Capítulo 1

Desenvolvimento de aplicações interativas integrando objetos reais e virtuais com o uso do padrão MPEG-V

Estêvão Bissoli Saleme e Celso Alberto Saibel Santos

Abstract

This chapter is about the development of interoperable interactive applications based on the ISO/IEC 23005 - MPEG-V Standard. This standard encompasses specifications regarding device capabilities, sensory effects, virtual world object characteristics and data format for interactions devices. The chapter also introduces the MPEG Metadata, a library to support the development of MPEG-V compliant applications. Finally, the library is used to implement two interactive applications using real world sensors and actuators.

Resumo

A norma ISO/IEC 23005, também conhecida como MPEG-V, provê uma arquitetura e especifica representações de informação para conectar entidades do mundo virtual (associadas a cenários de jogos, simuladores e aplicações) e do mundo real (ex. sensores e atuadores). Com foco na questão do desenvolvimento de aplicações interativas interoperáveis é introduzido o padrão MPEG-V, que envolve especificações sobre capacidade de dispositivos, efeitos sensoriais, características de objetos e formatos de dados para interação entre dispositivos, além da biblioteca MPEG Metadata e duas aplicações práticas que usam de sensores e atuadores combinados com padrão.

1.1. Introdução

Soluções heterogêneas direcionadas a um mesmo problema são a todo instante desenvolvidas pela indústria e pela academia. Cho (2010) relata um exemplo desta situação em parques temáticos envolvendo alta tecnologia, como Disney Land e Universal Studios. Ambos produzem conteúdos com múltiplos efeitos sensoriais, mas aplicam soluções distintas, inviabilizando a reutilização destes conteúdos em ambientes diferentes daqueles

onde foram produzidos. Esse é um problema típico de integração de aplicações em ambientes de interação avançada, em que tanto a entrada como saída de dados não utilizam um formato comum para interoperabilidade (Santos et al., 2015b).

Tendo em vista a questão da interoperabilidade em aplicações interativas, diversos trabalhos têm apresentado soluções baseadas no padrão MPEG-V. Waltl et al. (2013) introduziram um conjunto de ferramentas[1] para autoria (SEVino) e consumo (SESim e SEMP) de conteúdo multimídia audiovisual combinado com efeitos sensoriais. Também com foco na renderização de efeitos sensoriais sincronizados com conteúdo multimídia, mas também no reuso do renderizador de efeitos em outros tipos de aplicações (ex. jogos e aplicativos iDTV), Saleme e Santos (2015) criaram a plataforma PlaySEM[2], que permite o reuso das anotações compatíveis com o padrão MPEG-V produzidas pela ferramenta SEVino. Kim (2013) desenvolveu o Sensible Media Simulator (SMURF), uma ferramenta de autoria que possui uma interface Web desenvolvida com Adobe Flex e que permite a renderização de efeitos sensoriais usando um *plugin* Adobe Flash. Ela também é compatível com a SEVino e foi utilizada por Kim e Joo (2014) para construir uma aplicação voltada ao entretenimento em automóveis modernos, aproveitando-se da suas infraestruturas compostas por dispositivos de efeitos sensoriais (aquecedores, telas LCD, etc.) e sensores. No ambiente de realidade aumentada, Choi e Park (2013) desenvolveram um agregador de sensores (umidade, luz, temperatura, vento, etc.) compatível com o MPEG-V para coleta de informações do ambiente. Ao adentrar no ambiente com um *smartphone*, o agregador provê informações para o aplicativo criar uma imagem 4D do ambiente com as informações obtidas. Recentemente, Santos et al. (2015a) apresentaram uma abordagem orientada a eventos[3] para a construção de ambientes interativos combinados com efeitos sensoriais descritos através do padrão MPEG-V.

Os trabalhos citados ilustram que o desenvolvimento de aplicações interativas interoperáveis tendo como base o padrão MPEG-V tem despertado interesse da comunidade de multimídia (ou, MulSeMedia, de acordo com Ghinea et al., 2014) nos últimos anos. Nesse capítulo, questões envolvendo a estrutura do padrão MPEG-V, especificações sobre capacidade de dispositivos, efeitos sensoriais, características de objetos e formatos de dados para interação entre dispositivos são apresentadas. Além disso, aplicações práticas com uso de sensores e atuadores baseadas no padrão são exploradas. O capítulo está estruturado em 3 tópicos principais, que apresentam, em sequência, o padrão ISO/IEC 23005 MPEG-V, a biblioteca java MPEG *Metadata* e o desenvolvimento de aplicações interativas. A biblioteca facilita a ligação entre o XML no formato MPEG-V e objetos na aplicação. As aplicações interativas consistem em (*i*) integrar um sensor acelerômetro remoto com uma aplicação *desktop* (R2V - *Real World to Virtual World*) e (*ii*) integrar metadados de aplicações multimídia contendo efeitos sensoriais com atuadores tais como luz, vento e vibração (V2R - *Virtual World to Real World*).

[1]Ferramentas SEVino, SESim e SEMP disponíveis em http://selab.itec.aau.at/software-and-services/

[2]Demonstração do PlaySEM disponível em https://youtu.be/aka4jBZcoKk

[3]Demonstração do trabalho de Santos et al. (2015a) disponível em http://youtu.be/a5QPsB2zKpk

1.2. O padrão ISO/IEC 23005 - MPEG-V

Esse seção apresenta a arquitetura do MPEG-V e todas as 7 partes que o compõem: Parte 1 – arquitetura; parte 2 – informações de controle; parte 3 – efeitos sensoriais; parte 4 – características de objetos do mundo virtual; parte 5 – formato de dados para dispositivos de interação; parte 6 – ferramentas e tipos comuns e; parte 7 – conformidade e software de referência.

1.2.1. MPEG-V Parte 1 - Arquitetura

A especificação da arquitetura proposta para o MPEG-V é apresentada no documento que descreve o padrão ISO/IEC 23005-1 (Kim e Han, 2014). A Figura 1.1, extraída de Kim et al. (2012), ilustra a representação dessa arquitetura, bem como o relacionamento entre as partes que formam o conjunto de padrões MPEG-V.

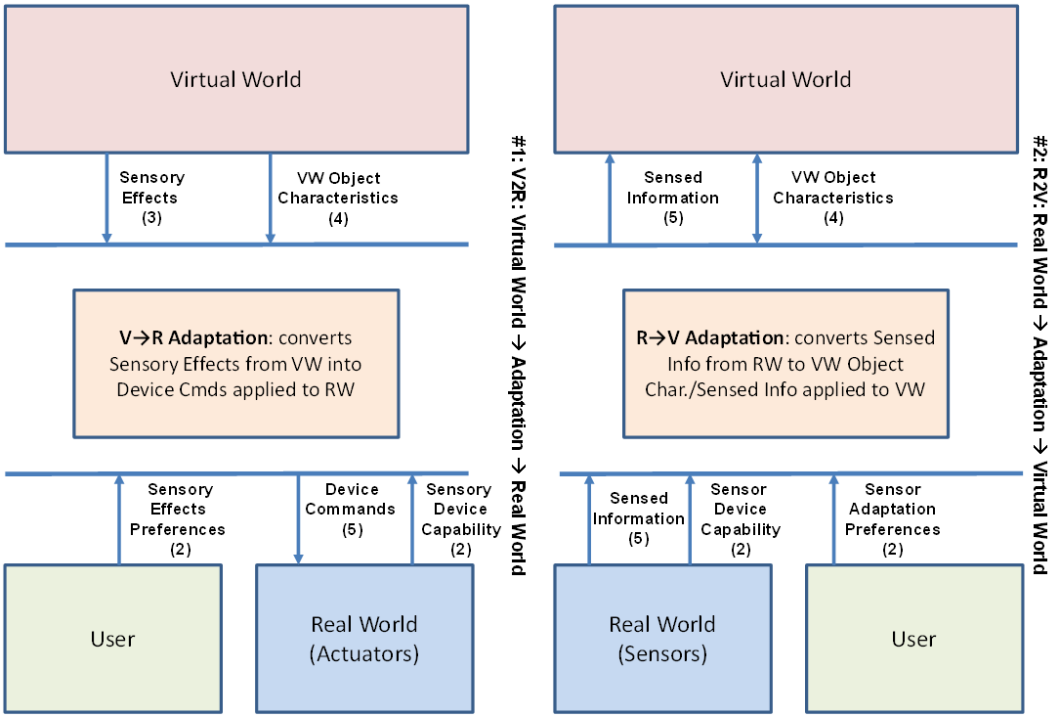

Figura 1.1. Arquitetura do MPEG-V, V2R e R2V. Fonte: (Kim et al., 2012)

Do mundo virtual para o mundo real (V2R), informações dos efeitos sensoriais (parte 3) e características de objetos virtuais (parte 4) podem ser encaminhadas para o ambiente físico do usuário, para que sejam reproduzidas. O motor de adaptação (V->R *adaptation*), que serve para converter descrições de efeitos sensoriais (parte 3) em comandos para dispositivos (parte 5), configurar preferências de efeitos sensoriais do usuário (parte 2) e descrever a capacidades dos dispositivos (parte 2), não está no âmbito da norma, preocupada somente com a representação das informações. Os atuadores podem ser compatíveis diretamente com a parte 3, não sendo necessária a conversão de um efeito em N comandos.

Do mundo real para o mundo virtual (R2V), informações obtidas do ambiente real do usuário (parte 5) e suas preferências (parte 2) podem ser encaminhadas para o

ambiente virtual. Por exemplo, informações de temperatura do ambiente real podem ser reproduzidas no ambiente virtual. Há ainda a possibilidade de interoperar informações entre mundos virtuais distintos (V2V), a partir da parte 4, como por exemplo, tornar compatíveis informações de características visuais de um avatar de um dado jogo com outro avatar em um jogo distinto. O motor de adaptação desse cenário também não é abrangido pela norma.

1.2.2. MPEG-V Parte 2 - Informações de Controle

A parte 2 do MPEG-V (ISO/IEC 23005-2) aborda aspectos normativos das informações de controle, incluindo descrição capacidade do dispositivo e informações de preferência do usuário (Han e Yoon, 2012). Seu escopo, representado na Figura 1.2, cobre interfaces entre o motor de adaptação, descrições de capacidade de sensores e atuadores no mundo real e informações de preferências dos usuários sobre esses dispositivos. A sintaxe e a semântica definidas na norma estabelecem o mecanismo para prover a interoperabilidade entre dispositivos e sistemas. Dessa forma, a parte 2 provê meios de fornecer informações extras para o motor de adaptação (que não é normativo) interagir com as demais partes do MPEG-V prevendo a capacidade disponível e as preferências configuradas pelo usuário.

