela definição de marcações relacionadas a sincronizações de mídias, definição de leiautes, animações, dentre outras características.

Portanto, o objetivo da linguagem é especificar uma maneira do autor descrever, de forma textual, uma apresentação multimídia. Assim, o autor pode determinar comportamentos temporais de uma apresentação multimídia, associar *hyperlinks* com objetos de mídia e definir o Leiaute de apresentação em um *display*.

A linguagem também divide a estrutura de um documento em duas partes, o cabeçalho (*head*) e o corpo (*body*), da mesma forma que NCL e outras linguagens padronizadas pelo W3C, como o HTML. No cabeçalho, se encontra a estrutura de leiaute da apresentação. Em seu módulo básico, o leiaute é composto por uma ou mais regiões (*region*) e um leiaute raiz (*root-layout*) responsável por determinar o tamanho da janela onde as regiões estarão presentes. A distribuição espacial das regiões é definida baseando-se nas distâncias com

relação ao ponto superior esquerdo da janela (*root-layout*). Para isso, são definidos os atributos de distância em relação ao topo (top), em relação à borda esquerda (*left*), em relação à direita (*right*) e também em relação à borda inferior (*bottom*) da janela (W3C, 2008) . A Figura 3.11 demonstra como são definidos esses conceitos em SMIL.

```
<layout>
    <root-layout width="640px" height="480px"/>
    <region id="whole" top="0px" left="0px" width="640px" height="480px" z-index="5"/>
    <region id="right" top="0px" left="320px" width="320px" height="480px" z-index="4">
        <region id="inset" top="140px" left="80" width="160px" height="200px" z-index="6"/>
        <region id="inset2" top="140px" left="80" width="160px" height="200px" z-index="6"/>
        <region id="inset3" top="140px" left="80" width="160px" height="200px" z-index="7"/>
    </region>
</layout>
```

Figura 3.11: Exemplo de uso de leiautes em SMIL

No corpo do documento é definida toda a semântica da apresentação. Basicamente, sua organização é estruturada pelo aninhamento de três composições principais, *par*, *seq* e *excl*. A composição *par* define que todos os seus elementos filhos serão exibidos em paralelo, enquanto a composição *seq* define que todos serão exibidos em sequência. A composição *excl* se comporta como uma composição *par*, no entanto há a restrição de que apenas um elemento filho pode ser exibido por vez. Cada composição desta pode conter, além de outras composições, objetos de mídia, como imagens (*image*), vídeos (*video*), áudios (*audio*) e textos (*text*). A associação destes objetos com as regiões é realizada através do atributo *region* do objeto de mídia. A Figura 3.12 ilustra esses conceitos.

```
<body>
    <par system-bitrate="75000">
        <audio src="audio/newsong1.snd"/>
        <video src="video/newsong1.avi" region="inset"/>
        <image src="lyrics/newsong1.gif" region="inset"/>
    </par>
    <seq>
        <img src="slide1.jpg" dur="10s" region="right"/>
        <img src="slide2.jpg" dur="10s" region="right"/>
        <img src="slide3.jpg" dur="10s" region="right"/>
    </seq>
    <excl dur="indefinite">
        <priorityClass peers="stop">
            <video src="video/movie1.avi" region="whole"/>
            <video src="video/movie2.avi" region="whole"/>
            <video src="video/movie3.avi" region="whole"/>
        </priorityClass>
    </excl>
</body>
```

Figura 3.12: Exemplo de uso das composições de SMIL

Composições SMIL foram levadas em consideração para a definição de algumas enti-

dades do modelo definido nesta dissertação. Isso porque apresentam semântica embutida, que evita a definição de elementos repetitivos para representar os mesmos relacionamentos, como acaba por ser necessário em NCL (múltiplos *links* para sequenciação, por exemplo). Além das composições SMIL, foram observados alguns atributos importantes de suas entidades, por definirem comportamentos das mídias que são interessantes quando confrontados com os requisitos de DOOH.

Em abril de 2012 o W3C extinguiu o comitê responsável pela evolução da linguagem, apontando o HTML5 (W3C, 2012b) como tópico relacionado (BUSSON et al., 2016) .

3.3.1 A-SMIL

A-SMIL (A-SMIL, 2010) é uma aplicação da linguagem SMIL em ambientes de *Digital Signage*. Algumas empresas estão fazendo uso dessa linguagem para a representação de seus conteúdos (INNES, 2005) . Basicamente, A-SMIL utiliza algumas das estruturas existentes no modelo SMIL. O objetivo é permitir ao produtor de conteúdos criar suas apresentações que serão exibidas no terminal do cliente por um aparelho de hardware (*player*) compatível com SMIL (IADEA, 2000) .

Podemos dizer que A-SMIL é uma espécie de perfil SMIL composto por alguns módulos que atendem aos requisitos existentes em DOOH. Desde SMIL 2.0, a linguagem é organizada em módulos, o que permite que parte da linguagem possa ser reutilizada por outras linguagens baseadas em XML. A linguagem aborda os objetos de mídias tradicionais, definidos pelo módulo *BasicMedia*, assim como todos os elementos de composição responsáveis pela semântica da apresentação, citados anteriormente: *par*, *seq* e *excl*, que estão presentes no módulo *BasicContentControl* (A-SMIL, 2010) .

Quanto à organização dos objetos de mídia na tela, A-SMIL faz uso do módulo *BasicLayout*. Com esse módulo, é possível definir o tamanho da janela na qual será exibido o conteúdo, assim como determinar o posicionamento de todas as regiões internas a essa janela onde serão exibidos os objetos de mídia associados. Sua estrutura segue o mesmo padrão definido pelo módulo *Structure*, no qual o documento é subdividido entre o cabeçalho (*head*) e seu corpo (*body*).

Além desses módulos básicos pertencentes ao SMIL, alguns requisitos importantes em DOOH também são tratados, como o agendamento de exibição dos objetos de mídias, sendo essa uma diferença destacada pelo seus utilizadores (INNES, 2005) . Trata-se

de uma restrição na exibição do conteúdo associado a uma condição de apresentação que segue os padrões definidos pela norma internacional ISO-8601 (ISO, 2004) . Esse padrão é responsável pela representação de diversos formatos de data e hora, usados para o intercâmbio e manipulação de documentos. Sua utilização em A-SMIL permite não só definir o dia em que determinado trecho do conteúdo será exibido, como também sua periodicidade. Para definir essa restrição na linguagem é preciso inserir os valores no atributo *begin* e *end* dos elementos definidos pelo módulo *BasicContentControl*. No entanto, vale ressaltar que, de acordo com A-SMIL (2010), o uso deste mecanismo só é permitido em composições internas a elementos *par* ou *seq*. Um exemplo pode ser visto na Figura 3.13.

```
<par dur="indefinite">
    <seq begin="wallclock(R/2000-01-01+w3T09:00/P1W)" end="wallclock(R/2000-01-01+w3T12:00/P1W)">
    ...
    </seq>
</par>
```

Figura 3.13: Recurso de agendamento de conteúdo em A-SMIL

A figura acima define que todos os objetos de mídias pertencentes ao elemento *seq* serão exibidos toda quarta-feira de cada semana entre 09:00 e 12:00 horas. Dessa forma, A-SMIL define o agendamento diretamente no modelo, o que pode ser um ponto negativo quando levado em conta a reusabilidade.

Como complemento, A-SMIL busca em outras linguagens uma forma de aprimorar o tratamento do conteúdo no terminal. Por exemplo, é possível utilizar *ECMAscript*(*JavaScipt*) como uma mídia HTML para alterar conteúdos que estão relacionados às condições dinâmicas, envolvendo fontes de dados da *Web*. Outro recurso interessante é a utilização do elemento SMIL *prefetch*. Esse recurso permite que grandes arquivos de mídia, como vídeos, possam ser "pré-carregados" para o *cache*, permitindo que sua execução posterior seja realizada normalmente, sem atrasos ou esperas indevidas por parte dos consumidores de conteúdo (A-SMIL, 2010) .

Em aspectos relacionados à DOOH, SMIL se apresenta como uma linguagem flexível o bastante para permitir sua plena utilização. No entanto, como seu domínio proposto inicialmente foi a *Web*, o alto número de entidades e suas complexidades fazem de sua construção e entendimento dificultadores de seu uso diretamente. Como forma de potencializar o uso de SMIL nesses ambientes, algumas empresas e trabalhos específicos, como INNES (2005) e Cazenave, Quint e Roisin (2011), têm experimentado mesclar a

linguagem SMIL, de modo a utilizar de seus pontos positivos com outras tecnologias *Web*. Tais soluções baseiam-se na estrutura do SMIL incorporada à utilização de CSS3, buscando a utilização de um padrão mais conhecido para adicionar formatação aos documentos. Além disso, essa forma híbrida provê suporte a *JavaScript* e HTML5 (W3C, 2012b) , o que possibilita expandir a utilização de SMIL para um ambiente mais interativo e imersivo (DUPIN, 2011) .

3.4 HTML5

A utilização de elementos multimídia na *Web* está crescendo de forma acelerada. Baseado nisso, o HTML5 (W3C, 2012b) surge como uma tecnologia para modernizar a *Web* que, até então, era voltada apenas para apresentação de documentos hipertextos com imagens. Com seu advento, tornou-se possível o desenvolvimento de inúmeras aplicações de modo a permitir uma maior integração com áudios e vídeos. Além disso, sua forte associação com uma linguagem de *scripts* permitiu dinamizar ainda mais e, até mesmo, criar conteúdos de animação.

Conforme citado em (BUSSON et al., 2016) , essa nova versão do HTML traz como funcionalidades: API para desenvolvimento de gráficos bidimensionais; Controle embutido de conteúdo multimídia; Elementos de áudio e vídeo como elementos de primeira classe; Aprimoramento do uso *off-line* e melhoria na depuração de erros.

Diretamente associado a essas melhorias, muitos navegadores já estão adotando esse padrão. A tendência é o aumento do número de dispositivos que possam interpretar esses novos conteúdos multimídia, não apenas computadores pessoais tradicionais, como também dispositivos móveis de diversos tipos e tecnologias.

A estrutura básica do HTML5 segue a mesma tratada na versão anterior. A Figura 3.14 ilustra uma aplicação.

A primeira linha define o tipo do documento que o navegador deve interpretar. Em versões anteriores, essa descrição era mais complexa, dada a necessidade de explicitar o arquivo DTD responsável pelas definições. O atributo *lang*, presente na raiz do arquivo HTML <*html*>, determina o idioma principal do documento. No cabeçalho (<*head*>) é possível inserir diversos metadados, os quais serão usados pelos exibidores, denominados *user-agents*. Quanto aos elementos necessários no cabeçalho, destaca-se o metadado com o atributo *charset*, responsável por informar ao navegador qual a codificação de caracteres

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <title>Áudio em HTML5</title>
  </head>
  <body>
    <audio controls autoplay>
      <source src="audio.ogg" />
      <!-- Mensagem explicando que o navegador não suporta áudio ou o formato usado. -->
      <p>Seu navegador não suporta áudio HTML5 ou o formato Opus.</p>
    </audio>
  </body>
</html>
```

Figura 3.14: Uso do elemento de primeira classe áudio em HTML5

utilizada. Além dessa informação, define-se ali o título da página (<*title*>), muito usado por mecanismos de busca na *Web*. O cabeçalho é também o local onde se faz referências a arquivos de estilos (CSS3) e *scripts* em geral.

Em seu corpo (*body*), são definidas as semânticas do documento. Anteriormente, o HTML era baseado apenas na utilização de elementos do tipo <*div*>. Em HTML5 novas *tags* semânticas foram inseridas, a fim de organizar e melhor definir os dados contidos ao longo do documento, facilitando sua interpretação e exibição pelos *user-agents*. Pode-se citar como novos elementos: O <*header*>, <*nav*>, <*section*>, <*article*>, <*footer*>, dentre outros.

Na Figura 3.14, observa-se o uso de um novo elemento de primeira classe <*audio*>, próprio para o tratamento de mídias do tipo áudio, em substituição aos antigos *plugins*. Nesse exemplo, um áudio referenciado pelo elemento <*source*> é executado com os controles básicos do *player* definidos pelo atributo *controls*. Além disso, o áudio é iniciado no mesmo instante em que seu arquivo é carregado, pois foi inserido o atributo *autoplay* no elemento <*audio*>. Caso o navegador não suporte esses recursos, uma mensagem de incompatibilidade será exibida ao usuário.

Essa é apenas uma ilustração simples de como o HTML5 aborda mídias do tipo áudio e vídeo. No entanto, percebe-se que esse é um controle responsável apenas pela exibição e que não trata os diversos relacionamentos possíveis entre mídias. Para esse controle de sincronismo, se faz necessária a utilização de linguagens de *script*, como *JavaScript*, passando, nesse caso, para uma abordagem imperativa.

Baseado nessas novas possibilidades proporcionadas pelo HTML5, o mercado de DOOH tem expandido o uso de TVs ligadas a computadores que fazem o uso de *players Web* (CAR-

NEIRO, 2013) , responsáveis por exibirem os conteúdos criados nessa linguagem. Além do sincronismo entre as mídias, o HTML5+*JavaScript* conta com o alto poder interativo, o que favorece a criação de aplicações mais imersivas em mídias DOOH. Além disso, destaca-se a interoperabilidade e adaptabilidade destes conteúdos web com relação a diversidade de dispositivos existentes, cada qual com seu tamanho, capacidade de processamento e resolução. Essas vantagens diminuem, consequentemente, o custo de implantação desta tecnologia.

No entanto, muitos pontos negativos ainda são destacados na utilização do HTML5, conforme elencado em Antonacci et al. (2000): O HTML ainda não é extensível; devido ao uso massivo de *scripts*, sua semântica é difícil de ser extraída; e os relacionamentos são limitados a *hyperlinks* do tipo "*go to*", exceto nos casos da utilização de *scripts*, que no entanto, dependem de profissionais capacitados para a criação dos conteúdos. Além disso, o terminal deve ter capacidade suficiente para processar e exibir os conteúdos recebidos.

3.5 MPEG-4

Como trabalho relacionado, pode-se citar o padrão MPEG-4 (AVARO et al., 2000) , que também faz uso de uma estrutura baseada em cenas para representar os objetos de mídias, no entanto seu propósito é diferente ao tratado nesta dissertação.

O MPEG-4 é um padrão utilizado para compressão de dados digitais de áudio e vídeo. Sua arquitetura para a codificação orientada a objetos de mídia é especificada na parte *Systems* (ISO/IEC 14496-1). Além deste modelo arquitetural, o MPEG-4 *Systems* define também os formatos adotados para a especificação das cenas audiovisuais (COSTA, 2005) .

No MPEG-4, uma cena audiovisual pode ser composta por vários objetos (naturais ou sintéticos). Objetos naturais são os elementos de áudio, vídeo e imagem presentes na cena. Objetos sintéticos são estruturas 2D que podem ser criadas pelo autor, como linhas, retângulos, círculos e etc. Em uma cena, os objetos presentes estão organizados de forma hierárquica, conforme imagem 3.15

Os objetos de mídias são representados pelas extremidades da hierarquia. Os demais nós são composições de objetos de mídias que vão se organizando até o nível de cena. Essa composições podem ser comparadas a contextos, como em NCL (SOARES; BARBOSA, 2012) , responsáveis apenas por organizar e agrupar os objetos. A estrutura da cena pode

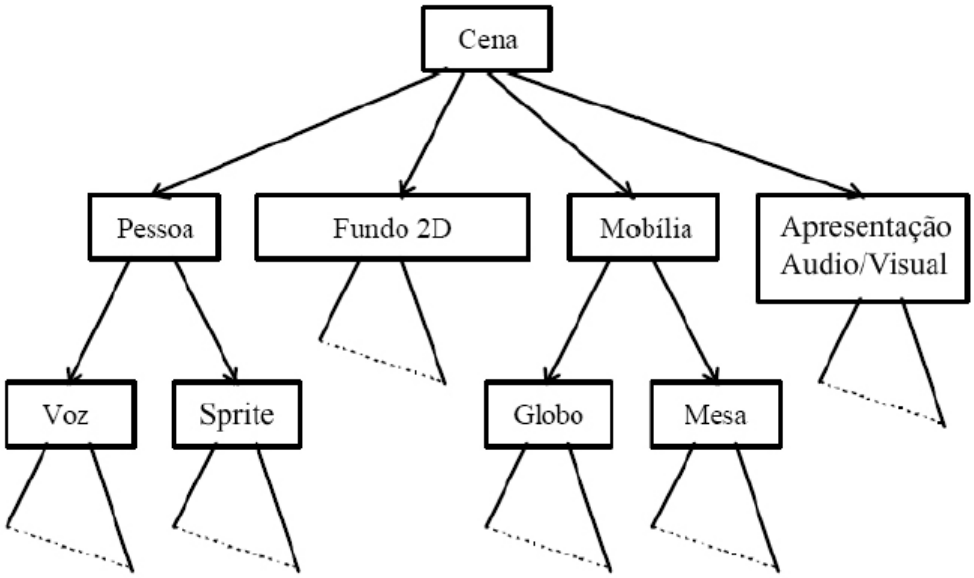


Figura 3.15: Estrutura Hierárquica de uma Cena MPEG-4.

ser modificada a qualquer tempo, de modo dinâmico, como a alteração de posicionamento dos objetos na cena (COSTA, 2005) .

Cada objeto de mídia pode ser codificado em um ou mais fluxos elementares, e para dar semântica e organizar esses objetos, a cena é também codificada em um fluxo paralelo, em formato binário, denominada BIFS. Esse formato foi definido com o intuito de reduzir o tamanho do conteúdo, de modo a facilitar o transporte. Por possuir um formato binário, seu conteúdo é verboso e de difícil autoria. Pensando em facilitar sua criação, desenvolveu-se um formato de sintaxe textual declarativa, denominado XMT. Esse formato possui dois níveis de representações, o XMT-A (alpha) e XMT-O (omega) (AVARO et al., 2000) .

XMT-A é a representação textual direta do BIFS. Nela são definidas todas as expressões que existiam no formato binário, portanto, trata-se de uma linguagem, apesar de textual, ainda muito verbosa e complexa. Para aumentar sua abstração e facilitar ainda mais a autoria, a linguagem XMT-O foi desenvolvida, a qual é baseada no SMIL 2.0. Sua estrutura faz uso das composições semânticas já citadas na Seção 3.3, são elas: <*par*>, <*seq*> e <*excl*>. Como essas entidades foram adaptadas para representar uma organização definida como cena, muitas das vezes é necessário utilizar relacionamentos que são compostos por vários níveis de aninhamento (BUSSON et al., 2016) . Portanto, observa-se na Figura 3.16 que, mesmo havendo o conceito inicial de cenas, sua especificação textual não define essas entidades em alto nível, se mantendo, ainda, como uma linguagem verbosa e de difícil compreensão.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<XMT-O xmlns="urn:mpeg:mpeg4:xmto:schema:2002" ...>
    <head>
        <layout type="xmt/xmt-basic-layout" metrics="pixel" >
            <topLayout id="window1" ...>
                <region id="title" size="600 60" translation="0 145" .../>
                ...
            </topLayout>
        </layout>
        <defs>
            <img id="defImg" src="picture.jpg" … />
        </defs>
        <meta name="copyright" content="(c)Copyright Telemidia" … />
    </head>
    <body>
        <par id="ncl_example-processed">
            <rectangle region="title" dur="indefinite" size="600 60">
                <material color="white" filled="true"/>
            </rectangle>
        <par id="coisaPele">
        <par id="ncl-composite-1">
            <audio id="samba" src="coisa.mp3" region="audioRegion1">
                <rectangle id="part1" begin="samba.begin+8.4s" end="samba.begin+18s">
                ...
                </rectangle>
            </audio>
            <lines region="audioRegion1" />
                ...
        </par>
        ...
        </par>
        <string id="title-coisaPele" textLines="Coisa de Pele">
            <material color="black"/>
            <fontStyle family="TYPEWRITER" ... size="22" />
        </string>
        ...
        <rectangle id="logotele1" ... begin="samba.begin" end="samba.end">
            <texture src="logo.jpg"/>
        </rectangle>
        <use xlink:href="#defImg" dur="indefinite" />
    </body>
</XMT-O>
```

Figura 3.16: Exemplo de documento especificado em XMT-O.

Para facilitar o entendimento do conceito de cena definido por MPEG-4, a Figura 3.17 demonstra uma cena e seus objetos de mídias. A cena é composta pelo objeto de imagem VOP1, responsável pela paisagem de fundo, além de um objeto de vídeo VOP2 e um objeto de imagem VOP3, todos esses sobrepostos à VOP1. Além da exibição de objetos de mídias, MPEG-4 também permite reconhecer eventos em uma cena, como a seleção de uma imagem por meio de um clique, que pode gerar uma determinada ação. Como ilustração, na Figura 3.17, o usuário poderá selecionar o objeto VOP2 e ser exibida uma fotografia sobre desmatamento (2). Transformações de especificações XMT-O para NCL, com tratamento de eventos, podem ser vistas com detalhes em (COSTA, 2005) .

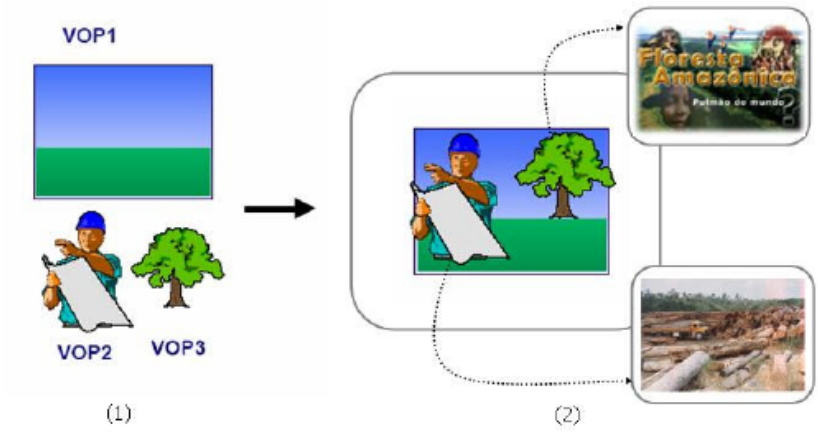


Figura 3.17: Cena em MPEG-4 com sincronização espaço-temporal.

3.6 INFLUÊNCIA DOS TRABALHOS RELACIONADOS

Baseado nestes trabalhos relacionados, algumas de suas características foram apropriadas e adaptadas para especificação do modelo proposto neste trabalho. São elas:

- Facilidade na criação de relacionamentos a partir das composições SMIL.
- Expressividade na criação de relacionamentos a partir dos elos causais NCL.
- Adaptação de conteúdos por meio da avaliação de expressões como em SMIL.
- Suporte a multidispositivos baseado em classes como em NCL.
- Conceito de cenas utilizado em SceneSync e MPEG-4.
- Facilidade na especificação de repetição de composições a partir de propriedades como em SMIL.
- Facilidade na especificação de duração máxima de composições a partir de propriedades como em SMIL.

4 MODELO STorM

De acordo com Antonacci et al. (2000), para que se defina uma linguagem declarativa é necessária a especificação de um modelo conceitual que estruture as entidades e se detalhe a semântica envolvida. Além disso, é interessante a utilização de uma meta-linguagem, como as baseadas em marcação, responsável por facilitar a criação de instâncias do modelo.

Portanto, neste capítulo é apresentada uma visão geral do modelo proposto nesta dissertação, denominado STorM (***S****cene- and* ***T****rack-****or****iented hypermedia* ***M****odel*), aliada a uma breve descrição de suas principais entidades. Ao final do capítulo, essas entidades são representadas sob a forma de elementos da linguagem STorML (STorM ***L****anguage*), concretizada como uma meta-linguagem XML.

Após o levantamento dos requisitos na Seção 2.4, um breve estudo foi realizado para determinar as entidades que estariam alinhadas ao domínio específico de DOOH. O objetivo foi encontrar conceitos e entidades que se aproximassem do vocabulário utilizado pelos profissionais da área de comunicação social que, em grande parte das corporações/instituições, são os responsáveis pela criação de conteúdos DOOH. Portanto, conceitos similares aos utilizados nos softwares de edição audiovisual, como Pro (2005), foram abordados. Não só levou-se em conta esses softwares, como também as próprias ferramentas gráficas usadas na criação de conteúdos DOOH estudas e mostradas na Seção 2.3. Dessa forma, buscou-se facilitar a assimilação dos conceitos com o intuito de favorecer a autoria de instâncias STorM. No entanto, trabalhos futuros precisam ser realizados com os autores de conteúdo na busca de indicadores de legibilidade e eficácia na assimilação e criação de instâncias do Modelo. A Figura 4.1 apresenta as entidades STorM.

STorM utiliza a abstração de cenas e trilhas como pilares para organizar o documento multimídia. As cenas são comparadas aos contextos de NCL (ANTONACCI et al., 2000) , como composições com escopo restrito, com o diferencial de apresentarem semântica de paralelismo incorporada e um nível de abstração maior. Uma cena representa a apresentação em paralelo de uma coleção de trilhas, que por sua vez correspondem à apresentação em sequência de uma coleção de objetos de mídia. Um nó de cena é representado pela entidade *Scene*, a qual não permite aninhamento de outras cenas. Nota-se, assim, que o comportamento de uma apresentação baseada em STorM pode ser comparada ao resul-

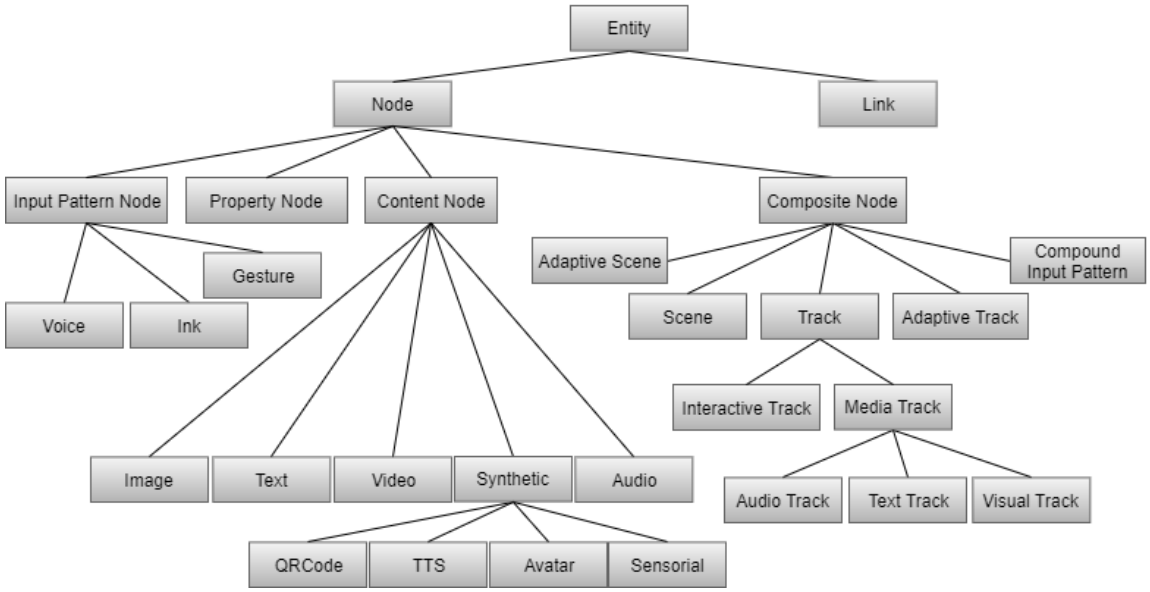


Figura 4.1: Entidades do Modelo STorM

tado da edição de um filme, produzido através de uma sequência de cenas, em que apenas uma cena é exibida por vez.

Cada trilha (*Track*) que compõe uma cena representa uma sequenciação de objetos de mídia (*Content Nodes*[1]). As trilhas são exibidas em uma ou, de forma replicada, mais regiões associadas. A Figura 4.2 ilustra essa aplicação do conceito de cenas e regiões associadas a trilhas, definidas em apresentações de mídia DOOH.

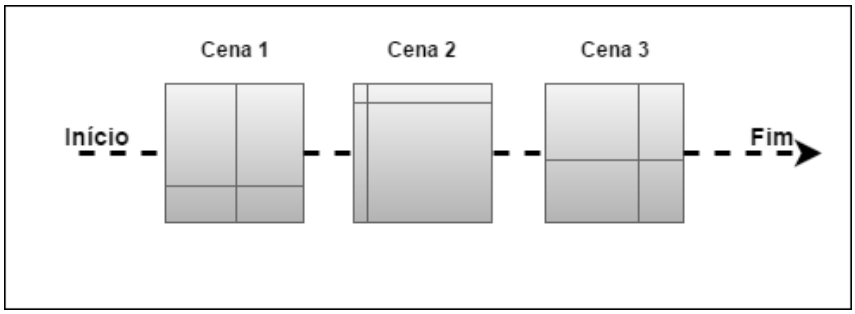


Figura 4.2: Cenas em DOOH: Visão Geral

Conceitualmente, as trilhas têm um comportamento de *timeline* inspirado nas trilhas utilizadas em *softwares* da indústria do audiovisual. Além disso, sua semântica se assemelha à composição <*seq*> utilizada em SMIL. O início das trilhas está relacionado ao início de sua respectiva cena e os objetos de mídia internos às trilhas apresentam um comportamento sequencial, na ordem em que são especificados. Cada trilha possui duração

[1]Entidade herdada de NCL: Nó de mídia, também chamada de mídia ou objeto de mídia, é um nó que representa um objeto em uma mídia qualquer (SOARES; BARBOSA, 2012) .

implícita determinada pela agregação das durações dos objetos de mídia internos. No entanto, quando a duração explícita é definida, sua duração implícita é ignorada. Como observação, ressalta-se a definição distinta do conceito de trilhas de STorM com relação ao modelo NCM mostrado em Soares e Barbosa (2012)[2].

