Pós-Graduação em Ciência da Computação

MARCELO FERNANDES DE SOUSA

UMA ABORDAGEM DE DESENVOLVIMENTO ORIENTADO A MODELOS PARA O DOMÍNIO DE APLICAÇÕES MULSEMEDIA

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE
2017

MARCELO FERNANDES DE SOUSA

UMA ABORDAGEM DE DESENVOLVIMENTO ORIENTADO A MODELOS PARA O DOMÍNIO DE APLICAÇÕES MULSEMEDIA

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação

Orientador: Prof. Carlos André Guimarães Ferraz
Co-Orientador: Prof. Raoni Kulesza

RECIFE
2017

Catalogação na fonte

Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

S725a Sousa, Marcelo Fernandes de

Uma abordagem de desenvolvimento orientado a modelos para o domínio de aplicações mulsemedia / Marcelo Fernandes de Sousa. – 2017.

157 f.: il., fig., tab.

Orientador: Carlos André Guimarães Ferraz.

Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2017.

Inclui referências e apêndices.

1. Ciência da computação. 2. Desenvolvimento dirigido por modelos. 3. Mulsemedia. I. Ferraz, Carlos André Guimarães (orientador). II. Título.

004 CDD (23. ed.) UFPE- MEI 2018-020

MARCELO FERNANDES DE SOUSA

UMA ABORDAGEM DE DESENVOLVIMENTO ORIENTADO A MODELOS PARA O DOMÍNIO DE APLICAÇÕES MULSEMEDIA

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação

Aprovado em: 11/09/2017

BANCA EXAMINADORA

_____
Prof. Dr. Carlos André Guimarães Ferraz (Orientador)
Centro de Informática / UFPE

_____
Prof^{o}. Dr. Sérgio Castelo Branco Soares (Avaliador Interno)
Centro de Informática / UFPE

_____
Prof^{a}. Dra. Veronica Teichrieb (Avaliadora Interna)
Centro de Informática / UFPE

_____
Prof^{o}. Dr. Celso Alberto Saibel Santos (Avaliador Externo)
Departamento de Informática / UFES

_____
Prof^{o}. Dr. André Menezes Marques das Neves (Avaliador Interno)
Departamento de Design / UFPE

_____
Prof^{o}. Dr. Windson Viana de Carvalho (Avaliador Externo)
Instituto Virtual / UFC

Para minha família...
Paloma, Assis, Nadia, Mykel e Myane

AGRADECIMENTOS

Primeiramente, agradeço ao autor da vida pelo privilégio de realizar este trabalho, por sempre guiar meus passos, colocar pessoas incríveis no meu caminho e renovar a minha fé e as minhas energias quando estavam se esvaindo.

À Paloma Fernandes, minha esposa, por todo amor, por todas as palavras de encorajamento, por todas as orações, pela paciência em passar tantos finais de semana e feriados sem a minha presença, por cuidar de mim e de tudo o que estivesse ao seu alcance para que eu me preocupasse apenas com esta Tese, por ter sido sábia e auxiliadora, e por toda a ajuda no trabalho. Sem ela seria impossível concluir esta pesquisa.

Ao professor Carlos Ferraz, por todas as orientações, pelas reuniões sempre produtivas e motivadoras, pela demonstração de confiança na pesquisa, por ser um modelo a ser seguido de docente, pelas conversas no almoço após as reuniões e, principalmente, pela amizade.

Ao professor Raoni Kulesza, pela co-orientação, pelo auxílio na definição do tema, por todo o conhecimento em engenharia de software compartilhado, pela total disponibilidade e atenção mesmo em meio a tantas atribuições, por abrir as portas do Lavid e da sua sala, pelas caronas após as reuniões, e, principalmente, pela amizade.

Aos professores membros da banca de defesa, pelos comentários e considerações que serviram para melhorar a minha pesquisa, em especial o professor Celso Saibel, que prontamente se dispôs a colaborar, tendo sido fundamental no entendimento do domínio estudado nesta pesquisa.

Aos colegas de pesquisa Ítalo Alves e Matheus Lima, pelas discussões que ajudaram a amadurecer as idéias, por auxiliarem na implemenção das provas de conceito e pelo suporte na execução dos experimentos.

Aos meus queridos alunos, docentes, colegas de trabalho e direção geral da IESP Faculdades, que sempre me apoiaram e incentivaram a concluir este trabalho.

Aos meus irmãos em cristo da Cidade Viva, em especial aos parceiros de ministério da Rede Nuvem, a minha conexão de casais e, principalmente, aos meus pastores-irmãos

Saulinho e Thiago e suas esposas Sabrina e Dayane, por todas as orações e, principalmente, pela amizade.

Aos meus avôs, tios, primos e todos os demais familiares que sempre torceram por mim e sempre passaram o melhor dos seus sentimentos. Em especial aos meus irmãos Mykel e Myane e meus cunhados Patrícia e Joel.

E finalmente, ao meu pai Assis e a minha mãe Nádia, por construírem uma família com fundação firme, por amarem a Cristo e buscarem seguir seus ensinamentos, por cada gota de suor derramado a fim de oferecer um futuro melhor para mim e meus irmãos, por cada livro comprado, por cada mensalidade do colégio paga, por cada conselho e palavras de incentivo. Honrar todo esse amor sempre foi e continuará sendo uma das minhas principais motivações de viver.

Esquecendo-me das coisas que para trás ficam e
avançando para as que diante de mim estão,
prossigo para o alvo.

—PAULO DE TARSO

RESUMO

As aplicações mulsemedia são aquelas que envolvem três ou mais dos sentidos humanos, promovendo o enriquecimento do conteúdo multimídia tradicional com novos objetos de mídia (olfativos, hápticos, etc.) e, consequentemente, o aumento da imersão e a melhoria da Qualidade de Experiência (QoE) do usuário, sendo a representação do conteúdo mulsemedia definida pelo padrão MPEG-V. Observando pesquisas, encontra-se na literatura esforços relacionados à autoria de efeitos sensoriais por meio de ferramentas, bem como pesquisas voltadas à reprodução e à renderização de efeitos sensoriais. Contudo, atualmente é possível identificar lacunas relacionadas ao sincronismo entre os objetos de mídia (áudio, vídeo, imagem, texto e efeitos sensoriais) que compõem uma aplicação mulsemedia; integração entre projetos de mídias, software e efeitos sensoriais; facilidades na integração de efeitos sensoriais com lógica imperativa e abstração das complexidades relacionadas a plataformas específicas de domínio de aplicação. Além disso, também é possível identificar uma lacuna na definição de processos, métodos e ferramentas que auxiliem o desenvolvimento sistemático de aplicações mulsemedia em conformidade com o padrão MPEG-V. O objetivo principal deste trabalho é propor uma abordagem de desenvolvimento orientado a modelos (MDD) que integre de modo sistemático as diferentes disciplinas envolvidas no desenvolvimento de aplicações mulsemedia contemplando soluções para as lacunas existentes. Nesta pesquisa, defende-se a tese de que o MDD pode reduzir a complexidade e diminuir o tempo de desenvolvimento de aplicações de mulsemedia, em especial, as que possuem como requisito forte integração com lógica de programação complexa. Para tanto, é realizada uma estruturação dos requisitos de famílias de aplicações; configuração das variabilidades de uma família por meio de um Modelo de *Features*; emprego de linguagens específicas de domínio para modelagem de visões que integram os projetos de mídia, software e os efeitos sensoriais; além da utilização de técnicas de metaprogramação para geração automática do código das aplicações em uma plataforma específica. Para demonstrar esta tese, é descrita uma abordagem MDD no domínio específico de mulsemedia e exemplos de uso. Por fim, é apresentado o projeto experimental envolvendo estudos empíricos quantitativos e qualitativos realizados com o intuito de determinar a viabilidade da abordagem, assim como os benefícios alcançados por meio da sua utilização. Os resultados obtidos mostram que a abordagem contribui para o avanço do desenvolvimento de software para aplicações mulsemedia: em média o tempo de desenvolvimento é 60% mais rápido quando comparado com outra abordagem não-MDD, e, em geral, 87,5% dos participantes classificaram a abordagem dirigida a modelos como extremamente útil ou útil no desenvolvimento de aplicações mulsemedia.

Palavras-chave: Desenvolvimento Dirigido por Modelos. Desenvolvimento Generativo. Linguagem de Domínio Específico. Mulsemedia. Efeitos Sensoriais.

ABSTRACT

The mulsemedia applications are those capable of engage three or more human senses, promoting the enrichment of multimedia traditional content with new media objects (olfactory, haptic, etc.) and, consequently, increasing immersion and improving the Quality of Experience (QoE). The mulsemedia content is represented by the MPEG-V standard. Observing previous research, we find in the literature efforts related to the authorship of sensorial effects through tools, as well as research aimed at reproduction and rendering of sensory effects. However, it is currently possible to identify a gap in the definition of processes, methods and tools to support the systematic development of mulsemedia applications in accordance with the MPEG-V standard. The main objective of this work is to propose a Model-driven approach (MDD) to integrate media, software and sensory effects projects. In this research, the thesis is argued that MDD can increase the productivity of the development of mulsemedia applications, in particular those with such strong integration requirement with complex programming logic. In particular, the structuring of the requirements of a applications family is performed; configuration of the common and variables parts of each category of applications through a Features Model; use of domain-specific languages for modeling views that integrate media design, software design, and sensory effects; and use of meta-programming techniques for automatic generation of application code. Finally, the experimental project is presented involving quantitative and qualitative empirical studies carried out with the purpose of determining the viability of the approach, as well as the benefits achieved through its use. The results show that the approach contributes to the development of software development for mulsemedia applications: on average the development time is 60% faster than the development without the use of MDD, and in general 87,5% of the participants classified that the approach as being extremely useful or useful in the development of mulsemedia applications.

Keywords: Model-driven Development. Generative Development. Domain-specific Language. Mulsemedia. Sensory Effects.

LISTA DE FIGURAS

LISTA DE TABELAS

ABREVIATURAS

ABNT	*Associação Brasileira de Normas Técnicas*
ADL	*Architecture Description Language*
API	*Application Programming Interface*
CIDL	*Control Information Description Language*
CIM	*Computational Independent Models*
CRITiCAL	*ConfiguRatIon Tool for Context Aware and mobiLe applications*
DCDV	*Device Capability Description Vocabulary*
DCU	*Dublin City University*
DCV	*Device Command Vocabulary*
DSL	*Domain Specific Language*
DSM	*Domain Specific Modeling*
EBNF	*Extended Backus–Naur Form*
ECGM	*Engine Cooperative Game Modeling*
EMF	*Eclipse Modeling Framework*
FM	*Feature Model*
FMP	*Feature Modeling Plugin*
FODA	*Feature-Oriented Domain Analysis*
FRASAD	*Framework of Sensor Application Development*
GQM	*Goal-Question Metric*
GSD	*Generative Software Development*
HCI	*Human Computer Interaction*
HTML5	*Hypertext Markup Language*
IIDL	*Interaction Interface Description Language*
IoT	*Internet of Things*
JET	*Java Emitter Template*
LoCCAM	*Loosely Coupled Context Acquisition Middleware*
LPS	*Linhas de Produto de Software*
M2C	*Model-to-Code*
M2M	*Model-to-Model*
M2T	*Model-to-Text*
MBD	*Model-based Development*
MBE	*Model-Based Engineering*
MDA	*Model Driven Architecture*
MDD	*Model-Driven Development*

MDE	*Model-driven Engineering*
MDGD	*Model-Driven Game Development*
MDSD	*Model-driven Software Development*
MML	*Multimedia Modeling Language*
MMORPG	*Massive Multiplayer Online Role-Playing Game*
MOF	*Meta-Object Facility*
MPEG	*Moving Picture Expert Group*
MulSeMedia	*Multiple Sensorial Media*
NCL	*Nested Context Language*
OMG	*Object Management Group*
PerGO	*Pervasive Game Ontology*
PIM	*Plataform Independent Model*
PLE	*Product Line Engineering*
PSM	*Platform Specific Model*
QoE	*Quality of Experience*
RAIL	*Reactive AI Language*
RoSE	*Representation of Sensory Effects*
RV	*Real para Virtual*
SAPV	*Sensor Adaptation Preference Vocabulary*
SBTVD	*Brazilian Digital Television System*
SCDV	*Sensor Capability Description Vocabulary*
SEDL	*Sensory Effect Description Language*
SEM	*Sensory Effect Metadata*
SEMP	*Sensory Effect Media Player*
SESim	*Sensory Effect Simulator*
SEV	*Sensory Effect Vocabulary*
SEVino	*Sensory Effect Video Annotation*
SIV	*Sensed Information Vocabulary*
SMIL	*Synchronized Multimedia Integration Language*
SMURF	*Sensible Media aUthoRing Factory*
SPL	*Software Product Line*
TEL	*Technology Enhanced Learning*
TS	*Transport Stream*
TVDI	*TV Digital Interativa*
UPV/EHU	*University of the Basque Country*
USPV	*User's Sensory Preference Vocabulary*

VR	*Virtual para Real*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A expressão ***mulsemedia*** (do inglês, *multiple sensorial media*) foi cunhada pelo pesquisador Gheorghita Ghinea em seu artigo tratando de Estado da arte, Perspectivas e Desafios (Ghinea, 2014) como uma resposta ao seguinte questionamento: "se os humanos aprendem e interagem por meio dos cinco sentidos, seria possível também fazê-lo digitalmente?". Mais precisamente, Ghinea argumenta que grande parte do conteúdo multimídia disponível atualmente é uma combinação de áudio, vídeo, e, algumas vezes, texto. Assim sendo, essas aplicações exploram apenas dois dos sentidos humanos (visão e audição), não considerando outras sensações como calor, umidade, aromas e sabores, que poderiam ser suportadas se esses conteúdos multimídia também engajassem os demais sentidos (paladar, olfato e tato). Diante do exposto, as aplicações mulsemedia são aquelas que envolvem três ou mais dos sentidos humanos, promovendo o enriquecimento do conteúdo multimídia tradicional com novos objetos de mídia (olfativos, hápticos, etc.).

Dessa forma, a mulsemedia visa estimular outros receptores sensoriais humanos como os mecanorreceptores, quimiorreceptores e termorreceptores, fato este que tende a contribuir para o aumento da imersão e a melhoria da Qualidade de Experiência (QoE) do usuário. Acerca deste ponto, Waltl, Timmerer e Hellwagner (2010) concluíram que, em média, existe um ganho de 10% quando se compara o audiovisual tradicional em relação ao conteúdo audiovisual enriquecido com efeitos sensoriais. Essa característica demonstra o potencial da mulsemedia e sua aplicabilidade em diversas áreas do conhecimento, como educação (ZOU *et al.*, 2017), publicidade (KRISHNA, 2012; PETIT *et al.*, 2015), uso terapêutico (ROBLES-BYKBAEV *et al.*, 2017), entretenimento (JALAL e MURRONI, 2017; ZHANG *et al*., 2016), dentre outras.

Vale destacar as iniciativas relacionadas à área de entretenimento. Notadamente, a indústria de filmes foi pioneira na exploração dos efeitos sensoriais, sendo os filmes 4D da Disney acrescidos de estímulos táteis e olfativos apresentados ao público desde os últimos 30-40 anos (GHINEA *et al.*, 2014). Um bom exemplo dessas aplicações é o chamado cinema 4D[2]. O já conhecido cinema 3D normalmente usa óculos para criar efeitos que saltam da tela criando a ilusão de volume e profundidade. Já no cinema 4D, além do efeito 3D, existe um ambiente para a reprodução dos filmes equipado com atuadores por meio dos quais costumam ser renderizados os seguintes efeitos sensoriais: movimento das cadeiras, vento, fumaça, bolhas de sabão, pulverizador d'água, aromas, etc. Cinemas e parques temáticos foram fortemente beneficiados pela mulsemedia, sendo esses exemplos os mais consolidados e que fazem parte do dia a dia das pessoas. Dessa forma, em um

[2] Outras nomenclaturas como 5D ou 7D também são facilmente encontradas. Referem-se à quantidade de efeitos sensoriais que os filmes possuem.

primeiro momento, as aplicações mulsemedia parecem ser uma realidade distante, pois necessitam de ambientes de execução com suporte de dispositivos atuadores capazes de reproduzir efeitos sensoriais. Além disso, existem dois desafiadores sentidos humanos, o paladar e o olfato, que não são facilmente estimulados por meio de dispositivos físicos. No entanto, o advento da IoT (do inglês, *Internet of Things*) aliado aos cenários cotidianos que se tornam cada vez mais ubíquos e repletos de dispositivos, como sensores e atuadores, abrem caminho para novas aplicações explorando a mulsemedia. Por exemplo, combinando *smartphones* a outros *gadgets* já é possível obter atuadores capazes de executar alguns efeitos sensoriais, como efeitos de luz, vibração e aromas. Além disso, pesquisas recentes (RANASINGHE *et al*., 2013; BRAUN e CHEOK, 2014; RANASINGHE e DO, 2016; SPENCE *et al*., 2017), apesar de apresentarem resultados incipientes, vêm avançando na investigação sobre como engajar os sentidos do paladar e do olfato indicando que em um futuro não tão distante a mulsemedia estará cada vez mais inserida no cotidiano da população.

Diante da relevância das aplicações mulsemedia surgiu à necessidade de normatização, pois boa parte das soluções que vinham sendo desenvolvidas eram proprietárias e, em muitos casos, patenteadas. Dessa forma, a comunidade MPEG (do inglês, *Moving Picture Expert Group*) iniciou em 2008 a normatização do padrão MPEG-V (ISO/IEC 23005) que já está em sua segunda versão publicada em 2013. Ele provê uma arquitetura e especifica representações de informação para conectar o mundo virtual (jogos, simuladores e aplicações) e o mundo real (sensores e atuadores). O MPEG-V envolve especificações sobre: capacidade de dispositivos, efeitos sensoriais, características de objetos do mundo virtual e formato de dados para interação entre dispositivos. Este padrão é destacado na Seção 4.2.

1.1 Desafios atuais

Ghinea (2014) afirma que o desenvolvimento de algoritmos e sistemas relacionados à mulsemedia ainda está em seus primórdios, sendo a construção dessas soluções um dos grandes desafios para os pesquisadores da área de multimídia nos próximos anos. Neste sentido, observando pesquisas anteriores para melhorar o suporte ao desenvolvimento de aplicações mulsemedia, é possível encontrar na literatura esforços relacionados à autoria de efeitos sensoriais (WALTL *et al*., 2013; KIM, 2013; CHOI; LEE e YOON, 2011; FREITAS *et al.* 2015). Além de ferramentas de autoria, também existem pesquisas recentes relacionadas à reprodução (*players*) e à renderização de efeitos sensoriais (SALEME e SANTOS, 2015; WALTL *et al*., 2013; CHO, 2010; KIM; JOO e LEE, 2013), bem como investigações sobre novos atuadores para efeitos sensoriais (RANASINGHE e DO, 2016; CHEOK, 2016; BRAUN e CHEOK, 2015; HARIRI *et al.*, 2016). Dessa forma, com base no estado da arte levantado é possível identificar as seguintes lacunas no suporte ao desenvolvimento de aplicações mulsemedia:

- **Sincronismo:** as ferramentas e linguagens relacionadas ao desenvolvimento de aplicações mulsemedia precisam dar suporte ao sincronismo entre os objetos de mídia (áudio, vídeo, imagem, texto e efeitos sensoriais) que a compõe a aplicação. Utilizando-se da categorização proposta por (CAZENAVE; QUINT e ROISIN, 2011) em relação à sincronização, as aplicações mulsemedia podem ser divididas em (i) aplicações dirigidas por mídia e (ii) aplicações dirigidas por eventos. As primeiras utilizam uma mídia principal de natureza contínua (áudio e vídeo) que serve como *backbone* por meio do qual os efeitos sensoriais são sincronizados na linha do tempo (*timeline*). Já as últimas, em vez de sincronismo por meio do paradigma de *timeline*, utilizam-se das relações temporais entre os objetos de mídia, como *links*, ou interações diretas com os usuários, como, por exemplo, um clique de botão que dispara um efeito sensorial. As ferramentas de autoria existentes tratam basicamente as aplicações dirigidas por mídias, seguindo o sincronismo na linha do tempo. Dessa forma, ainda não existe suporte consolidado de autoria para mídias discretas (imagem e texto) e sincronismo orientado a eventos. O suporte de linguagens também é limitado, sendo este tópico melhor abordado na Seção 4.4.1 desta tese, contudo, pesquisas recentes já trazem contribuições neste sentido (GUEDES; ALBUQUERQUE e BARBOSA, 2017).
- **Integração entre Projetos de Mídias, Software e Efeitos Sensoriais:** conforme mencionado, as soluções existentes suportam apenas a autoria de metadados de efeitos sensoriais e não as aplicações mulsemedia como um todo, ou seja, apesar da literatura apontar o potencial da mulsemedia em diversos domínios e áreas específicas, ainda não existe uma ferramenta ou processo de desenvolvimento que auxilie a criação de soluções que forneçam um suporte mais robusto no desenvolvimento de aplicações por meio de visões de projeto distintas (como estrutural, de apresentação e de interação) e que facilitem a implementação dos efeitos sensoriais de forma e integrada às mídias, tanto discretas quanto contínuas, além de dar suporte às questões relacionadas ao sincronismo.
- **Integração com lógica imperativa:** uma funcionalidade desejável que facilitaria a implementação de aplicações mulsemedia mais complexas é a possibilidade de integrar de forma simples os efeitos sensoriais a lógica imperativa. Isso possibilitaria a renderização de efeitos sensoriais de forma condicionada à resposta de serviços *Web*, como clima/tempo de uma região, localização geográfica; ou ainda, à resposta de uma consulta a um banco de dados, dentre outras situações possíveis.
- **Domínio de Aplicação:** quase todas as soluções de ferramentas existentes realizam a autoria de metadados de efeitos sensoriais em conformidade com o MPEG-V. O resultado dessas ferramentas é um arquivo XML denominado SEM (do inglês, *Sensoy Effect Metadada*) e, portanto, é considerada uma solução multiplataforma. Contudo, existe um grande esforço do

lado do domínio de aplicação (*Web*, *Mobile*, *Games*, IoT, etc.) em relação às adaptações necessárias para renderização dos efeitos sensoriais (TIMMERER; WALTL e HELLWAGNER, 2010). Dessa forma, faz-se necessário que as ferramentas de autoria também lidem com essas especificidades a fim de evitar que esforços em codificação de linguagem imperativa sejam necessários.

- **Abordagens de Desenvolvimento:** Sulema (2016) realiza um levantamento do estado da arte sobre software e hardware relacionado à mulsemedia. Deste levantamento é possível destacar que existem poucas soluções de software e nenhuma abordagem sistemática de desenvolvimento de aplicações. Notadamente, ainda existe uma carência na definição de abordagens de desenvolvimento que integrem de modo sistemático as diferentes disciplinas envolvidas no desenvolvimento de aplicações mulsemedia, envolvendo os diversos atores necessários para contemplar a sua natureza multidisciplinar, abstraindo as complexidades envolvidas relacionadas a plataformas de execução, atuadores e *players*, além de apontar soluções para as lacunas aqui discutidas.

A comunidade de engenharia de software constantemente desenvolve novas abordagens para melhorar o processo de desenvolvimento de software, sendo a utilização de um processo adequado considerado um fator chave no cumprimento da clássica restrição tripla de gerenciamento de projetos: tempo, orçamento e escopo (PMI, 2000). Uma das mais promissoras mudanças na área de engenharia de software nos últimos anos foi à adoção, tanto na indústria quanto na academia do Desenvolvimento Orientado a modelos (do inglês, *Model-Driven Development* - MDD) (KLEPPE; WARMER e BAST, 2003; KENT, 2002) como abordagem de desenvolvimento de software. A ideia principal do MDD está no fato dele considerar os modelos não apenas como uma documentação para tarefas de desenvolvimento e manutenção, mas como parte integrante do software, assim como o código. Os modelos permitem gerenciar os conceitos relacionados a um domínio e, através de uma série de transformações automáticas, é possível gerar código para plataformas específicas. Além disso, os modelos são importantes mecanismos de abstração por meio dos quais se facilita a integração de equipes multidisciplinares. Dessa forma, em virtude de tais características, o MDD pode contribuir com a estruturação dos passos necessários e diminuição da complexidade inerente ao desenvolvimento de aplicações mulsemedia. Corroborando esta argumentação e com objetivos de pesquisas semelhantes – redução da complexidade, diminuição do tempo de desenvolvimento e melhoria do reuso – pesquisas recentes têm aplicado MDD em domínios específicos com sucesso IoT (NGUYEN *et al*., 2015), *Mobile* (DUARTE *et al.*, 2015), *Games* (GUO *et al.*, 2015; PRADO e LUCREDIO, 2015; ZHU; ALF e HALLVARD, 2016), TV (MARQUES NETO, 2011; MEDEIROS *et al.*, 2015; KULESZA, 2013), Saúde (REYES; MUÑOZ-ARTEAGA e GONZÁLEZ-CALLEROS, 2017), entre outros.

Uma abordagem complementar ao MDD é o Desenvolvimento Generativo de Software (do inglês, *Generative Software Development* - GSD) (CZARNECKI, 2004), que procura diminuir a

distância semântica entre o espaço (domínio) do problema e o espaço (domínio) da solução, através de modelos de alto nível. O GSD tem como característica a utilização de tecnologias para automatização do desenvolvimento de software como, por exemplo, a meta-programação, a reflexão, a transformação de modelos etc. O objetivo é modelar e implementar uma "família" de sistemas de tal maneira que um determinado sistema pode ser automaticamente gerado a partir de especificações que refletem a solução descrita nos modelos, por meio de linguagens de domínio específico.

1.2 Tese

Este trabalho está contextualizado na premissa da literatura de que a combinação de conceitos e técnicas existentes no MDD em uma forma ainda inexplorada pode reduzir a complexidade, diminuir o tempo de desenvolvimento e melhorar o reuso no desenvolvimento aplicações mulsemedia, quando comparado com outras abordagens que não utilizam o MDD. Dessa forma, esta tese se concentra em identificar como aplicar o MDD por meio de uma abordagem de desenvolvimento de software no domínio da mulsemedia. A fim de investigar o impacto da adoção da abordagem, a principal questão de pesquisa a ser respondida é: **O uso de uma abordagem de desenvolvimento orientado a modelos aumenta a produtividade, em relação a tempo de implementação, do desenvolvimento de aplicações mulsemedia quando comparada com abordagens que não utilizam o MDD?** Para tanto, esta pesquisa visa responder às seguintes questões de pesquisas relacionadas à união entre tecnologias e ambientes de desenvolvimento mulsemedia e MDD:

- Quais são as etapas necessárias para aplicar o MDD no domínio das aplicações mulsemedia, de forma a tratar sua natureza interdisciplinar de desenvolvimento, mais especificamente, o projeto de software, projeto de mídia e de efeitos sensoriais das aplicações?
- Como representar as características comuns e variáveis no desenvolvimento de aplicação mulsemedia?
- Quais notações utilizar para permitir uma especificação integrada de um sistema em diferentes níveis de abstração e a partir de diferentes visões do projeto em aplicações mulsemedia?
- Quais são os artefatos necessários para conseguir uma geração de código (semi) automática de aplicações mulsemedia?
- Como tratar a integração dos efeitos sensoriais e lógica de programação imperativa por meio da abordagem proposta?
- Como avaliar o impacto do uso de uma abordagem generativa no tempo de desenvolvimento de aplicações mulsemedia?

Por meio da abordagem sistemática definida foram construídas três famílias de aplicações mulsemedia (*Multisensory Video*, *Multisensory Education* e *Multisensory E-commerce*) com base em técnicas de desenvolvimento generativo de software, sendo estas atividades utilizadas para avaliar o

presente trabalho. Também foram realizados estudos empíricos quantitativos visando avaliar a abordagem proposta e determinar o impacto no tempo de desenvolvimento de aplicações mulsemedia, bem como qualitativos a fim de obter opiniões gerais sobre a abordagem. Os resultados mostram que a abordagem contribui para o avanço do desenvolvimento de software para mulsemedia: em média o tempo de desenvolvimento é 60% mais rápido do que o desenvolvimento sem o uso de MDD, e em geral 87,5% das opiniões sobre a abordagem MDD são positivas (afirmaram ser extremamente útil e útil). Por fim, vale destacar as principais contribuições desta tese:

- Estruturação dos requisitos por meio da classificação de um conjunto de aplicações;
- Configuração das partes comuns e variáveis de cada categoria das aplicações com um Modelo de *Features;*
- Emprego de linguagens de domínio específico para modelagem das aplicações identificadas;
- Utilização de técnicas de metaprogramação para geração automática do código das aplicações;
- Extensão do metamodelo da linguagem MML (do inglês, *Multimedia Modeling Language*) (PLEUSS, 2014) para suportar a modelagem de efeitos sensoriais;
- Extensão do modelo HTML5 para suportar novas *tags* sensoriais, além do seu sincronismo com os demais objetos de mídia; e
- Integração dos efeitos sensoriais com serviços de lógica imperativa.

1.3 Objetivos

Visando contribuir com o processo de desenvolvimento de software de aplicações mulsemedia, esta tese tem como objetivo principal identificar como aplicar o MDD por meio de uma abordagem de desenvolvimento de software no domínio de aplicações mulsemedia. E como objetivos específicos:

- Definir as etapas necessárias para aplicar o MDD no domínio das aplicações mulsemedia, de forma a tratar sua natureza interdisciplinar de desenvolvimento, mais especificamente, os projetos de software, de mídia e de efeitos sensoriais das aplicações;
- Representar as características comuns e variáveis no desenvolvimento de aplicação mulsemedia por meio de técnica de análise de domínio;
- Definir notações necessárias para permitir uma especificação integrada de um sistema em diferentes níveis de abstração e a partir de diferentes visões do projeto em uma aplicação mulsemedia;
- Definir os artefatos necessários para conseguir uma geração de código (semi) automática de aplicações mulsemedia;
- Tratar da integração dos efeitos sensoriais e lógica de programação imperativa;

- Avaliar o impacto do uso de uma abordagem MDD no tempo de desenvolvimento de aplicações mulsemedia.

1.4 Definição do escopo

A mulsemedia é uma emergente área de pesquisa, sendo a saúde, educação, publicidade e entretenimento, dentre outras áreas do conhecimento, potenciais beneficiárias do aumento da imersão e da melhoria da QoE promovidas pela inserção de múltiplos efeitos sensoriais na multimídia tradicional (GHINEA, 2014; SULEMA, 2016). Além de possuir uma vasta aplicabilidade, a mulsemedia também exige um suporte tecnológico de hardware e software bastante significativo (SULEMA, 2016). Em relação ao hardware, uma gama de distintos atuadores é necessária para reprodução de diferentes efeitos sensoriais, como aromas, vibração, vento, variação de temperatura, etc. Além disso, em relação aos softwares, são necessários *players* para a renderização das mídias de efeitos sensoriais. Diante deste contexto e levando em consideração os objetivos desta pesquisa, é importante deixar claro que não fazem parte do escopo deste trabalho o desenvolvimento de novas soluções de hardware, tampouco a implementação de novos *players* para renderização de mídias sensoriais. Dessa forma, este trabalho optou por utilizar um cenário mais bem estabelecido em termos de tecnologias, fechando o escopo das aplicações mulsemedia no domínio *Web*/*Mobile*. Assim foi possível utilizar *browsers* como *players* para as mídias sensoriais e alguns atuadores existentes em celulares, sendo os efeitos mais complexos simulados. A principal motivação para esta estratégia foi permitir uma avaliação final através do uso da abordagem em cenários reais e que pudesse ser realizada sem custos elevados ou grandes dificuldades. Dessa maneira, presume-se que foi viável a realização de estudos empíricos com um esforço compatível com um trabalho de doutorado e também com avaliações e contribuições mais precisas.

Também é importante destacar não ser intenção deste trabalho investigar exaustivamente as linhas de pesquisa relacionadas às abordagens MDD e GSD, sendo essas duas técnicas combinadas para criação de uma metodologia nova voltada ao domínio de aplicações mulsemedia. Ainda sobre a abordagem, por uma questão de alinhamento com os objetivos de pesquisa, também não foi definido um processo de desenvolvimento completo, ficando de fora atividades como testes, manutenção e evolução dos artefatos gerados. Dessa maneira, os esforços foram direcionados para a definição dos artefatos e das atividades GSD e o mapeamento entre os espaços de problema e solução. Por fim, em relação ao suporte de ferramentas MDD, optou-se por utilizar as já existentes em virtude de serem soluções bastante maduras e empregadas tanto na indústria quanto na academia, sendo as opções adotadas também utilizadas por boa parte dos trabalhos relacionados a esta tese.

Sumarizando os pontos discutidos nesta Seção, são consideradas fora do escopo deste trabalho de tese as seguintes questões:

- Desenvolver novas soluções de hardware (atuadores) e de *players* para mídias sensoriais
- Definir as atividades de um processo completo para o desenvolvimento de aplicações mulsemedia;
- Prever atividades relacionadas à manutenção dos artefatos gerados, planejamento e execução de testes, e tratamento de mudança de requisitos;
- Criar novas tecnologias para o desenvolvimento generativo;
- Lidar com o projeto da interface gráfica do usuário considerando conceitos de usabilidade e experiência do usuário;

1.5 Estrutura da Tese

Neste primeiro Capítulo foram apresentadas a motivação e os objetivos da tese. O restante da tese está estruturado da seguinte maneira:

- No Capítulo 2, o estado da arte do desenvolvimento de aplicações mulsemedia é discutido;
- No Capítulo 3, discutem-se os fundamentos sobre a o desenvolvimento orientado a modelos, incluindo os principais conceitos, técnicas e abordagens existentes;
- No Capítulo 4, os conceitos gerais relacionados à mulsemedia são explicados;
- No Capítulo 5, uma visão geral da abordagem proposta é apresentada, bem como o seu detalhamento por meio do exemplo de uso da família de aplicações de vídeos multissensoriais – *Multisensory Video*;
- No Capítulo 6, são apresentados mais dois exemplos de uso utilizando a abordagem, além do projeto e resultados dos estudos empíricos realizados;
- No Capítulo 7, são debatidas as considerações finais, contribuições e trabalhos futuros;
- O Apêndice apresenta todos os dados levantados nos estudos empíricos qualitativos realizados para avaliar o impacto da abordagem MDD para o domínio de aplicações mulsemedia.

2 ESTADO DA ARTE DAS APLICAÇÕES MULSEMEDIA

Sulema (2016) realiza um levantamento do estado da arte sobre software e hardware relacionado à mulsemedia. Deste levantamento é possível destacar que existem poucas soluções de software e nenhuma abordagem sistemática de desenvolvimento de aplicações. Dessa forma, foram utilizados como referência trabalhos acadêmicos que tratam o desenvolvimento de aplicações mulsemedia por meio de ferramentas de autoria de efeitos sensoriais. O principal objetivo dessas ferramentas é propiciar facilidades ao autor de conteúdo multissensorial através da integração de metadados a estes conteúdos. Além disso, também foram considerados estudos que discutem os conceitos e viabilidade de uso do MDD em domínios específicos, embora nenhum deles seja voltado a mulsemedia. Portanto, nesta Seção é descrita uma análise desses estudos organizados da seguinte forma: a Seção 2.1 discute ferramentas para o desenvolvimento de aplicações mulsemedia; a Seção 2.2 apresenta as metodologias de desenvolvimento MDD para domínios específicos e, por fim, a Seção 2.3 realiza uma discussão sobre os assuntos abordados.

2.1 Desenvolvimento de aplicações Mulsemedia

O SEVino (do inglês, *Sensory Effect Video Annotation*) (WALTL *et al.*, 2013) é uma ferramenta de autoria por meio da qual é possível anotar vídeos com metadados de efeitos sensoriais gerando descrições compatíveis com o padrão MPEG-V. A ferramenta está na versão 2.0 que suporta efeitos sensoriais de vento, vibração, luz, temperatura, pulverizador d'água, aroma e névoa. As informações relacionadas à autoria dos efeitos são armazenadas em um arquivo SEM (do inglês, *Sensory Effects Metadata*) por meio da interface gráfica da ferramenta. Ela também conta com um simulador de efeitos sensoriais integrado a sua interface, chamando SESim (do inglês, *Sensory Effect Simulator*). Em relação aos detalhes de implementação, o SEVino possui código aberto e foi desenvolvido com a linguagem Java. Para a reprodução de vídeos é utilizada a API VLCJ[4] e o *framework* multimídia FFmpeg[5] é empregado para extrair quadros do vídeo a fim de disponibilizá-los em uma linha do tempo para facilitar a navegação do usuário.

Possuindo objetivos similares ao SEVino, a ferramenta de autoria SMURF (do inglês, *Sensible Media aUthoRing Factory*) (KIM, 2013) também realiza anotação de metadados de efeitos sensoriais em vídeos, contudo, diferentemente do SEVino, oferece um suporte maior a alguns elementos da linguagem SEDL definida na parte 3 do padrão MPEG-V, como elementos para criação de grupos de efeitos sensoriais que são renderizados de forma conjunta, dentre outros. Além disso, seus autores

[4] VLCJ: Disponível em <https://github.com/caprica/vlcj>

[5] FFMPEG: Disponível em <https://www.ffmpeg.org/>

apontam como diferenciais a interface *Web* mais facilmente acessível e intuitiva, funções de validação de instâncias XML automáticas e funcionalidades de autoria mais eficientes. Como alguns detalhes de implementação, sua interface gráfica foi desenvolvida com *Adobe Flex*[6] e o sistema de E/S é implementado em Java. A ferramenta oferece suporte aos efeitos sensoriais de luz, *flash* de luz, temperatura, vento, vibração, pulverizador d'água, aroma e névoa.

O RoSE *Studio* (do inglês, *Representation of Sensory Effects*) (CHOI; LEE e YOON, 2011) é mais uma ferramenta para a anotação de metadados de efeitos sensoriais em vídeos. Ela faz parte de um *framework* para transmissão de o serviço multissensoriais 4D baseado no padrão MPEG-V e, dessa forma, oferece a possibilidade de gerar o arquivo de metadados SEM multiplexado com o conteúdo do vídeo para ser transmitido usando o padrão MPEG-2 TS (*Transport Stream*).

Freitas *et al.* (2015) apresentam um sistema de informação para gestão e visualização de conteúdos multissensoriais. Trata-se de um sistema composto por um editor de efeitos sensoriais, um reprodutor e um servidor para guardar as experiências criadas. Da mesma forma que as ferramentas anteriores, uma mídia de vídeo serve como base e por meio dela são inseridos os efeitos sensoriais, bem como seus parâmetros, inclusive o atuador que estará vinculado. Assim que o desenvolvedor concluir a edição do conteúdo multissensorial, este é enviado para o servidor onde será armazenado.
A ferramenta foi construída utilizando o *framework* .NET (com a linguagem C#) da Microsoft. Já no servidor foi utilizada a linguagem PHP, base de dados *MySQL* e os *Web Services* em linguagem Java. Para a sincronização dos efeitos sensoriais foi utilizada a linguagem SMIL. Não há menção alguma sobre a implementação realizada levar em consideração a representação definida para os efeitos sensoriais no padrão MPEG-V. Além disso, a implementação está bastante vinculada ao hardware utilizado para executar os efeitos sensoriais, fato este que limita o reuso da solução.

2.2 Metodologias de desenvolvimento MDD para outros domínios

Tendo a abstração, reuso e automação como princípios chaves, o MDD tem sido amplamente utilizado em diferentes domínios com sucesso. Nesta subseção, são discutidos alguns aspectos relacionados a trabalhos recentes que aplicam a abordagem MDD em outros domínios, bem como são discutidas as influências e diferenças quando comparados ao presente trabalho.

Os trabalhos (PLEUSS, 2014; PLEUSS e HUSSMANN, 2011; PLEUSS, 2009) apresentam uma abordagem MDD que tem por objetivo integrar os projetos software, interface com usuário e mídias, ou seja, ela visa integrar os conceitos das áreas de aplicações interativas e aplicações multimídia por meio da linguagem MML (do inglês, *Multimedia Modeling Language*), uma DSL para modelagem de aplicações multimídia interativas. Os modelos MML são baseados na especificação

[6] *Adobe Flex*: disponível em <http://www.adobe.com/br/products/flex.html>

UML 2.0, sendo voltados para geração automática de código para múltiplas plataformas. Neste contexto, é útil distinguir os tipos de mídia que precisam de suporte em questões específicas relacionadas ao desenvolvimento de software. Dito isto, o presente trabalho adota a linguagem MML como DSL, estendendo os seus metamodelos para suportar mídias relacionadas aos efeitos sensoriais.

Outra área que vem recebendo bastante atenção da comunidade MDD é a de Jogos, sendo denominado aqui como MDGD (do inglês, *Model-Driven Game Development*). Contudo, o desenvolvimento de jogos baseado em *engines* é bastante consolidado no segmento de jogos e, dessa forma, o grande desafio da área consiste em aplicar MDD conciliando adequadamente as duas metodologias de desenvolvimento. Endereçando a questão exposta, Zhu, Wang e Trætteberg (2016) propõem uma abordagem hibrida chamada ECGM (do inglês, *Engine Cooperative Game Modeling*). Para uma melhor compreensão, é necessário entender que *engines* de jogos costumam possuir um conjunto de ferramentas por meio das quais os dados dos jogos, como *layout* de mundo e informações de personagens, são facilmente criados ou modelados. Dessa forma, os autores apontam como diferencial do trabalho o fato do ECGM usar geração de código e técnicas de transformação entre modelos não apenas para a geração do código relacionados à jogabilidade, mas também aos dados mencionados. A abordagem ECGM foi avaliada por meio de um estudo de caso em que uma DSL chamada RAIL (do inglês, *Reactive AI Language*) para jogos de ação e aventura foi implementada seguindo a abordagem proposta e tendo a *engine* Torque 2D como plataforma alvo, contudo, o trabalho não realiza estudos empíricos para avaliar os resultados de forma quantitativa.

Na área de TV Digital Interativa (TVDI), os trabalhos (KULESZA, 2013; KULESZA *et al.*, 2012) propõe uma abordagem MDD que procura uma melhor integração entre os projetos de mídia e software. Nesta pesquisa defende-se a tese de que o MDD pode aumentar a produtividade do desenvolvimento de aplicações de TV Digital, principalmente, as que possuem como requisito forte integração entre os objetos de mídia e a lógica da aplicação. Para tanto, é realizada uma estruturação dos requisitos de uma família de aplicações; configuração das partes comuns e variáveis de cada categoria das aplicações através de um Modelo de Características (do inglês, *Feature Model* - FM); emprego de linguagens específicas de domínio para modelagem de visões que integram o projeto de mídia e projeto de software; e utilização de técnicas de metaprogramação para geração automática do código das aplicações. Por fim são apresentados os resultados de uma avaliação envolvendo estudos empíricos, que buscou determinar a viabilidade da abordagem e os benefícios que podem ser alcançados com o emprego da mesma. Apesar de tratar um domínio diferente, a abordagem proposta nesta tese procura seguir alguns passos e adotar estratégias definidas em Kulesza (2013), como inclusão de uma atividade para estruturar os requisitos de uma família de aplicação Mulsemedia por meio de FMs, além da adoção da linguagem MML como DSL para modelar os *templates* das famílias de aplicação Mulsemedia. Os benefícios da utilização de DSLs e FMs são defendidos por Voelter e Visser (2011), sendo a utilização de forma conjunta destas duas técnicas responsáveis por ampliar as

possibilidades de derivação de produtos de uma LPS, facilitando o projeto do domínio por meio da definição e configuração dos *templates*.

Outra pesquisa dentro do contexto discutido nesta Seção foi desenvolvida por Reyes, Muñoz-Arteaga e González-Calleros (2017) em que os autores propõem um processo de desenvolvimento MDD para a produção de ambientes interativos voltados a terapia ocupacional. Para atingir este objetivo, o processo considera o envolvimento de um time de peritos multidisciplinar (equipes das áreas de saúde e de engenharia de software) para a construção dos ambientes interativos adaptados às necessidades de pacientes. Assim como esta tese, o trabalho adota a linguagem MML para modelar o ambiente interativo e se utiliza dos modelos para promover melhor integração entre a equipe multidisciplinar. Para validar a proposta, foi realizado um estudo de caso apresentando o desenvolvimento do ambiente interativo "Colocando objetos na cesta" (do inglês, *Placing objects in the basket)*. Este ambiente tem como objetivo para auxiliar na reabilitação de pacientes com limitações nos membros superiores por meio de *feedbacks* das tarefas realizadas capturados por meio do controlador LeapMotion[7] (plataforma tecnológica) que permite rastrear o movimento das mãos. Vale salientar que o trabalho em questão não explora técnicas de engenharia de software generativa que poderiam trazer benefícios na geração automática das aplicações por meio de ferramentas com maior poder de abstração, como *Wizards*, facilitando o desenvolvimento dos ambientes pelos usuários não especialistas. Para tanto, o processo proposto precisaria ser reorganizado a fim de prever uma etapa de engenharia de aplicação conforme define Czarnecki e Eisenecker (2000), abordagem esta explorada nesta tese.

No contexto da computação pervasiva, Duarte *et al.* (2015) propõe uma abordagem MDD chamada CRITiCAL (do inglês, *ConfiguRatIon Tool for Context Aware and mobiLe applications*) para a geração de aplicações móveis e sensíveis ao contexto. Neste domínio de aplicações, a utilização de *middlewares* é uma estratégia recorrente para resolver o problema relacionado à heterogeneidade de dispositivos e complexidade nos códigos para acessar os sensores. Dessa forma, a abordagem se propõe a combinar o paradigma MDD com o *middleware* LoCCAM (do inglês, *Loosely Coupled Context Acquisition Middleware*). Este *middleware* é adotado por permitir a aquisição adaptativa de informações contextuais em dispositivos Android[8]. O trabalho também propõe uma DSL visual chamada ContextRuleML, que tem como objetivo a modelagem de informações contextuais e de regras contextuais para criação de aplicações móveis e sensíveis ao contexto que utilize a plataforma de *middleware* para a aquisição contextual. A ContextRuleML objetiva gerar um esqueleto de projeto Android devidamente configurado para a utilização do *middleware*, incluindo um acesso transparente

[7]LeapMotion: Disponível em <http://blog.leapmotion.com>

[8]Android: Disponível em <http://developer.android.com/sdk/index.html>

às informações contextuais desejadas, de modo a abstrair ao desenvolvedor os detalhes da arquitetura interna do *middleware*. Este trabalho apresenta objetivos de pesquisa semelhantes ao desta tese, e também explora a utilização de *templates*, contudo não se utiliza do conceito de família de aplicações. Em relação à avaliação da abordagem, não são apresentados estudos empíricos quantitativos realizados com grupos de desenvolvedores a fim de avaliar redução de tempo de desenvolvimento, mas apresenta (i) avaliação de usabilidade por meio de questionário, (ii) avaliação da qualidade do código gerado e, por último, (iii) avaliação de desempenho.

2.3 Considerações sobre o Capítulo

Neste Capítulo foram apresentados trabalhos que lidam com o desenvolvimento de aplicações mulsemedia por meio de ferramentas de autoria. Além disso, também foram considerados estudos que discutem os conceitos e viabilidade de uso do MDD em domínios específicos. A Tabela 1 sumariza os trabalhos discutidos no Capítulo. A fim de melhor caracterizá-los, eles foram divididos entre ferramentas e abordagens. Também foram destacados quais dos trabalhos são relacionados à mulsemedia e, além disso, foi explicitado qual o domínio de aplicação utilizado nos estudos de casos realizados. Com base no levantamento realizado, é possível observar que esta Tese propõe pela primeira vez uma abordagem de desenvolvimento no domínio mulsemedia apoiada por ferramenta com foco no domínio de aplicações Web.

Tabela 1: Trabalhos Relacionados

Trabalho	**Ferramenta**	**Abordagem**	**Mulsemedia**	**Domínio de Aplicação dos Estudos de Caso**
SEVino (WALTL *et al.*, 2013)	X		X	Video Interativo
SMURF (KIM, 2013)	X		X	Video Interativo
RoSE Studio (CHOI; LEE; YOON, 2011)	X		X	Vídeo Interativo
Freitas *et al.* (2015)	X		X	Web
Pleuss (2014)		X		Jogos
Zhu, Wang e Trætteberg (2016)		X		Jogos
Kulesza (2013)	X	X		TV Digital
Reyes, Muñoz-Arteaga e González-Calleros (2017)		X		Ambientes Interativos
Duarte *et al*. (2015)		X		Dispositivos Móveis
Tese	X	X	X	Web

Ghinea (2014) afirma que o desenvolvimento de aplicações mulsemedia ainda está em seus primórdios, sendo o desenvolvimento destas soluções um dos grandes desafios para a multimídia. Neste sentido, a literatura na área relacionada a metodologias e ferramentas para apoiar o seu desenvolvimento ainda é bastante restrita. Acerca deste ponto, é possível identificar duas lacunas principais no suporte ao desenvolvimento de aplicações mulsemedia:

- estabelecimento de ambientes de desenvolvimento que ofereçam suporte aos pontos discutidos na Seção 1.1 de desafios atuais: sincronismo entre os objetos de mídia (áudio, vídeo, imagem, texto e efeitos sensoriais) que compõem uma aplicação mulsemedia; integração entre projetos de Mídias, Software e Efeitos Sensoriais; facilidades na integração de efeitos sensoriais com lógica imperativa; e por fim, abstração das complexidades específicas de domínio de aplicação;
- definição de abordagens de desenvolvimento que integrem de modo sistemático as diferentes disciplinas envolvidas no desenvolvimento de aplicações mulsemedia contemplando a soluções para as lacunas citadas em (i).

Por fim, as pesquisas abordadas neste Capítulo relacionadas ao desenvolvimento de aplicações mulsemedia e também os trabalhos sobre abordagens de desenvolvimento orientado a modelos influenciaram as decisões e contribuíram com soluções adotadas na abordagem desenvolvida nesta tese.

3 DESENVOLVIMENTO DE SOFTWARE ORIENTADO A MODELOS

Neste Capítulo são apresentados os principais conceitos e abordagens relacionados ao desenvolvimento de software orientado a modelos. Portanto, nesta Seção é descrita uma análise desses estudos organizados da seguinte forma: a Seção 2.1 discute ferramentas para o desenvolvimento de aplicações mulsemedia; a Seção 2.2 apresenta as metodologias de desenvolvimento MDD para domínios específicos e, por fim, a Seção 2.3 realiza uma discussão sobre os assuntos abordados.

3.1 Modelos, Metamodelos e Meta-Metamodelos

Silva (2015) realizou uma pesquisa do tipo *survey* na qual traz uma compilação do conceito de modelos segundo diversos autores de Engenharia de Software e da Engenharia Dirigida por Modelos (do inglês, *Model-driven Engineering* – MDE). Para Seidewitz (2003), modelo é um conjunto de instruções sobre o sistema em estudo. Já Kühne (2006) define modelo como uma abstração de um sistema (real ou baseado na linguagem) que permite a realização de previsões ou inferências. Selic (2003) conceitua modelo como uma representação reduzida de algum sistema que destaca as propriedades de interesse a partir de um determinado ponto de vista. Por fim, apesar das diversas outras definições na literatura, Bezivin e Gerbe (2001) apresenta modelo como uma simplificação de um sistema construído com um objetivo pretendido em mente, de tal forma que um modelo seja capaz de responder a perguntas no lugar do sistema original.