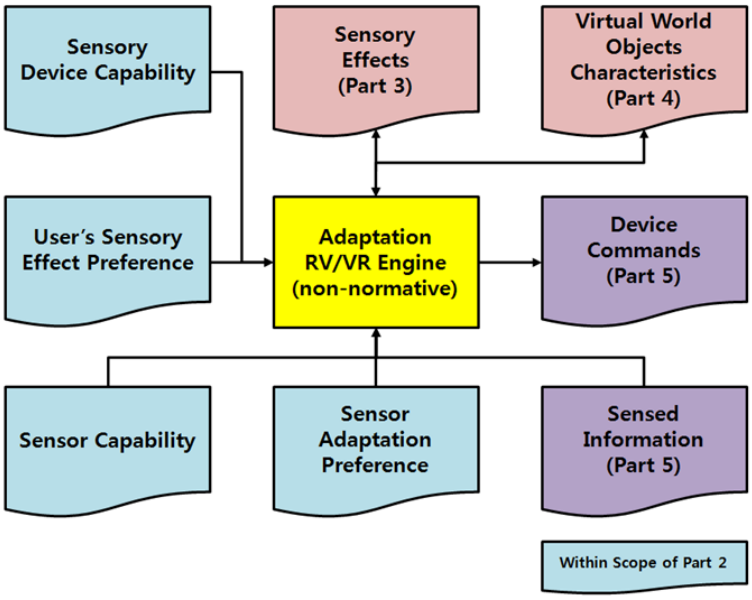

Figura 1.2. Escopo da parte 2 do MPEG-V destacada na cor azulada. Fonte: (Han e Yoon, 2012)

Essa parte especifica uma linguagem de marcação para descrição de controle da informação (CIDL - *Control Information Description Language*) que permite descrever uma estrutura básica de informações de controle (Kim e Han, 2014). Como parte da CIDL, quatro vocabulários apoiam a representação das informações:

- Vocabulário para descrição das capacidades dos dispositivos (DCDV - *Device Capability Description Vocabulary*). Exemplo: a velocidade máxima de vento suportada pelo dispositivo e o número de níveis de rotação;

- Vocabulário para descrição das capacidades dos sensores (SCDV - *Sensor Capability Description Vocabulary*). Exemplo: a temperatura mínima e máxima suportada por um sensor de temperatura;

- Vocabulário de preferências de efeitos sensoriais do usuário (SEPV - *User's Sensory Effect Preference Vocabulary*). Exemplo: o usuário quer restringir o efeito sensorial de vibração ao nível médio, pois acredita que o nível máximo pode ser perigoso para uma criança;

- Vocabulário para preferência do usuário para adaptação dos sensores (SAPV - *Sensor Adaptation Preference Vocabulary*). Exemplo: o usuário deseja restringir a temperatura ambiente entre 25C° e 50C°, pois não gosta de sentir frio. Mesmo que o sensor tenha capacidade, a temperatura não extrapolará os limites estabelecidos pela preferência do usuário.

Os exemplos seguintes reproduzidos a partir da norma (Han e Yoon, 2012) mostram o uso da CIDL, em conjunto dos vocabulários DCDV, SCDV, SEPV e SAPV.

```
<cidl:SensoryDeviceCapability xsi:type="dcdv:ScentCapabilityType"
id="scent01" maxIntensity="5" numOfLevels="2"
    locator="urn:mpeg:mpeg-v:01-
SI-LocationCS-NS:center">
  <dcdv:Scent>urn:mpeg:mpeg-v:01-SI-ScentCS-NS:rose</dcdv:Scent>
</cidl:SensoryDeviceCapability>
```

O exemplo anterior da CIDL com o DCDV ilustra a descrição das capacidades de um dispositivo que produz aromas. O dispositivo identificado por "scent01", é capaz de atingir a intensidade máxima de um aroma de 5 ml por hora, com dois níveis de controle. O dispositivo está localizado no centro e é capaz de produzir somente perfumes de rosa.

```
<cidl:SensorDeviceCapability
    xsi:type="scdv:TemperatureSensorCapabilityType" id="TS001"
    maxValue="120" minValue="-20" numOfLevels="1400" offset="1.0"
    unit="celsius">
  <cidl:Accuracy xsi:type="cidl:ValueAccuracy" value="0.1"/>
  <scdv:Location>
    <mpegvct:X>1.0</mpegvct:X>
    <mpegvct:Y>1.0</mpegvct:Y>
    <mpegvct:Z>-1.0</mpegvct:Z>
  </scdv:Location>
</cidl:SensorDeviceCapability>
```

O exemplo anterior da CIDL com o SCDV mostra a descrição das capacidades de um sensor de temperatura identificado por "TS001", capaz de obter informações de temperatura que variam entre -20C° e 120C°, perfazendo 1200 níveis de temperatura. A acurácia é de 0,1C°. A informação detectada é recebida na localização 1,00, 1,00, -1,00 (x, y e z).

```
<cidl:USPreference xsi:type="sepv:LightPrefType" activate="true"
unit="urn:mpeg:mpeg-v:01-CI-UnitTypeCS-NS:lux" maxIntensity="300">
<sepv:UnfavorableColor>
:mpeg:mpeg-v:01-SI-ColorCS-NS:alice_blue
</sepv:UnfavorableColor>
</cidl:USPreference>
```

O exemplo anterior da CIDL com o SEPV mostra a descrição da preferência do usuário por um efeito de luz de no máximo 300 *lux*. A cor "alice_blue" não é desejada.

```
<cidl:SAPreference xsi:type="sapv:TemperatureAdaptationPrefType"
    id="TSAP001" activate="true" minValue="0" maxValue="100"
    sensorAdaptationMode="strict" numOfLevels="10"/>
```

O exemplo anterior da CIDL com o SAPV mostra a descrição de preferência do usuário para adaptação dos sensores de temperatura. A temperatura desejada está entre 0C° e 100C° e terá somente 10 níveis entre os valores mínimo e máximo. Quando os valores da preferência não estão dentro do intervalo das capacidades dos dispositivos, eles devem ser ajustados adequadamente pelo motor de adaptação.

1.2.3. MPEG-V Parte 3 – Efeitos Sensoriais

A parte 3 do MPEG-V (ISO/IEC 23005-3) estrutura a descrição de informações sensoriais para estimular outros sentidos além da visão ou audição, como por exemplo, termoceptores, mecanoceptores e quimioceptores (Choi e Kim, 2012). Ela especifica a sintaxe e a semântica para descrever informações sensoriais sendo aplicável para melhorar a qualidade da experiência dos usuários enquanto consumidores de conteúdo multimídia/multissensoriais. As informações sensoriais podem ser encaminhadas diretamente para os dispositivos atuadores no mundo real ou então sofrerem uma transformação no motor de adaptação (não normativo) gerando como saída comandos mais simples para dispositivos compatíveis com a parte 5 do MPEG-V. A parte 3 da norma interage com as partes 2 e 5 da ISO/IEC 23005. A arquitetura e seu escopo (destacado em amarelo) são apresentados na Figura 1.3.

A parte 3 do MPEG-V especifica uma linguagem baseada em XML para descrição de efeitos sensoriais (SEDL - *Sensory Effect Description Language*), tais como luminosidade, aroma, vento, névoa, etc. (Choi e Kim, 2012). Os tipos de efeitos sensoriais não são parte da linguagem SEDL, mas são definidos em um vocabulário de efeitos sensoriais (SEV - *Sensory Effect Vocabulary*), fornecendo extensibilidade e flexibilidade para que desenvolvedores definam seus próprios efeitos sensoriais (Kim e Han, 2014). Uma descrição de metadados de efeitos sensoriais (SEM - *Sensory Effect Metadata*) pode estar associada a qualquer tipo de conteúdo multimídia, seja ele um jogo, um filme, uma música ou um website.

Waltl et al. (2009) conceituam um cenário que forma a cadeia da entrega das mídias até a renderização nos tocadores e atuadores. A mídia e o SEM podem ser obtidos a partir de uma fonte tradicional como DVD ou *Blu-Ray*, ou ainda, de serviços online. Em seguida o motor de adaptação faz a mediação para interpretar os recursos de mídia

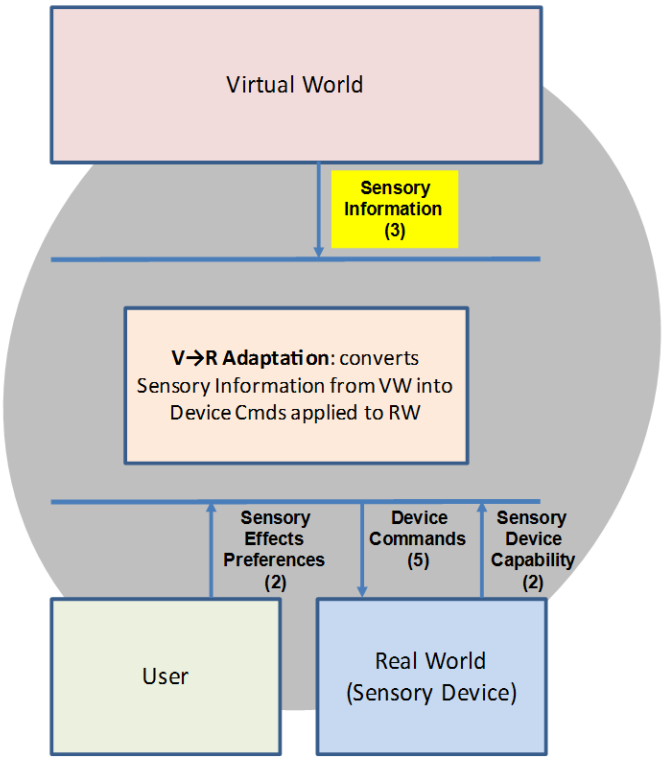

Figura 1.3. Escopo da parte 3 do MPEG-V destacada na caixa amarela e sua arquitetura. Fonte: (Choi e Kim, 2012)

acompanhados dos efeitos sensoriais de forma sincronizada, baseado nas preferências dos usuários e nas capacidades dos dispositivos. Assim, todo o conteúdo produzido pode ser entregue de acordo com o cenário específico de cada usuário para a renderização.

É importante ressaltar que não há necessariamente um mapeamento um-para-um (1:1) entre elementos ou tipos de dados de efeitos e os dispositivos sensoriais (Kim e Han, 2014). Um equipamento com capacidade de gerar temperaturas quente e fria poderia atender ao propósito de 2 tipos de efeitos sensoriais: calor e frio. Além desses 2 tipos, atualmente a norma define metadados (SEV) para efeitos de luz, luz colorida, *flash* de luz, vento, vibração, vaporização, perfume, névoa, correção de cor, movimento, cinestesia e tato (Choi e Kim, 2012).

Na descrição de um efeito sensorial é possível orquestrar a renderização dos efeitos em diversos dispositivos localizados em posições distintas no ambiente real do usuário. Nesta direção, um modelo de localização para os metadados de efeitos sensoriais (Figura 1.4) é definido no esquema de classificação “LocationCS” na ISO/IEC 23005-6 (Yoon e Han, 2012) e usado para apoiar a parte 3 do MPEG-V.

Os termos definidos no esquema “LocationCS” devem ser concatenados com o caractere “:” seguido da ordem do eixo x, y, e z para definir um local dentro do espaço tridimensional. *Wild cards* com o sinal “*” podem ser usados para se referir a um grupo de localização. Por exemplo, o valor “urn:mpeg:mpeg-v:01-SI-LocationCS-NS:*:*:back” indica que o efeito será renderizado considerando os atuadores que estejam em qualquer posição de x e y mas o eixo z restringe o efeito à parte de trás (*back*) do ambiente do

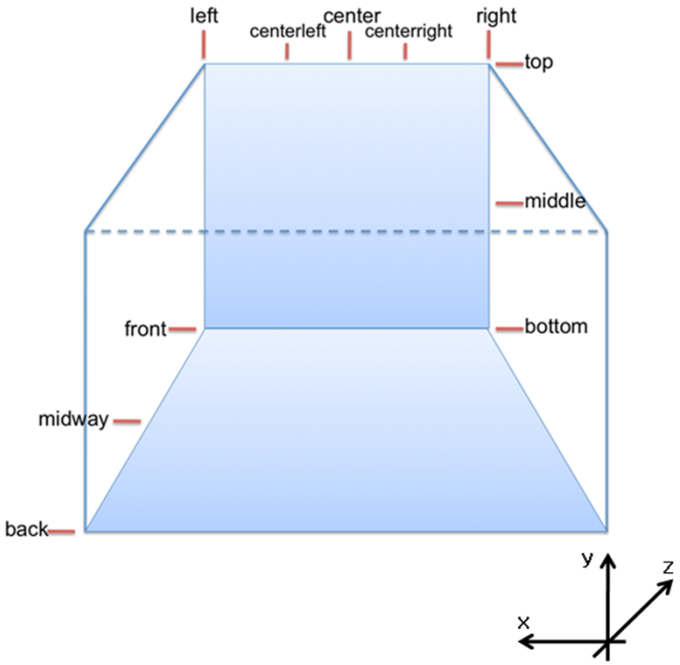

Figura 1.4. Modelo de localização para metadados de efeitos sensoriais e sistema de coordenadas de referência. Fonte: (Choi e Kim, 2012)

usuário.