Algumas exceções a essas definições se aplicam a um tipo especial de trilha, as trilhas interativas (*Interactive Track*). Trilhas interativas apresentam um comportamento diferente das trilhas de mídias contínuas (*Audio Track*, *Visual Track* e *Text Track*), as quais contêm objetos de mídia relacionados sequencialmente, de modo predeterminado. No caso das trilhas interativas, os relacionamentos ocorrem por meio do paradigma de eventos de causalidade, igualmente utilizado em NCL (SOARES; BARBOSA, 2012) . Dessa forma, trilhas interativas podem ser comparadas a contextos NCL e qualquer relacionamento explícito em uma trilha interativa deve ser especificado por meio das entidades elo (*Link*) herdadas, neste trabalho, da proposta preliminar do *EDTV Profile* da NCL 3.1 (SOARES G.F. LIMA, 2010) . No entanto, além da utilização de relacionamentos síncronos e assíncronos, a trilha interativa poderá ser utilizada para especificar aplicações autocontidas, denominadas *widgets*. Esses arquivos de *widgets* devem estar em conformidade com a especificação de aplicações empacotadas para o serviço de IPTV definida em (ITU, 2015a) . De forma similar às trilhas interativas, as trilhas de texto também poderão ser utilizadas para referenciar arquivos compostos por fragmentos de textos, cada qual com marcas temporais (*timestamp*) definidas para exibição. Esses arquivos são comumente utilizados na especificação de legendas (e.g. SRT, WebVTT, TTML, etc.).

Destaca-se, ainda, que as trilhas e seus objetos de mídia internos possuem propriedades que podem ser alteradas ou utilizadas em relacionamentos durante a exibição da cena, de modo a trazer maior dinamicidade e controle do conteúdo exibido. Dentre essas propriedades, destacam-se: A região (*region*) em que determinada trilha será exibida; o atraso para que uma mídia seja iniciada (*delay*); o volume de uma trilha ou mídia (*sound level*), dentre outras.

Com relação à personalização (SPYROU; IAKOVIDIS; MYLONAS, 2014) de conteúdo, este trabalho propõe os conceitos de cenas adaptativas (*Adaptive Scene*) e trilhas adaptativas (*Adaptive Track*). Tais entidades são tratadas como uma cena ou trilha que

[2]Em NCM, trilha (*trail*) é um recurso utilizado em sistemas hipermídia para o usuário se orientar no caminhamento entre os contextos aninhados. Portanto, aquelas trilhas são comparadas a um histórico de navegação de contextos e são geralmente denominadas como trilhas de navegação.

irá se adaptar ao contexto do terminal. Portanto, cena adaptativa é uma composição de cenas alternativas, das quais apenas uma será exibida, havendo, nesse caso, uma adaptação do todo de uma cena. Quando se pretende adaptar apenas parte do conteúdo de uma cena e não sua totalidade, utiliza-se a trilha adaptativa. Para auxiliar essas adaptações, os nós de propriedade (*Property Node*) são especificados no modelo para permitir um maior controle dos conteúdos que devem ser exibidos a partir da avaliação dessas propriedades, que representam normalmente propriedades de ambiente.

Além das mídias comumente utilizadas em outros modelos, como áudio, vídeo, texto e imagem, STorM propõe a utilização de mídias sintéticas (*Synthetic*), como especializações dos nós de conteúdo. Nós sintéticos são mídias geradas em tempo de execução com base em linguagens de marcação específicas, as quais podem representar vozes, avatares 3D, efeitos sensoriais e *QR Codes*. Os sintetizadores são os componentes de uma máquina de apresentação STorM responsáveis por interpretarem essas especificações.

Máquinas de apresentação STorM também incluem componentes reconhecedores, responsáveis pela geração de eventos baseados em especificações de padrões de entrada (*Input Pattern*). Padrões de entrada são fragmentos de informações (*node*) que representam um gesto, um comando de voz ou um movimento por meio de caneta. Além de sua utilização com base em uma entrada simples do usuário, o modelo permite a composição desses padrões de entradas (*Compound Input Pattern*), de modo a possibilitar diversas combinações (interações multimodais), como, por exemplo, um gesto em paralelo a uma fala. Desta forma, STorM torna-se mais imersivo e permite a criação de aplicações mais ubíquas, conforme destacado em Guedes et al. (2015).

De forma comparativa, em Guedes et al. (2015) as entradas são definidas por meio de uma entidade de mesmo nível de abstração das mídias. Portanto, os padrões de entrada precisam ser iniciados, como um objeto de mídia, para que a partir de então seja possível identificá-los. Já em STorM os padrões de entrada são associados às trilhas ou às mídias nos relacionamentos (*Link*), sendo, portanto, entidades distintas que não necessitam ser iniciadas. Ou seja, essas entidades sempre estarão associadas a um objeto de primeira classe do modelo e, portanto, a identificação do padrão de entrada ocorrerá quando o objeto associado estiver ocorrendo. Na Capítulo 6 esses relacionamentos serão exemplificados.

Portanto, observa-se que a definição de relacionamentos de sincronismo espacial e tem-

poral neste modelo é separada da definição do conteúdo dos objetos de mídia. Na literatura isso é conhecido como definição baseada na estrutura(*structure-base*), em contraste com a definição embutida no conteúdo dos objetos de mídia, conhecida como baseada no conteúdo da mídia (*media-based*). Esse é um requisito importante para modelos hipermídia, conforme levantado e utilizado em Soares e Barbosa (2012).

4.1 LINGUAGEM STorML

Nesta seção serão abordados os módulos da linguagem STorML e seus respectivos elementos de acordo com a meta-linguagem XML. O XML Schema STorML pode ser visto no Apêndice A. A Figura 4.3 apresenta o diagrama de elementos da linguagem, o qual fornece uma visão geral de como STorML é organizada. Tal segmentação foi adotada apenas para simplificar a descrição da linguagem. Uma modularização mais minuciosa da linguagem com o objetivo de facilitar a criação de perfis da mesma é deixada como trabalho futuro.

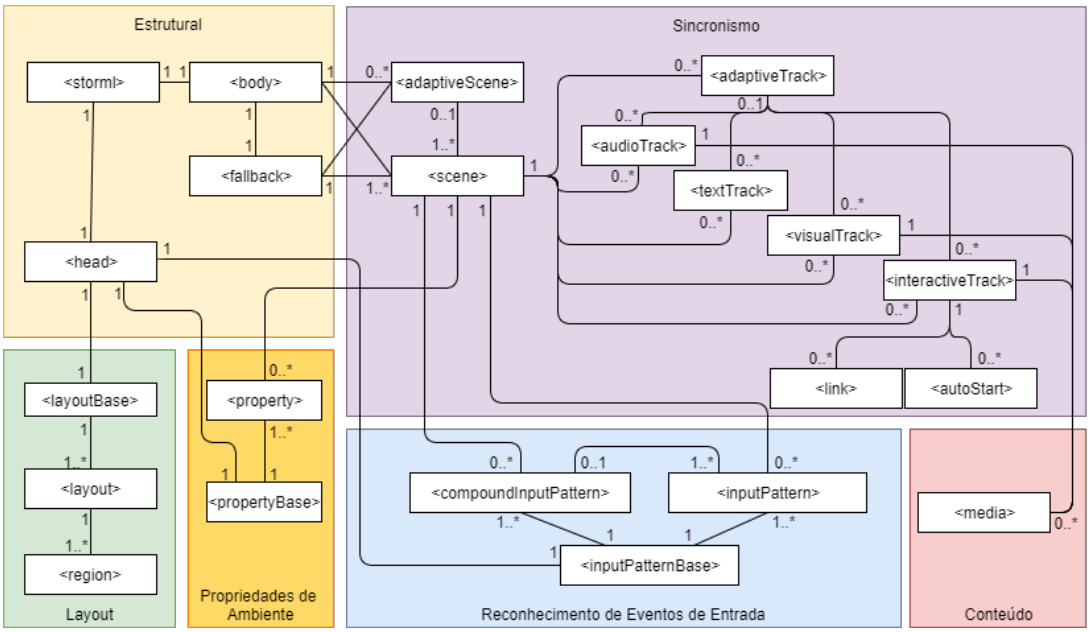


Figura 4.3: Diagrama de elementos da linguagem STorML

4.1.1 MÓDULO ESTRUTURAL

Assim como na maioria das linguagens mostradas no Capítulo 3, a estrutura básica formada pela linguagem STorML é composta pelo elemento raiz <*storml*> e seus elementos filhos <*head*> e <*body*>. O elemento raiz possui como atributos um identificador (*id*)

e a especificação do *namespace* padrão usado (*xmlns*). Como nos demais elementos, o identificador é obrigatório e é formado por uma cadeia de caracteres, devendo seu valor ser único em todo o documento. A Figura 4.4 ilustra a estrutura básica de STorML.

```
<storml id="" xmlns="">
    <head> <!-- Cabeçalho do Documento -->
        <layoutBase>
            <!-- Layouts da Apresentação -->
        </layoutBase>
        <inputPatternBase>
            <!-- Base de Padrões de Entrada Multimodais -->
        </inputPatternBase>
       <propertyBase>
            <!-- Base de propriedades globais do documento -->
       </propertyBase>
    </head>
    <body> <!-- Corpo do Documento -->
        <scene>
            <!-- Primeira Cena da Apresentação -->
        </scene>
        <scene>
            <!-- Segunda Cena da Apresentação -->
        </scene>
        <fallback>
            <!-- Apresentação Alternativa  -->
        </fallback>
    </body>
</storml>
```

Figura 4.4: Estrutura Básica da Linguagem STorML

A Tabela 4.1 lista os elementos presentes neste módulo. A notação para os filhos de elementos é a seguinte: o símbolo | denota uma lista de opções; o símbolo + denota a exigência de pelo menos um elemento filho; o símbolo * denota a exigência por zero ou mais elementos filhos; e o símbolo ? denota a exigência por zero ou um elemento filho.

Elementos	**Atributos**	**Elementos Filhos**
storml	*id*,*xmlns*	(head? | body?)
head		(layoutBase? | inputPatternBase? | propertyBase?)
body	*id*,*presentationMode*	(scene+ | fallback? | adaptiveScene+)
fallback	*id*	(scene+ | adaptiveScene+)
meta	*name*,*content*	empty

Tabela 4.1: Módulo Estrutural

O elemento <*head*> é responsável por conter a base de leiautes (<*layoutBase*>) que serão usados pelas cenas durante a apresentação. Além disso, o elemento <*head*> também contém a base de padrões de entrada que serão usados em relacionamentos condicionados a interações multimodais, existentes no corpo (<*body*>) da apresentação. O elemento <*body*> contém os elementos que descrevem a apresentação, a qual é formada pelo conjunto de cenas e suas respectivas trilhas e mídias. O atributo *presentationMode* de <*body*> é responsável por determinar se a ordem de apresentação das cenas será sequencial (*sequential*) ou aleatória (*random*). Esse recurso é interessante nos casos de uso que abordam a interatividade, os quais utilizam-se de caminhamentos não-lineares entre cenas. Uma vez especificado o modo sequencial, as cenas serão obrigatoriamente exibidas de forma linear, da maneira que estão dispostas no documento, não permitindo elos que propõem o salto entre cenas posicionadas em tempos distintos. Vale ressaltar que, independente do modo de apresentação, a primeira cena a ser exibida será sempre a primeira declarada no documento STorML.

Além dos elementos de composição <*scene*>, o corpo (<*body*>) também contém uma estrutura denominada <*fallback*>, responsável por conter uma ou mais cenas de *backup*, usada pelos terminais em situações de indisponibilidade da rede. O conteúdo de <*fallback*> apresenta a mesma estrutura do elemento <*body*>. Quanto ao elemento <*meta*>, apesar de ser especificado nesse módulo estrutural, ele está contido em todos os outros módulos, sendo responsável pela descrição do documento, de modo a facilitar a posterior busca, recuperação e reúso de elementos por meio de uma ferramenta gráfica.

4.1.2 MÓDULO *LAYOUT*

Esta seção apresenta os elementos responsáveis pela definição de como será a disposição das trilhas presentes em uma cena. A Tabela 4.2 lista os elementos presentes neste módulo.

Elementos	**Atributos**	**Elementos Filhos**
layoutBase		layout+
layout	*id*,*dev*,*dev-arrangement*	(meta* | region+)
region	*id*,*width*,*height*,*left*,*top*,*fit*,*z-index*	empty

Tabela 4.2: Módulo Layout

O elemento <*layoutBase*> contém todos os leiautes usados pela apresentação. Cada

elemento <*layout*> é responsável por especificar as regiões que serão usadas ao longo da apresentação. Cada *layout* está associado a uma classe de dispositivos por meio do atributo *dev*, o qual pode ter o valor ***main*** ou ***secondary(i)*** , onde i é um inteiro maior ou igual a zero, que determina a qual grupo de dispositivos secundários esse *layout* pertence. Caso o atributo *dev* seja omitido, a classe ***main*** será adotada. A associação de bases de *layout* à classes de dispositivos foi herdada de NCL (SOARES; BARBOSA, 2012) . No entanto, diferente de como é tratado em NCL, onde há a divisão entre dispositivos ativos e passivos, em STorML todos os dispositivos (***main*** ou ***secondary***) devem suportar documentos STorML, de forma similar aos grupos do tipo ativo de NCL, que suportam a apresentação de documentos NCL.

Figura 4.5: Múltiplos dispositivos em DOOH

Dispositivos principais (***main***) são terminais responsáveis por receberem o conteúdo, interpretá-los e exibi-los nos *displays* nele instalados. Além disso, eles são os encarregados de repassarem o conteúdo destinado aos dispositivos secundários (***secondary***).

Além da associação aos dispositivos secundários, o *layout* pode ainda ser configurado para diferentes disposições físicas de *displays*, denominadas arranjos físicos ou *videowall*. O atributo *dev-arrangement* é responsável por essa especificação e seu valor é uma tripla composta pela largura (quantidade de dispositivos na horizontal), altura (quantidade de dispositivos na vertical) e razão de aspecto de um dispositivo que compõe a instalação (*LxAxAR*). Como pré-requisito, todos os dispositivos que compõem a instalação necessitam ter a mesma proporção de tela (razão de aspecto). Caso esse atributo não seja utilizado, o valor 1x1x16:9 é assumido como padrão. A Figura 4.6 ilustra um arranjo composto por 6 dispositivos.

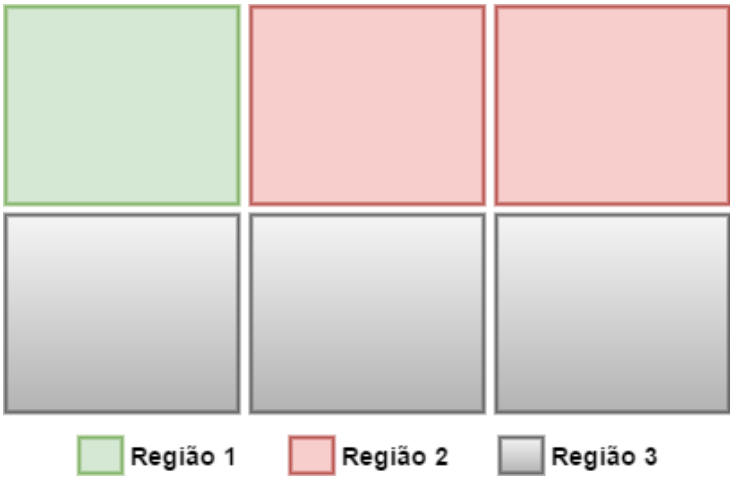


Figura 4.6: Exemplo de arranjo físico (3x2x4:3) com regiões de exemplo demarcadas

O elemento <*region*> delimita o espaço dentro de um *layout* onde será exibida uma trilha. O atributo *left* representa a distância do canto superior esquerdo da região em relação à extremidade esquerda do *display*. O atributo *top* define a distância do canto superior esquerdo da região em relação à extremidade superior do *display*. Caso esses atributos não sejam definidos, os valores padrões (*default*) serão de 0%. Os atributos *width* e *height* determinam a largura e altura da região, respectivamente. Caso esses atributos não sejam definidos, os valores padrões (*default*) serão de 100%. O atributo *fit* é responsável por definir como as mídias serão exibidas na região, de modo semelhante ao que é usado em NCL (SOARES; BARBOSA, 2012) e CSS (W3C, 2012a) . Os valores possíveis são *"fill"*, *"keep"* e *"meet"*. O valor *"fill"* determina o redimensionamento do conteúdo do objeto de mídia para que toque todas as bordas da região, distorcendo-o caso necessário. O valor *"keep"* verifica que, se caso a altura intrínseca ao conteúdo da mídia for menor que o atributo *height* da região, o objeto precisa ser renderizado a partir do topo e ter sua altura restante preenchida com a cor de *background* da trilha associada; caso a altura seja maior, o restante deve ser cortado. Da mesma maneira é feito com a largura. O valor *"meet"* causa o redimensionamento do conteúdo do objeto de mídia mantendo suas proporções até atingir uma das bordas da região. Caso haja um espaço vazio à direita ou abaixo da mídia, a região deve ser preenchida com a cor de *background* da trilha associada.

Por último, o atributo *z-index* representa a ordem da região na pilha de regiões. Uma região de maior valor *z-index* se sobrepõe àquela de menor valor. Todos os valores de posicionamento e tamanho das regiões são sempre representados de forma percentual, e

o cálculo é feito com base na razão de aspecto do dispositivo associado ao *layout*. Essa decisão foi tomada devido a diversidade de resoluções atuais. No caso do arranjo físico, a resolução é obtida pela agregação das razões de aspecto dos dispositivos que compõem a instalação.

4.1.3 MÓDULO DE SINCRONISMO

Esta seção apresenta os elementos responsáveis pelo sincronismo dos objetos de mídias, assim como os elementos usados na interatividade e adaptação de conteúdo baseada em expressões com valores específicos ou personalizáveis. A tabela 4.3 mostra os elementos deste módulo, seus atributos e respectivos elementos filhos.

Em STorML, o sincronismo é definido por meio da utilização de uma sequência de cenas <*scene*>, cada uma sendo uma composição com semântica de paralelismo entre seus componentes: as trilhas. Por sua vez, cada trilha <*track*> é uma composição com semântica de sequenciação incorporada, como as trilhas de mídias contínuas e de legendas (<*visualTrack*>, <*audioTrack*>) e (<*textTrack*>). Além disso, trilhas podem ser interativas (<*interactiveTrack*>), com semântica programável, cuja especificação se dará por meio de relacionamentos causais especificados por elos (<*link*>). As trilhas visuais (<*visualTrack*>) contêm mídias do tipo vídeo e imagem, as trilhas de áudio (<*audioTrack*>) contêm mídias do tipo áudio e as trilhas de texto (<*textTrack*>) contêm mídias do tipo texto de legenda (e.g. SRT, WebVVT...). Já as trilhas interativas (<*interactiveTrack*>) podem conter qualquer tipo de mídia e elos.

A exibição de uma cena é controlada pelos atributos *maxDur* e *repeatCount*. O *maxDur* tem seu valor em segundos e representa a duração máxima da cena. O *repeatCount* representa o número de vezes que essa cena deve se repetir antes de prosseguir com a apresentação. Seu valor é um número inteiro ou a palavra reservada "*indefinite*", que representa a repetição da cena indefinidamente. O atributo *refer* é utilizado quando se deseja reusar uma outra cena presente no corpo do documento. Esse reúso é equivalente ao tipo de instância *"newInstance"* tratado em NCL (SOARES; BARBOSA, 2012) e, nesse caso, a cena será uma cópia totalmente independente da outra, tratando-se de uma nova cena composta pelas cópias independentes das trilhas e mídias internas. Esse mesmo conceito é utilizado para todos os outros elementos da linguagem que fazem uso do atributo *refer*, cujo valor é o identificador (*id*) do elemento que se deseja reusar.

Elementos	Atributos	Elementos Filhos
adaptiveScene	*id*,*maxDur*,*repeatCount*,*refer*	(meta* | scene+)
scene	*id*,*repeatCount*,*refer*,*maxDur*,*expr*	(meta* | visualTrack+ | audioTrack+ | textTrack+ | interactiveTrack+ | adaptiveTrack+ | property+)
adaptiveTrack	*id*,*maxDur*,*refer*,*repeatCount*, *keepTiming*,*startMode*	meta*, (visualTrack | audioTrack | textTrack | interactiveTrack)+
interactiveTrack	*id*,*region*,*delay*,*maxDur*,*refer*, *repeatCount*,*contentType*,*src*, *startMode*,*expr*,*refreshTime*, *bgColor*,*bgImage*,*soundLevel*, *class*	(media* | link+ | meta* | autoStart+)
audioTrack	*id*,*region*,*delay*,*dur*,*refer*, *startMode*,*transition*,*src*, *contentType*,*repeatCount*, *soundLevel*,*class*	(media+ | meta*)
visualTrack	*id*,*region*,*delay*,*dur*,*refer*,*src*, *startMode*,*transition*,*bgColor*, *bgImage*,*repeatCount*,*soundLevel*, *contentType*,*class*	(media+ | meta*)
textTrack	*id*,*region*,*delay*,*dur*,*refer*, *repeatCount*,*contentType*,*src*, *startMode*,*expr*,*bgColor*,*bgImage*, *contentType*,*class*	meta*
link	empty	empty
autoStart	*media*	empty

Tabela 4.3: Módulo de Sincronismo

Assim como utilizado em diversos outros modelos, a adaptação de conteúdo é um recurso importante em linguagens de marcação responsáveis pela sincronização de mídias. Em (SPYROU; IAKOVIDIS; MYLONAS, 2014) , esse conceito é definido como personalização de conteúdos em linguagens hipermídias. NCL e SMIL, por exemplo, fazem uso do elemento <*switch*>, responsável por avaliar qual dos elementos internos será exibido. Basicamente, o primeiro elemento filho que atender a uma determinada regra será o escolhido para ser exibido. Com base no mesmo conceito, porém com um nível de abstração maior, foram definidos em STorML os elementos <*adaptiveScene*> e <*adaptiveTrack*>

com o objetivo de facilitar o entendimento do autor menos especializado.

A avaliação da regra em STorML é realizada pela análise, em tempo de execução, de uma expressão representada pelo atributo *expr*, de forma parecida ao utilizado em SMIL (W3C, 2008) . O primeiro elemento alternativo, que tenha sua expressão avaliada como verdadeira (*true*), será exibido. O valor desta expressão segue a sintaxe BNF (SLONNEGER; KURTZ, 1995) descrita no Apêndice D.1. Vale destacar que os operadores definidos nessa sintaxe dependem dos tipos das variáveis, ou seja, o autor da expressão deve atentar para os tipos de variáveis e os operadores utilizados, de modo a formalizar uma expressão válida.

Portanto, o elemento <*adaptiveScene*> é composto por diversas cenas, das quais apenas a primeira a ter seu valor de expressão avaliado como verdadeiro será exibida. O elemento <*adaptiveTrack*> é tratado da mesma forma, porém, este é composto por trilhas. O elemento filho que não tiver o atributo (*expr*) determinado, será adotado como padrão(*default*), caso nenhuma expressão seja validada. Tanto em uma <*adaptiveScene*> quanto em uma <*adaptiveTrack*>, uma única opção padrão(*default*) pode ser especificada.

O atributo *keepTiming*, do elemento <*adaptiveTrack*>, pode ser definido como *true* ou *false*, sendo esse último o padrão (*default*). O valor *true* especifica que, em caso de reavaliação das expressões durante a apresentação da trilha adaptativa, sua marca temporal atual deve ser mantida, levando a um reposicionamento ("***seek***") na *timeline* da trilha interna que teve sua expressão validada. Ou seja, a nova trilha a ser exibida iniciará a partir do mesmo instante de tempo em que a trilha anterior foi finalizada. O valor *false* significa que, em caso de reavaliação das expressões, a trilha que teve sua expressão validada será apresentada desde seu início. Dessa forma, será possível alternar a exibição das trilhas internas de modo a manter ou não, a sincronia após reavaliação das expressões.

A Figura 4.7 ilustra uma cena adaptativa que contém uma regra de exibição referente à localização do terminal. Essas aplicações são úteis, por exemplo, em terminais localizados em transportes públicos, os quais necessitam de um conteúdo relativo à região onde se encontram. O valor de localização é determinado por meio de uma tripla, composta pela latitude, longitude e o alcance (*range*). Apesar da expressão fazer uso de uma variável de sistema (*system.location*), as especificações destas variáveis não foram foco deste trabalho. Casos de uso de interatividade como esse, podem ser vistos em ITU-GROUP16 (2014). O

```
<adaptiveScene maxDuration="100s">
    <scene expr="system.location=='40.7143528,-74.0059731,100'">
        <visualTrack>
        ...
        </visualTrack>
        ...
    </scene>
    <scene expr="system.location=='41.29431726,-72.90115356,100'">
        <visualTrack>
        ...
        </visualTrack>
        ...
    </scene>
</adaptiveScene>
```

Figura 4.7: Exemplo de uso de Cena Adaptativa

atributo *maxDur* também pode ser usado no elemento *adaptiveScene*, caso o autor queira limitar a duração máxima de uma cena alternativa.

A Figura 4.8 ilustra uma trilha adaptativa que contém uma regra de exibição referente à grupos de terminais. Estes agrupamentos de terminais são muito utilizados em aplicações de DOOH como, por exemplo, em corporações que fazem uso de conteúdos adaptados de acordo com o setor onde o terminal está localizado.

```
<scene>
    ...
    <adaptiveTrack>
        <visualTrack expr="system.Group==01" id="a" region="rg1">
            <media id="img2" .../>
            <media id="img3" .../>
        </visualTrack>
        <visualTrack expr="system.Group==02" id="b" region="rg1">
            <media id="img1" .../>
        </visualTrack>
    </adaptiveTrack>
    ...
</scene>
```

Figura 4.8: Exemplo de uso de Trilha Adaptativa

As trilhas podem estar associadas a uma ou mais regiões de diferentes *layouts*. Cada trilha poderá ser exibida, de forma alternativa ou simultânea, em diferentes regiões. Na exibição simultânea, o conteúdo de uma trilha poderá ser exibida em diferentes regiões ao mesmo tempo. Como exemplo, pode-se ter uma sequência de mídias, as quais o autor deseja exibir tanto em uma determinada região no dispositivo principal, como também na região localizada em um dispositivo secundário. Vale ressaltar que a linguagem permite que sejam exibidas mais de uma trilha para uma mesma região na cena, no entanto, o autor deve estar atento aos problemas que podem ocorrer, caso haja a sobreposição de trilhas em um mesmo instante de tempo, na mesma região. Na Figura 4.9, é ilustrado o

uso de regiões simultâneas por trilhas em uma cena e a Figura 4.10 representa esse cenário em STorML.

