UNIVERSIDADE FEDERAL DO MARANHÃO

Programa de Pós-Graduação em Ciência da Computação

Danne Makleyston Gomes Pereira

Integrando a Internet das Coisas às Aplicações de TV Digital

São Luís
2018

UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Danne Makleyston Gomes Pereira

Integrando a Internet das Coisas às Aplicações de TV Digital

São Luís
2018

Danne Makleyston Gomes Pereira

Integrando a Internet das Coisas às Aplicações de TV Digital

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Maranhão como requisito parcial para a obtenção do grau de MESTRE em Ciência da Computação.

Orientador: Francisco José da Silva e Silva

Doutor em Ciência da Computação – UFMA

São Luís

2018

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Pereira, Danne.
Integrando a Internet das Coisas às Aplicações de TV Digital / Danne Pereira. – 2018.
98 f.

Coorientador(a): Carlos Neto.
Orientador(a): Francisco Silva.
Dissertação (Mestrado) – Programa de Pós-graduação em Ciência da Computação/ccet, Universidade Federal do Maranhão, São Luís – MA, 2018.

1. Internet das Coisas. 2. Mulsemedia. 3. TV Digital. I. Neto, Carlos. II. Silva, Francisco. III. Título.

Danne Makleyston Gomes Pereira

Integrando a Internet das Coisas às Aplicações de TV Digital

Este exemplar corresponde à redação final da dissertação devidamente corrigida e defendida por Danne Makleyston Gomes Pereira e aprovada pela comissão examinadora.

Aprovada em 27 de Jun de 2018

BANCA EXAMINADORA

Francisco José da Silva e Silva (orientador)

Doutor em Ciência da Computação – UFMA

Carlos de Salles Soares Neto

Doutor em Informática – UFMA

Sergio Colcher

Doutor em Informática – PUC-Rio

Rafael Fernandes Lopes

Doutor em Engenharia Elétrica – UFMA

Dedico esta dissertação a toda minha família.

Resumo

A TV digital permite, além de uma qualidade de imagem e som muito superior à televisão convencional, a possibilidade de interatividade e o oferecimento de novos serviços aos telespectadores. Por outro lado, a Internet das Coisas (IoT) é uma combinação entre a computação ubíqua e a Internet, na qual os dispositivos da IoT (*e. g.*, *smart objects*) podem coletar e trocar dados, cooperando com pessoas e o ambiente no qual se encontram. Muitos *smart objects* presentes em uma casa podem oferecer recursos que somados à reprodução de um conteúdo apresentado na TV pode enriquecer a experiência de se assistir conteúdo televisivo. Contudo, essa convergência entre dispositivos de IoT e aplicações de TV possui diversos desafios, tais como: a descoberta dinâmica de *smart objects* e dos serviços que eles disponibilizam; o suporte à heterogeneidade dos protocolos de comunicação utilizados pelos *smart objects*; a identificação dos telespectadores e seus perfis; a sincronização da mídia em reprodução com os *smart objects* presentes no ambiente físico; e abstrações de programação para o desenvolvimento de aplicações de TV que sejam integradas aos *smart objects* presentes no espaço físico de apresentação. Diante disso, este trabalho propõe uma arquitetura de *software*, baseada na perspectiva de *middleware* de IoT e de TV digital, que permite às aplicações de TV estarem ciente do contexto do ambiente físico no qual a TV se encontra e alterarem aspectos físicos do mesmo, além de identificarem os telespectadores presentes e receberem interações multimodais realizadas por eles. A arquitetura proposta faz uso do modelo conceitual IoTTV-Ont, também desenvolvido neste trabalho, capaz de descrever um domicílio com os *smart objects* e as pessoas presentes. Foi desenvolvido um protótipo que implementa uma materialização concreta da arquitetura proposta e o mesmo foi exposto a uma avaliação por meio de casos de uso, através da qual foi possível observar como os objetivos e os requisitos especificados nesse trabalho foram contemplados.

Palavras-chave: TV Digital, Internet das Coisas (IoT), *Mulsemedia*.

Abstract

In comparison to the conventional TV, Digital TV allows not only a better image and sound quality but also interactivity and the offering of new services to viewers. On the other hand, the Internet of Things (IoT) is a combination of ubiquitous computing and the Internet, in which IoT devices (e.g., smart objects) can get and exchange data, cooperating with people and the environment in which they are. Many smart objects located in a home can offer features that can be explored to enrich the experience of watching television. However, the convergence between IoT devices and TV applications has several challenges, such as the dynamic discovery of smart objects and the services they provide; the support for the heterogeneity of communication protocols used by smart objects; the identification of viewers and their profiles; the synchronization of the playing media with the smart objects actuators provided in the physical environment; and the need of providing programming abstractions for the development of TV applications that are aware of the smart objects deployed in the physical environment. This work proposes a software architecture, based on a middleware perspective, which allows TV applications to be aware of the physical environment context and interact with it. It also allows the identification of the viewers and their profiles, also providing multi-modal interactions with them and the application running in the digital TV. The proposed software architecture is based on a conceptual model called IoTTV-Ont, also developed as part of this work, that allows the description of a household with smart objects and the people present in it. A prototype that uses a particular implementation of the proposed software architecture was evaluated through use cases, exploring all the requirements specified in this work.

Keywords: Digital TV, Internet of Things (IoT), Mulsemedia.

Agradecimentos

Este trabalho foi fruto de muito esforço depositado por meu orientador e por mim durante a minha vida acadêmica no mestrado. Contudo, nada disso seria possível sem a contribuição direta e indireta de muitas pessoas.

Antes de tudo agradeço a Deus por tornar tudo isso realidade.

Agradeço ao meu orientador, o Prof. Francisco José da Silva e Silva, por ter me acompanhado durante essa jornada fornecendo apoio, dedicação, disposição e compreensão. Sendo sempre presente em todas os momentos deste trabalho.

À minha querida e amável esposa pela paciência, compreensão, e, principalmente, pelo apoio incondicional aplicado em todos os momentos desse mestrado. Principalmente, durante o início dessa minha empreitada, visto que teve a serenidade e força para abdicar de várias coisas em nossas vidas para que eu pudesse estar presente na UFMA e, assim, realizar meus estudos.

Ao meu co-orientador pelo apoio e paciência nas revisões e nas avaliações desse trabalho.

Ao Álan Guedes pela colaboração.

A todos os membros do LSDi, em especial aos meus amigos Anderson e Tércio. Ambos por estarem presente nessa caminhada compartilhando experiências e apoiando em diversas etapas desse estudo. Agradeço ainda, pelas discussões que sempre foram proveitosas e que sentirei saudades.

À minha família que sempre foi presentes em tudo que fiz na vida. Em especial, aos meus pais que nunca mediram esforços para propor uma educação à mim e a minha irmã na qual me espelho muito.

Ao Instituto Federal do Tocantins – IFTO Campus Colinas do Tocantins – por terem concedido o afastamento para capacitação no qual me permitiu dedicar ao mestrado.

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

Albert Einstein

Lista de Figuras

Lista de Tabelas

Lista de Siglas

API *Application Programming Interface*.

ATSC-T *Advanced Television System Committee*.

BLE *Bluetooth Low Energy*.

CASMOC *Comunication Abstraction and Smart Object Monitor Component*.

CBDP *Context-Based Digital Personality*.

CEP *Complex Event Processing*.

DASE *Digital Television Application Software Environment*.

DVB-T *Digital Video Broadcasting - Terrestrial*.

ECA *Event-Condition-Action*.

EPG *Electronic Programming Guide*.

EPL *Event Processing Languages*.

FOAF *Friend of a Friend*.

GPS *Global Position System*.

GUI *Graphical User Interface*.

IDE *Integrated Development Environment*.

iDTV TV Digital Interativa.

IoMT *Internet of Mobile Things*.

IoT *Internet of Things*.

IP *Internet Protocol*.

IPTV *Internet Protocol Television*.

ISDB-Tb *Integrated Services Digital Broadcasting Terrestrial*.

ITU-T *ITU Telecommunication Standardization Sector*.

JSON *JavaScript Object Notation*.

M-Hub *Mobile Hub*.

Mbps *Megabit per second*.

MEPA *Mobile Event Processing Agent*.

MHP *Multimedia Home Platform*.

MPEG-V *Moving Picture Experts Group - Media context and control*.

MQTT *Message Queuing Telemetry Transport*.

MR-UDP *Mobile Reliable UDP*.

MulseMedia *Multisensory Media*.

MVG *Cyclomatic Complexity*.

NCL *Nested Context Language*.

NCLOC *Non-Comment Lines of Code*.

OSGi *Open Services Gateway Initiative*.

OWL *Web Ontology Language*.

P2PTV *Peer-to-Peer TV*.

RAM *Random Access Memory*.

S2PA *Short-Range Sensor, Presence and Actuation*.

SBTVD Sistema Brasileiro de Televisão Digital.

SDDL *Scalable Data Distribution Layer*.

SDPEU *Sensor Data Processing and Enhancement Unit*.

SEM *Sensory Effect Metadata*.

SOUPA *Standard Ontology for Ubiquitous and Pervasive Applications*.

SQL *Structured Query Language*.

TCP *Transmission Control Protocol*.

UML *Unified Modeling Language*.

UPnP *Universal Plug and Play*.

W3C *World Wide Web Consortium*.

WPAN *Wireless Personal Area Network*.

XHTML *eXtensible Hypertext Markup Language*.

Sumário

1 Introdução

Pode-se definir a Internet das Coisas – *Internet of Things* (IoT) – como a interconexão de dispositivos físicos (também conhecidos como *smart objects*) equipados de eletrônica, software, sensores, atuadores e conectividade de rede que permitem aos mesmos coletarem e trocarem dados, cooperando com pessoas e o ambiente [2, 22, 33], independentemente do tempo e do lugar [39].

A IoT atende diversos domínios de aplicações como, por exemplo, o domínio residencial, em que os *smart objects* permitem cenários como os de monitoramento (*e.g.* usando sensores de temperatura e *kinetics*) e controle do ambiente (*e.g.* usando lâmpadas *dimmer*, termostatos e auto-falantes). Além disso, atualmente nesses ambientes, dispositivos de TV estendem seu tradicional consumo audiovisual e permitem a execução de aplicações como navegação web, Social TV e jogos [8].

Diante disso, torna-se interessante propiciar efeitos sensoriais aos espectadores de TV utilizando recursos de IoT presentes em seu ambiente físico de apresentação com objetivo de aumentar o grau de imersão ao conteúdo assistido. Esse processo de estender um conteúdo multimídia trazendo a geração de efeitos sensoriais junto ao mesmo é conhecido por *Multisensory Media* (MulseMedia) [52]. Ainda, com auxílio dos *smart objects*, se torna interessante permitir que o conteúdo em exibição na TV seja alterado diante dos dados de contexto do ambiente físico ou diante de interações multimodais realizadas pelos telespectadores. Com isso, diversos cenários podem ser explorados, desde os que gerenciam alterações dos aspectos do ambiente físico diante de narrativas do conteúdo apresentado até os que adaptam o conteúdo em execução de acordo com os perfis dos telespectadores presentes no espaço físico.

Para tanto, é necessário exportar para a aplicação em execução na TV o conhecimento relativo aos serviços de sensores e atuadores presentes no espaço físico de apresentação. Adicionalmente, a percepção do ambiente e dos usuários nele presentes pode ser aumentada através do uso de sensores disponíveis em muitos aparelhos celulares. Desta forma, com esse conhecimento dos sensores, atuadores e dos telespectadores, por meio de seus respectivos aparelhos celulares, é possível promover

o aumento no nível de imersão do usuário com o conteúdo transmitido na TV, seja adaptando o conteúdo aos dados de contexto do ambiente físico ou aos dados dos perfis dos telespectadores, bem como possibilitando novas formas de interação desses telespectadores com essas aplicações de TV, ou alterando aspectos do espaço físico diante da narrativa do conteúdo em reprodução.

Nesse contexto, este trabalho explora o desenvolvimento de aplicações de TV que cooperam com *smart objects* em ambientes residenciais. Todavia, o desenvolvimento de uma infraestrutura de *software* que permita esse grau de cooperação entre a TV e os *smart objects* não é uma tarefa trivial. Diversos desafios devem ser superados, tais como:

- A descoberta dinâmica de *smart objects* e dos serviços que eles disponibilizam, dado que um *smart object* pode entrar e sair do ambiente físico de apresentação e cada um pode oferecer diversos serviços distintos;
- O suporte à heterogeneidade dos protocolos de comunicação dos *smart objects*, visto que os mesmos podem utilizar diversos protocolos de tecnologias de curto alcance, tais como como Bluetooth e ZigBee;
- A representação do ambiente físico de apresentação, já que se deve modelar os *smart object* nele presentes de forma a exportar esse conhecimento à aplicação em execução na TV;
- A identificação dos telespectadores e seus perfis, de forma a permitir que a aplicação de TV se adeque aos perfis de seus telespectadores;
- A sincronização da mídia em reprodução com os *smart objects* presentes no ambiente físico, permitindo assim o aumento de imersão do usuário ao conteúdo apresentado;
- Abstrações de programação para o desenvolvimento de aplicações de TV, visto que a interação com os *smart objects* (recebimento e envio de mensagens) deve ser estruturada conforme a representação do modelo conceitual que define o domínio residencial.