A norma define também um modelo temporal para apoiar a descrição de metadados de efeitos sensoriais (Choi e Kim, 2012). Através do modelo é possível aumentar ou diminuir gradativamente a intensidade de um efeito sensorial, conforme mostra a Figura 1.5.

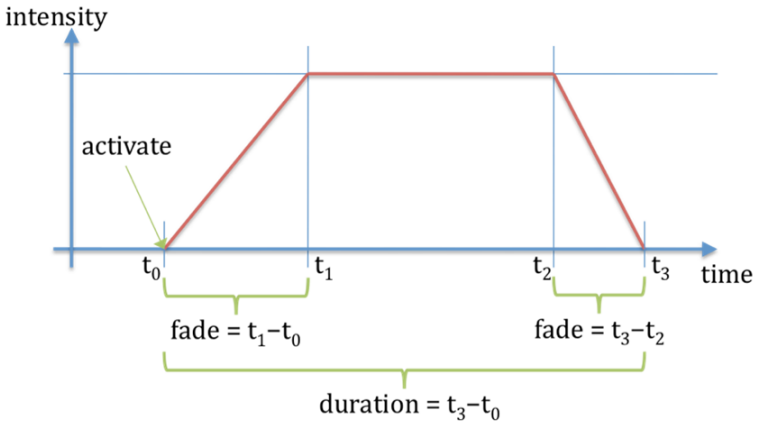

Figura 1.5. Modelo de tempo para metadados de efeitos sensoriais. Fonte: (Choi e Kim, 2012)

Para a compreensão do modelo, suponha um efeito de vento ativado em t_0 e desativado em t_3. Sua duração é t_3-t_0 e ele pode conter uma descrição para um aumento gradual, que é atribuído com duração de t_1-t_0. A duração t_1-t_0 é o tempo dentro do qual o efeito de vento deve atingir sua intensidade especificada. Da mesma maneira, uma diminuição gradual ocorre em t_2-t_3. Os trechos com descrições em XML apresentados no

final nessa seção ilustram um exemplo prático desse tipo de cenário.

Com o intuito de abstrair e facilitar o entendimento da SEDL, Waltl et al. (2009) apresentaram uma descrição formal da mesma por meio do Formalismo EBNF (*Extended Backus–Naur Form*). A seguir a descrição é apresentada e conceituada.

```
SEM ::= [DescriptionMetadata]
      (Declarations|GroupOfEffects|Effect|ReferenceEffect)+

Declarations ::= (GroupOfEffects|Effect|Parameter)+

GroupOfEffects ::= timestamp
              EffectDefinition
              EffectDefinition (EffectDefinition)*

Effect ::= timestamp
        EffectDefinition

EffectDefinition ::= [activate][duration][fade-in][fade-out]
                [alt][priority][intensity][position][adaptability]
```

O SEM é o elemento raiz e pode conter opcionalmente a descrição de metadados (*DescriptionMetadata*) que provê informações sobre o SEM. Em seguida, pelo menos uma das opções entre *Declarations, GroupOfEffects, Effect, e ReferenceEffect* deve ser fornecida. O elemento *Declarations* é usado para definir configurações comuns usadas pelos vários efeitos sensoriais e deve conter pelo menos uma das opções entre *GroupOfEffects, Effect e Parameter*. O elemento *GroupOfEffects* começa com uma marca de tempo (*timestamp*) que fornece quando o grupo de efeitos estará disponível para a aplicação e pode ser usada para fins de sincronização com a mídia associada. Também deve conter pelo menos duas definições de efeitos (*EffectDefinitions*) que correspondem a informações de efeitos sensoriais. O elemento *ReferenceEffect* permite referenciar um efeito criado anteriormente ou um grupo de efeitos. O elemento *Effect* é usado para descrever um efeito sensorial em uma determinada marca de tempo (*timestamp*). O elemento *EffectDefinition* possui atributos opcionais para configuração do efeito. *Activate* descreve se o efeito deve ser ativado; *duration* define a duração do efeito; *fade-in* e *fade-out* correspondem ao aumento/diminuição gradual do efeito; *alt* descreve um efeito alterativo caso o original não possa ser processado; *priority* diz respeito a prioridade entre os efeitos de um mesmo grupo; *intensity* define a intensidade do efeito de acordo com uma escala; *position* descreve a posição onde será renderizado o efeito em relação ao usuário, e por fim; *adaptability* que define tipos preferidos de adaptação do efeito correspondente.

Os exemplos seguintes reproduzidos a partir da norma (Choi e Kim, 2012) mostram o uso da SEDL em conjunto do vocabulário SEV, apresentando sequencialmente os tipos de efeitos para luz, vento e vibração.

```
<sedl:Effect xsi:type="sev:LightType" intensity-value="50.0"
   intensity-range="0.00001 32000.0" activate="true" color="#FF0000"
   si:pts="0"/>
...
```

```
<sedl:Effect xsi:type="sev:LightType" activate="false" color="#FF0000"
    si:pts="28"/>
```

O exemplo anterior da SEDL com o SEV mostra a descrição de um efeito de luz. A intensidade na ativação é de 50 *lux* (em um intervalo entre 0.00001 e 32000) com a cor "#FF0000". O efeito é ativado em si:pts="0" e desativa em si:pts="28".

```
<sedl:DescriptionMetadata>
 <sedl:ClassificationSchemeAlias alias="WCS"
     href="urn:mpeg:mpeg-v:01-SI-LocationCS-NS"/>
</sedl:DescriptionMetadata>
<sedl:Effect xsi:type="sev:WindType" fade="5"
    location=":WCS:*:*:front" intensity-value="3.0"
    intensity-range="0.0 12.0" activate="true" si:pts="0"/>
```

O exemplo anterior da SEDL com o SEV mostra a descrição de um efeito de vento. O efeito é ativado em si:pts="0", com a intensidade em 3.0 (em um intervalo entre 0 e 12 de acordo com a escala *Beaufort*) atingida após 5 segundos de aumento gradativo, para qualquer posição de x e y do atuador que esteja na frente do usuário.

```
<sedl:Effect xsi:type="sev:VibrationType" intensity-value="4.1"
    intensity-range="0.0 50.0" duration="7" fade="3" si:pts="0"/>
```

O exemplo anterior da SEDL com o SEV mostra a descrição de um efeito de vibração. O efeito é ativado em si:pts="0" com intensidade de 4.1 (em um intervalo entre 0 e 50 de acordo com a escala *Hertz*) com duração de 7 segundos e reduzido gradativamente a 3 segundos do fim.

O propósito básico da ISO/IEC 23005-3 é esquematizado na Figura 1.6 por um mapeamento das intenções do autor de conteúdo para metadados de efeitos sensoriais e capacidades de dispositivos (Choi e Kim, 2012). Em linhas gerais, um autor de conteúdo pretende estimular um sentido único ou múltiplos sentidos a partir da anotação de recursos multimídia com os metadados de efeitos sensoriais. Uma ferramenta de criação pode fornecer esse mapeamento e como saída gerar SEM compatível com a especificação da parte 3 do MPEG-V.

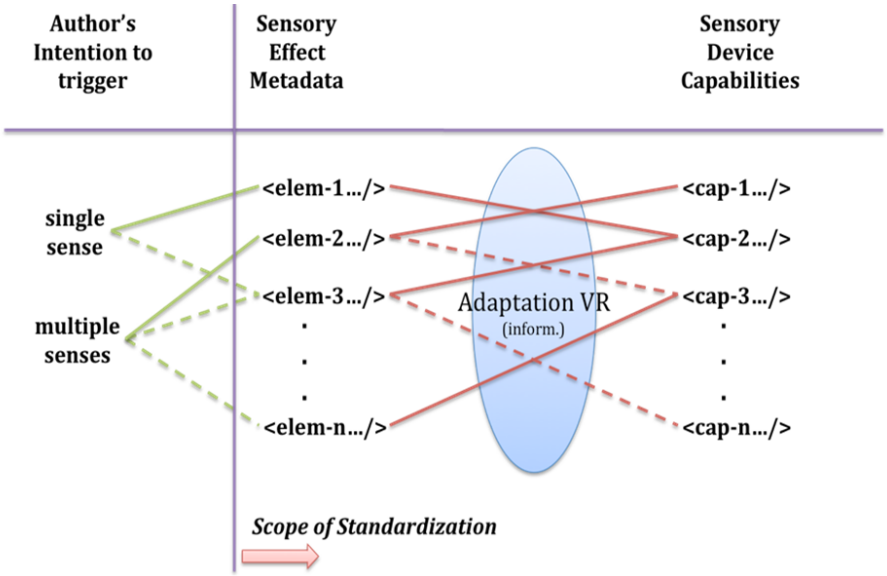

Figura 1.6. Mapeamento das intenções do autor dos metadados de efeitos sensoriais com a capacidade dos dispositivos. Fonte: (Choi e Kim, 2012)

1.2.4. MPEG-V Parte 4 – Características de Objetos do Mundo Virtual

A parte 4 do MPEG-V (ISO/IEC 23005-4) especifica sintaxe e semântica para descrever características de objetos do mundo virtual para torná-lo compatível com outro mundo virtual ou permitir que um objeto virtual seja controlado a partir de entradas (*inputs*) do mundo real (Han e Preda, 2012). Seu escopo pode ser observado na Figura 1.7.

Em um processo de *design* de ambientes virtuais, a criação de objetos pode ser uma tarefa demorada, pois envolve a caracterização individual de cada um associada a comportamentos, eventos e outras propriedades. A ideia principal do padrão é permitir a criação do objeto uma única vez e que ele possa ser utilizado em diferentes ambientes virtuais e que também possa ser compatível com propriedades do mundo real, por exemplo, informações de sensores fisiológicos dos usuários poderiam ser transportadas para o ambiente virtual com a utilização dos metadados descritos no padrão (Kim e Han, 2014).

O tipo de base das características de objetos do mundo virtual é composto pelos seguintes tipos de dados (Han e Preda, 2012):

- Identidade: contém um descritor de identificação como usuário associado ao objeto virtual, proprietário, direitos, etc.
- Som: contém recursos de som e propriedades relacionadas.
- Perfume: contém recursos de cheiro e as propriedades relacionadas.
- Controle: contém um conjunto de descritores para controlar movimento de um objeto virtual, orientação e escala.
- Evento: contém um conjunto de descritores que proporcionam eventos de entrada a partir de um mouse, teclado, etc.