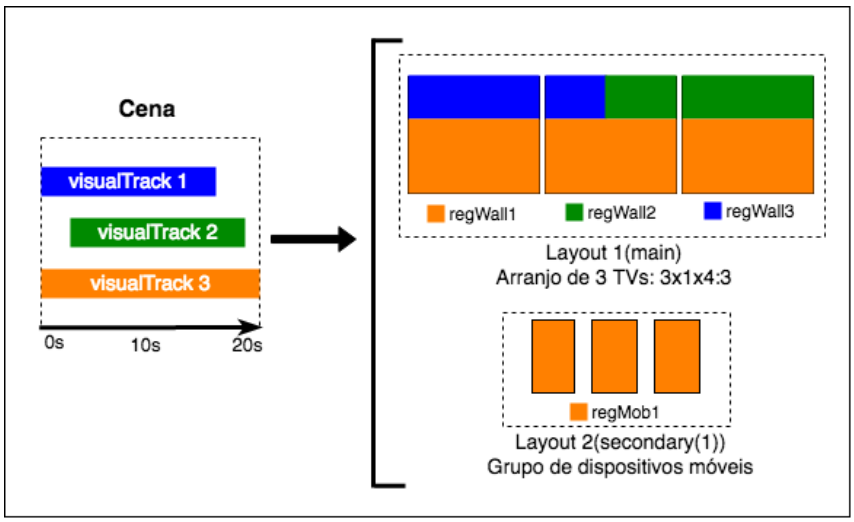


Figura 4.9: Visão espaço-temporal com exibição da cena em regiões concomitantes

```
<storml>
    <head>
        <layoutBase>
            <layout id="layout1" dev="main" dev-arrangement="3x1x4:3">
                <region id="backgrdWall" top="0%" left="0%" width="100%" height="100%" z-index="0"/>
                <region id="regWall3" top="0%" left="0%" width="50%" height="30%" z-index="1"/>
                <region id="regWall2" top="0%" left="50%" width="50%" height="30%" z-index="1"/>
                <region id="regWall1" top="30%" left="0%" width="100%" height="70%" z-index="1"/>
            </layout>
            <layout id="layout2" dev="main">
                <region id="backgrdTV" top="0%" left="0%" width="100%" height="100%" z-index="0"/>
                <region id="regTV" top="0%" left="0%" width="10%" height="10%" z-index="1"/>
            </layout>
        </layoutBase>
    </head>
    <scene id="scene1" maxDur="20s">
        <visualTrack id="backGroundTrack" dur="20s" region="backgrdWall || backgrdTV">
            <media id="img1" dur="20s"/>
        </visualTrack>
        <visualTrack id="visualTrack1" region="regWall3">
            <media id="img1" dur="10s"/>
            <media id="img2" dur="7s"/>
            <media id="img3" dur="3s"/>
        </visualTrack>
        <visualTrack id="visualTrack2" region="regWall2 || regTV">
            <media id="img4" delay="2s" dur="10s"/>
            <media id="vid1" dur="8s"/>
        </visualTrack>
        <audioTrack id="visualTrack3" region="regWall1">
            <media id="aud1" dur="10s"/>
            <media id="aud2" dur="10s"/>
        </audioTrack>
    </scene>
</storml>
```

Figura 4.10: Instância do Modelo com exibição de cena em regiões concomitantes

Da mesma forma que uma trilha pode ser exibida em regiões distintas, ao mesmo tempo, por meio do operador "&&", a linguagem permite a utilização de regiões alternativas por meio do operador "||". Nesse caso, a análise de qual região uma trilha será exibida, fica a cargo do terminal que receber o conteúdo. O operador "||" terá um compor-

tamento parecido com as avaliações de curto circuito realizadas em algumas linguagens imperativas. A primeira região da lista de opções que estiver disponível no terminal, será a escolhida para a trilha ser exibida, o restante das regiões serão ignoradas. Esse é um recurso interessante para facilitar a autoria de conteúdos STorML por autores que não sabem qual é o dispositivo principal (*main*) localizado no destino. Desta forma, o autor pode criar diversos *layouts* para diferentes dispositivos, de modo a possibilitar adaptações diversas, como uma espécie de responsividade. Para esse controle, o valor do atributo *region* segue uma sintaxe BNF (SLONNEGER; KURTZ, 1995) descrita no Apêndice D.2.

A linguagem permite a especificação de trilhas de mídias contínuas, representadas pela <*audioTrack*>, <*visualTrack*> e <*textTrack*>, e de trilhas interativas, definidas pelo elemento <*interactiveTrack*>. A <*visualTrack*> é a trilha responsável por conter apenas elementos visuais, como vídeo, texto e imagem. Além dos atributos *id*, *repeatCount*, *region* e *refer* já explicados anteriormente, as trilhas de mídias contínuas apresentam os atributos *delay*, *dur*, *transition*, *soundLevel*, *startMode*, *src*, *bgColor* e *bgImage*. Quanto ao *refer*, esse pode ser usado tanto em trilhas de mídias contínuas, como também em trilhas interativas (<*interactiveTrack*>). No entanto, em trilhas interativas seu uso não é trivial, sendo recomendado apenas quando essas tiverem elos autocontidos, ou seja, quando não possuírem elos que envolvam trilhas da mesma cena de origem.

O atributo *delay* determina o período de retardo, em segundos, para o início da trilha em relação ao início da cena. O atributo *dur* determina a duração, em segundos, que a trilha deve ser exibida, diferente do *maxDur* especificado no elemento <*scene*>, que apenas limita a duração de um cena para um melhor controle temporal de toda a apresentação. Portanto, seu valor é um decimal (p.ex, 10.5s) ou a palavra reservada "indefinite" que representa a duração máxima. No elemento <*scene*>, o atributo *dur* não faria sentido, pois a duração da cena está diretamente relacionada à trilha de maior duração, sendo possível, portanto, apenas limitar sua duração, assim como nos elementos <*adaptiveScene*> e <*adaptiveTrack*>, já mencionados. Da mesma forma, um atributo *maxDur* não faria sentido ser usado na trilha, pois sua duração é determinada pelo tempo total dos elementos de mídias internos (duração implícita) ou por um tempo específico determinado pelo autor (duração explícita). A duração explícita é útil, por exemplo, quando o autor tem interesse em sincronizar trilhas distintas de uma mesma cena.

O atributo *transition* é responsável por especificar os efeitos de transição que as mídias internas às trilhas de mídias contínuas sofrerão quando trocadas. A definição de seus possíveis valores não foram foco desta dissertação. Como exemplo, podemos citar os efeitos de *fade in* e *fade out*, comumente usados em *softwares* de edição audiovisual. O atributo *bgColor* e *bgImage* podem ser usados para definir a cor ou imagem de fundo da trilha, respectivamente. O atributo *soundLevel* é usado para o controle de volume de todos os objetos de mídias internos à trilha, seu valor varia entre "0" e "1", sendo "0" o valor mínimo (*mute*) e 1 o valor máximo. Esse atributo foi inserido devido a necessidade de se reproduzir conteúdos em ambientes que não são propícios a ruídos, conforme requisito destacado em Dupin (2011). Esse controle, realizado no lado da autoria, fornece ao autor uma autonomia, pois independe de como o dispositivo terminal está configurado. O atributo *src* é utilizado apenas, quando se pretende especificar uma única mídia na trilha, ao invés de se declarar cada elemento de mídia (<*media*> interno). Consequentemente, o atributo *contentType* deve definir o tipo da mídia referenciada em *src*.

Dependendo da apresentação, podem existir trilhas que necessitam ser iniciadas em momento posterior, ou seja, de forma não sincronizada às outras trilhas definidas na cena, as quais, por padrão, são iniciadas em paralelo conforme citado anteriormente. Essa diferenciação é importante para casos em que o autor deseja que uma trilha inicie a partir de um evento posterior. Para esse controle, o atributo *startMode* foi criado e os valores possíveis são: *Synced* e *Deferred*. O modo *Synced* determina que a trilha será iniciada no mesmo instante ao da cena, exceto quando especificado um *delay*, sendo esse o modo padrão caso o autor omita o atributo *startMode*. O modo *Deferred* determina que a trilha deverá aguardar uma ação posterior. Essa ação pode ser especificada por meio de um elo a partir de alguma trilha interativa, para que a trilha *deferred* seja iniciada no momento esperado.

A trilha <*audioTrack*> é responsável por conter apenas objetos do tipo áudio. Essa especialização de trilha é útil quando se deseja executar mais de uma trilha em um mesmo instante de tempo na cena, podendo, nesse caso, gerar efeitos de mixagem, recurso muito utilizado por autores de conteúdos audiovisuais. Além disso, possibilita a definição de conteúdos para audiodescrição e multi-idiomas. Os atributos de uma <*audioTrack*> seguem os mesmos conceitos estabelecidos para a <*visualTrack*>. Importante ressaltar que, apesar de parecer incoerente a definição de uma região (*region*) para uma trilha de áudio,

essa associação é necessária para determinar em qual dispositivo essa trilha deverá ser executada. Caso esse atributo seja omitido, o dispositivo principal (*main*) será o escolhido para a apresentação da trilha de áudio.

A trilha <*textTrack*> é responsável por conter apenas objetos de mídia do tipo texto. Além disso, essa trilha permite a referenciação de apenas uma mídia texto autocontida, ou seja, trechos de texto com marcas de tempo (*timestamp*) para exibição definidas, comumente utilizadas em documentos de legenda (e.g. SRT, WebVTT,TTML). Portanto, o arquivo de legenda deve ser especificado no atributo *src* e seu respectivo tipo em *contentType*. Em HTML5+*JavaScript* há um elemento parecido, utilizado para o controle de exibição de legendas na *Web* (W3C, 2015) .

A trilha <*interactiveTrack*> é a trilha responsável por definir toda a interatividade da apresentação STorML. A simplicidade do perfil NCL *Raw Profile* (SOARES et al., 2010) foi utilizada como referência em sua especificação. Portanto, além de tratar conteúdos autocontidos denominados *widgets*, essa trilha também permite especificar relacionamentos causais na ordem de N:M, sejam eles temporais ou espaciais, síncronos ou assíncronos. Os atributos de <*interactiveTrack*>, comuns aos das trilhas de mídias contínuas, seguem os mesmos conceitos. No entanto, o atributo *refreshTime* é acrescido. O *contentType* será usado apenas para conteúdos autocontidos e especifica qual o seu tipo, são eles: "*application/x-itu-iptv-widget*", "*application/x-ncl*" ou "*text/html*".

O tipo "*application/x-itu-iptv-widget*" foi baseado na recomendação ITU-T H.765 (ITU, 2015a) . Essa recomendação descreve serviços de *widgets* para IPTV que também podem ser utilizados no modelo tratado nesta dissertação. As tecnologias abordadas por essa recomendação são do tipo NCL(ITU-T H.761), LIME (ITU-T H.762), HTML, CSS(ITU-T H.763.1), dentre outros. *Widgets* são aplicações leves que detém uma interface gráfica de usuário (GUI) simples e de fácil acesso, que normalmente oferecem uma única funcionalidade, como calendário, agregadores de notícias (RSS), informações climáticas, relógio etc. Associado a esse tipo de trilha interativa, os atributos *src* e *refreshTime* são importantes. O *src* é obrigatório e especifica o caminho para o arquivo de conteúdo interativo. A extensão recomendada pela H.765 para esse arquivo é a ".iwp", sendo esse um arquivo empacotado composto por arquivos de configuração e pela aplicação. O atributo *refreshTime* é responsável por informar ao formatador de conteúdo, a periodicidade de recarregamento do *widget*, útil apenas em *widgets* de conteúdos passivos, como agregadores de notícias

(RSS).

Caso o tipo seja omitido, a trilha interativa será composta pelos próprios elementos STorML, usados, nesse caso, para especificar a interatividade no modelo através do uso de relacionamentos causais. Desse modo, permite-se criar relacionamentos tanto entre objetos de mídias internos a uma trilha interativa, como também destes com trilhas de mídias contínuas ou mesmo com objetos de mídias pertencentes a essas trilhas. A restrição de escopo para referências nos relacionamentos é delimitada pela cena na qual a trilha interativa se encontra. O elemento <*autoStart*> se assemelha ao elemento <*port*> de NCL, porém, com a semântica restrita a apenas informar quais mídias iniciarão juntamente com a respectiva trilha interativa.

Alguns novos recursos foram inseridos nos elos para possibilitar a utilização de âncoras temporais e espaciais sem o uso de elementos XML explícitos para tal, como era realizado em NCL por meio de suas interfaces declarativas (<*area*>). Para isso, houve uma sobrecarga na especificação dos elos textuais de modo a facilitar a autoria, sem perder a expressividade da linguagem. Portanto, STorML possibilita a associação de ações a instantes de tempo específicos por meio do uso de parênteses, como, por exemplo: "*onInstant idRefMedia(5s) then...*". Além das âncoras temporais, as âncoras espaciais também são possíveis através da definição dos limites entre colchetes, como, por exemplo: "*onSelection idRefMedia[0,0,10,10] then...*", sendo os valores internos definidos de maneira percentual na seguinte ordem: *top*, *left*, *width* e *height*. A Tabela 4.4 elenca os papéis (*role*) de condição dos elos e sua associação com os elementos STorML. O papel *onRecognition* foi inserido com base no trabalho que especifica o uso de interações multimodais em NCL, tratado em Guedes et al. (2015) e que será visto com mais detalhes no módulo de reconhecimento de eventos de entrada. Além disso, o papel *onChange* foi utilizado em substituição aos papeis de *onBeginAttribution* e *onEndAttribution* utilizados em NCL, de modo a elevar o nível de abstração e facilitar, em tese, o entendimento pelo autor de conteúdo com perfil não programador. Portanto, por meio dessa condição o autor poderá disparar ações no momento em que houver a mudança de valor de uma propriedade, seja essa propriedade declarada de forma explícita por meio da base de propriedades do documento STorML, ou propriedades das trilhas ou mídias previamente definidas e acessadas por meio da referência ao objeto seguido de sua propriedade (*"IdRef.property"*). Essas propriedades que podem ser utilizadas na condição *onChange* são elencadas na tabela 4.5.

Condição	Elemento Alvo STorML (Target)
onEnd	<adaptiveTrack>,<textTrack>, <audioTrack>,<visualTrack>, <interactiveTrack>,<media>
onBegin	<adaptiveTrack>,<textTrack>, <audioTrack>,<visualTrack>, <interactiveTrack>,<media>
onSelection	<adaptiveTrack>,<textTrack>, <audioTrack>,<visualTrack>, <interactiveTrack>,<media>
onPause	<adaptiveTrack>,<textTrack>, <audioTrack>,<visualTrack>, <interactiveTrack>,<media>
onResume	<adaptiveTrack>,<textTrack>, <audioTrack>,<visualTrack>, <interactiveTrack>,<media>
onChange	<property>, *"IdRef.property"*
onRecognition	<inputPattern>, <compoundInputPattern>
onInstant	<adaptiveTrack>,<textTrack>, <audioTrack>,<visualTrack>, <interactiveTrack>,<media>

Tabela 4.4: Papéis de Condição e os respectivos elementos associáveis na Linguagem STorML

A Tabela 4.5 exibe os papéis (*role*) de ação e sua associação com os componentes do modelo definido neste estudo. Além dos papéis de ação herdados de NCL em (SOARES; BARBOSA, 2012) e da proposta preliminar de versão 3.1 da NCL, este modelo propõe um novo papel com o objetivo de fornecer ao autor um maior controle do conteúdo presente na <*audioTrack*>,<*visualTrack*> e <*textTrack*>. Trata-se do papel de ação **"*seek*"**.

A ação **"*seek*"** é utilizada para o caminhamento na trilha. Seu valor pode ser definido em segundos, sendo possível determinar se o deslocamento na *timeline* será de avanço (+) ou retrocesso (-). Na ausência de sinal, o tempo é considerado absoluto, isto é, haverá o reposicionamento para o tempo determinado na *timeline* da trilha. Outra opção de parâmetro para o **"*seek*"** é a utilização de palavras reservadas que irão definir qual a semântica de posicionamento, são elas: ***next***, ***prev***, ***first*** e ***last***. Naturalmente, a ação com ***next*** exibe a mídia seguinte à atual, a ação com ***prev*** exibe a mídia anterior, com ***first*** exibe a primeira mídia da trilha e ***last*** a última mídia da trilha.

A ação **"*set*"** é utilizada para alterar valores de alguns atributos dos elementos STorML.

Ação	**Alvo (Target)**	
	Elemento STorML	**Propriedades**
pause	<audioTrack>,<visualTrack>, <interactiveTrack>,<media> <adaptiveTrack>,<textTrack>	empty
resume	<audioTrack>,<visualTrack>, <interactiveTrack>,<media> <adaptiveTrack>,<textTrack>	empty
start	<audioTrack>,<visualTrack>, <interactiveTrack>,<media>, <adaptiveTrack>,<textTrack>, <scene>	empty
stop	<audioTrack>,<visualTrack>, <interactiveTrack>,<media>, <adaptiveTrack>,<textTrack>, <scene>	empty
seek	<audioTrack>,<visualTrack>, <textTrack>	empty
set	<audioTrack>,<visualTrack>, <interactiveTrack>,<media>, <adaptiveTrack>,<textTrack>, <scene>,<property>	*dur*,*delay*,*repeatCount*, *maxDur*,*soundLevel*,*region*, *top*,*left*,*width*,*height*, *value*

Tabela 4.5: Papéis de ação e os respectivos elementos associáveis e propriedades customizáveis em STorML

Para isso, alguns atributos foram pré-definidos como propriedades e, portanto, customizáveis ao longo do documento STorML. Em NCL, por exemplo, para que um atributo de objeto de mídia seja customizável, esse deve ser declarado explicitamente como uma propriedade (<*property*>). Em STorML, os elementos customizáveis podem ser vistos na tabela 4.5 na coluna de propriedades. Destaca-se entre essas propriedades, a propriedade *region* da trilha. Por meio da ação *set* o autor de conteúdo poderá mudar, em tempo de execução, a região em que uma trilha está sendo exibida ou que será exibida. Diferente de NCL, na qual as regiões são utilizadas apenas no início da apresentação e toda a customização espacial futura depende das propriedades (<*property*>) de posicionamento (*top* e *left*) de cada objeto de mídia, em STorML as regiões são persistidas ao longo da apresentação, podendo ser utilizadas por meio de seus identificadores, para definir, de forma direta, a região onde uma trilha será exibida (p.ex: *"set region='idRefRegion"'*). Este é um recurso muito útil em interações multimodais, conforme será visto no Capítulo 6.

Com base nas listas de condições e ações mostradas, é possível definir diversos elos como, por exemplo, iniciar (***start***) uma trilha após a seleção (***onSelection***) de uma mídia. Outro exemplo seria a definição do fim de uma cena por meio da ação **"*stop*"**, caso o usuário selecione uma determinada mídia (***onSelection***). Neste caso, ocorrerá a exibição da próxima cena se o modo da apresentação estiver definido como sequencial (*sequential*), caso contrário (*random*) a próxima cena será aleatoriamente escolhida. No modo *random* o autor pode, também, fazer com que o elo inicie (***start***) uma cena específica, esse recurso não é possível no modo *sequential*. Esse caminhamento não linear é muito útil, por exemplo, em apresentações baseadas em objetos de aprendizagem, foco do trabalho Busson et al. (2016). Dessa forma, esse caminhamento não linear permite, também, o desenvolvimento de conteúdos destinados aos treinamentos corporativos. A Figura 4.11 exibe a máquina de estados da cena, conforme as ações mostradas na Tabela 4.5.

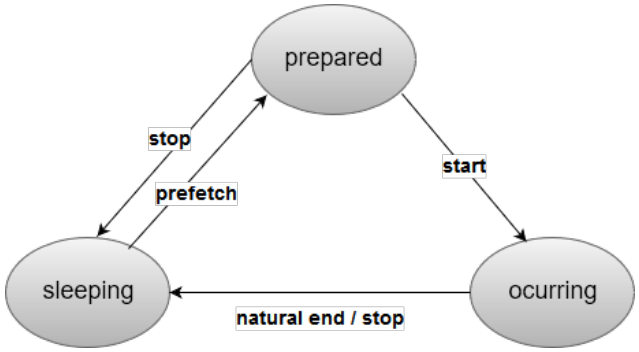


Figura 4.11: Máquina de Estados de uma Cena

Nessa Figura 4.11, observa-se, além das ações de "***start***" e "***stop***" que podem ser controladas por elos, as ações de "***prefetch***" e "***natural end***". Essas duas últimas são ações controladas exclusivamente pelo *player*, sendo a primeira delas responsável pelo carregamento e preparação de toda a cena. A partir do momento em que o estado da cena estiver em ***prepared***, essa poderá ser exibida pelo *player*, pois todos os recursos estarão alocados de forma a garantir QoS. A ação de "***natural end***" é realizada quando a duração da cena, limitada por sua trilha de maior duração, chega ao fim. Caso a duração máxima seja alcançada, a ação de "***natural end***" também será disparada pelo *player*. Esses mesmos conceitos são refletidos nas máquinas de estados da trilha e da mídia, conforme serão vistas mais a frente. Uma apresentação de modo sequencial (*sequential*) fará com que o *player* prepare (***prefetch***) a próxima cena antes de exibi-lá. No entanto, no modo

aleatório (*random*) a próxima cena pode não ser possível identificar, não garantindo, nesses casos, um nível adequado de QoS pelo *player*.

Importante ressaltar que a ação de "***start***" em um cena fará com que esta seja iniciada e automaticamente a cena anterior e todas suas trilhas sejam finalizadas. Ou seja, a trilha interativa capaz de criar os elos pertence à sua própria cena e qualquer transição no estado dessa cena causará também a transição no estado de todas suas trilhas e respectivas mídias para ***sleeping***. Observa-se também que a ação de "***pause***" não é abordada na máquina de estados da cena, pois apenas uma cena é executada por vez e, portanto, a ação de "***pause***" acarretaria a paralisação de toda cena, sendo impossível a geração de um evento que pudesse resultar em sua retomada (***resume***).

A Figura 4.12 exibe a máquina de estados da trilha com a nova ação **"*seek*"**, a qual só pode ter efeito caso a trilha esteja no estado ***ocurring***. O tempo corrido da trilha, equivalente a sua duração, determina o tempo em que sua máquina de estados ficará em ***ocurring***, independente de quantas e quais ações de **"*seek*"** sejam realizadas. A ação de **"*pause*"** leva a trilha ao estado de ***prepared***, pois todos os recursos necessários para sua execução continuam alocados, ou aguardando sua retomada (***resume***) ou sua finalização (***stop***). Da mesma forma, se a trilha estiver ocorrendo e uma ação de **"*stop*"** for executada, a trilha retorna ao estado de ***sleeping***, liberando os recursos alocados.

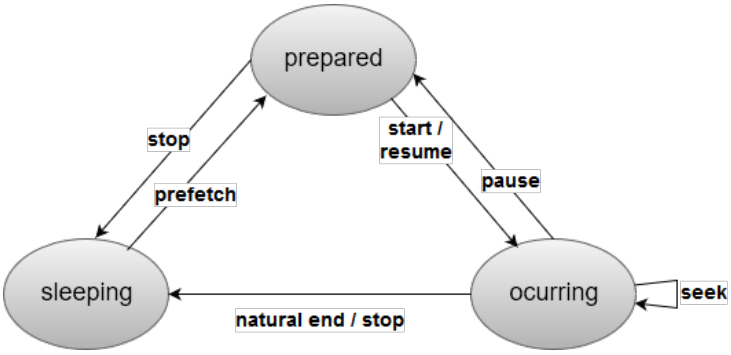


Figura 4.12: Máquina de Estados de uma Trilha

No que diz respeito as trilhas de mídias contínuas e comparando-as com contextos NCL, os quais necessitam de portas de entrada para as mídias serem referenciadas, na trilha de mídia contínua qualquer mídia pode ser referenciada por elos em trilhas interativas da mesma cena. Dessa forma, o controle de apresentação da trilha pode ser feito sobre a trilha como um todo (***start***, ***stop***, ***pause***, ***resume*** e ***seek***); ou pode ser feito individualmente sobre as mídias que a compõem (***start***, ***stop***, ***pause***, ***resume***). Nas ações sobre a

trilha, as máquinas de estados das mídias internas envolvidas também sofrerão transições, no entanto, nas ações sobre as mídias, a máquina de estados da trilha continuará no estado de ***ocurring***.

Portanto, caso o **"*seek*"** reposicione o ponteiro de execução para a mesma mídia que já estava ocorrendo, nenhuma mudança de estado da mídia ocorrerá. No entanto, caso o reposicionamento do ponteiro faça com que uma nova mídia seja exibida, o *player* deverá ser configurado para a ordem de execução dos eventos de **"*stop*"** da mídia que estava sendo apresentada no momento e o **"*start*"** da mídia que iniciará a execução. Ou seja, haverá transição de estados e esse controle será de responsabilidade do *player*, devendo os implementadores estar atentos aos comportamentos esperados.

Além disso, caso haja algum elo associado (***onBegin***, ***onEnd***...) aos elementos de mídias que estão na trilha entre o objeto de mídia que estava sendo exibido (***ocurring***) e o objeto de mídia que passou a ser exibido, o elo não será acionado. Isto ocorre pois a ação de **"*seek*"** "pulou" a execução dos objetos de mídia que estavam entre o objeto que sofreu a ação de **"*stop*"** e o novo objeto que está sendo exibido. Uma forma de efetuar esse controle, é alertar ao usuário através de uma ferramenta gráfica de criação de conteúdo, sobre a existência de elos que possam ser afetados pelas ações **"*seek*"**, caso essas existam em uma trilha interativa. A Figura 4.13 exibe a máquina de estados da mídia.

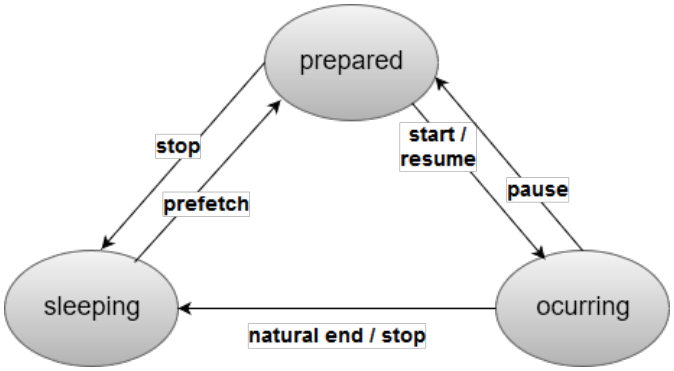


Figura 4.13: Máquina de Estados de uma Mídia

A Figura 4.14 ilustra a utilização de elos conforme a proposta preliminar NCL 3.1 (SOARES G.F. LIMA, 2010) , com o acréscimo do tipo de ação ***"seek"*** em conjunto com o componente trilha visual definido nesse modelo. A nova sintaxe BNF para descrição desse elo (<*link*>), pode ser vista no apêndice D em D.3. Esse exemplo ilustra, de forma resumida, uma aplicação de *slideshow* interativo, na qual a seleção de mídias definidas dentro de uma trilha interativa acarretará a mudança da imagem que está sendo exibida pela

trilha visual identificada pelo *id* ***images***. A seleção da mídia identificada como ***btnPrev*** exibirá a imagem anterior e a seleção da mídia identificada por ***btnNext*** exibirá a próxima imagem da trilha.

```
<body>
    <scene id="scene1">
        <visualTrack id="images" region="slideShowInterativo">
            <media type="image" id="a" dur="10s"/>
            <media type="image" id="b" dur="10s"/>
            <media type="image" id="c" dur="10s"/>
            <media type="image" id="d" dur="10s"/>
        </visualTrack>
        <interactiveTrack id="trilhaInterativa" region="selectionImages">
            <autoStart id="entry2" media="btnPrev" />
            <autoStart id="entry3" media="btnNext" />
            <media id="btnPrev" type="image" top="0%" left="25%" height="100%" width="25%"/>
            <media id="btnNext" type="image" top="0%" left="50%" height="100%" width="25%"/>
            <link id="1btn1ImgY">onSelection btnPrev then seek prev images</link>
            <link id="1btn1ImgZ">onSelection btnNext then seek next images</link>
        </interactiveTrack>
    </scene>
</body>
```

Figura 4.14: Exemplo de uso do tipo de ação ***"seek"*** na Linguagem STorML

De modo comparativo, para uma representação de *slideshow* interativo totalmente em NCL, seria necessário a criação de diversos elos temporais incorporando uma complexidade à autoria. Em Soares Neto (2010) é mostrado um exemplo de *slideshow* interativo em NCL que faz uso de *templates* baseados em sua linguagem TAL. Com o uso de *templates*, a complexidade na criação de elos também é reduzida, pois agiliza o desenvolvimento desses conteúdos que seguem um determinado padrão de relacionamento.

4.1.4 MÓDULO DE CONTEÚDO

Esta seção detalha os objetos de mídias que irão compor as trilhas (<*visualTrack*>, <*audioTrack*> e <*interactiveTrack*>). Esses objetos são representados pelos elementos da Tabela 4.6.

Elementos	**Atributos**	**Elementos Filhos**
media	*id*,*src*,*type*,*dur*,*delay*,*refer*,*top*,*left*,*width*,*height*, *clipBegin*,*clipEnd*,*opacity*,*rotate*	empty

Tabela 4.6: Módulo de Conteúdo

O elemento <*media*> tem como atributos: O identificador; o *src* que especifica o URI do conteúdo do objeto; o atributo *type* especifica qual formato *MIME Type* da mídia. Para sua especificação, foram pesquisados padrões adotados atualmente que definem os

tipos de mídia utilizados nas tecnologias de DOOH. Como resultado, foi encontrado uma especificação criada por POPAI (*Point of Purchase Advertising International*) (POPAI, 2016) , uma associação comercial internacional, que desenvolveu um padrão de formatos recomendados para aplicações de sinalização digital com o objetivo de aumentar a interoperabilidade desses serviços. Esse documento foi denominado *Screen-Media Formats* (POPAI, 2009) . Com base nesse documento e nos novos recursos inseridos, os formatos presentes na Tabela 4.7 foram propostos para o suporte.