Para superar esses desafios, inicialmente, desenvolveu-se um modelo conceitual, a ontologia IoTTV-Ont, capaz de expressar os conceitos e as relações de

um ambiente residencial, incluído os *smart objects* e as pessoas presentes nele. Com a descrição do ambiente físico é possível determinar quem está no cômodo no qual a TV se encontra, bem como determinar qual o perfil dessa pessoa. Ainda, é possível identificar quais *smart objects* também estão presentes nesse ambiente físico. Após a definição do modelo conceitual, foi desenvolvida uma arquitetura de *software* baseada na perspectiva dos *middleware* de TV digital e de IoT que permite integrá-los. Foi utilizada a ontologia IoTTV-Ont nessa arquitetura de *software* para, além de oferecer o conhecimento do ambiente físico, permitir a interoperação entre esses *middleware* por meio de um vocabulário comum. Desse modo, a arquitetura de *software* não é dependente exclusivamente de um *middleware* de TV, podendo ser utilizados diversos *middleware*, como Ginga [47], MHP [15] e DASE [50], uma vez que basta o *middleware* de TV digital conhecer tal vocabulário para trocar dados com o *middleware* de IoT. Adicionalmente, a utilização da ontologia IoTTV-Ont resolve também o problema da heterogeneidade dos *smart objects*, visto que os *smart objects* descobertos pelo *middleware* de IoT podem ser tratados pelos componentes da arquitetura de *software* por meio de seus respectivos valores semânticos.

A arquitetura de *software* foi materializada em um protótipo que faz uso do Ginga-NCL, *middleware* declarativo para desenvolvimento de aplicações de TV terrestre e IPTV [47], e do M-Hub [53], *middleware* que permite a descoberta dinâmica, estabelecimento de conexão, acesso e distribuição de dados de/para *smart objects*. Entretanto, nosso protótipo possui implementados padrões de projeto, como o padrão *adapter*, que permite o desacoplamento do *middleware* de IoT, possibilitando a utilização de outros *middleware* de IoT, como MOSDEN [38] e AndroSIXTH [19]. Por fim, nossa proposta foi validada por meio de casos de uso, nos quais esse protótipo foi exposto a cenários em que foi possível perceber como os objetivos desse trabalho e os requisitos da arquitetura foram contemplados.

1.1 Objetivos

A pesquisa a qual se refere esta dissertação de mestrado tem como objetivo geral investigar a integração de aplicações de TV digital com dispositivos de IoT presentes no espaço físico de apresentação de forma a: (i) prover um maior nível de imersão do telespectador diante do conteúdo em reprodução através dos mecanismos

de atuação presentes no ambiente; (ii) permitir a adaptação do conteúdo televisivo de acordo com o contexto do ambiente e dos usuários nele presentes; (iii) permitir novas formas de interação multimodal dos usuários com as aplicações de TV digital.

Para tanto, os objetivos específicos desta pesquisa são:

- Definir um modelo conceitual que descreva o espaço físico de apresentação;
- Projetar a arquitetura de *software* capaz de sanar os desafios expostos neste trabalho;
- Desenvolver um protótipo correspondente a uma implementação concreta da arquitetura a ser projetada;
- Avaliar o protótipo por meio de casos de uso a fim de demonstrar como os requisitos estabelecidos neste trabalho são atendidos em cada cenário.

1.2 Estrutura da Dissertação

A organização desta dissertação é a seguinte:

- O Capítulo 2 apresenta uma breve fundamentação teórica sobre TV digital em que é mostrado algumas características dos sistemas de TV digital, como também explana sobre *middleware* de TV digital dando ênfase ao *middleware* da TV digital brasileira, o *middleware* Ginga.
- O Capítulo 3 expõe uma breve fundamentação teórica sobre IoT na qual aborda as características essenciais de um *middleware* para IoT, apresentando o *middleware* M-Hub e suas propriedades.
- O Capítulo 4 mostra um modelo conceitual, denominado IoTTV-Ont, que permite a modelagem de uma residência incluindo as pessoas presentes no domicílio e os *smart objects* presentes em seu interior.
- O Capítulo 5 apresenta a arquitetura de *software* proposta neste trabalho responsável pela integração dos *middleware* de IoT e de TV digital. Esse capítulo descreve cada componente presente na arquitetura, bem como ocorre a dinâmica

das comunicações entre os mesmos. Ainda, é apresentado nesse capítulo algumas abstrações de programação para o desenvolvimento de aplicações para TV que levem em consideração o uso da IoT.

- O Capítulo 6 mostra a implementação de um protótipo que corresponde a uma implementação concreta da arquitetura de *software* proposta neste trabalho.
- O Capítulo 7 expõe a avaliação realizada sobre o protótipo desenvolvido que consiste em demonstrações de casos de uso.
- O Capítulo 8 discute relevantes trabalhos relacionados com esta pesquisa. Nesse capítulo os trabalhos foram organizados em grupos e realizada uma comparação entre os trabalhos de cada grupo com a proposta descrita nesta dissertação.
- O Capítulo 9 apresenta as conclusões obtidas a partir desta pesquisa e apresenta trabalhos futuros que podem ser desenvolvidos a partir deste esforço inicial.

2 TV Digital

A TV digital possibilita estender o tradicional consumo audiovisual, permitindo a execução de diversas aplicações de TV como navegação web, Social TV e jogos [8]. Conteúdos de TV digital podem ser disponibilizados por diversos meios, desde os que fazem uso de protocolos de Internet até os que fazem uso de emissão de sinais digitais via satélites [46], a saber:

- TV digital aberta, terrestre ou não, é o sistemas que recebe sinais provindos de radiodifusoras das emissoras, para TV digital aberta terrestre, ou provindos de satélites. Em ambos, os usuários necessitam de equipamentos adequados para a captação dos sinais (*e. g.* conversores digitais ou *set-top-boxes*);
- *Internet Protocol Television* (IPTV) é a tecnologia que permite a reprodução de um programa televisível por meio da Internet utilizando o Protocolo de Internet – *Internet Protocol* (IP) – para tal finalidade no qual sua transmissão é realizada por meio de *streaming* de vídeo em que o fornecimento é realizado por meio de provedores de Internet;
- WebTV é a transmissão de programas de TV que utiliza *streaming* de vídeo baseado em IP. Nesse modelo de TV, o usuário, geralmente, pode realizar o *download* do conteúdo em reprodução;
- *Peer-to-Peer TV* (P2PTV) é uma arquitetura colaborativa que utiliza a Internet para realizar a transmissão dos conteúdos em que os usuários ajudam na transmissão e na difusão dos conteúdos audiovisuais a nós conectados à rede.

Em um sistema de TV digital há diversos módulos funcionais que implementam padrões de referências encarregados por demodular, decodificar áudio e vídeo, utilizar canal de retorno, dentre outros [46]. Diversas regiões pelo mundo a fora desenvolveram os padrões nos quais esses módulos funcionam. O Sistema Brasileiro de Televisão Digital (SBTVD)[1] é o padrão brasileiro para a TV digital aberta

[1] Também conhecido por *Integrated Services Digital Broadcasting Terrestrial* (ISDB-Tb)

que também é utilizado em diversos outros países, como o Peru, Argentina, Chile, Venezuela, Equador e Costa Rica. Por outro lado, os Estados Unidos utilizam o padrão *Advanced Television System Committee* (ATSC-T) para a TV digital aberta terrestre. Já alguns países da Europa utilizam o padrão *Digital Video Broadcasting - Terrestrial* (DVB-T).

Em sistemas de TV digital aberta terrestre é possível realizar compressões nos dados a serem transmitidos para os receptores[2] dos telespectadores, que por sua vez realiza o processo para descomprimir esses dados quando recebidos. Não somente, esses sistemas de TV digital permitem uma alta taxa de envio de dados, como por exemplo o SBTVD que permite uma taxa de envio de 19,3 Mbps. Uma alta taxa de transmissão somada à compressão dos dados a serem enviados aos receptores dos telespectadores, permitem enviar, junto ao conteúdo audiovisual principal, aplicações de TV. Assim, o receptor do telespectador pode receber e executar essas aplicações juntamente com o conteúdo em reprodução.

Diversas aplicações de TV[3] podem requisitar dos telespectadores informações para o enriquecimento de um programa televisivo. Exemplos dessas aplicações são aquelas destinadas a promover uma votação entre os telespectadores de um programa televisivo ou aquelas que permitem o telespectador realizar compras de determinados produtos durante a execução de um programa de TV. Em ambas as situações, as aplicações de TV necessitam fazer uso da Internet. Desse modo, o módulo denominado canal de retorno (também conhecido por canal de interatividade) presente nos sistemas de TV digital aberta terrestre, possibilita realizar tal acesso, permitindo que as aplicações de TV se comuniquem com serviços web.

Os módulos funcionais de um sistema de TV digital trabalham em conjunto para proporcionar melhores experiências quanto às imagens e aos áudios executados, pois devido à tecnologia aplicada a esses módulos é possível obter uma melhor qualidade na execução dos mesmos. Com os recursos da TV digital também é possível explorar novas experiências de interação do telespectador, uma vez que as aplicações

[2]Também conhecidos por "*set-top-box*" e "conversor digital".

[3]Nesta dissertação trabalharemos os termos "aplicação de TV", "aplicativo de TV" e "*software* de TV" para representar o *software* executável em um ambiente de execução de aplicações de um *middleware* de TV digital. Os demais termos relacionados à programação de TV serão tratados como sinônimos de "conteúdo multimídia reproduzido na TV"

de TV podem fazer uso do canal de interatividade da mesma. No entanto, desenvolver aplicações de TV para estes sistemas de TV requisitaria de conhecimento especializado dos desenvolvedores quanto à execução de cada módulo funcional, dado que os padrões de referência desses módulos podem variar entre os diversos tipo de sistemas de TV digital. Desse modo, faz-se necessário um mecanismo capaz de se comunicar com os módulos funcionais da TV digital e que ofereça meios para que aplicações de TV possam usufruir dos mesmos.

2.1 *Middleware* para TV Digital

Um *middleware* de TV digital abstrai as especificidades de cada módulo que implementa os padrões de referência presentes em uma TV digital e oferece ao usuário uma interface de programação de aplicações – *Application Programming Interface* (API) – para o desenvolvimento de aplicações. Há diversos *middleware* para TV digital, tais como: o *middleware Digital Television Application Software Environment* (DASE) [50] pertencente ao sistema norte-americano de televisão, o *middleware Multimedia Home Platform* (MHP) [15] do sistema de TV europeu, e o *middleware* Ginga, *middleware* do sistema brasileiro de TV digital [47] e recomendação *ITU Telecommunication Standardization Sector* (ITU-T) para IPTV [28].

Os *middleware* de TV digital, comumente, oferecem suporte a ambientes de execução de aplicações desenvolvidas sob linguagens declarativas e linguagens imperativas [46]. No entanto, é possível estender as linguagens declarativas utilizando uma linguagem não-declarativa para atender alguns comportamentos específicos da aplicação quando estes não são abrangidos pela linguagem declarativa [48].

2.1.1 O *Middleware* Ginga

O *middleware* Ginga é o *middleware* da TV digital brasileira. Ele abstrai as especificidades dos módulos presentes no padrão SBTVD, como os reprodutores de áudio e vídeo (MPEG-4 HE-AAC@L4 e H.264 HP@L4.0, respectivamente), e oferece meios para que as aplicações de TV possam consumir tais recursos. A Figura 2.1 apresenta os padrões de referência implementados pelos módulos funcionais do SBTVD e o seu *middleware* Ginga. Essa figura exibe a camada do *middleware* Ginga

sobreposta aos padrões de referência, abstraindo as especificidades desses padrões para as aplicações de TV.