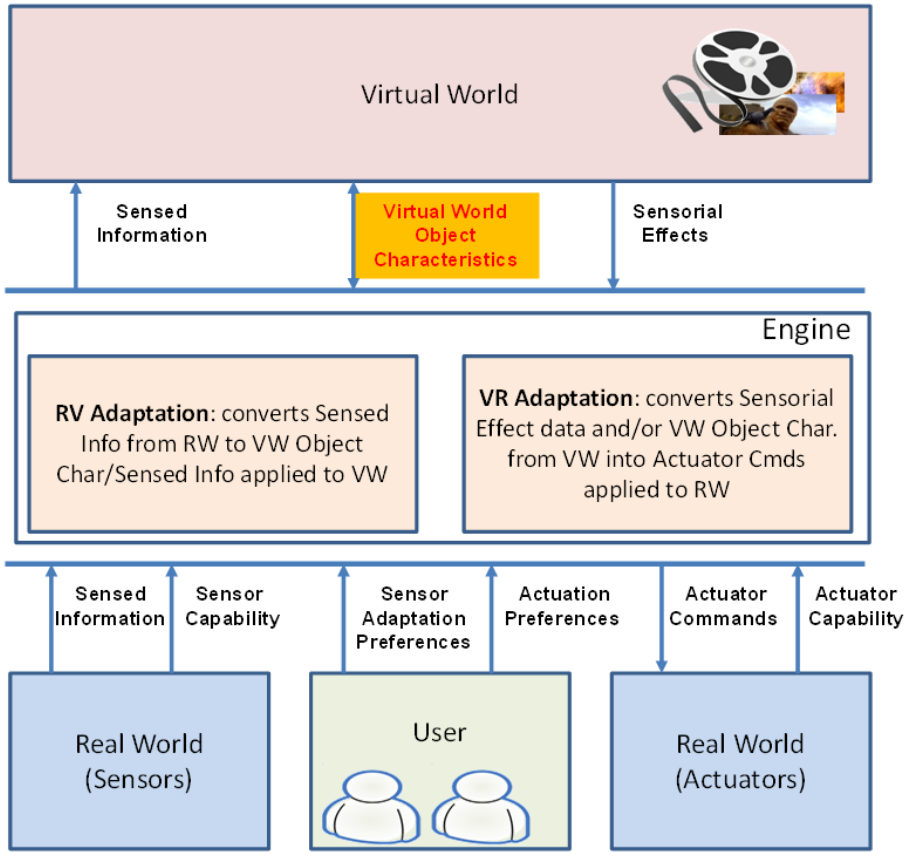

Figura 1.7. Escopo da parte 4 do MPEG-V destacada na caixa amarela. Fonte: (Kim e Han, 2014)

- Modelo de comportamento: contém um conjunto de descritores que definem as informações comportamento do objeto de acordo com os eventos de entrada.
- Id: contém um valor exclusivo para identificar cada objeto virtual individualmente.

Para descrever os metadados de avatares e de objetos virtuais, o tipo base é herdado e aspectos específicos são usados para cada metadado respectivo. Os metadados de um avatar no ambiente virtual podem ser utilizados para representar o usuário real no ambiente virtual e permitir a interação com o ambiente virtual. Os metadados são compostos de tipos de dados que descrevem recursos de aparência, recursos de animação, habilidades de comunicação, personalidade, recursos de controle, propriedades táteis e de gênero. Os metadados de um objeto virtual têm o propósito de caracterizar vários tipos de objetos no ambiente virtual e oferecer uma forma de interação com cada objeto. Os metadados descrevem os tipos de dados específicos e permitem descrever recursos de aparência, recursos de animação, propriedades táteis e componentes de objetos virtuais (Kim e Han, 2014).

Um trecho da norma ISO/IEC 23005-4 (Han e Preda, 2012) contendo um exemplo de descrição de um avatar e de objetos virtuais associados ao ambiente é apresentado a seguir:

```
<vwoc:VWOCInfo xsi:schemaLocation="urn:mpeg:mpeg-v:2012:01-VWOC-NS
    VWOCSchema.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:mpegvct="urn:mpeg:mpeg-v:2012:01-CT-NS"
    xmlns:vwoc="urn:mpeg:mpeg-v:2012:01-VWOC-NS"
    xmlns:r="urn:mpeg:mpeg21:2003:01-REL-R-NS"
    xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004">
   <vwoc:AvatarList>
     <vwoc:Avatar xsi:type="vwoc:AvatarType" id="AVATARID_1"
        gender="male">
       <vwoc:VWOC>
         <vwoc:SoundList>
           <vwoc:Sound loop="1" soundID="SOUNDID_10" duration="10"
              intensity="3" name="BurpSound">
           <vwoc:ResourcesURL>http://www.BurpSound.info</vwoc:ResourcesURL>
           </vwoc:Sound>
         </vwoc:SoundList>
         <vwoc:ScentList>
           <vwoc:Scent loop="2" duration="1" intensity="3"
              name="BurpingScent" scentID="SCENTID_11">
             <vwoc:ResourcesURL>http://www.Burp.info</vwoc:ResourcesURL>
           </vwoc:Scent>
         </vwoc:ScentList>
         <vwoc:ControlList>
           <vwoc:Control controlID="CTRLID_12">
             <vwoc:MotionFeatureControl>
               <vwoc:Position>
                 <mpegvct:X>1</mpegvct:X>
                 <mpegvct:Y>1</mpegvct:Y>
                 <mpegvct:Z>10</mpegvct:Z>
               </vwoc:Position>
               <vwoc:Orientation>
                 <mpegvct:X>0</mpegvct:X>
                 <mpegvct:Y>0</mpegvct:Y>
                 <mpegvct:Z>0</mpegvct:Z>
               </vwoc:Orientation>
               <vwoc:ScaleFactor>
                 <mpegvct:X>1</mpegvct:X>
                 <mpegvct:Y>1</mpegvct:Y>
                 <mpegvct:Z>3</mpegvct:Z>
               </vwoc:ScaleFactor>
             </vwoc:MotionFeatureControl>
           </vwoc:Control>
         </vwoc:ControlList>
         <vwoc:EventList>
           <vwoc:Event eventID="ID_13">
             <vwoc:Mouse>urn:mpeg:mpeg-v:01-VWOC-MouseEventCS-NS:click
                </vwoc:Mouse>
           </vwoc:Event>
         </vwoc:EventList>
       </vwoc:VWOC>
       <vwoc:BehaviorModelList>
         <vwoc:BehaviorModel>
           <vwoc:BehaviorInput eventIDRef="ID_13"/>
```

```
            <vwoc:BehaviorOutput controlIDRefs="CTRLID_12"
               scentIDRefs="SCENTID_11" soundIDRefs="SOUNDID_10"/>
          </vwoc:BehaviorModel>
        </vwoc:BehaviorModelList>
      </vwoc:Avatar>
    </vwoc:AvatarList>
</vwoc:VWOCInfo>
```

Explicando o exemplo, há um avatar com id="AVATARID_1", do gênero masculino, que possui um recurso de som, cheiro e controle de movimento associado. A referência de associação pode ser para um recurso interno ou externo. Há um controle de movimento identificado como "CTRLID_12" e um evento para *click* do mouse com id "ID_13". Uma lista associa comportamentos de entrada com as saídas desejadas. Quando ocorrer o evento *click* do mouse, o avatar executará as animações descritas nas propriedades "CTRLID_12" (controle de movimento), "SCENTID_11" (cheiro) e "SOUNDID_10" (som).

O MPEG-V parte 4 foi concebido para representar as características de objetos do mundo virtual e não inclui informação de geometria, som, animação, textura. Entretanto, o MPEG-V parte 4 pode ser combinado com o MPEG-4 Parte 11 (elementos gráficos) e Parte 16 (definir e animar avatares) em um ambiente interativo (Han e Preda, 2012).

O anexo da norma traz ainda descrições de esquemas de classificação que são utilizados para compor características de objetos virtuais, como por exemplo, esquemas de classificação para eventos de mouse, animações, etc.

1.2.5. MPEG-V Parte 5 – Formato de Dados para Dispositivos de Interação

A parte 5 do MPEG-V (ISO/IEC 23005-5) tem como objetivo fornecer formato de dados para dispositivos de interação que podem ser sensores ou atuadores. Não apenas o padrão MPEG-V pode ser beneficiado dessa norma, outros padrões internacionais como ISO/IEC 23007-2 (MPEG-U) ou ISO/IEC 14496-20 (especificações de representações de cenas) se referem ao uso do formato definido no padrão (Kim et al., 2012).

Na norma, é especificada sintaxe e semântica para formato de dados para dispositivos de interação, que são descrições de comandos para atuadores (V2R) e descrições de informações obtidas a partir de sensores no mundo real (R2V). Seu escopo pode ser observado na Figura 1.8. Uma linguagem para descrição de interface para interação (IIDL - *Interaction Interface Description Language*) baseada em XML permite definir interfaces para a troca de mensagens entre o mundo real e virtual e vice-versa (Kim et al., 2012).

Como parte da IIDL, dois vocabulários apoiam a descrição das mensagens para enviar os comandos (DCV - *Device Command Vocabulary*) ou obter informações a partir dos sensores (SIV - *Sensed Information Vocabulary*). O vocabulário DCV define descrições para comandar dispositivos atuadores de maneira mais simples que o vocabulário de efeitos sensoriais (SEV), por ser curta (Kim e Han, 2014). No entanto, para reproduzir um efeito sensorial de incremento de vento, por exemplo, mais de um comando com intensidades diferentes precisam ser descritos sequencialmente. Pode ser mais razoável para um fabricante implementar um atuador compatível com MPEG-V parte 5 pela simplicidade na execução e num cenário de efeitos sensoriais descritos com MPEG-V parte

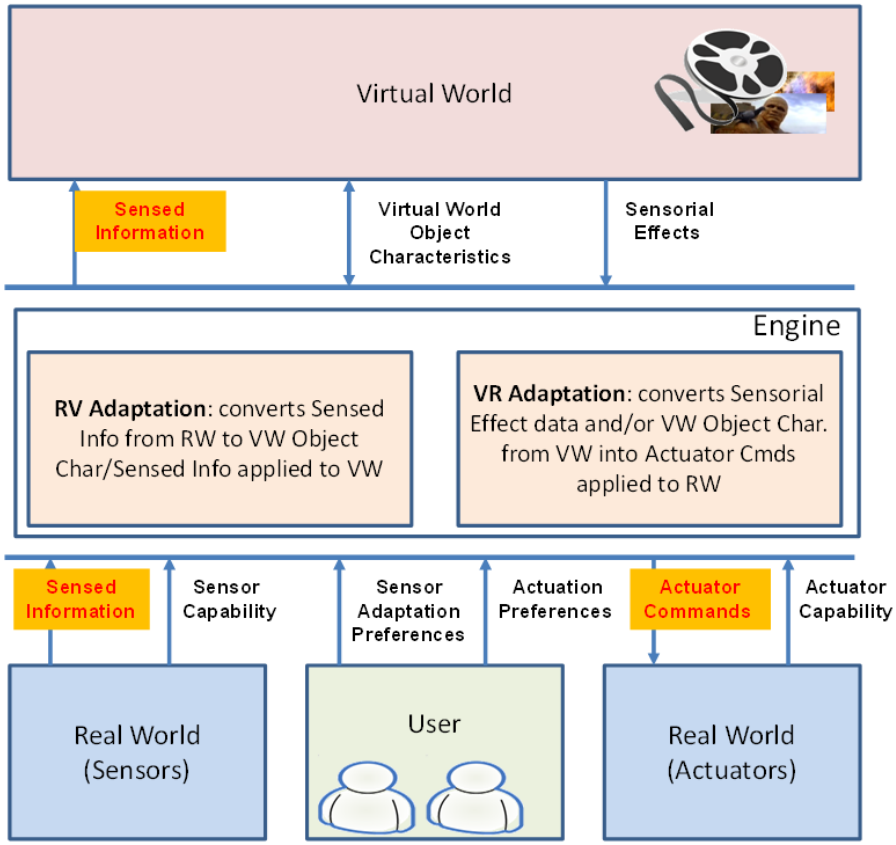

Figura 1.8. Escopo da parte 5 do MPEG-V destacada em caixas amarelas. Fonte: (Kim e Han, 2014)

3, pode ser necessário um motor de adaptação, conforme apresentado na Figura 1.1. O vocabulário SIV define descrições para obter informações a partir de sensores do mundo real, tais como luz, temperatura, acelerômetros, bio sensores, etc. (Kim e Han, 2014).