Extensão do Arquivo	**Tipo MIME**
.txt	text/plain
.htm, .html, .xhtml	text/html
.jpg, .jpeg	image/jpeg
.png	image/png
.svg	image/svg+xml
.mpeg, .mpg, .mp3	audio/mpeg
.mpg, .mpeg	video/mpeg
.vob	video/x-ms-vob
.mp4	video/mp4
.m4v	video/x-m4v
.mkv	video/x-matroska
.mk4	application/octet-stream
.ogm	application/ogg
.wmv	video/x-ms-wmv
.mjpg	video/x-motion-jpeg
.divx	video/divx
.xml	image/x-qrcode+xml[3]
.ssml	application/ssml+xml
.xml	application/x-bml+xml
.xml	application/x-sedl+xml
.srt	text/srt
.vtt	text/vtt
.ttml	application/xml+ttml

Tabela 4.7: Tipos MIME propostos

[3]Tipo MIME QRCode criado para auxiliar STorML

O atributo *dur* especifica a duração da mídia, caso esse atributo não seja informado sua duração implícita é considerada. O atributo *delay*, assim como na trilha, trata o tempo de atraso para que a mídia seja iniciada. O atributo *opacity* é utilizado para especificar o grau de transparência da mídia, seu valor é definido entre 0 a 1, onde 0 é a transparência máxima e 1 a mídia original, sendo esse último, o valor padrão (*default*). O atributo *rotate* define o grau de rotação da mídia e seu valor é definido entre *"-180deg"* e *"180deg"*, o sinal positivo representa a rotação no sentido horário e em caso negativo no sentido anti-horário, de modo similar ao utilizado em CSS (W3C, 2012a) . O atributo *refer* segue o mesmo conceito dos outros elementos da linguagem.

Para atender as múltiplas disposições espaciais das mídias presentes nas trilhas interativas, os atributos de posicionamento (*left* e *top*) e dimensionamento (*width* e *height*) foram também utilizados como propriedades dos objetos de mídias. Todos esses valores são relativos à região a qual a trilha pai está sendo exibida. Caso esses valores sejam omitidos, os valores de 0% para *left* e *top* e 100% para *width* e *height* serão adotados como padrão. O atributo *clipBegin* e *clipEnd* são utilizados para efetuar o recorte de uma mídia contínua sem que o usuário necessite fazer uma edição externa. Esse recorte é responsabilidade do *player*. O *clipBegin* especifica o tempo de início que será levado em conta na hora da exibição e o *clipEnd* o tempo de fim. O atributo *soundLevel* segue o mesmo conceito definido na trilha, porém, aplicado apenas ao objeto de mídia que o utiliza. Alguns desses atributos são utilizáveis apenas em mídias visuais.

O elemento *<media>* do tipo *image/qrcode+xml* foi inserido na linguagem devido sua utilização constante em DOOH (ITU-GROUP16, 2014) . QR Code (Quick Response Code) (SOON, 2008) é um código de barras bidimensional que atualmente pode ser lido, com facilidade, por diversos dispositivos móveis. O aumento de sua utilização está diretamente relacionada às informações que podem ser convertidas atualmente, como um texto (interativo), um endereço URI, um número de telefone, uma localização georreferenciada, um e-mail, um contato ou um SMS. Baseado nisso, foram pesquisados alguns padrões de codificação na camada de aplicação para tratar essas informações. No entanto, essa padronização ainda não foi elaborada por um órgão reconhecido internacionalmente, sendo realizada de forma independente por alguns projetos em específico. Nessa linguagem, o projeto de código aberto ZXing (MACKINTOSH et al., 2012) foi o escolhido devido sua maior utilização por parte dos aplicativos atuais.

Entretanto, apesar do projeto ZXing fornecer uma lista de tipos de código QR (ZXing..., 2016) , a utilização do conteúdo textual diretamente na linguagem contraria a regra seguida pelo modelo em que apenas a estrutura da apresentação deve ser especificada. Para solucionar essa questão, além de facilitar a criação e a interoperabilidade destas informações de QR Code, um padrão em XML foi criado para conter essas informações. Portanto, um arquivo XML deve ser referenciado no atributo *src* do elemento <*media*>. No apêndice C são demonstrados diversos casos de uso desta especificação e no apêndice B é mostrado seu XML Schema. Na Figura 4.15 pode-se observar uma exemplo de QR Code para conexão em rede sem fio utilizando-se deste padrão XML. A informação textual representada é a sequência **"WIFI:S:MorenoMacMini;T:WPA;P:@123Mac;H:false;"**. Essa é uma especificação ZXing reconhecida por leitores de *QR Code* em dispositivos *android* (ZXing..., 2016) .

```
<!-- Informações para conexão WI-FI via QR Code -->
<!-- Cor do QR Code: Preto, Cor de Fundo: Branco -->
<qrCode color="black" backgroundColor="white">
    <wifi id="wifiMorenoMac">
        <s>MorenoMacMini</s> <!-- SSID da Rede -->
        <t>WPA</t> <!-- Algoritmo de Criptografia -->
        <p>@123Mac</p> <!-- Senha -->
        <h>false</h> <!-- Campo opcional: True se o SSID está oculto. -->
    </wifi>
</qrCode>
```

Figura 4.15: Exemplo de arquivo XML com informações QR Code para conexão em rede sem fio

Com o XML definido, um servidor responsável por tratar o conteúdo antes de enviá-lo aos terminais poderá gerar a imagem *QR Code* correspondente às informações passadas. Esse servidor pode ser o mesmo responsável pela renderização dos conteúdos que serão tratados na Seção 6.2.1. Outra possibilidade é o terminal, que receber o conteúdo, conter um sintetizador específico para esses tipos de mídias, de modo a gerar a imagem de maneira dinâmica. Para esse XML, a imagem 4.16 exibe o *QR Code* correspondente.

Além das mídias comuns (áudio, vídeo, imagem etc.) e do *QR Code*, as mídias sintéticas especificadas em SSML[4], BML[5] e SEDL[6], são de grande importância para o domínio de DOOH. Isto porque efeitos sensoriais fazem uso destas mídias para aumentar as múltiplas formas de comunicação com a audiência. SSML (*Speech Synthesis Markup Language*)

[4]<http://www.w3.org/TR/speech-synthesis11/>
[5]http://www.mindmakers.org/projects/bml-1-0/wiki
[6]http://mpeg.chiariglione.org/standards/mpeg-v/sensoryinformation

Figura 4.16: QR Code para conexão em rede sem fio

é uma linguagem de marcação de conteúdos para sintetizadores de voz capaz de controlar aspectos como a pronúncia, volume, tom e ritmo. BML é uma especificação para a geração de avatares 3D. Já o SEDL (*Sensory Effect Description Language*) faz parte do *framework* MPEG-V para descrever efeitos sensoriais como luz, vento, nevoeiro e vibração da cadeira, a fim de aumentar a experiência de consumo de um conteúdo audiovisual.

4.1.5 MÓDULO DE PROPRIEDADES DE AMBIENTE

Com o objetivo de auxiliar as adaptações de conteúdos, e proporcionar uma maior dinamicidade à interatividade, o Módulo Propriedades de Ambiente foi inserido para controle de propriedades globais ou locais de uma apresentação. No caso das propriedades globais, essas podem ser acessadas por qualquer cena, já as propriedades locais podem ser acessadas apenas nas cenas às quais forem declaradas. A referência a essas propriedades é realizada por meio do atributo de expressão (*expr*), presente nos elementos de cena e trilha e válido somente quando contidos dentro de uma cena adaptativa e trilha adaptativa, respectivamente. Uma propriedade pode ou não ser inicializada, e essa inicialização é feita por meio do atributo *value*. Além disso, por meio da ação *set*, essas propriedades poderão sofrer alterações ao longo do documento STorML. Para a utilização de propriedades globais uma base deve ser especificada no cabeçalho do documento. Todo o conteúdo desta base pode ser visto na Tabela 4.8. No caso das propriedades locais, essas são declaradas diretamente na cena por meio do elemento <*property*>.

Elementos	Atributos	Elementos Filhos
propertyBase	empty	property+
property	*id*,*value*	empty

Tabela 4.8: Módulo de Propriedades de Ambiente

4.1.6 MÓDULO DE RECONHECIMENTO DE EVENTOS DE ENTRADA

Pensando em tratar o máximo de casos de uso possíveis em DOOH, conforme elencado em ITU-GROUP16 (2014), este trabalho propõe um módulo responsável por determinar alguns padrões de entradas multimodais e suas combinações a fim de realizar ações com base nos elos (<*link*>) criados nas trilhas interativas (<*interactiveTrack*>). De acordo com Guedes et al. (2015), entradas multimodais permitem que usuários interajam com os terminais DOOH por meio de múltiplas modalidades e sensores, de forma orquestrada. Uma modalidade é um estímulo sobre um determinado sentido humano (audição, olfato, tato, paladar, ou visão). Um sensor particular é um dispositivo, no qual, informações geradas pelo usuário são adquiridas (teclados para aquisição de texto, reconhecedores de gestos etc.). Esse conceito foi denominado MUI(*Multimodal User Interface*) conforme Bolt (1980).

Conforme citado, a proposta aqui definida se baseou no trabalho Guedes et al. (2015), o qual trata a utilização de entradas multimodais na linguagem NCL. No entanto, algumas modificações foram necessárias para se adequar à STorML, como, por exemplo, a adaptação da sintaxe dos elos que foram especificados na Subseção 3.1.1.

Para a utilização de eventos multimodais, uma base de padrões de entrada deve ser especificada no cabeçalho do documento STorML. Todo o conteúdo desta base pode ser visto na Tabela 4.9.

Elementos	Atributos	Elementos Filhos
inputPatternBase		(compoundInputPattern | inputPattern)+
compoundInputPattern	*id*,*mode*,*maxDur*	(meta* | compoundInputPattern* | inputPattern*)
inputPattern	*id*,*type*,*src*,*label*	empty

Tabela 4.9: Módulo de Reconhecimento de Eventos de Entrada

Um elemento <*inputPatternBase*> pode conter um ou mais padrões de entrada simples (<*inputPattern*>) ou compostos (<*compoundInputPattern*>). Um padrão de entrada simples não depende de uma combinação de modalidades de interação, como, por exemplo, o simples reconhecimento de voz da pronúncia referente a palavra. O atributo *id* é o identificador do padrão de entrada que será referenciado pelos elos no corpo do documento, internos às trilhas interativas. O atributo *type* do elemento <*inputPattern*>, define o tipo MIME do arquivo referenciado, podendo ser, por exemplo: ***"application/srgs+xml"*** (*Speech Recognition*), que trata o reconhecimento de voz; ***"application/gml+xml"*** (*Gesture Recognition*) responsável pelo reconhecimento de gestos e ***"application/inkml+xml"*** (*Ink Recognition*), responsável pelo reconhecimento de escrita. Apenas esses 3 tipos foram abordados na linguagem, pois são entradas que apresentam, atualmente, linguagens de marcação responsáveis por representá-las, como a SRGS (*Speech Recognition Grammar Specification*) [7], GDL (*Gesture Description Language*)[8], GestureML (*Gesture Markup Language*)[9] e InkXML (*Ink Markup Language*)[10].

O atributo *src* determina a URI do documento de conteúdo que representa o padrão de entrada a ser reconhecido, geralmente especificado com a extensão das linguagens citadas anteriormente. O atributo *label* pode ser utilizado para especificar apenas trecho do documento de conteúdo informado, recurso útil e necessário quando o documento contém várias modalidades de interação de um mesmo tipo. Além disso, o *label* especifica, em conjunto com a linguagem, qual o tipo de modalidade desse trecho, podendo ser espacial, como um apontamento em que as coordenadas são levadas em consideração, ou meramente temporal como um simples gesto em um determinado instante.

O padrão de entrada composto (<*compoundInputPattern*>) é utilizado para tratar a combinação de diversas modalidades de interação. O atributo *id* identifica esse elemento no documento. O atributo *mode* é usado para informar qual o modo de combinação entre as modalidades de interação internas ao elemento. Esses modos foram definidos com base em Nigay e Coutaz (1997), o qual conceitua 3 tipos de combinações: Redundante, quando apenas uma entrada , dentre as fornecidas, precisa ser reconhecida; complementar, quando todas as entradas são necessárias serem reconhecidas; ou sequencialmente complementares, quando todas as entradas necessitam ser reconhecidas, porém, devem ocorrer na ordem

[7] <http://www.w3.org/TR/speech-grammar/>
[8] <http://cci.up.krakow.pl/gdl/>
[9] <http://www.gestureml.org/>
[10] <http://www.w3.org/TR/InkML/>

sequencial especificada. Portanto, os valores possíveis para o atributo modo são, respectivamente, **“redundant”**, **“complementary”** e **“sequentiallyComplementary”**. Como forma de controlar o intervalo de tempo máximo permitido para duas entradas sequencialmente complementares serem identificadas, o atributo *maxDur* foi definido. A Figura 4.17 exibe como os padrões de entrada são representados em STorML.

```
<inputPatternBase>
    <compoundInputPattern id="putThatThere" mode="sequentiallyComplementary">
        <compoundInputPattern id="putThat" mode="complementary">
            <inputPattern id="putThatVoice" type="application/sgrs+xml" src="dialogo.sgrs" label="put-that"/>
            <inputPattern id="pointInput" type="application/gml+xml" src="gestures.gml" label="point"/>
        </compoundInputPattern>
        <compoundInputPattern id="there" mode="complementary">
            <InputPattern id="thereVoice" type="application/sgrs+xml" src="dialogo.sgrs" label="there"/>
            <InputPattern id="pointInput2" type="application/gml+xml" src="gestures.gml" label="point"/>
        </compoundInputPattern>
    </compoundInputPattern>
    <inputPattern id="nextVoiceCommand" type="application/sgrs+xml" src="dialogo.sgrs" label="next" />
    <InputPattern id="nextClaps" type="application/gml+xml" src="palmas.gml" label="palmasDuplas" />
</inputPatternBase>
```

Figura 4.17: Base de padrões de entradas multimodais

A Figura 4.17 mostra três padrões de entrada. O primeiro é um padrão de entrada composto identificado por **“putThatThere”**, os outros dois são padrões de entrada simples, identificados por **“nextVoiceCommand”** e **“nextClaps”**. O primeiro exemplifica como pode ser tratado, em STorML, o trabalho germinal da área **“put-that-there”** (BOLT, 1980) . Esse trabalho propõe um sistema para planejamento tático de unidades militares em um mapa, por meio de interações multimodais como gestos e comandos de voz. O usuário pode mover uma unidade apontando-lhe seu dedo e realizando o comando de voz **“put that”**, receber como resposta uma voz sintetizada com a pergunta **“where?”**, apontar para a posição desejada ao mesmo tempo em que falar **“there”** (GUEDES et al., 2015) .

Para tratar esse caso, um padrão de entrada composto de modo “sequentiallyComplementary” identificado por **“putThatThere”** foi definido, pois os dois padrões de entrada internos devem acontecer sequencialmente e são complementares. Portanto, primeiro é necessário reconhecer o padrão de entrada identificado pelo *id* **“putThat”** do tipo “Complementary”, que contém o comando de voz **“put that”** identificado como **“putThatVoice”** e o padrão de entrada referente ao gesto de apontamento identificado como **“pointInput”**, os quais devem ser reconhecidos em simultâneo, pois são complementares. Em seguida, o outro padrão de entrada composto do tipo “Complementary”, identificado por **“there”** deve ser reconhecido, ele contém os padrões de entrada simples de comando de voz **“there”**

identificado pelo *id* **“thereVoice”** e também o gesto de apontamento identificado como **“pointInput2”**.

Além desse padrão de entrada composto, a Figura 4.17 também exibe o padrão de entrada identificado por **“nextVoiceCommand”** com conteúdo para reconhecimento de fala, que faz referência a um arquivo externo da linguagem SGRS responsável por representar o comando de voz **“next”**, conforme informado pelo atributo *“label”*. O outro padrão de entrada identificado por **“nextClaps”** faz referência a um arquivo externo da linguagem GML, responsável por representar diversos gestos de “palmas”, no entanto, apenas o gesto de “palmas duplas” é levado em conta, conforme informado pelo atributo *“label”*.

Até aqui foram mostradas como são organizadas as informações dos padrões de entradas multimodais na linguagem. A semântica de como esses eventos serão utilizados com o objetivo de realizar as ações desejadas serão especificadas nos elos (<*link*>) presentes nas trilhas interativas. O Capítulo 6 mostrará, em mais detalhes, a utilização desses padrões de entrada, assim como as mídias sintetizadas responsáveis, por exemplo, pela pergunta **“where?”** citada no caso de uso acima.

5 *PLAYER* DE PROVA DE CONCEITO STorML

Como prova de conceito, um *player* simplificado foi desenvolvido em *JAVA* para simular a apresentação de documentos STorML. A exibição de cada mídia é tratada textualmente por meio de impressões sucessivas em um *label Swing*. Nesse *label* são exibidas as seguintes informações da mídia:***"type id tempo-atual/duração"***. Caso a mídia tenha *delay* definido, uma mensagem ***"Media Delay..."*** será exibida antes de seu início. Esse *label* está presente em um painel, instância da classe *Panel*, o qual está diretamente associado a apenas uma região. Cada região também está associada a apenas um painel. A Figura 5.1 ilustra uma execução nesse *player*. Nessa figura uma cena que detém um leiaute específico foi utilizada como ilustração, no entanto, a disposição espacial é realizada de forma dinâmica, ou seja, se baseia no(s) leiaute(s) utilizado(s) pela cena em execução.

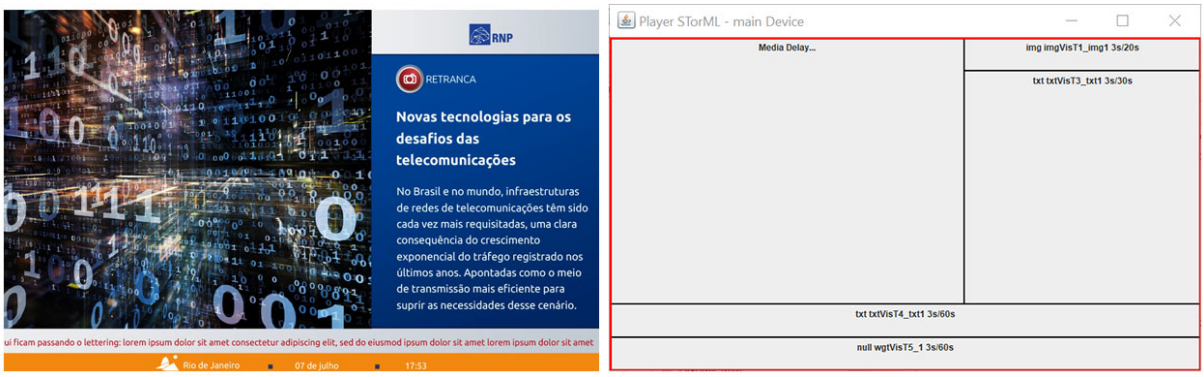


Figura 5.1: Ilustração de funcionamento do *Player*.

A imagem 5.2 mostra, de forma resumida, o diagrama de classes do *player*. A classe *Application* é a principal (*main*). Nela é tratada a exibição da janela para seleção do documento STorML a ser exibido e consequente iniciação do *parser*, de modo a criar os objetos do modelo de acordo com o documento selecionado. Cada objeto presente no documento STorML é criado pelo *parser*, de maneira encadeada, ao longo da estrutura hierárquica. Quanto aos atributos desta classe, destacam-se o atributo *"apresentacao"*, o qual é representado como uma lista de cenas e o atributo que referencia a cena atualmente exibida (cena corrente). Para tratar a execução em multidispositivos, uma lista de *JFrames* é definida, sendo cada *frame* instanciado por meio da classe *Display*.

A classe *Scene* é a classe responsável por representar cada cena definida no documento STorML. Nela os atributos de repetição, expressão, lista de trilhas, lista de *layouts*

utilizados e duração máxima são declarados. Para sua execução, a mesma implementa a interface *Runnable*, muito utilizada para tratar a execução de *threads* em *java*. Além disso, a mesma herda diversos atributos e comportamentos da classe *ObjectModel*. Essa classe foi criada para agregar as similaridades das entidades *Scene*, *Track* e *Media*. Dentre essas similaridades, pode-se citar os comportamentos de *start*, *stop*, *resume* e *pause*. Além disso, nela são definidos os estados de exibição possíveis para cada instância de subclasse, são eles: *SLEEPING*, *OCURRING* e *PREPARED*, conforme ilustrado na Subseção 4.1.3.

A classe *Track* é uma classe abstrata responsável por agregar todas as similaridades das trilhas adaptativas, trilha visuais, trilhas de áudio, trilhas de texto e trilhas interativas. No entanto, uma entidade intermediária, denominada *MediaTrack*, foi criada para tratar as trilhas que contêm apenas objetos de mídias, diferente da trilha adaptativa que contém apenas trilhas que são instâncias de *MediaTrack*. A classe *Track*, além de herdar de *ObjectModel*, também implementa a interface *Runnable*, com o objetivo de se ter uma *thread* responsável por cada trilha. Todas propriedades e comportamentos comuns são especificadas nessa classe, como o modo de início da trilha *startMode*. Por ser uma classe abstrata, ela apenas define a assinatura dos métodos exigidos nas classes derivadas, como: iniciar, parar, pausar e retomar uma trilha, assim como a mudança de região na qual a trilha está sendo exibida. Como forma de controlar o tempo de exibição da trilha, um atributo para tempo de início (*startTime*) e outro para relógio corrente (*currentTime*) são também declarados.

AdpativeTrack é a classe que representa as trilhas adaptativas e que também herda da classe abstrata *Track*. Seu papel é controlar a exibição da trilha interna que atender a uma determinada expressão presente em seu atributo *expr*. Desta forma, todos os métodos que referenciam a trilha adaptativa serão reportados à trilha interna atualmente válida pela expressão. Uma trilha adaptativa contém como atributo, uma lista de trilhas alternativas e a trilha alternativa que está ocorrendo, caso haja alguma. Seguindo os mesmos conceitos, a classe *AdpativeScene* representa as cenas adaptativas e herdam da classe *Scene*. Essa classe contém, também, a lista de cenas alternativas e a cena que está ocorrendo, caso haja alguma.

A classe *MediaTrack*, conforme citado, agrupa as propriedades e comportamentos semelhantes às trilhas interativas, trilhas visuais, trilhas de áudio e trilhas de texto. A grande diferença entre as classes derivadas de *MediaTrack*, está relacionada às trilhas in-

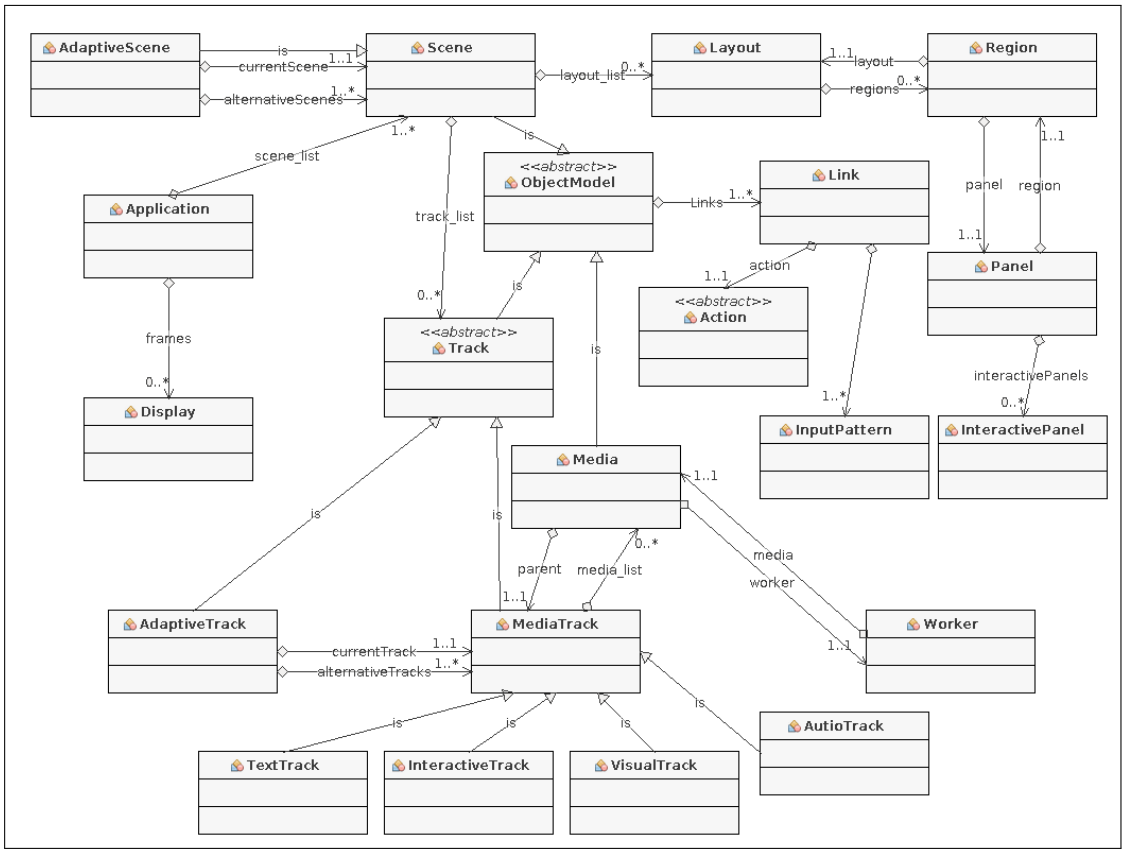


Figura 5.2: Diagrama de classes do Player STorML

terativas. Diferente das trilhas visuais, as trilhas interativas apresentam, além do painel (*Panel*) principal para cada região associada, painéis internos denominados painéis interativos (*InteractivePanel*) que são responsáveis pela exibição de cada mídia interna a uma trilha interativa. Conforme citado em seções anteriores, todo o relacionamento é baseado em elos causais, portanto, a exibição temporal e espacial das mídias internas são independentes. Ou seja, mais de uma mídia pode ser exibida em um mesmo instante de tempo, além disso, os eventos para cada mídia precisam ser identificados separadamente. Nesse caso, houve a necessidade de se criar esses painéis interativos, cada qual associado a uma mídia interna à trilha interativa.

Como propriedades, a classe *MediaTrack* detém de uma lista de mídias, as quais serão exibidas conforme estão dispostas sequencialmente, exceto no caso das trilhas interativas. Nessa última, os elos irão definir o momento de exibição de cada mídia, no entanto, a classe *InteractiveTrack* detém de uma lista de mídias que são iniciadas no mesmo instante ao início da trilha, determinada pelos elementos *autoStart* de STorML, citados na Subseção 4.1.3.

A classe *Media* representa o elemento de mídia STorML. Além de conter as propri-

edades de posicionamento, tamanho, delay e tipo, essa classe contém a referência à sua trilha pai de modo a facilitar os comportamentos da mídia, levando em conta a trilha pai correspondente. Por exemplo, qualquer ação no objeto de mídia vai depender de qual é o estado de exibição da trilha pai. Como forma de tratar a simulação de exibição das mídias, a instância *Media* estará sempre associada a uma instância da classe *Worker*. Essa classe deriva da classe Java *SwingWorker* responsável por tratar tarefas que ocorrem em paralelo à grande tarefa EDT responsável por encapsular todos os eventos *Swing*. Normalmente essa classe *SwingWorker* é utilizada para realizar trabalhos mais pesados, e que não interfiram no funcionamento da grande *Thread* principal EDT (Event Dispatch Thread). Dessa forma, evita-se o travamento da aplicação por aguardar a resposta de uma tarefa mais robusta. Diversas outras maneiras poderiam ser utilizadas nesse caso, como as próprias classes *Runnable*, no entanto, algumas facilidades presentes na classe *SwingWorker* foram levadas em conta, como a simplicidade na troca de mensagens entre a grande *thread* EDT e a *worker* associada a mídia que está sendo exibida. Desta forma, é possível não só tratar os atrasos (*delay*) para exibição da mídia, como também o *sleep* a cada segundo da mídia, sem que a *thread* principal EDT seja afetada.

Nesse *player*, todo relacionamento (*Link*) está diretamente relacionado a um ou mais objetos do modelo. A seleção de um objeto, por exemplo, fará com que o respectivo *link* seja executado, ou seja, todas as suas ações (*Action*) associadas serão executadas. Conforme se pode ver na Figura 5.2, a ação (*Action*) é uma classe abstrata, portanto, para cada tipo de ação, como *start*, *stop*, *resume*, *seek*, *set* e *pause* uma classe própria foi criada, no entanto, essas foram omitidas na Figura 5.2. Da mesma forma, os relacionamentos foram derivados em classes próprias, ou seja, para cada condição suportada em um *link*, uma classe foi criada: elos de seleção (*OnSelectionLink*), elos de reconhecimento (*OnRecognitionLink*), elos baseados em instante de tempo (*OnInstantLink*), dentre outras. Essas classes também foram omitidas da Figura 5.2. No caso dos elos de reconhecimento, esses dependem dos padrões de entrada definidos no cabeçalho, para isso a classe *InputPattern* foi criada.