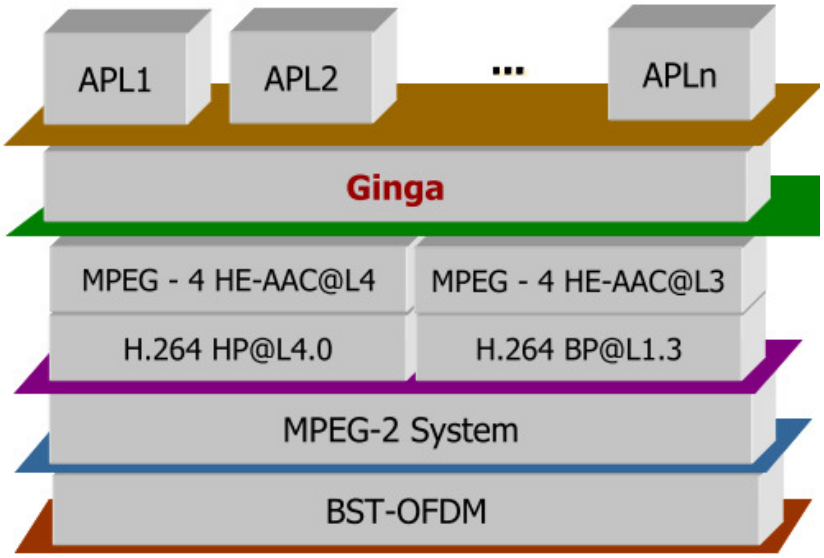


Figura 2.1: Padrões de referência do SBTVD com o *middleware* Ginga [46]

O Ginga oferece suporte à execução de aplicações de TV implementadas sob a liguagem declarativa *Nested Context Language* (NCL) (Ginga-NCL) e por meio da mesma oferece diversos suportes declarativos, tais como [46]:

- a edição de conteúdo em tempo de exibição que permite inserir, remover e alterar objetos de mídia durante sua reprodução;
- ao sincronismo de uma forma geral e, incluindo a interação do usuário com a aplicação de TV [45];
- a adaptação do conteúdo e como o mesmo é exibido, em que a adaptação pode ser realizada baseada em regras definidas pela aplicação ou pelas características do receptor, por exemplo;
- a exibição a diversos dispositivos, no qual se pode definir um *layout* de exibição para cada;
- e a "definição de relacionamentos de sincronismo espacial e temporal separado da definição do conteúdo dos objetos de mídia relacionados".

A linguagem NCL, diferentemente de outras linguagens declarativas (*e. g.* XHTML), permite a criação de marcações em um conteúdo multimídia independente

do tempo e do espaço em tela que a mesma ocupa. Em outras palavras, é possível utilizar NCL para definir que determinado conteúdo sofra algum efeito, como a sua interrupção de apresentação em um determinado instante dado em segundos ou permitir que um outro elemento de mídia seja posicionado em um determinado local da tela sobrepondo o conteúdo principal, por exemplo.

Por outro lado, como já dito, as linguagens declarativas podem não satisfazer por meio declarativo alguns comportamentos em uma aplicação de TV, sendo necessária a utilização de uma linguagem não-declarativa. A linguagem Lua é a linguagem que estende a linguagem NCL [48]. Ela, dentre outras características, é leve, portátil, facilmente embarcada (pois sua interpretação é realizada por uma biblioteca escrita na linguagem C), possui controle de memória e pode ser executada em múltiplas plataformas, pois é interpretada por *bytecodes* [41].

A linguagem Lua utilizada pelo Ginga-NCL (NCLua) foi adaptada para que sua aplicação fosse possível no ambiente de execução de TV digital. Nessa adaptação foram adicionados diversos módulos que permitem: a troca de dados entre documentos NCL e os *scripts* Lua (módulo *event*); a realização de desenhos livres na tela (módulo *canvas*); a persistência de dados (módulo *persistence*); dentre outros. No entanto, uma das principais diferenças entre um documento Lua puro para um documento NCLua é o fato deste último ser controlado por um documento NCL [11].

O módulo *event* é responsável por permitir, além da comunicação entre o *script* Lua e o documento NCL, interação com entidades externas à aplicação, como acesso ao canal de retorno e reconhecimento de eventos provindos do controle remoto. A Figura 2.2 mostra um objeto NCLua circundado por entidades geradoras de evento nos quais ele pode interagir. Os eventos são: eventos provindos da emissora; eventos gerados pelo usuário por meio do controle remoto; eventos oriundos do canal de interatividade; e os eventos gerados pelo formatador NCL. Os eventos a serem executados por esse objeto são enfileirados por ordem de chegada e tratados pelos seus respectivos tratadores de eventos implementados no *script* Lua [46].

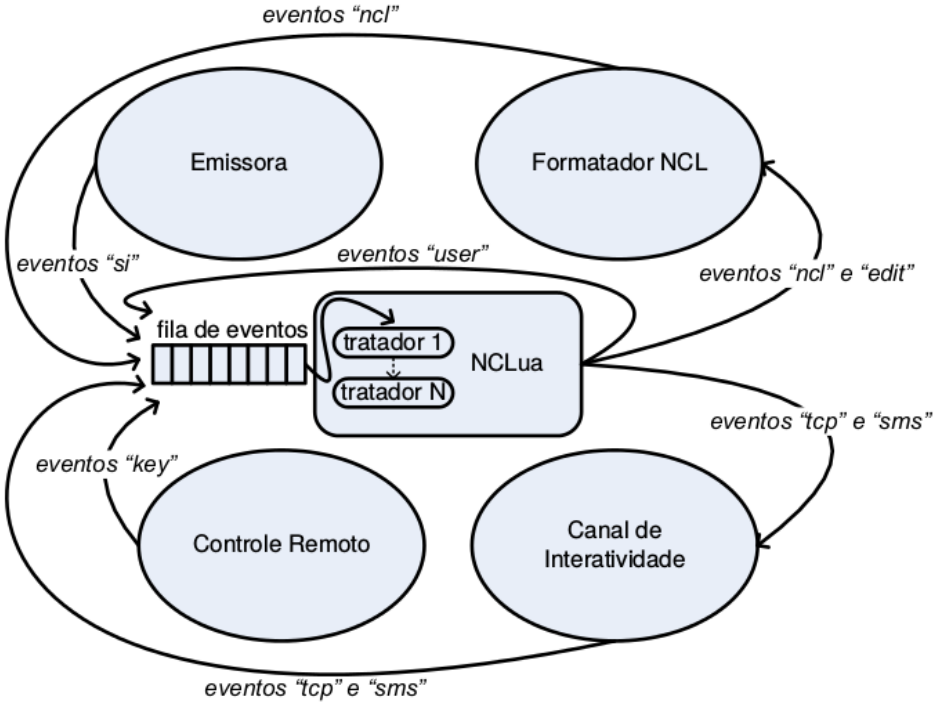


Figura 2.2: Relação entre um elemento NCLua e as entidades que podem gerar eventos [46]

2.2 Conclusão

Neste capítulo foi realizada uma breve apresentação sobre a TV digital, desde a apresentação do sistema de TV digital até a apresentação de *middleware* para a mesma. Quanto aos sistemas de TV digital foram abordados aspectos como a taxa de envio de dados pelas emissoras de TV digital aberta terrestre, que no SBTVD é de 19,3 Mbps, e a presença do canal de retorno nos receptores do sinal de TV digital, que permite o acesso à Internet às aplicações de TV.

A presença de um *middleware* para abstrair as especificidades dos módulos funcionais que compõe o sistema de TV digital aberta terrestre foi também apresentada nesse capítulo. Ainda, foi realizada uma breve apresentação do *middleware* Ginga e suas características quanto ao ambiente de execução de aplicações de TV, bem como algumas das propriedades presentes nas linguagens declarativas (NCL) e não declarativa (Lua) que permitem o desenvolvimento dessas aplicações.

3 Internet das Coisas

Pode-se definir a Internet das Coisas – *Internet of Things* (IoT) – como a interconexão de dispositivos físicos (também conhecidos como *smart objects*) equipados de eletrônica, *software*, sensores, atuadores e conectividade de rede que permitem aos mesmos coletarem e trocarem dados, cooperando com pessoas e o ambiente [2, 22, 33], independentemente do tempo e do lugar [39, 51]. Os *smart objects* possuem endereços exclusivos que permitem a troca de dados por meio de protocolos de comunicação, possibilitando o acesso aos recursos ofertados pelos mesmos por meio da Internet [22] ou por meio de tecnologias de comunicação de curto alcance [53]. Desta forma, a IoT integra aspectos e tecnologias de diferentes áreas, tais como a computação ubíqua, a ciência de contexto, os protocolos e tecnologias de comunicação, as redes de computadores e os *smart objects*.

A IoT atende diversos domínios de aplicações como, por exemplo, o domínio residencial, em que os *smart objects* permitem cenários como os de monitoramento (*e.g.* usando sensores de temperatura e *kinetics*) e controle do ambiente (*e.g.* usando lâmpadas *dimmer*, termostatos e auto-falantes). Nesse domínio é comum a presença de dispositivos pessoais (*e. g. smartphones* e *tablets*), nos quais podem fazer parte da IoT uma vez que, geralmente, possuem sensores capazes de capturar dados do ambiente no qual se encontram, além de possuirem poder de processamento e permitirem meios de comunicação para a troca de dados, seja com demais *smart objects* ou com a Internet.

Espera-se que haja bilhões de *smart objects* presente na IoT, em que cada um possui suas especificidades, como serviços ofertados e protocolos de comunicação, por exemplo. Para realizar uma comunicação com essa enorme quantidade e variedade de *smart objects* que fazem parte da IoT é necessária a existência de uma infraestrutura que os suportem [3]. Desse modo, um *middleware* para IoT capaz de se conectar aos *smart objects* e permitir meios de acesso aos mesmos se faz necessário.

3.1 *Middleware* para IoT

Um *middleware* para IoT permite que as especificidades técnicas dos *smart objects* sejam abstraídas possibilitando aos *software* fazerem uso dos serviços desses *smart objects* por meio de APIs. Diversos *smart objects* possuem tecnologias de curto alcance para se comunicarem. Porém, alguns não oferecem suporte à conexão com a Internet, o que impossibilita, de forma direta, o envio de seus dados para serviços em nuvens, por exemplo. Desse modo, um *middleware* para IoT pode atuar como ponte para esses *smart objects* publicarem seus dados na Internet [4, 57].

Para atender as necessidades da IoT, um *middleware* deve satisfazer alguns requisitos essenciais, como: ser escalável, com capacidade de atender inúmeros *smart objects*, bem como descobrí-los e gerenciá-los; oferecer suporte à heterogeneidade de protocolos de comunicação que os *smart objects* possuem; ser capaz de gerenciar o volume de dados que uma rede de IoT porventura produza; garantir a interoperação entre os *smart objects* e as aplicações que consomem seus serviços; e possuir implementações que garantam a segurança e a privacidade das informações dos *smart objects* [3, 9, 34]. A Figura 3.1 exibe as funcionalidades requisitadas para um *midleware* de IoT em um modelo de camadas em que a camada mais ao centro mostra as funcionalidades fundamentais de um *middleware* para IoT. Por outro lado, a camada mais externa aponta as funcionalidades não essenciais ao *middleware*, porém importantes, pois podem oferecer ao *middleware* de IoT serviços de processamento e armazenamento de dados, dentre outros [3].

Dispositivos de IoT se fazem presentes em diversos cenários. Dentre eles há cenários em que os *smart objects* são móveis – chamados de *Internet of Mobile Things* (IoMT), como no domínio residencial com o monitoramento de dados médicos coletados de sensores, geralmente, atrelados aos moradores e com isso pode-se haver uma locomoção constante desses equipamentos pela residência [18, 44]. Ainda, os dispositivos portáteis (*e. g. smartphones* e *tablets*) também podem ser transportados constantemente dentro da casa. Diante desses cenários, o *middleware* de IoT deve oferecer suporte à mobilidade desses *smart objects*, posto que os *smart objects* podem ora estar no raio de cobertura do *middleware* de IoT e ora estar fora do alcance do mesmo [35].