Diante das definições mencionadas, é possível perceber que modelos são abstrações realizadas para simplificar o entendimento dos sistemas. Dessa forma, o conceito de abstração é bastante intrínseco ao de modelos. Trata-se de uma das principais técnicas com que a mente humana enfrenta a complexidade, ocultando o que é irrelevante num sistema complexo a fim de reduzi-lo a algo compreensível. Um exemplo bastante familiar extraído das aulas de biologia é o estudo dos sistemas que compõe o corpo humano: cardiovascular, respiratório, digestório, nervoso, sensorial, endócrino, excretor, urinário, reprodutor, esquelético, muscular, dentre outros. Cada um deles envolve órgãos que atuam para a realização das funções vitais do organismo e normalmente são estudados, pelo menos em um primeiro momento, individualmente. Por exemplo, quando se estuda o sistema muscular, os demais sistemas são abstraídos a fim de simplificar e facilitar o entendimento. Quando se trata de engenharia de software, um sistema pode ser compreendido como um conceito genérico para representar uma aplicação, plataforma ou outro artefato de software (SILVA, 2015). É bastante útil abstrair os detalhes tecnológicos de implementação e tratar dos conceitos do domínio de forma mais direta utilizando um ou mais níveis de abstração. Assim, um modelo de um sistema pode funcionar como um meio de comunicação entre usuários, analistas e programadores de forma a esconder ou diminuir os aspectos relacionados às tecnologias de implementação.

Modelos podem ser definidos também a partir de um modelo chamado Metamodelo. Silva (2015) também lista uma série de autores que conceituam metamodelo. Notadamente, destaca-se a definição da OMG (do inglês, *Object Management Group*) que apresenta metamodelo, como sendo um modelo que define a linguagem para expressar um modelo (OMG, 2003). Já para Favre e Nguyen (2005), um metamodelo é um modelo de uma linguagem de modelos. Ainda sobre este assunto, Seidewitz (2003) afirma que um metamodelo é um modelo de especificação para o qual os sistemas em estudo que estão sendo especificados são modelos de uma determinada linguagem de modelagem. O ponto comum entre as três definições apresentadas é que metamodelos definem uma linguagem de modelagem. Por meio de uma linguagem de modelagem que se define a estrutura, a semântica e as restrições (sintaxe) de um grupo de modelos (MELLOR, 2004).

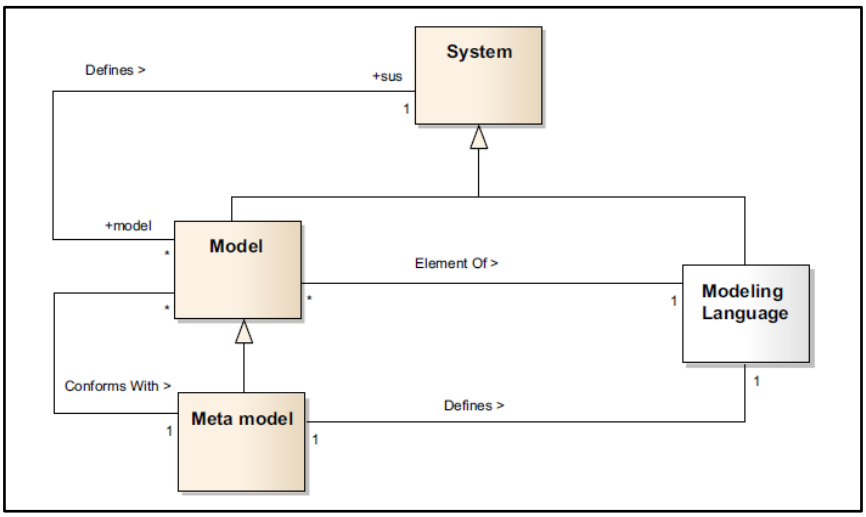


Figura 1: Relações entre modelo, linguagem de modelagem, metamodelo (SILVA, 2015)

Para uma melhor compreensão, na Figura 1, Silva (2015) se utiliza de diagramas de classes UML para representar a relação existente entre modelos, metamodelos e linguagem de modelagem. Por meio dela podemos inferir que:

- através do relacionamento **elemento de** (do inglês, *ElementOf*) entre as classes **Modelo** (do inglês, *Model*) e **Linguagem de Modelagem** (do inglês, *Modeling Language*), uma linguagem de modelagem é um conjunto de modelos, ou ainda, um modelo é um elemento de uma linguagem de modelagem;
- através do relacionamento **define** (do inglês, *Defines*) entre as classes **Metamodelo** (do inglês, *Meta model*) e **Linguagem de Modelagem**, um metamodelo é um modelo da estrutura de uma linguagem de modelagem, ou ainda, uma linguagem de modelagem é definida por um metamodelo;
- através do relacionamento **conforme** (do inglês, *ConformsWith*) entre as classes **Modelo** e **Metamodelo**, um modelo estar em conformidade com um metamodelo significa que ele deve satisfazer as regras definidas pelo seu metamodelo;

através do relacionamento **define** entre as classes **Sistema** (do inglês, *System***)** e **Modelo**, um sistema consiste em um conceito genérico para designar uma aplicação, plataforma de software ou qualquer outro artefato de software e costuma ser representado por um modelo, conforme a definição de modelos apresentadas.

A metamodelagem acarreta um problema devido à necessidade de existir um modelo em uma camada de abstração maior que a defina o metamodelo. Para superar esse entrave e definir um metamodelo que inicie a hierarquia é necessário utilizar uma linguagem que descreva a si mesma com a sua própria linguagem e esse modelo é definido como meta-metamodelo. Para uma melhor compreensão sobre meta-metamodelagem, é possível tomar como exemplo o caso dos dicionários de português, pois eles definem uma linguagem (a língua portuguesa) utilizando a própria língua portuguesa. Em outras palavras, a meta-meta modelagem configura uma hierarquia de metamodelos e tem como responsabilidade formar uma linguagem para a especificação de metamodelos (OMG, 2003).

Na área de metodologias de desenvolvimento de software, Demarco (1979) estabeleceu o conceito de desenvolvimento baseado em modelos (do inglês, *Model-Based Development* – MBD). No MBD é destacado que antes do desenvolvimento de um sistema de software deve ocorrer a construção de um modelo. Trata-se de um exercício de engenharia, assim como ocorre na engenheira civil em que o engenheiro antes de construir um prédio cria a sua planta (modelo). O modelo se foca sobre o mundo real, identificando, classificando e abstraindo os elementos que constituem o problema com o objetivo de organizá-los em uma estrutura formal. Contudo, existem dois problemas recorrentes em processos de desenvolvimento MBD:

- **o problema da produtividade, documentação e manutenção:** num processo de desenvolvimento baseado em modelos, a conexão entre os diagramas e o código vai se perdendo gradualmente quando se alcança a fase de codificação. Na prática, apesar dos modelos ajudarem no início do desenvolvimento para a geração inicial do código-fonte, os desenvolvedores não sincronizam mais as atividades, gerando uma desatualização dos modelos perante o código-fonte desenvolvido. A principal consequência desse comportamento é dificultar a manutenção em médio prazo do sistema.
- **o problema da flexibilidade e das mudanças tecnológicas**: na indústria de software, de tempos em tempos, surgem novas tecnologias que rapidamente se disseminam e as organizações precisam adotá-las para se manterem competitivas, além de se adequarem às novas soluções que essas tecnologias oferecem e que os clientes exigem. Nesse caso, o problema surge em virtude da quantidade de esforço gasto em tarefas específicas da plataforma, esforço este que não é reutilizado em outras plataformas.

3.2 Desenvolvimento orientado a modelos

O desenvolvimento orientado a modelos (do inglês, *Model-Driven Development* – MDD), também conhecido como desenvolvimento de software orientado a modelos (do inglês, *Model-driven Software Development* – MDSD), surgiu com o objetivo de amenizar os problemas citados na Seção anterior (KLEPPE; WARMER e BAST, 2003). Nessa abordagem, os modelos já não são tratados apenas como a documentação do software, mas fazem parte dele, constituindo um fator decisivo para aumentar a velocidade e a qualidade do seu desenvolvimento (CZARNECKI; STAHL e VOELTER, 2006). No MDD, os modelos são artefatos centrais e equivalentes aos próprios códigos, pois a partir deles, é possível gerar completamente o código da implementação, permitindo assim que o engenheiro de software não necessite manipular manualmente o código-fonte, concentrando-se em modelos de mais alto nível e abstraindo as complexidades de implementação de diferentes plataformas tecnológicas. Para tanto, os modelos são gerados de níveis mais abstratos para níveis mais concretos, sendo mapeados por meio de mecanismos de transformação automática que geram sucessivos refinamentos até completar a última transformação.

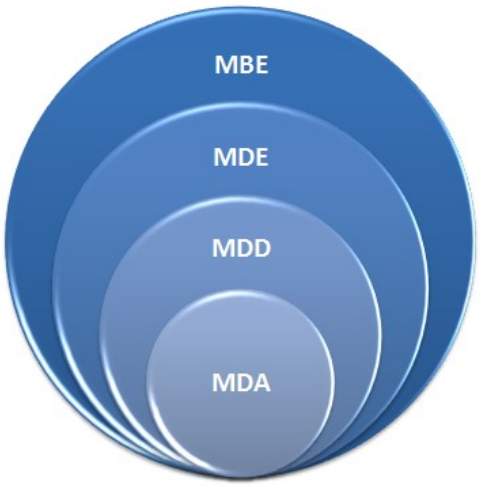


Figura 2: Relacionamento entre os diferentes acrônimos MD* (BRAMBILLA; CABOT e WIMMER, 2017)

Na área de desenvolvimento orientado a modelos existe uma grande quantidade de termos utilizados muitas vezes sem o devido entendimento dos significados. A fim de se obter clareza sobre isso, este trabalho adota as definições propostas por Brambilla, Cabot e Wimmer (2017) que apresentem suas relações como ilustradas na Figura 2. Iniciando a diferenciação por meio do termo mais abrangente, o superconjunto MBE (do inglês, *Model-Based Engineering*) pode ser considerado um sinônimo do já apresentado MBD (do inglês, *Model-Based Development*), referindo-se a um processo no qual os modelos de software, apesar de apresentarem importantes papéis, não são necessariamente considerados artefatos-chave do desenvolvimento. Já o MDD como definido

anteriormente, é a implementação (semi) automática, ou seja, gerada a partir dos modelos, uma das suas principais características. Ele é considerado subconjunto do MDE (do inglês, *Model-Driven Engineering*) que vai além do desenvolvimento puro de atividades, englobando outras tarefas baseadas em modelos de um processo completo de Engenharia de Software. Por fim, o MDA (do inglês, *Model-Driven Architecture*) é uma visão particular do MDD proposta pela OMG (do inglês, *Object Management Group*). Dessa forma, o MDA é um subconjunto do MDD e nele a modelagem e as linguagens de transformações utilizadas são padronizadas pelo OMG. Mais adiante o MDA será abordado em maiores detalhes.

A automatização das transformações é uma das características principais do MDD, e ela não se trata de uma atividade elementar, necessitando que um conjunto de elementos (ferramentas, transformadores, etc.) seja combinado a fim de viabilizar a abordagem. O elemento central do MDD são os modelos, e para a criação deles é necessária uma ferramenta de modelagem. Utilizando essa ferramenta, o engenheiro de software produz modelos que descrevem os conceitos do domínio e servem de entrada para transformadores que irão gerar outros modelos ou códigos-fonte. Para definir essas transformações também se faz necessário uma ferramenta que permita ao engenheiro de software construir regras de mapeamento de modelo para modelo ou de modelo para código. Finalmente, é necessário um mecanismo que efetivamente aplique as transformações definidas. Esse mecanismo deve não só executar as transformações, mas também manter as informações de rastreabilidade, possibilitando saber à origem de cada elemento gerado, sejam modelo ou código fonte (LUCRÉDIO, 2009).

Os principais benefícios do MDD de acordo com Liddle (2011), Pons, Gladini e Perez (2010) e Kleppe, Warmer e Bast (2003) são:

- **Produtividade:** apesar de existir um esforço inicial na modelagem e implementação das transformações, o MDD pode trazer um aumento na produtividade em relação ao tempo de desenvolvimento de software. O que ocorre é um redirecionamento dos esforços das atividades de arquiteturação e codificação do software, sendo utilizada uma maior fatia do tempo na modelagem e no entendimento do sistema. Entretanto, um único modelo é capaz de gerar por meio das transformações uma quantidade considerável de código;
- **Portabilidade e Reuso:** os modelos representam uma abstração do sistema, e não o sistema em si. Dessa forma, podem existir modelos em uma camada de abstração independente de plataforma, e a partir dele, serem gerados modelos dependentes de plataforma, além do próprio código-fonte. Isso significa que o MDD lida com mudanças tecnológicas mais facilmente, necessitando apenas da implementação de novas transformações para uma nova plataforma tecnológica, garantindo um melhor reuso da modelagem. Vale salientar que o reuso é realizado em nível de modelos, e não em nível de

código, sendo os modelos adaptados ou refinados, a fim de atender novos requisitos para posteriormente o código ser gerado.

- **Manutenção e documentação:** no desenvolvimento tradicional, durante a atividade de manutenção, é comum os desenvolvedores inserirem as modificações diretamente no código, não voltando para atualizar a documentação previamente realizada. Como os modelos são elementos centrais no MDD, as manutenções são realizadas diretamente neles, fato este que garante a consistência com o código-fonte e garante a documentação atualizada constantemente, tornando as tarefas de manutenção bem mais simples;
- **Comunicação:** a utilização de modelos consiste em um meio mais eficaz de comunicação entre os diferentes profissionais que formam uma equipe de desenvolvimento de software, uma vez que os modelos são mais abstratos que o código-fonte. Isso também possibilita que especialistas do domínio exerçam um papel mais ativo no processo, podendo utilizar diretamente os modelos para identificar os conceitos do negócio, enquanto especialistas em TI podem se concentrar nos elementos técnicos;
- **Correção:** a geração automática a partir dos modelos e das transformações evita falhas humanas de codificação, trazendo assim ganhos para a qualidade do software. Além disso, é possível identificar erros conceituais em um nível mais alto de abstração por meio dos próprios geradores.

Por outro lado, existem alguns pontos negativos na abordagem apontados por Thomas (2004) e Ambler (2003):

- **Complexidade:** as ferramentas de modelagem e geradores de código utilizados na abordagem MDD agregam uma maior complexidade ao processo, pois são mais difíceis de construir e manter;
- **Desempenho e flexibilidade:** o desempenho do código pode ser menor quando comparado a um código escrito manualmente, uma vez que os geradores de código podem adicionar, dependendo das regras de transformação, código inútil ou não otimizado. Ademais, outra consequência em detrimento de como os geradores de código foram implementados é o aumento da rigidez do software produzido;
- **Curva de aprendizado:** a mudança da abordagem de desenvolvimento tradicional para a abordagem MDD implica na necessidade da equipe de desenvolvimento adquirir um conhecimento específico e aprofundado sobre construção de linguagens, ferramentas de modelagem, transformações e geradores de código. Esse conhecimento permitirá a utilização e a criação de artefatos MDD, sendo importante destacar que o aprendizado dessas técnicas requer certo tempo para o treinamento;

- **Alto investimento inicial:** a adoção do MDD requer um maior investimento inicial, uma vez que a construção de uma infraestrutura de reutilização orientada a modelos requer mais tempo e esforço. Além disso, como já discutido, também é necessário o investimento na capacitação da equipe de desenvolvimento, seja por meio de treinamentos ou contratação de desenvolvedores especializados em MDD.

3.3 Arquitetura dirigida por modelos

Conforme já mencionado, a Arquitetura Dirigida por Modelos (MDA) (KLEPPE; WARMER e BAST, 2003) é um *framework* definido como padrão pela OMG para promover o desenvolvimento de software dirigido por modelos. Para isso, a abordagem MDA fornece um conjunto de ferramentas para: (i) especificar um sistema de maneira independe de plataforma; (ii) especificar as plataformas em si; (iii) definir uma plataforma particular alvo para o sistema; e, por fim, (iv) transformar a especificação do sistema na plataforma específica (MILLER, 2003). Com base no exposto, o MDA tem como objetivo promover interoperabilidade, reuso e portabilidade (OMG, 2003).

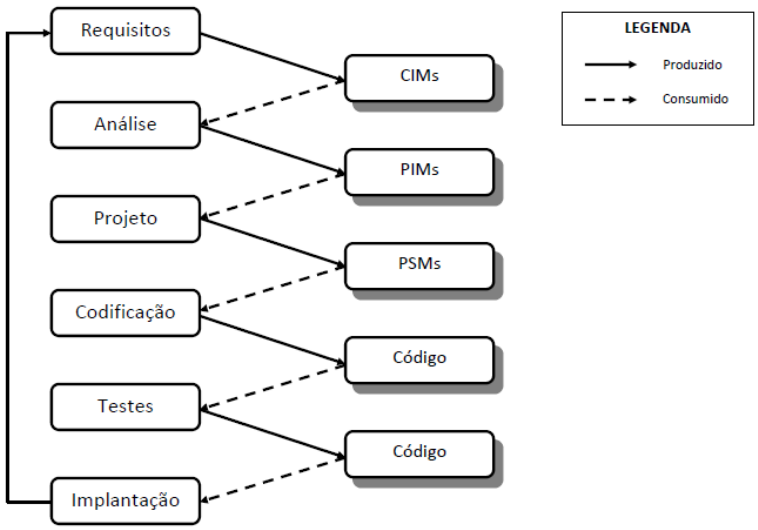


Figura 3: Processo de Desenvolvimento de software de acordo com o MDA, adaptado de Kleppe, Warmer e Bast (2003)

O processo de desenvolvimento por meio da abordagem MDA é semelhante aos ciclos tradicionais, diferindo principalmente nos artefatos resultantes em cada fase (KLEPPE; WARMER e BAST, 2003). Na Figura 3 é possível observar a relação entre as camadas arquiteturais do MDA com as fases tradicionais de desenvolvimento de software (definição dos requisitos, análise, projeto, codificação, testes e implantação), conforme é detalhado a seguir:

- **Modelo independente de computação** (do inglês, CIM – *Computation Independent Model*): esse modelo também é conhecido como modelo de domínio, pois representa apenas o vocabulário e conceitos do domínio da aplicação em questão de forma a não ficar dependente

de mudança arquitetural ou tecnológica. Assim, ele também representa o modelo de requisitos no processo numa terminologia que é familiar para profissionais do domínio do sistema

- **Modelo independente de plataforma** (do inglês, PIM – *Platform Independent Model*): esse modelo representa uma visão mais concreta do sistema, podendo trazer informações como arquitetura e esquema de persistência, por exemplo, contudo, ela não traz informações relacionadas à plataforma.
- **Modelo específico de plataforma** (do inglês, PSM – *Platform Specific Model*): esse modelo define uma visão baseada nas estruturas de implementação relacionadas em uma tecnologia específica. A idéia é tratar melhor a separação dos requisitos e do projeto das tecnologias de implementação numa arquitetura de software, fato este que permitirá alterar qualquer um desses três elementos mencionados de forma separada.

As transformações ou mapeamentos entre modelos são mecanismos primordiais no MDA e consistem no processo de traduzir um modelo em outro. Normalmente essa transformação ocorre no sentido de maior abstração para o de menor abstração, correspondendo ao processo de geração de código-fonte. Contudo, as transformações também podem ocorrer no sentido inverso em um processo de engenharia reversa, ou ainda, podem ocorrer de um modelo para ele mesmo, representando o seu refinamento. Apesar de a abordagem MDA orientar que as transformações ocorram por meio de ferramentas automatizadas, ela não define como as transformações devem ocorrer. Além disso, as transformações podem ser classificadas de três formas diferentes de acordo com os artefatos gerados (CZARNECKI e HELSEN, 2003):

- **M2M (do inglês, *Model-to-Model*):** o artefato-alvo gerado é outro modelo;
- **M2C (do inglês, *Model-to-Code*):** o artefato-alvo gerado é um código-fonte;
- **M2T (do inglês, *Modelo-to-Text*):** o artefato-alvo gerado é, além do código-fonte, outros artefatos textuais como casos de teste e diversas documentações.

O MDA considera diferentes tipos de transformações de modelo para modelo, a saber: CIM-CIM, CIM-PIM, PIM-PIM, PIM-PSM e PSM-PSM. Além disso, ele considera a transformação de modelos PSM no código-fonte e outros tipos de artefatos textuais (PSM-texto). Em teoria, um aplicativo desenvolvido sob a abordagem MDA é independente de plataforma, que permite que ele seja instalado em diferentes plataformas de computação, suportando tecnologias diferentes graças a essas transformações.

O MOF (do inglês, *Meta-Object Facility*) é o padrão OMG alicerce do MDA, consistindo em um meta-metamodelo para as definições das linguagens de modelagem, permitindo padronização para as ferramentas e transformações. Ele consiste em um padrão orientado a objetos com as seguintes características: (i) permite a definição de classes com atributos e relacionamentos de forma semelhante ao diagrama de classes UML; (ii) define uma interface padrão de acesso aos dados dos modelos; e (iii)

define regras para criação de interfaces específicas para cada metamodelo. Para uma melhor compreensão do papel do MOF, a Figura 4 apresenta as camadas de modelagem e meta-modelagem do MDA composta por quatro níveis: a primeira camada (M0) é constituída de artefatos relacionados a objetos e registros e dados; a segunda (M1) representa instâncias de modelos gerados a partir das definições dos metamodelos da camada M2, ou ainda, podem ser considerados metadados (dados que descrevem os dados); a terceira (M2) representa o metamodelo utilizado para definir os modelos, sendo UML um exemplo; e a quarta (M3) e última camada é composta pelas definições de meta-metamodelos, sendo o MOF, que define a UML, o exemplo deste nível. Como o meta-metamodelo é instancia de si mesmo, não existe quinto nível.

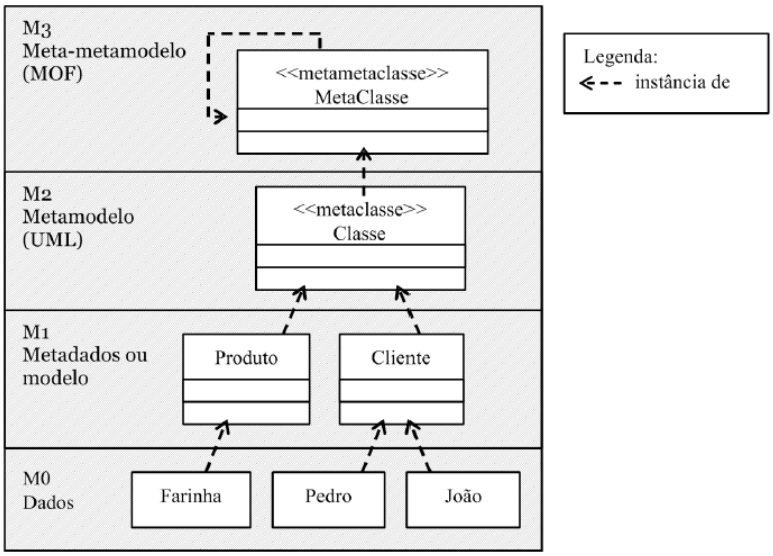


Figura 4: Arquitetura Clássica de Metamodelagem (LUCREDIO, 2009)

3.4 Modelagem Específica de Domínio

A Modelagem Específica de Domínio (do inglês, DSM – *Domain-Specific Modeling*) é uma abordagem que tem como ideia principal solucionar problemas com escopo mais fechado levando em consideração que este fator acarreta uma diminuição da complexidade na modelagem, favorecendo a melhoria no grau de automação das soluções. Dessa forma, o DSM se baseia em criar modelos que utilizam conceitos de um domínio particular utilizando Linguagens Específicas de Domínio (do inglês, DSL – *Domain-Specific Language*), sendo esse tipo de solução normalmente aplicado para um produto ou aplicação particular, ambiente ou plataforma específica.

Diferentemente das linguagens de propósito geral que são projetadas para se apropriar de diferentes tipos de aplicações e domínios, DSLs são linguagens com um conjunto pequeno de elementos, costumeiramente declarativas e, conforme já mencionado, criadas para uma classe específica de problema (DEURSEN; KLINT e VISSER, 2000; FOWLER, 2016). As DSLs podem ser

textuais, gráficas ou ainda uma combinação dessas duas formas, sendo compostas normalmente por três elementos (LUCREDIO, 2009):

- **a sintaxe abstrata,** que define os conceitos do domínio e as relações e restrições que se aplicam a eles, correspondendo ao metamodelo que define a estrutura dos modelos que podem ser criados pelas linguagens de modelagem;
- **a sintaxe concreta,** que provê um sistema para representar os conceitos do domínio de forma concreta;
- **a semântica,** que define o significado dos elementos da sintaxe abstrata, variando conforme o objetivo em questão. No contexto do MDD, a semântica é definida por meio de ações a serem executadas por um interpretador automático.

Segundo Czarnecki (2004), um dos grandes diferenciais em adotar DSLs está no suporte de ferramentas específicas do domínio. Uma DSL pode melhorar o ambiente de desenvolvimento através da inclusão de editores, depuradores, dentre outras ferramentas, além disso, o conhecimento capturado por uma DSL pode ser utilizado para disponibilizar um suporte mais inteligente por parte das ferramentas aos desenvolvedores. Em virtudes de tais facilidades, o desenvolvimento de sistemas de software utilizando DSLs vem aumentado. Isso pode ser explicado pelo fato das DSLs elevarem o nível de abstração, trazendo facilidades para geração de modelos ou código demonstrando potencial para aumentar a produtividade do desenvolvimento de sistemas ou aplicações em diferentes domínios.

3.5 Engenharia de Linha de Produto

Apesar das vantagens, o MDD não trata das questões relacionadas à reutilização do conhecimento contido nos modelos de um domínio específico. Ainda assim, é perceptível que as abordagens MDD visam reduzir esforços repetitivos, sendo este um objetivo comum a área de reuso de software. Este ponto que aproxima as duas áreas se destaca em metodologias e técnicas sistemáticas que identificam pontos comuns e variáveis de um domínio (LUCRÉDIO, 2009). Nesse sentido, no final da década de 90 surgiram propostas baseadas em Linhas de Produto de Software – LPS (do inglês, *Software Product Line*) (WEISS e LAI, 1999; CLEMENTS e NORTHROP, 2001), que consistem na representação de uma família de sistemas de software que compartilham um conjunto de funcionalidades comuns, satisfazendo as necessidades de um determinado domínio ou segmento de mercado. Elas são desenvolvidas de forma sistemática a partir de um conjunto de artefatos que formam a sua arquitetura. De maneira similar, Czarnecki e Eisenecker (2000) afirmam que uma LPS especifica um conjunto de produtos de software que contém características genéricas similares, permitindo assim a definição de uma infraestrutura comum de estruturação de artefatos. Com base nas definições apresentadas, a Engenharia de Linha de Produto (do inglês, PLE – *Product Line*

Engineering) investiga como administrar de forma eficaz um conjunto de produtos advindos de uma LPS (VOELTER e VISSER, 2011).

O desenvolvimento de uma LPS pode ser definido em duas fases, a Engenharia de Domínio e a Engenharia de Aplicação (POHL; BÖCKLE e LINDEN, 2005). A primeira delas corresponde a um conjunto de atividades para coletar, organizar e armazenar a experiência do desenvolvimento de software de um determinado domínio (como *Web*, *IoT*, *Games*, etc) na forma de artefatos reutilizáveis de tal maneira que facilite a sua aplicação em outros sistemas. Resumidamente, essa fase envolve a análise de domínio da aplicação, a delimitação do escopo da LPS e o desenvolvimento dos elementos, que definem uma arquitetura comum para a LPS (CZARNECKI e EISENECKER, 2000). A Engenharia de Aplicação, por sua vez, consiste na geração de produtos por meio da composição e integração dos artefatos definidos na Engenharia de Domínio, produzindo assim uma instância da LPS que também é definida como um produto da família de aplicação (CZARNECKI e EISENECKER, 2000). Essa atividade de geração de uma aplicação pertencente à família de aplicações é conhecida como Derivação de Produto, e a sua implementação consiste em um mecanismo primordial para a construção de uma LPS (DEELSTRA; SINNEMA e BOSCH, 2005). Esta atividade está relacionada com a Engenharia de Domínio no que diz respeito ao gerenciamento dos modelos generativos e das variabilidades associadas. Por outro lado, também está associada à Engenharia da Aplicação no ponto em que permite construir os produtos da LPS a partir das configurações dos modelos, artefatos e seleção das características (*features*) desejadas no produto final. Uma das principais propostas que tratam especificamente a derivação de produtos, considerando uma família de aplicações, é o Desenvolvimento Generativo de Software (CZARNECKI, 2004), sendo várias técnicas propostas por essa comunidade empregadas trabalho desenvolvido nesta tese. Antes de apresentá-la, para uma melhor compreensão, o conceito de modelo de *features* será discutido a seguir.

3.5.1 Modelo de Feature

Voelter e Visser (2011) definem um Modelo de *Features* (do inglês, FM – *Feature Model*) como sendo uma representação compacta das características (*features*) dos possíveis produtos de uma LPS, que podem ser propriedades, funcionalidades ou até mesmo restrições relevantes do sistema. O FM reúne informações sobre as *features* comuns e variáveis de uma LPS, modelando todos os possíveis produtos em um dado contexto, representando assim uma família de aplicações, e não uma única aplicação. A abordagem FODA (do inglês, *Feature-Oriented Domain Analysis*) proposta por Kang *et al.* (1990) é uma das estratégias mais consolidadas e adotadas para o gerenciamento de *features*. Ela prevê a criação de uma modelo de *feature* geralmente organizado em uma estrutura de árvore, podendo as *features* ser classificadas como: (i) *obrigatórias* — quando representam artefatos comuns a todos os produtos da LPS e devem obrigatoriamente estar presente nos produtos gerados; (ii)

opcionais — quando estão associados a artefatos cuja inclusão depende da seleção da *feature* pelo engenheiro de aplicação de acordo com a sua necessidade; e (iii) *alternativas* — quando são constituídas de agrupamentos de duas ou mais *features* mutuamente exclusivas, ou seja, apenas uma delas estará no produto final.

Ainda sobre as *features*, elas também podem ser classificadas da seguinte forma (KANG *et al*., 1998): (i) *capacitação* — características aparentes aos usuário, como serviços, operações e características não-funcionais; (ii) *tecnologia do domínio* — definem maneiras de se implementar serviços ou operações; (iii) *técnicas de implementação* — técnicas genéricas utilizadas para implementar serviços, operações e funções do domínio; (iv) *ambiente de operação* — representam o ambiente de execução das aplicações. Também existem as relações estruturais e conceituais das *features* (composição, generalização e implementação), além das regras complementares de relações de dependência ou exclusão mútua entre as *features*. Por fim, alguns autores propõem extensões para aumentar o poder de expressão dos MF, por meio de estratégias como a utilização de cardinalidade, atributos, extensão das categorias, modularização, dentre outras (CZARNECKI; HELSEN e EISENECKER, 2004).

3.5.2 Abordagem de Desenvolvimento Generativo

Czarnecki e Eisenecker (2000) definem o Desenvolvimento Generativo de Software (do inglês, GSD – *Generative Software Development*) como sendo uma abordagem para famílias de sistemas ou aplicações com foco na geração automática dos membros destas famílias a partir de uma ou mais especificações textual ou gráfica que utilize FM ou DSLs. Trata-se de uma abordagem complementar ao MDD que tem como objetivo central propiciar melhorias na qualidade do processo de derivação. Dessa forma, visa diminuir a distância semântica entre o espaço do problema e o espaço da solução por meio de modelos de alto nível que abstraem as complexidades da plataforma de implementação dos desenvolvedores de software. A principal característica do GSD é a grande utilização de tecnologias e ferramentas para automatização do desenvolvimento de software como, por exemplo, a meta-programação, a reflexão e a transformação de modelos. Além disso, também busca facilitar a seleção, composição e configuração dos artefatos e suas respectivas variabilidades.

A Figura 5 apresenta uma visão geral da abordagem GSD proposta por Czarnecki e Eisenecker (2000). No lado esquerdo estão detalhadas as fases relacionadas à etapa de Engenharia de Domínio já mencionada anteriormente, compreendendo a produção dos artefatos para implementar a LPS, sendo focada no desenvolvimento para reuso. Essa etapa possui três fases: análise, projeto e implementação. A (i) *análise de domínio* recebe como entrada artefatos que contemplam o conhecimento de domínio que apóiam a atividade de levantamento de requisitos do domínio normalmente especificada por meio de um FM; (ii) *projeto de domínio* recebe como entrada o FM

advindo da fase anterior e com base nele projeta a arquitetura de domínio ou arquitetura da LPS ou ainda a arquitetura da família de aplicação; (iii) *implementação de domínio*, com base na arquitetura definida são especificadas DSLs (*wizards*, diagramas etc.) e geradores de código-fonte que auxiliam uma derivação manual ou automática dos artefatos reutilizáveis.

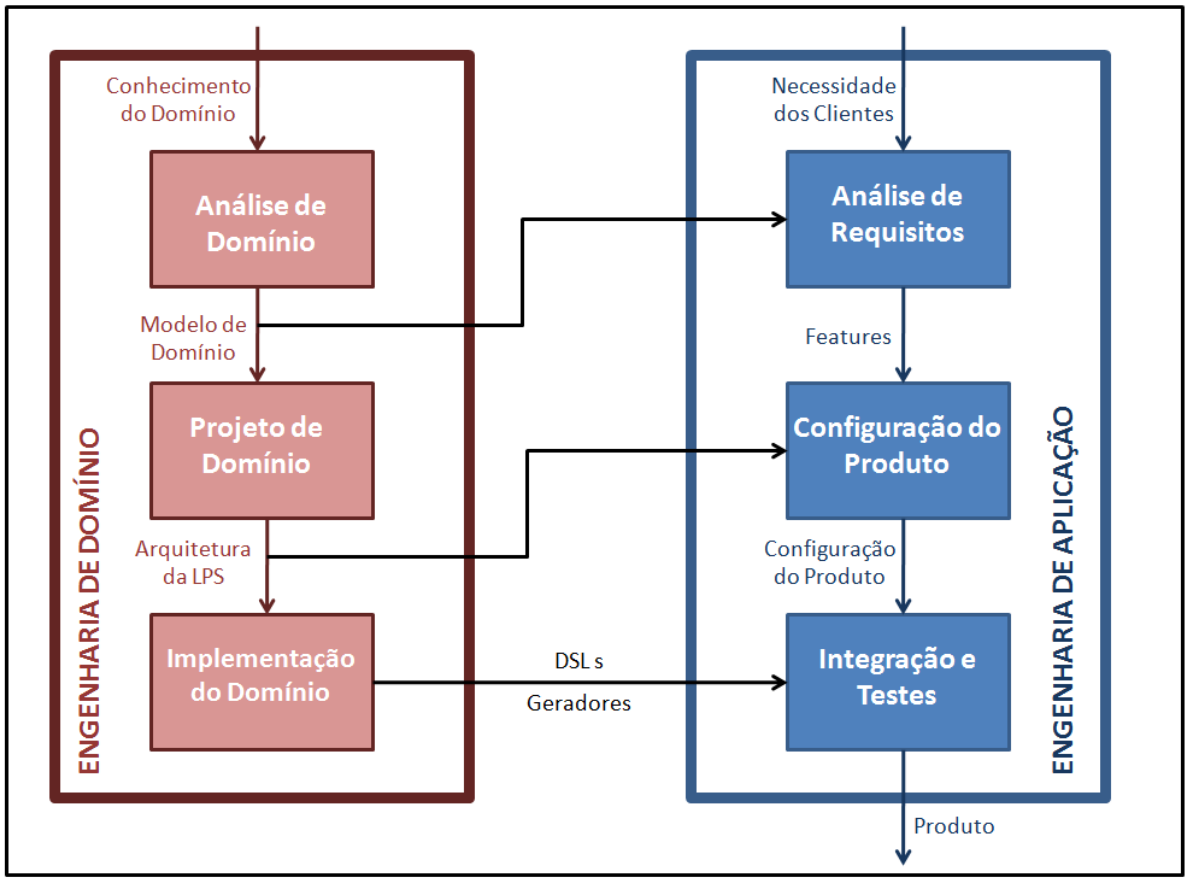


Figura 5: Visão Geral da Abordagem de Desenvolvimento Generativo (CZARNECKI e EISENECKER, 2000)

Dando continuidade, no lado direito da Figura 5 estão detalhadas as fases relacionadas à etapa de Engenharia de Aplicação, que tem como objetivo desenvolver produtos de software a partir de um conjunto de artefatos definidos na Engenharia de Domínio, tratando-se de uma fase de desenvolvimento com reuso. Essa etapa também possui três fases: análise de requisitos, configuração do produto e integração e testes. A (i) *análise de requisitos* recebe como entrada as necessidades dos clientes por meio das quais são definidos os requisitos da aplicação e, por conseguinte, as *features* utilizadas para geração do produto; a (ii) *configuração do produto* recebe as *features* definidas na fase anterior que servem para isolar os artefatos de sofware previamente definidos que as implementam; e, por fim, na (iii) *integração e testes*, uma especificação de *features*, geralmente formada por instâncias do FM e em alguns casos modelos de DSLs, serve como entrada dos geradores automatizados para derivação do produto final desejado. Vale salientar que o processo não é necessariamente sequencial e,

dessa forma, caso novos requisitos não previstos pela LPS surjam, será necessário voltar à etapa de Engenharia de Domínio para customização (CZARNECKI, 2004).

3.6 Eclipse Modeling Framework

O EMF (do inglês, *Eclipse Modeling Framework*) é uma abordagem baseada na plataforma Eclipse[16] que permite a manipulação de modelos de acordo com o meta-metamodelo *Ecore* ao invés do MOF. O *Ecore* é uma implementação do MOF que evoluiu com o passar dos anos, tornando-se uma alternativa mais simples e eficiente (STEINBERG *et al.*, 2008).

O fluxo de trabalho EMF é bastante pragmático e se dá da seguinte forma: um modelo é criado e definido no formato *Ecore*, que consiste basicamente em um subconjunto de diagramas de classe UML. Em seguida, a partir de um modelo *Ecore* é possível gerar código Java (KOEGEL e HELMING, 2017). Mais detalhadamente, o *framework* é dividido em duas partes: o próprio EMF e o EMF.Edit. O primeiro é composto pelo meta-metamodelo *Ecore*, que é responsável pelo suporte na produção da especificação e representação dos metamodelos, e pelo gerador EMF, que se trata de uma ferramenta de geração que cria uma implementação em Java do modelo previamente especificado. O segundo é utilizado como um conjunto de classes para criar representações gráficas do modelo, como editores gráficos, servindo para manipulá-lo. O gerador EMF utiliza a API do EMF.Edit como base principal para a geração dos editores gráficos dos modelos do usuário.

No EMF, a geração de código é realizada pela tecnologia para emissão de *templates* (do inglês, JET – *Java Emitter Template*) (POPMA, 2004). O JET consiste em um mecanismo de geração de código baseado em *templates*. Por meio da adição de código Java dentro dos *templates* o JET permite a realização de metaprogramação, ou seja, a construção de programas que constroem programas. O JET pode ser combinado a arquivos XML ou modelos EMF, fato este que permite que ele seja utilizado como gerador de código para uma DSL.

3.7 Considerações sobre o Capítulo

Neste Capítulo foram apresentados os principais conceitos e técnicas do desenvolvimento orientado a modelos. Uma discussão acerca das suas principais vantagens, como o aumento na produtividade em relação ao tempo de desenvolvimento de software, melhoria na comunicação entre as equipes envolvidas no desenvolvimento de software, dentre outras também foi apresentada. Acerca deste ponto, a importância da existência de um processo de desenvolvimento bem definido se tornou evidente, sendo necessário para suportar as boas práticas MDD comprovadas de forma sistemática.

---

[16] Eclipse: http://www.eclipse.org/

Caso contrário, os benefícios associados ao uso do MDD podem, além de perder o seu potencial, aumentar o impacto das desvantagens, como complexidade e alto investimento inicial.

Na literatura, é possível encontrar diversas contribuições nas áreas de abordagens para MDD. Porém, o uso do MDD para auxiliar o desenvolvimento de aplicações no domínio mulsemedia permanece inexplorado. Neste trabalho, procura-se combinar as abordagens propostas pela OMG-MDA, Modelagem Específica de Domínio e LPS/GSD num contexto específico de desenvolvimento de aplicações para mulsemedia. Tal domínio será detalhado no próximo Capítulo.

4 APLICAÇÕES MULSEMEDIA

Este Capítulo apresenta o domínio de aplicações mulsemedia, tendo como base os trabalhos realizados por Ghinea *et al.* (2014) e Sulema (2016). A organização deste Capítulo se dá da seguinte forma: a Seção 4.1 apresenta conceitos relacionados à mulsemedia e as principais áreas em que ela vem sendo explorada; a Seção 4.2 detalha o padrão MPEG-V que trata a interoperabilidade entre o mundo real e o mundo virtual; a Seção 4.3 relata as soluções de hardware enquanto a Seção 4.4 relata as soluções de software existentes para renderização e execução de efeitos sensoriais; e, por fim, a Seção 4.5 discute as considerações finais do Capítulo.

4.1 Conceitos Fundamentais sobre Mulsemedia

Atualmente, as aplicações multimídia consistem em uma combinação de áudio, vídeo, e, por algumas vezes, texto, explorando assim apenas dois dos sentidos humanos (visão e audição). Essas aplicações não consideram outras sensações como calor, umidade, aromas e sabores, que poderiam ser suportadas se esses conteúdos multimídia também engajassem os demais sentidos (GHINEA *et al.*, 2014). Todavia, 60% da comunicação humana se dá de forma não verbal e a grande parte da nossa percepção sobre o mundo ocorrer por meio de uma combinação dos nossos cinco sentidos (visão, audição, tato, paladar e olfato) (GHINEA *et al.*, 2014). Diante do exposto, as aplicações mulsemedia são aquelas que envolvem três ou mais dos sentidos humanos, promovendo o enriquecimento do conteúdo multimídia tradicional com novos objetos de mídia (olfativos, hápticos, etc.) e, consequentemente, o aumento da imersão e a melhoria da Qualidade de Experiência (QoE) do usuário (GHINEA *et al.*, 2014). Neste ponto, vale distinguir os conceitos: mulsemedia e multimodalidade. Segundo Oviatt (2007), interfaces multimodais processam duas ou mais modalidades de entrada do usuário combinadas (por exemplo, fala, caneta, toque, etc.) de forma coordenada com as modalidades de saída. Uma modalidade de entrada corresponde a uma informação gerada pelo usuário capturada por dispositivos de entrada (por exemplo, fala, caneta) ou sensores (por exemplo, sensor de movimento). Uma modalidade de saída corresponde a um estímulo aos sentidos humanos (audição, cheiro, toque, gosto ou visão) usando dispositivos audiovisuais ou atuadores. Por esse ponto de vista, as aplicações mulsemedia são aquelas que lidam com modalidades de saída.

Outros dois conceitos bastante relacionados à mulsemedia são imersão e QoE. O primeiro consiste em uma abstração do mundo físico promovida por um ambiente que forneça estímulos sensoriais e emocionais capazes de envolver o usuário em uma experiência prazerosa, podendo ser promovida tanto por ambientes computacionais quanto por filmes, livros, etc. (GHINEA; ANDRES; GULLIVER, 2011; WITMER; SINGER, 1998). Já o segundo se dá pela combinação de aspectos

técnicos, psicológicos e sociais que resultam no grau de satisfação ou rejeição de um usuário com um serviço ou aplicativo (LE CALLET; MÖLLER; PERKIS, 2013; MÖLLER; RAAKE, 2014).

Ghinea (2014) afirma que a mulsemedia é uma emergente área de pesquisa. A melhoria da imersão e o aumento da QoE dos usuários promovidas pela inserção de múltiplos efeitos sensoriais em aplicações multimídia são fatores motivacionais para pesquisas em diversas áreas do conhecimento. Em consonância com tal afirmação, na área de educação, Zou *et al.* (2017) conduziram um estudo objetivando analisar o impacto da adoção da mulsemedia como metodologia de aprendizagem aprimorada por tecnologia (do inglês, *Technology Enhanced Learning* - TEL), bem como a sua influência na QoE dos alunos. Para isso, os pesquisadores implementaram um ambiente para reproduzir vídeos anotados com efeitos sensoriais (vibração, aroma e vento). O estudo foi realizado com dois grupos de alunos de duas universidades européias, totalizando 42 participantes. A execução do experimento se deu por meio de duas etapas, sendo a primeira reservada para os participantes assistirem os vídeos alternadamente com e sem efeitos sensoriais, avaliando a experiência após cada vídeo assistido. Já a segunda etapa consistia em responder a um questionário para avaliar tanto o experimento de forma geral quanto os impactos no processo de aprendizagem.

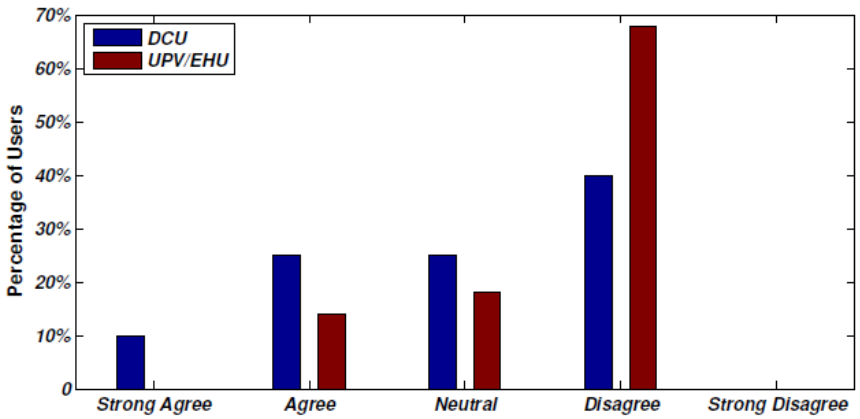


Figura 6: Resposta para a pergunta: “A experiência multissensorial não melhorou minha experiência de aprendizagem” (ZOU *et al.*,2017)

A Figura 6 apresenta o resultado para o seguinte questionamento: “A experiência multissensorial não melhorou minha experiência de aprendizagem?” É possível observar que, no caso dos alunos da universidade DCU (do inglês, *Dublin City University*), 35% concordaram e fortemente concordaram, 25% foram neutros e os 40% restantes discordaram da pergunta. Já no caso dos estudantes da universidade UPV/EHU (do inglês, *University of the Basque Country*), apenas 13,6% concordaram com a pergunta, 18,2% foram neutros e os 68,2% restantes não concordaram com a pergunta. Dessa forma, os pesquisadores concluíram que a maioria dos alunos participantes se demonstrou muito aberta para mulsemedia como TEL, bem como a sua utilização melhorou a experiência de aprendizagem.

A mulsemedia também demonstra grande potencial de aplicação na área de publicidade. A chamada publicidade sensorial refere-se à publicidade que envolve os sentidos do consumidor a ponto de afetar sua percepção, julgamento e comportamento (KRISHNA, 2012). Isso ocorre, por exemplo, quando se observa a imagem de um sanduíche. Apesar de explorar apenas um sentido, essa mídia é capaz de deixar os consumidores com a boca cheia d'água e influenciar as suas intenções de consumo. Este mesmo cenário pode ser ainda mais atrativo com a mulsemedia, tornando possível sentir a textura, o aroma e até mesmo o gosto de um produto antes de adquiri-lo por meio de um aplicativo e-commerce, por exemplo. É bem verdade que o suporte tecnológico necessário para promover a experiência previamente descrita ainda está em seus primórdios, entretanto, Petit *et al.* (2015) discorrem sobre alguns protótipos já existentes que apontam para esta direção, sendo apresentados em detalhes na Seção 4.3.

Ghinea *et al.* (2014) também destacou o potencial da mulsemedia no uso terapêutico para tratamentos de pessoas com necessidades especiais, como dificuldade de aprendizagem, autismo, mal de Alzheimer, demência, etc. Uma mostra disso é a pesquisa desenvolvida por Robles-Bykbaev *et al.* (2017) em que um ecossistema inteligente é utilizado para promover o desenvolvimento das habilidades relacionadas à comunicação social em crianças autistas. Por meio deste ecossistema, diferentes atividades são realizadas durante a terapia com o auxílio de um assistente robótico ou uma sala de estimulação multissensorial.

Como últimos exemplos, vale destacar as iniciativas relacionadas às áreas de entretenimento. Notadamente, a indústria de filmes foi pioneira na exploração dos efeitos sensoriais, sendo os filmes 4D da Disney acrescidos de estímulos táteis e olfativos apresentados ao público desde os últimos 30-40 anos (GHINEA *et al.*, 2014). Cinema, parques temáticos, TV Digital, IPTV e jogos devem ser fortemente beneficiados pela mulsemedia. Citando alguns trabalhos recentes neste contexto, Jalal e Murroni (2017) executaram um *survey* objetivando transmitir a comunidade científica qual o papel que a mulsemedia pode desempenhar no aprimoramento da QoE em futuros serviços de TV *broadcast*. Em outra pesquisa, Monks *et al.* (2017) realizaram um extenso estudo subjetivo cujos resultados apontaram para uma melhoria da QoE do usuário associada à mulsemedia baseada em vídeo 3D como sendo maior do que a experiência baseada em vídeo 2D. Por fim, Zhang *et al*. (2016) investigaram a utilização do sentido do olfato em jogos. Para tanto, os pesquisadores combinaram a utilização de músicas de fundo típicas usadas em jogos chineses com alguns odores como estímulos. Os resultados apontaram que congruência música-odor desempenhou um papel fundamental na percepção do agrado da música, sendo um resultado importante para a otimização do *design* dos jogos.

4.2 O Padrão MPEG-V

O padrão MPEG-V provê uma arquitetura e especifica representações de informação para conectar o mundo virtual (ex. jogos, simuladores e aplicações) e mundo real (ex. sensores e

atuadores) (KIM e HAN, 2014; KIM; HAN e YOON, 2012). Com foco na questão da interoperabilidade, o padrão envolve especificações sobre capacidade de dispositivos, efeitos sensoriais, características de objetos do mundo virtual e formato de dados para interação entre dispositivos. É possível perceber mais claramente a importância e necessidade do padrão observando os exemplos dos chamados jogos MMORPG (do inglês, *Massive Multiplayer Online Role-Playing Game*), como *World of Warcraft*[17]. Neste tipo de jogo é possível controlar um avatar no mundo virtual cooperando com outros avatares de outros jogadores. Com o advento de sensores modernos como o *kinect*, além de outros sensores (presença, temperatura, localização, etc.) e atuadores (ventiladores, lâmpadas de LED, cadeiras vibratórias, etc.) que podem estar espalhados pelo ambiente, torna-se possível participar de uma experiência colaborativa, imersiva e ubíqua de tal forma como se fosse parte da nossa imaginação (YOON *et al.*, 2015). Entretanto, as tecnologias relacionadas à experiência multissensorial descrita baseiam-se em produtos patenteados e não em um padrão que representa os dados dos sensores e atuadores no mundo real, além da maneira de interagir com o mundo virtual. Como resultado, cada mundo virtual proprietário fica isolado dos outros mundos virtuais. Isso dificulta a migração de um mundo virtual para outro, e, portanto, quando um mundo virtual perde o seu interesse, todos os artefatos produzidos, bem como a própria comunidade são perdidos (YOON *et al.*, 2015). Objetivando sanar as dificuldades mencionadas, a comunidade MPEG iniciou em 2008 a normatização do padrão MPEG-V (ISO/IEC 23005) que já está em sua segunda versão publicado em 2013. O padrão MPEG-V consiste das seguintes partes:

- ISO/IEC 23005-1: Arquitetura – provê uma visão geral do padrão MPEG-V por meio de sua arquitetura, casos de uso e aplicações (ISO/IEC 23005-1, 2014);
- ISO/IEC 23005-2: Informações de Controle – fornece as ferramentas para obter uma descrição das capacidades dos atuadores e sensores, as preferências do usuário em relação aos efeitos sensoriais, e as suas preferências em relação às adaptações do sensor. Esta parte especifica uma linguagem de descrição de controle da informação CIDL (do inglês, *Control Information Description Language*). Como parte da CIDL, quatro vocabulários apóiam a representação das informações: o vocabulário para especificar a capacidade do dispositivo – DCDV (do inglês, *Device Capability Description Vocabulary*); o vocabulário para especificar a capacidade dos sensores – SCDV (do inglês, *Sensor Capability Description Vocabulary*); o vocabulário para especificar preferências de efeitos sensoriais do usuário – USPV (do inglês, *User's Sensory Preference Vocabulary*); e o vocabulário para especificar a preferência do usuário para adaptação dos sensores – SAPV (do inglês, *Sensor Adaptation*

[17] https://worldofwarcraft.com/pt-br/

Preference Vocabulary), cujas sintaxes são definidas usando o esquema XML (ISO/IEC 23005-2, 2013);

- ISO/IEC 23005-3: Efeitos Sensorias – será abordada adiante, contudo, ela provê as ferramentas para descrever efeitos sensoriais em sincronia com conteúdo multimídia. (ISO/IEC 23005-3, 2013);
- ISO/IEC 23005-4: Características de Objetos do Mundo Virtual – especifica sintaxe e semântica para descrever características de avatares, ou objetos do mundo virtual, para torná-los compatíveis com outro mundo virtual, ou ainda, permitir que um objeto virtual seja controlado a partir de entradas (sensores) do mundo real (ISO/IEC 23005-4, 2013);
- ISO/IEC 23005-5: Formato de Dados para Dispositivos de Interação – tem como objetivo especificar o formato de dados ou interfaces para dispositivos de interação que podem ser sensores ou atuadores. Define a linguagem baseada em XML, IIDL (do inglês, *Interaction Interface Description Language*), para descrição de interface para interação e os vocabulários DCV (do inglês *Device Command Vocabulary*) e SIV (do inglês, *Sensed Information Vocabulary*) para apoiar a descrição de mensagens para os comandos e obter informações a partir dos comandos respectivamente (ISO/IEC 23005-5, 2013);
- ISO/IEC 23005-6: Ferramentas e Tipos Comuns – especificam 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. (ISO/IEC 23005-6, 2013);
- ISO/IEC 23005-7: Conformidade e Software de Referência – provêem software de referência e especifica um esquema de validação baseado no *Schematron*[18] é fornecido para testes de conformidade (ISO/IEC 23005-7, 2014).