InputPattern é a classe responsável por representar tanto os elementos de padrão de entrada simples como compostos. No caso dos compostos, uma lista de padrões simples é declarada. Além disso, um atributo *ObjectModel* é declarado nessa classe para referenciar a qual objeto do documento STorML o padrão de entrada está associado. Ou seja, esse

padrão de entrada só terá validade caso esse objeto esteja sendo executado. Para tratar os eventos de entradas multimodais, teclas e apontamento por meio do mouse foram utilizados de modo a simular um comando de voz e um apontamento para determinado objeto do documento STorML. Desta forma, pode-se verificar se o apontamento ocorreu na região onde o objeto associado se encontra e se esse apontamento ocorreu no intervalo de tempo em que esse objeto estava sendo exibido.

6 CASOS DE USO

Neste capítulo são mostrados alguns casos de uso, em STorML, que ilustram conteúdos possíveis para DOOH. O objetivo é ajudar o leitor a entender melhor os conceitos detalhados nos capítulos anteriores, além de ser uma prova de conceito com a utilização do *player* STorML especificado anteriormente.

Além do detalhamento de conteúdos para TV Corporativa, será mostrado uma maneira de melhorar conteúdos não interativos e determinísticos que serão disponibilizados aos terminais. Para isso, leva-se em conta a redução do arquivo por meio de renderizações sucessivas em camadas, de modo a facilitar sua difusão *multicast* ou *broadcast* por meio de uma rede com QoS garantida, como a RNP.

6.1 CONTEÚDOS DE TV CORPORATIVA

Como forma de ilustrar a utilização de STorML, esta seção apresenta três casos de uso, os quais abordam, respectivamente, um conteúdo interativo de *slideshow* em *videowall*, um outro conteúdo que busca mostrar a adaptação de conteúdo com uso de dispositivos secundário a fim de se aumentar a imersão e, finalmente, um conteúdo interativo que exibe a possibilidade de se usar a linguagem para treinamento corporativo com a utilização de entradas multimodais espelhadas na aplicação *"put-that-there"* citada anteriormente.

6.1.1 CONTEÚDO INTERATIVO DE *SLIDESHOW* EM *VIDEOWALL*

Nesse caso de uso, disponibilizado pela RNP (REDE..., 2009) , são exibidas imagens sincronizadas com alguns títulos de notícias de maneira rotativa. Além da sincronização baseada no tempo, a interatividade é permitida através da seleção de ícones responsáveis pela troca das imagens (*slideshow interativo*), também de modo sincronizado aos títulos de notícia correspondentes.

Esse caso é composto apenas por uma cena que busca mostrar a utilização do recurso de arranjo de monitores em STorML, onde a trilha visual que contém as imagens do *slideshow* será exibida em três monitores de forma simultânea. Todos esses monitores estão conectados ao mesmo dispositivo principal (*main*), portanto, são tratados como regiões desse dispositivo, conforme pode ser visto na listagem 6.1. No entanto, a interatividade

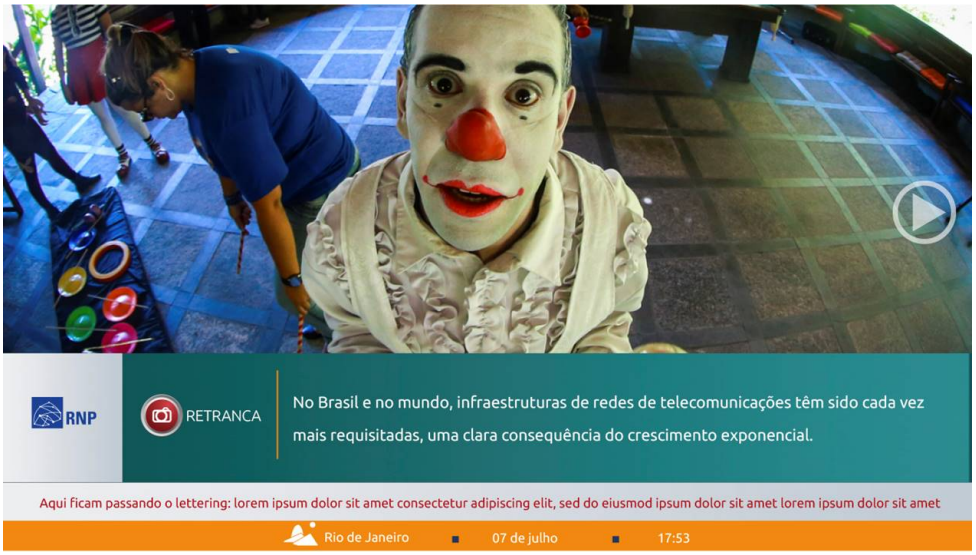


Figura 6.1: Caso de uso interativo

continuará disponível apenas no monitor central, portanto, a imagem selecionada no monitor central será replicada para os outros dois monitores do arranjo. A Figura 6.2 ilustra o arranjo.

Listagem 6.1: Cabeçalho STorML

```
<storml>
  <head>
    <layoutBase>
      <layout id="lyVideoWall" dev="main" dev-arrangement="3x1x16:9">
          <!-- Left TV -->
          <region id="rgLeftTV" top="0%" left="0%" width="33.33333%" height="100%" z-index="1"/>
        <!-- Central TV -->
        <region id="rgNewsSlides" top="0%" left="33.33333%" width="33.33333%" height="60%" z-index="1"/>
        <region id="rgNewsHeadline" top="60%" left="33.33333%" width="33.33333%" height="30%" z-index="1"/>
        <region id="rgScrollingText" top="90%" left="33.33333%" width="33.33333%" height="5%" z-index="1"/>
        <region id="rgWidget" top="95%" left="33.33333%" width="33.33333%" height="5%" z-index="1"/>
        <region id="rgNewsControls" top="25%" left="33.33333%" width="33.33333%" height="10%" z-index="1"/>
          <!-- Right TV -->
          <region id="rgRightTV" top="0%" left="66.66666%" width="33.33333%" height="100%" z-index="1"/>
      </layout>
    </layoutBase>
  </head>
```

A representação da primeira cena pode ser vista na listagem 6.2 e é composta por: Uma trilha visual, responsável por exibir o texto rolante; uma trilha interativa responsável por exibir objetos de mídia do tipo *widget*, com informações de temperatura, data e hora da localidade do terminal; uma trilha visual que contém as imagens do *slideshow* que são exibidas em regiões simultâneas (*rgNewsSlides* && *rgLeftTV* && *rgRightTV*), representadas por monitores alinhados, conforme ilustrado na Figura 6.2; e uma trilha interativa responsável por gerenciar a troca sincronizada das imagens presentes no *slideshow* e os títulos de notícia. Todo esse sincronismo é realizado por meio de elos (<*link*>) compostos

por uma condição (*OnSelection*) que deve ser atendida para que duas ações simultâneas (*seek next* ou *seek prev*) sejam executadas nas trilhas determinadas.

Listagem 6.2: Corpo da apresentação STorML

```
...
 <body presentationMode="sequential">
  <scene id="scNewsHead">
   <visualTrack id="vtSlides" repeatCount="indefinite" region="rgNewsSlides&&rgLeftTV&&rgRightTV">
      <media id="imgSlide1" type="image/jpg" src="..." dur="20s"/>
      <media id="imgSlide2" type="image/jpg" src="..." dur="20s"/>
      <media id="imgSlide3" type="image/jpg" src="..." dur="20s"/>
   </visualTrack>
   <visualTrack id="vtNewsHeadline" repeatCount="indefinite" region="rgNewsHeadline">
     <media id="imgHeadline1" type="image/jpg" src="..." dur="20s"/>
     <media id="imgHeadline2" type="image/jpg" src="..." dur="20s"/>
     <media id="imgHeadline3" type="image/jpg" src="..." dur="20s"/>
   </visualTrack>
   <visualTrack id="vtScrollingText" repeatCount="indefinite" region="rgScrollingText">
     <media id="txtScrollingText" type="text/plan" src="..."dur="indefinite"/>
   </visualTrack>
   <adaptiveTrack maxDur="300s" repeatCount="indefinite" id="adtWidget">
     <interactiveTrack expr="device.location==RJ" id="itWidgetRJ" src="..." type="application/x-itu-iptv-widget"
          region="rgWidget"/>
     <interactiveTrack expr="device.location==BH" id="itWidgetBH" src="..." type="application/x-itu-iptv-widget"
          region="rgWidget"/>
   </adaptiveTrack>
   <interactiveTrack id="itTrackControls" region="rgNewsControls" >
     <autoStart media="btnNext" />
     <autoStart media="btnPrev" />
     <media id="btnPrev" type="image/jpg" height="100%" width="20%"/>
     <media id="btnNext" type="image/jpg" left="80%" height="100%" width="20%"/>
     <link>onSelection btnNext then seek next vtSlides || seek next vtNewsHeadline end</link>
     <link>onSelection btnPrev then seek prev vtSlides || seek prev vtNewsHeadline end</link>
   </interactiveTrack>
 </scene>
 </body>
</storml>
```

Figura 6.2: Ilustração da primeira cena em arranjo 3x1x16:9

Como prova de conceito, esse documento foi executada no *player* STorML simplificado citado no Capítulo 5. A Figura 6.3 ilustra a execução com a possibilidade de seleção das mídias *btnPrev* e *btnNext* de modo a ocasionar a troca de imagens nas trilhas de maneira sincronizada.

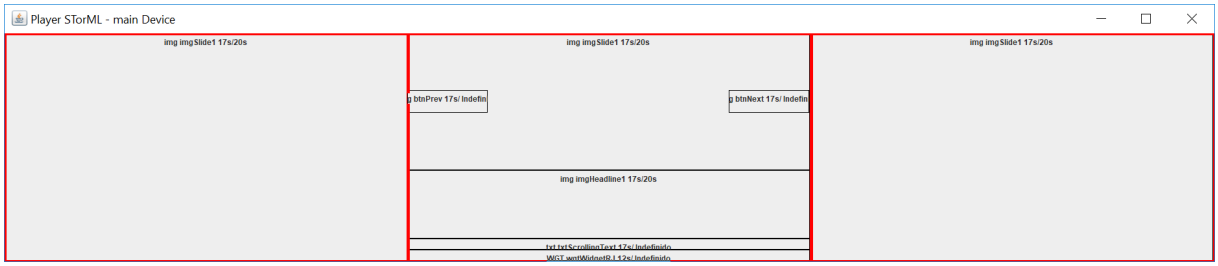


Figura 6.3: Execução do caso de uso no *player* STorML

6.1.2 CONTEÚDO ADAPTATIVO EM MULTIDISPOSITIVOS

Nesse caso de uso, criado apenas para exemplificar as funcionalidades de STorML, são exibidas informações de acordo com o contexto de localização do terminal. Além da simples exibição adaptada no terminal, o conteúdo é disponibilizado aos dispositivos secundários, como *smartphones*, conectados a estes terminais.

O caso de uso exemplifica a utilização de um terminal móvel, comumente encontrado em transportes coletivos, no âmbito da UFJF. A proposta é fornecer ao público da Universidade um conteúdo que está de acordo com a localização do terminal. Ou seja, um conteúdo com informações sobre o ICE (Instituto de Ciências Exatas) será mostrado caso o veículo esteja próximo a esse Instituto. Da mesma forma, um conteúdo próprio será exibido para o ICH (Instituto de Ciências Humanas). Caso o veículo esteja em qualquer outro ponto da instituição, um conteúdo padrão com informações gerais sobre a Universidade será exibido. Além do conteúdo totalmente diferenciado para cada localização, o caso de uso aborda a disponibilização do cardápio semanal do RU (Restaurante Universitário) em período de horário comumente utilizado para o almoço dos estudantes. Esse controle de conteúdo por meio do horário é tratado como uma adaptação de apenas parte do conteúdo total.

Para representar esse caso de uso em STorML, duas cenas principais foram definidas, uma responsável por exibir informações gerais sobre a Universidade e outra responsável pela exibição adaptada de acordo com a localização do terminal móvel. Essa última trata-se de uma cena adaptativa, a qual irá se especializar em uma das duas cenas internas, uma responsável por exibir o conteúdo referente ao ICE e outra referente ao ICH. A primeira cena ilustrada pela Figura 6.4, contém um vídeo institucional com legenda sobre a inauguração do Centro de Ciência da UFJF, notícias gerais e informações básicas de temperatura e horário.

Figura 6.4: Ilustração sobre a UFJF

A segunda cena, conforme dito, trata-se de uma cena adaptativa e que, portanto, irá especializar em duas cenas distintas. Essas cenas podem ser vistas, de maneira ilustrativa, nas Figuras 6.5 e 6.6. Cada uma delas contém um vídeo específico com legenda, notícias próprias de cada Instituto e também informações básicas de temperatura e horário.

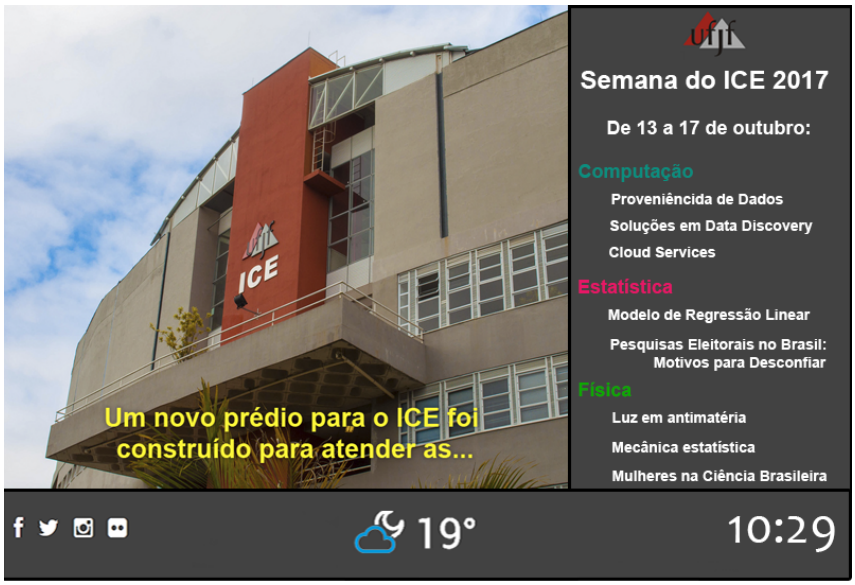


Figura 6.5: Ilustração da Cena sobre o ICE

Além destas exibições padrões de cada cena, em determinado horário cada cena terá seu conteúdo alterado na região onde normalmente são exibidas as notícias. Nessa região, no horário de 11:00 às 14:00, informações do cardápio semanal serão mostradas, além disso, essa mesma informação será exibida nos dispositivos secundários, exemplificados

Figura 6.6: Ilustração da Cena sobre ICH

por *smartphones*. Essa ilustração pode ser vista na Figura 6.7 para a localização do ICH, porém, da mesma forma acontecerá com todas as outras cenas. Em STorML, essas adaptações internas a uma cena são possíveis por meio do conceito de trilhas adaptativas.

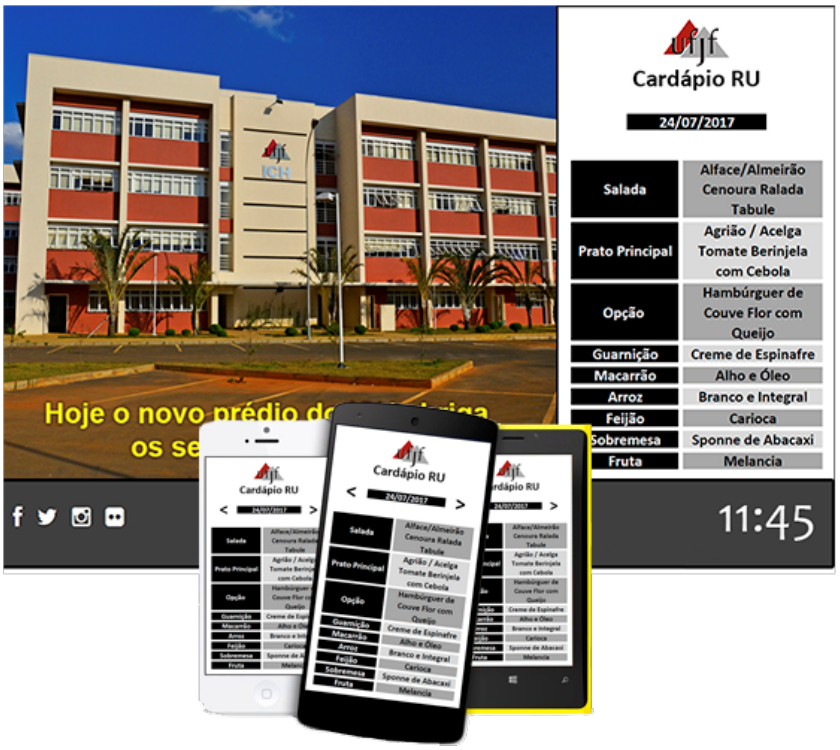


Figura 6.7: Ilustração da exibição do cardápio em dispositivo principal e secundário

Para tratar esse caso de uso, dois *layouts* foram definidos. O primeiro especifica as regiões do terminal principal. Esse *layout* será reutilizado por todas as cenas por

apresentarem a mesma composição espacial. O outro *layout* se refere a exibição das informações de cardápio no dispositivo secundário. Esse *layout* é composto por duas regiões. A primeira região delimita toda a tela do dispositivo secundário e será usada para exibir o cardápio diário. A outra região especifica os locais onde as imagens responsáveis pela interação estarão presentes. A listagem 6.3 mostra esses *layouts*.

Listagem 6.3: Cabeçalho STorML para caso de uso adaptativo

```
<storml>
<head>
   <layoutBase>
      <layout id="lyVideoWall" dev="main">
        <region id="rgFull" top="0%" left="0%" width="100%" height="100%" z-index="1"/>
        <region id="rgVideo_Images" top="0%" left="0%" width="60%" height="80%" z-index="2"/>
        <region id="rgLegenda" top="70%" left="10%" width="40%" height="10%" z-index="3"/>
        <region id="rgMainNews" top="0%" left="60%" width="40%" height="80%" z-index="2"/>
        <region id="rgWidget" top="80%" left="0%" width="100%" height="20%" z-index="2"/>
      </layout>
      <layout id="lyMobRU" dev="secondary(1)">
        <region id="rgMobRU" top="0%" left="0%" width="100%" height="100%" z-index="1"/>
        <region id="rgNavigationArrowsMobRU" top="20%" left="0%" width="100%" height="15%" z-index="2"/>
      </layout>
   </layoutBase>
</head>
```

A listagem 6.4 mostra todo o conteúdo do documento STorML necessário para representar esse caso de uso. Nesse documento existem três cenas que serão exibidas sequencialmente (*presentationMode*), cada qual com duração máxima de 5 minutos e composta por sete trilhas. A primeira trilha é uma trilha visual responsável por exibir o vídeo institucional de cada cena. A segunda trata-se de uma trilha de texto responsável por representar a legenda referente ao vídeo institucional. Nessa trilha, um objeto de mídia texto é referenciado e esse objeto é composto por diversos trechos de texto, cada qual com seu momento de exibição autocontido. A terceira trilha trata-se de uma trilha adaptativa que irá determinar qual o conteúdo para a região identificada por *rgMainNews* a partir da análise de expressão (*expr*) presente nas duas trilhas visuais internas. Com base no horário, apenas uma trilha será exibida, são elas: uma responsável pelas notícias institucionais (*default*) ou outra responsável pelas informações de cardápio. Observa-se que, pelo tempo de duração agregado das mídias internas, a reavaliação das expressões irá ocorrer a cada 60 segundos (*repeatCount="indefinite"*) até que a cena atinja sua duração máxima. Essa avaliação ocorre da mesma maneira em todas as outras cenas desse caso de uso.

Listagem 6.4: Corpo da instância STorM para caso de uso adaptativo

```
<body presentationMode="sequential">
    <!-- Cena sobre UFJF -->
    <scene maxDur="300s" id="scMainUFJF">
        <visualTrack id="vtUFJF" region="rgVideo_Images" repeatCount="indefinite">
            <media id="vidUFJF" type="video" src="..." dur="60s"/>
        </visualTrack>
        <textTrack id="ttLegendaVidUFJF" src="..." type="text/str" dur="60s" repeatCount="indefinite" region="rgLegenda" />
        <adaptiveTrack id="adtInfoICE" repeatCount="indefinite">
            <visualTrack id="vtCardapioRU" expr="system.time>='11:00'&&system.time<='14:00'" region="rgMainNews">
                <media id="imgCardapioDia1" type="image" src="..." dur="12s"/>
                <media id="imgCardapioDia2" type="image" src="..." dur="12s"/>
                <media id="imgCardapioDia3" type="image" src="..." dur="12s"/>
                <media id="imgCardapioDia4" type="image" src="..." dur="12s"/>
                <media id="imgCardapioDia5" type="image" src="..." dur="12s"/>
            </visualTrack>
            <visualTrack id="vtNewsUFJF" region="rgMainNews">
                <media id="imgNewsUFJF1" type="image" src="..." dur="20s"/>
                <media id="imgNewsUFJF2" type="image" src="..." dur="20s"/>
                <media id="imgNewsUFJF3" type="image" src="..." dur="20s"/>
            </visualTrack>
        </adaptiveTrack>
        <visualTrack id="vtCardapioRUMob" startMode="deferred" refer="vtCardapioRU" repeatCount="indefinite"
            region="rgMobRU"/>
        <interactiveTrack id="itCardapioRU" startMode="deferred" region="rgNavigationArrowsMobRU">
            <autoStart media="btnNext"/>
            <autoStart media="btnPrev"/>
            <media id="btnNext" type="image" top="0%" left="0%" width="10%" height="100%" dur="indefinite"/>
            <media id="btnPrev" type="image" top="0%" left="90%" width="10%" height="100%" dur="indefinite"/>
            <link>onSelection btnNext then seek next vtCardapioRUMob</link>
            <link>onSelection btnPrev then seek prev vtCardapioRUMob</link>
        </interactiveTrack>
        <interactiveTrack id="itStartMobTracks" region="rgFull" >
            <link>onBegin vtCardapioRU then start vtCardapioRUMob || start itCardapioRU end</link>
        </interactiveTrack>
        <interativeTrack id="wgtTempHour" type="application/x-itu-iptv-widget" src="..." region="rgWidget"
            repeatCount="indefinite" dur="60s"/>
    </scene>
    <adaptiveScene id="adsICE_or_ICH">
        <!-- Cena sobre ICE -->
        <scene maxDur="300s" id="scICE" expr="system.location=='-21.775304, -43.371353,10'">
            <visualTrack id="vtICE" region="rgVideo_Images">
                <media id="vidICE" type="video" src="..." dur="60s"/>
            </visualTrack>
            <textTrack id="ttLegendaVidICE" src="...str" type="text/str" dur="60s" region="rgLegenda" />
            <adaptiveTrack id="adtNewsICE_or_Cardapio" repeatCount="indefinite">
                <visualTrack id="vtCardapioRUICE" refer="vtCardapioRU"/>
                <visualTrack id="vtNewsICE" region="rgMainNews">
                    <media id="imgNewsICE1" type="image" src="..." dur="20s"/>
                    <media id="imgNewsICE2" type="image" src="..." dur="20s"/>
                    <media id="imgNewsICE3" type="image" src="..." dur="20s"/>
                </visualTrack>
            </adaptiveTrack>
            <visualTrack id="vtCardapioRUMob2" startMode="deferred" refer="vtCardapioRU" repeatCount="indefinite"
                region="rgMobRU"/>
            <interactiveTrack id="itCardapioRU2" startMode="deferred" region="rgNavigationArrowsMobRU">
                <autoStart media="btnNext2"/>
                <autoStart media="btnPrev2"/>
                <media id="btnNext2" type="image" top="0%" left="0%" width="10%" height="100%" dur="indefinite"/>
                <media id="btnPrev2" type="image" top="0%" left="90%" width="10%" height="100%" dur="indefinite"/>
                <link>onSelection btnNext2 then seek next vtCardapioRUMob2</link>
                <link>onSelection btnPrev2 then seek prev vtCardapioRUMob2</link>
            </interactiveTrack>
            <interactiveTrack id="itStartMobTracks2" region="rgFull" >
                <link>onBegin vtCardapioRUICE then start vtCardapioRUMob2 || start itCardapioRU2 end</link>
            </interactiveTrack>
```

```
            <interativeTrack id="wgtTempHour2" refer="wgtTempHour"/>
        </scene>
        <!-- Cena sobre ICH -->
        <scene maxDur="300s" id="scICH" expr="system.location=='-21.773223,-43.366100,10'">
            <visualTrack id="vtICH" region="rgVideo_Images">
                <media id="vidICH" type="video" src="..." dur="60s"/>
            </visualTrack>
            <textTrack id="ttLegendaVidICH" src="..." type="text/str" dur="60s" region="rgLegenda" />
            <adaptiveTrack id="adtNewsICH_or_Cardapio" repeatCount="indefinite">
                <visualTrack id="vtCardapioRUICH" refer="vtCardapioRU"/>
                <visualTrack id="vtNewsICH" region="rgMainNews">
                    <media id="imgNewsICH1" type="image" src="..." dur="20s"/>
                    <media id="imgNewsICH2" type="image" src="..." dur="20s"/>
                    <media id="imgNewsICH3" type="image" src="..." dur="20s"/>
                </visualTrack>
            </adaptiveTrack>
            <visualTrack id="vtCardapioRUMob3" startMode="deferred" refer="vtCardapioRU" repeatCount="indefinite"
                region="rgMobRU"/>
            <interactiveTrack id="itCardapioRU3" startMode="deferred" region="rgNavigationArrowsMobRU">
                <autoStart media="btnNext3"/>
                <autoStart media="btnPrev3"/>
                <media id="btnNext3" type="image" top="0%" left="0%" width="10%" height="100%" dur="indefinite"/>
                <media id="btnPrev3" type="image" top="0%" left="90%" width="10%" height="100%" dur="indefinite"/>
                <link>onSelection btnNext3 then seek next vtCardapioRUMob3</link>
                <link>onSelection btnPrev3 then seek prev vtCardapioRUMob3</link>
            </interactiveTrack>
            <interactiveTrack id="itStartMobTracks3" region="rgFull">
                <link>onBegin vtCardapioRUICH then start vtCardapioRUMob3 || start itCardapioRU3 end</link>
            </interactiveTrack>
            <interativeTrack id="wgtTempHour3" refer="wgtTempHour"/>
        </scene>
    </adaptiveScene>
</body>
</storml>
```

A partir do momento em que a trilha visual interna, responsável pela exibição do cardápio no dispositivo principal, iniciar, ou seja, quando o horário determinado acontecer, duas trilhas responsáveis por prover o mesmo conteúdo nos dispositivos secundários irão iniciar por meio de um elo causal definido pela trilha interativa, identificada por *itStartMobTracks*. Uma destas trilhas é visual e contém as mesmas imagens da trilha identificada por *vtCardapioRU*. Por conter os mesmos objetos de mídia, uma cópia foi criada por meio do atributo *refer*, no entanto, a propriedade de região foi sobrescrita por um novo valor (*rgMobRU*). A outra trilha é uma trilha interativa que define por meio de elos o recurso de navegação entre os cardápios diários. Portanto, o recurso de interatividade estará disponível apenas nos dispositivos secundários, de modo a garantir a liberdade dos usuários na navegação entre as mídias. Observa-se que, dentre todas as trilhas presentes na cena, apenas essas duas não iniciarão no mesmo instante ao da cena, pois o atributo *startMode* foi definido como *deferred*, sendo exibidas apenas por meio do elo causal citado, o qual necessita de uma condição (*onBegin*) para executar duas ações em paralelo (||). A última trilha, identificada por *wgtTempHour*, é utilizada para exibir as informações de tempera-

tura e horário e são representadas como *widgets*, sendo todo seu conteúdo autocontido. Todas as outras cenas fazem uso dos mesmos *layouts* e também de trilhas parecidas. Portanto, outras cópias das trilhas similares foram definidas por meio do atributo *refer*. Como prova de conceito, a Figura 6.8 mostra a execução da primeira cena, responsável por exibir informações da UFJF, em horário onde o conteúdo de cardápio é exibido tanto no dispositivo principal como no secundário. Percebe-se que no dispositivo secundário a interação para troca de imagens é possível por meio da seleção das imagens *btnNext* e *btnPrev*.

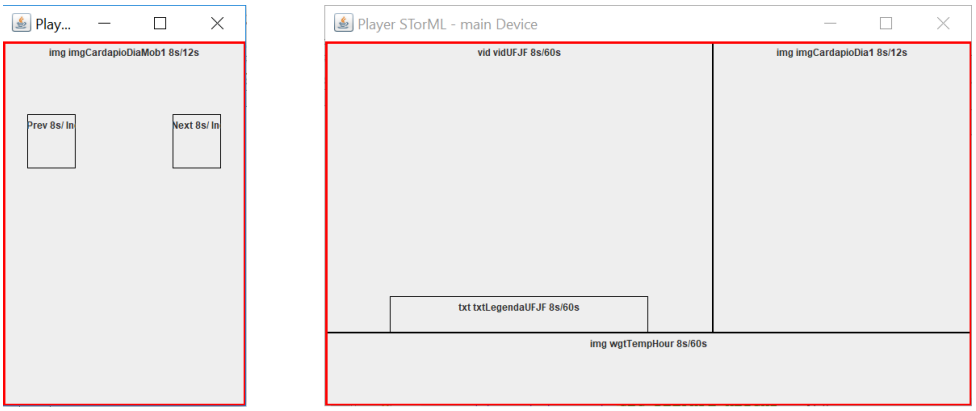


Figura 6.8: Execução de cena sobre UFJF em dispositivo principal e secundário.