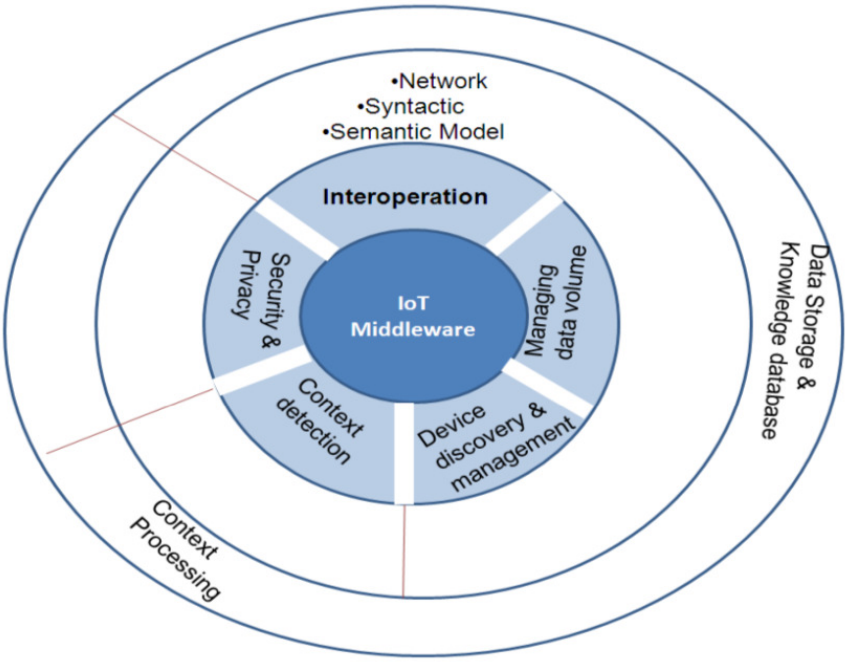


Figura 3.1: Funcionalidades de um *middleware* para IoT [3]

3.1.1 O *Middleware* M-Hub

O *middleware* M-Hub [53] é um *middleware* de uso geral (executado em um dispositivo móvel pessoal convencional) capaz de descobrir *smart objects* próximos e permitir a comunicação *unicast*, *broadcast* e *multicast* com os mesmos. O M-Hub é um nó especial na rede que atua como um *gateway*, em que os objetos se conectam para enviar dados capturados do ambiente físico para a Internet. Ainda, o M-Hub realiza a coleta de dados dos sensores internos ao dispositivo móvel no qual está em execução, que podem ser utilizados para enriquecer os dados dos *smart objects* (*e. g.* atribuindo dados de localização do *Global Position System* (GPS) aos dados dos *smart objects*) ou fornecendo esses dados dos sensores internos do dispositivos móveis para serem consumidos por alguma aplicação (*e. g.* aplicações que fazem uso de dados de acelerômetro ou giroscópio).

Esse *middleware* de IoT oferece suporte aos *smart objects* móveis presentes no ambiente, como também oferece suporte à leitura dos dados de sensores do próprio dispositivo onde está sendo executado. Esse suporte à mobilidade é útil em cenários em que ou os *smart objects* ou o *gateway* (M-Hub) ou ambos são móveis. Desse modo, pode-se ter cenários onde os *smart objects* são estáticos, enquanto o M-Hub é móvel no qual se desloca no ambiente, se aproximando dos *smart objects* e se conectando para

realizar uma comunicação de forma oportunista. Há também cenários em que o M-Hub não está se movimentando, enquanto os *smart objects* se deslocam no ambiente, aproximando-se ou afastando-se do mesmo. Por fim, há cenários em que ambos equipamentos estão em movimento [18].

A Figura 3.2 apresenta a arquitetura que compreende o M-Hub, na qual é possível visualizar os principais componentes desse *middleware*. A descoberta de serviços, no M-Hub, é realizada por um protocolo de comunicação genérico e que independe de tecnologia de comunicação, denominado *Short-Range Sensor, Presence and Actuation* (S2PA). Este serviço oferece uma API para utilização dos serviços ofertados pelos *smart objects*, que por sua vez podem usar diferentes tecnologias de comunicação de curto alcance, oferecendo assim suporte à heterogeneidade de protocolos de comunicação. O S2PA foi desenvolvido para, além de descobrir os serviços dos *smart objects* próximos, se conectar e poder trocar dados com os mesmos, recebendo dados dos sensores e enviando comandos aos atuadores, além de gerar notificações sobre *smart objects* que se desconectam. Outra característica do M-Hub é a gestão do consumo energético, que pode alterar a quantidade de consultas realizadas pelo S2PA no ambiente diante de uma baixa reserva energética no dispositivo, por exemplo.

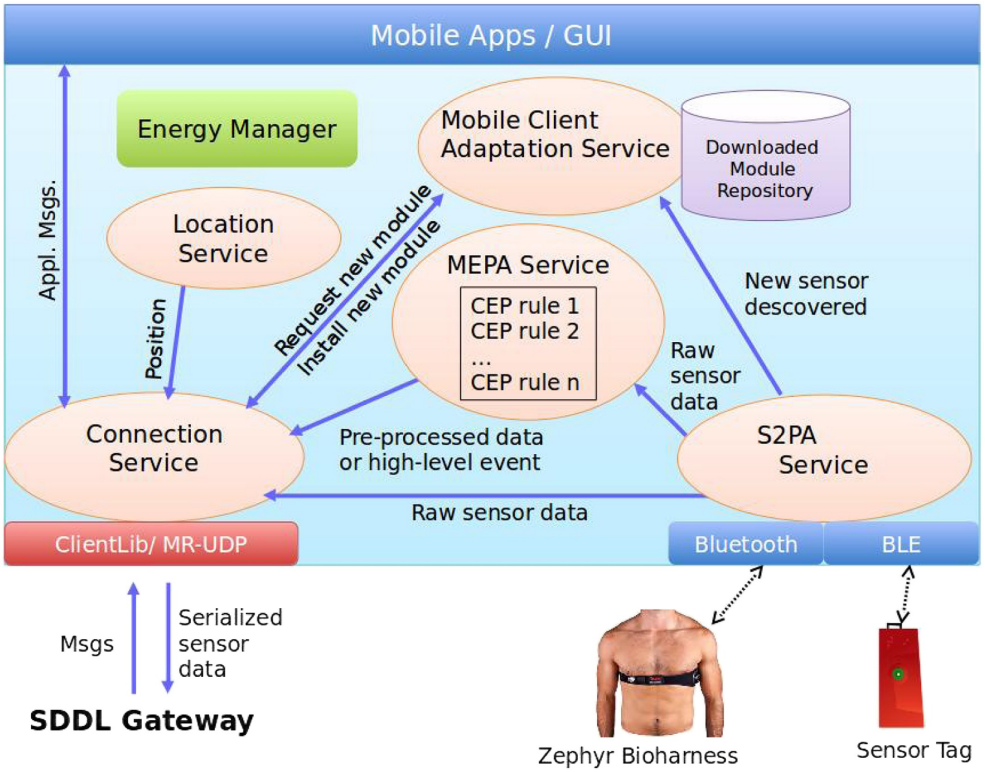


Figura 3.2: Principais componentes do M-Hub [18]

Para o S2PA se comunicar com o *smart object* descoberto no ambiente, o mesmo deve possuir o *driver* do *smart object*. Esse *middleware* de IoT permite o *download* desses *drivers* em tempo de execução por meio de um serviço em execução em uma nuvem *Scalable Data Distribution Layer* (SDDL) no qual o componente *Mobile Client Adaptation Service* gerencia a utilização dos mesmos após serem baixados e armazenados no *Downloded Module Repository*. O *MEPA Service* permite uma análise contínua sobre os fluxos de dados lidos dos *smart objects* em busca de identificar padrões definidos pela aplicação móvel. O componente *Location Service* permite enriquecer os dados dos *smart objects* lidos, atribuindo aos mesmos dados referentes à localização. Esse *middleware* de IoT permite uma comunicação com a Internet por meio do componente *Connection Service* em que uma biblioteca denominada *ClientLib* é utilizada para estabelecer a comunicação, via *Mobile Reliable UDP* (MR-UDP), com um *gatway* SDDL.

3.2 Conclusão

Este capítulo exibiu um breve fundamental teórico sobre IoT no qual foi apresentada a necessidade de utilizar um *middleware* para abstrair as particularidades dos *smart objects* oferecendo meios de acesso aos mesmos, permitindo consumir os dados provindos dos sensores e enviar comandos aos atuadores. Também foram discutidos características do *middleware* para IoT denominado M-Hub em que foram abordados o funcionamento dos seus principais componentes. Dentre eles, o S2PA é o componente responsável por realizar a descoberta e a comunicação com os *smart objects* próximos, utilizando tecnologias de curto alcance.

4 Modelo Conceitual

A descrição de um ambiente residencial permite que diversas ferramentas tecnológicas possam estar ciente da configuração física desse espaço. Deste modo, a modelagem do ambiente no qual a TV está localizada, bem como os *smart objects* alocados nesse mesmo cômodo, permitem às aplicações de TV estarem ciente do espaço físico de apresentação, podendo assim utilizar os recursos oferecidos pelos *smart objects* presentes de maneira pontual, por exemplo, acionando apenas os atuadores presentes na sala de jantar.

Adicionalmente, a descrição das pessoas e dos seus respectivos perfis presentes nesse ambiente permite que sistemas de *software* possam ser ajustados aos mesmos. Mais precisamente, pode-se utilizar dados relativos às pessoas para personalizar e gerenciar o ambiente físico, como alterar a intensidade da iluminação ou regular o volume do som produzido por alto-falantes. Além disso, esses indivíduos podem interagir com o *software* enviando comandos que alteram seus estados. Exemplos dessas interações são aquelas geradas por um telespectador por meio de um controle remoto para alterar o canal de TV. Portanto, a descrição das interações realizadas pelas pessoas com os *software* permitem aos mesmos identificarem interações das mesmas sem ambiguidade.

Para expressar um ambiente doméstico foi desenvolvida uma ontologia de domínio, visto que "uma ontologia é uma lógica que explica o significado pretendido de um vocabulário formal, ou seja, seu compromisso ontológico com uma conceituação particular do mundo" [21]. Assim, ela especifica formalmente conceitos e relações de um domínio [20].

Além da possibilidade de modelar domínios particulares com uma ontologia, ela oferece benefícios adicionais para ferramentas de *software*, tais como: auxílio em soluções para questões de heterogeneidade dos *smart objects*, dado que a ontologia oferece valores semânticos sobre os mesmos, ao invés dos valores atribuídos por seus fabricantes para cada serviço; e a interoperabilidade entre ferramentas tecnológicas (*e.g.*, *software* e *middleware*), visto que os termos expressos em uma

ontologia podem ser utilizados como vocabulário comum entre as partes que trocam mensagens.

4.1 A Ontologia IoTTV-Ont

Neste trabalho foi desenvolvida uma ontologia denominada IoTTV-Ont, que permite representar residências com as pessoas e os *smart objects* presentes nelas. O desenvolvimento dessa ontologia foi realizado utilizando o método *Development 101* [36]. Esse método de desenvolvimento de ontologias possui uma sequência de passos para sua completude, são eles: determinar o domínio e o escopo da ontologia; considerar a reutilização de ontologias existentes; enumerar termos importantes para a ontologia; definir a estrutura hierárquica das classes; definir as propriedades; definir as restrições; e gerar as instâncias.

Para o desenvolvimento da IoTTV-Ont, o ambiente residencial foi considerado como o domínio da ontologia. A finalidade de seu desenvolvimento foi permitir interoperabilidade entre os *middleware* de TV digital e IoT para aumentar o nível de imersão dos telespectadores a um conteúdo assistido, bem como permitir que os telespectadores sejam identificados e possam interagir com esses conteúdos.

Diante disso, foram considerados alguns conceitos necessários para atender os requisitos impostos pela arquitetura de *software* proposta neste trabalho (que são devidamente apresentados no Capítulo 5), a saber: conceitos que definem os cômodos presentes em uma casa devido sua modelagem permitir o gerenciamento de *smart objects* de forma mais precisa, permitindo identificar os *smart objects* que estejam na sala ou no quarto, por exemplo; conceitos que especifiquem os *smart objects*, pois assim é possível identificar os *smart objects* pela sua semântica e não pelo seu identificador definido por seu fabricante; conceitos que definem pessoas e seus perfis, visto que é objetivado neste trabalho possibilitar que o conteúdo seja adaptado aos telespectadores presentes no ambiente físico; e conceitos que permitam definir interações multimodais por parte dos telespectadores para com a aplicação em execução na TV.

Foi realizada uma pesquisa por ontologias já desenvolvidas que conceituam o ambiente residencial, bem como os indivíduos presentes nele. Assim, diversas ontologias foram elencadas, tais como DogOnt [5], DomoML [49], *Context-Based Digital*

Personality (CBDP) [29] e *Standard Ontology for Ubiquitous and Pervasive Applications* (SOUPA) [10]. Diversos aspectos dessas ontologias puderam ser aproveitados na IoTTV-Ont nos quais optamos por reutilizar alguns termos de classes das mesmas com intuito de reaproveitar os conceitos e suas relações, além de suas hierarquias. A representação da IoTTV-Ont foi feita utilizado o modelo *Web Ontology Language* (OWL) recomendado pelo *World Wide Web Consortium* (W3C) [54].

Dentre essas ontologias, a DogOnt e a DomoML possuem conceitos que definem uma residência desde o seu firmamento até os *smart objects* presentes nela. Ambos oferecem meios para descrever funcionalidades dos *smart objects* independentemente da tecnologia utilizada pelos mesmos, oferecendo assim suporte a esse tipo heterogeneidade. A diferença entre essas ontologias está associada ao modo como são representadas as funções de cada *smart object*, uma vez que a DogOnt utiliza o modelo descritivo para especificá-las, ou seja, são definidas as funcionalidades e, posteriormente, associadas aos *smart objects*. Já a DomoML possui uma abordagem composicional em que as funcionalidades são derivadas da composição de outras funcionalidades. No desenvolvimento de nossa ontologia optamos por utilizar o modelo descritivo presente na DogOnt.