A pesquisa desenvolvida nesta tese tem como o escopo as aplicações mulsemedia e, por este motivo, nesta Seção são abordadas em mais detalhes a ISO/IEC 23005-1, relaciona a arquitetura do padrão MPEG-V, e a ISO/IEC 23005-3, relacionada aos efeitos sensoriais por estarem mais diretamente relacionada ao tema.

4.2.1 ISO/IEC 23005-1: Arquitetura MPEG-V

A Figura 7 ilustra a representação da arquitetura do padrão MPEG-V definida na parte ISO/IEC 23005-1. Nela é possível observar como se dá o relacionamento entre as partes que formam o padrão MPEG-V além dos cenários de transição de dados do mundo virtual para o mundo real (Figura

[18] *Schematron*: definido através da parte 3 da ISO/IEC 19747-3 (*Document Schema Definition Languages* – Parte 3: *Rule-based validation – Schematron*)

7a) e também do mundo real para o mundo virtual (Figura 7b). No primeiro cenário (V→R) as informações dos efeitos sensoriais (parte 3) e características de objetos virtuais (parte 4) podem ser encaminhadas para o ambiente físico do usuário a fim de serem reproduzidos em dispositivos físicos capazes de renderizar os efeitos no mundo real. Também é possível observar o motor de adaptação V→R, que serve para converter descrições de efeitos sensoriais (parte 3) em comandos para dispositivos (parte 5), podendo levar em consideração as preferências de efeitos sensoriais do usuário (parte 2) e também as capacidades dos dispositivos renderizadores (parte 2). Esse motor de adaptação está diretamente ligado a plataforma específica de implementação da solução, não estando sua definição no âmbito da norma. Contudo, vale salientar que os atuadores podem ser compatíveis diretamente com a parte 3 do MPEG-V, não sendo necessária a conversão de um efeito sensorial. Levando em consideração a definição de Ghinea (2014) para a mulsemedia, este é o cenário de transição de dados que está diretamente relacionado às aplicações mulsemedia que visão engajar os cinco sentidos humanos conforme discutido anteriormente.

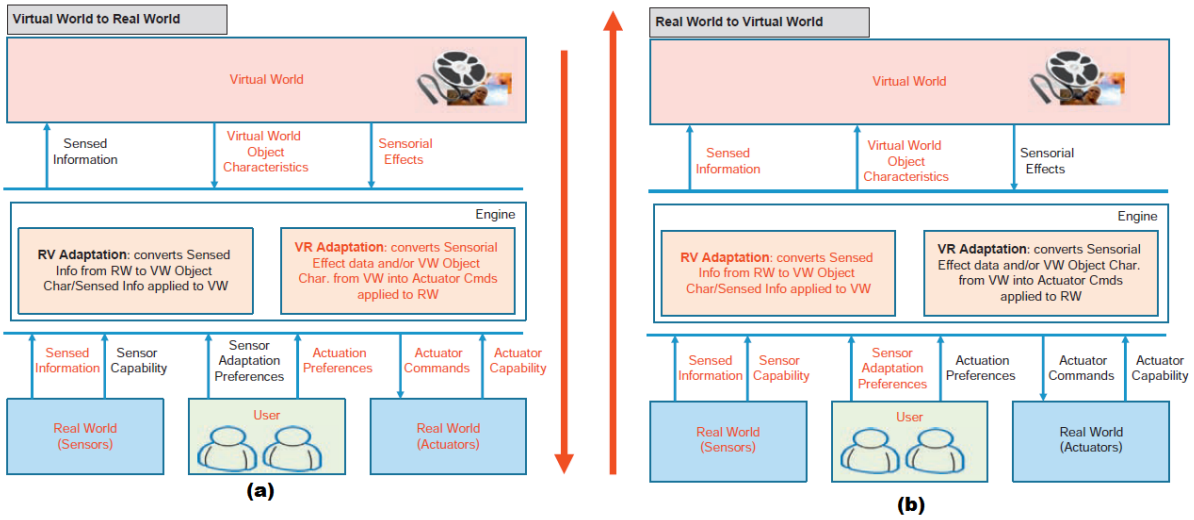


Figura 7: Arquitetura do MPEG-V (YOON; KIM e HAN; HAN e PREDA, 2015)

Já no segundo cenário (R→V), as informações obtidas do ambiente real do usuário (parte 5) e suas preferências (parte 2) são encaminhadas para o ambiente virtual, ou seja, por meio de sensores e interfaces multimodais no mundo real os dados são capturados modificando o comportamento ou configuração dos ambientes virtuais. A figura do motor de adaptação se mantém com funções similares neste novo cenário e mais uma vez não faz parte da norma. Apesar de relacionado, o cenário R→V foge do escopo tratado desta tese, estando mais voltando a pesquisas relacionadas a aplicações multimodais.

4.2.2 ISO/IEC 23005-1: Sensory Effect Description Language – SEDL

A parte 3 do MPEG-V especifica uma linguagem baseada em XML chamada SEDL (do inglês, *Sensory Effect Description Language*) para a descrição de efeitos sensoriais (CHOI e KIM, 2012). Por meio da linguagem SEDL é possível definir blocos de construção e atributos comuns a fim de gerar um arquivo contendo a descrição dos metadados relacionados aos efeitos sensoriais chamado SEM (do inglês, *Sensory Effect Metadata*). Entretanto, os efeitos sensoriais em si não fazem parte da linguagem SEDL, estando eles definidos em um vocabulário de efeitos sensoriais que é denominado como SEV (do inglês, *Sensory Effect Vocabulary*). O SEV está relacionado a todos os efeitos sensoriais, adicionando atributos específicos a determinados efeitos como, por exemplo, os atributos para definir um tio de aroma (rosas, fumaça, etc.) e uma cor (branca, azul, etc.) aos efeitos sensoriais aroma e luz, respectivamente. Dessa forma, existe maior extensibilidade e flexibilidade para que desenvolvedores definam seus próprios efeitos sensoriais (KIM e HAN, 2014). Atualmente a norma define metadados (SEV) para efeitos de luz, luz colorida, *flash* de luz, vento, vibração, vaporização, aromas, névoa, correção de cor, movimento, cinestesia e tato (CHOI e KIM, 2012). Em resumo, a linguagem SEDL juntamente com os vocabulários SEV formam metadados de efeitos sensoriais SEM que podem ser associados a qualquer tipo de conteúdo multimídia, como jogos, filmes, músicas ou até mesmo páginas *Web*.

A ISO/IEC 23005-6 define um modelo de localização (Figura 8a) e modelo temporal (Figura 8b) para os metadados de efeitos sensoriais. Abordando inicialmente o modelo de localização, a norma define um esquema *LocationCS* que pode ser melhor compreendido por meio do exemplo "urn:mpeg:mpeg-v:01-SI-LocationCS-NS:*:*:front", significando que o efeito será renderizado considerando os dispositivos atuadores que estejam em qualquer posição de x e y mas o eixo z restringe o à parte da frente (*front*) do usuário. Já o modelo temporal permite o controle mais refinado das variações de intensidade dos efeitos sensoriais ao longo do tempo. Por exemplo, observando a Figura 8b é possível notar o aumento gradual da intensidade de um determinado efeito (vibração ou vento, por exemplo) em $fade = t_1 - t_0$. De maneira semelhante, a diminuição gradual em $fade = t_3 - t_2$.

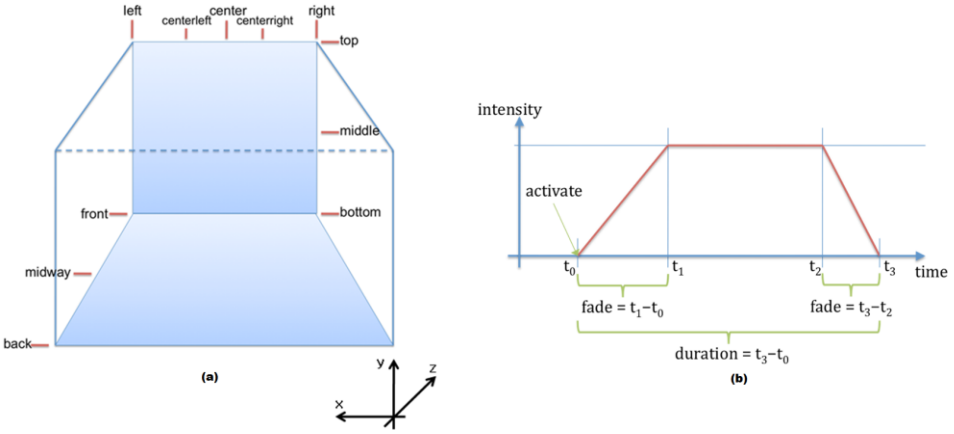


Figura 8: Modelo de localização (a) e modelo temporal (b) para metadados de efeitos sensoriais (CHOI e KIM, 2012)

Dando continuidade, Waltl, Timmerer e Hellwagner (2009) apresentaram uma descrição formal da linguagem SEDL por meio do Formalismo EBNF (do inglês, *Extended Backus–Naur Form*) objetivando abstrair e facilitar o seu entendimento. 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 principal, podendo conter opcionalmente a descrição de metadados (*DescriptionMetadata*) que fornece informações sobre o SEM. De forma semelhante, pelo menos uma das opções entre *Declarations*, *GroupOfEffects*, *Effect*, e *ReferenceEffect* deve existir no documento. Estes elementos, bem como os demais envolvidos, significam respectivamente:

- *Declarations* é utilizado para configurar aspectos comuns aos vários efeitos sensoriais e necessita de ao menos uma das opções entre *GroupOfEffects*, *Effect* e *Parameter*.
- *GroupOfEffects* é utilizado para agregar mais de um efeito sensorial na mesma marca de tempo (*timestamp*), definindo assim quando um grupo de efeitos estará disponível para a

aplicação, sendo utilizado para fins de sincronização. Deve conter pelo menos duas definições de efeitos (*EffectDefinitions*).

- *ReferenceEffect* permite referenciar um efeito criado anteriormente ou um grupo de efeitos.
- *Effect* é usado para descrever um efeito sensorial em uma determinada marca de tempo (*timestamp*).
- *EffectDefinition* possui atributos opcionais para configuração do efeito, sendo eles:
    - *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 alternativo caso o original não possa ser processado;
    - *priority* diz respeito à 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
    - *adaptability* que define tipos preferidos de adaptação do efeito correspondente.

4.3 Mulsemedia Hardware

Sulema (2016) afirma que novas mídias demandam a implementação de novas soluções tecnológicas de hardware e software. Neste sentido, o desenvolvimento de aplicações mulsemedia ainda está em seus primórdios, sendo o desenvolvimento destas soluções um dos grandes desafios para a multimídia nos próximos 10 anos (GHINEA, 2014). Notadamente, os dispositivos atuadores são essenciais no contexto das aplicações mulsemedia, pois eles são responsáveis por materializar no mundo real os efeitos sensoriais programados no mundo virtual. Assim, a fim de engajar os cinco sentidos humanos, equipamentos diversos são utilizados, como dispositivos para interações hápticas, ventiladores, lâmpadas, pulverizadores de ar e de água, difusores de aromas, dentre outros. Uma visão geral sobre os diversos dispositivos existentes é apresentada por Sulema (2016) e Ghinea (2014), dessa forma, são destacados nesta Seção alguns esforços recentes relacionados ao olfato e ao paladar, pois são apontados como os dois sentidos mais desafiadores, necessitando ainda de grande esforço em pesquisas para criação de soluções.

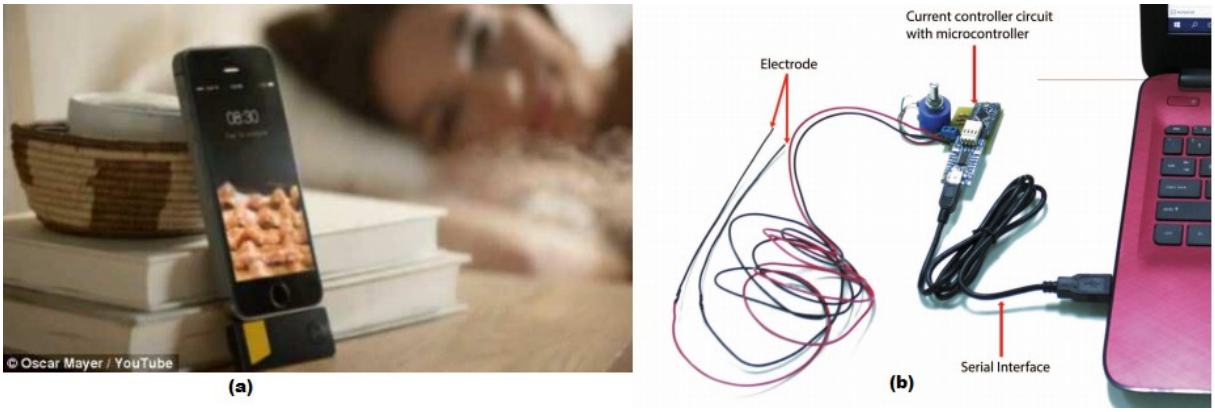


Figura 9: Soluções para estímulos olfativos – (a) *Scentee* (BRAUN e CHEOK, 2015) e (b) *Digital Smell Interface* (HARIRI *et al.*, 2016)

Acerca deste último ponto, Spence *et al*. (2017), com o foco no domínio de HCI (do inglês, *Human Computer Interaction*), resumem o estado da arte das possibilidades e armadilhas relacionadas aos estímulos desses dois sentidos. Duas abordagens vêm sendo investigadas, a primeira por meio de estímulos puramente digitais, e a segunda por meio da entrega analógica controlada digitalmente dos estímulos químicos. Certamente a primeira abordagem é mais promissora, pois dispensa a necessidade de recarregar os dispositivos, todavia, também é muito mais complexa e bastante incipiente no momento atual, sendo talvez até impossível de se atingir totalmente (SPENCE *et al*., 2017). Para ilustrar essas abordagens, a Figura 9 reúne duas soluções de dispositivos para estímulos olfativos. Na primeira, Braun e Cheok (2015) abordam a utilização do *Scentee* (Figura 9a), um dispositivo de exibição olfativa que se conecta ao soquete de áudio de um aparelho iOS ou Android. A imagem referenciada é um registro de uma bem sucedida campanha publicitária em que um aplicativo de alarme foi implementado com o *Scentee* para levar aos usuários o cheiro e ao som de bacon torrando. Já na segunda solução, Hariri *et al.* (2016) apresentam um protótipo de interface digital para gerar sensações de cheiro (do inglês, *Digital Smell Interface* – Figura 9b) por meio do estímulo dos receptores olfativos da concha nasal com pulsos elétricos fracos. Entretanto, a sensibilidade e eficácia dos estímulos ainda estão em fase de investigação.

Apesar da sua imensa importância, o sentido do paladar não vem recebendo a devida atenção, sendo a impressora 3D de alimentos[19] o único exemplo real que envolve este sentido (SULEMA, 2016). Não obstante, protótipos capazes de gerar alguns sabores, como salgado, azedo, amargo e doce vêm sendo investigados e duas dessas iniciativas são ilustradas na Figura 10. Ranasinghe e Do (2016) desenvolveram um protótipo para simular a sensação de doçura (do inglês, *Virtual Sweet* – Figura 10a) por meio de estímulos térmicos aplicados na ponta da língua humana. Para tanto, o sistema gera

[19] https://3dprinting.com/food

rápidos estímulos de aquecimento e resfriamento través de uma grade de elementos *Peltier*. Em outra pesquisa relacionada, Cheok (2016) apresenta um protótipo para criação de sabores digitalmente (do inglês, *Digital Taste Interface* – Figura 10b) por meio de corrente elétrica. Quando este aparelho é colocado por um usuário na superfície da sua língua, de maneira controlada e segura, é aplicada uma corrente elétrica a partir de uma fonte digital na membrana da língua no momento que ela faz contato físico com o dispositivo. Essa corrente percorre a língua do usuário e estimula as células relacionadas ao sabor, sinalizando para o cérebro que um gosto está sendo percebido.

Figura 10: Soluções para estímulos gustativos – (a) *Virtual Sweet* (RANASINGHE e DO, 2016) e (b) *Digital Taste Interface* (CHEOK, 2016)

Como últimos exemplos de atuadores para renderização de efeitos sensoriais, a Figura 11 reúne três soluções comerciais já mais estabelecidas e citadas em trabalhos relacionados à área. Contando com um conjunto de dispositivos capazes de renderizar efeitos de luz, vibração e vento, o *Philips amBX Gaming PC peripherals* (PHILIPS, 2017) (Figura 11a) é um aparelho criado pela empresa *Philips* utilizando a tecnologia amBX com o objetivo de proporcionar uma maior experiência sensorial principalmente em jogos, mas também podendo ser utilizado em outras aplicações multimídia. Como suporte, o equipamento dispõe de uma SDK (do inglês, *Software Development Kit*) e uma API (do inglês, *Application Programming Interface*) na linguagem C, contudo, está atualmente descontinuada. Também fazendo uso da tecnologia amBX, criado pela *Philips* e fornecido pela empresa *MadCatz*, o dispositivo de iluminação o *Cyborg Gaming Lights* (AMBX, 2017) (Figura 11b) é capaz de gerar 16 milhões de cores para realizar iluminação dinâmica. Completando os dispositivos, o *Dale Air Vortex Activ* (DELAIR, 2017) (Figura 11c) é um emissor de aromas programável desenvolvido pela empresa *DaleAir*. Cada aroma é armazenado em cartuchos substituíveis, sendo possível disparar até quatro aromas simultaneamente. Vale salientar que os três dispositivos mencionados são suportados originalmente apenas no sistema operacional Windows.

Figura 11: Atuadores para efeitos sensoriais – (a) *Philips amBX Gaming PC peripherals* (PHILIPS, 2017) e (b) *Mad Catz Cyborg Gaming Lights* (AMBX, 2017) e (c) *Dale Air Vortex Activ* (DELAIR, 2017)

4.4 Mulsemedia Software

Os softwares relacionados à mulsemedia são principalmente ferramentas de autoria e *players* de mídias sensoriais. Os primeiros, conforme já discutido no Capítulo 2 de trabalhos relacionados, permitem a criação de conteúdo audiovisual sincronizados com efeitos sensoriais de forma simples, enquanto os últimos são responsáveis por renderizar os efeitos sensoriais, ou seja, materializá-los no mundo real por meio dos atuadores, como os descritos na Seção anterior (Sulema, 2016).

O trabalho desenvolvido por Waltl *et al*. (2013) traz duas contribuições relacionadas a este tópico desenvolvidas seguindo as definições do padrão MPEG-V, o *player* de mídias sensoriais SEMP (Figura 12a) e o simulador de efeitos sensoriais SESim (Figura 12b). O SEMP (do inglês, *Sensory Effect Media Player*) foi implementado com base na API *DirectShow* da *Microsoft* para execução de aplicações multimídia para a plataforma Windows. Ele suporta os três atuadores de efeitos sensoriais previamente apresentados: *Gaming PC peripherals, Cyborg Gaming Lights* e o *Vortex Activ*. Já o SESim (do inglês, *Sensory Effect Simulator*) é uma ferramenta desenvolvida para simular os efeitos sensoriais contidos em um arquivo de descrição SEM. Atualmente o simulador é integrado a ferramenta de autoria de metadados sensoriais SEVino, sendo parte da cadeia de ferramentas proposta por Waltl *et al*. (2013). Sua interface foi inspirada no produto da Philips *amBX Gaming PC peripherals*, tendo sido desenvolvida em Java além de utilizar a biblioteca VLCJ para a decodificação e renderização de áudio e vídeo. Atualmente a ferramenta suporta sete efeitos sensoriais: luz, vento, névoa, vibração, temperatura, pulverizador d'água e aroma.

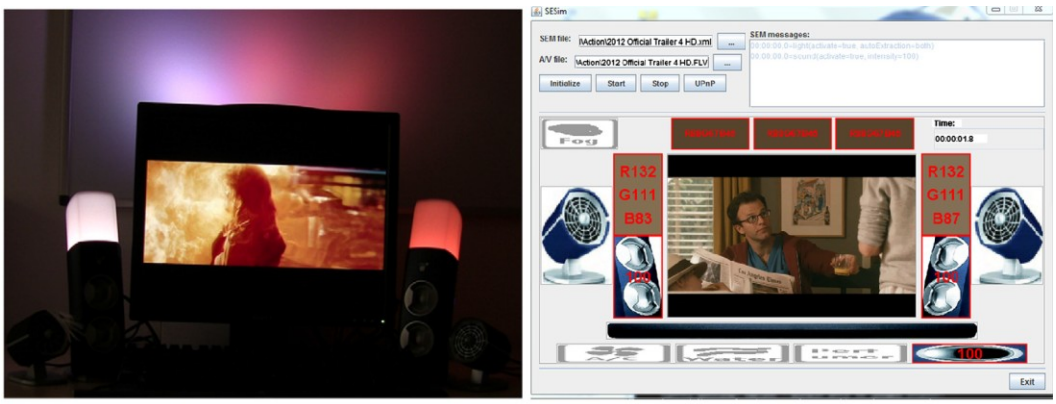


Figura 12: Ferramentas propostas por Waltl *et al*. (2013) – (a) SEMP funcionado com os atuadores amBX e (b) Interface do simulador SESim

Saleme e Santos (2015) perceberam que as soluções existentes para reprodução de efeitos sensoriais, como SEMP e o SESim, embutem o controle dos atuadores em aplicações baseadas em linha de tempo, fato que limita a reutilização em outros tipos de aplicação ou diferentes reprodutores de mídia. Dessa forma, eles desenvolveram uma solução compatível com o padrão MPEG-V chamada PlaySEM que consiste em uma abordagem para rendeziração de efeitos sensoriais de forma desacoplada aos *players* de mídias sensoriais usando comunicação UPnP.

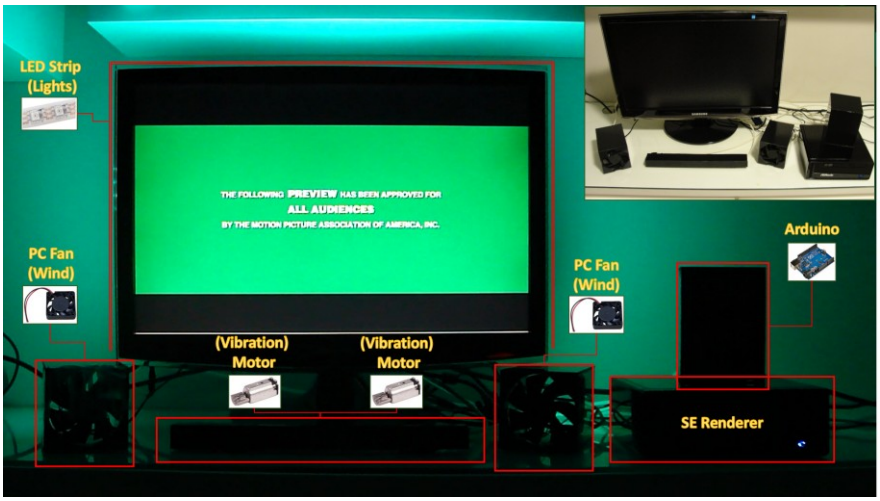


Figura 13: Ambiente de execução do SE Renderer. Fonte: (Saleme e Santos, 2015)

A plataforma PlaySEM é composta por três componentes: (i) PlaySEM SE *Video Player*, responsável pela reprodução de vídeo e entrada dos metadados SEM; (ii) PlaySEM SER (do inglês, *Sensorial Effect Renderer*), responsável pela conversão de metadados SEM em comandos para controlar os dispositivos atuadores; e (iii) PlaySEM *Device Control*, vinculado ao SER, trata-se de uma placa microcontroladora que recebe os comandos para controlar os dispositivos físicos. A Figura 13 apresenta o conjunto de hardwares (arduino, fila de LED, motor de vibração e ventilador de PC) no

ambiente real do usuário funcionando em conjunto de um mini-pc executando o SE *Renderer* e um monitor apresentando a saída de um computador executando um vídeo no SE *Video Player*. Os autores também propuseram uma abordagem baseada em eventos para o desenvolvimento de ambientes interativos utilizando o PlaySEM (SANTOS; NETO e SALEME, 2015) e avaliaram se a plataforma distribuída introduz atrasos relevantes para a reprodução dos efeitos sensoriais no ambiente do usuário (SALEME; CELESTRINE e SANTOS, 2017). Os resultados mostraram valores na faixa de 27ms a 67ms em média gastos ao longo do processo antes da ativação efetiva dos dispositivos de efeitos sensoriais em uma rede com fio.

4.4.1 Linguagens para o desenvolvimento de Aplicações Mulsemedia

As linguagens para desenvolvimento de aplicações multimídia são linguagens de programação declarativas que se concentram no arranjo espaço-temporal dos objetos de mídia (comumente vídeo, áudio, texto e animações) em uma apresentação multimídia (MEIXNER, 2017). Elas são responsáveis pela definição de documentos multimídia estruturados, que permitem aos desenvolvedores focarem sua atenção apenas nos objetos de mídia e sua sincronização, pois a estrutura do documento se mantém a mesma diante de mudanças relacionadas a software e hardware em virtude de sua organização desacoplada entre conteúdo e ambiente de execução (BULTERMAN e HARDMAN, 2005). Naturalmente, essas linguagens passam a ser mecanismos centrais para desenvolvimento de aplicações mulsemedia com recursos interativos, residindo nelas às respostas sobre questões importantes como, por exemplo, o sincronismo entre os já habituais objetos de mídia e os novos objetos de mídias sensoriais, além da possibilidade de integrar efeitos sensoriais com lógica de programação imperativa. Notadamente, os principais exemplos dessas linguagens são o HTML5, (do inglês, *HyperText Markup Language*) (W3C, 2014), o SMIL (do inglês, *Synchronized Multimedia Integration Language*) (W3C, 2012), e o NCL (*Nested Context Language*) (ITU, 2014) (GUEDES; ALBUQUERQUE e BARBOSA, 2017; MEIXNER, 2017). Entretanto, nenhuma dessas linguagens foi pensada originalmente para suportar efeitos sensoriais, necessitando de extensões para cumprir esse fim.

Conforme mencionado, o suporte ao sincronismo entre mídias é um ponto chave nessas linguagens. Para uma melhor compreensão sobre essa questão, Cazenave, Quint e Roisin (2011) propõem uma categorização para as aplicações multimída interativas em relação à sincronização que também é adequada ao novo cenário das agora aplicações mulsemedia interativas, dividindo-as em (i) aplicações orientadas a mídia e (ii) aplicações orientadas a evento. As primeiras utilizam uma mídia principal de natureza contínua (áudio e vídeo) que serve como *backbone* por meio do qual os efeitos sensoriais e demais mídias são sincronizados na linha do tempo (*timeline*). Por outro lado, as últimas se utilizam de relações temporais entre os objetos de mídia, como *links*, ou ainda interações diretas com os usuários, como um clique de botão que dispara um efeito sensorial, por exemplo. Das três

linguagens mencionadas, NCL e SMIL, desde suas concepções, foram pensadas como linguagens multimídias para sincronização de mídias. Por outro lado, o HTML5 amadureceu mais recentemente no suporte multimídia, tendo suas versões anteriores sido utilizados por bastante tempo apenas para na organização de conteúdo e da estrutura de sites *Web*.

Dessa forma, para a obtenção de um sincronismo mais refinado a ponto de alcançar tanto aplicações orientadas a mídia quanto aplicações orientadas a evento, NCL e SMIL seriam caminhos mais curtos. Apesar disso, levando em consideração outras questões além do sincronismo, o autor deste trabalhou optou no início da pesquisa por utilizar a linguagem HTML5 como plataforma específica, sendo estes os motivos: (i) HTML5 é o padrão recomendado pelo W3C para o desenvolvimento de aplicações *Web*; (ii) ele permite alcançar uma vasta gama de usuários por meio de diversos dispositivos, como *smarphones*, *tablets*, *desktops* e *notebooks*; (iii) os dispositivos *mobile* suportados possuem uma grande quantidade de sensores e atuadores bastante adequados e necessários ao desenvolvimento de aplicações mulsemedia; (iv) por meio da combinação do HTML5 a linguagem imperativa JavaScript é possível implementar o suporte aos efeitos sensoriais, bem como tratar parte do sincronismo entre objetos de mídias e efeitos sensoriais. Além dessas vantagens listadas sobre a utilização do HTML5, também vale ressaltar como um ponto desencorajador na utilização do SMIL o fato do grupo de trabalho que dava suporte a linguagem ter sido fechado em 2011. Além disso, a linguagem não é suportada adequadamente por boa parte dos *browsers* existentes.

Sobre o NCL, apesar de no início desta pesquisa a linguagem não estar preparada para o sincronismo entre efeitos sensoriais e demais objetos de mídia, em trabalho recente, Guedes, Albuquerque e Barbosa (2017) definiram um *framework* para o desenvolvimento de aplicações multimodais lidando com esta questão. Para tanto, os autores realizam uma extensão na linguagem NCL a fim de instanciar os quatro conceitos do framework, que de forma simplificada são: (i) *Recognizer* que descreve uma modalidade de entrada; (ii) *Synthesizer* que descreve como gerar uma modalidade de saída; (iii) *User context* que define informações relacionadas a contexto de usuário ou ambiente; (iv) *Modality selection* que consiste em estruturas de controle condicional como *if-then-else* e *switch*; e, por fim, (iv) *Multimodal transition* que define o comportamento das aplicações através da definição de condições e ações. A linguagem NCL é padrão para TV Digital e IPTV e, com a extensão implementada, este domínio passar a ter uma tecnologia mais madura no suporte ao desenvolvimento de aplicações mulsemedia.

4.5 Considerações sobre o Capítulo

Este Capítulo teve como foco apresentar o domínio das aplicações mulsemedia, sendo apresentadas as principais motivações e áreas de aplicação apontadas na literatura. Além disso, também foi mostrada uma visão geral sobre o padrão MPEG-V que permite a interoperabilidade entre o mundo real e virtual. Mais precisamente, uma atenção especial foi dada à parte 3 relacionada à

linguagem de descrição de efeitos sensoriais SEDL, que define uma representação sistemática de conteúdo relacionado aos efeitos sensoriais. Adicionalmente, foram apresentadas soluções de hardware e software existentes, além de uma discussão acerca de como as linguagens para o desenvolvimento de aplicações multimídia podem ser utilizadas no desenvolvimento de aplicações mulsemedia.

Diante do exposto neste Capítulo e também levando em consideração as discussões do Capítulo 2 sobre o estado da arte no desenvolvimento de aplicações mulsemedia, atualmente é possível identificar uma lacuna na definição de processos, métodos e ferramentas que auxiliem o desenvolvimento sistemático de aplicações mulsemedia em conformidade com o padrão MPEG-V. Mais precisamente, os dois pontos levantados foram:

- (i) estabelecimento de ambientes de desenvolvimento que ofereçam suporte aos pontos discutidos na Seção 1.1 de desafios atuais: sincronismo entre os objetos de mídia (áudio, vídeo, imagem, texto e efeitos sensoriais) que compõem uma aplicação mulsemedia; integração entre projetos de Mídias, Software e Efeitos Sensoriais; facilidades na integração de efeitos sensoriais com lógica imperativa; e por fim, abstração das complexidades específicas de domínio de aplicação;
- (ii) definição de abordagens de desenvolvimento que integrem de modo sistemático as diferentes disciplinas envolvidas no desenvolvimento de aplicações mulsemedia contemplando a soluções para as lacunas citados em (i).

Assim, o próximo Capítulo propõe uma abordagem MDD que procura sistematizar as atividades para melhor atender esses dois desafios a fim de melhorar o desenvolvimento de aplicações no domínio de aplicações mulsemedia.

5 ABORDAGEM MDD PARA AS APLICAÇÕES MULSEMEDIA

Este Capítulo apresenta a abordagem orientada a modelos para a integração de projetos de mídia, software e efeitos sensoriais no domínio das aplicações mulsemedia. Para a elaboração desta abordagem foram adotadas as práticas e tarefas já bem estabelecidas definidas pelas comunidades MDD, LPS e/ou GSD, e os pontos fortes dos trabalhos relacionados a esta Tese.

Este Capítulo segue a seguinte organização: a Seção 5.1 apresentada a visão geral da abordagem, separando as atividades e artefatos gerados de acordo com as fases de Engenharia de Domínio e Engenharia de Aplicação do GSD propostas por Czarnecki e Eisenecker (2000). As seções seguintes detalham as etapas da abordagem proposta, sendo as três primeiras etapas: Análise de Domínio (Seção 5.2), Projeto de Domínio (Seção 5.3) e Implementação de Domínio (Seção 5.4), pertencentes à fase de Engenharia de Domínio e a quarta e última etapa, Derivação de Produto (Seção 5.5), pertencente à fase de Engenharia de Aplicação. Por fim, a Seção 5.6 concretiza a abordagem por meio de um exemplo de uso tratando a família de aplicações de vídeos com múltiplos efeitos sensoriais: *Multisensory Video*.

5.1 Visão Geral da Abordagem Generativa

O Desenvolvimento Generativo de Software objetiva especificar, modelar e implementar famílias de sistemas de modo que um dado sistema possa ser automaticamente gerado a partir de uma especificação de *features* normalmente expressa através de alguma DSL. A Figura 14 apresenta a visão geral da abordagem GSD para a integração de projetos de mídia, software e efeitos sensoriais no domínio das aplicações Mulsemedia. Como pode ser observada, a abordagem foi dividida em quatro etapas: (i) Análise de Domínio, (ii) Projeto de Domínio, (iii) Implementação de Domínio e (iv) Derivação de Produto, sendo as três primeiras pertencentes à Engenharia de Domínio e a última pertencente à Engenharia de Aplicação. Nas seções seguintes, cada uma das etapas mencionadas é especificada, detalhando quais são as atividades realizadas, quais atores são responsáveis por uma determinada atividade, quais são os artefatos de entrada necessários para que uma atividade seja realizada e, por fim, quais são os artefatos gerados como saída de cada atividade da abordagem generativa.

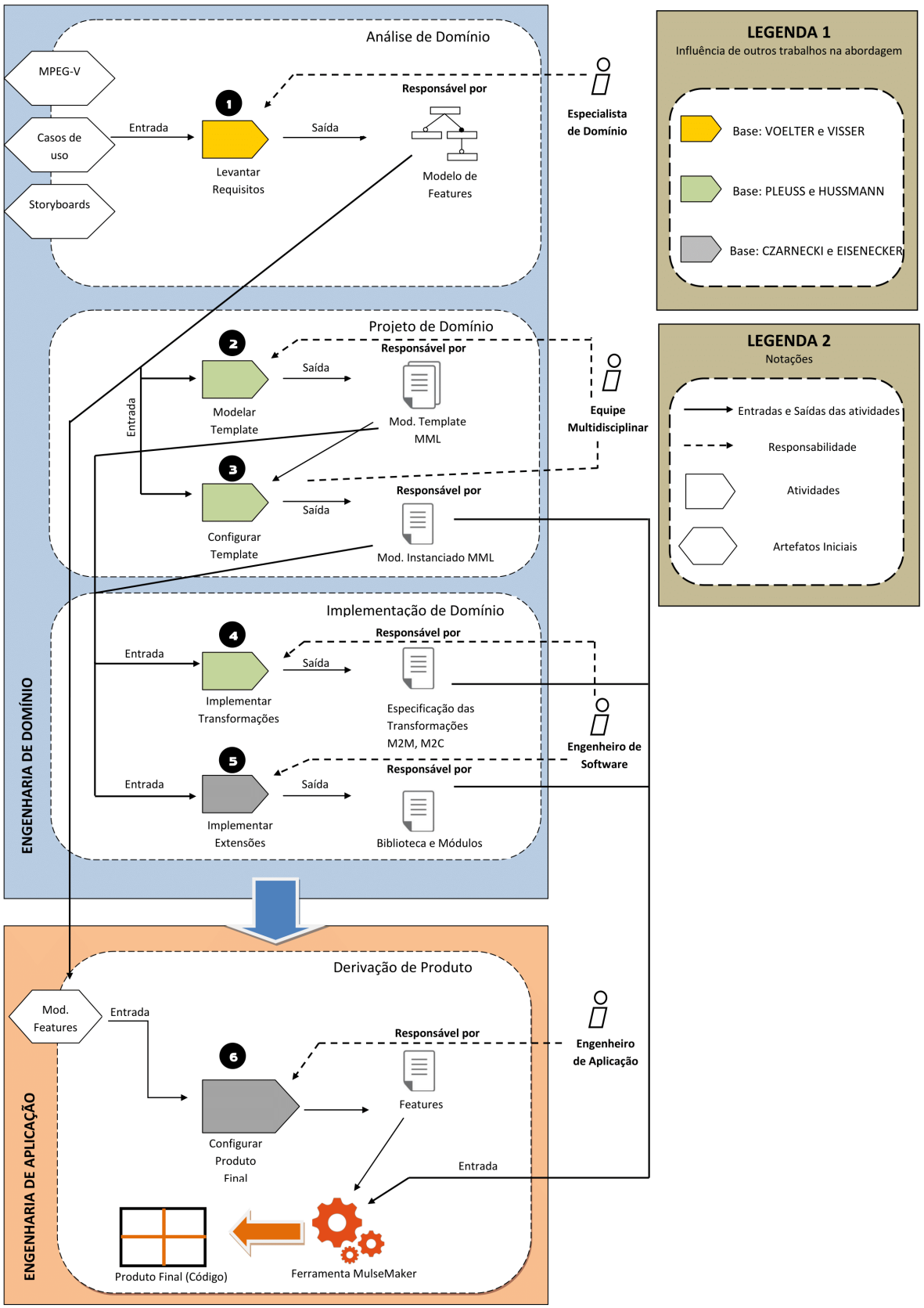


Figura 14: Abordagem Generativa para o Desenvolvimento de Aplicações Mulsemedia

Para uma melhor compreensão da abordagem, as figuras em forma de hexágono representam os artefatos iniciais de entrada das fases de Engenharia de Domínio e de Aplicação. As setas tracejadas indicam as responsabilidades de cada ator sobre as atividades, sendo estas últimas representadas por figuras em forma de pentágonos. Em relação às setas contínuas, elas podem representar os artefatos de entrada quanto tem por destino uma atividade, caso contrário, elas representam artefatos de saída. Ainda sobre a Figura 14, uma legenda relaciona a influência dos principais trabalhos correlatos na abordagem proposta com as atividades definidas. O presente trabalho é bastante influenciado pela abordagem MDD proposta por Pleuss e Hussmann (2011), que tem por objetivo integrar os conceitos das áreas de aplicações interativas e aplicações multimídia por meio da linguagem MML (do inglês, *Mutimedia Modeling Language*), uma DSL para modelagem de aplicações multimídia. A abordagem proposta nesta tese adota uma extensão da linguagem MML para modelar o *Template* que representa uma família de aplicação Mulsemedia. Já o trabalho de Voelter e Visser (2011) defende que DSLs e FMs (do inglês, *Feature Model*) utilizados em conjunto ampliam as possibilidades de derivação de produtos de uma LPS. A utilização de DSLs favorece o reuso de software, pois amplia o leque de funcionalidades eventualmente não suportadas pelos FMs. Dessa forma, a fim de facilitar o projeto do domínio por meio da definição e configuração dos *templates*, a primeira etapa da abordagem proposta provê uma atividade para estruturar os requisitos de uma família de aplicação mulsemedia por meio de um FM. Além desses trabalhos, a abordagem ainda é influenciada pelo trabalho de Czarnecki e Eisenecker (2000), que define o conceito de Programação Generativa. Trata-se de uma abordagem de desenvolvimento que tem como objetivo modelar famílias de sistemas de software, de modo que, por meio da utilização de artefatos reutilizáveis, produtos customizados possam ser construídos sob demanda. Vale salientar que no contexto deste trabalho a abordagem desenvolvida lida exclusivamente com família de aplicações e utiliza técnicas da área de LPS. Dessa forma, procurou-se combinar e estender as técnicas já estabelecidas de uma forma ainda inexplorada em uma metodologia nova para o domínio de aplicações mulsemedia.

Por fim, também é utilizada a nomenclatura (CIM, PIM e PSM) das camadas arquiteturais do MDA (OMG, 2003) para facilitar o entendimento do uso de modelos em cada etapa da abordagem. Outro ponto importante é que as atividades realizadas em cada etapa são apresentadas sequencialmente, contudo, na prática, essas atividades podem ser realizadas de modo iterativo e incremental.

5.1.1 Análise de Domínio

Pertencendo à fase de Engenharia de Domínio, a Análise de Domínio é a primeira etapa da abordagem generativa proposta, possuindo apenas a atividade *Levantar Requisitos (Atividade 1)*. Para tanto, a atividade mencionada recebe como entrada artefatos (*storyboards*, casos de uso e definições

do padrão MPEG-V) que especificam os requisitos da família de aplicação multimídia sensorial, ou seja, estes artefatos devem auxiliar na definição do escopo correspondente a um conjunto de elementos de uma categoria de aplicação mulsemedia. Em relação aos atores envolvidos, vale ressaltar mais uma vez a natureza multidisciplinar das aplicações mulsemedia, sendo elas investigadas e aplicadas em áreas como entretenimento, educação, comércio eletrônico, dentre outras, fato este que justifica o envolvimento de uma equipe multidisciplinar formada por especialistas da área de Software, da área de Mídia e também especialistas de Domínio. Por exemplo, caso uma família de aplicação mulsemedia seja desenvolvida para a área de Educação, professores de disciplinas específicas (especialista de domínio) estarão envolvidos, discutindo e esboçando *storyboards* a fim de encontrar a melhor forma de explorar as mídias sensoriais no processo de aprendizagem juntamente com os especialistas de mídia. Estes últimos serão os profissionais que lidarão com efeitos sensoriais e conhecerão as definições do padrão MPEG-V, bem como as demais mídias tradicionais da aplicação. Por fim, também participando das discussões, analistas de requisitos definem os casos de uso que servirão para delimitar o escopo da família de aplicação mulsemedia para educação.

Finalmente, diante do que foi discutido e com base nos artefatos de entrada mencionados, é realizada a atividade *Levantar Requisítos*. Nela, um ator especialista de domínio ou engenheiro de requisitos utiliza uma técnica de análise de domínio chamada de Modelagem de Características (do inglês, *Feature Modeling* - FM) (KANG *et al.*, 1990) para representar as características de todas as aplicações pertencentes a uma família de aplicação. Por meio do FM é possível identificar as funcionalidades e propriedades que são comuns a todas as aplicações da família. Dessa forma, é possível estruturar os requisitos para facilitar as atividades seguintes relacionadas ao Projeto de Software da categoria da aplicação. Vale salientar que o MDA define os artefatos produzidos nesta primeira etapa como CIM, uma vez que são independentes de representação computacional.

Por fim, um ponto importante a ser ressaltado sobre o presente trabalho diz respeito ao fato de não ter sido encontrado na literatura uma taxonomia sobre o domínio de aplicações mulsemedia. A existência de uma taxonomia seria importante para auxiliar na elaboração dos possíveis cenários de uso das categorias de aplicações. Acredita-se que isso se dá em virtude das pesquisas em mulsemedia ainda estarem em seus primórdios e, dessa forma, os requisitos e desafios por trás da especificação dessas aplicações ainda estão em fase de investigação, carecendo de novos artefatos, ferramentas de suporte e estudos acerca da dinâmica entre as equipes multidisciplinares envolvidas nesta atividade. Contudo, vale salientar que estas discussões não fazem parte das investigações deste trabalho, sendo esta etapa apresentada com o objetivo de contextualizar as etapas seguintes, que são o foco principal desta tese.

5.1.2 Projeto de Domínio

O Projeto de Domínio é a segunda etapa da Engenharia de Domínio e é realizada por meio de duas atividades: *Modelar Template* (Atividade 2) e *Configurar Template* (Atividade 3). O objetivo desta etapa é definir uma arquitetura de referência para uma determinada família (ou categoria) de aplicação mulsemedia. A Figura 15 apresenta de forma detalhada os artefatos de entrada e saída em cada uma das duas atividades mencionadas, bem como detalha como se dá o envolvimento da equipe multidisciplinar na criação dos modelos.

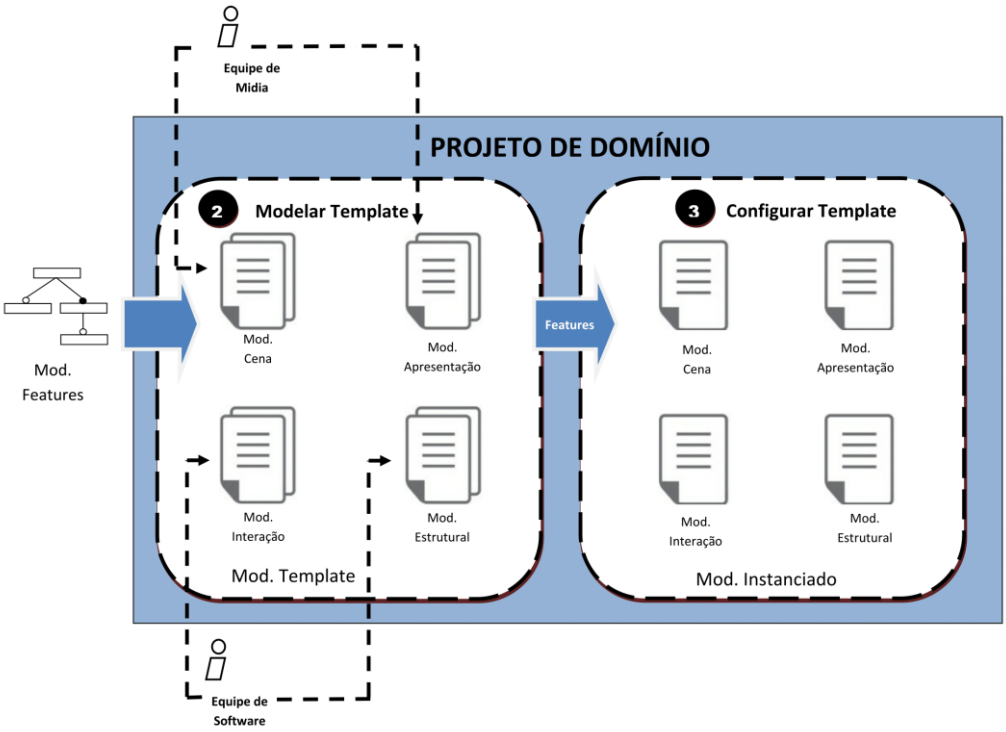


Figura 15: Detalhamento da etapa de Projeto de Domínio da abordagem proposta

Como pode ser observado, a Atividade 2, *Modelar Template*, recebe como artefato de entrada o FM gerado na Análise de Domínio e, com base nele, define o Modelo *Template* por meio da Linguagem MML, uma DSL para a modelagem de aplicações Multimídia. Na abordagem desenvolvida, foram adotados quatro tipos de visões de projeto da MML e, além disso, foi realizada uma extensão para suportar os novos elementos de mídias sensoriais. Os modelos adotados são definidos a seguir, enquanto o detalhamento da extensão realizada é apresentado no detalhamento da abordagem (Seção 5.2):

- **Modelo de Cena:** são representados através de diagramas de estados UML, em que cada estado representa uma cena, e suas transições representam as mudanças entre as cenas. Além disso, cada cena possui um modelo de apresentação e um modelo de interação correspondente;

- **Modelo Estrutural:** define a estrutura estática da aplicação. Ele descreve as classes do domínio e as mídias da aplicação. As classes são especificadas através de diagramas de classes usuais do UML e as mídias são representadas através de associações com componentes de mídias específicos da linguagem MML;
- **Modelo de Apresentação:** especifica a interface gráfica das aplicações multimídias. Este modelo contém elementos que representam abstrações de componentes de interface gráfica como um botão ou campo de texto, indivisíveis e independentes de plataforma, bem como informações sobre as instâncias de mídias contidas no modelo; e
- **Modelo de Interação:** modela a interação do usuário e o comportamento esperado para cena que o modelo está associado. O referido modelo especifica como os eventos da interface gráfica disparam as operações da lógica da aplicação, além do comportamento pré-definido entre as instâncias de mídias e a cena. Atualmente não lida com múltiplos usuários, sendo está uma possível atividade em trabalhos futuros.

Os modelos de cena e de apresentação são de responsabilidade da equipe de mídia, já o modelo estrutural e o modelo de interação são de responsabilidade da equipe de software. Dessa forma, a linguagem MML promove uma melhor estruturação da Família da Aplicação; e ao mesmo tempo, favorece a colaboração de profissionais da equipe de mídia e da equipe de software. Assim, o Modelo *Template* é a especificação de uma estrutura geral da solução, a interface e os relacionamentos entre os diferentes aspectos que a compõe, sendo capaz de derivar diversos modelos diferentes decorrentes das configurações realizadas no FM.

Em seguida, a Atividade 3, *Configurar Template*, recebe como entrada o conjunto de modelos MML que representa o Modelo *Template* e, de acordo com as *features* e transformações do tipo M2M (do inglês, *Model-to-Model*) definidas na etapa de implementação de domínio, gera o conjunto de modelos que representa uma versão de uma aplicação, chamado de *Modelo Instanciado*. Para expressar as variabilidades da aplicação – ou seja, as mais diversas características que podem estar presentes ou ausentes em uma aplicação pertencente a uma determinada família de aplicações – os elementos do *Modelo Template* são anotados por meio de estereótipos UML que servirão para expressar: (i) condição de presença (CP) e (ii) multiplicidade de elementos (ME). Estas anotações são definidas em termos das características e podem ser avaliadas de acordo com uma determinada configuração que é escolhida no momento da derivação da aplicação. Por exemplo, uma CP permite definir se um elemento do *Modelo Template* estará presente ou ausente no *Modelo Instanciado* e uma ME pode indicar quantos elementos de um determinado tipo do *Modelo Template* estarão presentes no *Modelo Instanciado*.