6.1.3 CONTEÚDO DE APRENDIZAGEM COM INTERAÇÃO MULTIMODAL

Com o objetivo de mostrar a capacidade STorML na criação de conteúdos voltados a disseminação de conhecimento na corporação e também a possibilidade de caminhamento não-linear entre cenas, um caso de uso utilizado no trabalho SceneSync (BUSSON et al., 2016) , o qual aborda o domínio específico de OA (Objeto de Aprendizagem), foi retratado em STorML com o acréscimo de novos recursos de interação multimodal e controle no caminhamento temporal da trilha. O caso de uso consiste de um objeto de aprendizagem (OA) não-linear, sobre algoritmos de ordenação. O caso de uso foi denominado "Aula de algoritmos de ordenação", e é composto de três cenas. Na primeira cena, o instrutor introduz conceitos gerais sobre algoritmos de ordenação. Em determinado momento, ele pergunta ao usuário se deseja aprender mais sobre o algoritmo de ordenação *Insertsort* ou *Quicksort*. Dependendo da escolha, o objeto de aprendizagem iniciará a cena que aborda o tema de interesse.

Portanto, a primeira cena contém um vídeo que introduz os conceitos dos algoritmos de ordenação, uma imagem que ilustra a opção *insertSort* e outra imagem que ilustra a opção *quickSort*. A segunda cena trata sobre o algoritmo *insertsort* e contém uma imagem de fundo, um vídeo que aborda o algoritmo de ordenação *insertsort* e uma outra imagem que ilustra um exemplo do algoritmo de ordenação *insertsort*. A terceira cena trata sobre o algoritmo *quicksort* e contém também uma imagem de fundo, um vídeo que aborda os conceitos da ordenação *quicksort* e uma imagem que ilustra um exemplo do algoritmo. A fim de se mostrar os recursos de interação multimodal, o usuário poderá efetuar a troca de regiões das trilhas que contém os objetos de mídia na segunda e terceira cena. Ou seja, caso o leitor tenha interesse em visualizar a imagem do algoritmo em uma região maior, o mesmo poderá efetuar a troca com a região do vídeo e vice-versa. A Figura 6.9 ilustra esse caso de uso.

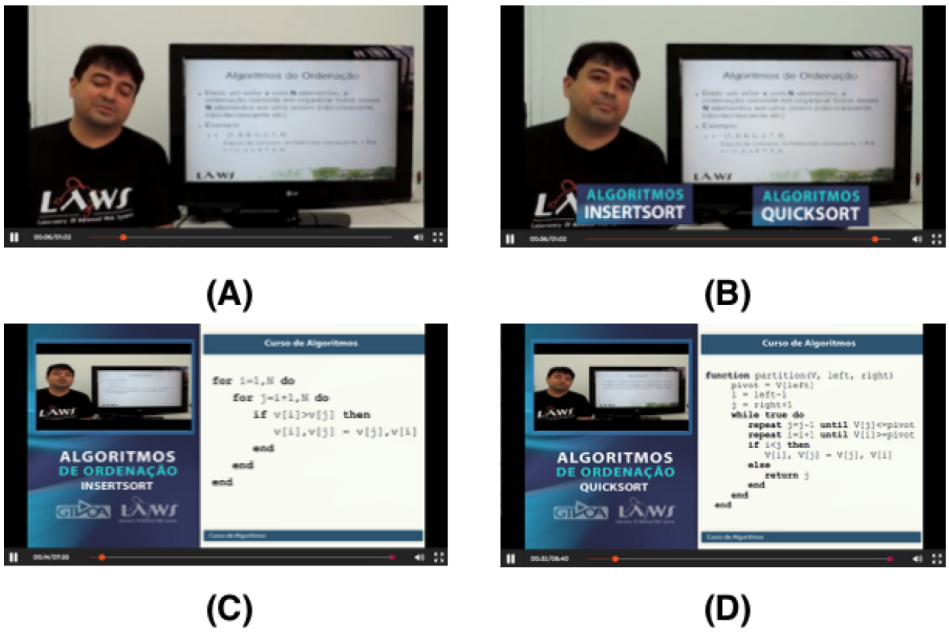


Figura 6.9: Ilustração da Aula em execução[1]

Em **(A)** e **(B)** é retratado os dois momentos da primeira cena. Primeiramente o vídeo responsável pela introdução aos conceitos de algoritmos é iniciado, e aos 45 segundos duas imagens são exibidas para que o leitor possa fazer sua escolha, sobre qual caminho seguir. No entanto, nessa remodelagem STorML, foram inseridas novas funcionalidades de controle temporal na trilha responsável pelo vídeo introdutório, possibilitando ao leitor fazer uso de ações *seek*, *pause* e *resume*. No caso do *seek*, o usuário poderá fazer o avanço (+)

[1]<http://dl.acm.org/citation.cfm?id=2976855>

ou retrocesso (-) de 5 segundos, a cada clique. Em **(B)** e **(C)** são mostradas a primeira e segunda cena, respectivamente. Percebe-se que ambas apresentam o mesmo *layout*, além disso, cada cena apresenta uma imagem com o objetivo de retornar à primeira cena, de modo a permitir um caminhamento não linear entre as cenas. Esse caminhamento é possível pois o modo de apresentação é definido como aleatório (*presentationMode="random"*). A listagem 6.5 mostra o documento StorML completo para esse caso.

Listagem 6.5: Corpo do documento STorML para caso de uso de aprendizagem

```
<storml>
  <head>
    <layoutBase>
      <layout dev="main">
        <region id="rgVideoIntroducao" top="0%" left="10%" width="80%" height="100%" z-index="2" />
        <region id="rgButtons" top="70%" left="10%" width="80%" height="30%" z-index="2"/>
      </layout>
      <layout dev="main">
        <region id="rgBackGroundAlg" top="0%" left="0%" width="100%" height="100%" z-index="1"/>
        <region id="rgInteractive" top="80%" left="0%" width="20%" height="20%" z-index="2"/>
        <region id="rgLarge" top="0%" left="27%" width="73%" height="100%" z-index="2" />
        <region id="rgSmall" top="15%" left="5%" width="17%" height="30%" z-index="2"/>
      </layout>
    </layoutBase>
    <inputPatternBase>
      <compoundInputPattern id="putThatThere" mode="sequentiallyComplementary">
        <compoundInputPattern id="putThat" mode="complementary">
          <inputPattern id="putThatVoice" type="audioRecognition" src="dialogo.sgrs" label="p"/>
          <inputPattern id="pointInput" type="gestureRecognition" src="gestures.gml" label="point"/>
        </compoundInputPattern>
        <compoundInputPattern id="there" mode="complementary">
          <inputPattern id="thereVoice" type="audioRecognition" src="dialogo.sgrs" label="t"/>
          <inputPattern id="pointInput2" type="gestureRecognition" src="gestures.gml" label="point"/>
        </compoundInputPattern>
      </compoundInputPattern>
    </inputPatternBase>
    <propertyBase>
        <property id="varTemp"/>
    </propertyBase>
  </head>
  <body presentationMode="random">
    <scene id="scIntroducao">
      <visualTrack id="vtVideo" repeatCount="indefinite" region="rgVideoIntroducao">
        <media id="vidAlg" type="image" src="..." dur="60s"/>
      </visualTrack>
      <interactiveTrack id="itNavigationControlScenes" region="rgButtons">
        <autoStart media="imgBtnPause" />
        <autoStart media="imgBtnResume" />
        <autoStart media="imgBtnSeekadd5s" />
        <autoStart media="imgBtnSeeksub5s" />
        <media id="imgBtnPause" src="..." type="image" top="70%" left="10%" width="20%" height="30%"
            dur="indefinite"/>
        <media id="imgBtnResume" src="..." type="image" top="70%" left="30%" width="20%" height="30%"
            dur="indefinite"/>
        <media id="imgBtnSeekadd5s" src="..." type="image" top="70%" left="50%" width="20%" height="30%"
            dur="indefinite"/>
        <media id="imgBtnSeeksub5s" src="..." type="image" top="70%" left="70%" width="20%" height="30%"
            dur="indefinite"/>
        <media id="imgInsert" src="..." top="5%" type="image" left="0%" width="20%" height="50%" dur="indefinite"/>
        <media id="imgQuick" src="..." top="5%" type="image" left="80%" width="20%" height="50%" dur="indefinite"/>
        <link>onSelection imgInsert then start scInsert end</link>
        <link>onSelection imgQuick then start scQuick end</link>
```

```
        <link>onInstant vidAlg(45s) then start imgInsert || start imgQuick end</link>
        <link>onSelection imgBtnPause then pause vtVideo end</link>
        <link>onSelection imgBtnResume then resume vtVideo end</link>
        <link>onSelection imgBtnSeekadd5s then seek +5s vtVideo end</link>
        <link>onSelection imgBtnSeeksub5s then seek −5s vtVideo end</link>
      </interactiveTrack>
    </scene>
    <scene id="scInsert">
      <visualTrack id="vtbackground" region="rgBackGroundAlg" >
        <media id="imgBackgroundInsert" type="image" src="..." dur="indefinite"/>
      </visualTrack>
      <visualTrack id="vtViInsert" repeatCount="indefinite" region="rgLarge">
        <media id="imgVidInsert" type="video" src="..." dur="60s"/>
      </visualTrack>
      <visualTrack id="vtImInsert" region="rgSmall">
        <media id="imgAlgInsert" type="image" src="..." dur="indefinite"/>
      </visualTrack>
      <interactiveTrack id="itInteraction" region="rgInteractive" dur="indefinite">
        <autoStart media="imgBackToHome" />
        <media id="imgBackToHome" type="image" src="..." top="0%" left="0%" width="100%" height="80%"
            dur="indefinite"/>
        <media id="synSynteticWhere" type="application/ssml+xml" src="..." dur="3s"/>
        <link>onSelection imgBackToHome then start scIntroducao end</link>
        <link>onRecognition putThatThere with putThat>>vtViInsert,there>>vtImInsert then set varTemp=vtViInsert.region ; set
            vtViInsert.region=vtImInsert.region ; set vtImInsert.region=varTemp end</link>
        <link>onRecognition putThatThere with putThat>>vtImInsert,there>>vtViInsert then set varTemp=vtImInsert.region ; set
            vtImInsert.region=vtViInsert.region ; set vtViInsert.region=varTemp end</link>
        <link>onRecognition putThat>>vtViInsert or putThat>>vtImInsert then start synSynteticWhere end</link>
      </interactiveTrack>
    </scene>
     <scene id="scQuick">
      <visualTrack id="vtbackgroundQuick" region="rgBackGroundAlg">
        <media id="imgBackgroundQuick" type="image" src="..." dur="indefinite"/>
      </visualTrack>
      <visualTrack id="vtViQuick" region="rgLarge">
        <media id="imgVidQuick" type="video" src="..." dur="60s"/>
      </visualTrack>
      <visualTrack id="vtImQuick" region="rgSmall">
        <media id="imgAlgQuick" type="image" src="..." dur="indefinite"/>
      </visualTrack>
      <interactiveTrack id="itInteraction2" region="rgInteractive" dur="indefinite">
        <autoStart media="imgBackToHome2" />
        <media id="imgBackToHome2" type="image" src="..." top="0%" left="0%" width="100%" height="80%"
            dur="indefinite"/>
        <media id="synSynteticWhere2" type="application/ssml+xml" src="..." dur="3s"/>
        <link>onSelection imgBackToHome2 then start scIntroducao end</link>
        <link>onRecognition putThatThere with putThat>>vtViQuick,there>>vtImQuick then set varTemp=vtViQuick.region ; set
            vtViQuick.region=vtImQuick.region ; set vtImQuick.region=varTemp end</link>
        <link>onRecognition putThatThere with putThat>>vtImQuick,there>>vtViQuick then set varTemp=vtImQuick.region ; set
            vtImQuick.region=vtViQuick.region ; set vtViQuick.region=varTemp end</link>
        <link>onRecognition putThat>>vtViQuick or putThat>>vtImQuick then start synSynteticWhere2 end</link>
      </interactiveTrack>
    </scene>
  </body>
</storml>
```

A Figura 6.11 mostra os mesmo estados de exibição mostrados na Figura 6.9, porém, executados agora no *player* STorML. Observa-se na figura, o acréscimo das mídias responsáveis pelo controle temporal da trilha do vídeo introdutório (**A** e **B**) e da mídia responsável pelo retorno à cena inicial (**C** e **D**).

Espelhado na aplicação *"put-That-There"*, a qual é comumente utilizada em outros

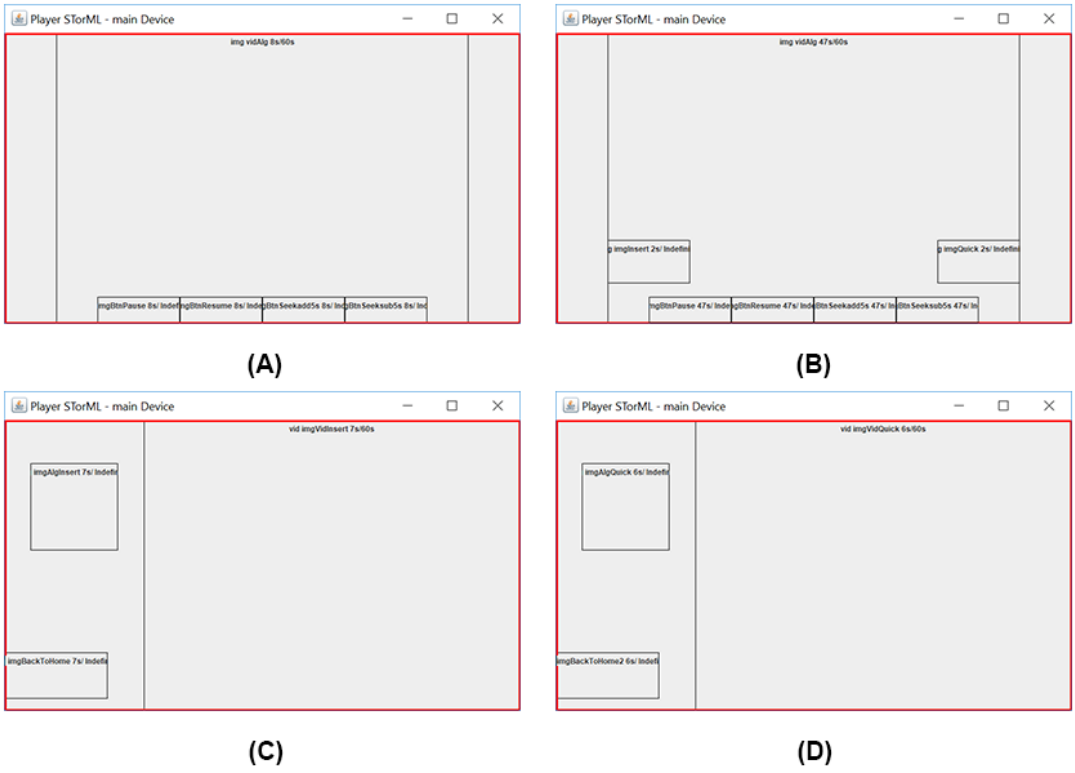


Figura 6.10: Ilustração da Aula em execução no *player* STorML.

estudos (GUEDES et al., 2016) , esse caso de uso mostrado aborda uma interação multimodal por meio de uma simulação. Para essa simulação, foram utilizados os eventos de apontamento do mouse combinado com os eventos de teclado definidos como padrões de entrada na base presente no cabeçalho (*inputPatternBase*). Observa-se que nessa base há apenas um padrão de entrada composto, definido e identificado como *putThatThere*. Esse padrão é composto por outros dois padrões de entrada também compostos os quais precisam ocorrer em sequência (*sequentiallyComplementary*) para que o padrão *putThatThere* seja totalmente reconhecido. O primeiro padrão composto identificado por *putThat* contém dois padrões de entrada simples que devem ocorrer simultaneamente (*complementary*). O primeiro padrão simples identificado por *putThatVoice* apesar de estar definido no documento STorML como uma linguagem de representação de voz (SGRS) que representa o comando de voz "put-that", como forma de simulação esse comando foi retratado como evento de teclado ao pressionar a tecla **"p"**, definido como valor do atributo *label*. O outro padrão simples identificado por *pointInput* representa um gesto de apontamento, e como forma de simulação ele foi retratado em STorML como apontamento de mouse (*label="point"*). O segundo padrão composto retrata os mesmos conceitos do primeiro, no entanto, o comando de voz "there" é retratado pela tecla **"t"** e o outro padrão simples tem

o mesmo significado de apontamento do mouse.

Com base nesses padrões de entrada, alguns elos de reconhecimento (*"onRecognition"*) foram definidos. Conforme citado anteriormente, o espectador poderá fazer a escolha de qual OA será exibido na maior região e na menor região do *layout* usado pelas cenas responsáveis pelos OAs do algoritmo *QuickSort* e *InsertSort*. Portanto, o elo, ao identificar os eventos correspondentes ao padrão de entrada *putThatThere*, fará a troca de regiões das trilhas responsáveis pela exibição de cada OA. Essa troca é realizada por meio de três ações sequenciais (*";"*) do tipo *"set"*, de modo a alterar as propriedades *"region"* das trilhas envolvidas. Para isso, a utilização de uma propriedade da apresentação, responsável por guardar temporariamente o valor das regiões envolvidas, foi definida para viabilizar a troca de regiões (*varTemp*). Vale destacar a atenção que o autor deve ter na hora de especificar esses relacionamentos causais, casos como *"onBegin x then stop x ; start x end"* podem causar *loops* infinitos e comprometer a aplicação. Além disso, ações de *set* em paralelo (||) podem ocasionar condições de corrida quando acessadas as mesmas propriedades (variáveis) nas diversas ações (p.ex, *"onBegin x then set w.region=y.region || set y.region=w.region"*). Dessa forma, o comportamento pode não ser o esperado pelo autor.

No caso da cena responsável pelo algoritmo *InsertSort*, o primeiro elo criado define que o padrão composto *putThatThere* tem sua associação detalhada por meio da palavra reservada *"with"*, especificando que o primeiro padrão composto interno identificado por *putThat* está associado a trilha *vtViInsert* (*putThat>>vtViInsert*). Essa associação especifica que o reconhecimento desse padrão de entrada composto deve acontecer no momento em que a trilha associada estiver ocorrendo, portanto, o evento de tecla **"p"** deve ser realizado no momento em que a trilha estiver sendo exibida e o apontamento do mouse deve ocorrer sobre a região onde a trilha está sendo exibida, tudo isso de forma simultânea, conforme especificado na base de padrões de entrada. O segundo padrão de entrada composto interno identificado por *there* está associado a trilha *vtImInsert* (*there>>vtImInsert*), portanto, segue o mesmo conceito do reconhecimento anteriormente detalhado, exceto que a tecla agora associada é **"t"** e o apontamento deve ocorrer sobre a trilha *vtImInsert*. Observa-se que para a troca de regiões, dois elos foram criados, pois a troca das trilhas pode ocorrer a partir de qualquer trilha desejada. Um terceiro elo foi criado para que no momento em que for identificado o padrão de entrada *putThat* associado a trilha *vtVi-*

Insert ou *vtImInsert* um objeto de mídia do tipo *application/ssml+xml* responsável pela pronúncia *"Where"* seja iniciado (*start*), seu início está diretamente relacionado ao sintetizador responsável por interpretar o conteúdo XML e executá-lo. A cena responsável pelo algoritmo *QuickSort* detém dos mesmos conceitos explicados para a cena *InsertSort*.

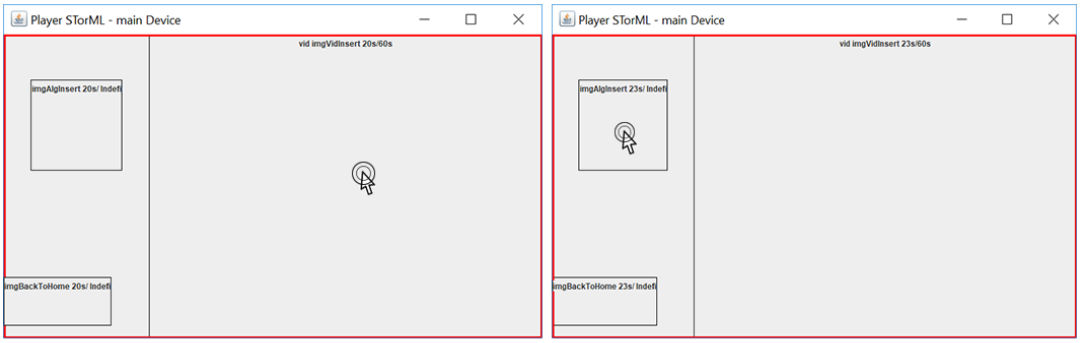


Figura 6.11: Simulação de interação multimodal em *player* STorML.

A Figura 6.11 ilustra, na cena responsável pelo algoritmo *InsertSort*, os dois momentos da interação multimodal. Em **(A)** o espectador faz o apontamento para o painel da trilha que se deseja "colocar" em outra região (*set*). Em simultâneo ao evento de apontamento, o espectador aperta a tecla **"p"** de seu teclado. Logo em seguida em **(B)**, o espectador faz o apontamento para o painel onde está sendo exibido a imagem ilustrativa do algoritmo presente na menor região e aperta a tecla "t" em simultâneo. Nesse momento, de acordo com o elo definido, ele "informa" ao *player* o desejo de trocar as regiões de cada trilha. Desta forma, a imagem do algoritmo passará a ser exibida na maior região e o vídeo passará a ser exibido na menor região, tudo isso sem qualquer interrupção na exibição dos objetos de mídias das trilhas. A Figura 6.12 mostra a cena após a troca de regiões das trilhas envolvidas no elo.

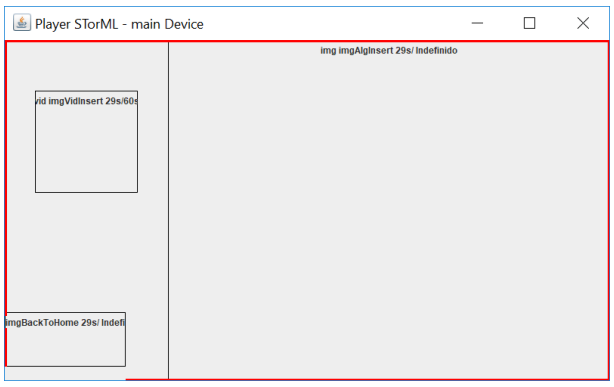


Figura 6.12: Ilustração da cena sobre *InsertSort* após interação multimodal.

6.2 TRANSFORMAÇÕES PRÉ-TRANSMISSÃO

6.2.1 RENDERIZAÇÃO DA INSTÂNCIA STorM

Sabendo que muito dos conteúdos disponibilizados para DOOH são não interativos e determinísticos, esse trabalho propõe uma maneira de renderizar essas partes na instância do modelo a fim de facilitar sua disseminação e, em alguns casos, possibilitar até a renderização total do conteúdo. Desta forma, o sincronismo entre os terminais será facilitado em arquiteturas *streaming* de modo a garantir à TV Corporativa uma real característica de TV, como o próprio nome sugere. Baseado nisso, abre-se a possibilidade de utilizar STorML em ambientes IPTV com QoS garantido, como, por exemplo, o trabalho realizado na RNP por meio do GT-IpêTeVê (2014).

Em aspectos gerais, a proposta é avaliar cena a cena verificando e tratando o conteúdo das suas trilhas de mídias contínuas (<*visualTrack*>, <*textTrack*> e <*audioTrack*>) determinísticas. Essas trilhas são determinísticas quando não apresentam na cena que as contém, elos (<*link*>) associados a essas trilhas ou a qualquer objeto de mídia interno à elas. Em um primeiro momento, a trilha determinística tem todos seus objetos de mídia unidos sequencialmente (**etapa 1**) resultando em apenas uma mídia. Após a etapa 1, há a sobreposição das mídias resultantes de cada trilha determinística de acordo com cada *layout* associado à trilha (**etapa 2**). Portanto, a renderização ocorre em camadas que são sobrepostas a um plano de fundo (*background*) que tenha o tamanho máximo do dispositivo de saída (*width="100%"* e *height="100%"*) levando em conta o atributo *z-index* da região a qual a trilha está associada, além do tamanho (*width* e *height*) e posicionamento (*top* e *left*) desta região. Todo o conteúdo do elemento <*fallback*> é ignorado na renderização sendo entregue ao terminal da forma que foi descrito via modo *pull*. A Figura 6.13 exibe o digrama de fluxo de como deve ser o tratamento de uma instância do modelo STorM (XML).

Vale ressaltar que, além das trilhas interativas(<*interactiveTrack*>), as trilhas adaptativas (<*adaptiveTrack*>) e as cenas adaptativas (<*adaptiveScene*>) são ignoradas nessa renderização por serem conteúdos não determinísticos. Com isso, a mídia resultante de todo o processo de renderização ficará com "espaços" sem preenchimento caso alguma região do *layout* seja destinada a um conteúdo não determinístico.

Além disso, para que a renderização ocorra, o renderizador deve ter cadastrado as re-

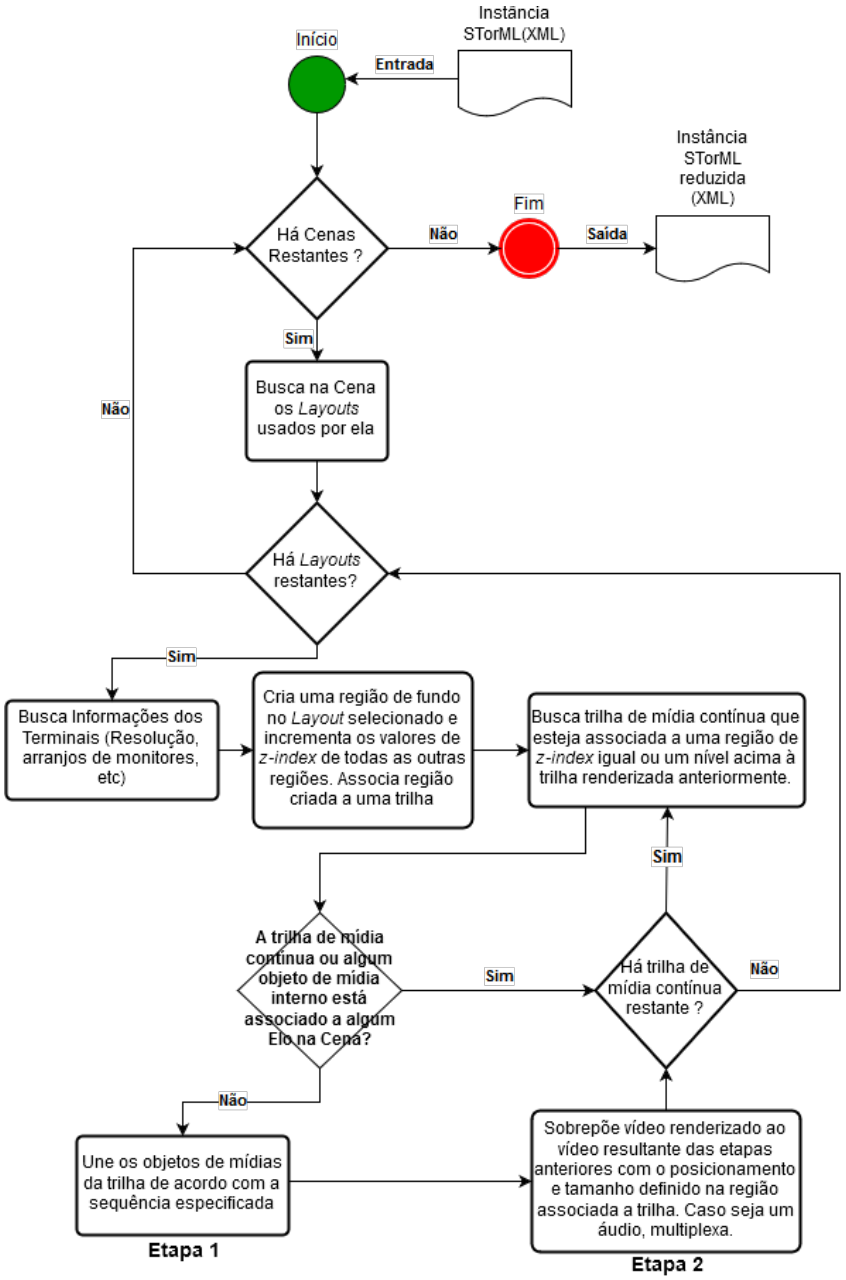


Figura 6.13: Diagrama de Fluxo do Processo de Renderização de uma instância do modelo

soluções padrões para cada razão de aspecto definida na base de *layouts* (<*layoutBase*>). Isso é necessário pois todos os valores de dimensão e posicionamento são tratados de forma percentual em STorML. Essa configuração será definida para cada corporação cadastrada no servidor de renderização. Desta forma, o renderizador terá as especificidades da plataforma IPTV e dos dispositivos de saída de cada corporação. O algoritmo 1 demonstra, em alto nível, essa proposta de transformação.