Desta forma, para especificar os *smart objects* com suas funções e seus estados, foram utilizados os termos de classes *Controllable*, *Functionality* e *State*, respectivamente, da DogOnt. A Figura 4.1 apresenta a ontologia IoTTV-Ont em que é possível visualizar suas principais classes, inclusive as classes que definem os *smart objects*. A classe *State* representa o estado em que um *smart object* se encontra em determinado instante. Um estado pode ser de valor contínuo ou discreto, assim essa classe possui duas subclasses: *ContinuousState* e *DiscreteState*. A classe *Functionality* descreve as funções que um *smart object* possui, que podem ser: funcionalidade de notificação (*NotificationFunctionality*), que é a capacidade de um *smart object* notificar seu estado de forma autônoma; funcionalidade de consulta (*QueryFunctionality*), que permite que um *smart object* seja consultado a qualquer momento no qual o resultado da consulta é seu estado; e funcionalidade de controle (*ControlFunctionality*), que permite o envio de comandos para os *smart objects* atuarem no ambiente. Já a classe *Controllable* possui subclasses, são elas: *Appliances*, *Actuator* e *Sensor*. Em *Appliances* foi adicionado a subclasse denominada *Portables* que representa todo dispositivo de computação móvel que pode ser transportado com facilidade (*e. g. laptops,*

smartphones e *tablets*). Para descrever os conceitos do ambiente físico (*e. g.* sala e quartos) foi utilizado o termo de classe *Environment* que é equivalente à classe *Building Environment*, presente em DogOnt e DomoML.

Já as ontologias CBDP e SOUPA possuem conceitos que especificam as pessoas presentes em um ambiente. A CBDP representa conceitualmente, além de pessoas, preferências das mesmas quanto as configurações de aplicações que fazem seu uso. Já a SOUPA define as "pessoas" de forma equivalente ao conceito "Person" presente na ontologia FOAF[1] [6]. A FOAF, por sua vez, fornece mecanismos básicos que permitem especificar perfis de pessoas e seus relacionamentos. Essa ontologia é bastante utilizada em sites de relacionamento possibilitando a troca de dados dos seus usuários entre esses sites. Devido a FOAF ser uma ontologia pluralmente utilizada na Internet, foi adotada a abordagem utilizada em SOUPA de reutilizar o FOAF, para expressar na IoTTV-Ont os conceitos relacionados aos indivíduos presentes em uma residência. Desta forma, o termo de classe *Person*, equivalente ao definido no FOAF, foi utilizado para descrever o perfil das pessoas presentes no ambiente físico. A classe *Person* é uma subclasse de *Agente* e diferente da classe *Group*, que por sua vez, representa um grupo de pessoas.

Para identificar qual pessoa está presente em determinado ambiente foi pressuposto que cada dispositivo portátil pertence a um telespectador. Consequentemente, quando detectado algum dispositivo portátil, assume-se a presença do telespectador proprietário do mesmo. Para tanto, foi definido uma relação do tipo *hasPortables* da classe *Person* com a classe *Portables*. Essa relação define um proprietário (instância da classe *Person*) para cada dispositivo portátil (instância da classe *Portables*). Ainda, a classe *Portables* possui uma propriedade denominada *Id*, do tipo `literal`, que distingue suas instâncias.

No escopo deste trabalho é especificado que as interações dos telespectadores com a aplicação em execução na TV devem ser realizadas por meio do uso dos sensores internos dos dispositivos móveis (*e. g.* sensor de acelerômetro e giroscópio). Essas interações, na IoTTV-Ont, são descritas utilizando conceitos estabelecidos em [56]. Nesse caso, os termos *Event* e *Modality*, bem como suas respectivas hierarquias de classes foram reutilizados, aproveitando também

[1]Dicionário de termos relacionados às pessoas que podem ser usados em dados estruturados (por exemplo: RDF, JSON-LD, Linked Data) [6].

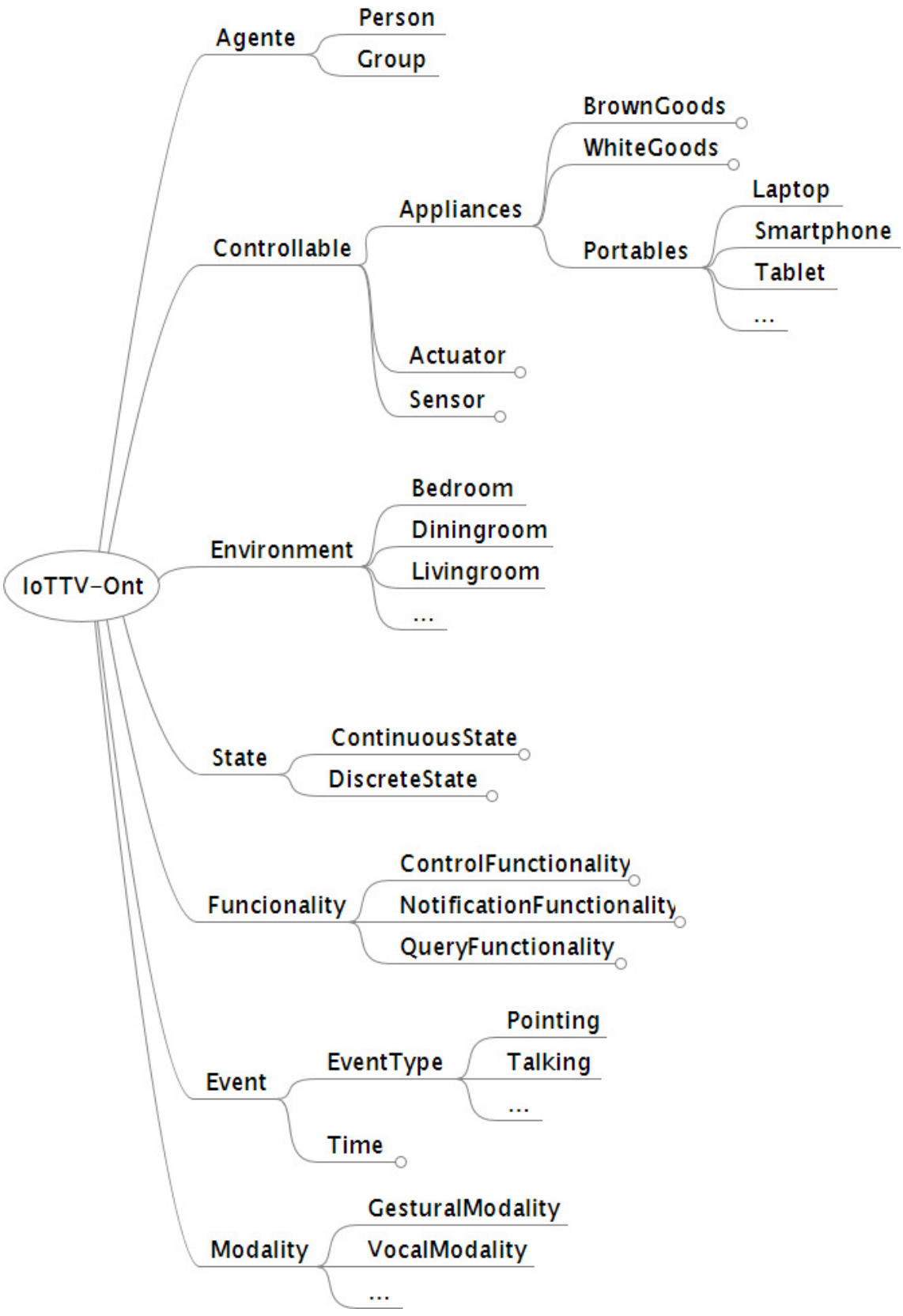


Figura 4.1: Principais classes da IoTTV-Ont

seus conceitos e relações. A classe *Modality* representa a forma na qual o usuário realiza a interação com a aplicação, seja ela gestual (*GesturalModality*) ou por voz (*VocalModality*), por exemplo. A classe *Event* descreve o evento gerado pela interação do usuário com a aplicação de TV. Essa classe possui subclasses que definem o tempo (*Time*) no qual se inicializa e finaliza o evento e o tipo do evento gerado (*EventType*) que podem ser *Pointing*, *Talking*, dentre outros. Entre *Modality* e *Event* há uma relação denominada *hasEvent* que associa qual evento é gerado por uma determinada modalidade. Assim, ao realizar um gesto com a mão (*GesturalModality*), por exemplo, um evento do tipo *Pointing* é associado.

Deste modo, fazendo o uso da IoTTV-Ont na proposta deste trabalho, é possível oferecer semântica aos dados trafegados na convergência entre IoT e TV digital, permitindo descrever cenários nos quais é possível identificar quem está no ambiente físico de apresentação e qual seu perfil, por exemplo. Dessarte, é possível adaptar o conteúdo apresentado e aspectos do ambiente físico conforme os perfis dos telespectadores. Por exemplo, em uma transmissão de um jogo de futebol no qual várias pessoas estão assistindo a este jogo em uma sala, se um time de futebol faz um gol, o ambiente pode ser ajustado para o perfil de torcedores desse time que estão presentes no ambiente físico, como também o próprio conteúdo em exibição pode perceber alterações, como, por exemplo, o surgimento de elementos de mídias sendo reproduzidos simultaneamente ao jogo durante a comemoração do gol realizado.

Outros cenários podem explorar alterações dos aspectos do ambiente físico de forma pontual, ou seja, é possível alterar somente um *smart object*, posto que cada *smart object* possui propriedades que os diferenciam. Também é possível alterar todos os *smart objects* pertencentes ao mesmo tipo (*e. g.* todas as lâmpadas da sala independentemente do fabricante). Ou ainda, é possível acionar somente as lâmpadas que possuem funcionalidades de alterar a cor de sua iluminação (*e. g. ColorDimmableLight*).

Portanto, a IoTTV-Ont permite a interoperação entre os *middleware* de TV digital e IoT, uma vez que é fornecido um vocabulário comum para troca de dados entre os mesmos por meio do seu uso. Também é possível identificar os perfis dos telespectadores presentes e descrever, semanticamente, suas interações realizadas por diversos modos para a aplicação de TV. Ainda, fazendo uso dessa ontologia, é possível

realizar anotações semânticas sobre os *smart objects* presentes no ambiente, oferecendo desta forma, suporte à heterogeneidade dos *smart objects*.

4.2 Conclusão

Esse capítulo apresentou possíveis problemas no qual o uso de uma ontologia pode auxiliar em soluções, como a heterogeneidade dos *smart objects* presentes em um ambiente físico, visto que cada *smart object* pode conter propriedades específicas, e a interoperabilidade o entre *middleware* de TV digital e de IoT, uma vez que a mesma pode servir como dicionário comum para a troca de mensagens, evitando ambiguidade nos significados desses dados para cada um deles.

Foi desenvolvido um modelo conceitual, a ontologia IoTTV-Ont, para representar residências, *smart objects* e pessoas presentes em seu interior. A ontologia foi implementada sob a linguagem OWL, utilizando a abordagem de desenvolvimento de ontologias *Development 101* e foi desenvolvida baseada em outras ontologias de domínios que permitem modelar um ambiente residencial, pessoas e seus perfis e os eventos gerados por interações multimodais dos indivíduos com uma aplicação.

5 Arquitetura de *Software* Proposta

Este capítulo tem por objetivo descrever uma arquitetura de *software* que permita a interoperação entre os *middleware* de TV digital e IoT. No entanto, a descrição dessa arquitetura independe de especificações concretas dos *middleware* para IoT (*e. g.* MOSDEN [38] e AndroSIXTH [19]) e de TV digital (*e.g.* MHP [15] e DASE [50]).

Para o desenvolvimento dessa arquitetura, foi adotada uma abordagem na perspectiva dos *middleware* de IoT e de TV digital que permite a abstração das especificidades técnicas tanto na IoT, quando, dentre outros aspectos, trata a heterogeneidade dos protocolos de comunicação dos *smart objects*, quanto na TV digital, abstraindo a utilização dos módulos funcionais dos *set-top-boxes*, como a utilização do canal de retorno, por exemplo. Dessarte, as aplicações de TV digital, podem, sem conhecer as particularidades dos *smart objects*, utilizar serviços dos mesmos, ou seja, podem consumir dados de sensores presentes no ambiente físico de apresentação e fazer uso de serviços de atuadores. Uma materialização dessa arquitetura é exposta no Capítulo 6, no qual é apresentado o desenvolvimento de um protótipo que faz uso dos *middleware* M-Hub e Ginga, para IoT e TV digital, respectivamente.