Os exemplos seguintes reproduzidos a partir da norma (Kim et al., 2012) mostram o uso da IIDL em conjunto dos vocabulários DCV e SIV:

```
<iidl:InteractionInfo>
  <iidl:DeviceCommandList>
    <iidl:DeviceCommand xsi:type="dcv:WindType" id="wind01"
      deviceIdRef="wind001" activate="true" intensity="30">
      <iidl:TimeStamp xsi:type="mpegvct:AbsoluteTimeType"
         absTime="1:30:23"/>
    </iidl:DeviceCommand>
  </iidl:DeviceCommandList>
</iidl:InteractionInfo>
```

O exemplo anterior da IIDL com o DCV mostra um comando de vento para um dispositivo que dispara vento. O comando identificado por “wind01” para o dispositivo identificado com o id “wind001”, com intensidade de 30% da intensidade máxima no tempo absoluto decorrido “1:30:23”.

```
<iidl:InteractionInfo>
  <iidl:SensedInfoList>
    <iidl:SensedInfo xsi:type="siv:TemperatureSensorType" id="TS001"
      sensorIdRef="TSID001" activate="true" value="36.5">
      <iidl:TimeStamp xsi:type="mpegvct:ClockTickTimeType"
         timeScale="100"
        pts="60000"/>
    </iidl:SensedInfo>
  </iidl:SensedInfoList>
</iidl:InteractionInfo>
```

O exemplo anterior da IIDL com o SIV mostra a obtenção da informação de um sensor de temperatura. A informação que precisa ser obtida tem o identificar "TS001" com referência para o dispositivo com id "TSID001". O Sensor será ativado com o valor 36.5C°. Deve ser mantido até o tempo "60000" e o valor colhido a cada "100" pulsos por segundo.

1.2.6. MPEG-V Parte 6 – Ferramentas e Tipos Comuns

A parte 6 do MPEG-V (ISO/IEC 23005-6) especifica sintaxe e semântica para ferramentas e tipos de dados comuns para uso nas outras partes do padrão, como por exemplo tipos relacionados a cores, tempo, unidades, etc.

Contém um anexo que fornece esquemas de classificação para os tipos de dados que são identificadas por uma sequência única de caracteres formada pela URN "urn:mpeg: mpeg-v: 01-CI-NameCS-NS", em que o nome (*Name*) deve ser substituído com o nome do esquema de classificação (Yoon e Han, 2012). Por exemplo, a URN "urn:mpeg:mpeg-v:01-CI-UnitTypeCS-NS" identifica o esquema de classificação que prevê valores para *UnitTypeCS*, que podem ser centímetros, metros, quilômetros, milhas, polegadas, etc.

Além do esquema de classificação *UnitTypeCS*, outros esquemas estão incluídos no anexo da norma: *ColorCS, LocationCS, ScentCS, ShakeDirectionCS, SpinDirectionCS, SprayingTypeCS, TactileEffectCS, WaveDirectionCS, WaveStartDirectionCS, TactileDisplayCS*, e esquemas de classificação de características do esqueleto humano.

A seguir, uma reprodução do trecho da norma (Yoon e Han, 2012) contendo 3 termos para o esquema de classificação de cores (*ColorCS*):

```
<ClassificationScheme uri="urn:mpeg:mpeg-v:01-SI-ColorCS-NS">
  <Term termID="alice_blue">
    <Name xml:lang="en">Alice blue</Name>
    <Definition>Describes the color Alice blue. Hex: #F0F8FF; RGB: 240,
       248, 255; HSV: 208 degrees, 6%, 100%.</Definition>
  </Term>
  <Term termID="alizarin">
    <Name xml:lang="en">Alizarin</Name>
    <Definition>Describes the color Alizarin. Hex: #E32636; RGB: 227,
       38, 54; HSV: 355 degrees, 83%, 89%.</Definition>
  </Term>
  <Term termID="amaranth">
```

```
   <Name xml:lang="en">Amaranth</Name>
   <Definition>Describes the color Amaranth. Hex: #E52B50; RGB: 229,
      43, 80; HSV: 345 degrees, 78%, 64%.</Definition>
 </Term>
```

O identificador do termo é acrescentado à URI do esquema de classificação para formar uma URN que representa as cores RGB descritas na definição do termo. Desse modo, a URN que representa a cor "Alice blue", por exemplo, é "urn:mpeg:mpeg-v:01-SI-ColorCS-NS:alice_blue".

1.2.7. MPEG-V Parte 7 – Conformidade e Software de Referência

A parte 7 do MPEG-V (ISO/IEC 23005-7) especifica regras de conformidade e software de referência com a finalidade de validar e esclarecer as especificações descritas com as várias partes do MPEG-V. Os módulos de software de referência disponíveis são especificados na forma de interfaces de programação de aplicativos (API), de acordo com a norma ISO/IEC 23006 (Kim et al., 2013).

Um esquema de validação baseado no Schematron (Kim et al., 2013), definido na parte 3 da ISO/IEC 19747-3 (*Document Schema Definition Languages – Part 3: Rule-based validation* – Schematron), é fornecido para testes de conformidade. As regras são executadas em conjunto da ferramenta Saxon para validar as descrições.

Um trecho de validação reproduzida a partir da norma (Kim et al., 2013) é descrita a seguir:

```
<pattern name="SEM element">
   <!-- R1.0: Check the conformance of SEM -->
   <rule context="sedl:SEM">
     <assert test="@si:timeScale">
       The SEM element shall have a timeScale attribute.
     </assert>
   </rule>
</pattern>
```

Se um elemento "SEM" de um arquivo de metadados de efeitos sensoriais não contém o atributo "si:timeScale", então, a descrição não está em conformidade.

1.3. A biblioteca MPEG Metadata

Os metadados de aplicações com sensores ou atuadores baseados no padrão MPEG-V precisam ser processados. Para facilitar a ligação entre o XML (descrito com o padrão MPEG-V) e objetos da aplicação, foi criada a biblioteca MPEG *Metadata*[4]. Ela compreende todos os objetos possíveis do padrão MPEG-V (parte 1 até 5) e parte dos padrões MPEG-7 e MPEG-21, possibilitando o empacotamento (*marshalling*) e desempacotamento (*unmarshalling*) automático do XML na aplicação por meio da API JAXB (*Java Architecture for XML Binding*).

[4]Código-fonte da biblioteca MPEG Metadata disponível em https://github.com/estevaosaleme/MpegMetadata

Para definir a estrutura e regras que um XML deve seguir e validá-lo, um arquivo XSD (XML *Schema Definition*) pode ser criado. Por exemplo, é possível definir quais atributos a *tag* <*effect*> da parte 3 do MPEG-V pode conter. Por meio da ferramenta Eclipse, é possível gerar classes Java a partir de esquemas XSD e vice-versa, usando também a API JAXB. Dessa forma, os esquemas XSD disponibilizados pelo grupo MPEG foram transformados em objetos Java dando origem a essa biblioteca.

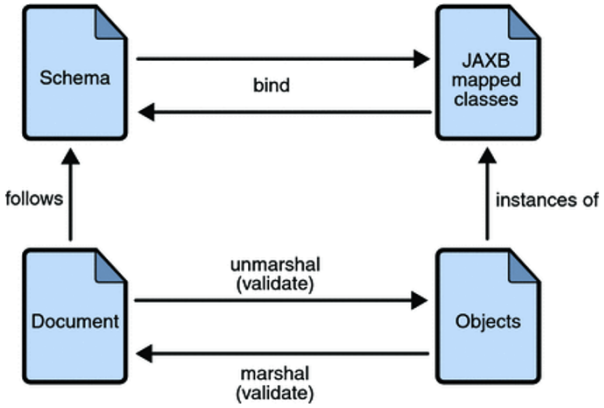

Figura 1.9. Processo de ligação JAXB. Fonte: (Oracle, 2015)

A Figura 1.9 apresenta o processo de ligação utilizando JAXB. Resumidamente, as etapas do processo de ligação com o JAXB são:

- Gerar classes: um esquema XSD é usado como entrada para gerar as classes;
- *Unmarshal*: documentos XML escritos de acordo com as restrições no esquema de origem são desempacotados pelo JAXB, dando origem a uma árvore de conteúdo;
- Validar (opcional): Se a árvore for modificada, o JAXB pode validar o conteúdo do XML de acordo com a nova árvore;
- Processar dados: os dados do XML podem ser lidos ou modificados com as classes geradas para ligação com o JAXB;
- *Marshal*: documentos XML são criados por meio do empacotamento a partir das classes mapeadas com o JAXB.

Os exemplos[5] a seguir demonstram como efetuar o *marshalling* a partir de um objeto java e *unmarshalling* a partir de um XML válido utilizando a biblioteca MPEG *Metadata*.

Exemplo 1, *marshalling* (objeto para XML):

```
public class Marshal {
  public static void main(String[] args) {
```

[5]Código-fonte contendo exemplos de uso da biblioteca MPEG *Metadata* disponível em https://github.com/estevaosaleme/MpegMetadataExample

```
        // Cria o objeto AccelerationSensorType (da biblioteca MPEG
           Metadata)
        AccelerationSensorType acel = new AccelerationSensorType();
        // Preenche as informações
        acel.setSensorIdRef("DEV_ID_001");
        acel.setActivate(true);
        Float3DVectorType data = new Float3DVectorType();
        data.setX(0.09f);
        data.setY(-0.45f);
        data.setZ(9.85f);
        acel.setAcceleration(data);
        // Cria lista de informações coletadas
        SensedInfoListType sil = new SensedInfoListType();
        // Adiciona informação do sensor
        sil.getSensedInfo().add(acel);
        // Cria um elemento para receber lista de informações coletadas
           ou comandos
        InteractionInfoType ii = new InteractionInfoType();
        ii.setSensedInfoList(sil);
      try {
        // Cria contexto JAXB e inicializa Marshaller
        JAXBContext jaxbContext =
           JAXBContext.newInstance(InteractionInfoType.class);
        Marshaller jaxbMarshaller = jaxbContext.createMarshaller();
        // Formata saída do XML
        jaxbMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
           true);
        // Cria elemento raiz para gerar saída XML
        JAXBElement<InteractionInfoType> rootElement = new
           JAXBElement<InteractionInfoType>(
               new QName("InteractionInfo"), InteractionInfoType.class,
                  ii);
        // Escreve XML de saída no console
        jaxbMarshaller.marshal(rootElement, System.out);
      } catch (JAXBException e) {
        e.printStackTrace();
      }
    }
}
```

No exemplo 1, o objeto “acel” da classe “AccelerationSensorType” foi instanciado e seus atributos foram preenchidos. Uma lista de informações coletadas “sil”, da classe “SensedInfoListType” foi criada e os dados de “acel” incluídos nela. Note que outras informações coletadas poderiam ser adicionadas nessa lista, como por exemplo, dados de sensores de força, pressão, movimento, etc. Em seguida, o contexto JAXB é criado e o “Marshaller” é inicializado. Para melhorar a formatação da saída a propriedade “JAXB_FORMATTED_OUTPUT” é configurada. Um elemento raiz (*root*) baseado na classe “InteractionInfoType” é criado com o nome “InteractionInfo” para manter a compatibilidade com a definição do padrão. Por fim, o objeto “rootElement” é transformado em XML através do método “marshal” e exibido no console da aplicação.