Como forma de exemplificar seu funcionamento, a Figura 6.14 demonstra uma instância, composta por uma cena associada a dois *layouts* alternativos, antes de ser tratada

Algoritmo 1: Processo de transformação da instância do modelo

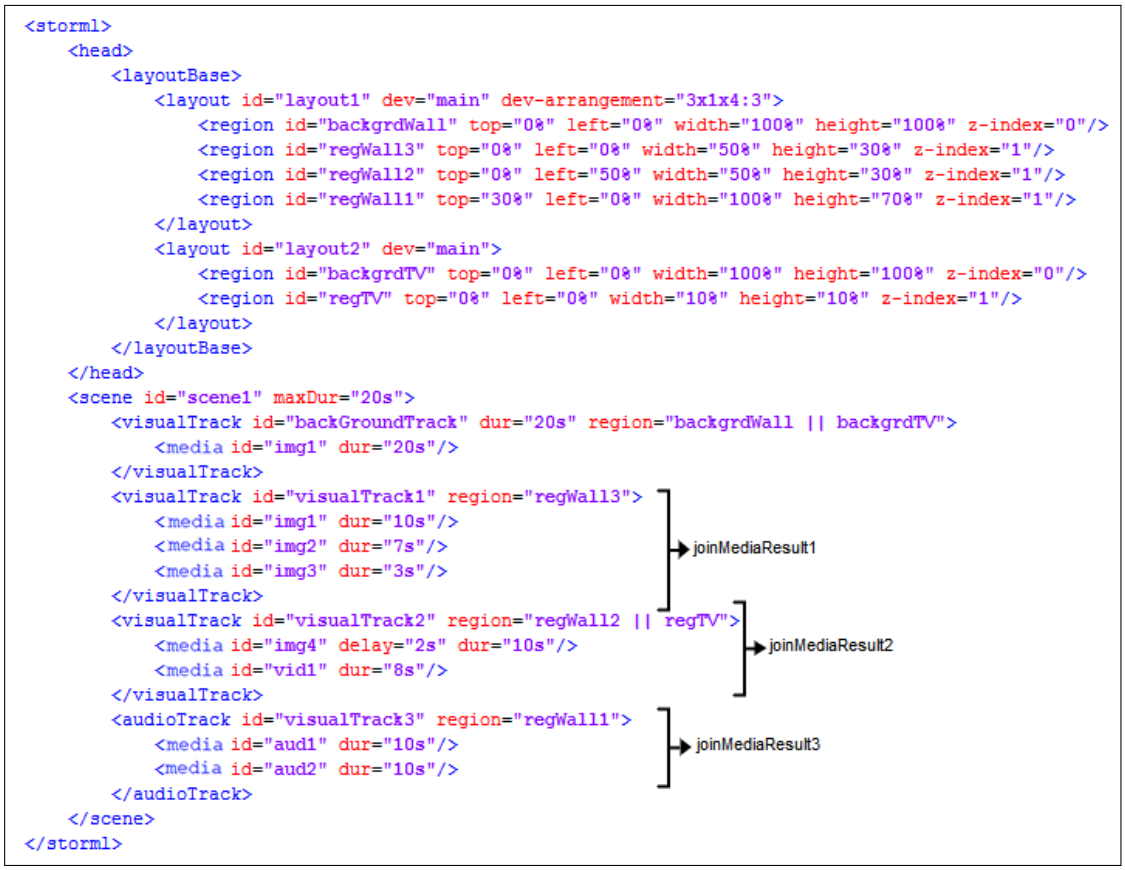
```
input  : full STorML instance (XML)
output: reduced STorML instance (XML)
while not at end of this document do
    currentScene ← get Next Scene;
    while Has Layout associated with currentScene do
        currentLayout ← get Layout;
        /* Background track is associated with a background region.  Background
           region has minimum z-index, maximum size(width='100%' and
           height='100%') and specific position (top='0%' and left='0%') of
           currentLayout.  Background track has the same duration of the
           currentScene                                                    */
        Increments the value of all z-index by one;
        backgroundRegion ← create background region in currentLayout;
        backgroundTrack ← create background track associated with backgroundRegion;
        mediaResult ← create still image into backgroundTrack with maximum
         currentLayout size;
        while currentScene has visualTrack or audioTrack do
            currentTrack ← get audioTrack or visualTrack with same z-index region or one
             level up of previous visualTrack;
            idRefList ← get currentTrack ID and all your internal media ID;
            top,left ← get currentTrack region position;
            height,width ← get currentTrack region size;
            if HasIDinLink(idRefList) then
                Ignore Track; // not deterministic track
                go to next Track;
            else
                mediaList ← get all internal media from currentTrack;
                joinMediaResult ← JoinMediaObjects (mediaList); // step 1
                if currentTrack is visualTrack then
                    mediaResult ← Render
                     (mediaResult,joinMediaResult,top,left,width,height); // step 2
                else
                    mediaResult ← Multiplex (mediaResult,joinMediaResult); // step
                     2
                end if
                MergeTrackElements (currentTrack,backgroundTrack);
            end if
        end while
    end while
end while
```

pelo algoritmo de transformação. As mídias que irão sofrer a unificação (**etapa 1**) estão destacadas nessa imagem.

Após a etapa 1, a instância ficará da forma representada pela Figura 6.15. As mídias que serão renderizadas e as trilhas que serão unificadas são mostradas nessa imagem.

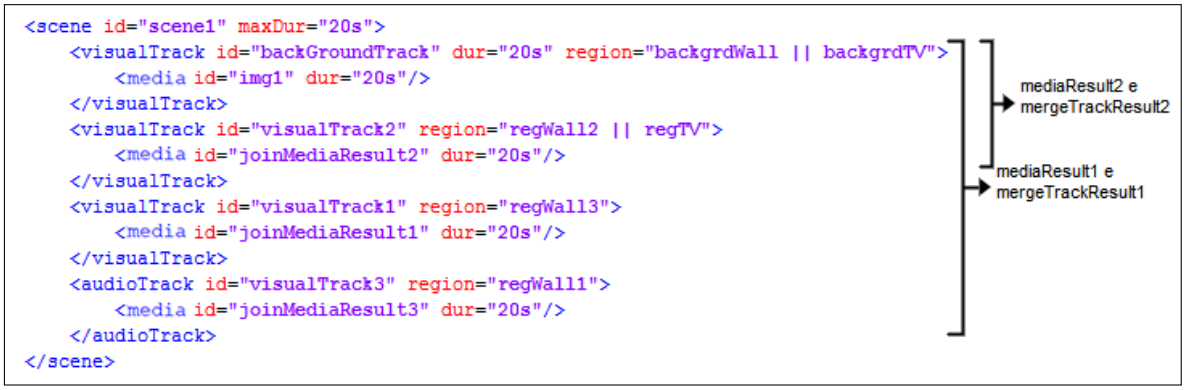
Após a etapa 2, a instância ficará da forma representada pela Figura 6.16. Essa é a instância final, resultado do processo de transformação para esse caso de uso.

Ao replicar esse processamento a diversas cenas pode-se ter uma instância totalmente compacta e pronta para ser entregue aos terminais por meio de *streaming*. Essa é uma al-




```
<storml>
    <head>
        <layoutBase>
            <layout id="layout1" dev="main" dev-arrangement="3x1x4:3">
                <region id="backgrdWall" top="0%" left="0%" width="100%" height="100%" z-index="0"/>
                <region id="regWall3" top="0%" left="0%" width="50%" height="30%" z-index="1"/>
                <region id="regWall2" top="0%" left="50%" width="50%" height="30%" z-index="1"/>
                <region id="regWall1" top="30%" left="0%" width="100%" height="70%" z-index="1"/>
            </layout>
            <layout id="layout2" dev="main">
                <region id="backgrdTV" top="0%" left="0%" width="100%" height="100%" z-index="0"/>
                <region id="regTV" top="0%" left="0%" width="10%" height="10%" z-index="1"/>
            </layout>
        </layoutBase>
    </head>
    <scene id="scene1" maxDur="20s">
        <visualTrack id="backGroundTrack" dur="20s" region="backgrdWall || backgrdTV">
            <media id="img1" dur="20s"/>
        </visualTrack>
        <visualTrack id="visualTrack1" region="regWall3">
            <media id="img1" dur="10s"/>
            <media id="img2" dur="7s"/>
            <media id="img3" dur="3s"/>
        </visualTrack>
        <visualTrack id="visualTrack2" region="regWall2 || regTV">
            <media id="img4" delay="2s" dur="10s"/>
            <media id="vid1" dur="8s"/>
        </visualTrack>
        <audioTrack id="visualTrack3" region="regWall1">
            <media id="aud1" dur="10s"/>
            <media id="aud2" dur="10s"/>
        </audioTrack>
    </scene>
</storml>
```

Figura 6.14: Instância de entrada do processo de transformação baseado na renderização




```
<scene id="scene1" maxDur="20s">
    <visualTrack id="backGroundTrack" dur="20s" region="backgrdWall || backgrdTV">
        <media id="img1" dur="20s"/>
    </visualTrack>
    <visualTrack id="visualTrack2" region="regWall2 || regTV">
        <media id="joinMediaResult2" dur="20s"/>
    </visualTrack>
    <visualTrack id="visualTrack1" region="regWall3">
        <media id="joinMediaResult1" dur="20s"/>
    </visualTrack>
    <audioTrack id="visualTrack3" region="regWall1">
        <media id="joinMediaResult3" dur="20s"/>
    </audioTrack>
</scene>
```

Figura 6.15: Cena da instância após a etapa 1

```
<scene id="scene1" maxDur="20s">
    <visualTrack id="mergeTrackResult1" dur="20s" region="backgrdTV">
        <media id="MediaResult1" dur="20s"/>
    </visualTrack>
    <visualTrack id="mergeTrakResult2" dur="20s" region="backgrdWall">
        <media id="mediaResult2" dur="20s"/>
    </visualTrack>
</scene>
```

Figura 6.16: Cena da instância após a etapa 2

ternativa aplicada e útil apenas a conteúdos determinísticos, os quais são muito utilizados atualmente em sistemas de TV Corporativa. De forma comparativa, podemos citar as tecnologias usadas na entrega de conteúdo em outras linguagens hipermídias, como o NCL. Nesse caso, como o serviço é destinado ao ambiente de TV Digital, o usuário pode sintonizar em um canal a qualquer instante da distribuição de dados, por isso utiliza-se uma estrutura cíclica para enviá-los, denominada carrossel de dados (DSM-CC) (ISO/IEC, 1998) . Nessa estrutura, os dados são enviados em módulos a cada ciclo, portanto, o receptor que se juntar ao serviço e ainda não tenha obtido o dado desejado, deve esperar por um novo ciclo de fluxo de dados.

Baseado nisso, estudos como o realizado em Josué, Moreno e Costa (2016), buscam, também, uma melhoria na transmissão de dados. Nesse estudo uma análise é feita em cima do conteúdo hipermídia NCL, buscando os eventos determinísticos e identificando os momentos em que as mídias presentes serão necessárias na apresentação. Com essa análise se torna possível escalonar a transmissão dos objetos de mídia, otimizando a construção do carrossel e levando a um menor uso de recursos em sistemas de comunicação multimídia por difusão (*broadcast*). Essa análise se baseia em uma estrutura denominada HTG (*Hypermedia Temporal Graph*) (COSTA, 2010) , a qual modela o comportamento temporal de aplicações hipermídia.

Diferente da melhoria citada no trabalho acima, nessa dissertação o processo de transformação assume a garantia de um QoS satisfatório da rede e ocorre antes da apresentação, pois trata diretamente a estrutura da instância. Portanto, por serem propostas que ocorrem em momentos distintos, uma proposta futura pode ser a utilização destes trabalhos de forma conjunta. Em um primeiro momento pode-se haver o processo de transformação da instância, conforme destacado nessa dissertação, e em um segundo momento pode-se haver um estudo do cronograma de apresentação e a ordem em que as cenas serão exibidas para enviá-las de maneira gradativa de forma a otimizar a transmissão e reduzir, assim, os recursos utilizados na transmissão destas aplicações.

6.2.2 SEPARAÇÃO DO CONTEÚDO *FALLBACK*

Para que o conteúdo usado como garantia em caso de indisponibilidade da rede (*fallback*) seja corretamente executado no terminal, ele deve ser separado do restante da instância STorM antes do processo de renderização especificado em 6.2.1. Para isso, o algoritmo

Algoritmo 2: Processo de separação do conteúdo *fallback*

```
input  : full STorML instance (XML)
output: fallback STorML instance (XML)
outputFile ← create new STorML file;
currentElement ← get first element from fallback;
while not at end of fallback do
    if currentElement is a Adaptive Scene then
        adaptiveCurrentScene ← get this Adaptive Scene;
        while Has global property in expr attribute of adaptiveCurrentScene do
            globalProperty ← get this global property;
            export globalProperty to Property Base Element into outputFile ;
        end while
        export adaptiveCurrentScene to body element outputFile ;
    else if currentElement is a Scene then
        currentScene ← get this Scene;
        while Has Layout associated with currentScene do
            currentLayout ← get this Layout;
            export currentLayout to layout base element into outputFile ;
        end while
        while Has Adaptive Track in currentScene do
            currentAdaptiveTrack ← get this Adaptive Track;
            while Has global property in expr attribute of currentAdaptiveTrack do
                globalProperty ← get this global property;
                export globalProperty to Property Base Element into outputFile ;
            end while
        end while
        while Has Input Pattern in link of currentScene do
            inputPattern ← get this Input Pattern;
            export inputPattern to Input Pattern Base Element into outputFile ;
        end while
        while Has global property in expr attribute of currentScene do
            globalProperty ← get this global property;
            export globalProperty to Property Base Element into outputFile ;
        end while
        export currentScene to body element outputFile;
    else
        currentElement ← get next element from fallback;
    end if
end while
```

2 deve ser utilizado de modo a buscar todos os *layouts* correspondentes, assim como os elementos presentes nos elos e as propriedades globais usadas nos conteúdos adaptativos.

7 CONCLUSÕES

O presente trabalho apresenta o modelo STorM e sua respectiva linguagem declarativa STorML. A proposta desse modelo foi buscar unir os benefícios das composições com semânticas incorporadas, como as utilizada em SMIL, com a expressividade do paradigma de causalidade utilizado em NCL levando em conta entidades com um maior nível de abstração. Esse alto nível de abstração se refere aos conceitos utilizados nas entidades, comumente vistos nas ferramentas audiovisuais usadas na área da comunicação social, a qual detém grande parte da mão de obra na autoria de conteúdos DOOH.

Além disso, apesar de ser um modelo voltado inicialmente para o domínio de DOOH, entende-se que sua utilização pode ser possível em outras áreas da comunicação social, devido a familiaridade com os conceitos usados em outros domínios pertencentes a mesma área. Destaca-se, ainda, a possibilidade em utilizar esse mesmo modelo para transmissões baseadas em *stream*, como IPTV e Digital Terrestre. Muitas corporações, como a RNP (REDE..., 2009) , possuem um controle de toda sua rede, o que proporciona um QoS adequado para difusão desse tipo de conteúdo sobre IP. Nesses casos, a transformação nas instâncias que apresentam conteúdos determinísticos será de grande importância, conforme processo detalhado na Seção 6.2.1.

Vale ressaltar que, como o foco desse trabalho foi expor as abordagens de sincronização e relacionamentos, algumas propriedades do objeto de mídia STorML foram omitidas. A proposta futura é tratar propriedades que envolvam estilos de apresentação por meio de arquivos referenciados, de forma parecida ao CSS3 (W3C, 2012a) utilizado pelo HTML (W3C, 2012b) . Pretende-se, portanto, fazer uso do atributo *class* presente nos elementos *Track*, sendo seu valor a URI do arquivo de estilo. Dentre essas propriedades que poderão ser utilizadas em um arquivo de estilo externo, destaca-se as características da fonte de um objeto de mídia texto, que foram omitidas nesse trabalho, cor ou imagem de fundo da trilha, efeitos de transições de mídias na trilha, transparência, rotação, texto rolante (*scrolling text*), dentre outros.

Fazendo uma breve comparação com outras linguagens hipermídias apresentadas nesse trabalho, foram destacadas algumas vantagens e desvantagens de STorML. Como vantagens cita-se:

- Agregação de facilidades presentes em duas linguagens distintas (SMIL x NCL):
  - Composições com semântica de sequenciamento incorporada. Trilhas visuais e de áudio se assemelham ao elemento <*seq*> de SMIL.
  - Composição cena com semântica de paralelismo incorporada. Se assemelha ao componente <*par*> de SMIL, no entanto, com escopo bem definido e restrito.
  - Composição trilha interativa com semântica baseada totalmente em relacionamentos causais. Se assemelha aos contextos de NCL, porém, com possibilidade de referenciar ou controlar, por meio de elos, qualquer outro elemento interno à cena a qual pertence. Em NCL, por exemplo, há a restrição de referenciar mídias em diferentes contextos apenas por meio de elementos <*port*>.
- Novo recurso de caminhamento entre objetos de mídias em trilhas de áudio e vídeo sem a necessidade de *templates*.
- Recurso de interações multimodais.
- Facilidade na criação de conteúdo interativo quando comparado ao HTML5+*JavaScript*.
- Recurso de âncoras temporais e espaciais sem a necessidade de declaração explícita de elemento (p.ex: Interface <*area*> usada em NCL).

Como desvantagens, destaca-se:

- Falta de aninhamento de entidades em vários níveis, conforme utilizado em SMIL e NCL.
- Pouco reuso quando comparada com NCL.
- Falta de composição semelhante ao <*excl*> de SMIL.
- Diferente de SMIL, STorML não detém do recurso de agendamento de exibição de conteúdo diretamente na linguagem.
- Expressividade inferior quando comparado ao HTML5+*JavaScript*.

7.0.1 CONTRIBUIÇÕES DA DISSERTAÇÃO

De forma resumida, as principais contribuições do presente trabalho são as seguintes:

- Criação de um modelo hipermídia e sua respectiva linguagem declarativa para autoria hipermídia em DOOH, possibilitando:
  - Representar conteúdos DOOH.
  - Facilitar autoria textual por profissionais especializados em DOOH.
  - Facilitar a navegação entre objetos de mídias por meio do caminhamento em trilhas.
- Definição de uma nova entidade denominada trilha.
- Autoria de conteúdos interativos abordando suas múltiplas modalidades.
- Autoria de conteúdos com *layouts* que suportam *videowall*.
- Representação XML de conteúdo *QR Code* de modo a possibilitar sua utilização como objetos de mídia e consequente geração dinâmica.
- Proposta de transformação em documentos STorML, por meio de renderizações sucessivas pré-transmissão, com o objetivo de favorecer sua difusão.

7.0.2 TRABALHOS FUTUROS

O presente trabalho abriu possibilidades de aprofundamento nos seguintes assuntos, passíveis de exploração em trabalhos futuros:

1. Desenvolver uma ferramenta gráfica que automatizará a criação de conteúdos STorML.
2. Especificar *Templates* que facilitarão não só o reúso de *layouts* como também de relacionamentos.
3. Propor a conversão STorML para outras representações, como o HTML5 e SceneSync, possibilitando uma maior imersão da linguagem.
4. Elaborar uma análise mais abrangente na identificação de conteúdos determinísticos internos a trilha interativa, talvez com o auxílio de estruturas como o HTG.

5. Realizar pesquisas com autores em busca de indicadores de legibilidade, usabilidade e eficácia da linguagem.

6. Realizar estudo com programadores a fim de se obter indicadores de produtividade na autoria de conteúdos para DOOH utilizando STorML em comparação com a autoria direta por meio de linguagens imperativas.

7. Propor novas sintaxes para especificação da instância STorM ao invés do XML utilizado em STorML, de modo a facilitar a autoria ou a integração com tecnologias como HTML5 (p.ex, JSON).

8. Implementar um *workflow* que abordará tanto a etapa de criação do conteúdo, como a programação semanal do conteúdo de acordo com os grupos de terminais disponíveis, além da maneira em como difundir toda a programação aos terminais.

9. Implementar transformação proposta na Seção 6.2.1. Estudos e testes breves realizados em laboratório apontam a ferramenta FFmpeg como opção nesse processo de renderizações sucessivas das partes determinísticas da instância.

10. Estudar uma metodologia em como disponibilizar o conteúdo STorML destinado aos dispositivos secundários.

REFERÊNCIAS

4YOUSEE. *4YouSee - Digital Signage Technology*. 2014. Disponível em: <https://www.4yousee.com.br/>.

A-SMIL. *SMIL for Digital Signage*. 2010. Disponível em: <http://www.a-smil.org/>.

ABNT. *Norma NBR 15606-2 "Televisão digital terrestre - Codificação de dados e especificações de transmissão para radiofusão digital - Parte 2: Ginga-NCL para receptores fixos e móveis - Linguagem de aplicação XML para codificação de aplicações"*. Rio de Janeiro, Brasil, 2016. Disponível em: <<https://www.abntcatalogo.com.br/norma.aspx?ID=351837>>.

ANTONACCI, M. J. et al. Ncl: Uma linguagem declarativa para especificação de documentos hipermídia na web. *VI Simpósio Brasileiro de Sistemas Multimídia e Hipermídia-SBMídia2000, Natal, Rio Grande do Norte*, 2000.

AVARO, O. et al. Mpeg-4 systems: Overview. *Signal Processing: Image Communication*, Elsevier, v. 15, n. 4, p. 281–298, 2000.

BOLT, R. A. *"Put-that-there": Voice and gesture at the graphics interface*. [S.l.]: ACM, 1980.

BUSSON, A. J. G. et al. Scenesync: A hypermedia authoring language for temporal synchronism of learning objects. In: ACM. *Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web*. [S.l.], 2016. p. 175–182.

CARNEIRO, M. P. *Um player web para redes de ecrãs públicos*. Tese (Doutorado), 2013.

CAZENAVE, F.; QUINT, V.; ROISIN, C. Timesheets.js: when smil meets html5 and css3. In: ACM. *Proceedings of the 11th ACM symposium on Document engineering*. [S.l.], 2011. p. 43–52.

COSTA, R. *Integração e Interoperabilidade de Documentos MPEG-4 e NCL*. Tese (Doutorado) — Dissertação de Mestrado, Departamento de Informática, PUC-Rio, Rio de Janeiro, Brasil, 2005.

COSTA, R. M. de R. *Controle do Sincronismo Temporal de Aplicações Hipermídia*. Tese (Doutorado) — Programa de Pós-graduação em Informática da PUC-Rio, Agosto 2010.

DAYARATHNA, M.; WITHANA, A.; SUGIURA, K. Infoshare: design and implementation of scalable multimedia signage architecture for wireless ubiquitous environments. *Wireless Personal Communications*, Springer, v. 60, n. 1, p. 3–27, 2011.

DUPIN, F. *Digital signage:the right information in all the right places*. Geneva, Switzerland, 2011.

FREITAS, D. O. d. et al. Uma arquitetura de software baseada em serviços para sinalização digital interativa. Universidade Federal de São Carlos, 2013.

GT-IPêTEVê. *GT-IpêTeVê: Serviço de Televisão IP de Alcance Global*. 2014. Disponível em: <<http://gt-ipeteve.ice.ufjf.br.>>.

GUEDES, Á. L. et al. Extending ncl to support multiuser and multimodal interactions. In: ACM. *Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web*. [S.l.], 2016. p. 39–46.

GUEDES, A. L. V. et al. Specification of multimodal interactions in ncl. In: ACM. *Proceedings of the 21st Brazilian Symposium on Multimedia and the Web*. [S.l.], 2015. p. 181–187.

IADEA. *IADEA Player*. 2000. Disponível em: <<http://www.iadea.com/>>.

INC, M. S. *Digital Signage for Everyone*. 2011. Disponível em: <<http://www.digitalsignage.com/>>.

INNES. *INNES Digital Signage and IPTV*. December 2005. Disponível em: <<http://www.innes.pro/>>.

ISO. *ISO 8601 "Data elements and interchange formats – Information interchange – Representation of dates and times"*. Genebra, Suíça, 2004. Disponível em: <<https://www.iso.org/standard/40874.html>>.

ISO/IEC. *Information technology – Generic coding of moving pictures and associated audio information - Part 6: Extensions for DSM-CC*. [S.l.], 1998.

ITU. *Recomendação UIT-T H.761 "Nested context language (NCL) and Ginga-NCL"*. Genebra, Suíça, 2014. Disponível em: <<https://www.itu.int/rec/T-REC-H.761-201411-I>>.

ITU. *H.765 Packaged IPTV application (widget) service*. Geneva, Switzerland, 2015.

ITU. *H.781 Digital signage: Functional architecture*. Geneva, Switzerland, 2015.

ITU-GROUP16. *HSTP-DS-UCIS Digital signage: Use-cases of interactive services*. Geneva, Switzerland, 2014.

JOSUÉ, M. I. P.; MORENO, M. F.; COSTA, R. M. d. R. Hypermedia content transmission plan: Managing the broadcast/multicast delivery. In: ACM. *Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web*. [S.l.], 2016. p. 87–90.

MACKINTOSH, A. et al. *Zxing, open source library to read 1D/2D barcodes*. 2012.

NETO, C. S. *Autoria de Documentos Hipermídia Orientada a Templates*. Tese (Doutorado) — Doctoral Thesis, Informatics Department of PUC-Rio., 2010.

NIGAY, L.; COUTAZ, J. Multifeature systems: The care properties and their impact on software design. *Intelligence and multimodality in multimedia interfaces*, 1997.

POPAI. *Screen-Media Formats - A POPAI Digital Signage Standards Document*. Los Angeles, 2009.

POPAI. *POPAI - The Global Association For Marketing at Retail*. 2016. Disponível em: <<http://www.popaibrasil.com.br/>>.

PRO, A. P. 2.0 user guide. *Changing clip attributes, publication page and*, p. 157–158, 2005.

REDE Nacional de Pesquisa - RNP. mar 2009. Disponível em: <<http://www.rnp.br/en/backbone/index.php>>.

SAMSUNG. *MagicInfo Samsung*. 1999. Disponível em: <<http://displaysolutions.samsung.com/solutions/by-applications/magicinfo>>.

SLONNEGER, K.; KURTZ, B. L. *Formal syntax and semantics of programming languages*. [S.l.]: Addison-Wesley Reading, 1995.

SOARES G.F. LIMA, C. S. N. L. *NCL 3.1 enhanced dtv profile*. 2010.

SOARES, L. *G e Barbosa SDJ Programando em NCL 3.0. Editora Campus*. [S.l.]: Elsevier, 2009.

SOARES, L. et al. *Towards the ncl raw profile*. 2010.

SOARES, L. F. G.; BARBOSA, S. D. J. Programando em ncl 3.0. *Desenvolvimento de Aplicações para o Middleware Ginga, TV digital e Web*, v. 1, 2012.

SOARES, L. F. G.; RODRIGUES, R. F. Nested context model 3.0: Part 1–ncm core. *Monografias em Ciência da Computação do Departamento de Informática, PUC-Rio*, n. 18/05, 2005.

SOON, T. J. Qr code. *Synthesis Journal*, v. 2008, p. 59–78, 2008.

SPYROU, E.; IAKOVIDIS, D.; MYLONAS, P. *Semantic Multimedia Analysis and Processing*. [S.l.]: CRC Press, 2014.

STALDER, U. Digital out-of-home media: Means and effects of digital media in public space. In: _____. *Pervasive Advertising*. London: Springer London, 2011. p. 31–56. ISBN 978-0-85729-352-7. Disponível em: <<https://doi.org/10.1007/978-0-85729-352-7_2>>.

W3C. *Synchronized Multimedia Integration Language (SMIL 3.0)*. December 2008. Disponível em: <<http://www.w3.org/TR/REC-smil/>>.

W3C. *Cascading Style Sheets Basic User Interface Model Level 3*. 2012. Disponível em: <<http://www.w3.org/TR/css3-ui/>>.

W3C. *HTML5 : a vocabulary and associated APIs for HTML and XHTML*. 2012. Disponível em: <<http://www.w3.org/TR/html5/>>.

W3C. *Sourcing In-band Media Resource Tracks from Media Containers into HTML*. 2015. Disponível em: <<https://dev.w3.org/html5/html-sourcing-inband-tracks/>>.

WANT, R.; SCHILIT, B. N. Interactive digital signage. *Computer*, IEEE Computer Society, Los Alamitos, CA, USA, v. 45, n. undefined, p. 21–24, 2012. ISSN 0018-9162.

XIBO. *XIBO - Digital Signage Content Management System*. 2010. Disponível em: <<http://xibo.org.uk/>>.