Vale salientar que para o MDA, os modelos gerados nesta etapa (Modelo *Template* e Modelo Instanciado) são conhecidos como PIM, ou seja, eles são independentes de plataforma, o que permite a sua transformação em modelos e/ou código-fonte para plataformas específicas e diferentes.

5.1.3 Implementação de Domínio

Seguindo o fluxo da abordagem, a próxima etapa é a de Implementação de Domínio, sendo esta a última etapa da fase de Engenharia de Domínio. Esta etapa tem como objetivo desenvolver soluções para as plataformas específicas por meio das transformações que definem o mapeamento entre uma Família de Aplicação modelada com a linguagem MML e o produto final implementado em termos dos artefatos de código reusáveis definidos para a Família de Aplicações. Para tanto, uma equipe de Engenheiros de Software é responsável por realizar duas atividades: *Implementar Transformações* (Atividade 4) e *Implementar Extensões* (Atividade 5).

A atividade 4, *Implementar Transformações*, recebe como entrada os modelos MML (Modelo *Template* e Modelo Instanciado) para implementar estratégias de derivação automática ou manual (transformadores) que mapeiam os modelos MML em modelos específicos de plataforma e, em sequência, no código-final do produto (aplicação mulsemedia). Assim, para cada Modelo Instanciado, composto por quatro tipos de modelo MML: estrutural, cena, apresentação e interação, devem ser criados mecanismos de transformação que procurem por elementos específicos de cada tipo de diagrama UML. Por exemplo, o Modelo de Interação MML é composto por diagramas de atividades e, dessa forma, os parâmetros, fluxos e ações serão mapeados. Já no caso do Modelo de Cenas MML, os diagramas de estado e as transições que o compõe serão mapeados. Para o diagrama estrutural são consultadas as propriedades. Assim, cada elemento presente nos diagramas UML será convertido em elemento da plataforma da aplicação mulsemedia especificada. Os artefatos resultantes desta atividade são as especificações de transformações M2M e M2C (do inglês, *Model-to-Code*) necessárias para geração de código na plataforma específica definida como alvo.

A Atividade 5, *Implementar Extensões*, é a última atividade da Engenharia de Domínio. Nesta atividade são desenvolvidos artefatos de software reutilizáveis necessários para a geração do código-final na Plataforma Específica definida como, por exemplo, componentes, módulos e bibliotecas.

Os artefatos gerados nesta etapa da Engenharia de Domínio são utilizados na Atividade 6, Configura*r Produto*, da Engenharia de Aplicação. Para tanto, são adotadas ferramentas de autoria, como *Wizards*, que utilizam as transformações e artefatos de software reutilizáveis, abstraindo as complexidades da Engenharia de Domínio para os atores responsáveis pela atividade da Engenharia de Aplicação. Por fim, vale ressaltar que o MDA define as transformações e artefatos gerados nesta etapa *como modelos PSM (do inglês,* Platform Specific Model*).*

5.1.4 Derivação de Produtos

A Derivação de Produtos é a etapa final da abordagem generativa proposta e está inserida no contexto da Engenharia de Aplicação. Como mencionado anteriormente, a Engenharia de Aplicação tem como objetivo elaborar o produto de software a partir de um conjunto de requisitos e combinação dos artefatos definidos na Engenharia de Domínio. Nesta etapa foi definida a Atividade 6, Configurar Produto Final, que pode ser executada por profissionais não-especialistas, com pouca ou nenhuma experiência em programação. Nesse caso, esses atores são conhecidos como engenheiros de aplicação. Esta atividade recebe como artefato de entrada uma especificação de requisitos a fim de identificar se existe algum *Modelo de Features* definido pela engenharia de domínio que pode ser utilizado. Logo em seguida, depois da escolha anterior (*features),* o engenheiro de aplicação realiza uma configuração manual a fim de definir as características que ele deseja para a aplicação a ser gerada. Normalmente são ajustes detalhados e informações de instância (valores de propriedades) para o código-fonte do Produto Final. Essa configuração pode ser apoiada por uma ferramenta de autoria, como um *Wizard*. O resultado desta etapa é a geração do Produto Final, correspondendo ao código-fonte de uma aplicação mulsemedia (um jogo, aplicação de TV, página *Web* ou um vídeo multissensoriais, por exemplo) .

Vale salientar que para chegar a este código final ocorrem os seguintes mapeamentos (desenvolvidos na etapa de Implementação de Domínio da Engenharia de Aplicação) entre modelos: (i) um mapeamento de *Modelo Template* para *Modelo Instanciado*, uma (ii) transformação de *Modelo Instanciado* para modelo *Modelo Específico de Plataforma* e, em seguida, a geração do código final da aplicação mulsemedia.

5.2 Detalhamento da Abordagem

Esta Seção apresenta o detalhamento da abordagem generativa proposta por meio da criação da família de aplicação de vídeos multissensoriais inspirada no cenário apresentado por Yoon (2013): *Multisensory Video*. Trata-se de uma aplicação simples, contudo, suficiente para apresentar os passos da abordagem de forma didática. Além disso, ela também serviu como prova de conceito no primeiro experimento controlado realizado que será apresentado no Capítulo 6.

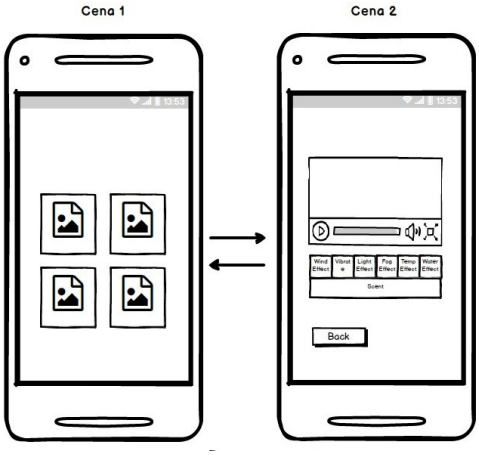


Figura 15: *Storyboard Multisensory Video*

A Figura 16 apresenta uma das aplicações da família *Multisensory Video* composta por 2 (duas) cenas. Como ainda não existem ferramentas de prototipação que levem em consideração os efeitos sensoriais, eles serão descritos textualmente sempre que ocorrerem a seguir:

- **Cena 1:** corresponde a tela inicial com as opções de vídeos disponíveis; e
- **Cena 2:** corresponde à reprodução do conteúdo do vídeo de forma síncrona com efeitos sensoriais, cujos instantes de início e fim são controlados a partir de uma timeline;

Outra informação importante diz respeito ao ambiente de execução para a renderização dos efeitos sensoriais. Conforme previamente mencionado, este trabalho optou por se utilizar de um cenário mais bem estabelecido em termos de tecnologias, fechando o escopo das aplicações mulsemedia no domínio *Web*/*Mobile*. Assim foi possível utilizar *browsers* como *players* para as mídias sensoriais e alguns atuadores existentes em celulares (como *vibracall* e *flash* de luz), sendo os efeitos mais complexos simulados. A principal motivação para esta estratégia foi permitir uma avaliação final através do uso da abordagem em cenários reais e que pudesse ser realizada sem muitas dificuldades. Contudo, vale ressaltar que a abordagem MDD desenvolvida prevê a reutilização dos modelos CIM e PIM para outras plataformas específicas, dependendo apenas da realização das atividades 4 e 5 da Engenharia de Domínio para outras plataformas específicas de execução e até mesmo para tecnologias que surgirão futuramente.

5.2.1 Tecnologias empregadas

As principais tecnologias empregadas na concretização da abordagem foram o *framework* EMF (STEINBERG *et al*., 2008) para a criação do Modelo *Template*, o *plugin* FMP (ANTKIEWICZ e CZARNECKI, 2004) para criação do modelo de *Features* e a linguagem MOFScript (OLDEVIK, 2011) para a transformação M2C a fim de gerar o código-final da aplicação mulsemedia.

O *framework* EMF, já introduzindo em detalhes no Capítulo 3, é um conjunto de *plugins* Eclipse que pode ser usado para especificar metamodelos do usuário compatíveis com o meta-metamodelo Ecore e um conjunto de transformadores que possibilitam implementar práticas MDD. Já o FMP (do inglês, *Feature Modeling Plugin*) é um *plugin* para o Eclipse que utiliza como base o *framework* EMF para tratar a criação e configuração de características em termos de modelos. Nele, a criação das características é especificada usando a modelagem baseada na cardinalidade. Por último, o MOFScript é uma linguagem voltada para transformações de modelo para código também baseada no EMF e no Ecore. Ela permite uma independência de metamodelo, podendo utilizar tanto um modelo definido pelo metamodelo da UML quanto um modelo baseado num metamodelo próprio do usuário.

5.2.2 Atividade 01 – Levantar Requisitos

Esta atividade produz como artefato o Modelo de *Features* denominado *MultisensoryVideo Feature Model*, representando as características de todas as aplicações da família de vídeos multissensoriais. Como pode ser observado na Figura 17, o FM é organizado no modelo `VideoEffectsInstantiation`, que basicamente modela as definições da parte-3 do MPEG-V em que os metadados de efeitos sensoriais (SEM) são descritos em conformidade com a linguagem de descrição de efeitos sensoriais (SEDL) e seus vocabulários (SEV). Com exceção da *feature* `VideoQuantity` que define quantos vídeos sensoriais haverá na aplicação final, todas as demais *features* modeladas são opcionais, possuindo cardinalidade [0..*], e representam efeitos sensoriais que podem ser associados a um vídeo. Os seguintes efeitos são modelados: vento, nevoeiro, vibração, temperatura, pulverizador d'água, aroma e luz. Também é possível verificar na Figura 17 (a) os parâmetros do efeito de vento definidos de acordo com o MPEG-V: `duration`, `fade`, `alt`, `priority`, `location`, `autoExtraction`, `intensityValue`, `intensityRange`, e `timestamp`. Esses parâmetros são comuns a todos os efeitos sensoriais. No entanto, existem alguns parâmetros específicos como, por exemplo, a cor (`color`) em um efeito de luz.

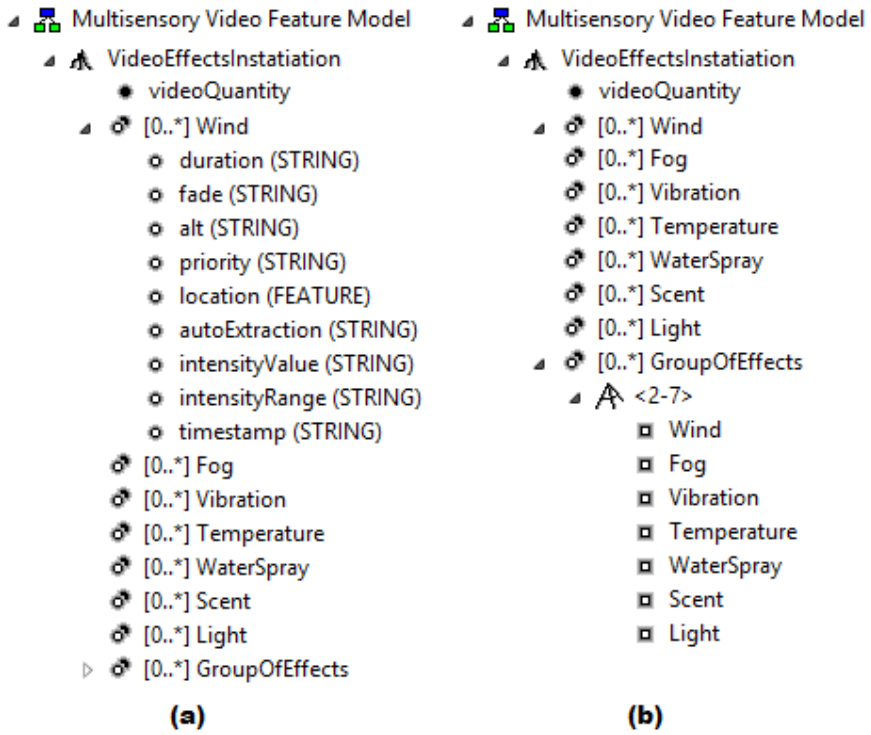


Figura 16: Modelo de *Features* da Família de *Multisensory Video*

Ainda sobre o modelo `VideoEffecsInstantiation`, ele fornece o recurso opcional `GroupOfEffects` para associar mais de um efeito sensorial ao mesmo `timestamp` (Figura 17b). Mais precisamente, a cardinalidade `<2-7>` significa que estarão associados ao mesmo `timestamp` no mínimo 2 (dois) e no máximo 7 (sete) efeitos sensoriais. Uma explosão é um bom exemplo para ilustrar o uso de grupo de efeitos sensoriais, pois em decorrência dela ocorrem efeitos de vibração (grandes explosões geram tremores ao seu redor), aroma (a combustão e fumaça possuem aroma característico), aumento da temperatura (explosões liberam grande quantidade de energia) e corrente de ar (a força de uma explosão pode deslocar o ar ao seu redor).

5.2.3 Atividade 02 – Modelar Template

O Modelo *Template* de *Multisensory Video* especifica a união de todas as possíveis soluções da família de aplicações de vídeos com múltiplos efeitos sensoriais. Para tanto, ele é descrito por meio dos modelos relacionados às visões estruturais e comportamentais da linguagem MML. Conforme já discutido, MML é uma linguagem de modelagem visual baseada em UML 2.0 que suporta o processo de *design* no desenvolvimento de aplicativos multimídia. O objetivo dos modelos MML é o desenvolvimento de aplicativos. Neste sentido, é útil distinguir os tipos de mídia que precisam de suporte de desenvolvimento diferente. Então, o metamodelo MML define diferentes tipos de mídia como base para geração do código. Elas são classificadas em mídias contínuas (video, áudio,

animações 2D e 3D) e mídias discretas (imagem, gráfico e texto). Dito isto, a versão corrente da linguagem MML não suporta tipos de mídia relacionados aos efeitos sensoriais que estimulam os sentidos além da visão e audição. As tecnologias relacionadas à mulsemedia ainda eram muito incipientes na época de sua criação, e de acordo com Pleuss (2009), não é útil incluir tipos de mídia não estabelecidos em abordagens de modelagem. No entanto, o cenário hoje é diferente, já existindo um padrão que fornece uma arquitetura e especifica as representações de informação para conectar o mundo virtual e o mundo real, o MPEG-V (vide Capítulo 4). O padrão é centrado na questão da interoperabilidade, envolvendo especificações sobre recursos do dispositivo, efeitos sensoriais, características de objetos virtuais e formato de dados para a interação do dispositivo.

O metamodelo MML foi estendido com base na parte 3 do MPEG-V, que especifica a linguagem SEDL para descrição de efeitos sensoriais e os vocabulários de efeito sensorial (SEV). Na Figura 18 é possível observar os diagramas na cor rosa que representam os efeitos sensoriais sob a nova mídia Mulsemedia. Nesta primeira extensão foram adicionados ao metamodelo MML os seguintes efeitos sensoriais: `WaterSprayer`, `Temperature`, `Light`, `Vibration`, `Wind`, `Scent` e `Fog` para representar respectivamente pulverizador d'água, temperatura, luz, vibração, vento, aroma e névoa. Além disso, também é possível realizar a composição de no mínimo dois desses efeitos em grupos de efeitos sensoriais.

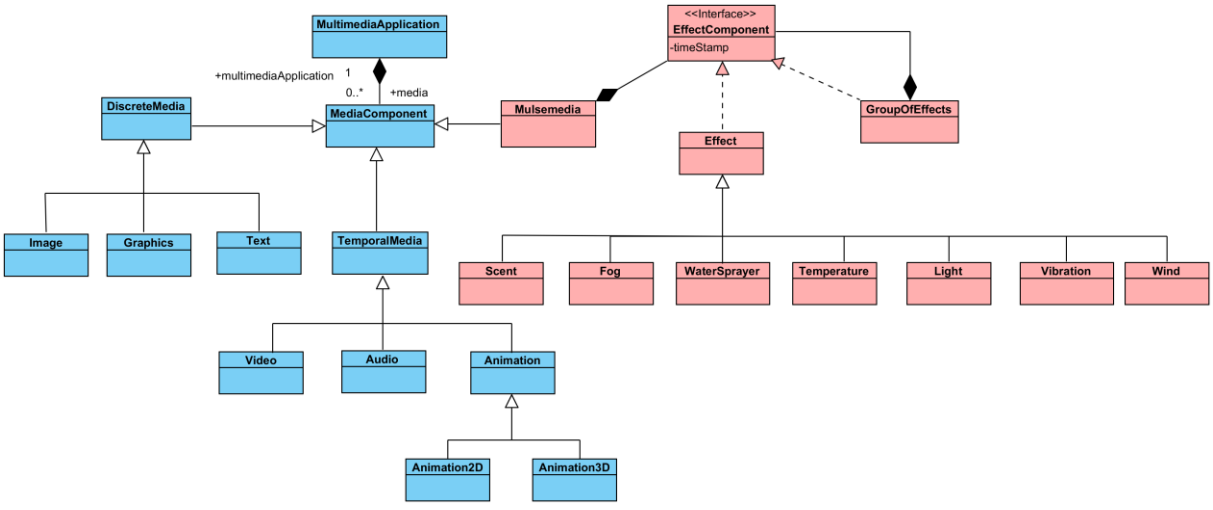


Figura 17: Metamodelo MML Estendido com novas mídias de efeitos sensoriais – Adaptado de (PLEUSS, 2009)

Dando continuidade, para definir o modelo *Template*, além do novo perfil MML com estereótipos específicos do domínio de aplicações mulsemedia, também é utilizado um perfil de Variabilidades com estereótipos para representar variabilidades de condição de presença (CP) e multiplicidade de elementos (ME) nos modelos UML. Esses estereótipos do perfil de variabilidades

são associados às características do modelo de Features. Dessa forma, o Modelo *Template* de *Multisensory Video* especifica a união de todas as possíveis soluções da família de aplicações de vídeos com múltiplos efeitos sensoriais, sendo composto pelos modelos estrutural, de cena, de apresentação e de interação, anotados com os respectivos estereótipos.

Seguindo as orientações da linguagem específica de domínio MML, a primeira modelagem a ser realizada é a estrutural. Nela, são evidenciados os elementos do domínio, bem como relacionamento entre eles e as respectivas mídias discretas ou contínuas. Na Figura 19, é ilustrado o modelo estrutural de *Multisensory Video*. É possível observar que existem as classes `Video` e `VideoEffects`. A primeira delas representa o objeto de mídia de vídeo e, portanto, é anotado com o estereótipo `<<video>>`, conforme define a linguagem MML. Além disso, também possui o estereótipo `<<videoQuantity>>` relacionado à variabilidade de multiplicidade de elementos. Este estereótipo está relacionado à característica (*feature*) `videoQuantity` do modelo de *Features* e, por meio dela, o engenheiro de aplicação definirá quantos vídeos multissensoriais existirão no produto final gerado. Já a segunda classe (`VideoEffects`) está relacionada às mídias sensoriais e, portanto, é anotada pelo estereótipo `<<EffectComponent>>`, que foi criado no metamodelo MML para representar essas novas mídias até então não previstas. Também é importante destacar o relacionamento de agregação entre as classes `Video` e `VideoEffects`, que significa que um ou mais objetos de efeitos sensoriais compostos, que agrupam um conjunto de instâncias, podem ser associados ao `Video`.

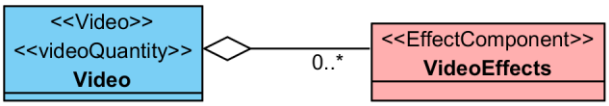


Figura 18: Modelo Estrutural *Multisensory Video*

O próximo modelo MML definido é o de Cena (Figura 20). O seu objetivo é especificar o comportamento geral da aplicação através de uma máquina de estados com um conjunto de estados (cenas) e de transições entre eles. A família *Multisensory Video* possui duas cenas na sua versão atual, sendo a primeira cena (`ChooseVideo`) correspondendo à tela inicial com as opções de vídeos multissensoriais disponíveis; e a segunda cena (`WatchSensoryVideo`) correspondendo à reprodução do vídeo multissensorial escolhido, bem como os efeitos sensoriais executados nos atuadores em sincronia com o vídeo de acordo com o *timestamp* de cada efeito sensorial.

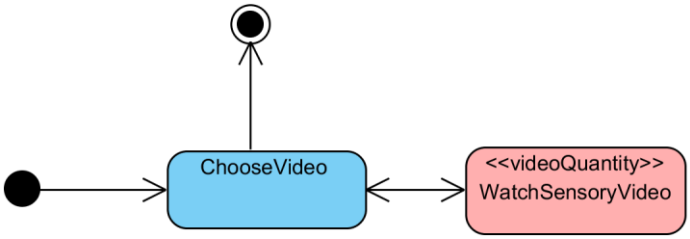


Figura 19: Modelo de Cenas *Multisensory Video*

Dando continuidade, o modelo de apresentação da cena `WatchSensoryVideo` (Figura 21) descreve os atuadores de efeitos sensoriais através das classes `WindEffectAc`, `TemperatureEffectAc`, `VibrationEffectAc`, `ScentEffectAc`, `WaterSprayEffectAc`, `LightEffectAc` e `FogEffectAc`, as quais representam respectivamente os atuadores para renderização os efeitos sensoriais de vento, temperatura, vibração, aroma, pulverizador d'água, luz e névoa respectivamente. Elas são anotadas com o estereótipo `<<OutputComponent>>` que, segundo a linguagem MML, representa um componente abstrato de saída de dados e, no contexto deste trabalho, esses dados de saída são efeitos sensoriais. O modelo de apresentação também representa elementos da interface gráfica e, no caso da cena em questão, a classe `VideoScreen` representa o *player* de vídeo. Todos os elementos visuais e sensoriais anotados com o estereótipo `<<OutputComponent>>` são organizados em um pacote chamado `PresentationUnit`.

Os valores dos componentes visuais a serem exibidos são representados tomando como base uma dependência partindo da classe para o objeto do modelo estrutural, o qual é responsável por armazenar os dados. Por exemplo, a duração do vídeo principal é obtida utilizando a dependência cujo valor é a propriedade `duration` do objeto `mainVideo`.

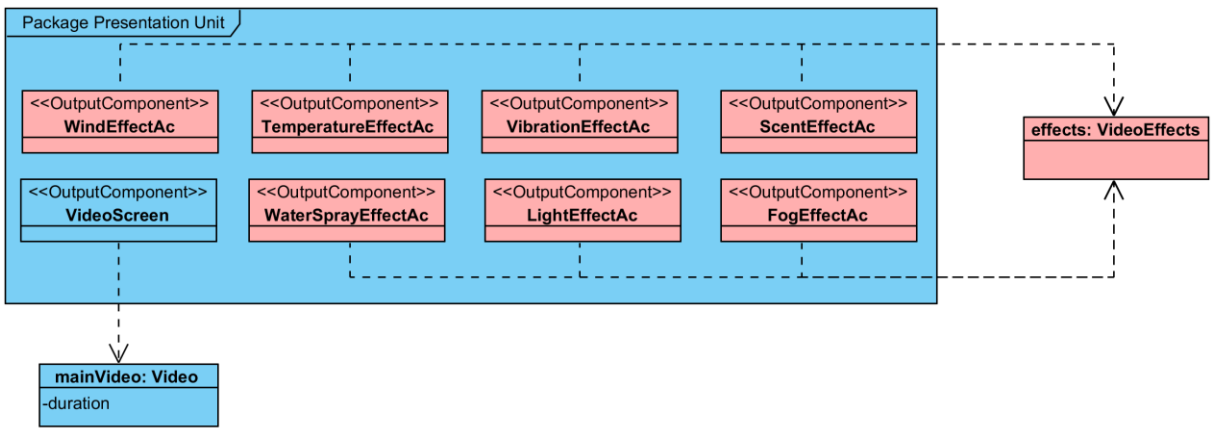


Figura 20: Modelo de Apresentação da Cena `WatchSensoryVideo`

Por último, na Figura 22 é possível observar o modelo de Interação da cena `WatchSensoryVideo`. Este modelo consiste em um diagrama de atividades UML anotado com as

variabilidades e, a partir dele, deve-se derivar o relacionamento temporal e a interatividade entre os objetos de mídia. Os pinos de saída das atividades representam condições que devem ser satisfeitas para que ações sejam realizadas, sendo estas definidas pelos pinos de entrada, quando existirem.

Assim, o fluxo 1 trata da inicialização da interface gráfica, representando o fluxo existente entre o parâmetro da atividade `showSensoryVideo` e o pino de entrada `start` que pertence ao componente da interface da cena (`ButtonBack`). Além disso, também envia a propriedade `idBind` para a atividade `PlayVideo` identificando assim o vídeo que será apresentado juntamente com os efeitos sensoriais executados paralelamente por meio da atividade `playEffects` que recebe a lista de efeitos `listOfEffects`. Por fim, o fluxo 2 especifica a ação do usuário em retornar para a cena anterior. Para tanto, este fluxo começa a partir do pino de saída `onClickStart` do componente `ButtonBack` anotado com estereótipo `<<ActionComponent>>`, que indica uma ação do sistema a ser desencadeada por meio de um evento (clique no botão da interface gráfica). Ao ser acionado pelo usuário, chama a propriedade `backScene` do componente `leaveScene`, que recebe o valor `true` indicando o fim da cena. Vale salientar que como as modelagens têm como foco a geração de código, o sincronismo fino estará relacionado a detalhes de implementação da plataforma alvo.

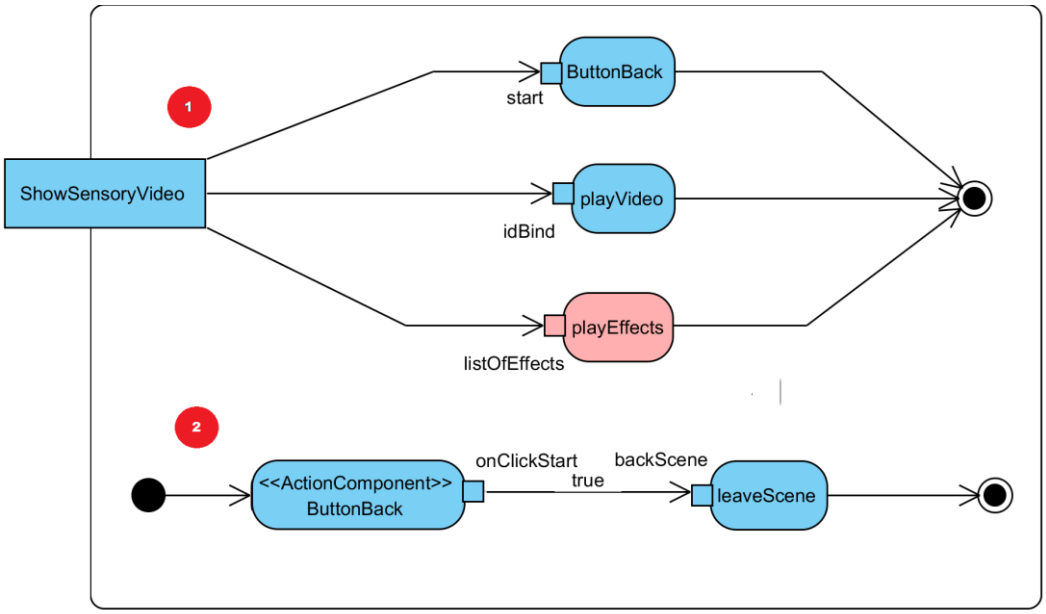


Figura 21: Modelo de Interação da Cena `WatchSensoryVideo`

5.2.4 Atividade 03 – Configurar Template

Esta atividade tem como artefato de saída o *Modelo Instanciado* que representa o processamento das variabilidades do *Modelo Template* de acordo com as opções do engenheiro da aplicação. Essas opções são satisfeitas através de uma configuração do modelo *de Features*, que é

realizada por meio de uma ferramenta de autoria chamada MulSeMaker. Esta ferramenta possui para cada família de aplicação definida na Engenharia de Domínio um *Wizard* correspondente, que auxilia o engenheiro de aplicação no preenchimento das características estruturais desejadas, bem como os valores de instância pendentes para a criação da aplicação final. Mais precisamente, nos modelos instanciados, os estereótipos referentes ao Perfil de Variabilidades (presença e multiplicidade de elementos) são substituídos pelas definições do engenheiro de aplicação, sendo essas questões descritas em maiores detalhes na Atividade 06: Configurar Produto Final. Já as transformações e mapeamentos que ocorrem entre o Modelo *Template* e o Modelo Instanciado são apresentados a seguir na Atividade 04: Implementar Transformações.

A Figura 23 apresenta o modelo instanciado do diagrama estrutural de *Multisensory Video*. Trata-se de um diagrama de objetos em que cada classe do modelo *Template* estrutural, com exceção das não selecionadas na etapa de configuração do modelo de *Features*, é mapeada para um objeto com seus valores de instância já definidos, como pode ser observado nos objetos `vibration0` e `GoE0`. Vale ressaltar que os atributos dessas instâncias correspondem a parâmetros dos efeitos sensoriais definidos na parte 3 do MPEG-V, conforme discutido na Seção 4.2.2. Continuando com a descrição do modelo, foi definida uma configuração em que a aplicação possui um vídeo associado a efeitos sensoriais de vibração, temperatura, pulverizador d'água, vento e luz, além de um `GroupOfEffects` composto pelos efeitos de aroma e névoa. Vale salientar que alguns atributos são omitidos sem comprometer o entendimento do modelo.

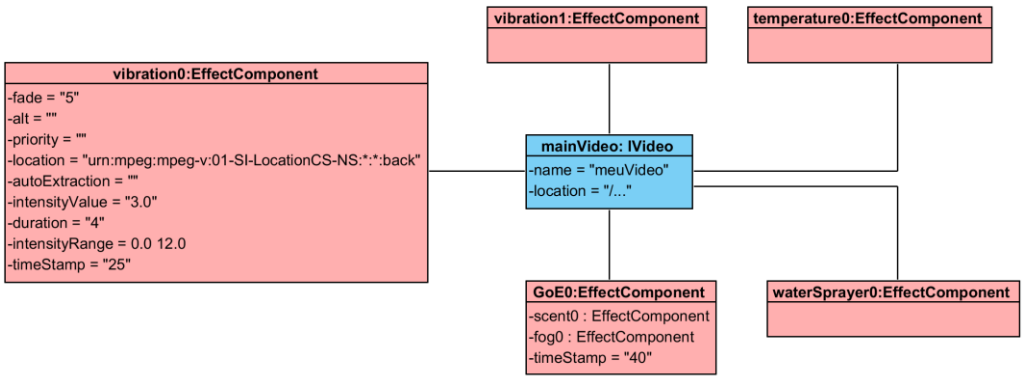


Figura 22: Modelo Instanciado Estrutural da *Multisensory Video*

Da mesma forma, o modelo de apresentação (Figura 24) instanciado representa o processamento das variabilidades associadas de presença e multiplicidade no Modelo *Template*. Em relação à presença, os atuadores para os efeitos sensoriais de vento e luz são omitidos no modelo como exemplo, pois o engenheiro de aplicação decidiu não utilizar tais efeitos para esta aplicação da família *Multisensory Video*. Já em relação à multiplicidade, um único atuador pode renderizar o mesmo efeito sensorial quantas vezes forem necessárias. Por exemplo, em um vídeo multissensorial, o efeito de vibração pode ocorrer em três momentos distintos e isso não significa que será necessário

três atuadores *vibracall* distintos. Contudo, também podem ocorrer casos em que existam mais de um atuador para o mesmo efeito sensorial. Por exemplo, dois atuadores para efeitos de luz estrategicamente posicionados em locais distintos no ambiente do usuário. Neste caso, o modelo de localização definido na parte 3 do MPEG-V (*LocationCS*) pode ser utilizado para auxiliar a identificação de qual atuador será utilizado na execução do efeito sensorial e, para este caso, o modelo de apresentação definiria dois atuadores do mesmo tipo.

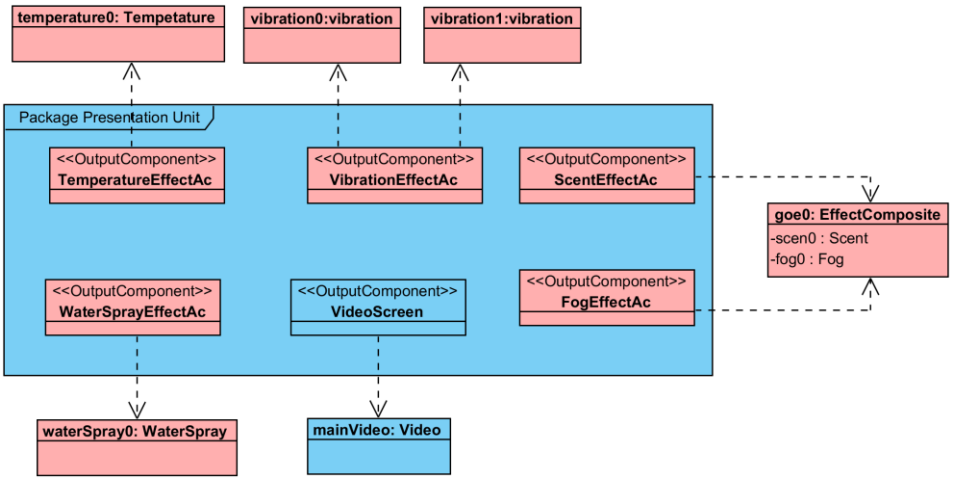


Figura 23: Modelo Instanciado de Apresentação da *Multisensory Video*

Neste ponto da abordagem é importante ressaltar que até esta atividade (Configurar *Template*), todos os modelos (FM, Modelo *Template* e Modelo Instanciado) desenvolvidos são independentes de plataforma, sendo considerados como uma arquitetura de referência para uma determinada família de aplicação mulsemedia. Dessa forma, a partir destes modelos, a família mulsemedia modelada pode ser mapeada para uma determinada plataforma ou domínio específico. No caso deste trabalho, conforme já mencionado, a *Web* foi definida como plataforma alvo e, dessa forma, as transformações e mapeamentos apresentados na Seção a seguir serão concretizados para a tecnologia HTML5, padrão da W3C para o domínio de aplicações *Web*. Contudo, é importante ressaltar a promoção do reuso em virtude dos modelos desenvolvidos até aqui, pois novas transformações a partir destes modelos podem ser desenvolvidas à medida que novas tecnologias forem surgindo, ou ainda, para domínios de aplicações diferentes.

5.2.5 Atividade 04 – Implementar Transformações

Dando continuidade, antes de detalhar a Atividade 4, é importante apresentar um conceito considerado chave no GSD: o mapeamento entre Espaço do Problema e Espaço da Solução. O primeiro refere-se ao conjunto de abstrações específicas de domínio que podem ser utilizadas para especificar o membro desejado da família do sistema, enquanto o segundo consiste em abstrações

orientadas à implementação que podem ser instanciadas para criar implementações das especificações do Espaço do Problema. Dessa forma, o mapeamento entre os espaços toma uma especificação e retorna a implementação correspondente (CZARNECKI, 2004). Vale ressaltar que o Espaço do Problema de alguém pode ser o Espaço da Solução de outrem, ou seja, em um primeiro momento, aquele que é o Espaço do Problema pode se tornar o Espaço da Solução em um segundo momento. Essa situação é conceituada como encadeamento de mapeamentos (do inglês, *Chaining of mappings*).

Com base no que foi discutido, é possível observar na Figura 25 que num primeiro momento o Modelo de *Features*, *Wizard* e o Modelo *Template* representam o Espaço do Problema, enquanto o Modelo Instanciado representa o Espaço da Solução. Num segundo momento os Modelos Instanciados representam o Espaço do Problema, e o Modelo Específico (HTML5) e código-fonte da aplicação, o espaço da solução. Para tato, ocorrem dois mapeamentos: (1) Modelo *Template* para Modelo Instanciado, que consiste em transformação do tipo M2M e (2) Modelo Instanciado para Modelo Específico (HTML5) e código-fonte da aplicação, que consistem em uma transformação M2C. Essas transformações são mais detalhadas a seguir:

- a transformação (1), do tipo M2M, recebe como entrada conjunto de modelos MML do Modelo *Template* e a configuração definida pelo *Wizard* para gerar o Modelo Instanciado. O algoritmo utilizado para transformação entre modelos é baseado na "Abordagem *Template*" (CZARNECKI e ANTKIEWICZ, 2005; CZARNECKI e HELSEN, 2006). Com base neste algoritmo, a transformação começa pelo modelo estrutural e em seguida passa para o modelo de cenas. Para cada modelo de cena existe um modelo de apresentação e interação associado. Dessa forma, uma vez gerado o modelo de cenas do Modelo Instanciado, inicia-se a geração de cada modelo de apresentação e de interação. Cada modelo é processado recursivamente a partir do elemento raiz e percorrendo a árvore em largura, ou seja, verificando em primeiro lugar os elementos de cada nível e os seus filhos da esquerda para a direita.
- a transformação (2), do tipo M2C, é realizada em dois passos, sendo o primeiro (i) o mapeamento do Modelo Instanciando para o modelo HTML5 e o segundo (ii) a transformação deste mapeamento em código-fonte HTML5. Para tanto, os elementos HTML5 são definidos como um metamodelo por meio do *framework* EMF, e a linguagem MOFScript[22] é utilizada para geração do código-final da aplicação.

[22] Disponível em: https://eclipse.org/gmt/mofscript/

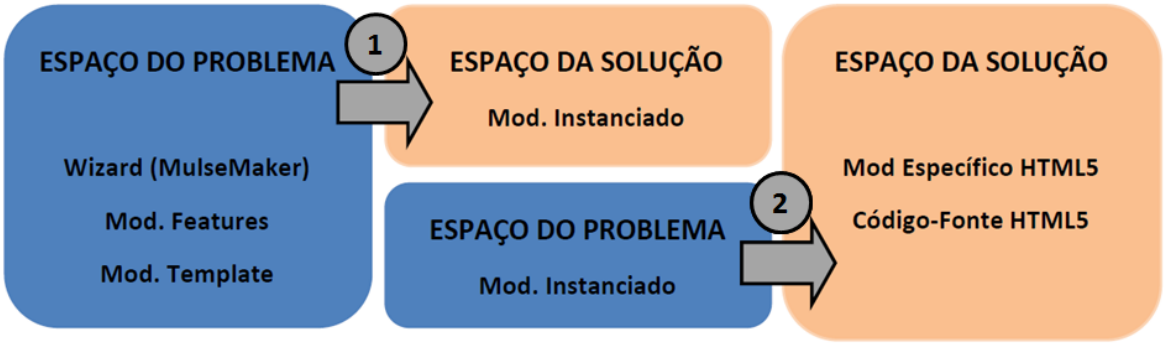


Figura 24: Encadeamento de Mapeamentos

A Figura 26 apresenta um resumo do mapeamento do Modelo Instanciado MML para o modelo HTML5. Para cada um dos quatro tipos de modelo (estrutural, cena, apresentação e interação) representando o *template*, o mecanismo de transformação procura por elementos específicos de cada tipo de diagrama UML. Para diagrama de atividades, parâmetros, fluxos e ações são buscados. Já em relação a diagramas de estado, as transições são examinadas e em relação ao diagrama estrutural são consultadas as propriedades. Cada elemento presente em um dos diagramas UML é mapeado para um elemento HTML5. Por exemplo, os atributos de texto, imagem e vídeo do modelo estrutural são transformados, respectivamente, em elementos HTML de parágrafo `<p>`, de imagem `<img>` e de vídeo `<video>`. Já o modelo de interação é mapeado em links `<a>` ou em botões `<button>` para navegação entre páginas, solicitar algum serviço *Web*, etc. Mais uma vez, é importante salientar que esta atividade é está diretamente relacionada a uma plataforma tecnológica específica. Dessa forma, caso seja interesse do cliente final que a ferramenta realize geração de código para outras plataformas, como TV Digital, Cinema 7D, instalações multimídia, dentre outras, será necessário a implementação de novas transformações para tais plataformas.

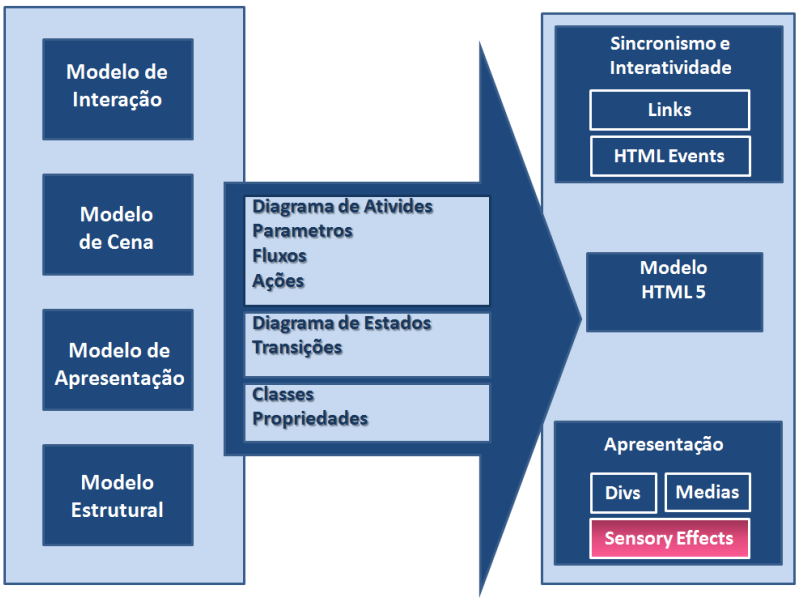


Figura 25: Mapeamentos de elementos do modelo MML para o modelo HTML5

A fim de ilustrar de forma mais clara como ocorre a geração de código dos efeitos sensoriais, na Figura 27 é possível observar o código do modelo instanciado de um vídeo anotado com um efeito sensorial de vibração (parte superior da figura) e o respectivo código HTML5 gerado com a *tag* `<vibration-effect>`, que faz parte da extensão realizada no modelo HTML5 apresentada em detalhes na Seção seguinte.

```
39    <video url="video1.mp4" title="Video 1">
40      <effectsList>
41        <sensorialEffects
42          startTime="1.0"
43          type="vibration"
44          endTime="3.0"
45          intensity="10.0"/>
46      </effectsList>
47    </video>
```

```
 8   <video id="video1" src="assets/videos/video1.mp4"></video>
 9   <mulsemedia-box id="box-video1">
10         <vibration-effect
11         intensity="10.0"
12         startTime="1.0"
13         endTime="3.0">
14     </vibration-effect>
15   </mulsemedia-box>
```

Figura 26: Trecho de código gerado a partir de modelo instanciado na família *Multisensory Video*

5.2.6 Atividade 05 – Implementar Extensões

O HTML5 não provê atualmente suporte declarativo ao desenvolvimento de aplicações Mulsemedia. Dessa forma, a fim de evitar a necessidade de programação imperativa em *JavaScript* para renderização dos efeitos sensoriais, foi criada uma extensão da linguagem HTML5 por meio de novas *tags* sensoriais. Para realizar esta atividade, os parâmetros e tipos de efeitos sensoriais definidos

pelo padrão MPEG-V foram tomados como base. Além disso, com o objetivo de promover o reuso e encapsulamento das novas *tags* sensoriais foi utilizado um conjunto de tecnologias (*Templates*, *Shadow DOM*, *Custom Elements* e *HTML Imports*) da W3C, chamado *Web Components*, que funcionam como *widgets* de interface de usuário reutilizáveis. Os *Templates* definem partes de código inertes à página até que sejam ativados pelo *JavaScript*. Dentro da *tag* `<template>` é definida a estrutura do conteúdo estático dos componentes que, neste caso, são as novas *tags* de efeitos sensoriais. Com o *Custom Elements* é possível criar elementos customizados com nomes e *scripts*. Após a criação do componente, ele é transferido para uma árvore DOM, chamada *Shadow* DOM, na qual irá formar seu próprio escopo, encapsulando seus dados e garantindo que não ocorram conflitos futuros a respeito de *styles*, *names* e IDs de elementos. O registro destes novos elementos é realizado pelo padrão *Custom Elements*, que tornam esses novos tipos em elementos DOM. Em seguida, é possível utilizar as novas *tags* dos elementos personalizados na marcação HTML após importá-los por meio da *tag* `<link>`, conforme código a seguir:

```
1    <link rel="import" href="wc/mulsemedia-box.html">
```

Foram criadas algumas *tags* para possibilitar a simulação dos efeitos sensoriais utilizando *Web Components*. A primeira delas foi a *tag* `<mulsemedia-box>` que tem como objetivo simular os efeitos sensoriais, indicando qual efeito está sendo reproduzido em determinado momento e seus atributos peculiares como tipo, intensidade. Esta *tag* possui dois atributos: *idBind* e *onClickStart*. O primeiro atributo representa uma ligação entre o simulador de efeitos sensoriais e o elemento associado a ele. O segundo atributo é opcional e tem como finalidade disparar um efeito sensorial a partir de um evento, como, por exemplo, um clique de um botão. Foram criadas *tags* para representar os efeitos sensoriais de vibração, luminosidade, temperatura, aroma, névoa e pulverizador d'água. Elas são, respectivamente: `<vibration-effect>`, `<wind-effect>`, `<light-effect>`, `<temperature-effect>`, `<scent-effect>`, `<fog-effect>`, `<water-spray>`. Os atributos em comum desse conjunto de *tags* são `startTime`, `endTime`, `intensity`, `fade`. Os atributos `startTime` e `endTime` representam, respectivamente, o tempo inicial e final, em segundos, que o efeito sensorial será reproduzido. O terceiro atributo, `intensity`, representa o nível de intensidade do efeito sensorial. O `fade` é um atributo opcional e representa uma taxa de crescimento ou decrescimento da intensidade do efeito. Existe também um atributo exclusivo para a *tag* `<scent-effect>` que representa o tipo de aroma `type_scent`. Assim, questões como a implementação e o sincronismo dos efeitos sensoriais são resolvidos no *JavaScript* de forma totalmente encapsulada, favorecendo desenvolvedores não especialistas

5.2.7 Exemplos de Uso

Nesta Seção será demonstrada a utilização das novas *tags* sensoriais por meio de exemplos a fim de apresentar como ocorre a sincronização e a integração dos efeitos sensoriais. Duas formas são possíveis: (i) sincronização de efeitos sensoriais com a reprodução de mídias contínuas (vídeos e áudios) de acordo com suas linhas temporais (*timeline*); e (ii) na sincronização de efeitos sensoriais na utilização de mídias discretas (imagens e textos) por meio do paradigma orientado a eventos. O primeiro exemplo ilustra uma situação (i) em que efeitos sensoriais são sincronizados a um vídeo de acordo com sua linha do tempo:

```
1   <video id="video1" src="video/energy.mp4"></video>
2   ...
3   <mulsemedia-box idBind="video1">
4     <temperature-effect intensity="20" startTime="9" fade="0.5" endTime="18"></temperature-effect>
5     <scent-effect intensity="50.0" startTime="12" endTime="18" type_scent="combustion"> </scent-effect>
6     <wind-effect intensity="100" startTime="27.5" endTime="32"> </wind-effect>
7     <light-effect intensity="95" endTime="32" startTime="27.5"> </light-effect>
8     <fog-effect intensity="50.0" endTime="32" startTime="28.5"> </fog-effect>
9     <water-spray intensity="50" startTime="43" endTime="46"> </water-spray>
10    <vibration-effect intensity="50" startTime="43" endTime="46"> </vibration-effect>
11  </mulsemedia-box>
```

Vale ressaltar que o atributo `onClickStart` não foi utilizado neste caso, pois o sincronismo dos efeitos sensoriais está relacionado com a *timeline* do vídeo representado pela ligação `idBind="video1"`(linha 03). Dessa forma, quando o vídeo ligado ao `<mulsemedia-box>` alcançar 9 segundos de tempo transcorrido, um efeito de temperatura com intensidade 20 será disparado, com crescimento de 0.5 de intensidade até o vídeo atingir 18 segundos de duração (linha 04). Note que para esses casos o uso do atributo `idBind` é obrigatório.

De forma similar, o segundo exemplo ilustra uma situação (ii) em que efeitos sensoriais são vinculados a uma imagem e possuem como gatilho para a execução um evento de clique de botão que é associado ao atributo `onClickStart` de `<mulsemedia-box>` (linha 3):

```
1   <img id="img1" src="img/class.jpg">
2   <button id="buttonImg1" type="button">Hydraulic Energy</button>
3   <mulsemedia-box idbind="img1" onClickStart="buttonImg1">
4     <vibration-effect intensity="50.0" startTime="1" endTime="5" fade="0.5"></vibration-effect>
5     <light-effect intensity="50.0" startTime="1" endTime="5"></light-effect>
6     <water-spray intensity="40.0" startTime="1" endTime="5"></water-spray>
7   </mulsemedia-box>
```

Quando o evento de clique de botão ocorre, a imagem ligada a `<mulsemedia-box>` é exibida de forma sincronizada com efeitos sensoriais definidos (neste caso vibração, luz e pulverizador

d'água) e são renderizados conforme os parâmetros definidos até `endTime` atingir 5 segundos (linha 4). Ainda sobre efeitos sensoriais para as mídias discretas, o desenvolvedor pode optar por iniciar os efeitos no momento em que a mídia é renderizada na tela (a partir de uma transição de telas em um aplicativo de celular, por exemplo). Para tanto, basta não utilizar o atributo optativo `onClickStart` de `<mulsemedia-box>`.

5.2.8 Integração com Serviços de Lógica Complexa

Uma importante funcionalidade implementada neste trabalho é a possibilidade de integrar de forma simples efeitos sensoriais à lógica imperativa de programação, ou seja, isso significa ter a resposta de serviços Web (por exemplo, uma consulta a um banco de dados, um serviço de localização ou ainda um serviço de clima/tempo, etc.) influenciando na execução dos efeitos sensoriais. Para tanto, outra *tag* foi criada para oferecer suporte ao cenário descrito: `<conditional-effect>`. Ela possui quatro atributos (`service`, `in`, `op` e `ifReturn`) e o seu funcionamento se dá da seguinte forma: o atributo `service` determina qual o serviço utilizado; o `in` é o ID do `<input>` e seu valor será o parâmetro de entrada a ser enviado para o servidor; o `op` é o operador de comparação utilizado para avaliar a condição; o `ifReturn` é a condição a ser avaliada de acordo com a resposta do serviço. Como *Multisensory Video* não possui integração com serviços de lógica imperativa, aqui é apresentado um trecho de código da família *Multisensory E-Commerce*, como pode ser observado a seguir:

```
1   <img id="img2" src="img/deal.jpg">
2   <input id="cardnumber" type="text" name="cardnumber">
3   <button id="buttonImg2" type="button">Buy Deal</button>
4   ...
5   <mulsemedia-box idbind="img2" onClickStart="buttonImg2">
6     <temperature-effect intensity="40.0" startTime="1" endTime="4"></temperature-effect>
7     <conditional-effect service="dealService" in="cardnumber" ifReturn="1" op="=">
8       <vibration-effect intensity="50.0" startTime="1" endTime="5" fade="0.5"> </vibration-effect>
9     </conditional-effect>
10    <conditional-effect service="dealService" in="cardnumber" ifReturn="1" op="!=">
11      <light-effect intensity="50.0" startTime="1" endTime="5"></light-effect>
12    </conditional-effect>
13  </mulsemedia-box>
```

Neste último exemplo, o serviço de transação financeira por cartão de crédito `dealService` recebe o número do cartão `cardnumber`. A ideia é que os efeitos executados dependam da autorização da transação financeira. Observe que a *tag* `<conditional-effect>` (linhas 7 e 10) deve ser adicionada como um *child* da *tag* `<mulsemedia-box>` (linha 5). O retorno do serviço é comparado com o valor especificado pelo atributo `ifReturn` utilizando o operador identificado pelo atributo `op` (linhas 7 e 10) e caso resulte em verdadeiro os efeitos atribuídos como *child* do `<conditional-effect>` serão disparados, caso contrário, os efeitos não serão executados. Note

ainda que neste exemplo o efeito `<temperature-effect>` (linha 6) está fora de qualquer condição e, portanto, será executado independentemente da resposta do servidor.