O exemplo demonstrado pode ser adaptado para criar arquivos XML MPEG-V para efetuar comandos em dispositivos, bastando para isso criar o objeto de comando

desejado e adicionar numa lista da classe “DeviceCommandList”.

Exemplo 2, *unmarshalling* (XML para objeto):

```
public class UnMarshal {
  public static void main(String[] args) {
    // Declaração de um XML seguindo o padrão MPEG-V, contendo dados
       de um acelerômetro
    String xml = "<?xml version=\"1.0\" encoding=\"UTF-8\"
       standalone=\"yes\"?>"+
      "<InteractionInfo xmlns:ns2=\"urn:mpeg:mpeg-v:2010:01-CT-NS\"
         xmlns:ns3=\"urn:mpeg:mpeg-v:2010:01-DCV-NS\"
         xmlns:ns4=\"urn:mpeg:mpeg7:schema:2004\"
         xmlns:ns5=\"urn:mpeg:mpeg-v:2010:01-SIV-NS\"
         xmlns:ns6=\"urn:mpeg:mpeg-v:2010:01-IIDL-NS\">"+
      "   <ns6:SensedInfoList>"+
      "      <ns6:SensedInfo
         xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
         xsi:type=\"ns5:AccelerationSensorType\"
         sensorIdRef=\"DEV_ID_001\" activate=\"true\">"+
      "         <ns5:Acceleration>"+
      "            <ns2:X>0.09</ns2:X>"+
      "            <ns2:Y>-0.45</ns2:Y>"+
      "            <ns2:Z>9.85</ns2:Z>"+
      "         </ns5:Acceleration>"+
      "      </ns6:SensedInfo>"+
      "   </ns6:SensedInfoList>"+
      "</InteractionInfo>";
    // Cria instância da classe StringReader para ser utilizada no
       processo de unmarshall do JAXB
    StringReader reader = new StringReader(xml);
    try {
       // Cria contexto JAXB e inicializa Marshaller
      JAXBContext jaxbContext =
         JAXBContext.newInstance(InteractionInfoType.class);
      Unmarshaller jaxbUnmarshaller =
         jaxbContext.createUnmarshaller();
      // Cria objeto para receber dados a partir do XML
      JAXBElement<InteractionInfoType> jii =
         jaxbUnmarshaller.unmarshal(new StreamSource(reader),
         InteractionInfoType.class);
      // Obtém os valores a partir do ponto inicial da hierarquia
         (InteractionInfoType)
      InteractionInfoType ii = jii.getValue();
      // Verifica se a lista de informações coletadas não está vazia
      if (ii.getSensedInfoList() != null &&
         ii.getSensedInfoList().getSensedInfo() != null){
        // Obtém os dados do acelerômetro
        AccelerationSensorType si = (AccelerationSensorType)ii.
           getSensedInfoList().getSensedInfo().get(0);
        // Exibe as informações no console
        System.out.println("ID: " + si.getSensorIdRef());
        System.out.println("ACTIVATE: " + si.isActivate());
        System.out.println("X: " + si.getAcceleration().getX());
        System.out.println("Y: " + si.getAcceleration().getY());
        System.out.println("Z: " + si.getAcceleration().getZ());
```

```
        }
      } catch (JAXBException e) {
        e.printStackTrace();
      }
    }
}
```

No exemplo 2, foi criada uma *string* contendo a declaração do XML gerado a partir da saída do exemplo 1 para realizar o processo de *unmarshall*. Uma instância da classe *StringReader* para ser utilizada no processo de *unmarshalling* do JAXB é criada no objeto "reader". Assim como no exemplo 1, o contexto JAXB é criado e o "Marshaller" é inicializado. Em seguida, o objeto "jii" da classe "JAXBElement<InteractionInfoType>" recebe os dados do XML através do método "unmarshal", que retorna um elemento JAXB. Então, os valores do XML são lidos a partir do ponto inicial da hierarquia, a classe "InteractionInfoType", instanciada no objeto "ii". Por fim, é verificado se o objeto "ii" contém informações e em caso positivo, eles são exibidos no console.

Note que os exemplos foram construídos a partir do conhecimento prévio do tipo de informação que seria processada (sensor acelerômetro). Em aplicações onde o tipo de informação não é conhecido, é necessário efetuar uma verificação da instância do objeto para trabalhar com um determinado tipo de informação específico.

1.4. Desenvolvendo aplicações interativas com MPEG-V

A partir dos conceitos apresentados nas seções anteriores é possível explorar com maior nível de detalhe o desenvolvimento de aplicações interativas interoperáveis através do padrão MPEG-V. Nessa seção são apresentadas duas soluções utilizando o padrão. Uma integra dados de um sensor acelerômetro remoto com uma aplicação *desktop* por meio do MPEG-V (R2V) e a outra integra metadados de aplicações multimídia contendo efeitos sensoriais com atuadores tais como luz, vento e vibração (V2R).

1.4.1. Ferramentas necessárias

Para o desenvolvimento das aplicações conforme proposto são requeridas as seguintes ferramentas: (*i*) Computador com sistema operacional Linux ou Windows, (*ii*) Java 7 ou superior, (*iii*) Git, (*iv*) Eclipse IDE for Java Developers, (*v*) *Plugin* Android Development Tools (ADT) para Eclipse IDE e (*vi*) *Smartphone* com sistema operacional Android.

1.4.2. Aplicação com sensor compatível com MPEG-V

Grande parte dos *smartphones* que rodam o Android possuem sensores de movimento com a capacidade de fornecer dados brutos de alta precisão (DeveloperAndroid, 2015), que medem a força de aceleração e rotação em torno de 3 eixos (x, y e z) e são úteis para detectar rotação e agitação. Nessa solução, denominada SensedApplicationExample, o acelerômetro é usado para detectar movimentos de agitação. Trata-se de uma solução didática para perceber como a informação pode ser conduzida do mundo real para o mundo virtual através do padrão MPEG-V.

A solução consiste em integrar dados de um sensor acelerômetro remoto (AccelerometerCollector) com uma aplicação *desktop* por meio de uma rede local. Ela captura

dados de um acelerômetro de um *smartphone* Android e transmite para a aplicação SenaMove para gerar números aleatórios da Sena quando o *smartphone* é agitado.

A Figura 1.10 mostra as etapas da integração de dados do AccelerometerCollector com o SenaMove. Na etapa *(1)* a aplicação AccelerometerCollector é instalada em um *smartphone* rodando Android. Ao executá-la *(2)* são capturados eventos de movimento do acelerômetro, contendo a posição dos eixos *x*, *y* e *z*. Em seguida, esses dados são encapsulados dentro de um XML no formato MPEG-V *(3)*. Na interface é necessário indicar o endereço IPv4 para destino das informações. Após a conexão com a aplicação, os dados são transmitidos por meio do protocolo UDP para a aplicação SenaMove, que por sua vez detecta movimentos de agitação (baseados nos eixos e no tempo em que ocorreu) para gerar números aleatórios da Sena na sua interface *(4)*.

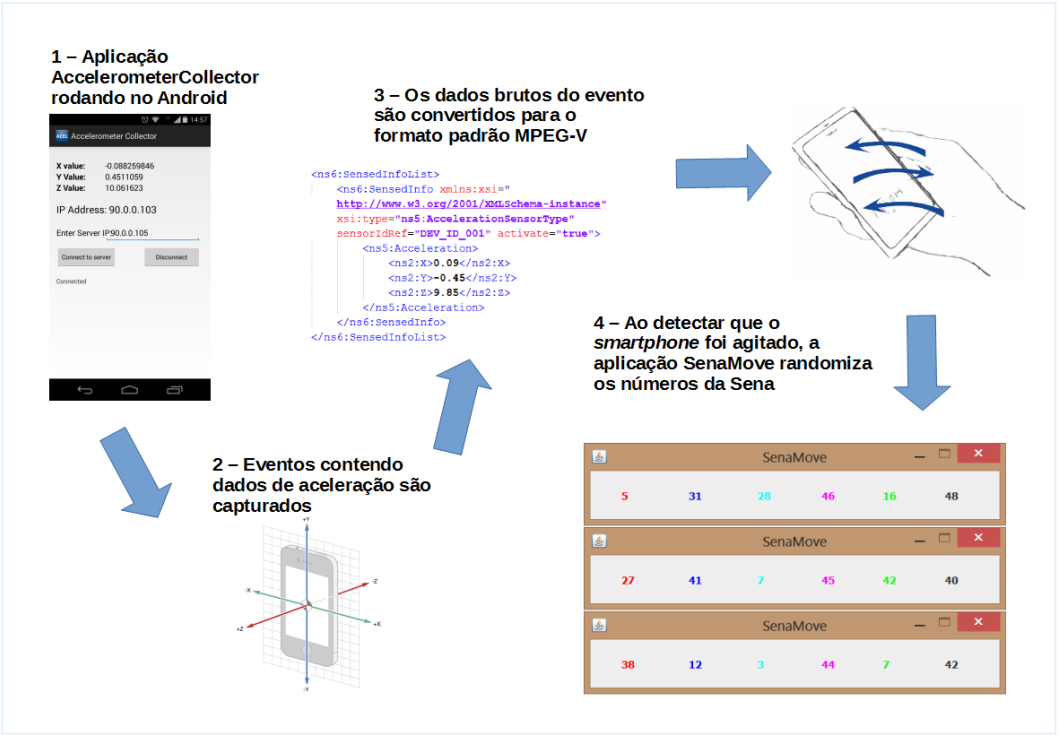

Figura 1.10. Etapas da integração de dados do acelerômetro Android com aplicação *desktop*.