ZXing Barcode Contents. September 2016. Disponível em: <<https://github.com/zxing/zxing/wiki/Barcode-Contents>>.

Apêndice A - A XSD STorML

```
1  <schema xmlns="http://www.w3.org/2001/XMLSchema"
       elementFormDefault="qualified" attributeFormDefault="unqualified" >
2     <xs:complexType name="adaptive_scene_t">
3        <xs:choice>
4           <xs:element name="meta" maxOccurs="unbounded">
5            <xs:complexType>
6              <xs:attribute name="name" type="xs:string" use="required"/>
7              <xs:attribute name="value" type="xs:string" use="required"/>
8            </xs:complexType>
9          </xs:element>
10         <xs:element name="scene" type="scene_t" maxOccurs="unbounded"/>
11       </xs:choice>
12       <xs:attribute name="id" type="xs:string" use="required"/>
13        <xs:attribute name="maxDur" type="xs:string"/>
14       <xs:attribute name="refer" type="xs:string"/>
15    </xs:complexType>
16     <xs:complexType name="adpative_track_t">
17       <xs:choice>
18          <xs:element name="meta" maxOccurs="unbounded">
19           <xs:complexType>
20             <xs:attribute name="name" type="xs:string" use="required"/>
21             <xs:attribute name="value" type="xs:string" use="required"/>
22           </xs:complexType>
23         </xs:element>
24         <xs:element name="visualTrack" type="visual_track_t"
              maxOccurs="unbounded"/>
25         <xs:element name="audioTrack" type="audio_track_t"
              maxOccurs="unbounded"/>
26         <xs:element name="textTrack" type="text_track_t"
```

```
                maxOccurs="unbounded"/>
27          <xs:element name="interactiveTrack" type="interactive_track_t"
                maxOccurs="unbounded"/>
28        </xs:choice>
29        <xs:attribute name="id" type="xs:string" use="required"/>
30         <xs:attribute name="maxDur" type="xs:string"/>
31        <xs:attribute name="refer" type="xs:string"/>
32        <xs:attribute name="keepTiming" type="xs:boolean"/>
33    </xs:complexType>
34    <xs:simpleType name="interactive_type_t">
35        <xs:restriction base="xs:string">
36          <xs:enumeration value="application/x-itu-iptv-widget"/>
37          <xs:enumeration value="application/x-ncl"/>
38          <xs:enumeration value="text/html"/>
39        </xs:restriction>
40    </xs:simpleType>
41     <xs:complexType name="interactive_track_t">
42        <xs:choice>
43            <xs:element name="meta" maxOccurs="unbounded">
44             <xs:complexType>
45                 <xs:attribute name="name" type="xs:string" use="required"/>
46               <xs:attribute name="value" type="xs:string" use="required"/>
47             </xs:complexType>
48          </xs:element>
49           <xs:element name="media" maxOccurs="unbounded">
50             <xs:complexType>
51                <xs:attribute name="id" type="xs:string" use="required"/>
52                <xs:attribute name="dur" type="xs:string"/>
53                    <xs:attribute name="delay" type="xs:string" />
54                    <xs:attribute name="src" type="xs:string"
                          use="required"/>
55                    <xs:attribute name="opacity" type="xs:string"/>
```

```
56                  <xs:attribute name="rotate" type="xs:string"/>
57                  <xs:attribute name="type" type="xs:string"
                        use="required"/>
58                  <xs:attribute name="refer" type="xs:string"/>
59                  <xs:attribute name="top" type="xs:string"/>
60                  <xs:attribute name="left" type="xs:string"/>
61                  <xs:attribute name="width" type="xs:string"/>
62                  <xs:attribute name="height" type="xs:string"/>
63                  <xs:attribute name="clipBegin" type="xs:string"/>
64                  <xs:attribute name="clipEnd" type="xs:string"/>
65              </xs:complexType>
66          </xs:element>
67              <xs:element name="autoStart" maxOccurs="unbounded">
68                  <xs:complexType>
69                      <xs:attribute name="media" type="xs:string"
                            use="required"/>
70                  <xs:attribute name="id" type="xs:string"/>
71                  </xs:complexType>
72              </xs:element>
73              <xs:element name="link" maxOccurs="unbounded">
74                  <xs:complexType>
75                      <xs:simpleContent>
76                          <xs:extension base="xs:string">
77                          <xs:attribute name="id" type="xs:string" />
78                          </xs:extension>
79                      </xs:simpleContent>
80                  </xs:complexType>
81              </xs:element>
82          </xs:choice>
83          <xs:attribute name="id" type="xs:string" use="required"/>
84           <xs:attribute name="region" type="xs:string"/>
85          <xs:attribute name="delay" type="xs:string"/>
```

```
86        <xs:attribute name="expr" type="xs:string"/>
87        <xs:attribute name="contentType" type="interactive_type_t"/>
88        <xs:attribute name="src" type="xs:string"/>
89        <xs:attribute name="refreshTime" type="xs:string"/>
90        <xs:attribute name="maxDur" type="xs:string"/>
91        <xs:attribute name="bgColor" type="xs:string"/>
92        <xs:attribute name="bgImage" type="xs:string"/>
93        <xs:attribute name="refer" type="xs:string"/>
94        <xs:attribute name="startMode" type="start_mode_t"/>
95        <xs:attribute name="repeatCount" type="xs:unsignedInt"/>
96        <xs:attribute name="sondLevel" type="xs:string"/>
97    </xs:complexType>
98    <xs:complexType name="text_track_t">
99        <xs:choice>
100           <xs:element name="meta" maxOccurs="unbounded">
101            <xs:complexType>
102               <xs:attribute name="name" type="xs:string" use="required"/>
103              <xs:attribute name="value" type="xs:string" use="required"/>
104            </xs:complexType>
105         </xs:element>
106       </xs:choice>
107       <xs:attribute name="id" type="xs:string" use="required"/>
108        <xs:attribute name="region" type="xs:string"/>
109       <xs:attribute name="delay" type="xs:string"/>
110       <xs:attribute name="expr" type="xs:string"/>
111       <xs:attribute name="contentType" type="interactive_type_t"/>
112       <xs:attribute name="src" type="xs:string"/>
113       <xs:attribute name="dur" type="xs:string"/>
114       <xs:attribute name="bgColor" type="xs:string"/>
115       <xs:attribute name="bgImage" type="xs:string"/>
116       <xs:attribute name="refer" type="xs:string"/>
117       <xs:attribute name="startMode" type="start_mode_t"/>
```

```
118       <xs:attribute name="repeatCount" type="xs:unsignedInt"/>
119     </xs:complexType>
120      <xs:complexType name="audio_track_t">
121        <xs:choice>
122            <xs:element name="meta" maxOccurs="unbounded">
123             <xs:complexType>
124                 <xs:attribute name="name" type="xs:string" use="required"/>
125               <xs:attribute name="value" type="xs:string" use="required"/>
126             </xs:complexType>
127          </xs:element>
128           <xs:element name="media" maxOccurs="unbounded">
129             <xs:complexType>
130               <xs:attribute name="id" type="xs:string" use="required"/>
131               <xs:attribute name="dur" type="xs:string"/>
132                   <xs:attribute name="delay" type="xs:string" />
133                   <xs:attribute name="src" type="xs:string"
                        use="required"/>
134                   <xs:attribute name="opacity" type="xs:string"/>
135                   <xs:attribute name="rotate" type="xs:string"/>
136                   <xs:attribute name="type" type="xs:string"
                        use="required"/>
137                   <xs:attribute name="refer" type="xs:string"/>
138                   <xs:attribute name="top" type="xs:string"/>
139                   <xs:attribute name="left" type="xs:string"/>
140                   <xs:attribute name="width" type="xs:string"/>
141                   <xs:attribute name="height" type="xs:string"/>
142                   <xs:attribute name="clipBegin" type="xs:string"/>
143                   <xs:attribute name="clipEnd" type="xs:string"/>
144             </xs:complexType>
145          </xs:element>
146       </xs:choice>
147       <xs:attribute name="id" type="xs:string" use="required"/>
```

```
148         <xs:attribute name="region" type="xs:string" use="required"/>
149         <xs:attribute name="delay" type="xs:string"/>
150         <xs:attribute name="expr" type="xs:string"/>
151         <xs:attribute name="dur" type="xs:string"/>
152         <xs:attribute name="refer" type="xs:string"/>
153         <xs:attribute name="startMode" type="start_mode_t"/>
154         <xs:attribute name="repeatCount" type="xs:unsignedInt"/>
155         <xs:attribute name="soundLevel" type="xs:string"/>
156         <xs:attribute name="transition" type="xs:string"/>
157         <xs:attribute name="contentType" type="xs:string" />
158         <xs:attribute name="src" type="xs:string"/>
159     </xs:complexType>
160     <xs:simpleType name="start_mode_t">
161         <xs:restriction base="xs:string">
162             <xs:enumeration value="synced"/>
163             <xs:enumeration value="deferred"/>
164         </xs:restriction>
165     </xs:simpleType>
166      <xs:complexType name="visual_track_t">
167         <xs:choice>
168             <xs:element name="meta" maxOccurs="unbounded">
169                 <xs:complexType>
170                     <xs:attribute name="name" type="xs:string" use="required"/>
171                     <xs:attribute name="value" type="xs:string" use="required"/>
172                 </xs:complexType>
173             </xs:element>
174             <xs:element name="media" maxOccurs="unbounded">
175                 <xs:complexType>
176                     <xs:attribute name="id" type="xs:string" use="required"/>
177                     <xs:attribute name="dur" type="xs:string"/>
178                         <xs:attribute name="delay" type="xs:string" />
179                         <xs:attribute name="src" type="xs:string"
```

```
                    use="required"/>
180                 <xs:attribute name="opacity" type="xs:string"/>
181                 <xs:attribute name="rotate" type="xs:string"/>
182                 <xs:attribute name="type" type="xs:string"
                    use="required"/>
183                 <xs:attribute name="refer" type="xs:string"/>
184                 <xs:attribute name="top" type="xs:string"/>
185                 <xs:attribute name="left" type="xs:string"/>
186                 <xs:attribute name="width" type="xs:string"/>
187                 <xs:attribute name="height" type="xs:string"/>
188                 <xs:attribute name="clipBegin" type="xs:string"/>
189                 <xs:attribute name="clipEnd" type="xs:string"/>
190             </xs:complexType>
191         </xs:element>
192     </xs:choice>
193     <xs:attribute name="id" type="xs:string" use="required"/>
194      <xs:attribute name="region" type="xs:string" use="required"/>
195     <xs:attribute name="transition" type="xs:string"/>
196     <xs:attribute name="delay" type="xs:string"/>
197     <xs:attribute name="expr" type="xs:string"/>
198     <xs:attribute name="startMode" type="start_mode_t"/>
199     <xs:attribute name="dur" type="xs:string"/>
200     <xs:attribute name="refer" type="xs:string"/>
201     <xs:attribute name="repeatCount" type="xs:unsignedInt"/>
202     <xs:attribute name="soundLevel" type="xs:string"/>
203     <xs:attribute name="src" type="xs:string"/>
204     <xs:attribute name="contentType" type="xs:string"/>
205     <xs:attribute name="bgColor" type="xs:string"/>
206     <xs:attribute name="bgImage" type="xs:string"/>
207 </xs:complexType>
208  <xs:complexType name="scene_t">
209     <xs:sequence>
```

```
210             <xs:element name="meta" maxOccurs="unbounded">
211              <xs:complexType>
212                  <xs:attribute name="name" type="xs:string" use="required"/>
213                <xs:attribute name="value" type="xs:string" use="required"/>
214              </xs:complexType>
215          </xs:element>
216          <xs:element name="visualTrack" type="visual_track_t"
                 maxOccurs="unbounded"/>
217          <xs:element name="audioTrack" type="audio_track_t"
                 maxOccurs="unbounded"/>
218          <xs:element name="textTrack" type="text_track_t"
                 maxOccurs="unbounded"/>
219          <xs:element name="interactiveTrack" type="interactive_track_t"
                 maxOccurs="unbounded"/>
220          <xs:element name="adpativeTrack" type="adpative_track_t"
                 maxOccurs="unbounded"/>
221      </xs:sequence>
222      <xs:attribute name="id" type="xs:string" use="required"/>
223       <xs:attribute name="repeatCount" type="xs:unsignedInt"/>
224       <xs:attribute name="refer" type="xs:string"/>
225      <xs:attribute name="maxDur" type="xs:unsignedInt"/>
226  </xs:complexType>
227  <xs:simpleType name="presentation_mode_t">
228      <xs:restriction base="xs:string">
229          <xs:enumeration value="sequential"/>
230          <xs:enumeration value="random"/>
231      </xs:restriction>
232  </xs:simpleType>
233  <xs:complexType name="body_t">
234      <xs:choice>
235          <xs:element name="scene" type="scene_t"
                 maxOccurs="unbounded"/>
```

```
236         <xs:element name="adaptiveScene" type="adaptive_scene_t"
                maxOccurs="unbounded"/>
237         <xs:element name="fallback" type="body_t" maxOccurs="5"/>
238       </xs:choice>
239       <xs:attribute name="presentationMode" type="presentation_mode_t"/>
240     </xs:complexType>
241     <xs:simpleType name="mode_coumpound_input_pattern_t">
242       <xs:restriction base="xs:string">
243         <xs:enumeration value="complementary"/>
244         <xs:enumeration value="sequentiallyComplementary"/>
245         <xs:enumeration value="redundant"/>
246       </xs:restriction>
247     </xs:simpleType>
248     <xs:complexType name="compound_input_pattern_t">
249       <xs:choice>
250         <xs:element name="compoundInputPattern"
              type="compound_input_pattern_t" maxOccurs="unbounded"/>
251         <xs:element name="inputPattern" maxOccurs="unbounded">
252           <xs:attribute name="id" type="xs:string" use="required"/>
253           <xs:attribute name="type" type="type_recognition_t"/>
254           <xs:attribute name="src" type="xs:string"/>
255           <xs:attribute name="label" type="xs:string"/>
256         </xs:element>
257       </xs:choice>
258       <xs:attribute name="id" type="xs:string" use="required"/>
259       <xs:attribute name="mode" type="mode_coumpound_input_pattern_t"/>
260       <xs:attribute name="refer" type="xs:string"/>
261       <xs:attribute name="maxDur" type="xs:string"/>
262     </xs:complexType>
263     <xs:complexType name="input_pattern_base_t">
264       <xs:choice>
265         <xs:element name="compoundInputPattern"
```

```
                type="compound_input_pattern_t" maxOccurs="unbounded"/>
266           <xs:element name="inputPattern" maxOccurs="unbounded">
267              <xs:attribute name="id" type="xs:string" use="required"/>
268              <xs:attribute name="type" type="xs:string" use="required"/>
269              <xs:attribute name="src" type="xs:string" use="required"/>
270              <xs:attribute name="label" type="xs:string"/>
271           </xs:element>
272        </xs:choice>
273     </xs:complexType>
274     <xs:complexType name="layout_t">
275        <xs:sequence>
276           <xs:element name="meta" maxOccurs="unbounded">
277              <xs:complexType>
278               <xs:attribute name="name" type="xs:string" use="required"/>
279                 <xs:attribute name="value" type="xs:string" use="required"/>
280              </xs:complexType>
281           </xs:element>
282           <xs:element name="region" maxOccurs="unbounded">
283              <xs:complexType>
284                 <xs:attribute name="id" type="xs:string" use="required"/>
285                 <xs:attribute name="left" type="xs:string"/>
286                 <xs:attribute name="top" type="xs:integer"/>
287                 <xs:attribute name="width" type="xs:string"/>
288                 <xs:attribute name="height" type="xs:string"/>
289                 <xs:attribute name="z-index" type="xs:integer"/>
290                 <xs:attribute name="refer" type="xs:string"/>
291              </xs:complexType>
292           </xs:element>
293        </xs:sequence>
294        <xs:attribute name="id" type="xs:string" use="required"/>
295        <xs:attribute name="dev" type="xs:string"/>
296        <xs:attribute name="dev-arrangement" type="xs:string"/>
```

```
297     </xs:complexType>
298     <xs:complexType name="layout_base_t">
299       <xs:sequence>
300         <xs:element name="layout" type="layout_t"
                maxOccurs="unbounded"/>
301       </xs:sequence>
302     </xs:complexType>
303     <xs:complexType name="value_base_t">
304       <xs:sequence>
305         <xs:element name="value" maxOccurs="unbounded">
306           <xs:complexType>
307             <xs:attribute name="id" type="xs:string" use="required"/>
308             <xs:attribute name="init" type="xs:string"/>
309           </xs:complexType>
310         </xs:element>
311       </xs:sequence>
312     </xs:complexType>
313     <xs:complexType name="head_t">
314       <xs:sequence>
315         <xs:element name="layoutBase" type="layout_base_t"/>
316         <xs:element name="inputPatternBase" type="input_pattern_base_t"/>
317         <xs:element name="valueBase" type="value_base_t"/>
318       </xs:sequence>
319     </xs:complexType>
320     <xs:complexType name="storml_t">
321       <xs:sequence>
322         <xs:element name="head" type="head_t"/>
323         <xs:element name="body" type="body_t"/>
324       </xs:sequence>
325     </xs:complexType>
326     <xs:element name="storml" type="storml_t">
327   </xs:schema>
```

Apêndice B - A XSD QR CODE

```
1  <?xml version="1.0" enconding="UTF-8"?>
2  <xs:schema xmlns:xs=”http://www.w3.org/2001/XMLSchema”
       elementFormDefault=”qualified” attributeFormDefault=”unqualified”>
3      <xs:element name=”qrCode”>
4          <xs:complexType>
5              <xs:attribute name=”color” type=”xs:string” default=”black”/>
6              <xs:attribute name=”backgroundColor” type=”xs:string”
                   default=”white”/>
7              <xs:element name=”id” type=”xs:string”/>
8              <xs:choice>
9                  <xs:element name=”wifi”>
10                     <xs:complexType>
11                         <xs:sequence>
12                             <xs:element name=”s” type=”xs:string”/>
13                             <xs:element name=”t” type=”xs:string”/>
14                             <xs:element name=”p” type=”xs:string”/>
15                             <xs:element name=”h” type=”xs:string”/>
16                         </xs:sequence>
17                     </xs:complexType>
18                 </xs:element>
19                 <xs:element name=”phoneNumber”>
20                     <xs:complexType>
21                         <xs:sequence>
22                             <xs:element name=”number” type=”xs:int”/>
23                         </xs:sequence>
24                     </xs:complexType>
25                 </xs:element>
26                 <xs:element name=”sms”>
27                     <xs:complexType>
```

```
28                  <xs:sequence>
29                      <xs:element name="smsTo" type="xs:string"/>
30                      <xs:element name="message" type="xs:string"/>
31                  </xs:sequence>
32              </xs:complexType>
33          </xs:element>
34          <xs:element name="url">
35              <xs:complexType>
36                  <xs:sequence>
37                      <xs:element name="title" type="xs:string"/>
38                      <xs:element name="url" type="xs:string"/>
39                  </xs:sequence>
40              </xs:complexType>
41          </xs:element>
42          <xs:element name="location">
43              <xs:complexType>
44                  <xs:sequence>
45                      <xs:element name="geo" type="xs:string"/>
46                  </xs:sequence>
47              </xs:complexType>
48          </xs:element>
49          <xs:element name="vCard">
50              <xs:complexType>
51                  <xs:sequence>
52                      <xs:element name="begin" type="xs:string"/>
53                      <xs:element name="n" type="xs:string"/>
54                      <xs:element name="tel" type="xs:string"/>
55                      <xs:element name="email" type="xs:string"/>
56                      <xs:element name="org" type="xs:string"/>
57                      <xs:element name="version" type="xs:string"/>
58                      <xs:element name="end" type="xs:string"/>
59                  </xs:sequence>
```

```
60                  </xs:complexType>
61              </xs:element>
62              <xs:element name="calendarEvent">
63                  <xs:complexType>
64                      <xs:sequence>
65                          <xs:element name="begin" type="xs:string"/>
66                          <xs:element name="summary" type="xs:string"/>
67                          <xs:element name="dtStart" type="xs:string"/>
68                          <xs:element name="dtEnd" type="xs:string"/>
69                          <xs:element name="location" type="xs:string"/>
70                          <xs:element name="description" type="xs:string"/>
71                          <xs:element name="end" type="xs:string"/>
72                      </xs:sequence>
73                  </xs:complexType>
74              </xs:element>
75              <xs:element name="text">
76                  <xs:complexType>
77                      <xs:sequence>
78                          <xs:element name="idDescription"
                                type="xs:string"/>
79                      </xs:sequence>
80                  </xs:complexType>
81              </xs:element>
82              <xs:element name="emailAdress">
83                  <xs:complexType>
84                      <xs:sequence>
85                          <xs:element name="mailTo" type="xs:string"/>
86                      </xs:sequence>
87                  </xs:complexType>
88              </xs:element>
89          </xs:choice>
90      </xs:complexType>
```

```
91        </xs:element>
92  </xs:schema>
```

Apêndice C - CASOS DE USO DO XML PARA QR CODE

```
<qrCode color="black" backgroundColor="white" xmlns="">
    <emailAddress id="emailAdress">
        <mailTo>moreno@ice.ufjf.br</mailTo>
    </emailAddress>
</qrCode>
```

```
<qrCode color="black" backgroundColor="white" xmlns="">
    <text id="descricao">
        <content>Site Google</content>
    </text>
</qrCode>
```

```
<qrCode color="black" backgroundColor="white" xmlns="">
    <calendarEvent id="eventoCopa2018">
        <begin>VEVENT</begin>
        <summary>IpeTeVe - Brasil vs. Colombia</summary>
        <dtStart>20160907T004500Z</dtStart>
        <dtEnd>20160907T004500Z</dtEnd>
        <location>http://glo.bo/2bKYjDJ</location>
        <description>Copa do Mundo 2018</description>
        <end>VEVENT</end>
    </calendarEvent>
</qrCode>
```

```
<qrCode color="black" backgroundColor="white" xmlns="">
    <contactInformation id="contatoMoreno">
        <begin>VCARD</begin>
        <n>Marcelo Moreno</n>
        <tel>+553221023311</tel>
```

```
6         <email>moreno@ice.ufjf.br</email>
7         <org>UFJF</org>
8         <version>http://glo.bo/2bKYjDJ</version>
9         <end>VCARD</end>
10     </contactInformation>
11 </qrCode>
```

```
1  <qrCode color=”black” backgroundColor=”white” xmlns=””>
2     <location id=”rodoviariaJF”>
3         <geo>-21.775917,-43.372299</geo>
4     </location>
5  </qrCode>
```

```
1  <qrCode color=”black” backgroundColor=”white” xmlns=””>
2     <phoneNumber id=”andreTel”>
3         <tel>+553221023315</tel>
4     </phoneNumber>
5  </qrCode>
```

```
1  <qrCode color=”black” backgroundColor=”white” xmlns=””>
2     <sms id=”smsToAndre”>
3         <smsTo>+5532988678295</smsTo>
4         <message>Prezado, estou interessado na tecnologia IPTV. Favor
              me contactar por SMS.</message>
5     </sms>
6  </qrCode>
```

```
1  <qrCode color=”black” backgroundColor=”white” xmlns=””>
2     <url id=”urlIpeteve”>
3         <content>https://ipeteve.com.br</content>
4     </url>
5  </qrCode>
```

```
1  <?xml version="1.0" enconding="UTF-8"?>
2  <qrCode color=”black” backgroundColor=”white” xmlns=””>
```

```
3     <url id="urlGloboAPP">
4         <content>market://details?id=com.globo.globotv</content>
5     </url>
6 </qrCode>
```

```
1 <qrCode color="black" backgroundColor="white" xmlns="">
2     <wifi id="wifiMorenoMac">
3         <s>MorenoMacMini</s>
4         <t>WPA</t>
5         <p>funciona</p>
6         <h>false</h>
7     </wifi>
8 </qrCode>
```

Apêndice D - SINTAXES

⟨*expr*⟩ ::= ⟨*condition*⟩ {⟨*lop*⟩ ⟨*expr*⟩}?

⟨*condition*⟩ ::= ⟨*whitespace*⟩ ⟨*string*⟩ ⟨*whitespace*⟩ ⟨*rop*⟩ ⟨*whitespace*⟩ ⟨*string*⟩

⟨*lop*⟩ ::= `&&` | `||`

⟨*rop*⟩ ::= `<` | `>` | `<=` | `>=` | `==` | `!=`

⟨*string*⟩ ::= ‘character sequence’

⟨*whitespace*⟩ ::= ‘ ’ ⟨*whitespace*⟩ | ‘ ’

Sintaxe D.1: Sintaxe BNF do atributo *expr*.

⟨*regList*⟩ ::= ⟨*regExpr*⟩ | `(` ⟨*regList*⟩ `)` ⟨*lop*⟩ `(` ⟨*regList*⟩ `)`

⟨*regExpr*⟩ ::= ⟨*string*⟩ ⟨*lop*⟩ ⟨*string*⟩

⟨*lop*⟩ ::= `&&` | `||`

⟨*string*⟩ ::= ‘character sequence’

Sintaxe D.2: Sintaxe BNF do atributo *region*.

⟨*link*⟩ ::= ⟨*condList*⟩ `then` ⟨*actList*⟩ `end;`

⟨*condList*⟩ ::= ⟨*condList*⟩ ⟨*withparams*⟩ {⟨*lop*⟩ ⟨*condList*⟩}? | ⟨*condition*⟩ {⟨*lop*⟩ ⟨*condList*⟩}? ;

⟨*condition*⟩ ::= ⟨*condname*⟩ ⟨*perspective*⟩ ⟨*withparams*⟩ | ⟨*assessment*⟩ ⟨*withparams*⟩ | `onRecognition` ⟨*association*⟩ ⟨*withparams*⟩ ;

⟨*assessmt*⟩ ::= ⟨*assesexpr*⟩ ⟨*rop*⟩ ⟨*assesexpr*⟩ ;

⟨*assesexpr*⟩ ::= ⟨*perspective*⟩ {`+` ⟨*string*⟩}? | ⟨*string*⟩ {`+` ⟨*perspective*⟩}? ;

⟨*condname*⟩ ::= `onBegin` | `onBeginAttribution` | `onEndAttribution` | `onEnd` | `onPause` | `onResume` | `onSelection` | `onInstant` ;

⟨*actList*⟩ ::= ⟨*actlist*⟩ ⟨*withparams*⟩ {⟨*aop*⟩ ⟨*actlist*⟩}? | ⟨*action*⟩ {⟨*aop*⟩ ⟨*actlist*⟩}? ;

⟨*action*⟩ ::= ⟨*actionset*⟩ ⟨*perspective*⟩ ⟨*withparams*⟩ | `set` ⟨*perspective*⟩ `=` ⟨*string*⟩ ⟨*withparams*⟩ | `set` ⟨*perspective*⟩ `=` ⟨*perspective*⟩ ⟨*withparams*⟩ | `seek` ⟨*seekValue*⟩ ⟨*perspective*⟩ ;

⟨*actionset*⟩ ::= `pause` | `resume` | `start` | `stop` ;

⟨*seekValue*⟩ ::= `next` | `first` | `last` | `prev` | ⟨*time*⟩ | `+`⟨*time*⟩ | `-`⟨*time*⟩ ;

⟨*time*⟩ ::= ⟨*number*⟩`s` ;

⟨*bounds*⟩ ::= ⟨*number*⟩`,`⟨*number*⟩`,`⟨*number*⟩`,`⟨*number*⟩ ;

⟨*number*⟩ ::= ⟨*digit*⟩ | ⟨*number*⟩ ⟨*digit*⟩ ;

⟨*digit*⟩ ::= `0` | `1` | `2` | `3` | `4` | `5` | `6` | `7` | `8` | `9` ;

⟨*association*⟩ ::= ⟨*ref*⟩ {`>>` ⟨*perspective*⟩}? ;

⟨*perspective*⟩ ::= ⟨*ref*⟩ {`.` ⟨*ref*⟩}* {`(`⟨*time*⟩`)`}? | ⟨*ref*⟩ {`.` ⟨*ref*⟩}* {`[`⟨*bounds*⟩`]`}?;

⟨*withparams*⟩ ::= { `with` {⟨*parameter*⟩`,`}* ⟨*parameter*⟩{`,`}? }? ;

⟨*parameter*⟩ ::= ⟨*perspective*⟩ `=` ⟨*string*⟩ | ⟨*ref*⟩ `>>` ⟨*perspective*⟩ ;

⟨*string*⟩ = ‘character sequence’ | "character sequence" ;

⟨*aop*⟩ ::= `||` | {`;`}? ;

⟨*lop*⟩ ::= `and` | `or` ;

⟨*rop*⟩ ::= `==` | `<` | `>` | `!=` | `<=` | `>=` ;

⟨*ref*⟩ ::= {`a-z` | `A-Z` | `_` | `:`} | {`a-z` | `A-Z` | `_` | `:` | `.` | `-` | `0-9`}+ ;

Sintaxe D.3: Sintaxe BNF do Elo (<*link*>).