5.2.9 Atividade 06 – Configurar Produto Final

Na última atividade da abordagem, *Configurar Produto Final*, todo o esforço investido na Engenharia de Domínio é aproveitado por usuários não-especialistas que se utilizarão de uma ferramenta com alto nível de abstração para desenvolver aplicações mulsemedia de acordo com suas necessidades. Com o intuito de facilitar a compreensão do comportamento da ferramenta MDD e fornecer uma visão geral das funcionalidades oferecidas aos desenvolvedores de aplicações mulsemedia, a seguir são detalhados os casos de uso observados na Figura 28. Vale salientar que o desenvolvedor aqui corresponde ao especialista de domínio.

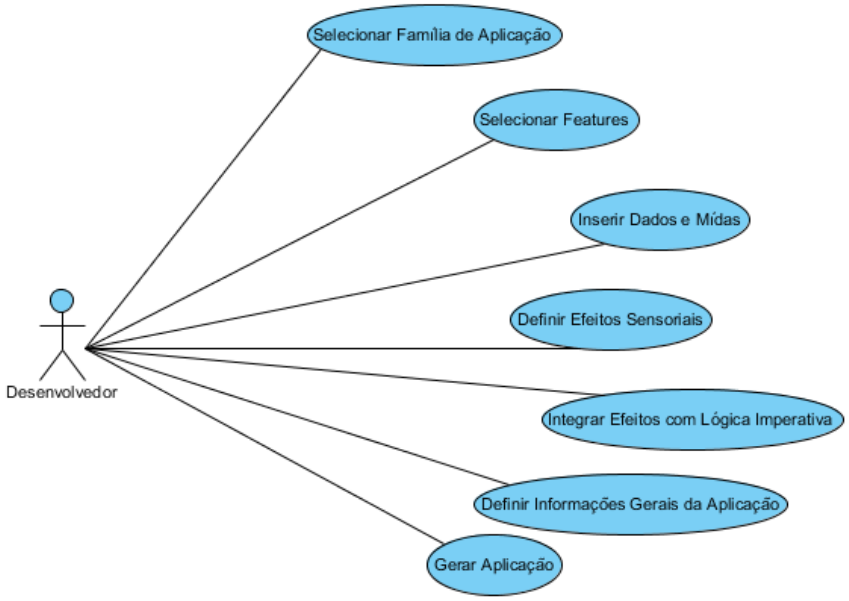


Figura 27: Diagrama de Casos de Uso da Ferramenta MulSeMaker

- **Selecionar família de aplicação:** a ferramenta MulSeMaker permite ao desenvolvedor selecionar a família de aplicação desejada para gerar um produto desta família. No presente momento, existem as famílias *Multisensory Video*, *Multisensory E-Commerce* e *Multisensory Education*. Para cada uma delas existe um *Wizard* correspondente na ferramenta MulSeMaker que auxiliará o preenchimento do *template* com valores de instância referentes aos elementos de mídia (vídeos, imagens textos, efeitos sensoriais e serviços *Web*). Também são definidas as características estruturais da aplicação, como a utilização ou não de uma cena opcional da família;

- **Selecionar *Features*:** permite ao desenvolvedor determinar o aspecto do produto final por meio da seleção das características (*features*) pré-definidas;
- **Inserir dados e mídias:** permite ao desenvolvedor inserir os valores de instância e as mídias que serão utilizadas no produto final;
- **Definir efeitos sensoriais:** permite ao desenvolvedor definir os efeitos sensoriais para cada elemento de mídia inserido no formulário de dados, bem como seus parâmetros. A Figura 29 apresenta a tela correspondente à inserção de um efeito sensorial em um vídeo previamente definido;
- **Integrar Efeitos com Lógica Imperativa:** permite ao desenvolvedor integrar efeitos sensoriais à lógica de programação imperativa, como serviços *Web*, consultas a banco de dados, etc;
- **Definir informações gerais da aplicação:** permite ao desenvolvedor definir informações gerais do produto final, como o nome da aplicação, nome da empresa detentora da aplicação, cores, etc;
- **Gerar Aplicação:** permite ao desenvolvedor emitir o comando Criar Aplicação, que enviará todas as informações de instância definidas para o servidor onde o produto final será gerado.

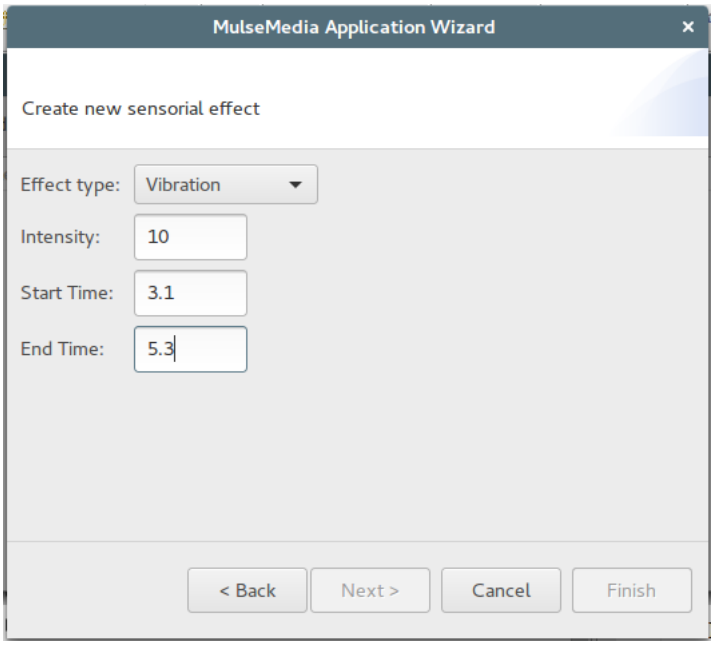


Figura 28: Tela da Ferramenta MulSeMaker com *Wizard* da Família *Multisensory Vídeo*

5.3 Considerações sobre o Capítulo

Este Capítulo descreveu a abordagem MDD para o desenvolvimento de aplicações mulsemedia proposta neste trabalho. Foi apresentada uma visão geral da abordagem, além do seu

detalhamento, que foi realizado por meio da concretização da abordagem em um exemplo de uso tratando a família de aplicações de vídeos multissensoriais: *Multisensory Video*.

A abordagem foi organizada com base nas fases de Engenharia de Domínio e Engenharia de Aplicação do GSD propostas por Czarnecki e Eisenecker (2000). Suas etapas e principais artefatos gerados por meio das atividades realizadas se dão da seguinte forma:

- **Engenharia de Domínio:** (i) Análise de Domínio – especificação do Modelo de *Features*; (ii) Projeto de Domínio – projeto do Modelo de *Template* e Modelo Instanciado; e (iii) Implementação de Domínio – implementação das transformações M2M e M2C, além das extensões necessárias (bibliotecas e módulos).
- **Engenharia de Aplicação:** (i) Derivação de Produto – geração da aplicação de acordo com as definições do engenheiro de aplicação.

O próximo Capítulo apresenta o projeto de avaliação que visa utilizar provas de conceitos e estudos empíricos.

6 AVALIAÇÃO

Este Capítulo apresenta os estudos realizados a fim de avaliar o impacto da adoção de uma abordagem de desenvolvimento orientado a modelos no domínio de aplicações mulsemedia. Na seção 6.1, são utilizada as técnicas de persuasão e implementação (SHAW, 2001) para ilustrar o emprego da abordagem MDD no desenvolvimento de dois exemplos de uso. Já na Seção 6.2, são apresentados os estudos empíricos (SHAW, 2001) realizados por meio do projeto de experimentos controlados para coletar e analisar dados (quantitativos e qualitativos) do uso abordagem em comparação com outras metodologias.

6.1 Implementação das Provas de Conceito

Nesta Seção são implementadas duas provas de conceito definidas com base nos seguintes critérios: (i) cenários apontados na literatura como potenciais beneficiários da mulsemedia (vide Capítulo 4 sobre mulsemedia); (ii) aplicações mulsemedia possuindo requisito de integração entre efeitos sensoriais com lógica de aplicação complexa e, por fim, (iii) aplicações mulsemedia no subdomínio *Web* cujas atividades 4 e 5 (Implementar Transformações e Extensões) da abordagem MDD proposta foram realizadas (vide Capítulo 5 sobre abordagem MDD para aplicações mulsemedia).

As principais tecnologias empregadas na concretização da abordagem foram o *framework* EMF (STEINBERG *et al.*, 2008) para a criação do Modelo *Template*, o *plugin* FMP (ANTKIEWICZ e CZARNECKI, 2004) para criação do modelo de *Features* e a linguagem MOFScript (OLDEVIK, 2011) para a transformação M2C a fim de gerar o código-final da aplicação mulsemedia. Essas tecnologias foram utilizadas na implementação de cada família de aplicação que são manipuladas por meio da ferramenta MulSeMaker, sendo está ultima implementada em Java.

6.1.1 Exemplo de Uso 1: Multisensory E-commerce

Neste exemplo de uso foi definida a aplicação *Multisensory E-commerce*. Trata-se de uma aplicação em que os usuários podem observar e adquirir determinados produtos por meio de transação eletrônica. Esses produtos podem ser apresentados por meio de vídeos, imagens e textos anotados com efeitos sensoriais. Se o usuário se interessar por algum produto, é possível realizar a aquisição por meio de serviços para compras e pagamentos *online*, e, além disso, a resposta da transação influencia quais efeitos são executados. As áreas de publicidade e de comércio eletrônico foram apontadas por Sulema (2016), Petit *et al.* (2015) e Guinea (2014), dentre outros, como grandes beneficiárias da mulsemedia, sendo o cenário da prova de conceito descrito motivado, em especial, pelo conceito de publicidade sensorial (do inglês, *sensory branding*). Trata-se de uma abordagem de publicidade que

capacita marcas a criar associações emocionais e conexões com as pessoas através de experiências multissensoriais, que solidificam sentimentos positivos, pensamentos e opiniões sobre uma marca (KRISHNA, 2012). Um exemplo real disso foi o anúncio criado para dispositivos móveis pela rede de TV americana *Showtime*[23] a fim de divulgar o seriado *Homeland*[24]. O anúncio consistia em um vídeo curto com cenas para promover a próxima temporada da atração, contudo, o vídeo entregava também vibrações produzidas por meio do atuador de *vibracall* dos ceulares, correspondendo às cenas de ação, tais como bombas explodindo (JOHNSON, 2014).

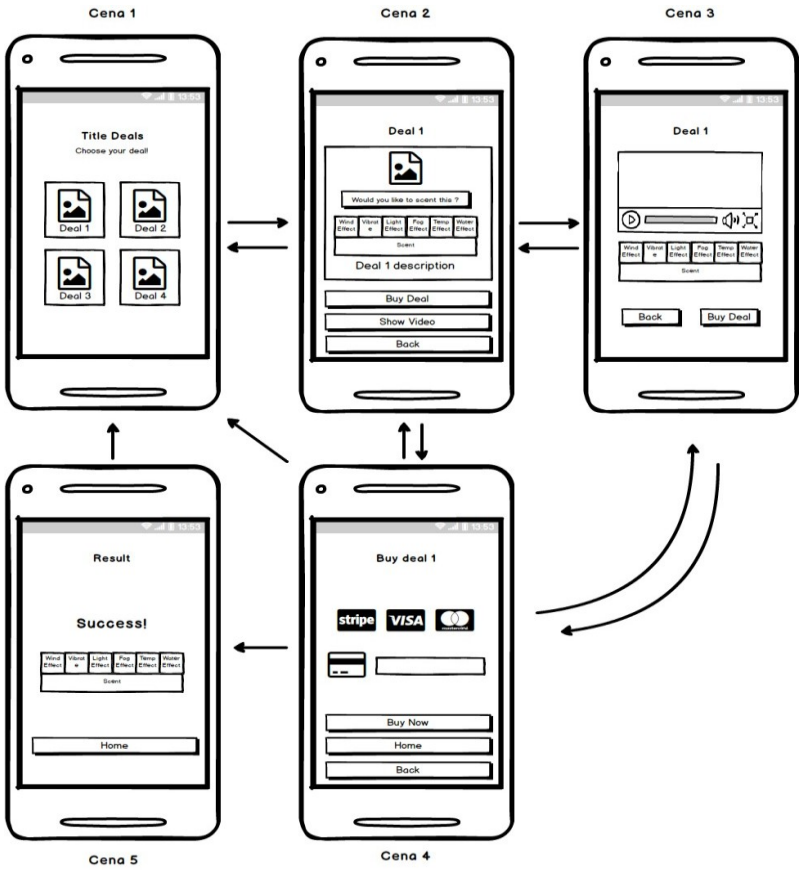


Figura 29: *Storyboard* da aplicação *Multisensory E-Commerce*

A Figura 30 apresenta uma das aplicações da família *Multisensory E-commerce* composta por 5 (cinco) cenas. Como ainda não existem ferramentas de prototipação que levem em consideração os efeitos sensoriais, eles serão descritos textualmente sempre que ocorrerem, como a seguir:

- **Cena 1:** corresponde à tela inicial com as opções de produtos disponíveis para compra ilustrados por pequenas imagens e o nome do produto;

[23] Disponível em: <http://www.sho.com/sho/home>

[24] Disponível em: <http://www.sho.com/sho/homeland/home>

- **Cena 2:** corresponde a uma descrição textual do produto acompanhada de um *banner* sensorial. Nesta cena, o usuário pode disparar um efeito sensorial relacionado ao *banner* com um toque na tela, caso deseje;
- **Cena 3:** corresponde a um vídeo multissensorial sobre o produto;
- **Cena 4:** corresponde à tela para compra do produto por meio de algumas opções de serviços de transação e pagamentos online; e
- **Cena 5:** corresponde à resposta da transação financeira. Nela, dependendo da resposta do serviço, um efeito sensorial diferente é executado**.**

6.1.1.1 Atividade 1 – Levantar Requisitos

Conforme já explicado, o Modelo de *Features* (FM) é usado para representar as características de todas as aplicações pertencentes a uma determinada família. O *Multisensory E-Commerce* FM é apresentado na Figura 31 e está organizado nos seguintes modelos: `Structural`, `DataInstantiation`, `VideoEffectsInstantiation` e `ImageEffectsInstantiation`. A Figura 31a apresenta o modelo `Structural` em detalhes. A característica obrigatória `DealsQuantity` define o número de produtos diferentes que serão oferecidos na aplicação final. A *feature* obrigatória de `VideoInformation` está relacionada a uma apresentação por meio de vídeo multissensorial sobre um produto. Já a *feature* opcional `ImagemInformation` está relacionada à opção de também utilizar um *banner* multissensorial na aplicação final ou não. Por fim, a *feature* opcional `FinantialTransactionService` define os diferentes tipos de serviços que podem ser oferecidos para realização de uma transação financeira. Ainda na Figura 31a, é possível observar em detalhes as *features* do modelo `DataInstantiation` que são relacionadas aos valores de instância da aplicação.

Já o modelo `videoEffecsInstantiation` é apresentado em detalhes na Figura 31b. Basicamente, ele modela as características do SEDL. Todas as *features* são opcionais e representam efeitos sensoriais que podem ser associados a um vídeo. Todos os efeitos têm cardinalidade [0..*]. Os seguintes efeitos são modelados: vento, nevoeiro, vibração, temperatura, pulverizador d'água, aroma e luz. Também é possível observar parâmetros opcionais definidos de acordo com o padrão MPEG-V: `duration`, `fade`, `alt`, `priority`, `location`, `autoExtraction`, `intensityValue`, `intensityRange` e `timestamp`. Esses parâmetros são comuns a todos os efeitos sensoriais e são descritos em maiores detalhes no Capítulo 4 sobre mulsemedia. Ainda sobre o modelo `VideoEffecsInstantiation`, ele fornece a *feature* opcional `GroupOfEffects` para associar mais de um efeito sensorial ao mesmo `timestamp` (Figura 31c). Mais precisamente, a cardinalidade `<2-7>` significa que estarão associados ao mesmo `timestamp` no mínimo 2 (dois) e no máximo 7

(sete) efeitos sensoriais. Por último, apesar de o modelo `ImageEffectsInstantiation` estar com seus detalhes omitidos, ele foi definido da mesma forma que `videoEffecsInstantiation`.

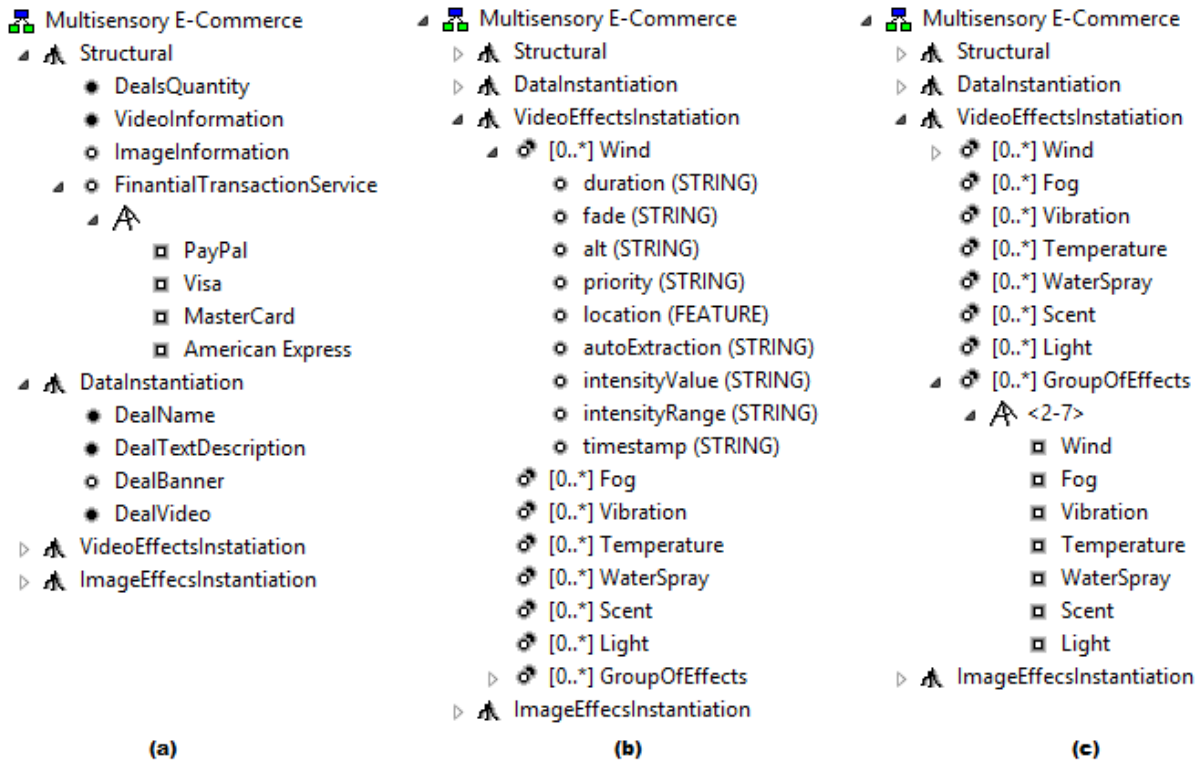


Figura 30: Modelo de *Features* da Família *Multisensory E-Commerce*

6.1.1.2 Atividade 2 – Modelar Template

O *Multisensory E-Commerce Template Model* é descrito através dos modelos relacionados às visões estruturais e comportamentais da linguagem MML. Seguindo as diretrizes da MML, o primeiro passo é a modelagem estrutural. Ela mostra os elementos do domínio, bem como a sua relação com os respectivos objetos de mídia. A Figura 32 mostra o modelo estrutural da família *Multisensory E-Commerce*. Ele contém uma classe que representa o aplicativo (`MultisensoryECommerce`), que tem associações com uma ou mais ofertas de produtos (`Deal`) e também com uma classe representando um serviço *Web* para transação financeira (`FinancialTransaction`). Ainda neste modelo, é possível observar elementos de domínio associados aos seguintes elementos de mídia: `DealBanner`, `DealDescription` e `DealVideo`. Eles são, respectivamente, um objeto de mídia de imagem para exibir a oferta do produto visualmente, um objeto de mídia de texto para descrever a oferta do produto em mais detalhes e, finalmente, um objeto de mídia de vídeo também para demonstrar a oferta do produto aos clientes. Cada uma dessas classes possui um estereótipo de acordo com seu tipo, sendo o primeiro anotado com o estereótipo `<<Text>>`, o segundo anotado com o estereótipo `<<Image>>` e o terceiro anotado com o estereótipo `<<Video>>`.

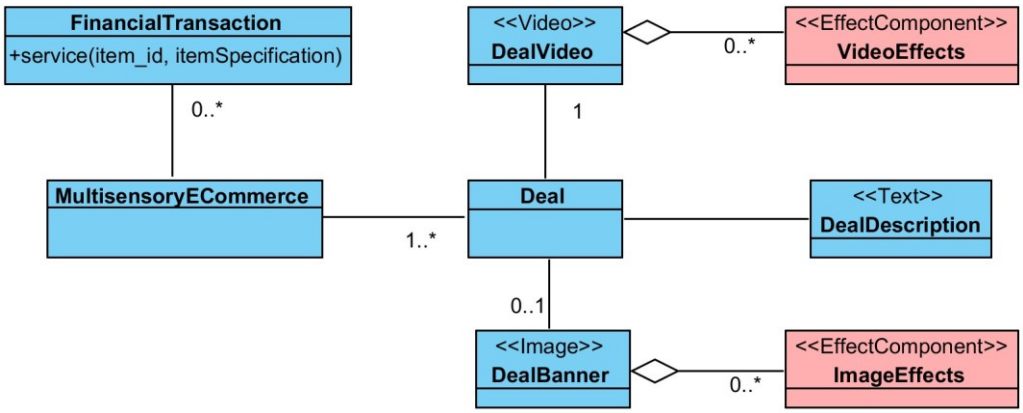


Figura 31: Modelo Estrutural do *Multisensory E-Commerce Template Model*

No que diz respeito aos efeitos sensoriais, é possível observar os elementos de domínio `VideoEffects` e `ImageEffects`. O relacionamento de agregação entre `DealVideo` e `VideoEffects` significa que é possível que um ou mais objetos de efeitos sensoriais, ou ainda, grupos de efeitos sensoriais (`groupOfEffects`), podem ser associados ao `DealVideo`. O mesmo ocorre entre `DealBanner` e `ImageEffects`. O estereótipo `<<EffectComponent>>` foi criado no metamodelo MML para representar os efeitos sensoriais.

O próximo modelo é o de Cena, que deve especificar o comportamento geral da aplicação por meio de uma máquina de estados em que cada estado corresponde a uma cena e as transições entre os estados correspondem às transições entre as cenas. A Figura 33 apresenta o modelo de Cena da família *Multisensory E-Commerce Template Model*, em que é possível notar a presença de 5 (cinco) estados representando as seguintes cenas: (i) cena para listar os produtos disponíveis (`ChooseDeal`), (ii) cena opcional detalhando o produto com uma imagem, efeito sensorial e texto (`DealImageDescription`), (iii) cena detalhando o produto com vídeo multissensorial (`DealVideoDescription`), (iv) cena com opções de pagamento para compra do produto (`PurchaseDescription`) e (v) cena com resposta da transação financeira (`PurchseResponse`).

Tendo em vista as questões relacionadas à variabilidade, o modelo de cena está anotado com três estereótipos: `<<DealsQuantity>>`, `<<ImageDescription>>` e `<<not_ImageDescription>>`. O primeiro é aplicado às cenas `DealImageDescription` e `DealVideoDescription`, sendo o significado desta variabilidade de multiplicidade relacionado à criação de cópias das referidas cenas para cada produto da aplicação, ou seja, cada produto terá a sua cena de vídeo multissensorial e cena de *banner* multissensorial. O segundo e o terceiro estereótipos estão associados à presença ou ausência da cena opcional `DealImageDescription`.

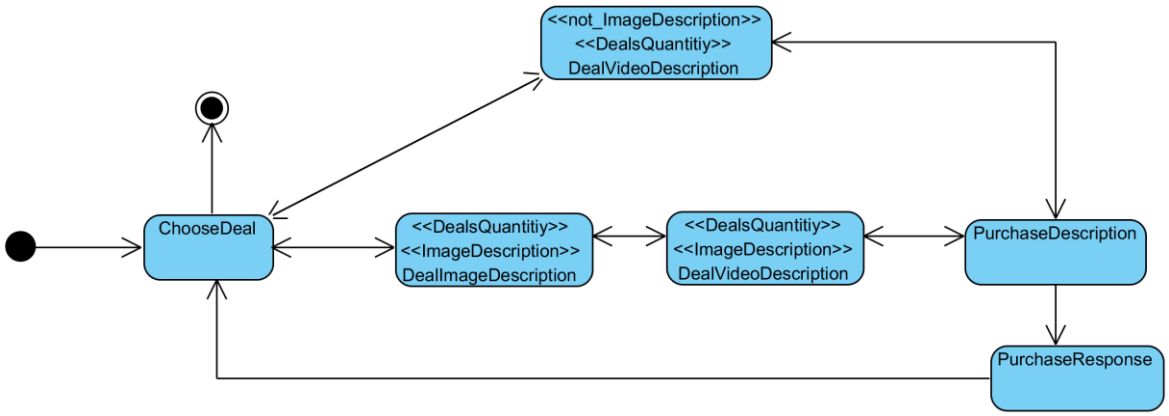


Figura 32: Modelo de Cena do *Multisensory E-Commerce Template Model*

Segundo as definições da MML, cada cena possui um modelo de apresentação e um modelo de interação associados. Dessa forma, a Figura 34 mostra o Modelo de Apresentação associado à cena `DealImageDescription` que descreve os elementos da interface gráfica. Nesta cena, o usuário pode observar as seguintes informações: (i) um *banner* multissensorial relacionado a determinado produto escolhido (`ImageScreen`), (ii) uma descrição textual do produto (`TextDescription`), e (iii) um botão para disparar um efeito sensorial ou grupo de efeitos relacionado ao *banner* (`PlayEffect`). Também estão presentes no modelo os atuadores de efeitos sensoriais: `WaterSprayEffectAc`, `VibrationEffectAc`, `WindEffectAc`, `TemperatureEffectAc`, `LightEffectAc`, `FogEffectAc`, `ScentEffectAc`.

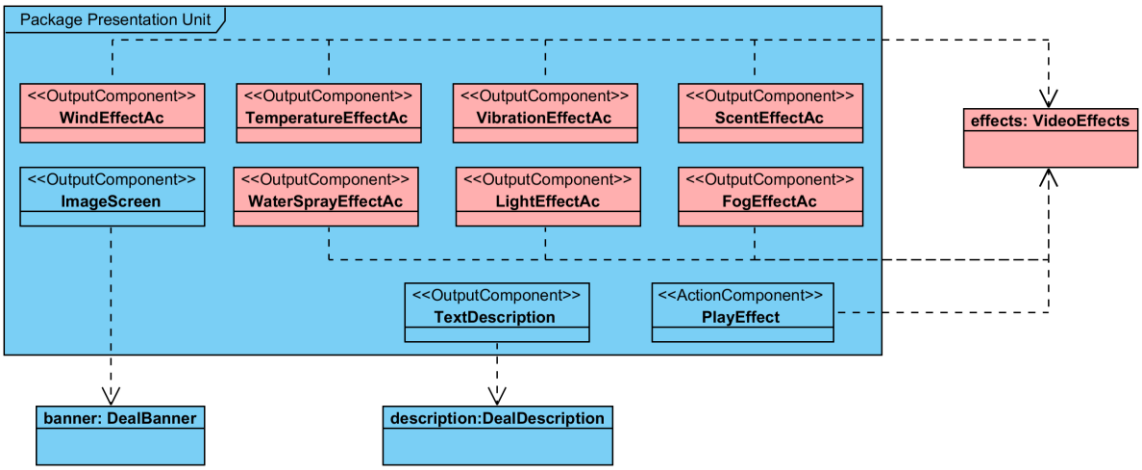


Figura 33: Modelo de Apresentação da cena `PurchaseDescription`

Ainda sobre o modelo de apresentação, todos os atuadores, além de `ImageScreen` e `TextDescription` estão anotados com o estereótipo `<<OutputComponent>>` que, de acordo com o MML, representa um componente abstrato de saída de dados para mostrar informações ao usuário. Já o estereótipo `<<ActionComponent>>` anotado a classe `PlayEffect` permite ao usuário desencadear uma ação do sistema por meio de um evento, como um clique de botão. Os

componentes visuais que podem ser exibidos são representados por uma relação de dependência da classe ao objeto do modelo estrutural, que é responsável por armazenar os dados. Por exemplo, os parâmetros dos efeitos sensoriais são obtidos por meio dessas dependências.

Por último, na Figura 35 é possível observar o modelo de Interação da cena `PurchaseDescription`. Este modelo consiste em um diagrama de atividades UML anotado com as variabilidades, e representa ações realizadas em decorrência de eventos. Dessa forma, por meio deste modelo foi definida a interatividade entre os objetos de mídia, bem como o relacionamento temporal. Para auxiliar na compreensão, a figura foi marcada para identificar os três fluxos principais. Além disso, os pinos de saída das atividades representam condições que devem ser satisfeitas para que ações sejam realizadas, sendo estas definidas pelos pinos de entrada.

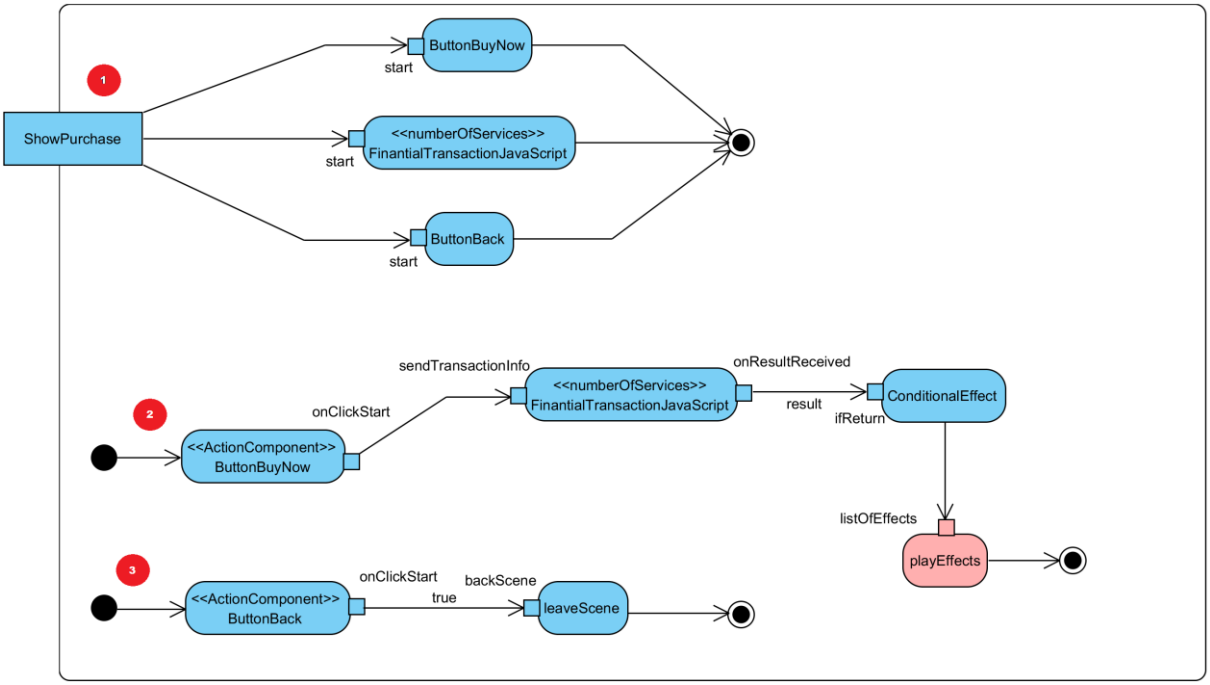


Figura 34: Modelo de Interação da Cena `PurchaseDescription`

Assim, o fluxo 1 trata da inicialização da interface gráfica, representando os fluxos existentes entre o parâmetro da atividade `showPurchase` e os pinos de entrada `start` que pertencem aos componentes da interface da cena. Já o fluxo 2 representa a compra de um produto por meio de uma transação financeira, sendo os efeitos sensoriais condicionados ao sucesso ou falha desta transação. Mais precisamente, este fluxo começa a partir do pino de saída `onClickStart` do componente `ButtonBuyNow` (anotado com o estereótipo `ActionComponent`) que ao ser acionado por meio do clique do usuário no botão da interface gráfica chama a ação `sendTransactionInfo` do componente `FinancialTransactionJavaScript`. Esta ação é responsável por enviar ao servidor as informações necessárias para efetuar a transação financeira. O servidor, por sua vez, com

as informações necessárias à disposição, verifica se é possível concretizar a transação financeira e retorna o resultado (`result`), que é recebido pelo pino de saída `onResultReceived`. Em seguida, `result` é utilizado no componente `conditionalEffect` que compara o valor de `result` com a propriedade `ifReturn` do mesmo componente a fim de verificar se a transação financeira foi bem sucedida ou não. Com base nessa avaliação é definida a lista de efeitos sensoriais (`listOfEffects`) que seguirá para o módulo de apresentação (`playEffects`). Por fim, o fluxo 3 especifica a ação do usuário em retornar para a cena anterior. Para tanto, este fluxo começa a partir do pino de saída `onClickStart` do componente `ButtonBack` (anotado com estereótipo `ActionComponent`) que ao ser acionado por meio do clique do usuário no botão da interface gráfica chama a propriedade `backScene` do componente `leaveScene` que recebe o valor `true` indicando o fim da cena.

6.1.1.3 Atividade 3 – Configurar Template

Conforme já mencionado, a atividade *Configurar Template* consiste no processamento das variabilidades do *Modelo Template* de acordo com as opções do engenheiro da aplicação satisfeitas através de uma configuração do Modelo de *Features* que é criada pela ferramenta MulseMaker. Essa atividade obtém como resultado o Modelo Instanciado da família *Multisensory E-Commerce*, e para demonstrá-lo, são apresentados os modelos instanciados estrutural e de cena.

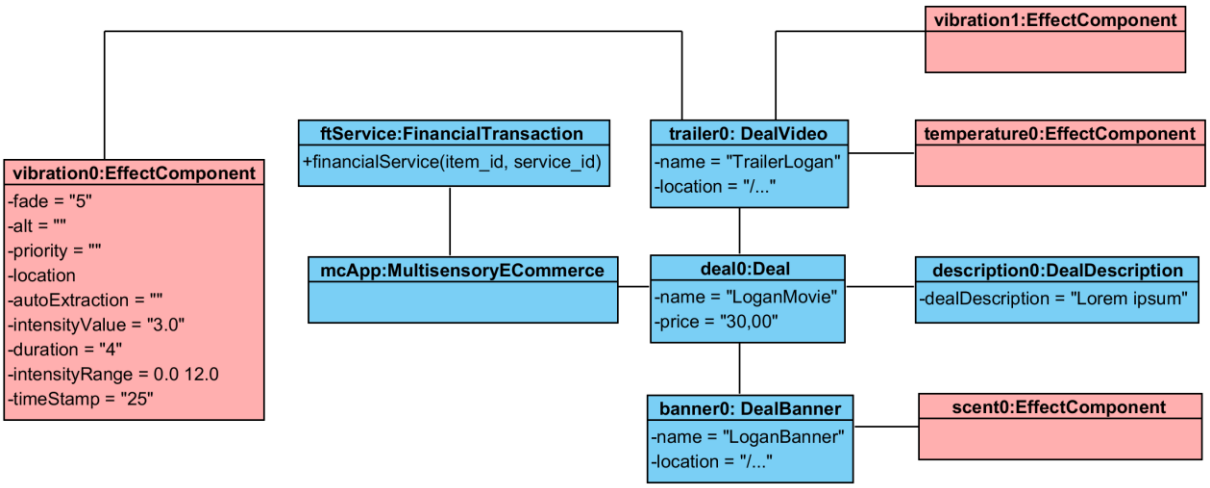


Figura 35: Modelo Instanciado Estrutural de *Multisensory E-Commerce*

Na Figura 36 é possível observar a instância do modelo estrutural obtida pela transformação do modelo estrutural em um diagrama de objetos. Neste exemplo, o engenheiro de aplicação optou pela seguinte configuração: apenas um produto para ser negociado (`Deal0`); um vídeo multissensorial (`Trailer0`) acompanhado de três efeitos sensoriais (`Vibration0`, `Vibration1` e `Temperature0`) e uma imagem representando o *banner* multissensorial do produto acompanhada de um efeito sensorial (`scent0`). Além disso, foram preenchidos os valores das propriedades e os

parâmetros dos efeitos sensoriais associados às mídias de vídeo e imagem. Note que os parâmetros dos efeitos sensoriais foram omitidos, com exceção de `vibration0` por não comprometer o entendimento e reduzir as dimensões do modelo.

O modelo de cenas instanciado é apresentado na Figura 37. Nela, é possível notar que o engenheiro de aplicação optou por utilizar a cena opcional `DealImageDescription`, bem como definiu a variabilidade de multiplicidade de produtos `DealsQuantity` como sendo 2 (dois), dessa forma, as cenas `DealImageDescription` e `DealVideoDescription` estão duplicadas, sendo cada uma relacionada a um produto diferente.

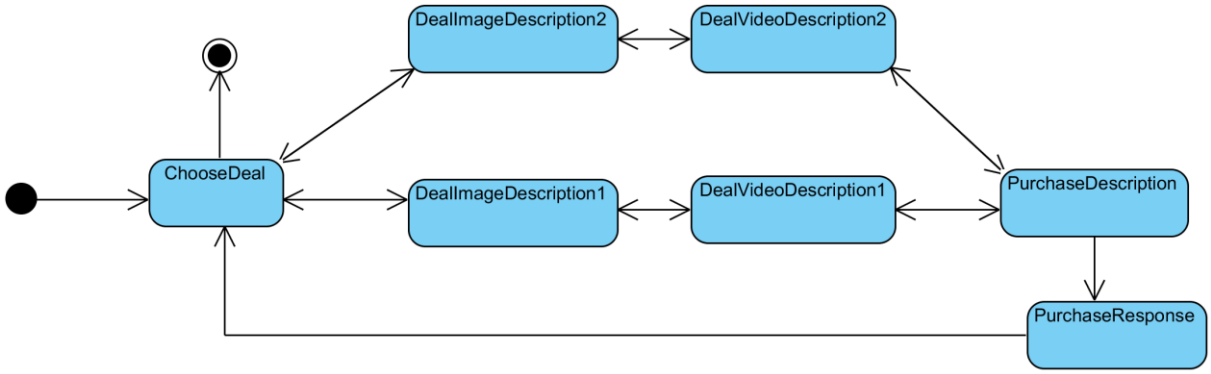


Figura 36: Modelo Instanciado de Cena de *Multisensory E-Commerce*

6.1.1.4 Atividade 6 – Configurar Produto Final

A última atividade *Configurar Produto Final* é realizada na etapa de engenharia de aplicação por meio da ferramenta MulSeMaker que disponibiliza um conjunto de formulários de aplicações (*Wizards*) por meio dos quais o usuário especialista de domínio (neste caso poderiam ser profissionais da área de marketing, comunicação, mídias digitais, dentre outros) preenche as informações pendentes dos *templates* (modelos MML anotados com variabilidades) relacionadas às variabilidades e aos valores de instância para que a aplicação final possa finalmente ser gerada. Para tanto, por meio da ferramenta MulSeMaker é escolhido o *Wizard* para configurar uma aplicação da família *Multisensory E-Commerce*. A Figura 38 apresenta Ferramenta MulSeMaker utilizando o *Wizard* da Família *Multisensory E-Commerce*. É possível observar a tela de Seleção das *Features* (Figura 38a) por meio da qual são definidas as características estarão presentes na aplicação final gerada. Por exemplo, são definidas quais as mídias principais serão utilizadas (áudio e vídeo) e quais delas possuirão efeitos sensoriais. Além disso, também são definidas as opções de serviços *Web* para transação financeira que serão disponibilizadas na aplicação gerada. Também é possível observar a tela para inserção das informações relacionadas ao vídeo (Figura 38b). A tela seguinte permite a inserção dos efeitos sensoriais na mídia de vídeo e é similar a interface apresentada na Figura 29 do *Wizard* da família *Multisensory Video*.

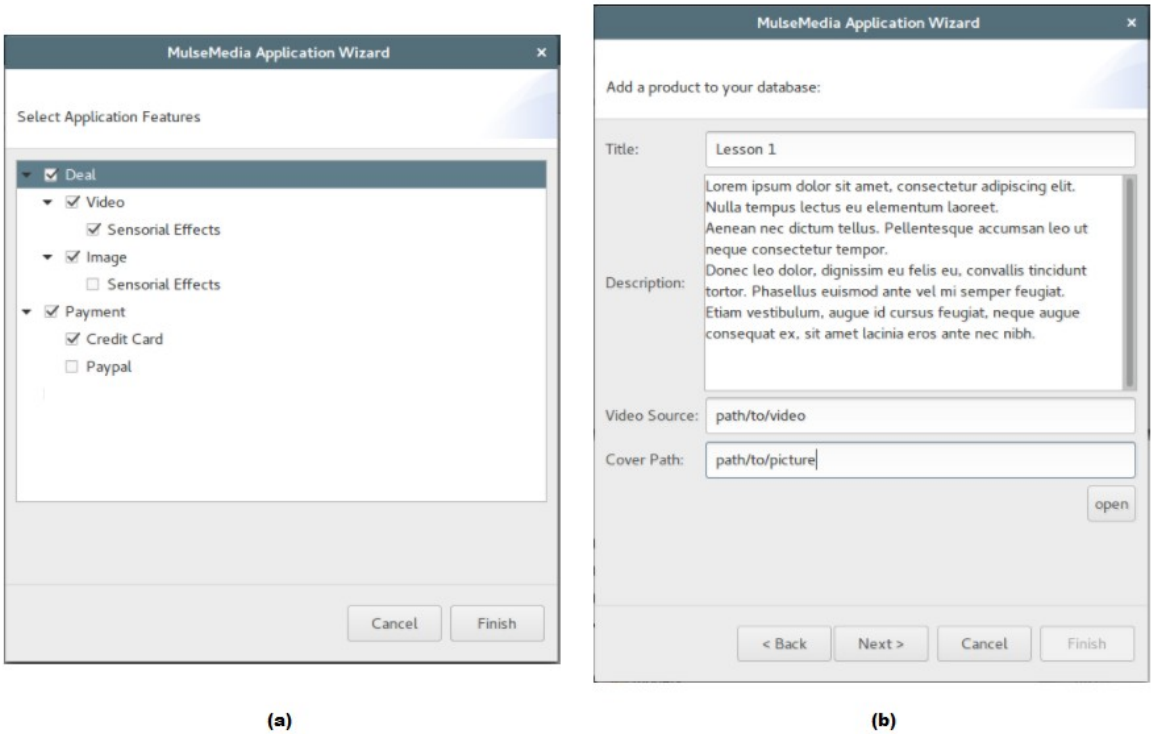


Figura 37: Ferramenta MulSeMaker com *Wizard* da Família *Multisensory E-Commerce* – (a) Tela de Seleção das *Features* utilizadas (b) Tela para adição das informações referentes ao vídeo

6.1.2 Exemplo de Uso 2: Multisensory Education

Neste exemplo de uso foi definida a aplicação *Multisensory Education*. Trata-se de uma aplicação em que os usuários podem realizar cursos por meio de aulas com conteúdos multissensoriais. Esses cursos são compostos por vídeos, imagens e textos anotados com efeitos sensoriais sobre o conteúdo a ser ministrado. Se o aluno se interessar, é possível realizar a uma revisão multissensorial dos principais conceitos abordados após a aula e, além disso, existe um *quiz* sobre o conteúdo ministrado como última atividade, sendo um efeito sensorial executado ao final dependendo do resultado obtido. A área de educação também foi apontada por Sulema (2016) e Guinea (2014), dentre outros, como grande beneficiária da mulsemedia, já existindo, inclusive, resultados que comprovam essa afirmação, como em Zou *et al.* (2017).

A Figura 39 apresenta uma das aplicações da família *Multisensory Education* composta por 5 (cinco) cenas descritas a seguir:

- **Cena 1:** corresponde à tela inicial com as opções de aulas à disposição dos alunos;
- **Cena 2:** corresponde a uma aula multissensorial que consiste em um vídeo anotado com efeitos sensoriais de forma a melhorar a QoE do aluno no decorrer da aula
- **Cena 3:** corresponde à revisão da aula previamente assistida. Os conceitos são apresentados de forma textual associado a uma imagem e um efeito sensorial correspondente;

- **Cena 4:** corresponde a um *quiz* tradicional a ser respondido para avaliar os conhecimentos obtidos; e
- **Cena 5:** corresponde ao resultado final do *quiz* que é apresentado juntamente com os efeitos sensoriais vinculados ao resultado obtido.

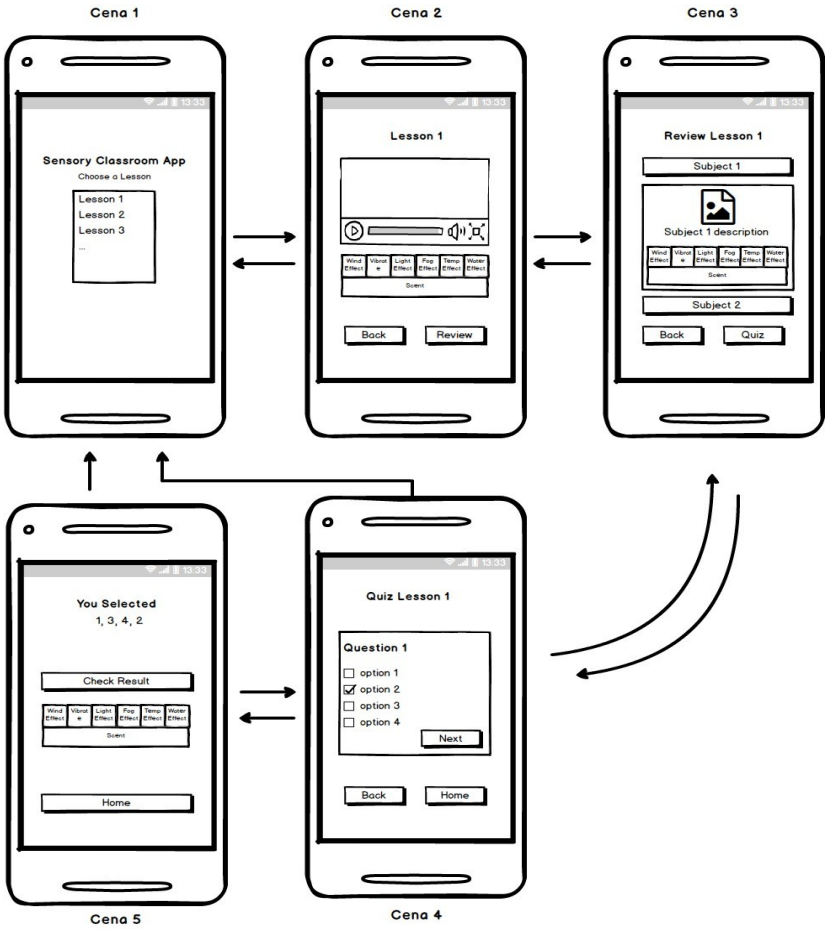


Figura 38: *Storyboard* da Família *Multisensory Education*

6.1.2.1 Atividade 1 – Levantar Requisitos

O Modelo de *Features* (FM) da família *Multisensory Education* está organizado nos seguintes modelos: `Structural`, `DataInstantiation` e `VideoEffectsInstantiation`. A Figura 40a apresenta o modelo `Structural` em detalhes e nela é possível observar as seguintes *features*: (i) a *feature* obrigatória `LessonsQuantity` que define o número de lições diferentes que serão oferecidas na aplicação final; (ii) a *feature* opcional `ReviewScene` que está relacionada à presença ou ausência de uma cena de revisão multissensorial na aplicação final; e a (iii) *feaure* opcional `numberOfTopics` relacionada a quantos tópicos serão abordados na revisão multissensorial. Na Figura 40b, é possível observar em detalhes as *features* do modelo `DataInstantiation` que são relacionadas aos valores de instância da aplicação. Aqui, vale destacar a *feature* `LessonImageReview` de cardinalidade [0..3] que corresponde à cena de revisão multissensorial

composta por definições textuais (`ReviewTextDefinition`) de conceitos apresentados na aula combinadas a uma imagem com efeitos sensoriais associados (`ReviewSensoryEffects`). Esses efeitos sensoriais devem ser explorados de tal forma que reforce o conceito apresentado. Por exemplo, na aula sobre fontes de energia, o conceito de energia eólica é apresentado e associado ao efeito sensorial de vento. Essa é uma atividade realizada pelo especialista de domínio na etapa de engenharia de aplicação. A Figura 40c detalha o modelo `Quiz` que possui *features* associadas às perguntas (no mínimo 3 e no máximo 5), alternativas e resposta correta. Também é definido neste modelo efeitos sensoriais para o caso do aluno se sair bem na avaliação (`SuccessSensoryEffect`), bem como para o caso de ocorrer um resultado insatisfatório (`FailureSensoryEffect`). Por fim, vale salientar que os modelos `videoEffecsInstantiation`, `SuccessSensoryEffect`, `FailureSensoryEffect` relacionados a efeitos sensoriais possuem as mesmas características dos exemplos anteriores, sendo seu detalhamento aqui omitido.

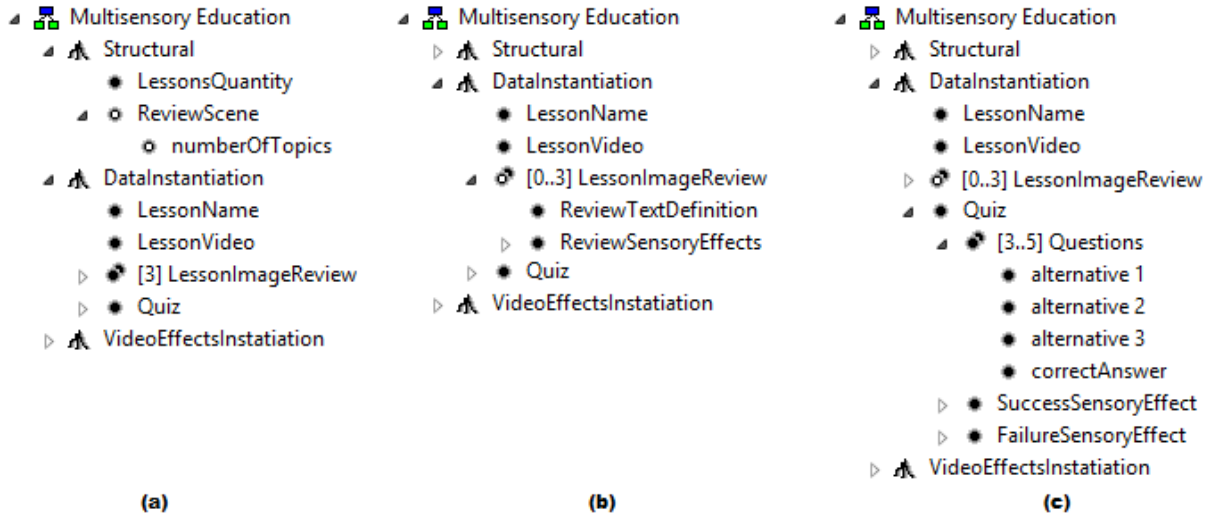


Figura 39: Modelo de Features da Família *Multisensory Education*

6.1.2.2 Atividade 2 – Modelar Template

A Figura 41 apresenta o modelo estrutural do *Multisensory Education Template Model*. Ele contém uma classe que representa o aplicativo (`MultisensoryEducation`), que tem associações com uma ou mais lições multissensoriais (`Lessons`). Uma lição corresponde a um vídeo multissensorial (`LessonVideo`), possivelmente seguido por uma revisão multissensorial (`ReviewLesson`) e encerrando com um quiz para avaliar os conhecimentos obtidos (`Quiz`). A revisão, conforme já abordado, está relacionada a até três imagens (`ImageReview`), sendo esta última associada a um conceito (`ReviewTopic`) discutido na aula e a efeitos sensoriais (`ImageEffects`) que devem ser utilizados de forma conjunta de tal maneira que reforcem o

processo cognitivo. Por fim, a classe `Quiz` está associada à classe `QuizEffects` que são efeitos sensoriais definidos para serem executados juntamente com o resultado da avaliação.