Para o desenvolvimento dessa arquitetura de *software* foram elencados alguns requisitos funcionais e não funcionais. Os requisitos funcionais foram listados da seguinte forma:

- **RF1.** A infraestrutura de *software* deve oferecer suporte a diversos protocolos de comunicação dada a heterogeneidade dos *smart objects* e suas tecnologias de comunicação, visto que é possível que diferentes *smart objects* possam estar presentes no mesmo ambiente físico de apresentação;
- **RF2.** A infraestrutura de *software* deve oferecer mecanismos para a descoberta dinâmica de serviços de *smart objects*, uma vez que esses *smart objects* podem entrar e sair da área de cobertura do *middleware* de IoT;

- **RF3.** A infraestrutura de *software* deve permitir a uma aplicação em execução na TV a possibilidade de alterar aspectos do ambiente físico (*e. g.* alteração da iluminação e temperatura) utilizando-se dos serviços dos *smart objects*.

- **RF4.** A infraestrutura de *software* deve permitir que a aplicação de TV se adapte com base nos dados de contexto do ambiente físico. Desta forma, os *smart objects* coletam dados do ambiente físico e a aplicação de TV os consomem podendo interferir no conteúdo audiovisual apresentado. Por exemplo, pode-se utilizar sensores de presença para informar a presença dos usuários no ambiente físico em que na ausência do mesmo o conteúdo possa ser interrompido.

- **RF5.** A infraestrutura de *software* deve permitir a aquisição de dados de sensores embutidos em dispositivo pessoais móveis, como *smartphones* e *tablets*, uma vez que esses dados podem ser utilizados para auxiliar em novos modos do telespectador interagir com a aplicação em execução na TV.

- **RF6.** A infraestrutura de *software* deve oferecer suporte à semântica, dado que os *smart objects* podem conter identificadores de serviços distintos para serviços semelhantes, ocasionados, geralmente, por seus diferentes fabricantes.

Quanto ao requisito não funcional, foi elencado o seguinte item:

- **RNF1.** A latência da comunicação entre a TV e os *smart objects* deve ser baixa. Isso se deve à sensibilidade de sincronismos de aplicações de TV, como por exemplo, uma aplicação para a TV que acione os *smart objects* de forma sincronizada com momentos de sua apresentação.

5.1 Visão Geral da Arquitetura

Para intermediar a comunicação entre as aplicações da TV com os *smart objects*, foi pressuposta a utilização de dispositivos portáteis (*e. g. smartphones* e *tablets*). Esses dispositivos permitem a execução de aplicações, além de possuírem tecnologias de comunicação, como Bluetooth, infravermelho e Wi-Fi. Ainda, esses dispositivos podem conter diversos sensores embutidos, como acelerômetro e giroscópio, que podem ser utilizados para interações multimodais dos telespectadores com a aplicação em execução na TV.

A comunicação da TV digital com o dispositivo portátil deve ser realizada utilizando o seu canal de retorno (que faz uso da pilha TCP/IP) e a conectividade Wi-Fi dos dispositivos móveis. Dessa maneira, a comunicação pode ser realizada por meio da Internet ou simplesmente por meio da rede local. Já a comunicação entre os *smart objects* e o dispositivo portátil deve ser realizada por meio das comunicações *Wireless Personal Area Network* (WPAN), como o *Bluetooth Classic* e *Bluetooth Low Energy* (BLE).

A Figura 5.1 mostra uma visão geral da arquitetura proposta, na qual é possível visualizar todos os elementos que a compreende, são eles: os dispostos no ambiente físico: *smart objects*; os compreendidos no dispositivo portátil: o *middleware* de IoT, o SDPEU, o *framework* para ontologias, a base de conhecimento, o *micro broker*, o motor CEP e a aplicação móvel; e os elementos localizados no *middleware* de TV Digital Interativa (iDTV): o CASMOC e a aplicação de TV. Nas seções a seguir cada um desses elementos é discutido.

5.1.1 *Middleware* de IoT e iDTV

Nesta arquitetura de *software*, a comunicação entre os *smart objects* e o dispositivo móvel é realizada por meio de um *middleware* de IoT, que também é encarregado de: oferecer suporte à heterogeneidade de protocolos e tecnologias de comunicação WPAN; realizar a descoberta dinâmica dos *smart objects* e de seus serviços; trocar dados com os *smart objects* por meio das tecnologias de curto alcance presentes no dispositivo móvel; gerar notificações quando houver desconexões desses *smart objects* e coletar dados dos sensores internos aos dispositivos portáteis dos telespectadores presentes no espaço físico de apresentação. Essas características de um *middleware* de IoT são essenciais para a infraestrutura de *software* notificar à aplicação na TV as interações realizadas pelos telespectadores, como também exportar a ciência de contexto do ambiente físico para a mesma. Quanto ao *middleware* de TV digital, é possível explorar, nesta arquitetura de *software*, qualquer um que ofereça um ambiente de execução de aplicaç de TV, seja desenvolvida sob linguagem declarativa ou imperativa, e que possua meios para utilização dos módulos funcionais da TV digital, como o acesso ao canal de retorno, por exemplo. Ainda, o *middleware* de TV digital deve oferecer meios para que se possa, por meio das aplicações de TV, interagir com as mídias em reprodução independentemente do seu tempo de execução ou do

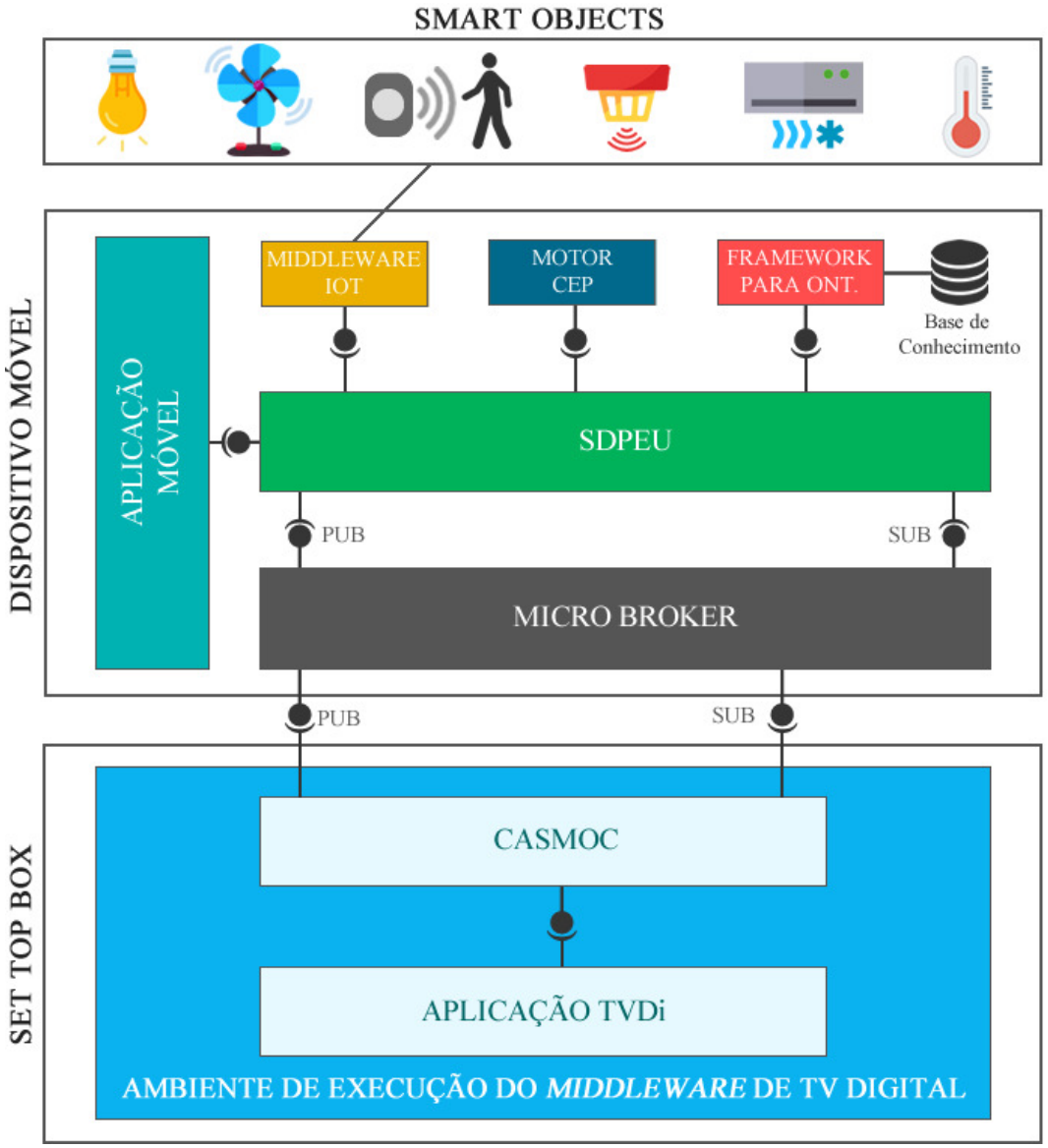


Figura 5.1: Visão geral da arquitetura de *software* proposta

seu espaço na tela. Com isso é possível associar o envio de comandos que alterem aspectos do ambiente físico a instantes da reprodução desse conteúdo multimídia.

5.1.2 *Framework* para Ontologias e Base de Conhecimento

A heterogeneidade dos *smart objects* pode limitar as ferramentas tecnológicas aos *smart objects* de determinados fabricantes, uma vez que os mesmos podem adotar propriedades específicas para seus produtos. Diante disso, optou-se por realizar notações semânticas sobre os *smart objects* e seus serviços descobertos pelo *middleware* de IoT. Desta forma, os componentes presentes em uma implementação concreta dessa arquitetura de *software* podem tratar os *smart objects* presentes no

ambiente por seus respectivos valores semânticos e não pelos valores de identificação adotados por seus fabricantes. Para tanto, esse registro semântico deve ser dado com base na ontologia IoTTV-Ont (Capítulo 4) devido à mesma permitir especificar os *smart objects* e o próprio ambiente físico no qual os mesmos se encontram. Desta forma, uma materialização dessa arquitetura de *software* oferece suporte a diversos *smart objects* independentemente do seu fabricante. A utilização dessa ontologia permite ainda a interoperação entre os *middleware* de IoT e da TV digital, uma vez que estabelece um dicionário comum entre eles para a troca de dados, evitando assim, ambiguidades na utilização dos serviços dos *smart objects*. Tanto a IoTTV-Ont, quanto as suas instâncias são compreendidas pela Base de Conhecimento na qual seu acesso ocorre por meio de um *framework* para ontologias. Esse *framework* para ontologias deve permitir consultas ao esquema da IoTTV-Ont, possibilitando assim a realização de consultas envolvendo as relações entre os conceitos estabelecidos na ontologia.

5.1.3 Motor CEP

Os sensores presentes nos dispositivos móveis (*e. g.* sensores de som, giroscópio, acelerômetro, toques, dentre outros) podem ser utilizados como meio para geração de dados em interações multimodais dos telespectadores com a aplicação em execução na TV. Contudo, esses sensores produzem um fluxo de dados que devem ser analisados em um período de tempo relativamente curto para não prejudicar o tempo de resposta ao telespectador, evitando assim prejuízos à imersão. Desta forma, optou-se por utilizar um motor de processamento de eventos complexos – *Complex Event Processing* (CEP) – para rastrear e analisar esses fluxos de dados. O *Complex Event Processing* (CEP) é destinado a analisar e controlar séries complexas de eventos inter-relacionados por meio de ferramentas e técnicas que o compreende [32]. A análise realizada pelo motor CEP sobre um fluxo de dados tem um custo de tempo próximo ao tempo real. Não somente, o motor CEP permite detectar situações durante o fluxo de dados, desde que essas situações estejam previamente definidas em forma de regras CEP. Desse modo, o CEP permite correlacionar fluxos de eventos com padrões de interesse, resultando em outros eventos complexos derivados dos eventos de entrada [13]. Em nossa arquitetura, o evento resultante da relação entre os fluxos de dados dos sensores internos ao dispositivo portátil e os padrões de interações predefinidos pelos

telespectadores deve ser o comando de interação multimodal do telespectador com a aplicação de TV.

5.1.4 *Sensor Data Processing and Enhancement Unit* - SDPEU

Foi conceituado o componente denominado *Sensor Data Processing and Enhancement Unit* (SDPEU), que interliga todos os outros componentes alocados no dispositivo portátil e faz uso de seus respectivos recursos. Mais precisamente, o SDPEU é encarregado de receber os dados provindos do *middleware* de IoT e anotá-los semanticamente fazendo uso do *framework* para ontologia e, em seguida, o mesmo deve armazenar esses dados em memória. No entanto, quando o fluxo de dados enviado do *middleware* de IoT para o SDPEU for de algum sensor interno ao dispositivo portátil, o mesmo, utilizando o motor CEP, deve buscar nesse fluxo de dados interações dos telespectadores com a aplicação de TV. Também é tarefa do SDPEU manter atualizado em memória os dados dos *smart objects* descobertos, ou seja, sempre que novos valores dos dados de serviços dos *smart objects* são lidos pelo *middleware* de IoT, o SDPEU ao recebê-los deve atualizar os dados mantidos em memória com esses novos valores. O SDPEU deve oferecer uma interface de programação de aplicação (API) para que se possa utilizar em aplicações móveis. O componente SDPEU também é incumbido de exportar os dados de contexto do ambiente físico de apresentação para a aplicação de TV, que resume-se em publicações dos dados mantidos em memória em um *micro broker*.