O trecho de código do AccelerometerCollector para Android, coletando e transmitindo dados do acelerômetro no formato MPEG-V, é apresentado e interpretado a seguir:

```
public void onSensorChanged(SensorEvent event) {
  // Verifica se evento édo tipo acelerômetro
  if (event.sensor.getType()==Sensor.TYPE_ACCELEROMETER){
    // Obtém os valores dos eixos x, y e z
    ax=event.values[0];
    ay=event.values[1];
    az=event.values[2];
    // String no formato MPEG-V écriada com os dados dos eixos
    StringBuilder stbAcelerometerMetadata = new StringBuilder();
    stbAcelerometerMetadata.append("<?xml version=\"1.0\"
       encoding=\"UTF-8\" standalone=\"yes\"?>");
```

```
stbAcelerometerMetadata.append("<InteractionInfo
    xmlns:ns2=\"urn:mpeg:mpeg-v:2010:01-CT-NS\"
    xmlns:ns3=\"urn:mpeg:mpeg-v:2010:01-DCV-NS\"
    xmlns:ns4=\"urn:mpeg:mpeg7:schema:2004\"
    xmlns:ns5=\"urn:mpeg:mpeg-v:2010:01-SIV-NS\"
    xmlns:ns6=\"urn:mpeg:mpeg-v:2010:01-IIDL-NS\">");
stbAcelerometerMetadata.append(" <ns6:SensedInfoList>");
stbAcelerometerMetadata.append(" <ns6:SensedInfo
    xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
    xsi:type=\"ns5:AccelerationSensorType\"
    sensorIdRef=\"DEV_ID_001\" activate=\"true\">");
stbAcelerometerMetadata.append(" <ns5:Acceleration>");
stbAcelerometerMetadata.append("    <ns2:X>" + ax + "</ns2:X>");
stbAcelerometerMetadata.append("    <ns2:Y>" + ay + "</ns2:Y>");
stbAcelerometerMetadata.append("    <ns2:Z>" + az + "</ns2:Z>");
stbAcelerometerMetadata.append(" </ns5:Acceleration>");
stbAcelerometerMetadata.append(" </ns6:SensedInfo>");
stbAcelerometerMetadata.append(" </ns6:SensedInfoList>");
stbAcelerometerMetadata.append("</InteractionInfo>");
// Se a aplicação desktop está conectada os dados são
    transmitidos
if (connected){
  sendMessage(edtServerIp.getText().toString(),
     stbAcelerometerMetadata.toString());
  ...
```

O acelerômetro mede a força de aceleração em m/s^2 que é aplicada em um dispositivo nos três eixos físicos (x, y e z), incluindo a força da gravidade (DeveloperAndroid, 2015). Quando o dispositivo encontra-se deitado sobre uma mesa com a tela para cima as seguinte condições sobre sua orientação natural são verdadeiras:

- Valor de aceleração do eixo x é positivo quando a parte direita é levantada;
- Valor de aceleração do eixo y é positivo quando a parte de cima é levantada;
- Valor estacionário do eixo z é A + 9.81 m/s^2 (inclui a força gravitacional de magnitude 9.81 m/s^2); A aceleração é obtida removendo a força da gravidade (-9.81 m/s^2).

A medida que o *smartphone* é movimentado são gerados eventos no objeto "event" da classe "SensorEvent". O objeto inclui dados brutos dos sensores, o tipo de sensor que gerou o evento, a acurácia dos dados e a marca de tempo que o evento ocorreu. No código pode ser observado que há uma verificação se evento é do tipo acelerômetro. Caso positivo, os dados dos eixos x, y e z são inseridos dentro de uma *string* formatada com o MPEG-V. Não foi utilizada a biblioteca MPEG *Metadata* por falta de suporte nativo do Android ao JAXB. Por fim, caso a aplicação *desktop* esteja conectada, os dados são transmitidos através do método "sendMessage", que se comunica por meio do protocolo UDP com a aplicação SenaMove.

No lado do SenaMove as informações são recebidas e processadas. O trecho de código seguinte mostra o SenaMove recebendo e tratando dados enviados pela aplicação AccelerometerCollector:

```
...
// Cria um soquete de datagrama na porta 12345
serverSocket = new DatagramSocket(12345,ipHostAddress);
System.out.println("SERVER OPENED ON
    "+ipHostAddress.getHostAddress());
byte[] receiveData = new byte[1024];
while (true) {
    // Cria um pacote de datagrama
   DatagramPacket receivePacket = new DatagramPacket(receiveData,
       receiveData.length);
   // Recebe o pacote transmitido
   serverSocket.receive(receivePacket);
   // Transforma os dados do pacote em String
   String message = new String(receivePacket.getData()).trim();
   System.out.println(new Date().getTime() + " RECEIVED: " +
       message);
   // Se a mensagem não for ping, processa os dados do sensor
   if (!message.startsWith("ping")){
     message = message.substring(0,
        message.lastIndexOf("</InteractionInfo>") + 18);
     StringReader reader = new StringReader(message);
     try {
       JAXBElement<InteractionInfoType> jii =
           jaxbUnmarshaller.unmarshal(new StreamSource(reader),
           InteractionInfoType.class);
       InteractionInfoType ii = jii.getValue();
       if (ii.getSensedInfoList() != null &&
           ii.getSensedInfoList().getSensedInfo() != null){
         AccelerationSensorType si = (AccelerationSensorType)ii.
             getSensedInfoList().getSensedInfo().get(0);
         long curTime = System.currentTimeMillis();
         // Verifica se a diferença do tempo entre o evento atual e
             o último evento émaior que 100ms
         if ((curTime - senaMove.lastUpdate) > 100) {
           long diffTime = (curTime - senaMove.lastUpdate);
           senaMove.lastUpdate = curTime;
           float speed = Math.abs(si.getAcceleration().getX() +
               si.getAcceleration().getY() +
               si.getAcceleration().getZ() - senaMove.lastX -
               senaMove.lastY - senaMove.lastZ)/ diffTime * 10000;
           // Se a agitação édetectada os números aleatórios são
               obtidos
           if (speed > SHAKE_THRESHOLD) {
             senaMove.getRandomNumber();
             ...
```

O trecho do código-fonte do SenaMove mostra a instância de uma classe que provê a comunicação por meio do protocolo UDP. Ao receber os dados do acelerômetro no formato MPEG-V, eles são desempacotados no objeto “jii” e então, caso o objeto não seja nulo, o intervalo de tempo entre eventos é verificado para evitar processamento desnecessário. Finalmente, ao ser detectado uma velocidade maior que o limiar para agitação, os números aleatórios são gerados e exibidos na interface da aplicação SenaMove.

O código-fonte da solução está disponível no GitHub[6]. É possível também testar a solução sem a necessidade do ambiente de desenvolvimento, uma *release*[7] contendo os arquivos AccelerometerCollect.apk (para Android 4.4.2) e SenaMove.jar (para Java 7 ou superior) está disponível.

1.4.3. Aplicação com atuadores compatíveis com MPEG-V

De acordo com Ghinea et al. (2014), MulSeMedia (*Multiple Sensorial Media*) combina a mídia tradicional (por exemplo, texto, imagem e vídeo) com objetos que objetivam estimular outros sentidos humanos (ver seção 1.2.3). Esses sentidos podem ser ativados por atuadores capazes de produzir efeitos luminosos, vento, vibração, aroma, etc.

A solução com atuadores compatíveis com MPEG-V a ser explorada parte desse contexto. Do mundo virtual para o mundo real (V2R) ela integra metadados de aplicações multimídia contendo efeitos sensoriais com atuadores (luz, vento e vibração). A solução, desenvolvida por Saleme e Santos (2015), é a plataforma PlaySEM, que compreende um *Video Player* (SE) e um renderizador de efeitos sensoriais (SE *Renderer*) remoto composto de três atuadores (fita de LED, para efeitos luminosos; motores de vibração de celular, para efeito de vibração; *coolers* de PC, para efeito de vento).

De modo sintético, o funcionamento da plataforma PlaySEM é assim: o usuário visualizador de conteúdo fornece como entrada a mídia e o SEM descrito com o padrão MPEG-V para reprodução no SE *Video Player*, que por sua vez, se comunica com o software SE *Renderer* por meio da interface UPnP para o processamento do SEM. O SE *Renderer* processa o SEM convertendo-os em mensagens (*bytes*) que serão encaminhadas para um microcontrolador Arduino[8] para acionar os dispositivos físicos (Figura 1.11) que renderizarão os efeitos de luz, vento e vibração. Na ausência do Arduino, o SE *Renderer* pode exibir as mensagens de saída no próprio console da aplicação, sendo necessário apenas alterar o atributo "emulateDevices" para "true" no arquivo de configuração da aplicação (config.properties).

Na execução do SE *Video Player* primeiramente um dispositivo renderizador de efeitos sensoriais deve ser selecionado na interface. Em seguida, um vídeo contendo SEM deve ser fornecido. Se o vídeo possuir um arquivo SEM com extensão ".sem" ou ".xml", então o SE *Video Player* carrega o arquivo para a memória. O SE *Video Player* faz assinatura de notificação de eventos (publish/subscribe) UPnP do SE *Renderer* que indicará a mudança de estado do processamento do SEM e transmite-o para o renderizador. O trecho de código seguinte mostra o SE *Video Player* assinando a notificação através do método "activeReceiveEvents" e transmitindo o SEM através do método "setSem" para o SE *Renderer:*

```
  private void prepareSEDevice(){
    try {
      // Assina notificação de eventos do SE Renderer
```

[6]Código-fonte da solução SensedApplicationExample disponível em https://github.com/estevaosaleme/SensedApplicationExample

[7]*Release* da solução SensedApplicationExample disponível em https://github.com/estevaosaleme/SensedApplicationExample/releases

[8]O Arduino é um pequeno computador que você pode programar para processar entradas e saídas entre o dispositivo e os componentes externos conectados a ele (McRoberts, 2011).

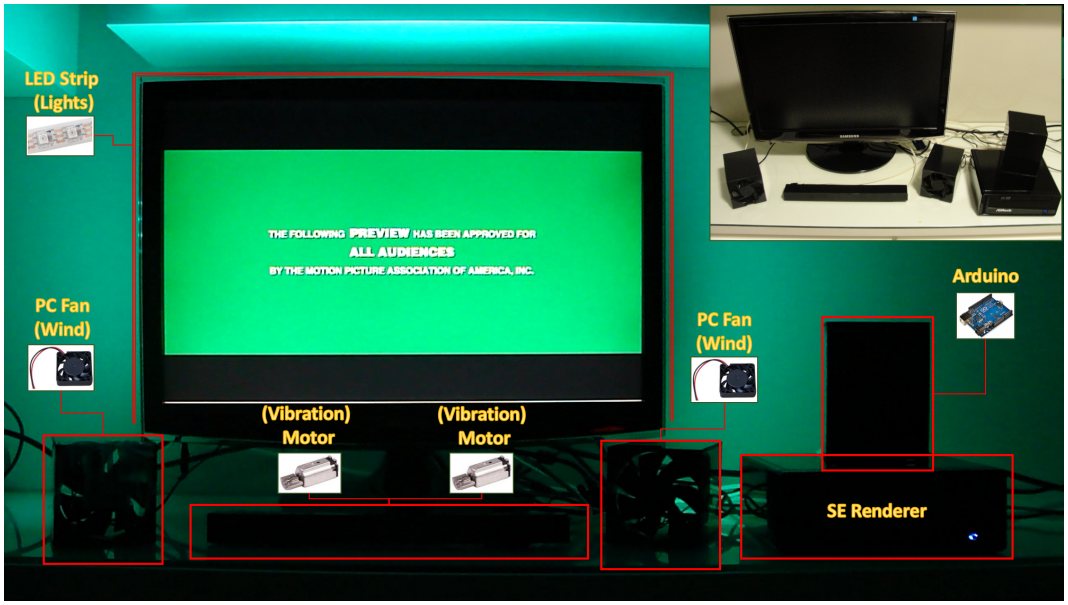

Figura 1.11. Ambiente de execução do SE *Renderer* com o conjunto de atuadores de luz, vento e vibração. Fonte: (Saleme e Santos, 2015)

```
CommandSERendererDevice.activeReceiveEvents();
Thread.sleep(100);
// Cria um timer para aguardar loading por 30 segundos
timerSemWaitMessage = new Timer(30000, new ActionListener(){
     public void actionPerformed(ActionEvent e) {
        finishFailLoadSEM();
     }
  });
timerSemWaitMessage.start();
// Envia metadados de efeitos sensoriais para o SE Renderer
CommandSERendererDevice.setSem(VideoPlayer.sem,
    embeddedMediaPlayerComponent.getMediaPlayer().
 ...
```

Após enviar o SEM para o SE *Renderer* uma mensagem de carregamento é exibida na interface do SE *Video Player* e ela é removida em caso de sucesso ou falha. No meio do processo o SE *Renderer* pode ser desligado e o SE *Video Player* ficar aguardando infinitamente por uma notificação de evento que nunca ocorrerá. Para contornar esse possível problema, um mecanismo para aguardar por no máximo 30 segundos o carregamento e processamento remoto do SEM no SE *Renderer* foi criado. Caso o tempo expire, o método "finishFailLoadSEM" é invocado para avisar a falha ao usuário da aplicação e remover a mensagem de carregamento.