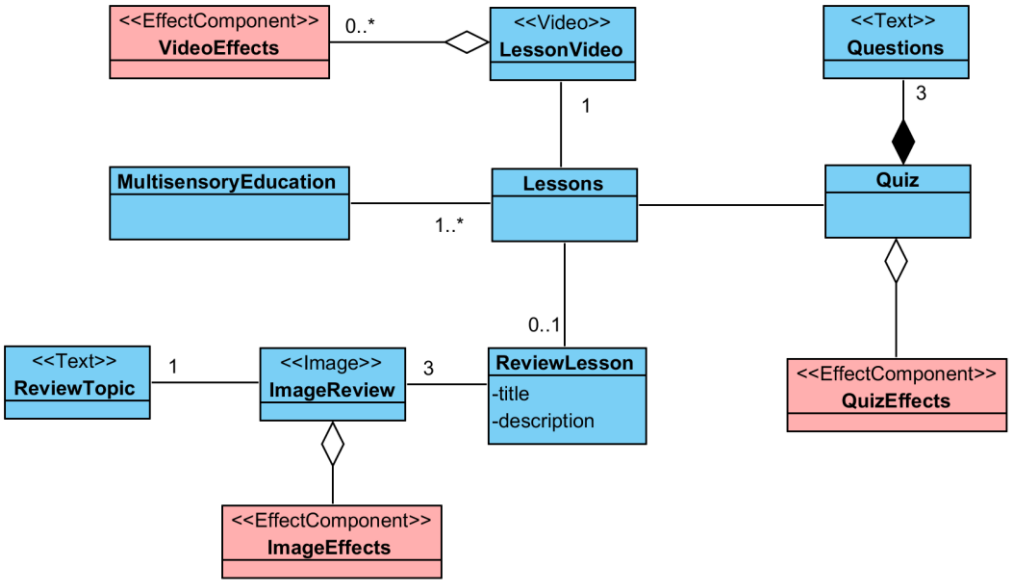


Figura 40: Modelo Estrutural do *Multisensory Education Template Model*

O próximo modelo do *Template* da família *Multisensory Education* é o de Cena e está ilustrado na Figura 42, por meio da qual é possível observar as transições entre os de 5 (cinco) estados possíveis, representando as seguintes cenas: (i) cena para listar as lições disponíveis (`ChooseLesson`); (ii) cena representando a aula multissensorial por meio de vídeos e efeitos sensoriais sincronizados na linha do tempo (`LessonVideo`); (iii) cena de revisão multissensorial reforçando conceitos discutidos na cena `LessonVideo` por meio de imagens, efeitos sensoriais e definições textuais (`Review`); (iv) cena com perguntas e respostas para avaliar os conhecimentos obtidos (`Quiz`); e (v) cena final em que um efeito sensorial é apresentado juntamente com resultado da avaliação realizada na cena `Quiz`, sendo este efeito definido de acordo com o resultado obtido, ou seja, existirá um efeito associado a um bom desempenho e outro efeito associado a um mau desempenho.

Tendo em vista as questões relacionadas à variabilidade, o modelo de cena está anotado com três estereótipos: `<<LessonsQuantity>>`, `<<review>>` e `<<not_review>>`. O primeiro é aplicado as cenas `LessonVideo` e `Review`, sendo o significado desta variabilidade de multiplicidade relacionado à criação de cópias das referidas cenas para cada lição da aplicação, ou seja, cada lição terá a sua cena de vídeo multissensorial e cena de revisão. O segundo e o terceiro estereótipos estão associados à presença ou ausência da cena opcional `Review`.

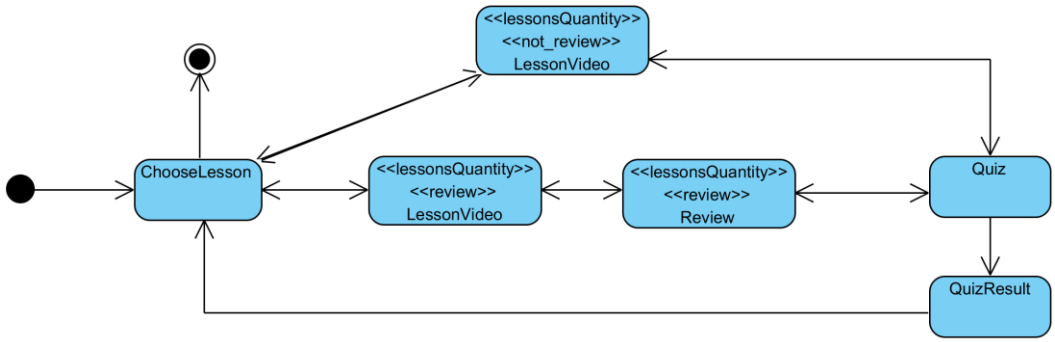


Figura 41: Modelo de Cena do *Multisensory Education Template Model*

Continuando, a Figura 43 mostra o Modelo de Apresentação associado à cena `Review`. Basicamente, nesta cena o usuário pode revisar conceitos importantes apresentados na cena anterior (`LessonVideo`) por meio de imagens e efeitos sensoriais associados. Como pode ser observado, de forma similar ao modelo da família *Multisensory E-Commerce*, os atuadores de efeitos sensoriais são representados através das classes `WaterSprayEffectAc`, `VibrationEffectAc`, `WindEffectAc`, `TemperatureEffectAc` e `LightEffectAc`, `ScentprayEffectAc` e `FogEffectAc` anotados com o estereótipo `<<OutputComponent>>`. Além destes, ainda existem os seguintes elementos de interface gráfica: (i) `Topic` representado a explicação textual de um conceito discutido na aula da cena `LessonVideo`, (ii) `ImageTopic` representando uma imagem associada a um conceito discutido na aula da cena `LessonVideo` e (iii) `PlayTopic` representando um botão (`ActionComponent`) que ao ser tocado revela na tela `Topic` e `ImageTopic`, além do efeito sensorial associado. `Topic` e `ImageTopic` estão anotados com o estereótipo `<<numberOFTopics>>` relacionado a variabilidade de multiplicidade.

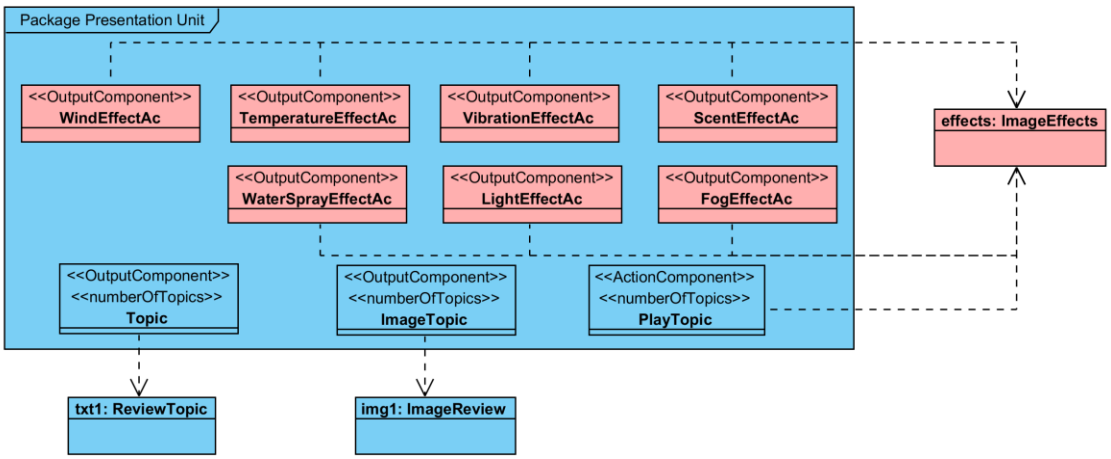


Figura 42: Modelo de Apresentação da cena `Review`

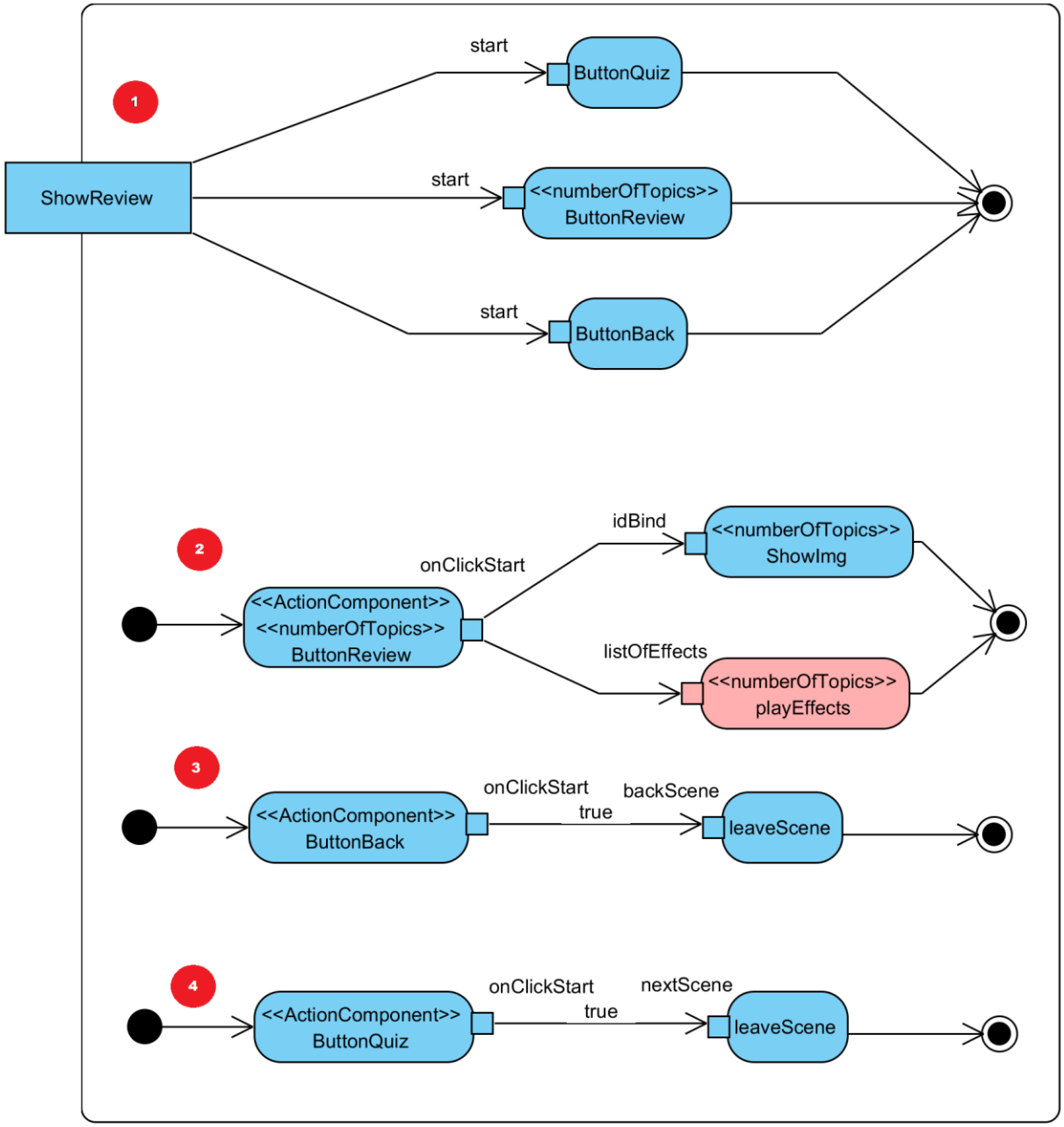


Figura 43: Modelo de Interação da Cena *Review*

Por último, na Figura 44 é possível observar o modelo de Interação da cena `Review`. O fluxo 1 trata da inicialização da interface gráfica (`ButtonQuiz`, `ButtonReview` e `ButtonBack`), representando os fluxos existentes entre o parâmetro da atividade `showReview` e os pinos de entrada `start` que pertencem aos componentes da interface da cena. O estereótipo `<<numberOfTopics>>` aplicado a `ButtonReview` está relacionado à variabilidade de multiplicidade de elementos, ou seja, por meio dela o engenheiro de aplicação irá definir quantos

tópicos relacionados ao conteúdo ministrado na aula serão utilizados na revisão. Já o fluxo 2 representa a revisão em si, começando a partir do pino de saída `onClickStart` do componente `ButtonReview` (anotado com o estereótipo `ActionComponent`) que ao ser acionado por meio do clique do usuário no botão da interface gráfica realiza duas ações: (i) envia a propriedade `idBind` para a atividade `ShowImg` identificando assim a imagem que será apresentada juntamente com os (ii) efeitos sensoriais executados paralelamente por meio da atividade `playEffects` que recebe a lista de efeitos `listOfEffects`. Por fim, de forma semelhante ao modelo de interação de *Multisensory E-Commerce*, os fluxos 3 e 4 especificam a ação do usuário em retornar para a cena anterior e seguir para a cena posterior, respectivamente.

6.1.2.3 Atividade 3 – Configurar Template

A Figura 45 apresenta o modelo instanciado estrutural da família *Multisensory Education* que consiste no resultado do processamento das variabilidades e preenchimento dos valores de instância com base nas opções definidas pelo engenheiro de aplicação. Neste exemplo, optou-se por não utilizar as mídias relacionadas à revisão multissensorial, não aparecendo nenhuma delas no modelo instanciado. Assim, optou-se pela seguinte configuração: apenas uma aula (`chemistry`); um vídeo multissensorial (`soe`) acompanhado de um efeito sensorial (`Scent0`); e, por fim, o *quiz* (`quiz`) composto por três questões e associado a dois efeitos sensoriais (`vibration0` e `light0`) que são utilizados para a apresentação do resultado do *quiz*. Um dos efeitos será utilizado para caso de sucesso e o outro para uma má avaliação.

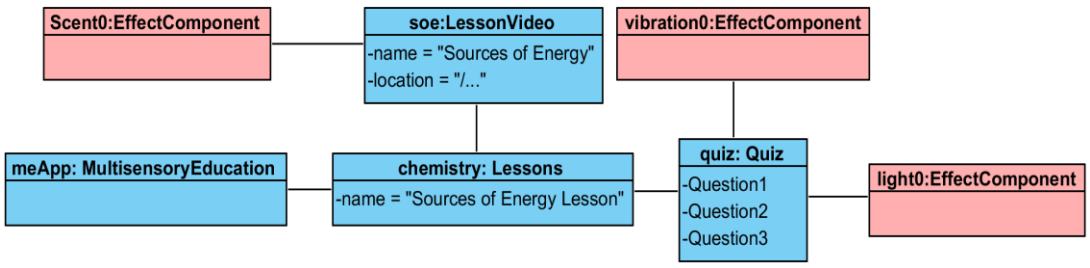


Figura 44: Modelo Instanciado Estrutural de *Multisensory Education*

O modelo de cenas instanciado é apresentado na Figura 46. Nela, é possível notar que o engenheiro de aplicação optou por não utilizar a cena opcional `Review` e, dessa forma, após assistir o vídeo multissensorial correspondente a lição, o aluno irá diretamente realizar o *quiz* para avaliar o conhecimento retido da aula. Também foi decidido aplicar a variabilidade de multiplicidade de `LessonsQuantity` como sendo 1 (um), dessa forma, a cena `LessonVideo` aparece apenas uma vez, sendo relacionada a apenas uma lição.

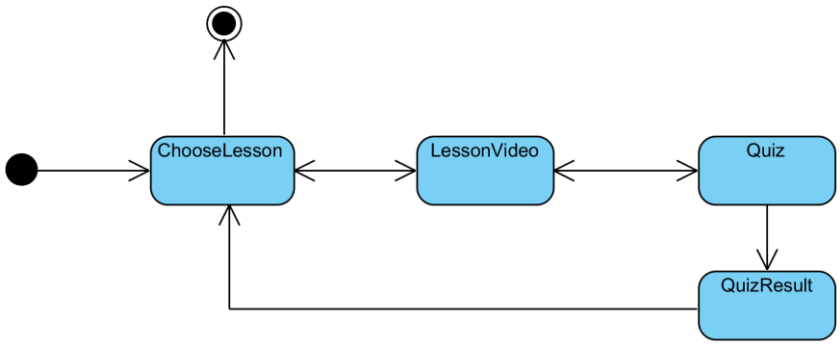


Figura 45: Modelo Instanciado de Cena de *Multisensory Education*

O modelo instanciado de apresentação da cena *Review* apresentado na Figura 47. Nela, é possível notar que o engenheiro de aplicação optou por utilizar apenas os efeitos sensoriais de vibração e aromas. Também foi decidido aplicar a variabilidade de multiplicidade de `numberOfTopics` como sendo 3 (três) e, dessa forma, aparecem três tópicos a serem revisados, cada um com uma imagem e um botão associado.

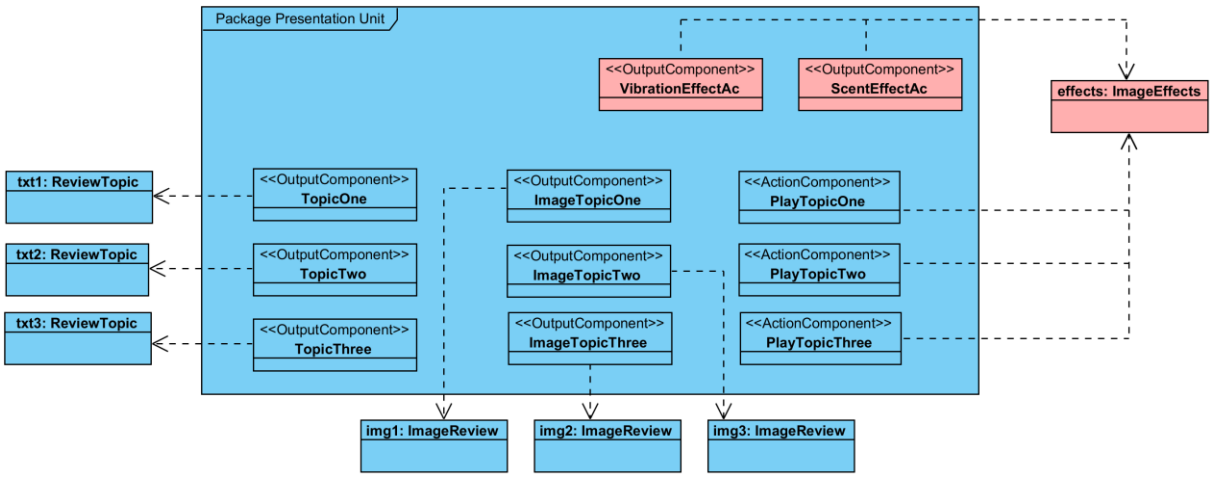


Figura 46: Modelo Instanciado de Apresentação da Cena *Review*

6.1.2.4 Atividade 6 – Configurar Produto Final

A última atividade *Configurar Produto Final* é realizada na etapa de engenharia de aplicação por meio da ferramenta MulSeMaker que disponibiliza um conjunto de formulários de aplicações (*Wizards*) por meio dos quais o usuário especialista de domínio (neste caso poderiam ser os próprios professores que preparam as aulas multissensoriais) preenche as informações pendentes dos *templates*. A Figura 48 apresenta a Ferramenta MulSeMaker utilizando o *Wizard* da Família *Multisensory Education*. É possível observar a tela de Seleção das *Features* (Figura 48a) por meio da qual são definidas as características que estarão presentes na aplicação final gerada. Por exemplo, será definida se a aula será composta apenas por um vídeo multissensorial, se existirá uma cena de revisão e se haverá um *Quiz* para avaliar os conhecimentos obtidos. Também é possível observar a tela para

inserção das informações relacionadas ao *Quiz* em que o engenheiro de aplicação informa a pergunta, alternativas e resposta correta (Figura 48b). Por fim, a tela seguinte (Figura 48c) permite a inserção dos efeitos sensoriais que serão executados de forma condicionada ao resultado obtido no *Quiz*, ou seja, efeitos sensoriais diferentes serão executados de acordo com a nota obtida.

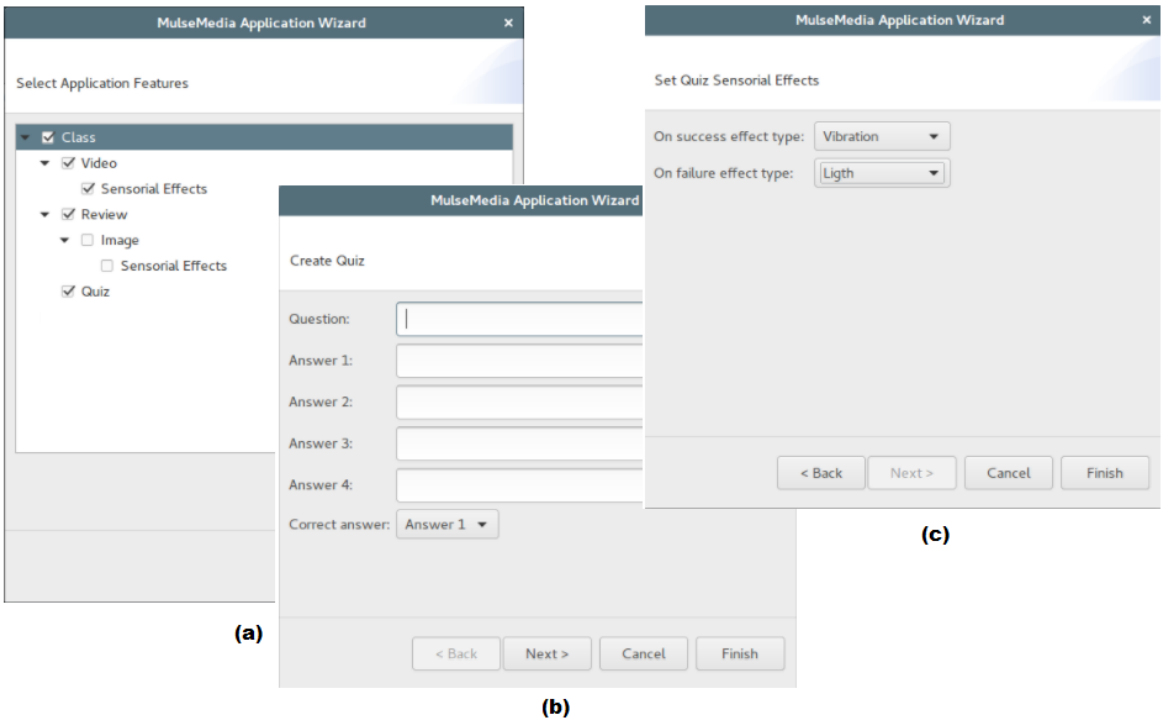


Figura 47: Ferramenta MulSeMaker com *Wizard* da Família *Multisensory Education* – (a) Tela de Seleção das *Features*; (b) Tela para criação do *Quiz*; (c) Tela para inserção dos efeitos sensoriais vinculados ao resultado do *Quiz*

6.2 Avaliação com estudos empíricos

Segundo Wohlin *et al.* (2000), o método empírico utiliza estudos primários baseados em evidências para avaliar modelos propostos, sendo conveniente para obter resultados objetivos e estatisticamente significativos para entender, controlar, prever e melhorar processos de desenvolvimento de software. Um estudo empírico pode ser realizado por meio de três estratégias diferentes:

- ***surveys***: realizados com o objetivo de avaliar e analisar uma população, da qual uma amostra é retirada (BABBIE, 1990). Trata-se de uma avaliação qualitativa que ocorre, geralmente, através de entrevistas, questionários e formulários.
- **estudos de caso**: executados para investigar um fenômeno em um intervalo de tempo específico e em ambiente real. Este fenômeno pode ser a adoção de uma metodol ogia, ferramenta ou tecnologia da Engenharia de Software.

- **experimentos**: empregados com o intuito de verificar uma relação causa-efeito em ambientes controlados. Em resumo, um experimento coleta métricas para comparar duas ou mais causas ao mesmo tempo em que controla variáveis que podem influenciar o efeito. Vale salientar que os resultados de cada efeito são expressos em termos de nível de significância definidos por análise estatística, de modo a evitar conclusões geradas por situações aleatórias.

Neste trabalho são realizados estudos empíricos a fim de avaliar o impacto da adoção de uma abordagem de desenvolvimento orientado a modelos no domínio de aplicações mulsemedia. Para tanto, em um primeiro momento os sujeitos envolvidos, que são caracterizados adiante, participam de um experimento controlado (avaliação quantitativa), seguido da aplicação de questionário (avaliação qualitativa). Foi adotada a estruturação dos elementos do experimento proposta por Wohlin *et al.* (2000) que divide o processo em 5 etapas: (i) definição; (ii) planejamento; (iii) operação do experimento; (iv) análise e interpretação dos resultados; e (v) apresentação dos resultados. Os experimentos também terão sua validade avaliada através de quatro tipos de ameaças: (i) conclusão, (ii) construção, (iii) interna e (iv) externa. Nas próximas seções, tanto a estruturação do experimento quanto a avaliação da sua validade são apresentadas detalhadamente.

6.2.1 Definição dos experimentos.

Para avaliar o impacto do uso da abordagem MDD no desenvolvimento de aplicações mulsemedia foram conduzidos estudos empíricos: um experimento controlado (dados quantitativos) seguido de um questionário (dados qualitativos), ou seja, um estudo realizado com dois instrumentos. Esse estudo foi aplicado com mais de um grupo de sujeitos, conforme será mais detalhado adiante.

Para a definição dos experimentos realizados foi utilizado o paradigma GQM (do inglês, *Goal-Question Metric*) (BASILI; CALDIERA; ROMBACH, 1994). Neste paradigma, os objetivos da avaliação devem ser rastreados a um conjunto de dados que os definem de forma operacional, sendo este rastreamento realizado por meio de questões que caracterizam os objetivos de forma específica. Assim, seguindo o formato GQM, são definidos dois objetivos para esta avaliação que estão descritos por meio das respectivas questões específicas:

G1. **O objeto de estudo** é caracterizado por abordagens de desenvolvimento de aplicações mulsemedia que tratam a integração entre projetos de mídia, software e efeitos sensoriais, **com o propósito** de comparar as abordagens com foco no tempo de implementação de aplicações, **do ponto de vista** do desenvolvedor de aplicações Mulsemedia**, no contexto** do Desenvolvimento Orientado a Modelos no domínio específico de aplicações Mulsemedia.

Q1. O uso de abordagens distintas traz medidas diferentes de tempo no desenvolvimento de aplicações Mulsemedia?

M1. O tempo de implementação de uma aplicação tendo como unidade de medida os segundos.

G2. **O objeto de estudo** é caracterizado por abordagens de desenvolvimento de aplicações Mulsemedia que tratam a integração entre projetos de mídia, software e efeitos sensoriais, **com o propósito** de determinar os benefícios obtidos e dificuldades de utilização, **do ponto de vista** do desenvolvedor de aplicações Mulsemedia**, no contexto** do Desenvolvimento Orientado a Modelos no domínio específico de aplicações Mulsemedia.

Q2. Os sujeitos que utilizaram a abordagem MDD perceberam, durante as atividades referentes à engenharia de aplicação, algum benefício no desenvolvimento das aplicações mulsemedia quando comparado ao desenvolvimento sem a abordagem MDD?

M2. Questionário.

6.2.2 Seleção de Contexto

Em relação à seleção de contexto (WOHLIN *et al.*, 2000), os experimentos foram realizados em ambiente de laboratório por ser considerado mais adequado em relação ao controle. Para representar o contexto do estudo foram utilizados estudantes de graduação e pós-graduação representando os desenvolvedores de aplicações mulsemedia por dois motivos: (i) pela facilidade de acesso que permitiu a realização de treinamentos que uniformizaram o conhecimento sobre o domínio mulsemedia, as linguagens e ferramentas utilizadas; (ii) como a mulsemedia é uma área nova, a existência de profissionais com conhecimento sobre o tema é bastante restrita, reforçando mais uma vez a necessidade de utilizar estudantes com disponibilidade para treinamento sobre o domínio em estudo; (iii) com os treinamentos realizados, acredita-se que foi possível obter uma aproximação mais adequada dos participantes do experimento ao perfil do publico final que fará uso da ferramenta MulSeMaker, que são profissionais da área de multimídia que futuramente utilizarão as ferramentas desenvolvidas para criar esse novo tipo de aplicação. Dessa forma, no contexto desta Tese, desenvolvedores de aplicações mulsemedia são estudantes com conhecimento em tecnologias Web e Multimídia, que passaram por treinamento sobre Mulsemedia. O perfil desses estudantes foi levantado por meio de questionário e está detalhando no Apêndice A. Acerca dos problemas abordados, foram escolhidas duas aplicações-problema que possuem requisito de integração com lógica de aplicação imperativa, contudo, no primeiro experimento foram utilizadas aplicações diferentes conforme será abordado mais adiante.

6.2.3 Variáveis

Conforme já mencionado, o tempo de implementação representa a métrica de produtividade de desenvolvimento da aplicação-problema e por isso é a **variável dependente**, ou seja, variável resposta do experimento. Vale salientar que as medidas de tempo de implementação em todos os experimentos devem considerar apenas as fases da Engenharia de Aplicação, não considerando os esforços realizados na etapa de engenharia de domínio. Em relação às **variáveis independentes**, ou seja, aquelas que definem como empregar as abordagens, foram definidas:

- implementação das aplicações mulsemedia (aplicação-problema), que são melhor caracterizadas adiante;
- especificação dos requisitos da aplicação-problema por meio de *storyboards* e *checklist*;
- implementação das aplicações-problema pertencentes ao domínio de aplicações mulsemedia de modo iterativo e incremental com aumento da complexidade a cada iteração; e
- utilização de ambientes de desenvolvimento de aplicações de aplicações Web (HTML5/JavaScript) para uso da extensão HTML5 proposta, além da utilização de ferramenta de autoria de efeitos sensoriais (Sevino) em aplicações mulsemedia.

6.2.4 Fatores e níveis

Os experimentos possuem um único fator que é a abordagem de desenvolvimento de aplicações mulsemedia. Os níveis, por outro lado, foram definidos com base na abordagem disponível no período da realização dos experimentos, estando a abordagem mais refinada nos últimos experimentos realizados. Nos primeiros experimentos, foram utilizados apenas a parte da abordagem até a geração de efeitos sensoriais com apoio da ferramenta SEVino. Esta ferramenta foi escolhida por ser lidar com autoria de efeitos sensoriais, além de ser uma das mais citadas na literatura e estar disponível para instalação e utilização. Na execução do último experimento, já foi possível utilizar a abordagem até a integração com código imperativo e ferramenta MulSeMaker. Dessa forma, foi possível avaliar o impacto das abordagens não-MDD (M1) e das abordagens MDD (M2 e M3) no tempo de desenvolvimento (TI) das aplicações mulsemedia. Vale salientar que as ferramentas MulSeMaker e Sevino não são comparadas diretamente por possuírem naturezas e objetivos distintos. Assim, em virtude da data de execução, foram definidos 3 (três) níveis diferentes:

- M1: emprego de editores textuais sem apoio de ferramenta de autoria especializada, ou seja, não existe suporte à estruturação de requisitos e à integração entre o projeto de mídia e projeto de software e lógica de programação imperativa;

- M2: emprego da ferramenta SEVino representando o uso de uma abordagem MDD, mas sem qualquer suporte à estruturação de requisitos e a integração entre os efeitos sensoriais com o projeto de software e lógica de programação imperativa;
- M3: emprego da ferramenta MulSeMaker que permite a geração rápida de aplicações mulsemedia, representando o uso de uma abordagem MDD e que oferece suporte à estruturação de requisitos; integração entre os efeitos sensoriais com o projeto de software e lógica de programação imperativa;

6.2.5 Hipóteses

Na realização de experimentos é comum definir como hipótese algo que o experimentador deseja investigar, sendo projetados experimentos com o objetivo de testar esta hipótese. Este conceito é conhecido como hipótese nula. Dessa forma, considerando Mx, x= 1, 2, 3 as metodologias utilizadas nos experimentos e a métrica TI (Mx) (tempo de implementação da aplicação-problema), a partir da hipótese geral, foram definidas as seguintes hipóteses nulas e alternativas a fim de contemplar as variáveis dependentes e independentes (WOHLIN *et al.*, 2000):

- **Hipótese Nula** ($H_0 1$): o uso da abordagem MDD não reduz o tempo de desenvolvimento (TI) das aplicações mulsemedia em comparação com a abordagem não-MDD.

$$H_0 1: TI\ (M1) \leq TI(M2)$$
$$H_0 2: TI\ (M1) \leq TI(M3)$$

- **Negação da Hipótese Nula**($H_1 1$): o uso da abordagem MDD reduz o tempo de desenvolvimento (TI) das aplicações mulsemedia em comparação com a abordagem não-MDD.

$$H_1 1: TI\ (M1) > TI(M2)$$
$$H_1 2: TI\ (M1) > TI(M3)$$

Após a execução do experimento, as hipóteses apresentadas são submetidas a testes estatísticos a fim de se obter informações definitivas para rejeitar (ou não) as hipóteses nulas em favor das hipóteses alternativas. Estes testes são considerados a principal fonte de informação para chegar às conclusões finais do estudo experimental.

6.2.6 Objetos Experimentais

Os objetos experimentais são as especificações das aplicações-problema que servem de entrada para a metodologia utilizada nos estudos. Nos experimentos foram utilizadas as seguintes aplicações: *Multisensory Video*; *Multisensory E-Commerce* e *Multisensory Education*, conforme detalhado mais adiante na descrição de cada experimento. As especificações de cada aplicação-problema foram descritas por meio da utilização de *Storyboards* e *checklists* de funcionalidades. Além disso, tendo em vista que não é objetivo do experimento analisar a etapa da produção multimídia, também foram disponibilizados os objetos de mídia (textos, imagens e vídeos) utilizados nas aplicações.

6.2.7 Instrumentação

Seguindo Wohlin *et al.* (2000), a instrumentação dos experimentos realizados neste estudo ocorreu por meio dos seguintes elementos:

- **Objeto:** especificação da aplicação mulsemedia
- **Diretrizes:** *storyboards* utilizados para descrever o desenvolvimento da aplicação-problema; uma *checklist* das funcionalidades a serem desenvolvidas que serviu para verificar o término do desenvolvimento da aplicação-problema; o fornecimento dos objetos de mídia necessários para o desenvolvimento da aplicação-problema; um documento com instruções gerais para os sujeitos do experimento; e, por fim, foram disponibilizados módulos com código-fonte imperativo utilizados na integração com efeitos sensoriais.
- **Ferramentas de suporte:** laboratório de informática composto por máquinas de configurações iguais e com um ambiente necessário para o desenvolvimento e teste das aplicações mulsemedia a serem desenvolvidas. Também foi utilizada uma ferramenta para contabilizar o tempo de implementação de cada aplicação-problema, que será a diferença entre início do experimento e a finalização de todas as funcionalidades (*checklist*) requeridas para a aplicação-problema. Uma atenção especial foi dada à coleta do tempo para evitar ações mal intencionadas e o uso incorreto da ferramenta, sendo as seguintes instruções passadas aos sujeitos que participaram do experimento: (i) pausar a contagem do tempo quando necessário realizar atividades não relacionadas ao desenvolvimento das aplicações-problema; (ii) solicitar permissão dos executores do experimento para pausar e para retomar a contabilidade do tempo de desenvolvimento das aplicações-problema. Toda a instrumentação foi apresentada nos treinamentos realizados.

6.2.8 Projeto Experimental

Em uma configuração que envolve duas metodologias, duas aplicações mulsemedia e N sujeitos, optou-se por usar um plano experimental de quadrado latino (JEDLITSCHKA; PFAHL, 2005), sendo esta decisão também embasada por trabalhos prévios sobre avaliações de metodologias de desenvolvimento em Engenharia de Software (SÁNCHEZ, 2011) e em estudos na área de abordagens de Linhas de Produto de Software (ALMEIDA, 2010; ACCIOLY, 2012; KULESZA, 2013) que também seguiram a mesma estratégia. Para uma melhor compreensão acerca deste plano experimental, a Tabela 1 ilustra uma configuração de quadrado latino seguindo o modelo de "linhas cruzadas com colunas embutidas dentro das réplicas" (SÁNCHEZ, 2011). As metodologias envolvidas, bem como as aplicações-problema e sujeitos são dispostos de forma aleatória da seguinte maneira: os sujeitos são organizados em linhas e as aplicações-problema nas colunas, assim, para cada linha e coluna de uma réplica de um quadrado a abordagem utilizada aparece apenas uma vez e para cada abordagem avaliada um sujeito desenvolve uma aplicação-problema.

Tabela 2: Projeto Experimental com Quadrado Latino

	Aplicação 1	**Aplicação 2**
Sujeito 1	MDD	Não-MDD
Sujeito 2	Não-MDD	MDD

6.2.8.1 Primeiro Experimento

O primeiro experimento realizado teve como objetivo principal obter os primeiros indícios sobre os benefícios alcançados na adoção de uma metodologia MDD para o desenvolvimento de aplicações mulsemedia. Para tanto, foram utilizadas duas abordagens (MDD e não-MDD), sendo cada uma delas utilizada para desenvolver duas aplicações pertencentes à família *Multisensory Video*: (i) *App1* consiste no *trailer* anotado de efeitos sensoriais do filme *Iron Man 2* e (ii) *App2* consiste no *trailer* anotado de efeitos sensoriais do filme *Babylon A.D.*. Considerando o projeto do quadrado latino, um sujeito que implementa uma aplicação *App1* utilizando a abordagem MDD, vai obrigatoriamente, num segundo momento, implementar a aplicação *App2* utilizando a abordagem Não-MDD.

Nesta fase inicial da pesquisa, as extensões HTML5 realizadas ainda não estavam disponíveis. Dessa forma, os efeitos sensoriais foram implementados diretamente por meio da linguagem SEDL. Na abordagem Não-MDD, os participantes utilizaram apenas editores XML para edição textual do código SEDL, sendo o código escrito manualmente sem apoio de ferramenta de autoria especializada. Por outro lado, na abordagem MDD, os sujeitos se utilizaram da ferramenta SEVino para anotar os

efeitos sensoriais nos vídeos. No contexto deste experimento, a ferramenta SEVino representou a metodologia MDD, permitindo a geração automática do código-fonte de um projeto (dos efeitos sensoriais) genérico da categoria de aplicação de vídeos multissensoriais. É importante ressaltar que apesar da complexidade desta família de aplicação ser simples, ela cumpriu bem o propósito deste experimento em obter os primeiros indícios favoráveis ou não a adoção da abordagem MDD para o desenvolvimento de aplicações mulsemedia.

6.2.8.1.1 Participantes

O primeiro experimento contou com 10 (dez) estudantes que participavam de um processo de seleção de estágio para um laboratório de pesquisa da Universidade Federal da Paraíba (UFPB). Os estudantes cursavam entre o quinto e décimo período dos cursos de graduação em Ciência da Computação e Engenharia da Computação da UFPB e de Sistemas de Informação do Instituto de Educação Superior da Paraíba (IESP), ambas localizadas na cidade de João Pessoa. Dentre os sujeitos, aproximadamente 70% alegaram possuir mais de três anos de experiência com programação. Todos os participantes alegaram conhecer a linguagem HTML5. Uma caracterização mais detalhada acerca da experiência, conhecimento de linguagens e tecnologias dominadas pelos sujeitos são apresentados no Apêndice A.

6.2.8.1.2 Execução do Experimento

O experimento foi realizado no dia 27 de Janeiro de 2017. Ele ocorreu em um laboratório especializado da UFPB durante aproximadamente quatro horas em apenas um dia. Primeiramente foi realizado um treinamento de duas horas abordando o conceito de Mulsemedia, o padrão MPEG-V, a linguagem SEDL e a ferramenta SEVino. Durante o treinamento os sujeitos demonstraram facilidade na assimilação do conteúdo em virtude de já estarem familiarizados com o desenvolvimento de aplicativos multimídia.

Nas últimas duas horas, foi realizada a execução do experimento. A definição das 5 (cinco) réplicas dos quadrados latinos foi realizada pela formação das duplas por meio de um sorteio. Cada computador foi numerado e à medida que o estudante entrava no laboratório, ele retirava uma ficha aleatoriamente para definir sua bancada e, consequentemente, sua réplica de quadrado latino. Depois foi sorteado qual seria a primeira aplicação-exemplo a ser desenvolvida, e por fim, o sorteio de que abordagem seria utilizada inicialmente pelo sujeito da primeira linha do quadrado latino (Sujeito 1 da Tabela 1). O aplicativo *TestWatcher* foi utilizado para coletar o tempo de execução (em segundos) de cada participante. O autor deste trabalho, mais dois alunos de pós-graduação da UFPB executaram o experimento.

6.2.8.1.3 Análise dos Dados Quantitativos

Para a análise dos dados coletados foi empregada a ferramenta *R Statistical Software*[26] por meio da qual foram realizadas as computações estatísticas e geração dos gráficos que facilitaram as interpretações dos resultados obtidos. Na Tabela 2 é possível verificar as medidas de tempo coletadas (média e desvio padrão) referentes ao primeiro experimento. Nota-se que o valor médio do tempo de implementação da abordagem MDD representa um ganho de produtividade em relação ao tempo de desenvolvimento na ordem de 3,07, ou seja, a abordagem MDD teve uma diminuição média de aproximadamente 67% no tempo de desenvolvimento. Os resultados obtidos no cálculo do desvio padrão apontam para uma disparidade relacionada ao nível de formação dos participantes.

Tabela 3: Medidas de tempo médio e desvio padrão do primeiro experimento

	MDD	**Não-MDD**
Tempo Médio (segundos)	951,1	2923,1
Tempo Médio (minutos)	15,6	48,6
Desvio Padrão	450	1001

Continuando a análise dos resultados, a Figura 49 apresenta um gráfico *box-plot* que apresenta medidas de quartis, média, mediana e pontos-limite. Observando o gráfico é possível concluir que, no geral, a abordagem MDD tem valores menores para o tempo de implementação quando comparada a abordagem Não-MDD. Os valores das medianas se encontram relativamente distantes.

[26]Disponível em: http://www.r-project.org/

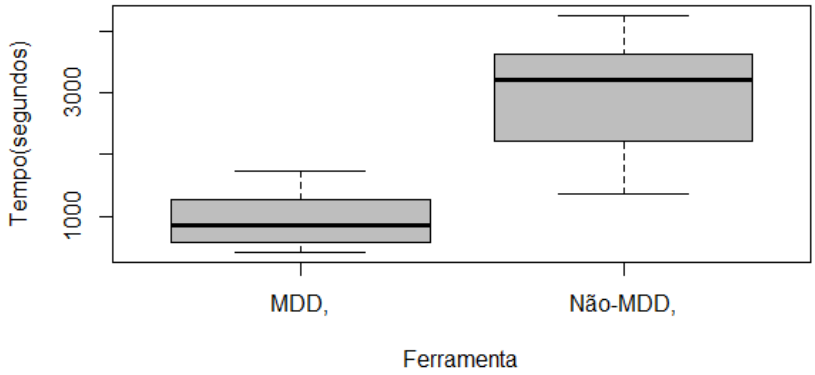


Figura 48: Gráfico *box-plot* do primeiro experimento

Adicionalmente, a Figura 50 mostra que em todos os casos a abordagem MDD obteve um tempo de implementação menor do que a abordagem Não-MDD quando são observados os resultados individuais de cada sujeito.

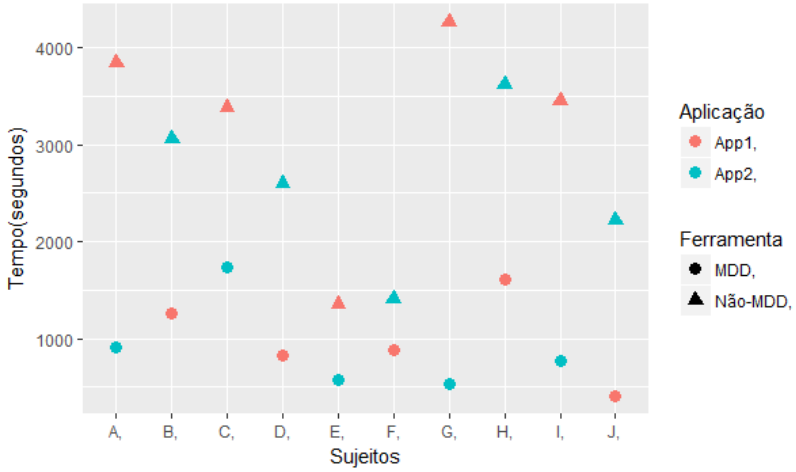


Figura 49: Resultados individuais para cada abordagem

Os dados foram submetidos a duas técnicas: uma técnica de transformação do tipo *Box-Cox* (SAKIA, 1992) e o Teste de Aditividade de *Tukey* (BOX; HUNTER; HUNTER, 2005). O primeiro permite verificar se os resultados obtidos possuem anomalias de não-aditividade e não-normalidade. Esse diagnóstico consiste em avaliar se uma curva gerada pelo método tem seu valor máximo entre 0 e 1. Caso isso ocorra, não se faz necessário realizar transformação nos dados. Para os dados deste primeiro experimento, é possível analisar a curva gerada pelo método na Figura 51 e concluir que não é necessário alterá-los.

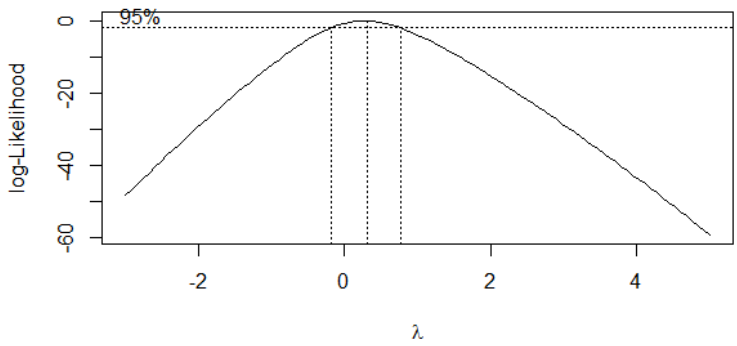


Figura 50: Método Box-Cox para o primeiro experimento

Dando continuidade, o segundo passo é verificar se o modelo de efeitos é aditivo por meio do Teste de Aditividade de *Tukey*. Assim, verificam-se os fatores das colunas e linhas do quadrado latino a fim de checar se eles afetam de maneira significativa as respostas. Levando em consideração o contexto deste experimento, se a variação do tempo de desenvolvimento das duas aplicações-problema escolhidas podem ser comparadas diretamente, este teste possui as seguintes hipóteses:

H_0: o modelo é aditivo

H_1: H_0 é falso

A conclusão foi que o modelo é aditivo, pois o resultado do Teste de Aditividade de *Tukey* neste experimento retornou um valor de p igual a 0.043, não dando evidência significativa para rejeitar a hipótese H_0.

Estando a aditividade assegurada, o próximo passo foi aplicar a análise estatística utilizando ANOVA sobre o modelo para comparar o efeito das duas abordagens no tempo de desenvolvimento das aplicações. Dessa forma, com base no valor destacado de vermelho na Tabela 3 que mostra os resultados da execução do teste estatístico, é possível afirmar que, com um intervalo de confiança de 95% (valor de p < 0,05), podemos rejeitar a hipótese $H_0$1 da Seção, ou seja, a abordagem M1 teve efeito significativo sobre o tempo de desenvolvimento das aplicações em relação à outra abordagem M2. Assim, pode-se concluir que o uso da abordagem MDD reduz o tempo de desenvolvimento (TI) das aplicações mulsemedia em comparação com a abordagem não-MDD.

Tabela 4: Resultado da ANOVA do primeiro experimento

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
replica	4	5218256	1304564	3.077	0.082445
application	1	744208	744208	1.755	0.221779
tool	1	19443920	19443920	45.866	0.000142
replica:subject	5	1491825	298365	0.704	0.636398
Residuals	8	3391437	423930		

6.2.8.1.4 Análise dos Dados Qualitativos

No final do experimento quantitativo os participantes receberam um Questionário que teve por objetivo descobrir as opiniões gerais sobre as duas abordagens utilizadas. Todas as perguntas e respostas podem ser analisadas na integra no Apêndice A. A seguir, um resumo das respostas obtidas:

a) **46,2%** afirmaram que a ferramenta SEVino foi **extremamente útil** em comparação com a programação textual com SEDL e **38,5%** afirmaram que foi **muito útil**;
b) **46,2%** afirmaram que a funcionalidade de *wizards* para escolher e definir as aplicações desenvolvidas a partir de *templates* (modelos pré-definidos) seria **de alguma utilidade** enquanto **23,1%** afirmaram que **seria muito útil** e **15,4%** afirmaram que seria **extremamente útil**
c) **46,2%** afirmaram que seria **mais difícil** integrar efeitos sensoriais no vídeo com lógica de aplicação complexa utilizando o editor textual em comparação com o uso de um editor visual e **15,4%** afirmaram que seria **muito mais difícil**;
d) **53,8%** afirmaram que seria **fácil** implementar a funcionalidade de integração do código imperativo com os efeitos sensoriais utilizando uma extensão da ferramenta SEVino que permitisse configurar essa integração de modo visual.

Em resumo, analisando as respostas dos participantes, (a) deixa bastante clara a preferência dos participantes na utilização de uma ferramenta visual (SEVino) que abstraia a necessidade de codificação por meio de editores textuais. Já (b) demonstra que os participantes não enxergaram como de extrema utilidade a utilização de *templates* em uma nova ferramenta para o desenvolvimento de aplicações Mulsemedia. Acredita-se que a simplicidade das aplicações da família *Multisensory Video* aliada à falta de conhecimento dos participantes sobre MDD colaborou com este resultado, já que este último tema não foi trabalhado no treinamento que antecedeu o experimento. Por outro lado, (c) e (d) corroboram a importância de um suporte ferramental na implementação da funcionalidade de integração de efeitos sensoriais com lógica de programação imperativa, viabilizando assim o desenvolvimento de aplicações mulsemedia mais complexas e de forma simples.

6.2.8.2 Segundo Experimento

O segundo experimento realizado incluiu uma mudança na configuração utilizada anteriormente e teve como objetivo principal analisar o desenvolvimento de aplicações *Web* multissensoriais. Dessa forma, este experimento representa uma primeira avaliação da abordagem MDD proposta. As aplicações-problema (*App1* e *App2*) utilizadas neste experimento são apresentadas na Figura 52, onde é possível observar 4 (quatro) telas das aplicações desenvolvidas identificadas por (a), (b), (c) e (d). As telas (a) e (b) são de uma aplicação *Web/Mobile* de aula multissensorial sobre

“Fontes de Energia”, sendo uma aplicação pertencente à família *Multisensory Education*. Já as telas (c) e (d) são de uma aplicação *Web/Mobile* para venda de ingressos de sessão de cinema, sendo uma aplicação pertencente à família *Multisensory E-Commerce*.