5.1.5 *Micro Broker*

O *micro broker* é o componente responsável por permitir a troca de dados entre o SDPEU e a aplicação de TV, uma vez que intermedia essa comunicação recebendo os dados de uma das partes e os enviando para as aplicações que possuem interesse sobre esses dados. Esse *micro broker* deve ser baseado no paradigma publicador-subscritor devido esse modelo permitir o desacoplamento das partes que o utilizam, possibilitando que os mesmos não necessitem estabelecer uma relação de comunicação direta entre si [14]. Nesse paradigma, os subscritores possuem a capacidade de expressar seu interesse em um evento, enquanto os publicadores

são encarregados de registrar tais eventos. Dessa forma, todos os subscritores são notificados sempre que um evento, no qual possuem interesse, for registrado por um publicador [1]. Portanto, é atribuição do *micro broker* gerenciar o recebimento e o envio dos dados trafegados por ele.

A utilização do *micro broker* nesta arquitetura permite que diversas aplicações de TV possam ser subscritoras em um mesmo *micro broker*, possibilitando assim o envio de dados do ambiente físico coletados por um *middleware* de IoT para todas essas aplicações de TV interessadas. Ainda, é possível que aplicações nos dispositivos portáteis presentes no ambiente físico possam ser subscritoras de tópicos de um mesmo *micro broker*, possibilitando também o recebimento de dados provindos da aplicação de TV, tais como *Electronic Programming Guide* (EPG).

5.1.6 *Comunication Abstraction and Smart Object Monitor Component* - CASMOC

Por fim, foi conceituado o componente que deve ser importado pela aplicação de TV, denominado *Comunication Abstraction and Smart Object Monitor Component* (CASMOC), que permite a mesma se comunicar com o *micro broker* possibilitando o envio e o recebimento de dados com o mesmo. Esse módulo deve possuir um sistema de armazenamento de dados para registrar os dados semânticos dos serviços e estados dos *smart objects* detectados no ambiente, atualizando-os sempre que novos valores forem lidos pelo *middleware* de IoT. Assim, as aplicações de TV possuem a ciência do contexto do ambiente físico na qual a TV está inserida. Além disso, o CASMOC deve fornecer abstrações de desenvolvimento de aplicações de TV, fornecendo uma API para que essas aplicações possam ser notificadas quando os dados de contexto do ambiente físico forem recebidos, como também deve permitir consultas sobre quais *smart objects* estão presente no ambiente ou quais telespectadores se encontram na sala, por exemplo, além de oferecer recursos para publicar no *micro broker* comandos de atuação no ambiente.

5.2 Interação dos Componentes para a Realização das Funcionalidades Previstas

Uma implementação concreta dessa arquitetura de *software* possibilita atender aos objetivos expressos no escopo deste trabalho, a saber: permitir que as aplicações de TV possam alterar aspectos do ambiente físico de apresentação; permitir que as aplicações de TV possam estar cientes dos dados de contexto do ambiente físico; e possibilitar que os telespectadores possam interagir com as aplicações de TV por diversos modos. Então, esta seção apresenta como os componentes da arquitetura proposta devem se comportar para atenderem esses objetivos.

5.2.1 Tornando a Aplicação da TV Digital Ciente do Espaço Físico de Apresentação

A fim de exportar para as aplicações de TV o conhecimento relativo ao contexto do ambiente físico no qual a TV está localizada, deve-se primeiramente descobrir quais serviços estão sendo oferecidos no respectivo ambiente. Para isso, o *middleware* de IoT se encarrega de realizar tais descobertas, já que o mesmo é responsável por realizar buscas por *smart objects* e seus serviços ofertados no ambiente físico. A Figura 5.2 apresenta um diagrama de sequência que mostra as etapas a serem seguidas pela infraestrutura de *software* para realizar a exportação desses dados às aplicações de TV. Após a inicialização do SDPEU (etapa 1) o mesmo dá início ao *micro broker* e se conecta com o mesmo (etapas 2–3). Em seguida o SDPEU dá inicio ao processo de descoberta de serviços do *middleware* de IoT (etapa 4). Por outro lado, a aplicação de TV, por meio do CASMOC se conecta e se subscreve no *micro broker*, registrando interesses por determinados *smart objects* (etapas 5–6). O serviço de descoberta do *middleware* de IoT deve permanecer ativo em segundo plano até que seja finalizado pelo SDPEU. Após a descoberta dos *smart objects* presentes no ambiente físico, devem ser coletados dados referentes aos mesmos, como o identificador do *smart object* e seus serviços ofertados, por exemplo (etapa 7). Esses dados são fornecidos para o SDPEU poder consumi-los. O SDPEU, por sua vez, deve utilizar a API do *framework* para ontologias a fim consultar informações semânticas sobre esses dados de *smart objects* em uma base de conhecimento (etapas 8–11). Com isso, o

SDPEU possui informações semânticas sobre os *smart objects* descobertos no ambiente físico. Posteriormente, essas informações são armazenadas localmente (etapa 12) e, em seguida, publicadas no *micro broker* (etapa 13). O armazenamento local dessas informações é realizado devido à necessidade de informar às aplicações de TV quais *smart objects* e quais serviços estão presentes no ambiente físico, caso a aplicação de TV tenha sido inicializada após essas descobertas. Portanto, esses dados armazenados devem ser atualizados sempre que novos *smart objects* forem detectados no ambiente, bem como quando esses perderem comunicação com o *middleware* de IoT, nesse último caso deve-se removê-lo do armazenamento local.

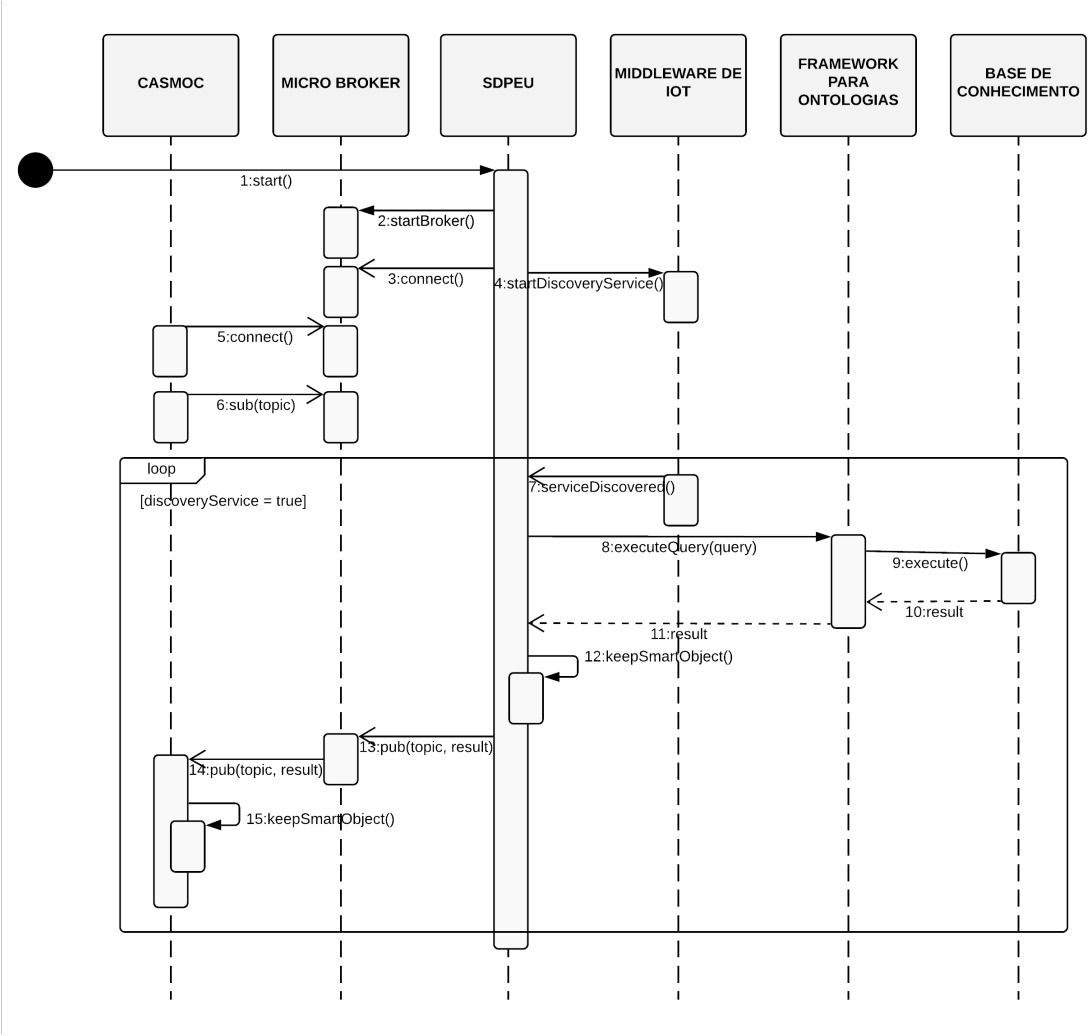


Figura 5.2: Diagrama de Sequência: envio de dados relativos aos *smart objects* à aplicação de TV

Quando o CASMOC recebe os dados semânticos dos *smart objects* publicados no *micro broker* pelo SDPEU (etapa 14) o mesmo deve armazenar, ou atualizar caso já possua armazenado, localmente essas informações (etapa 15). Com

isso, as aplicações de TV podem realizar consultas posteriores sobre os *smart objects* utilizando como fonte de buscas esse armazenamento local, uma vez que se mantém atualizado. Ainda, o CASMOC deve notificar à aplicação de TV sempre que novos valores dos *smart objects*, nos quais possui interesse, forem recebidos, possibilitando assim que essas aplicações possam reagir instantaneamente a esses novos dados.

5.2.2 Adaptação do Espaço Físico de Apresentação ao Conteúdo em Reprodução na TV

Para que uma aplicação possa alterar aspectos do ambiente físico de apresentação, a mesma deve ter ciência de quais *smart objects* estão presentes nesse espaço físico, visto que deve utilizar esses serviços dos *smart objects* para realizar tais alterações no ambiente. Deste modo, é necessário que se ocorra primeiro a descoberta dos *smart objects* e de seus serviços, conforme explanado na Seção 5.2.1.

A Figura 5.3 apresenta um diagrama de sequência que mostra como ocorrem os envios de comandos para alterar aspectos do ambiente físico a partir de aplicações de TV. Após a inicialização do SDPEU (etapa 1) e da inicialização e estabelecimento da conexão com o *micro broker* (etapas 2–3), a aplicação de TV, por meio do CASMOC, também deve se conectar ao *micro broker* (etapa 4).

A aplicação de TV deve fazer o uso da API do CASMOC para consultar em seu armazenamento local de *smart objects* os serviços disponíveis no ambiente físico (etapas 5–7). Também, utilizando a API do CASMOC, a aplicação de TV publica no *micro broker* comandos de atuação sobre os *smart objects* consultados no armazenamento local (etapas 8–9). O SDPEU recebe esses dados, uma vez que este já tenha registrado seu interesse sobre os dados provindos das aplicações de TV em sua inicialização (etapa 10). O SDPEU por sua vez, deve buscar em seu armazenamento local de *smart objects* os dados físicos do *smart object* no qual se deseja utilizar seus serviços (etapa 11) e, em seguida, o mesmo deve encaminhar comandos para o *middleware* de IoT acionar o respectivo *smart object* (etapas 12–13).