O trecho de código seguinte mostra o monitoramento dos eventos recebidos no SE *Video Player*. Caso o valor da variável "semPrepared" do SE *Renderer* seja "1 ou *yes* ou *true* ou *on*", o método "finishSucessLoadSEM" é acionado, para remover a mensagem de carregamento e iniciar a reprodução da mídia:

```
public void eventReceived(GENASubscription sub) {
  @SuppressWarnings("unchecked")
```

```
Map<String, StateVariableValue> values = sub.getCurrentValues();
String semPrepared =
    values.get("SemPrepared").getValue().toString();
if ("1".equalsIgnoreCase(semPrepared) ||
    "yes".equalsIgnoreCase(semPrepared) ||
     "true".equalsIgnoreCase(semPrepared) ||
        "on".equalsIgnoreCase(semPrepared)){
  MediaPlayerActions.finishSucessLoadSEM();
  ...
```

Do lado do SE *Renderer*, o SEM precisa ser recebido e processado. O trecho de código seguinte mostra o SE *Renderer* recebendo metadados de efeitos sensoriais e disparando processamento:

```
@UpnpAction
public void setSem(
    @UpnpInputArgument(name = "SensoryEffectMetadata")
    String newSensoryEffectMetadata,
    @UpnpInputArgument(name = "Duration")
    String newDuration) {
  semPrepared = false;
  // Dispara evento para assinantes para informar que o SEM não foi
     processado
  getPropertyChangeSupport().firePropertyChange("SemPrepared", "",
     semPrepared);
  // Atribui internamente os dados recebidos pelo serviço UPnP
  sensoryEffectMetadata = newSensoryEffectMetadata;
  duration = newDuration;
  currentTime = "0";
  lightAutoExtraction = false;
  if (SERendererDevice.debugMode)
    System.out.println("SetSem");
  // Instancia e inicia thread que realizará o parser do MPEG-V em
     comandos para dispositivos físicos
  Thread semParser = new Thread(new
     SEMParser(sensoryEffectMetadata, Long.parseLong(duration)));
  semParser.start();
}
```

No processamento do SEM ocorre a transformação do XML para os objetos da aplicação por meio da biblioteca MPEG *Metadata*. Durante a iteração dos elementos, mensagens para produzir os efeitos de luz, vento e vibração são preparadas para serem encaminhadas para o Arduino. Quando não existirem mais elementos a serem processados, o SE *Renderer* notifica aos assinantes do serviço que a preparação dos efeitos sensoriais foi finalizada, permitindo assim a reprodução do vídeo em conjunto da execução das ações preparadas na linha de tempo de efeitos. O trecho de código a seguir mostra a notificação:

```
 @Override
public void run() {
  try {
    // Realiza o parser do XML para objetos
    parse(sensoryEffectMetadata);
```

```
...
SERendererService.semPrepared = true;
try {
   Thread.sleep(100);
   // Dispara evento para assinantes para informar que o SEM não
      foi processado
   SERendererService.getPropertyChangeSupport().
      firePropertyChange("SemPrepared", "",
      SERendererService.semPrepared);
...
```

De volta ao SE *Video Player*, antes de iniciar a reprodução, ele verifica se o SEM contém auto-extração de cores do vídeo. Caso positivo, ele extrai as cores em três partes (esquerda, centro e direita) e entrega no formato hexadecimal para o SE *Renderer*. Esse processo perdura até que todo o conteúdo do vídeo tenha sido reproduzido.

Durante a execução do SE *Renderer*, enquanto o estado do renderizador for igual a "PLAY" (reproduzindo) e a duração for maior ou igual ao tempo atual, o relógio é incrementado e é verificada na pilha de mensagens da linha de tempo se há alguma ação para ser executada naquele instante. Em caso positivo, o SE *Renderer* envia a mensagem para o Arduino para que o efeito seja renderizado no ambiente do usuário.

O código-fonte e um *wiki* contendo instruções sobre a instalação da solução sem a necessidade do ambiente de desenvolvimento está disponível no GitHub no repositório SE *Video Player*[9] e SE *Renderer*[10].

1.5. Conclusão

Com foco no desenvolvimento de aplicações interativas interoperáveis foram apresentados conceitos e exemplos de todas as 7 partes do padrão MPEG-V, além da biblioteca MPEG *Metadata* que facilita a ligação entre o XML (suportando todas as partes do MPEG-V) e objetos da aplicação. Os conceitos apresentados foram colocados em prática a partir de exemplos de aplicações compostas por sensores ou atuadores, envolvendo a comunicação remota com outros sistemas/dispositivos.

A primeira solução mostrou como integrar dados de um sensor acelerômetro remoto com uma aplicação *desktop* por meio do MPEG-V. A segunda integrou metadados de aplicações multimídia contendo efeitos sensoriais com atuadores (luz, vento e vibração). Os códigos-fonte das soluções foram exploradas com profundidade permitindo o entendimento de como os dados de aplicações interativas são intercambiadas entre o mundo real e o mundo virtual e vice-versa.

Apesar do padrão MPEG-V ser uma solução de interoperabilidade escalável através de seus vocabulários, dependendo dos requisitos da aplicação pode ser necessário o uso de outros padrões de interoperabilidade para aplicações interativas. Por exemplo, para voz, o VoiceXML ou SSML; gestos com semântica, o MPEG-U; para emoções, o Emo-

[9]Repositório do software PlaySEM SE *Video Player* disponível em https://github.com/estevaosaleme/PlaySEM_SEVideoPlayer

[10]Repositório do software PlaySEM SE *Renderer* disponível em https://github.com/estevaosaleme/PlaySEM_SERenderer

tionML. Para os padrões baseados em XML as mesmas técnicas apresentadas aqui para integrar os objetos reais e virtuais podem ser usadas, substituindo a biblioteca MPEG *Metadata* por uma nova biblioteca JAXB gerada a partir de esquemas XSD correspondente ao padrão que será usado.

Referências

Cho, H. Y. (2010). Event-Based control of 4D effects using MPEG RoSE. Master's thesis, School of Mechanical, Aerospace and S. Engineering. Korea Adv. Inst. of Science and Technology., South Korea. Disponível em: http://hdl.handle.net/10203/45825.

Choi, B. S. e Kim, S. K. (2012). *Text of ISO/IEC FDIS 23005-3 2nd edition Sensory Information*. Shanghai, China. MPEG Group Meeting, ISO/IEC JTC 1/SC 29/WG 11/N13059.

Choi, S. G. e Park, M. R. (2013). An architecture and method using MPEG-V metadata in smartphone and sensor aggregator. In *15th International Conference on Advanced Communication Technology (ICACT)*, pages 139–142.

DeveloperAndroid. Sensors Overview [online]. (2015). Disponível em: http://developer.android.com/guide/topics/sensors/sensors_overview.html. [Acessado em 12.9.2015].

Ghinea, G., Timmerer, C., Lin, W., e Gulliver, S. (2014). Mulsemedia: State of the Art, Perspectives, and Challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 11(1s):17:1–17:23. Disponível em: http://dx.doi.org/10.1145/2617994.

Han, J. J. e Preda, M. (2012). *Text of ISO/IEC FDIS 23005-4 2nd edition Virtual World Object Characteristics*. Shanghai, China. MPEG Group Meeting, ISO/IEC JTC 1/SC 29/WG 11/N13061.

Han, J. J. e Yoon, K. R. (2012). *Text of ISO/IEC FDIS 23005-2 2nd edition Control Information*. Shanghai, China. MPEG Group Meeting, ISO/IEC JTC 1/SC 29/WG 11/N13057.

Kim, S. K. (2013). Authoring multisensorial content. *Sig. Proc.: Image Comm.*, 28(2):162–167. Disponível em: http://dx.doi.org/10.1016/j.image.2012.10.011.

Kim, S. K. e Han, J. J. (2014). *Text of white paper on MPEG-V*. San Jose, USA. MPEG Group Meeting, ISO/IEC JTC 1/SC 29/WG 11 W14187.

Kim, S. K., Han, J. J., e Yoon, K. R. (2012). *Text of ISO/IEC FDIS 23005-5 2nd edition Data Formats for Interaction Device*. Shanghai, China. MPEG Group Meeting, ISO/IEC JTC 1/SC 29/WG 11/N13063.

Kim, S. K. e Joo, Y. S. (2014). Sensible Media Simulation in an Automobile Application and Human Responses to Sensory Effects. *ETRI Journal*, 35(6):1001–1010. Disponível em: http://dx.doi.org/10.4218/etrij.13.2013.0038.

Kim, S. K., Joo, Y. S., e Choi, B. S. (2013). *Text of ISO/IEC FDIS 23005-7 2nd edition Conformance and reference software*. Vienna, Austria. MPEG Group Meeting, ISO/IEC JTC 1/SC 29/WG 11/N13812.

McRoberts, M. (2011). *Arduino básico*. Novatec Editora, 1 edition.

Oracle. JAXB Architecture [online]. (2015). Disponível em: http://docs.oracle.com/javase/tutorial/jaxb/intro/arch.html. [Acessado em 24.2.2015].

Saleme, E. B. e Santos, C. A. S. (2015). PlaySEM: a platform for rendering MulSeMedia compatible with MPEG-V. In *Proceedings of the 21th Brazilian Symposium on Multimedia and the Web*, WebMedia '15, Manaus/AM, Brazil. ACM. Disponível em: http://dx.doi.org/10.1145/2820426.2820450.

Santos, C. A. S., Rehem Neto, A. N., e Saleme, E. B. (2015a). An Event Driven Approach for Integrating Multi-Sensory effects to Interactive Environments (to appear). In *IEEE International Conference on Systems, Man, and Cybernetics (SMC2015)*, Hong Kong.

Santos, C. A. S., Saleme, E. B., e Andrade, J. C. S. (2015b). A Systematic Review of Data Exchange Formats in Advanced Interaction Environments. *International Journal of Multimedia and Ubiquitous Engineering (IJMUE)*, 10(5). Disponível em: http://dx.doi.org/10.14257/ijmue.2015.10.5.13.

Waltl, M., Rainer, B., Timmerer, C., e Hellwagner, H. (2013). An end-to-end tool chain for Sensory Experience based on MPEG-V. *Signal Proc.: Image Comm.*, 28(2). Disponível em: http://dx.doi.org/10.1016/j.image.2012.10.009.

Waltl, M., Timmerer, C., e Hellwagner, H. (2009). A Test-Bed for Quality of Multimedia Experience Evaluation of Sensory Effects. In *First International Workshop on Quality of Multimedia Eexperience (QoMEX 2009)*, San Diego, CA, USA. Disponível em: http://dx.doi.org/10.1109/QOMEX.2009.5246962.

Yoon, K. R. e Han, J. J. (2012). *Text of ISO/IEC FDIS 23005-6 2nd edition Common Data Format*. Shanghai, China. MPEG Group Meeting, ISO/IEC JTC 1/SC 29/WG 11/N13065.