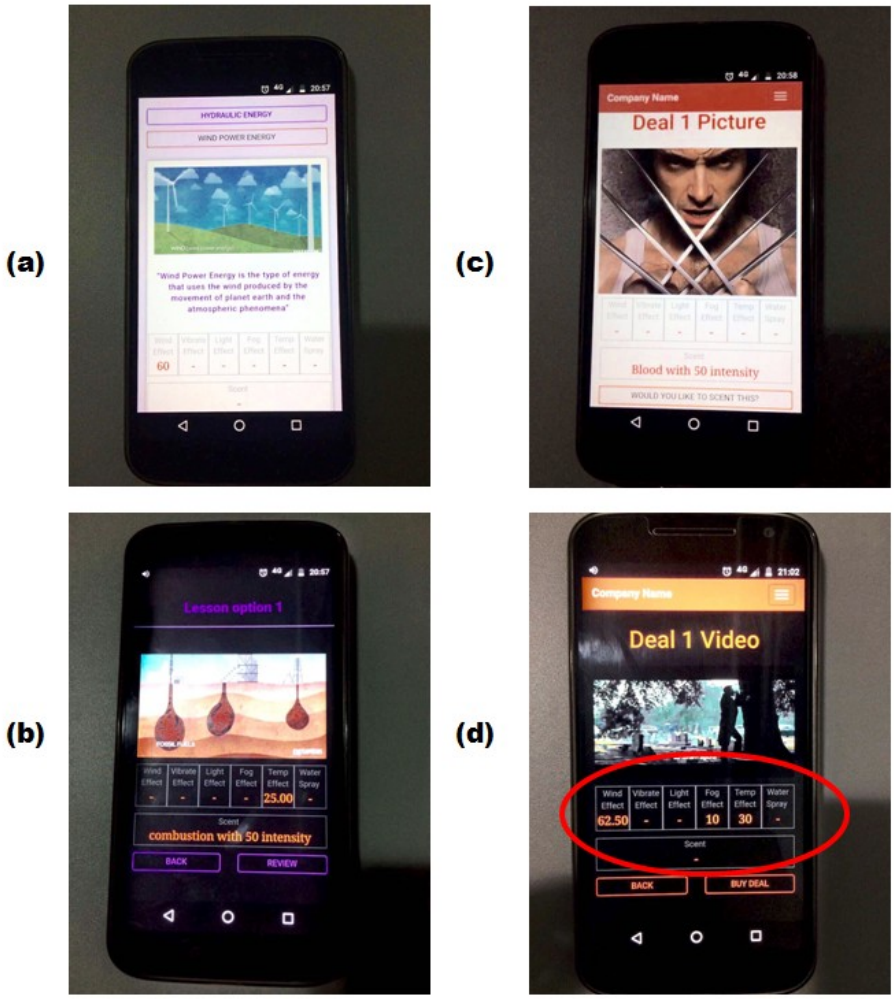


Figura 51: Aplicações *Web/Mobile Multisensory Education* (a, b) e *Multisensory E-Commerce* (c, d)

A fim de evitar viés, as aplicações (variáveis independentes) foram definidas seguindo critérios para garantir a equivalência na complexidade de desenvolvimento, conforme pode ser observado na Tabela 4. Para cada aplicação foram implementadas a mesma quantidade de telas, bem como foram utilizadas a mesma quantidade de mídias e efeitos sensoriais no experimento. Como ambiente de execução para os aplicativos desenvolvidos, foi utilizado um *smarphone* Moto G4 com Android 6.0. Os efeitos sensoriais de vibração e luz foram renderizados por meio dos atuadores *vibracall* e o *flash* da câmera, respectivamente. Os demais efeitos foram simulados por meio da tag `<mulsemedia-box>`, conforme pode ser observado de forma destacada na tela (d) da Figura 52.

Tabela 5: Equivalência na complexidade de desenvolvimento

	Multisensory E-Commerce	**Multisensory Education**
Imagem	*Banner* Multissensorial ilustrado na Figura 4 (c)	Revisão Multissensorial ilustrada na Figura 4 (a)
Vídeo	Vídeo do *trailer* do filme com efeitos sensoriais ilustrado na Figura 4 (d)	Vídeo de aula com efeitos sensoriais ilustrado na Figura 4 (b)
Serviço Web	Transação financeira em que o sucesso influencia os efeitos sensoriais executados	*Quiz* sobre o conteúdo apresentado na vídeo-aula em que o resultado influencia os efeitos sensoriais executados

6.2.8.2.1 Participantes

O segundo experimento contou com 9 (nove) estudantes cursando entre o quinto e décimo período dos cursos de graduação em Ciência da Computação e Engenharia da Computação da UFPB e de Sistemas de Informação do Instituto de Educação Superior da Paraíba (IESP). Dentre os dados coletados destes participantes, 3 (três) amostras foram descartadas por terem cometido algum erro na coleta. Assim, apenas os dados coletados de 6 (seis) estudantes foram utilizados nos testes estatísticos. Dentre os sujeitos, 44,4% alegaram possuir mais de três anos de experiência com programação e 33,3% alegaram possuir mais de dois anos. Neste caso, todos conheciam a linguagem HTML, sendo que apenas 44,4% estavam bem familiarizados com HTML5.

6.2.8.2.2 Execução do Experimento

O segundo experimento foi realizado no dia 28 de Abril de 2017. Ele ocorreu em um laboratório especializado da UFPB durante quatro horas em apenas um dia. Primeiramente foi realizado um treinamento de duas horas abordando a metodologia MDD proposta (extensão do HTML5 com novas *tags* sensoriais e a ferramenta MDD MulSeMaker[27]). Durante o treinamento os sujeitos demonstraram facilidade na assimilação do conteúdo em virtude de já estarem familiarizados com o desenvolvimento *Web*. Aproximadamente nas últimas duas horas foi realizada a execução do experimento. A definição das 3 (três) réplicas dos quadrados latinos se deu conforme já mencionado no primeiro experimento.

[27] Disponível em: https://goo.gl/Ma8TJm

6.2.8.2.3 Análise dos Dados Quantitativos

Seguindo os mesmos passos do primeiro experimento, na Tabela 5 é possível verificar as medidas de tempo coletadas (média e desvio padrão) referentes ao segundo experimento. Nota-se que o valor médio do tempo de implementação da abordagem MDD representa um ganho de produtividade em relação ao tempo de desenvolvimento na ordem de 2,77, ou seja, a abordagem MDD teve uma diminuição média de aproximadamente 64% no tempo de desenvolvimento. Interessante notar que apesar das aplicações envolvidas neste segundo experimento serem consideravelmente mais complexas, o ganho na produtividade (relacionado ao tempo de desenvolvimento) na utilização da abordagem MDD com a ferramenta MulSeMaker foi muito próximo ao do primeiro experimento. Os valores do desvio padrão estão ligeiramente menores em relação ao primeiro experimento.

Tabela 6: Medidas de tempo médio e desvio padrão do segundo experimento

	MDD	**Não-MDD**
Tempo Médio (segundos)	1.395	3.866
Tempo Médio (minutos)	23,24	64,43
Desvio Padrão	360	698

Continuando a análise dos resultados, a Figura 53 apresenta um gráfico *box-plot* por meio do qual é possível concluir mais uma vez que, no geral, a abordagem MDD tem valores menores para o tempo de implementação quando comparada a abordagem Não-MDD. Os valores das medianas permanecem relativamente distantes. Vale ressaltar que na abordagem MDD os valores de medição de tempo se manter mais próximos da mediana quando comparado ao gráfico da abordagem Não-MDD. Acredita-se que esse fato é um indício de que a abordagem MDD favorece aos usuários menos experientes.

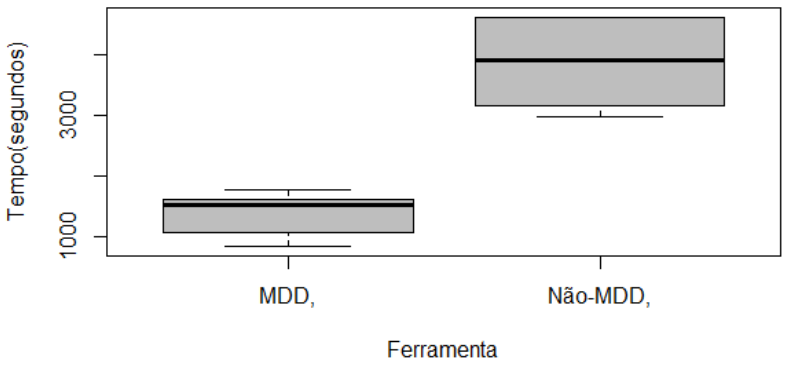


Figura 52: Gráfico *box-plot* do segundo experimento

Adicionalmente, também foram observados os resultados individuais de cada sujeito. A Figura 54 mostra que em todos os casos a abordagem MDD obteve um tempo de implementação menor do que a abordagem Não-MDD. Também é notória a maior homogeneidade dos tempos coletados neste segundo experimento, atestando um grau de formação mais próximo dos participantes, dado este que corrobora com os menores valores de desvio padrão obtidos.

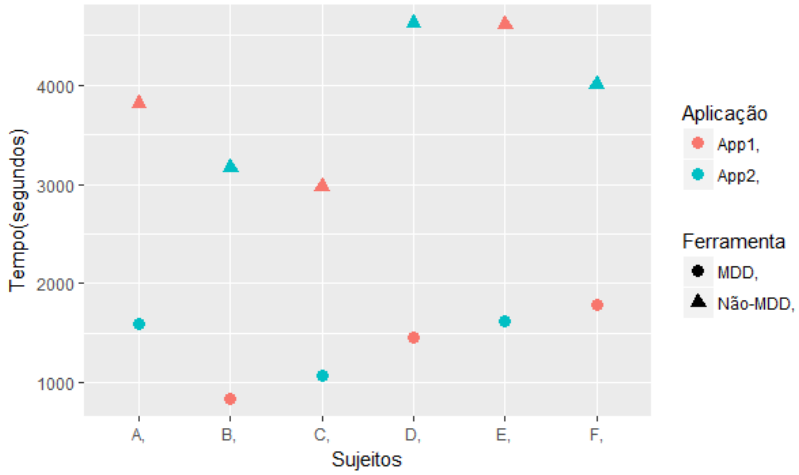


Figura 53: Resultados individuais para cada abordagem do segundo experimento

Seguindo para a etapa dos testes estatísticos, o resultado da transformação *Box-Cox* (Figura 55) também comprovou a aditividade dos dados e o Teste de Aditividade de *Tukey* obteve valor de 0.164 também corroborando na evidência da aditividade do modelo.

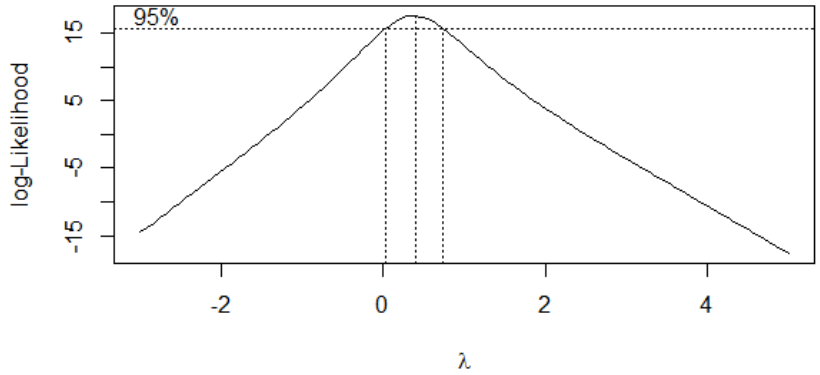


Figura 54: Método Box-Cox para o segundo experimento

Por fim, a Tabela 6 mostra os resultados da execução da análise estatística utilizando ANOVA e, mais uma vez, é possível afirmar que, com um intervalo de confiança de 95% (valor de p < 0,05), podemos rejeitar a hipótese $H_0 2$ da Seção, ou seja, a abordagem M1 teve efeito significativo sobre o tempo de desenvolvimento das aplicações em relação à outra abordagem M3. Dessa forma, pode-se concluir mais uma vez que o uso da abordagem MDD reduz o tempo de desenvolvimento (TI) das aplicações mulsemedia em comparação com a abordagem não-MDD

Tabela 7: Resultado ANOVA do segundo experimento

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
replica	2	903411	451705	3.079	0.155038
application	1	29601	29601	0.202	0.676537
tool	1	18317523	18317523	124.875	0.000365
replica:subject	3	1565094	521698	3.557	0.126005
Residuals	4	586747	146687		

6.2.8.2.4 Análise dos Dados Qualitativos

Novamente, no final do experimento quantitativo os participantes receberam um Questionário que teve por objetivo descobrir as opiniões gerais sobre as duas abordagens utilizadas. Todas as perguntas e respostas podem ser analisadas na integra no Apêndice A. A seguir, um resumo das respostas obtidas:

a) **66,7%** afirmaram que a ferramenta MDD foi **extremamente útil** em comparação com a programação textual com HTML5/JavaScript e **11,1%** afirmaram que foi **muito útil**;
b) **44,4%** afirmaram sobre o uso de *Wizards* para escolher e definir as funcionalidades das aplicações de modelos pré-definidos na ferramenta de geração de código que foi **muito útil** e **22,2%** afirmou que foi **extremamente útil**;
c) **55,6%** afirmaram que a funcionalidade de integração de efeitos sensoriais à lógica complexa de aplicação em javascript foi **muito fácil** de implementar e **22,2%** afirmaram que foi **fácil**;

d) **66,7%** afirmaram que integrar efeitos sensoriais a cenas usando código JavaScript foi **mais difícil** de se realizar quando comparado a ferramenta MDD proposta e **11,1%** afirmaram que foi **muito mais difícil**.

6.2.8.3 Terceiro Experimento

O terceiro experimento realizado manteve a configuração utilizada anteriormente e também teve como objetivo principal analisar o desenvolvimento de aplicações *Web* multissensoriais. Dessa forma, as aplicações-problema (*App1* e *App2*) utilizadas neste experimento são as mesmas do segundo experimento, sendo os sujeitos envolvidos diferentes dos dois primeiros experimentos. A intenção foi avaliar mais exaustivamente a abordagem com um número maior de sujeitos com perfis diferentes.

6.2.8.3.1 Participantes

O terceiro experimento contou com 8 (oito) estudantes cursando a disciplina Tópicos Especiais em Desenvolvimento Web do curso de graduação em Ciência da Computação da UFPB. Dentre os dados coletados destes participantes, 2 (duas) amostras foram descartadas por terem cometido algum erro na coleta. Assim, apenas os dados coletados de 6 estudantes foram utilizados nos testes estatísticos. Dentre os sujeitos, 62,5% alegaram possuir mais de três anos de experiência com programação e 12,5 % alegaram possuir mais de dois anos. Neste caso, 87,5% conheciam bem a linguagem HTML, sendo que apenas 62,5% estavam bem familiarizados com HTML5.

6.2.8.3.2 Execução do Experimento

O terceiro experimento foi realizado no dia 8 de Setembro de 2017. De forma similar ao experimento anterior, este ocorreu em um laboratório especializado da UFPB durante quatro horas em apenas um dia. Também da mesma forma que o segundo experimento, foi realizado um treinamento de duas primeiras horas, e a execução do experimento ocorreu durante as duas últimas horas. A definição das 3 (três) réplicas dos quadrados latinos se deu conforme já mencionado anteriormente.

6.2.8.3.3 Análise dos Dados Quantitativos

Seguindo os mesmos passos, na Tabela 7 é possível verificar as medidas de tempo coletadas (média e desvio padrão) referentes ao segundo experimento. É possível notar que o valor médio do tempo de implementação da abordagem MDD representa um ganho de produtividade em relação ao tempo de desenvolvimento na ordem de 2,19, ou seja, a abordagem MDD teve uma diminuição média de aproximadamente 55% no tempo de desenvolvimento. Vale destacar que, em comparação ao

segundo experimento, o ganho na produtividade (relacionado ao tempo de desenvolvimento) na utilização da abordagem MDD com a ferramenta MulSeMaker foi menor. Acredita-se que esse fato está diretamente relacionando com o perfil dos participantes deste terceiro experimento. De acordo com o levantamento realizado, 62,5% dos sujeitos alegaram possuir mais de três anos de experiência com programação enquanto apenas 44,4% alegaram o mesmo no segundo experimento. Assim, há um indício de que a abordagem MDD é mais adequada a profissionais não-especialistas, sendo a percepção de valor da abordagem proporcional a inexperiência dos sujeitos. Por fim, os valores do desvio padrão estão ligeiramente menores em relação ao primeiro experimento.

Tabela 8: Medidas de tempo médio e desvio padrão do segundo experimento

	MDD	**Não-MDD**
Tempo Médio (segundos)	1.291	2.828
Tempo Médio (minutos)	21,25	47,13
Desvio Padrão	301	687

Continuando a análise dos resultados, a Figura 56 apresenta um gráfico *box-plot* por meio do qual é possível concluir mais uma vez que, no geral, assim como nos experimentos anteriores, a abordagem MDD tem valores menores para o tempo de implementação quando comparada a abordagem Não-MDD. Os valores das medianas permanecem relativamente distantes.

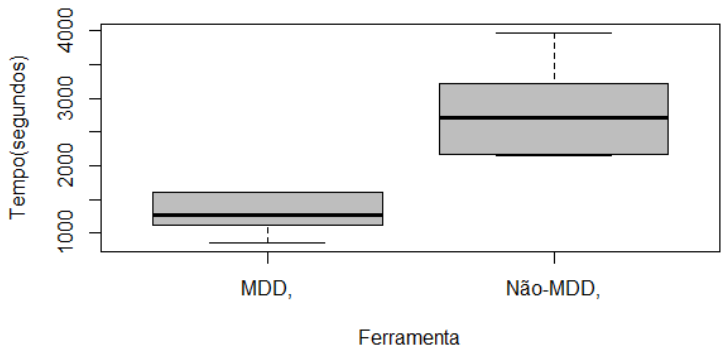


Figura 55: Gráfico *box-plot* do segundo experimento

Adicionalmente, a Figura 57 mostra que em todos os casos, analisando os resultados individuais de cada sujeito, a abordagem MDD obteve um tempo de implementação menor do que a abordagem Não-MDD.

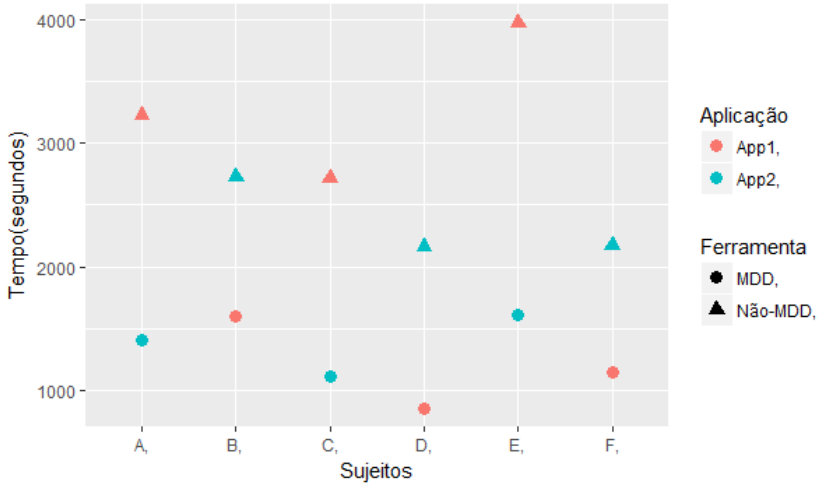


Figura 56: Resultados individuais para cada abordagem do segundo experimento

Seguindo para a etapa dos testes estatísticos, o resultado da transformação *Box-Cox* (Figura 58) também comprovou a aditividade dos dados e o Teste de Aditividade de *Tukey* obteve valor de 0.117 também corroborando na evidência da aditividade do modelo.

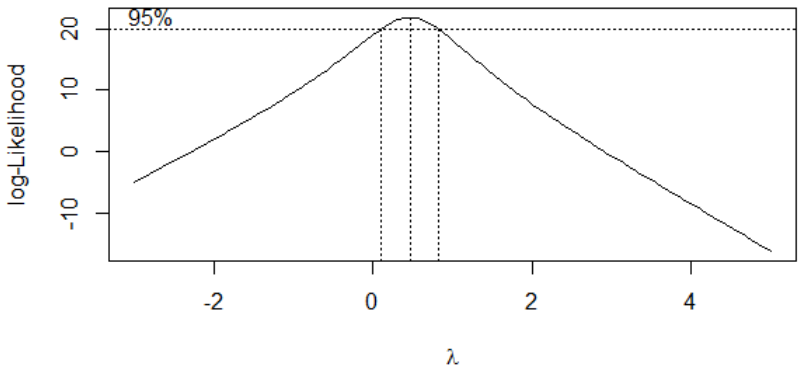


Figura 57: Método Box-Cox para o segundo experimento

Por fim, a Tabela 8 mostra os resultados da execução da análise estatística utilizando ANOVA e, mais uma vez, é possível afirmar que, com um intervalo de confiança de 95% (valor de p < 0,05), podemos rejeitar a hipótese $H_0 2$, ou seja, a abordagem M1 teve efeito significativo sobre o tempo de desenvolvimento das aplicações em relação à outra abordagem M3. Dessa forma, é possível concluir mais uma vez que o uso da abordagem MDD reduz o tempo de desenvolvimento (TI) das aplicações mulsemedia em comparação com a abordagem não-MDD

Tabela 9: Resultado ANOVA do segundo experimento

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
replica	2	720161	360081	8.406	0.03694	*
application	1	447760	447760	10.453	0.03188	*
tool	1	7090181	7090181	165.518	0.00021	**
replica:subject	3	1470936	490312	11.446	0.01969	*
Residuals	4	171345	42836			

6.2.8.3.4 Análise dos Dados Qualitativos

Novamente, no final do experimento quantitativo os participantes receberam um Questionário que teve por objetivo descobrir as opiniões gerais sobre as duas abordagens utilizadas. Todas as perguntas e respostas podem ser analisadas na integra no Apêndice A. A seguir, um resumo das respostas obtidas:

e) **37.5%** afirmaram que a ferramenta MDD foi **extremamente útil** em comparação com a programação textual com HTML5/JavaScript e **62,5%** afirmaram que foi **muito útil**;
f) **62,5%** afirmaram sobre o uso de *Wizards* para escolher e definir as funcionalidades das aplicações de modelos pré-definidos na ferramenta de geração de código que foi **muito útil** e **37,5%** afirmaram que foi **extremamente útil**;
g) **37,5%** afirmaram que a funcionalidade de integração de efeitos sensoriais à lógica complexa de aplicação em javascript foi **muito fácil** de implementar e **62,5%** afirmaram que foi **fácil**;
h) **37,5%** afirmaram que integrar efeitos sensoriais a cenas usando código JavaScript foi **mais difícil** de realizar quando comparado a ferramenta MDD proposta e **25%** afirmaram que foi **muito mais difícil**.

6.2.9 Ameaças a Validade

Nesta Seção são descritos os elementos que podem ameaçar a validade dos experimentos, categorizados de acordo com (WOHLIN *et. al.*, 2012) em ameaças: de conclusão, interna, de construção e externa.

Validade de Conclusão

A ameaça à validade de conclusão se refere ao baixo poder estatístico e a violação de premissas assumidas para o teste estatístico. Para eliminar esta ameaça foi empregado o teste de hipótese, método amplamente sólido e difundido. Complementarmente, a ferramenta R Studio serviu de apoio à análise de dados, minimizando a possibilidade de erros associados ao seu tratamento manual. Já sobre os testes estatísticos utilizados, inicialmente foram aplicadas as técnicas de transformação *Box-Cox* e aditividade de *Tukey* para garantir que os resultados obtidos não possuíam anomalias de não-

normalidade e não-aditividade, respectivamente. Uma vez garantida a aditividade, a ANOVA foi aplicada sobre o modelo para comparar o efeito das duas abordagens (tratamento) no tempo de desenvolvimento das aplicações (variável resposta).

Também é apontado como ameaça de validade de conclusão o aspecto social do experimento. Os comportamentos dos sujeitos envolvidos podem ser guiados para aumentar a chance de rejeitar (ou não) a hipótese nula de acordo com sua preferência pessoal em relação a alguma metodologia. Para evitar tal ameaça os sujeitos não tomaram conhecimento dos objetivos e as hipóteses do experimento.

Validade Interna

Um dos elementos que podem introduzir uma ameaça à validade interna é a instrumentação inadequada para a realização do experimento. Neste caso, houve uma preocupação relevante com a instrumentação, sendo realizado treinamento com os participantes sobre a devida utilização. Outro ponto importante é o controle do experimento. É importante que o uso das ferramentas não seja influenciado pelo seu ambiente de execução. Por fim, outros aspectos que foram controlados para não prejudicar a validade interna foram: (i) a atuação dos sujeitos por meio do uso de uma documentação que guiaria cada etapa da execução do experimento e, adicionalmente, antes da execução final, foi realizada uma atividade de familiarização com o ambiente de execução do experimento; (ii) o objeto do experimento através de documentação detalhada da sua especificação e da disponibilização dos seus elementos de mídia.

Validade de Construção

Neste ponto, os elementos do projeto experimental são a principal ameaça, como o tamanho das amostras utilizadas. Neste ponto, foi definida uma quantidade de réplicas de experimentos maior que a quantidade definida no tamanho das amostras do projeto experimental, de forma a obter significância estatística dos dados. Além disso, as análises estudadas consideraram um nível de confiança de 95%, possibilitando uma probabilidade satisfatória relacionada às zonas de rejeição das hipóteses nulas (JAIN, 1991). Outra importante ameaça considerada diz respeito aos testes estatísticos que foram utilizados. Neste caso, o autor do experimento se aprofundou em assuntos de aspectos estatísticos, além consultar outros pesquisadores na área de experimentação em engenharia de software para revisar o projeto do experimento.

Validade Externa

A validade externa diz respeito à capacidade de generalizar os resultados obtidos em determinado experimento controlado para situações além do escopo do mesmo. A validade externa deste experimento está relacionada aos perfis dos sujeitos, o objeto do experimento utilizado, a instrumentação e o projeto experimental elaborados no estudo, pois estes elementos em conjunto podem prejudicar a generalização dos resultados. Mais precisamente, as principais ameaças no contexto deste trabalho são os sujeitos e o objeto de estudo. No caso dos sujeitos, a mulsemedia ainda não é uma área difundida e, dessa forma, profissionais com conhecimento no assunto são escassos. Diante disso, apesar de não representar precisamente os profissionais da área de desenvolvimento de aplicações mulsemedia, foram utilizados alunos de graduação e de pós-graduação como sujeitos dos estudos empíricos realizados. Para aproximar o perfil dos sujeitos ao dos desenvolvedores de aplicações mulsemedia foram realizados treinamentos. Já em relação ao uso de apenas um objeto, tal característica pode prejudicar a generalidade do experimento, pois é possível que outras categorias de aplicações com requisitos diferentes apresentem resultados diferentes para as metodologias. Contudo, acredita-se que para o escopo definido das aplicações mulsemedia na *Web*, as características presentes (como vídeos, imagens e textos multissensoriais) utilizadas nas famílias implementadas também estarão presentes em boa parte das demais famílias que possam vir a ser criadas.

6.3 Considerações sobre o Capítulo

Este Capítulo relatou as avaliações empíricas relacionadas à tese defendida neste trabalho de que o uso de uma abordagem de desenvolvimento orientado a modelos aumenta a produtividade, em relação a tempo de implementação, do desenvolvimento de aplicações mulsemedia quando comparada com abordagens que não utilizam o MDD. Para tanto, foram desenvolvidas duas provas de conceito exploratórias e estudos empíricos. Mais precisamente, foram realizados dois experimentos controlados seguidos de avaliação qualitativa por meio de questionário, todos realizados em ambiente acadêmico. Em relação aos estudos exploratórios, foi possível concluir que a sistematização da aplicação da abordagem em diferentes famílias de aplicação foi possível ser realizada de forma satisfatória. Já os estudos empíricos objetivaram verificar os efeitos da utilização da abordagem MDD no desenvolvimento de aplicações mulsemedia quando comparado a outras abordagens.

Acerca dos resultados obtidos por meio dos estudos empíricos realizados, vale destacar que no primeiro experimento o ganho de produtividade em relação ao tempo de desenvolvimento foi na ordem de 3,07, ou seja, a abordagem MDD teve uma diminuição média de aproximadamente 67% no tempo de desenvolvimento quando comparada à abordagem Não-MDD. Já no segundo experimento, apesar de terem sido desenvolvidas aplicações mais complexas, inclusive possuindo integração de efeitos sensoriais com lógica imperativa, o ganho de produtividade se manteve muito próximo ao obtido no primeiro experimento, tendo sido na ordem de 2,77, ou seja, uma diminuição média de aproximadamente 64% no tempo de desenvolvimento. O *feedback* dos participantes foi coletado por

meio de questionário ao fim dos experimentos, e evidenciou a preferência dos participantes pela utilização da abordagem MDD. No primeiro experimento, **46,2% afirmaram** que a ferramenta Sevino (abordagem MDD) foi extremamente útil em comparação com a programação textual com SEDL. Já no segundo experimento, **75% afirmaram** que a ferramenta MDD foi extremamente útil em comparação com a programação textual com HTML5/JavaScript. Acredita-se que o percentual bem maior no segundo experimento foi em virtude da maior complexidade das aplicações desenvolvidas. Acerca do terceiro e último experimento realizado, os resultados apontaram uma relação inversamente proporcional do perfil dos sujeitos com a produtividade (relacionada ao Tempo de Implementação), ou seja, a abordagem MDD demonstrou-se mais eficaz com usuários com menor experiência, sendo provavelmente mais adequada para usuários não-especialistas. Mais estudos neste sentido devem ser realizados em trabalhos futuros a fim de investigar os indícios obtidos nos experimentos até aqui realizados. Mais uma vez, é importante deixar claro que apenas a execução da Engenharia de Aplicação da abordagem foi considerada nos estudos empíricos. Os dados levantados nos experimentos estão disponíveis[28].

Duas falhas foram identificadas após a execução dos estudos experimentais. A primeira no processo de definição da posição dos sujeitos no plano experimental, a distribuição não foi totalmente aleatória, uma vez que a ordem de chegada influenciava. Deveria ter sido realizado um sorteio com todos os sujeitos presentes para se definir a posição nos quadrados latinos. Já a segunda está relacionada à coleta do tempo que foi realizada pelos próprios participantes sob a supervisão dos executores do experimento. Essa estratégia é muito vulnerável a falhas como, por exemplo, esquecimento de pausar e despausar o software de coleta do tempo, fato este que ocorreu durante os experimentos e ocasionou a perda de algumas amostras. Um sistema centralizado em que essa gerencia ficasse na responsabilidade dos executores seria mais adequado para evitar tais acontecimentos.

Outro ponto que merece ser destacado é que, embora as duas técnicas de avaliação empregadas representem apenas um subconjunto do que pode ser avaliado, acredita-se que os resultados foram significativos e constituem um ponto inicial no sentido de definir uma metodologia para avaliar e melhorar o desenvolvimento de aplicações mulsemedia.

[28] Disponível em: https://goo.gl/UtYBmK

7 CONCLUSÃO

O presente trabalho apresentou uma abordagem de desenvolvimento orientado a modelos para integrar de modo sistemático as diferentes disciplinas envolvidas no desenvolvimento de aplicações mulsemedia.

O Capítulo 1 contextualizou o domínio de pesquisa e apresentou os seus desafios atuais, que motivaram os objetivos da pesquisa realizada neste trabalho. Também foi estabelecido o escopo da pesquisa, bem como as justificativas para tanto.

O Capítulo 2 apresentou trabalhos que lidam com o desenvolvimento de aplicações mulsemedia por meio de ferramentas de autoria, e estudos que discutem os conceitos e viabilidade de uso do MDD em domínios específicos que influenciaram as decisões e contribuíram com soluções adotadas na abordagem desenvolvida nesta tese.

O Capítulo 3 apresentou os principais conceitos e técnicas do desenvolvimento orientado a modelos, discutiu os pontos positivos e negativos acerca da sua adoção. Até então, o uso do MDD para auxiliar o desenvolvimento de aplicações no domínio mulsemedia permanecia inexplorado. Este trabalhou buscou combinar as abordagens propostas pela OMG-MDA, Modelagem Específica de Domínio e LPS/GSD em uma abordagem nova para contexto específico do desenvolvimento de aplicações mulsemedia.

O Capítulo 4 apresentou o domínio das aplicações mulsemedia, sendo apontadas as principais motivações para o seu desenvolvimento e as áreas de aplicação indicadas na literatura como potenciais beneficiárias. Além disso, também foi mostrada uma visão geral sobre o padrão MPEG-V que permite a interoperabilidade entre o mundo real e virtual. Mais precisamente, uma atenção especial foi dada a parte 3 relacionada à linguagem de descrição de efeitos sensoriais SEDL que define uma representação sistemática de conteúdo relacionado aos efeitos sensoriais. Adicionalmente, foram apresentadas soluções de hardware e software existentes, além de uma discussão acerca de como as linguagens para o desenvolvimento de aplicações multimídia podem ser utilizadas no desenvolvimento de aplicações mulsemedia.

O Capítulo 5 descreveu a abordagem MDD para o desenvolvimento de aplicações mulsemedia proposta neste trabalho. Foi apresentada uma visão geral da abordagem, além do seu detalhamento, que foi realizado por meio da concretização da abordagem em um exemplo de uso tratando a família de aplicações de vídeos multissensoriais: *Multisensory Video*. A abordagem foi organizada com base nas fases de Engenharia de Domínio e Engenharia de Aplicação do GSD propostas por Czarnecki e Eisenecker (2000).

O Capítulo 6 relatou as avaliações quantitativas e qualitativas relacionadas à tese defendida neste trabalho. Para tanto, foram desenvolvidas duas provas de conceitos exploratórias e estudos

empíricos. Mais precisamente, foram realizados dois experimentos controlados seguidos de avaliação qualitativa por meio de questionário, todos realizados em ambiente acadêmico. Em relação aos estudos exploratórios, o intuito foi avaliar a sistematização da aplicação da abordagem em diferentes famílias de aplicação (*Multisensory E-Commerce* e *Multisensory Education*). Já estudos empíricos objetivaram verificar os efeitos da utilização da abordagem MDD no desenvolvimento de aplicações mulsemedia quando comparado a outras abordagens.

7.1 Contribuições

Este trabalho propõe uma abordagem de desenvolvimento orientado a modelos para integrar os projetos de mídia, software e efeitos sensoriais no domínio de aplicações mulsemedia. A intenção foi promover a integração de forma sistemática das diferentes disciplinas e profissionais envolvidos na construção dessas aplicações. Esta tese defendeu que o uso de uma abordagem de desenvolvimento orientado a modelos aumenta a produtividade, em relação a tempo de implementação, do desenvolvimento de aplicações mulsemedia quando comparada com abordagens que não utilizam o MDD. Diante dos esforços realizados ao longo deste trabalho, é possível apontar as seguintes contribuições alcançadas:

- Uma abordagem sistemática que organiza e detalha de forma didática as atividades a serem realizadas para geração dos artefatos envolvidos na concretização da abordagem. Dessa forma, acredita-se que este trabalho trouxe uma contribuição significativa sobre como explorar os benéficos do MDD no domínio de aplicações mulsemedia;
- Extensão do metamodelo MML para suportar a modelagem de efeitos sensoriais;
- Detalhamento das atividades de análise e projeto a fim de definir um método concreto para utilização de forma integrada dos modelos MML em conjunto com um Modelo de *Features* e técnicas de transformação (M2M e M2C);
- Extensão do modelo HTML5 para suportar: (i) o desenvolvimento de forma declarativa de efeitos sensoriais por meio das novas *tags* `<vibration-effect>`, `<wind-effect>`, `<light-effect>`, `<temperature-effect>`, `<scent-effect>`, `<fog-effect>`, `<water-spray>`, e `<mulsemedia-box>`; (ii) a integração de forma declarativa entre efeitos sensoriais e lógica imperativa de programação por meio da *tag* `<conditional-effect>`; (iii) a implementação de soluções de sincronismo entre os efeitos sensoriais e os demais objetos de mídia (vídeo, imagem e texto). Mais precisamente, foram implementadas soluções em aplicações dirigidas por mídia (efeitos sensoriais sincronizados com vídeo na linha do tempo); e também em aplicações dirigidas por eventos (efeitos sensoriais executados em conjunto com imagem e texto por um clique de botão);

- Modelagem e implementação de três famílias de aplicações de mulsemedia. A primeira, *Multisensory Video*, foi utilizada como exemplo de uso no detalhamento da abordagem e como aplicação-problema do primeiro experimento realizado. As outras duas, *Multisensory E-Commerce* e *Multisensory Education*, foram utilizadas como prova de conceito nos estudos exploratórios e como aplicações-problema nos estudos empíricos realizados.
- O Planejamento e a realização de estudos empíricos. Com base nas pesquisas realizadas na literatura, é possível afirmar que este trabalho apresenta a primeira proposta de abordagem de desenvolvimento para o domínio de aplicações mulsemedia. Os estudos empíricos realizados foram fundamentais para avaliar os reais benefícios e limitações da abordagem desenvolvida. Além disso, trazem informações relevantes para a comunidade científica da área de engenharia de software e mulsemedia.

7.2 Trabalhos Futuros

Nesta Seção, algumas limitações e pontos inexplorados durante o desenvolvimento desta tese são listados a fim de receberem contribuições. Assim, os seguintes itens se destacam como oportunidade de trabalhos futuros:

- A especificação de requisitos que ocorre na Análise de Domínio não foi o foco da pesquisa realizada neste trabalho. Dessa forma, alguns pontos não atacados foram: (i) a realização de uma taxonomia de aplicações mulsemedia; (ii) investigação mais aprofundada acerca da dinâmica entre as equipes multidisciplinares envolvidas nesta fase; (iii) avaliação sobre a necessidade de adequação dos artefatos e ferramentas utilizados para especificação dos requisitos. Essas três questões levantadas são importantes para auxiliar na elaboração dos possíveis cenários de uso das categorias de aplicações definidas na Análise de Domínio;
- A abordagem MDD proposta não possui todas as etapas e atividades que podem ocorrer em um processo de desenvolvimento de software. Dessa forma, é possível incluir novas fases e atividades relacionadas a disciplinas não estudadas nesta tese, como aplicar MDD na especificação de requisitos para extraí-los automaticamente, dentre outras possibilidades;
- Realizar novos estudos exploratórios para avaliar a sistematização das atividades na reutilização dos modelos CIM e PIM aqui desenvolvidos na implementação de transformações para outras plataformas específicas (PSM);
- Não existe ainda uma grande quantidade de aplicações mulsemedia desenvolvidas. Esse fato foi um empecilho na definição de diferentes famílias de aplicações. Com o avanço das pesquisas na área, será um trabalho futuro desenvolver novas famílias de aplicação

mulsemedia com base nas áreas apontadas na literatura como potenciais beneficiárias dos efeitos sensoriais (Jogos, Saúde, Entretenimento, etc.);

- Melhorias na interface da ferramenta MulSeMaker por meio da inclusão de um *player* para auxiliar na visualização das mídias, bem como *timelines* para facilitar a inclusão dos efeitos sensoriais, conforme faz o SEVino.
- Considerar na abordagem outros requisitos apontados no padrão MPEG-V, como capacidade dos dispositivos, preferência dos usuários, além da adaptação dos efeitos sensoriais com base nessas informações. Além desses pontos, apesar de modelados, algumas funcionalidades como `GroupOfEffects` não chegaram a serem completamente estendidos no modelo HTML5;
- Replicar os estudos empíricos levando em consideração (i) sujeitos com disponibilidade maior de tempo e perfil não acadêmicos; (ii) inclusão de outras métricas de produtividade de desenvolvimento e (iii) utilização de novas aplicações-problema com diferentes pontos de integração entre efeitos sensoriais e lógica imperativa de programação; e
- Realizar avaliações relacionadas à QoE dos sujeitos ao interagirem com as aplicações mulsemedia desenvolvidas por meio da abordagem proposta nesta Tese.

7.3 Considerações Finais

Este trabalho se concentrou em identificar como aplicar o MDD por meio de uma abordagem de desenvolvimento de software no domínio de aplicações mulsemedia. A fim de investigar o impacto da adoção da abordagem, a principal questão de pesquisa definida no início da pesquisa foi: **O uso de uma abordagem de desenvolvimento orientado a modelos aumenta a produtividade, em relação a tempo de implementação, do desenvolvimento de aplicações mulsemedia quando comparada com abordagens que não utilizam o MDD?** Com base nas investigações realizadas, é possível afirmar que, apesar do esforço inicial em virtude das atividades relacionadas à engenharia de domínio, ou seja, uma vez que o desenvolvimento das famílias de aplicações está realizado, o desenvolvimento de aplicações a partir dos *templates* (modelos MML) se torna mais rápido. Mais precisamente, os experimentos controlados apresentaram ganhos de produtividade na ordem de 3,07 (primeiro experimento), 2,77 (segundo experimento) e 2,21 (terceiro experimento) no tempo de desenvolvimento quando a abordagem MDD foi comparada a abordagem Não-MDD. As avaliações qualitativas também apontaram uma percepção de utilidade da abordagem proposta, tendo 46,2% dos participantes do primeiro experimento, 75% dos participantes do segundo experimento e 37,5% dos participantes do terceiro experimento afirmado que a abordagem MDD foi extremamente útil em comparação a abordagem Não-MDD.

REFERÊNCIAS

ACCIOLY, P. R. G. Comparing Different Testing Strategies for Software Product Lines. 103f. Dissertação (Mestrado), Universidade Federal de Pernambuco, Recife, 2012.

ALMEIDA, R. B., Modeling Software Product Line Variability in Use Case Scenarios — An Approach Based on Crosscutting Mechanisms. 2010. 150f. Tese (Doutorado), Universidade Federal de Pernambuco, Recife, Brasil.

AMBLER, S. W. Agile model driven development is good enough. IEEE Software, v. 20, n. 5, p. 71–73, 2003.

AMBX. Cyborg Gaming Lights. 2017. Disponível em: < http://www.ambx.com/product/cyborg-gaming-lights >. Acesso em: 30.07.2017.

ANTKIEWICZ, M.; CZARNECKI, K. FeaturePlugin: feature modeling plug-in for Eclipse. In: Proceedings of the 2004 OOPSLA workshop on eclipse technology eXchange. ACM, 2004. p. 67-72.

BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. Experience factory. Encyclopedia of software engineering, 1994.

BABBIE, E. R. Survey research methods. 1990. Wadsworth.

BEZIVIN, J.; GERBE, O.Towards a precise definition of the OMG/MDA framework. Proceedings 16th Annual International Conference On Automated Software Engineering (ase 2001), [s.l.], p.272-280, nov. 2001. Institute of Electrical & Electronics Engineers (IEEE). http://dx.doi.org/10.1109/ase.2001.989813.

BOX G. E. P., HUNTER, S. J., HUNTER, W. G. Statistics for experimenters: design, innovation, and discovery. Wiley-Interscience, 2005.

BRAMBILLA, M; CABOT, J; WIMMER, M. Model-driven software engineering in practice. Synthesis Lectures on Software Engineering, v. 2, n. 2, p. 1-192, 2017.

BRANDT, A. Sensory Marketing Is the Next Frontier in Mobile Advertising: Mobile Ads Must Appeal to Users' Emotions through Sight, Sound and Touch. 2015. Disponível em: <http://adage.com/article/digitalnext/sensory-marketing-frontier-mobile-advertising/296655/>. Acesso em: 12.08.2017.

BRAUN, M. H.; CHEOK, A. D. Towards an olfactory computer-dream interface. In: Proceedings of the 11th Conference on Advances in Computer Entertainment Technology. ACM, 2014. p. 54.

BRAUN, M. H.; CHEOK, A. D. Using scent actuation for engaging user experiences. In: Proceedings of the 12th International Conference on Advances in Computer Entertainment Technology. ACM, 2015. p. 54.

BULTERMAN, D. C.; HARDMAN, L. Structured multimedia authoring. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), v. 1, n. 1, p. 89-109, 2005.

CAZENAVE, F.; QUINT, V.; ROISIN, C. Timesheets. js: when SMIL meets HTML5 and CSS3. In: Proceedings of the 11th ACM symposium on Document engineering. ACM, 2011. p. 43-52.

CHEOK, A. D. Electric and Magnetic User Interfaces for Digital Smell and Taste. In: Hyperconnectivity. Springer London, 2016. p. 27-40.

CHO, H.-Y. Event-Based control of 4D effects using MPEG RoSE. 2010. 63f. Dissertação (Mestrado), Instituto Avançado de Ciência e Tecnologia da Coréia.

CHOI, Bumsuk; LEE, Eun-seo; YOON, Kyoungro. Streaming Media with Sensory Effect. 2011 International Conference On Information Science And Applications, Jeju Island, p.1-6, abr. 2011. Institute of Electrical & Electronics Engineers (IEEE). http://dx.doi.org/10.1109/icisa.2011.5772390.

CHOI, B. S.; KIM, S. K. Text of ISO/IEC FDIS 23005-3 2nd edition Sensory Information. 2012.

CLEMENTS, P., NORTHROP, L. Software Product Lines: Practices and Patterns, Addison-Wesley Professional, 2001.

CZARNECKI, K.; EISENECKER, U. W. Generative Programming: Methods, Tools, and Applications. Boston: Addison-Wesley, 2000.

CZARNECKI, K. Overview of generative software development. In: BANÂTRE, J.-P. et al. (Ed.). Unconventional Programming Paradigms, International Workshop (UPP 2004). Springer, (Lecture Notes in Computer Science, v. 3566), p. 326–341, 2004.

CZARNECKI, K.; HELSEN, S.; EISENECKER, U. Staged configuration using feature models. In: NORD, R. L. (Ed.). In: Proceedings of the Third Software Product Line Conference. Boston, MA: Springer, 2004. (LNCS 3154), p. 266–283.

CZARNECKI, K.; ANTKIEWICZ M. Mapping features to models: A template approach based on superimposed variants. In: 4th International Conference on Generative Programming and Component Enginering (GPCE 2005). 2005. Springer. p. 422-437. Tallinn.

CZARNECKI, K.; HELSEN, S. Feature-based survey of model transformation approaches. IBM Systems Journal, v. 45, n. 3, p. 621–645, 2006.

CZARNECKI, K.; STAHL, T.; VOELTER, M. Model-Driven Software Development: Technology. Engineering, Management. Wiley, 2006.

DAVID, Frankel S. Model driven architecture: applying MDA to enterprise computing. 2003.

DALEAIR. Interoperability. 2017. Disponível em: <http://www.daleair.com/vortex-activ>. Acesso em: 30.07.2017.

DEELSTRA S.; SINNEMA, M.; BOSCH J. Product derivation in software product families: a case study. The Journal of Systems and Software. 2005. 74(2): p.173–194.

DEELSTRA S.; SINNEMA, M.; BOSCH J. Product derivation in software product families: a case study. The Journal of Systems and Software. 2005. 74(2): p.173–194

DEMARCO, T. Structured Analysis and System Specification, Englewood Cliffs, Nova Jersey, 1979.

DEURSEN, A. V.; KLINT, P.; VISSER, J. Domain-specific languages: An annotated bibliography. ACM Sigplan Notices, v. 35, n. 6, p. 26-36, 2000.

DUARTE, P. A. S.; P. A.; BARRETO, F. M., GOMES, F. A. A., CARVALHO, W. V.; TRINTA, F. A. M. CRITiCAL: A Configuration Tool for Context Aware and mobiLe Applications. In: Computer

Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual. IEEE, 2015. p. 159-168.

FAVRE, J.; NGUYEN, T. Towards a Megamodel to Model Software Evolution Through Transformations. Electronic Notes In Theoretical Computer Science, [s.l.], v. 127, n. 3, p.59-74, abr. 2005. Elsevier BV. http://dx.doi.org/10.1016/j.entcs.2004.08.034.

FOWLER, M. Language Workbenches: The Killer-App for Domain Specific Languages?, Disponível em: <http://martinfowler.com/articles/languageWorkbench.html>. Acesso em: 15 de Abril. 2016

FREITAS, J.; MEIRA, C.; MELO, M.; BARBOSA, L.; BESSA, M. Information system for the management and visualization of multisensorial contents. In: Information Systems and Technologies (CISTI), 2015 10th Iberian Conference on. IEEE, 2015. p. 1-7.

GHINEA, G.; ANDRES, F.; GULLIVER, S. Multiple sensorial media advances and applications: New developments in Mulsemedia. Information Science Reference, IGI Global, 2011. ISBN 9781609608224.

GHINEA, G.; TIMMERER, C.; LIN, W.; GULLIVER, S. R. Mulsemedia: State of the art, perspectives, and challenges. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), v. 11, n. 1s, p. 17, 2014.

GREENFIELD, Jack; SHORT, Keith. Software factories: assembling applications with patterns, models, frameworks and tools. In: Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications. ACM, 2003. p. 16-27

GUEDES, Á. L. V.; ALBUQUERQUE, A. R. G.; BARBOSA, S. D. J. Extending multimedia languages to support multimodal user interactions. Multimedia Tools and Applications, v. 76, n. 4, p. 5691-5720, 2017.

GUO, Hong et al. PerGO: An Ontology towards Model Driven Pervasive Game Development. In: On the Move to Meaningful Internet Systems: OTM 2014 Workshops. Springer Berlin Heidelberg, 2014. p. 651-654.

GUO, Hong et al. RealCoins: A Case Study of Enhanced Model Driven Development for Pervasive Games. International Journal of Multimedia and Ubiquitous Engineering, v. 10, n. 5, p. 395-410, 2015.

HARIRI, S.; MUSTAFA, N. A.; KARUNANAYAKA, K.; CHEOK, A. D. Electrical stimulation of olfactory receptors for digitizing smell. In: MVAR@ ICMI. 2016. p. 4:1-4:4.

ISO/IEC 23005-1: 2014 Information technology—Media context and control—Part 1: Architecture, January 2014.

ISO/IEC 23005-2: 2013 Information technology—Media context and control—Part 2: Control information, November 2013.

ISO/IEC 23005-3: 2013 Information technology—Media context and control—Part 3: Sensory information, November 2013.

ISO/IEC 23005-4: 2013 Information technology—Media context and control—Part 4: Virtual world object characteristics, November 2013.

ISO/IEC 23005-5: 2013 Information technology—Media context and control—Part 5: Data formats for interaction devices, November 2013.

ISO/IEC 23005-6: 2013 Information technology—Media context and control—Part 6: Common types and tools, November 2013.

ISO/IEC 23005-7: 2014 Information technology—Media context and control—Part 7: Conformance and reference software, January 2014.

ITU: 2014. International Telecommunication Union. H.761: Nested context language (NCL) and Ginga-NCL. Disponível em: https://www.itu.int/rec/T-REC-H.761.

JAIN, R. The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling. 1991. John Wiley.

JALAL, L; MURRONI, M. Enhancing TV broadcasting services: A survey on mulsemedia quality of experience. In: Broadband Multimedia Systems and Broadcasting (BMSB), 2017 IEEE International Symposium on. IEEE, 2017. p. 1-7.

JEDLITSCHKA, A.; PFAHL, D. Reporting guidelines for controlled experiments in software engineering, in International Symposium on Empirical Software Engineering, Australia, 2005.

JOHNSON, L. Showtime's Mobile Ad Sets Off a Virtual Bomb in Your Hand: Feel the suspense — literally. 2014. Disponível em: < http://www.adweek.com/news/technology/showtime-s-mobile-ad-sets-virtual-bomb-your-hand-160250 >. Acesso em: 21 abr. 2016.

KANG K. C., COHEN S., HESS J., Novak W., A. Peterson. Feature-Oriented Domain Analysis (FODA) – Feasibility Study, Technical Report CMU/SEI-90-TR-21, SEI/CMU, 1990.

KANG, K. C.; KIM, S.; LEE, J.; KIM, K.; SHIN, E.; HUH, M. FORM: A Feature-Oriented Reuse Method with domain-specific reference architectures. In: Annals of Software Engineering Notes, v.05, 1998, p.143–168.

KEMPTHORNE, O.; HINKELMANN, K. Design and Analysis of Experiments. Vol. I: Introduction to Experimental Design. Wiley-Interscience. 1994.

KENT, S. Model driven engineering. In: Integrated formal methods. Springer Berlin/Heidelberg, 2002. p. 286-298.

KIM, S. K.; HAN, J. J.; YOON, K. R. Text of ISO/IEC FDIS 23005-5 2nd edition Data Formats for Interaction Device. 2012.

KIM, Sang-kyun. Authoring multisensorial content. Signal Processing: Image Communication, Yongin-si, v. 28, n. 2, p.162-167, fev. 2013. Elsevier BV. http://dx.doi.org/10.1016/j.image.2012.10.011.

KIM, Sang-kyun; JOO, Yong-soo; LEE, Youngmi. Sensible Media Simulation in an Automobile Application and Human Responses to Sensory Effects. Etri J, [s.l.], v. 35, n. 6, p.1001-1010, 2 dez. 2013. Electronics and Telecommunications Research Institute (ETRI). http://dx.doi.org/10.4218/etrij.13.2013.0038.

KIM, S. K.; HAN, J. J. Text of white paper on MPEG-V. In: San Jose, USA. MPEG Group Meeting, ISO/IEC JTC. 2014.

KLEPPE, A.; WARMER, J.; BAST, W. MDA Explained - The Model Driven Architecture: Practice and Promise. [S.l.]: Addison-Wesley. 2003. (Object Technology Series).

KOEGEL, M.; HELMING J. What Every Eclipse Developer Should Know About EMF. EMF Tutorial. Disponível em <https://eclipsesource.com/blogs/tutorials/emf-tutorial/>. Acesso em: 20 Agosto de 2017.

KRISHNA, A. An integrative review of sensory marketing: Engaging the senses to affect perception, judgment and behavior. Journal of Consumer Psychology, v. 22, n. 3, p. 332-351, 2012.

KULESZA, R. Uma Abordagem de Desenvolvimento Orientado a Modelos para a Integração entre Projetos de Mídia e Software no Domínio de Aplicações de TV Digital. 2013. 176f. Tese (Doutorado), UFPE, Recife, Brasil.

KULESZA, R.; MEIRA, S. R.; FERREIRA, T. P.; ALEXANDRE, E. S.; GUIDO, F. L. S., NETO, M. C. M.; SANTOS, C. A. A model-driven approach for integration of interactive applications and web services: a case study in interactive digital TV platform. In: Multimedia and Expo Workshops (ICMEW), 2012 IEEE International Conference on. IEEE, 2012. p. 266-271.

KULESZA, R.; MEIRA, S. R. D. L.; FERREIRA, T. P.; SILVA, N. V. R., ALEXANDRE, E. F. D. S.; SOUZA, F. G. L. Towards a generative software development approach for rapid prototyping iDTV applications. In: Proceedings of the 18th Brazilian symposium on Multimedia and the web. ACM, 2012. p. 91-98.

KÜHNE, T. Matters of (Meta-) Modeling. Softw Syst Model, [s.l.], v. 5, n. 4, p.369-385, 27 jul. 2006. Springer Science + Business Media. http://dx.doi.org/10.1007/s10270-006-0017-9.

LEE, J.; MUTHIG, D. Feature-oriented variability management in product line engineering. Communications of the ACM, v. 49, n. 12, p. 55–59, 2006.

LE CALLET , P; MÖLLER, S.; PERKIS, A., Qualinet White Paper on Definitions of Quality of Experience. European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003), v1.2, 2013.

LIDDLE, S. W. Model-Driven Software Development. Handbook of Conceptual Modeling: Theory, Practice, and Research Challenges. Springer-Verlag. 2011. p. 17-54.

LONIEWSKI, Grzegorz; INSFRAN, Emilio; ABRAHÃO, Silvia. A systematic review of the use of requirements engineering techniques in model-driven development. In: Model driven engineering languages and systems. Springer Berlin Heidelberg, 2010. p. 213-227.

LUCRÉDIO, D. Uma Abordagem Orientada a Modelos para Reutilização de Software. 2009. 287f. Tese (Doutorado), USP São Carlos, Brasil.

MARQUES NETO, M. C. Contribuições para a Modelagem de Aplicações Multimídia em TV digital interativa. 2011. 170f. Tese (Doutorado). Universidade Federal da Bahia, Salvador, Brasil.

MEDEIROS, A. L.; CAVALCANTE, E.; BATISTA, T.; SILVA, E. ArchSPL-MDD: An ADL-Based Model-Driven Strategy for Automatic Variability Management. 2015 In Brazilian Symposium On Components, Architectures And Reuse Software, Belo Horizonte, p.120-129, set. 2015. Institute of Electrical & Electronics Engineers (IEEE).

MEIXNER, B. Hypervideos and Interactive Multimedia Presentations. ACM Computing Surveys (CSUR), v. 50, n. 1, p. 9, 2017.

MELLOR, S.J. MDA Distilled – Principles of Model Driven Architecture [S.l.]: Addison-Wesley, 2004.

MILLER, M. J. Miller, J. Mukerji. MDA guide version, v. 1, n. 1, 2003.

MONKS, J.; OLARU, A.; TAL, I.; MUNTEAN, G. M. Quality of experience assessment of 3D video synchronised with multisensorial media components. In: Broadband Multimedia Systems and Broadcasting (BMSB), 2017 IEEE International Symposium on. IEEE, 2017. p. 1-6.

MÖLLER, S.; RAAKE, A. (Ed.). Quality of experience: advanced concepts, applications and methods. Springer, 2014.

NGUYEN, X. T.; TRAN, H. T.; BARAKI, H.;GEIHS, K. FRASAD: A framework for model-driven IoT Application Development. 2015 Ieee 2nd World Forum On Internet Of Things (wf-iot), Milan, p.387-392, dez. 2015. Institute of Electrical & Electronics Engineers (IEEE). http://dx.doi.org/10.1109/wf-iot.2015.7389085.

OBJECT MANAGEMENT GROUP (OMG). MDA Guide Version 1.0.1. [S.l.], 2003. Disponível em:

<http://www.omg.org>. Acesso em: 27 Agosto de 2017.

OLDEVIK, J. MOFScript User Guide. 2011.

OVIATT, S. Multimodal Interfaces. Hum-Comput Interact Handb. CRC Press, 413–432, 2007.

PETIT, O.; CHEOK, A. D.; SPENCE, C.; VELASCO, C.; KARUNANAYAKA, K. T. Sensory marketing in light of new technologies. In: Proceedings of the 12th International Conference on Advances in Computer Entertainment Technology. ACM, 2015. p. 53.

PHILIPS. Philips amBX Gaming PC peripherals SGC5103BD Premium kit. 2017. Disponível em: < https://goo.gl/Lr9TiM >. Acesso em: 30 Julho de 2017.

PLEUSS, A. Model-driven development of interactive multimedia applications. 2009. 287f. Tese (Doutorado), Universidade de Munique, Munique, Alemanha.

PLEUSS, A.; HUSSMANN, H. Model-Driven Development of Interactive Multimedia Applications with MML. Studies in Computation Intelligence. Springer. 2011. v. 340, p.199-218.

PLEUSS, A. Generating code skeletons for individual media elements in model-driven development of interactive systems. In: Proceedings of the 2014 ACM SIGCHI symposium on Engineering interactive computing systems. ACM, 2014. p. 155-160.

PM, I. A guide to the project management body of knowledge (PMBOK guide). Project Management Institute, Newtown Square, 2000.

POHL, K; BÖCKLE, G.; LINDEN , F. J. V. D. Software Product Line Engineering: Foundations, Principles and Techniques. New York: Springer, 2005.

PONS, C; GIADINI, R; PEREZ, G. Desarrollo de Software Dirigido por Modelos – Conceptos teóricos y su aplicación práctica. EDULP. 2010.

POPMA, R. JET Tutorial Part 1 (Introduction to JET). May 2004. Eclipse Corner Article. Disponível em: <http://www.eclipse.org/articles/Article-JET/jet_tutorial1.html>. Acesso em: 20 Agosto de 2017

PRADO, E. F.; LUCREDIO, D. A Flexible Model-Driven Game Development Approach. 2015 In Brazilian Symposium On Components, Architectures And Reuse Software, Belo Horizonte, p.130-139, set. 2015. Institute of Electrical & Electronics Engineers (IEEE). http://dx.doi.org/10.1109/sbcars.2015.24.

PREDA, M.; JOVANOVA, B. Avatar interoperability and control in virtual Worlds. Signal Processing: Image Communication, [s.l.], v. 28, n. 2, p.168-180, fev. 2013. Elsevier BV. http://dx.doi.org/10.1016/j.image.2012.10.012

RANASINGHE, N.; CHEOK, A.; NAKATSU, R.; DO, E. Y. L. Simulating the sensation of taste for immersive experiences. In: Proceedings of the 2013 ACM international workshop on Immersive media experiences. ACM, 2013. p. 29-34

RANASINGHE, N.; DO, E. Y. Virtual Sweet: Simulating Sweet Sensation Using Thermal Stimulation on the Tip of the Tongue. In: Proceedings of the 29th Annual Symposium on User Interface Software and Technology. ACM, 2016. p. 127-128.

REYES, H. C.; MUÑOZ-ARTEAGA, J.; GONZÁLEZ-CALLEROS, J. M. Model-Driven Development of Interactive Environments for Occupational Therapy. In: HCI for Children with Disabilities. Springer International Publishing, 2017. p. 91-113.

ROBLES-BYKBAEV, V.; LÓPEZ-NORES, M.; GALÁN-MENA, J. A.; WONG, V. C. L.; QUISI-PERALTA, D.; LIMA-JUMA, D.; ... PAZOS-ARIAS, J. J. An Intelligent Ecosystem to Support the Development of Communication Skills in Children with Autism: An Experience Based on Ontologies, Multi-Sensory Stimulation Rooms, and Robotic Assistants. In: Smart Technology Applications in Business Environments. IGI Global, 2017. p. 109-133.

SÁNCHEZ, I. F. H. Quadrados latinos com aplicações em engenharia de software. 100f. Dissertação (Mestrado), Universidade Federal de Pernambuco, Recife, 2011.

SALEME, E. B.; SANTOS, C. A. S. PlaySEM: a Platform for Rendering Mulsemedia Compatible with MPEG-V. Proceedings Of The 21st Brazilian Symposium On Multimedia And The Web - Webmedia '15, [s.l.], p.145-148, 2015. Association for Computing Machinery (ACM). http://dx.doi.org/10.1145/2820426.2820450.

SALEME, E. B.; CELESTRINI, J. R.; SANTOS, C. A. S. Time Evaluation for the Integration of a Gestural Interactive Application with a Distributed Mulsemedia Platform. In: Proceedings of the 8th ACM on Multimedia Systems Conference. ACM, 2017. p. 308-314.

SANTOS, C. A. S.; NETO, A. N. R.; SALEME, E. B. An Event Driven Approach for Integrating Multi-Sensory effects to Interactive Environments. In: Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on. IEEE, 2015. p. 981-986.

SARAIVA, Diego et al. Architecting a Model-Driven Aspect-Oriented Product Line for a Digital TV Middleware: A Refactoring Experience. Software Architecture, Copenhagen, v. 6285, p.166-181, 2010. Springer Science + Business Media. http://dx.doi.org/10.1007/978-3-642-15114-9_14.

SAKIA R. M. The box-cox transformation technique: A review. Journal of the Royal Statistical Society. Series D (The Statistician), 41(2), 1992.

SELIC, B. The pragmatics of model-driven development. Ieee Softw., [s.l.], v. 20, n. 5, p.19-25, set. 2003. Institute of Electrical & Electronics Engineers (IEEE). http://dx.doi.org/10.1109/ms.2003.1231146.

SHAW M. The coming-of-age of software architecture research. In: Proceedings of the 23rd International Conference on Software Engineering. Washington (ICSE 2001), DC, USA: IEEE Computer Society, 2001, p. 656.

SILVA, A. R. Model-driven engineering: A survey supported by the unified conceptual model. Computer Languages, Systems & Structures, [s.l.], v. 43, p.139-155, jun. 2015. Elsevier BV. http://dx.doi.org/10.1016/j.cl.2015.06.001.

SPENCE, C.; OBRIST, M.; VELASCO, C.; RANASINGHE, N. Digitizing the chemical senses: Possibilities & pitfalls. International Journal of Human-Computer Studies, 2017.

STEINBERG, D., BUDINSKY, F., MERKS, E., PATERNOSTRO, M. EMF: Eclipse Modeling Framework. 2.a Edição, Addison-Wesley Professional, 2008.

SULEMA, Y. Mulsemedia vs. Multimedia: State of the art and future trends. In: Systems, Signals and Image Processing (IWSSIP), 2016 International Conference on. IEEE, 2016. p. 1-5.

THOMAS, D. MDA: Revenge of the modelers or UML utopia? IEEE Software, v. 21, n. 3, p. 15–17, 2004.

TIMMERER, C.; WALTL, M.; HELLWAGNER, H. Are Sensory Effects Ready for the World Wide Web? In: Proceedings of the Workshop on Interoperable Social Multimedia Applications (WISMA'10). Barcelona, Spain. 2010.

UTTING, Mark; LEGEARD, Bruno. Practical model-based testing: a tools approach. Morgan Kaufmann, 2010.

VOELTER, M.; VISSER, E. Product Line Engineering using. Proceeding of the 2011 15th InternationalSoftware Product Line Conference (SPLC). Washington: IEEE Computer Society. 2011. p. 70-79.

YOON, K. End-to-end framework for 4-D broadcasting based on MPEG-V standard. Signal Processing: Image Communication, [s.l.], v. 28, n. 2, p.127-135, fev. 2013. Elsevier BV. http://dx.doi.org/10.1016/j.image.2012.10.008.

YOON, K.; KIM, S. K.; HAN, J. J.; HAN, S.; PREDA, M. MPEG-V: Bridging the Virtual and Real World. Academic Press, 2015.

WOHLIN, C.; RUNESON, P.; HOST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLEN, A. Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers, 2000.

W3C. 2012. Synchronized Multimedia (SMIL). Disponível em: http://www.w3.org/AudioVideo/

W3C. 2014. HTML5—A vocabulary and associated APIs for HTML and XHTML (W3C recommendation). Disponível em: http://www.w3.org/TR/html5/Overview.html.

WALTL, M.; TIMMERER, C.; HELLWAGNER, H. A Test-Bed for Quality of Multimedia Experience Evaluation of Sensory Effects. In: First International Workshop on Quality of Multimedia Eexperience (QoMEX 2009). [s.n.], 2009.

WALTL, M.; TIMMERER, C.; HELLWAGNER, H. Improving the Quality of Multimedia E experience through Sensory Effects. In: 2nd International Workshop on Quality of Multimedia E experience (QoMEX'10). [s.n.], 2010.

WALTL, Markus et al. An end-to-end tool chain for Sensory Experience based on MPEG-V. Signal Processing: Image Communication, New York, v. 28, n. 2, p.136-150, fev. 2013. Elsevier BV. http://dx.doi.org/10.1016/j.image.2012.10.009.

WEISS, D.; LAI, C. Software Product-Line Engineering: A Family-Based Software Development Process, Addison-Wesley Professional, 1999.

WITMER, B. G.; SINGER, M. J. Measuring presence in virtual environments: A presence questionnaire. Presence: Teleoper. Virtual Environ., MIT Press, Cambridge, MA, USA, v. 7, n. 3, p. 225–240, jun. 1998.

WOHLIN, C.; RUNESON, P.; HOST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLEN, A. Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers, 2000.

ZHANG, L.; SUN, S.; XING, B.; FU, J.; YU, S. Exploring Olfaction for Enhancing Multisensory and Emotional Game Experience. In: International Conference on Technologies for E-Learning and Digital Entertainment. Springer International Publishing, 2016. p. 111-121.

ZHU, M.; WANG, A. I.; TRÆTTEBERG, H. Engine-cooperative game modeling (ecgm): Bridge model-driven game development and game engine tool-chains. In: Proceedings of the 13th International Conference on Advances in Computer Entertainment Technology. ACM, 2016. p. 22.

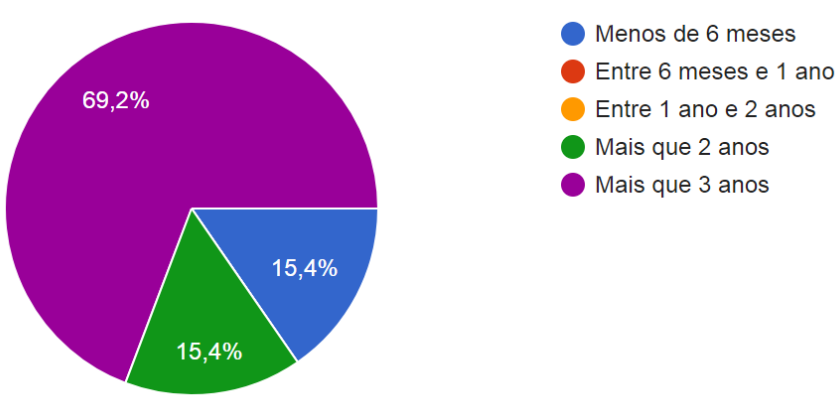
ZOU, L.; TAL, I.; COVACI, A.; IBARROLA, E.; GHINEA, G.; MUNTEAN, G. M. Can Multisensorial Media Improve Learner Experience?. In: Proceedings of the 8th ACM on Multimedia Systems Conference. ACM, 2017. p. 315-320.

APÊNDICE A – ESTUDOS QUALITATIVOS

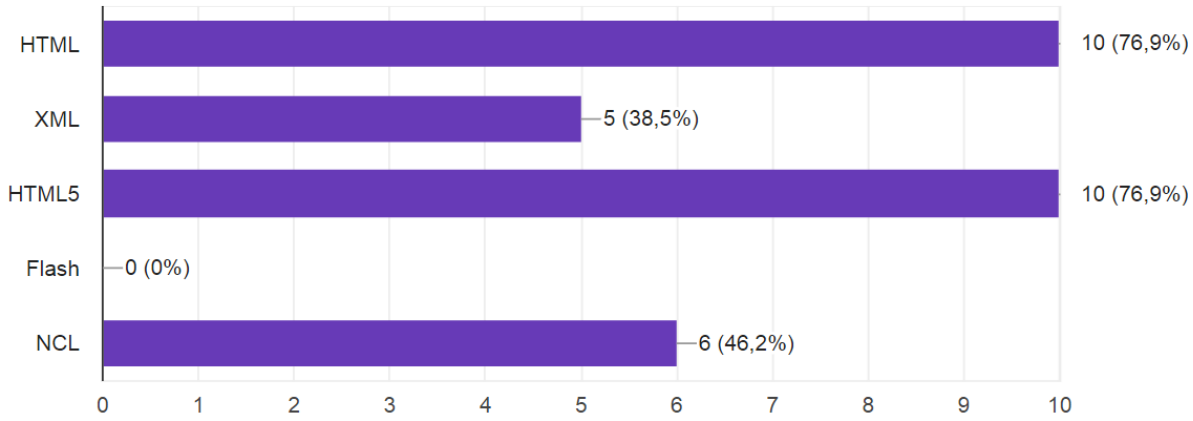
A.1 Estudo Qualitativo do Primeiro Experimento

A.1.1 Perfil dos Sujeitos

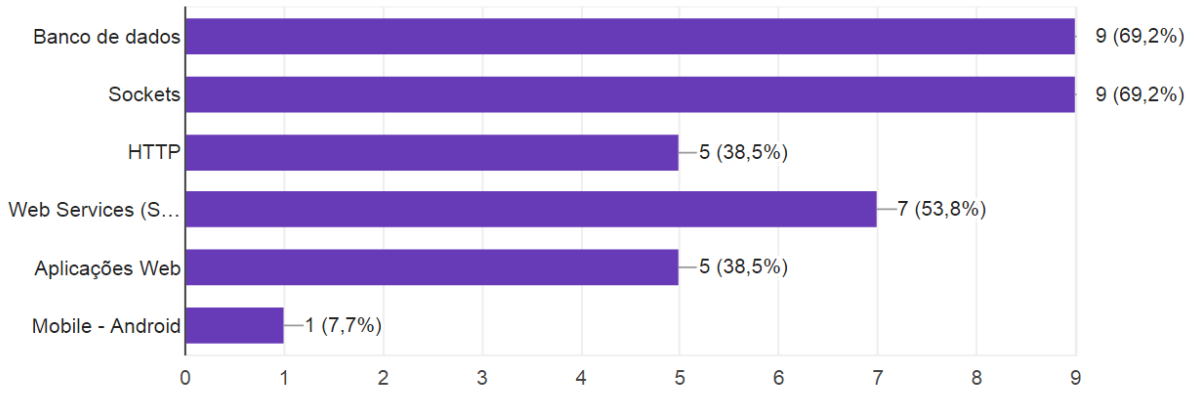
1. Tempo de experiência em desenvolvimento de Software



2. Experiência no desenvolvimento de aplicações por meio de linguagens declarativas

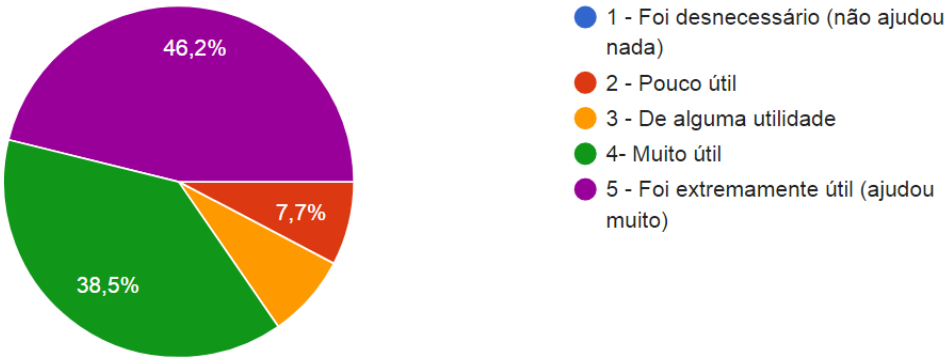


3. Experiência com outras tecnologias

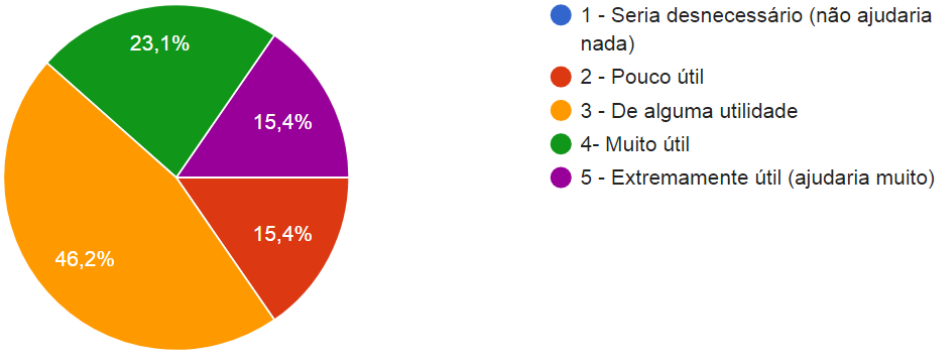


A.1.2 Avaliação da Abordagem

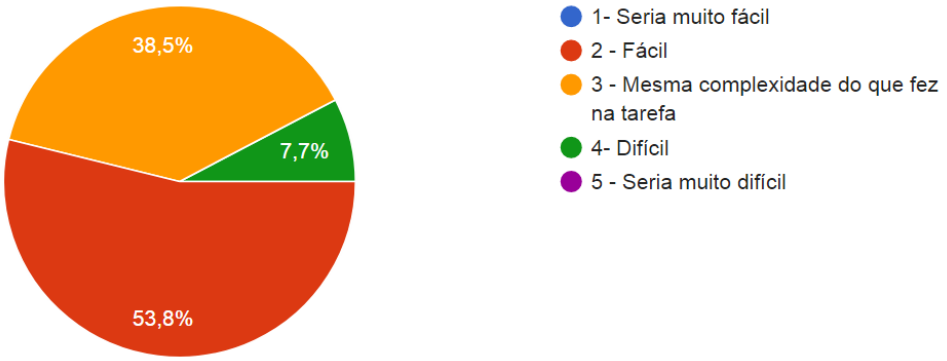
4. Numa escala de 1 a 5, em que 1 indica que não ajudou e 5 que ajudou muito, como você avalia a utilidade da ferramenta SEVino comparada a programação textual com XML executada na tarefa?



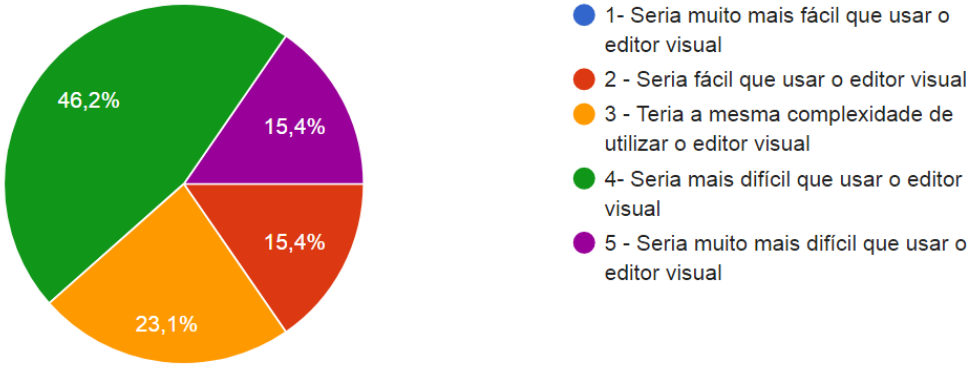
5. Numa escala de 1 a 5, em que 1 indica que não ajudou e 5 que ajudou muito, como você avalia se a ferramenta SEVino tivesse uma funcionalidade de formulário (*wizards*) para escolher e definir as aplicações desenvolvidas a partir de *templates* (modelos pré-definidos)?



6. Considere o cenário que se na tarefa fosse pedido uma funcionalidade para integrar alguns efeitos no vídeo com um trecho de programa com lógica de aplicação em javascript (por exemplo, se conectar ao banco de dados para exibir uma informação remota na tela junto com um efeito sensorial). Numa escala de 1 a 5, onde 1 indica que seria fácil e 5 que seria difícil, como você avalia implementar essa funcionalidade de integração do código javascript com o código XML utilizando uma extensão da ferramenta SEVino que permitesse configurar essa integração de modo visual?



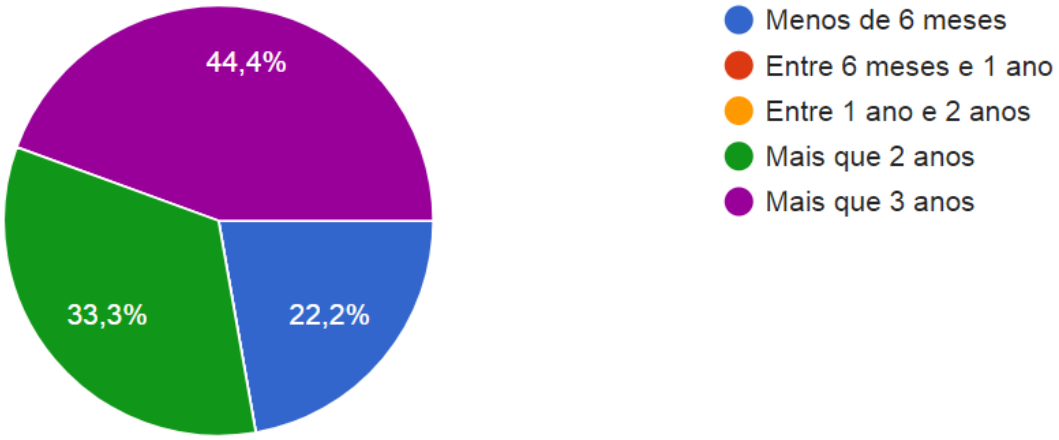
7. Considere o cenário que se na tarefa fosse pedido uma funcionalidade para integrar alguns efeitos no vídeo com um trecho de programa com lógica de aplicação em javascript (por exemplo, se conectar ao banco de dados para exibir uma informação remota na tela junto com um efeito sensorial). Numa escala de 1 a 5, onde 1 indica que seria fácil e 5 que seria difícil, como você avaliar implementar essa funcionalidade de integração do código javascript com o código XML utilizando o editor textual da tarefa.



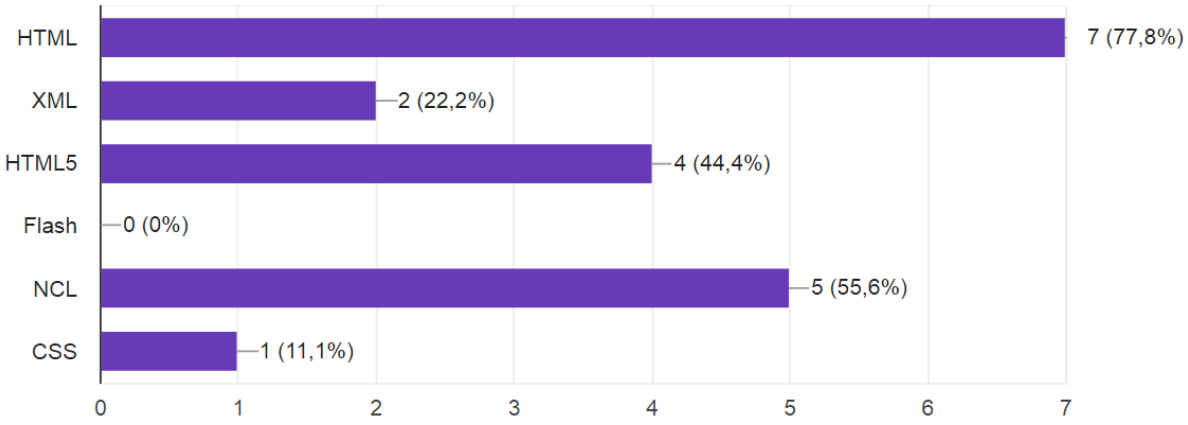
A.2 Estudo Qualitativo do Segundo Experimento

A.2.1 Perfil dos Sujeitos

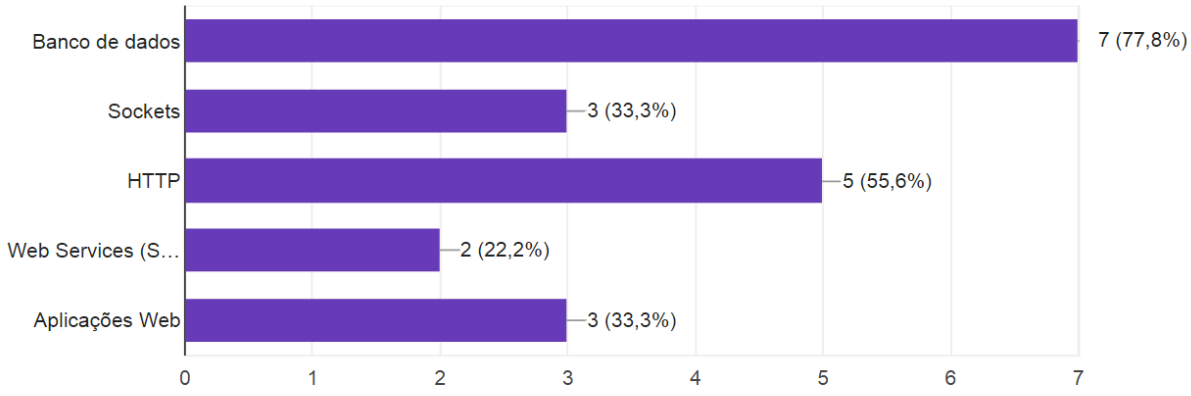
1. Tempo de experiência em desenvolvimento de Software



2. Experiência no desenvolvimento de aplicações por meio de linguagens declarativas

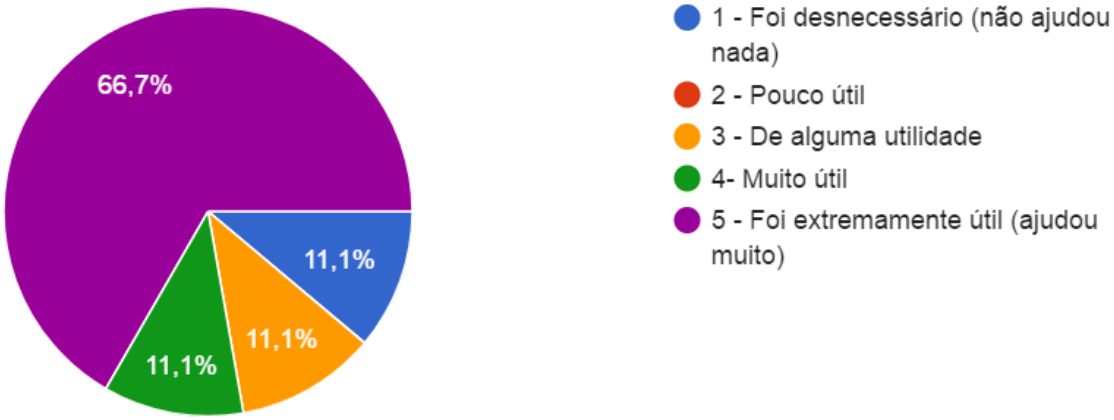


3. Experiência com outras tecnologias

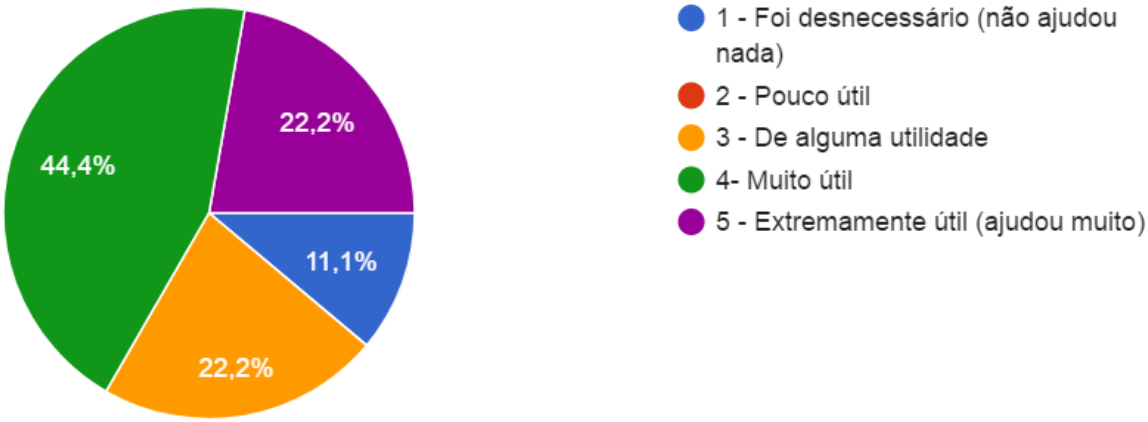


A.2.2 Avaliação da Abordagem

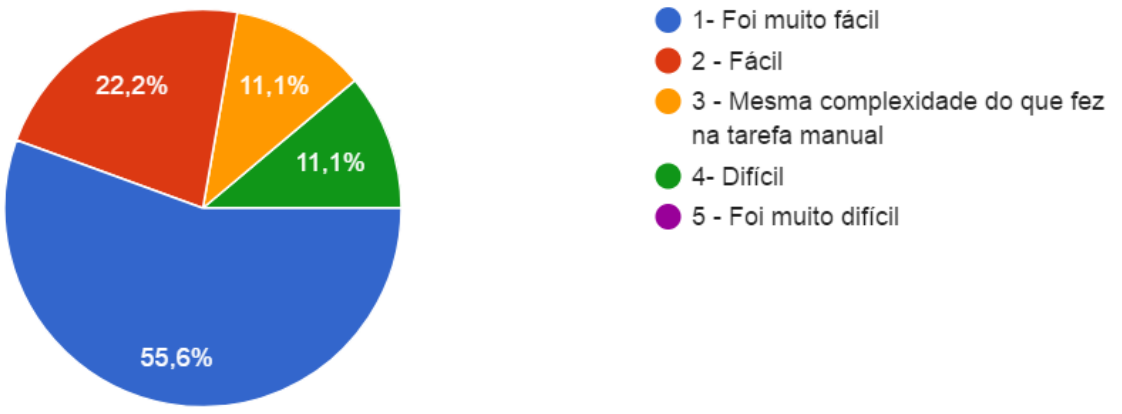
4. Numa escala de 1 a 5, onde 1 indica que não ajudou e 5 ajudou muito. Como você avalia a utilidade da ferramenta de geração comparado a programação textual com HTML/Javascript executada na tarefa?



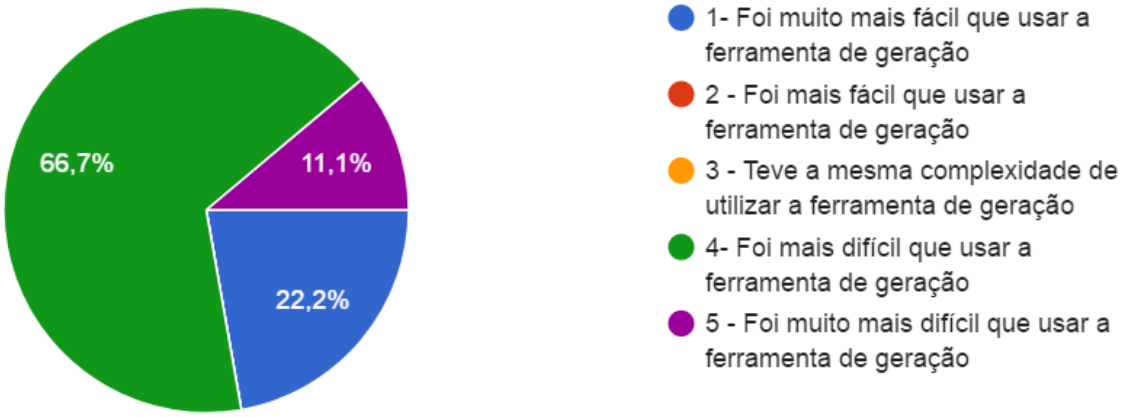
5. Numa escala de 1 a 5, onde 1 indica que não ajudou e 5 que ajudou muito. Como você avalia a ferramenta de geração em relação ao uso de formulários (wizards) para escolher e definir as funcionalidades das aplicações a partir de templates (modelos pré-definidos)?



6. Em relação a funcionalidade de integrar efeitos na cena final da aplicação com um trecho de programa com lógica de aplicação em javascript (no caso, exibir efeitos e uma mídia de áudio diferentes). Numa escala de 1 a 5, onde 1 indica que foi fácil e 5 que foi difícil, como você avalia a implementação dessa funcionalidade utilizando a ferramenta de geração?



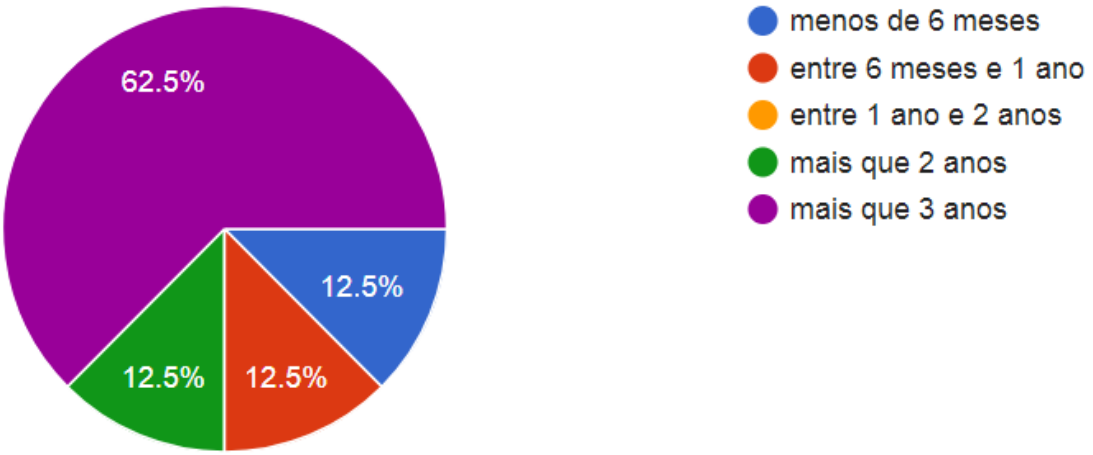
7. Em relação à funcionalidade de integrar efeitos na cena final da aplicação com um trecho de programa com lógica de aplicação em javascript (no caso, exibir efeitos e uma mídia de áudio diferentes). Numa escala de 1 a 5, onde 1 indica que foi fácil e 5 que foi difícil, como você avalia a implementação dessa funcionalidade utilizando a programação com o editor textual?



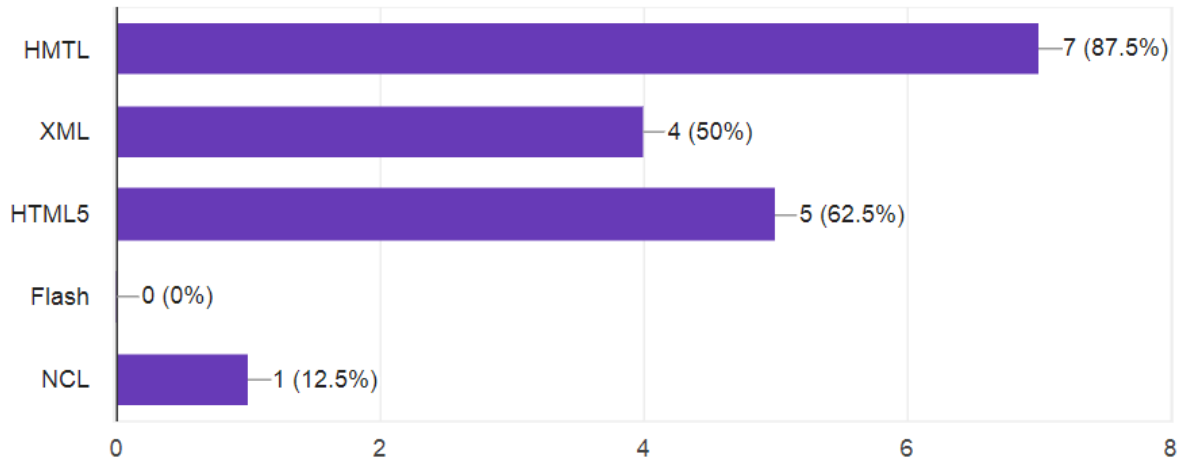
A.3 Estudo Qualitativo do Terceiro Experimento

A.3.1 Perfil dos Sujeitos

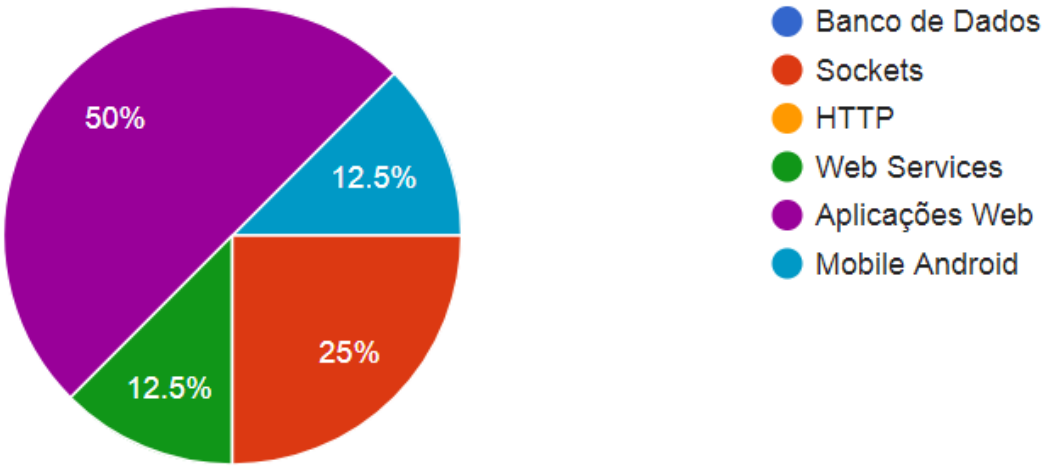
1. Tempo de experiência em desenvolvimento de Software



2. Experiência no desenvolvimento de aplicações por meio de linguagens declarativas

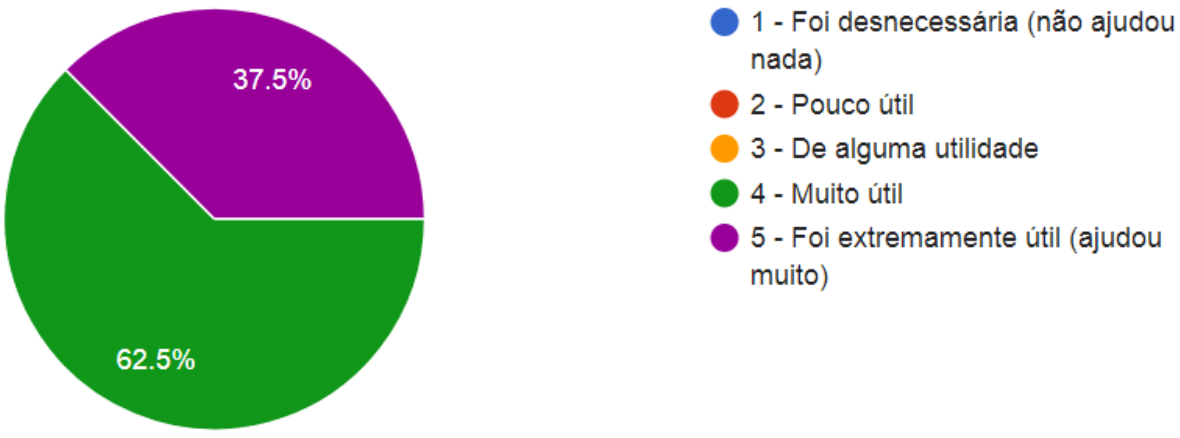


3. Experiência com outras tecnologias

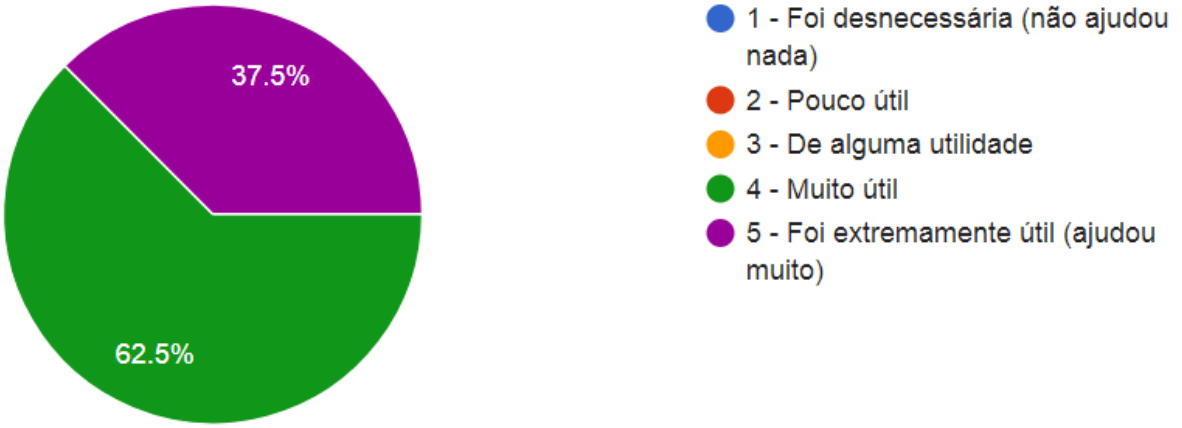


A.3.2 Avaliação da Abordagem

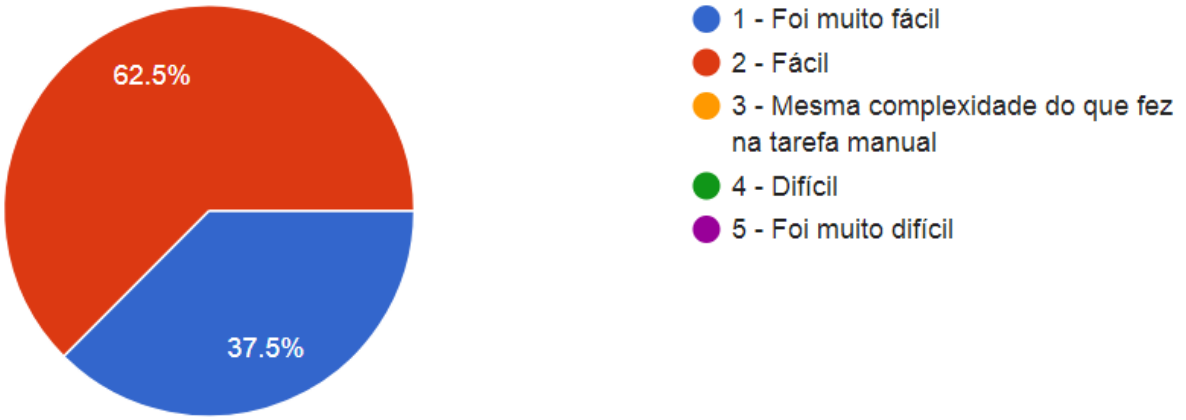
4. Numa escala de 1 a 5, onde 1 indica que não ajudou e 5 ajudou muito. Como você avalia a utilidade da ferramenta de geração comparado a programação textual com HTML/Javascript executada na tarefa?



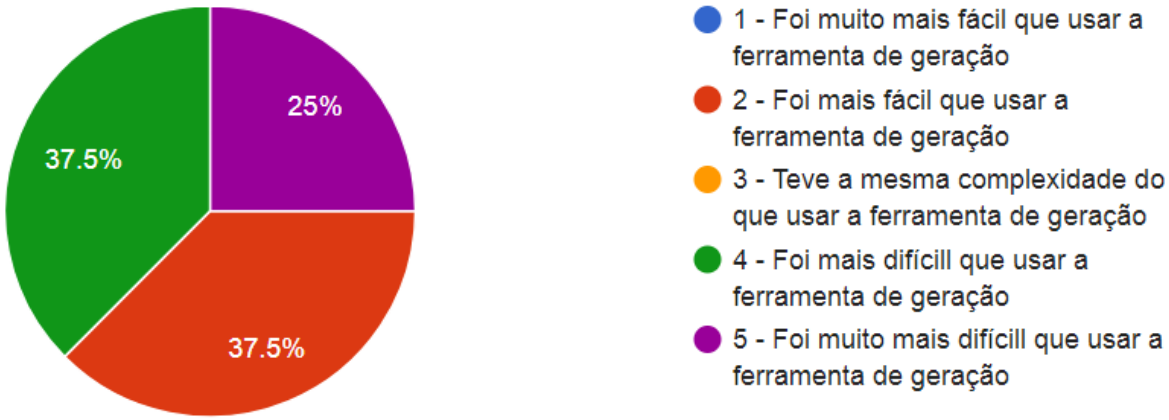
5. Numa escala de 1 a 5, onde 1 indica que não ajudou e 5 que ajudou muito. Como você avalia a ferramenta de geração em relação ao uso de formulários (wizards) para escolher e definir as funcionalidades das aplicações a partir de templates (modelos pré-definidos)?



6. Em relação a funcionalidade de integrar efeitos na cena final da aplicação com um trecho de programa com lógica de aplicação em javascript (no caso, exibir efeitos e uma mídia de áudio diferentes). Numa escala de 1 a 5, onde 1 indica que foi fácil e 5 que foi difícil, como você avalia a implementação dessa funcionalidade utilizando a ferramenta de geração?



7. Em relação à funcionalidade de integrar efeitos na cena final da aplicação com um trecho de programa com lógica de aplicação em javascript (no caso, exibir efeitos e uma mídia de áudio diferentes). Numa escala de 1 a 5, onde 1 indica que foi fácil e 5 que foi difícil, como você avalia a implementação dessa funcionalidade utilizando a programação com o editor textual?


25%
37.5%
37.5%
1 - Foi muito mais fácil que usar a ferramenta de geração
2 - Foi mais fácil que usar a ferramenta de geração
3 - Teve a mesma complexidade do que usar a ferramenta de geração
4 - Foi mais difícill que usar a ferramenta de geração
5 - Foi muito mais difícill que usar a ferramenta de geração