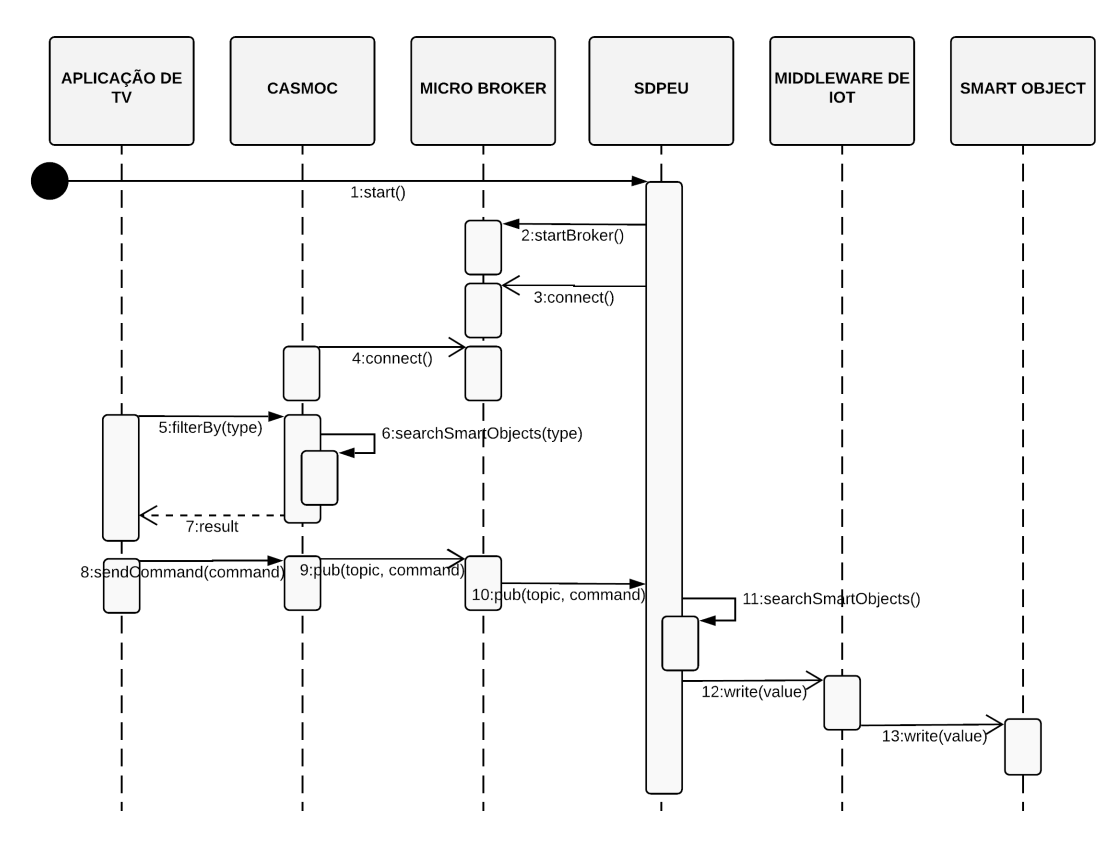


Figura 5.3: Diagrama de Sequência: envio de comandos de aplicações de TV para alterar aspectos do ambiente físico

5.2.3 Interação Multimodal a Partir da Arquitetura Proposta

Essa arquitetura de *software* permite que diversos telespectadores possam interagir com a aplicação de TV simultaneamente. Isso é possível devido um *micro broker* disposto em um dispositivo portátil permitir que os demais dispositivos portáteis presentes no ambiente físico publiquem no mesmo os dados de interações realizadas por seus respectivos telespectadores.

Para tal, faz necessário especificar qual dispositivo portátil presente no ambiente deve ser utilizado para intermediar a comunicação da aplicação de TV com os *smart objects* e com os demais dispositivos portáteis. Esse dispositivo intermediador é denominado de *master*, enquanto os demais dispositivos portáteis são denominados de *slaves*. Todos os comandos de interação realizados pelos telespectadores são enviados para o *micro broker* do dispositivo portátil *master*. A Figura 5.4 apresenta uma visão geral quanto ao envio de eventos de interações produzidas por diversos telespectadores para as aplicações de TV. A imagem exibe diversos dispositivos

portáteis *slaves* que produzem eventos de interações nas quais são enviadas ao dispositivo portátil *master*, que por sua vez os envia para as aplicações de TV.

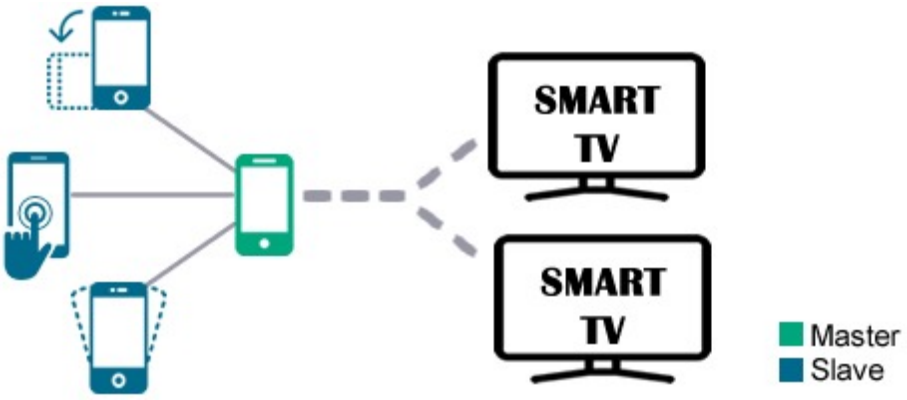


Figura 5.4: Visão geral das interações dos telespectadores com as aplicações de TV

Para a detecção dos eventos de interações realizados pelos telespectadores com a aplicação de TV é necessário que os telespectadores registrem antecipadamente os modos de interações que desejam utilizar para cada instância de eventos presentes na IoTTV-Ont. Cada SDPEU dos dispositivos portáteis *slaves* deve ser responsável por analisar, através do uso do motor CEP, o fluxo de dados dos sensores internos ao dispositivo portátil. Quando detectado algum comando nesse fluxo de dados compatível com alguma das interações registradas pelo telespectador, o SDPEU deve publicar o evento detectado (anotado semanticamente) em um tópico no *micro broker* do dispositivo *master*. Assim, a aplicação de TV, por meio do CASMOC, recebe esses dados já com uma notação semântica. A autonomia dos dispositivos portáteis *slaves* de analisar esses fluxos de dados ocorre devido às diversas maneiras de realizar um movimento, pois as possíveis escolhas de interações são relativas a cada telespectador. Por exemplo, um telespectador pode utilizar um *smartphone* para definir um movimento com a mão e com uma determinada velocidade no movimento, fazendo uso do sensor de acelerômetro presente no mesmo, no qual este movimento pode ser associado ao evento "Left", enquanto outro telespectador utilizando os mesmos sensores, no entanto com mais intensidade no movimento, pode gerar o mesmo evento de interação.

Portanto, tanto para os telespectadores poderem enviar eventos de interações multimodais, por meio dos dispositivo portáteis *slaves*, quanto para as aplicações de TV receberem esses dados de eventos dos telespectadores é preciso que ambos conheçam o endereço de IP do dispositivo portátil *master*. Desta forma,

diversas aplicações de TV em execução em diversas TV digitais podem consumir os mesmos dados de interações dos telespectadores presentes no ambiente. Como essa infraestrutura de *software* permite uma comunicação bidirecional entre as aplicações de TV e os dispositivos portáteis, a mesma pode enviar dados de contexto do conteúdo apresentado para os dispositivos portáteis presentes. Nesse caso, uma aplicação móvel que faça uso de uma implementação concreta dessa arquitetura pode oferecer meios para o telespectador selecionar em qual aplicação de TV o mesmo deseja enviar seus comandos de interação, quando houver mais de uma TV com aplicações diferentes presentes no ambiente físico.

5.3 Abstrações de Programação

A Seção 5.1 apresenta os componentes e suas respectivas responsabilidades dentro da arquitetura de *software* proposta. Dentre eles, apenas os componentes SDPEU e CASMOC devem oferecer meios para aplicações móveis e de TV, respectivamente, poderem interagir com os recursos oferecidos por uma implementação concreta dessa arquitetura.

O SDPEU deve oferecer uma API para aplicações móveis poderem, dentre outras funcionalidades, inicializá-lo e interrompe-lo; registrar padrões de interações dos telespectadores utilizando dados de sensores internos ao dispositivo portátil para com isso possibilitar a identificação, por meio do CEP, de eventos gerados por interações multimodais dos telespectadores; registrar dados de perfis dos telespectadores com base na IoTTV-Ont; e registrar o endereço de IP do dispositivo portátil *master*.

Por outro lado, o CASMOC também deve oferecer uma API para as aplicações de TV poderem, dentre outras funcionalidades, definir configurações do sistema, tal como o endereço de IP do dispositivo portátil *master*; expressar seus interesses por determinados *smart objects*, visto que nem sempre todos os *smart objects* presentes no espaço físico de apresentação são de interesse da aplicação de TV; receber dados referentes aos *smart objects* interessados; e receber dados relativos aos telespectadores, tais como seus respectivos perfis.

Para as aplicações de TV fazerem uso dos serviços dos *smart objects* presentes no ambiente, eles devem conhecer os valores semânticos referentes aos serviços dos mesmos, bem como os valores semânticos dos dados que esses *smart objects* produzem. Desta forma, deve-se utilizar como referência para o desenvolvimento das aplicações de TV os termos e relações presentes na ontologia adotada na arquitetura, a IoTTV-Ont.

Diante disso, este trabalho sugere abstrações para o desenvolvimento de aplicações de TV utilizando como base a ontologia IoTTV-Ont. Mais precisamente, os termos expressos em IoTTV-Ont e as relações que os mesmos possuem são utilizados para descrever os recursos a serem usados pela aplicação de TV, seja para consumir dados dos sensores ou acionar atuadores.

Para exemplificar as abstrações de programação sugeridas foi utilizada a linguagem Lua para escrever possíveis codificações utilizando o CASMOC em uma aplicação de TV para o *middleware* Ginga-NCL. As exemplificações são dividas em duas perspectivas: uma ilustrando como a aplicação de TV percebe o espaço físico e a outra mostrando como a aplicação de TV envia comandos para os *smart objects* atuarem no espaço físico de apresentação.

5.3.1 Percepção do Espaço Físico

Para uma aplicação de TV perceber o ambiente no qual a TV se encontra, a mesma, em sua codificação, deve especificar todos os *smart objects* que deseja estar ciente quando forem descobertos no espaço físico ou quando os seus dados forem lidos. Com isso, a aplicação de TV pode reagir conforme o recebimento desses valores. Ainda, deve-se definir quais ações se deve realizar diante das interações multimodais realizadas pelos telespectadores com a aplicação de TV. Em ambas as implementações, o desenvolvedor deve utilizar o padrão de projeto *observer* para definir *listeners* responsáveis por receber os valores enviados para o CASMOC e fornecer às aplicações de TV.

A Codificação 5.1 apresenta um exemplo em que é exposto o interesse em receber dados de sensores de temperatura presentes dentro de um quarto e também define ações a serem realizadas quando houver interação dos telespectadores com a aplicação de TV.

Código 5.1: Exemplo de código em Lua utilizando o CASMOC para o recebimento de dados

```
1  local CASMOC = require ("CASMOC")
2
3  local mqtt_settings = {}
4  mqtt_settings.host_master = '192.168.0.10'
5  CASMOC.setMqttSettings(mqtt_settings)
6
7  local temperaturesensor = {}
8  temperaturesensor.Environment = 'Bedroom'
9  temperaturesensor.Type = "TemperatureSensor"
10 function temperaturesensor:receiveNotification(NotificationFunctionality)
11  if(NotificationFunctionality['TemperatureMeasurementNotificationFunctionality']
12              .Notification['TemperatureMeasurementNotification'])then
13   local notification =
14     NotificationFunctionality['TemperatureMeasurementNotificationFunctionality']
15                  .Notification['TemperatureMeasurementNotification']
16   print("TemperatureSensor: "..notification
17                  .notificationParamName['temperatureValue'])
18  end
19 end
20
21 local eventsListener = {}
22 function eventsListener:receiveEvent(Portable, Events)
23  for i = 1, #Events do
24   print("The user "..Portable.Person.givenName
25                  .." generated the event: "..Events[i])
26  end
27 end
28 CASMOC.setSmartObjectsListener({temperaturesensor})
29 CASMOC.setEventsListener(eventsListener)
30 CASMOC.start()
```

Este exemplo de código apresenta na linha 1 o comando que realiza a importação do CASMOC à aplicação de TV. As linhas 3–5 especificam o endereço de IP do dispositivo portátil *master* no qual esta aplicação deve trocar dados. As linhas 7–19 definem o *smart object* que a aplicação de TV deseja receber suas notificações

(*e. g.* alteração de seu estado). Por espelhar a ontologia IoTTV-Ont, o CASMOC utiliza os termos definidos nessa ontologia para representar as propriedades dos *smart objects* nos desenvolvimentos das aplicações de TV. Mais precisamente, a linha 8 define qual o ambiente físico (*Environment*) se deseja limitar a receber dados de *smart objects*. Nessa codificação, deseja-se receber dados apenas dos *smart objects* presentes no quarto (*Bedroom*). Já a linha 9 especifica o tipo (*Type*) do *smart object* que se deseja receber dados (*e. g. TemperatureSensor*). Também, tanto o termo *Bedroom* (linha 8) quanto o *TemperatureSensor* (linha 9) são definidos na IoTTV-Ont. As linhas 10–19 foram codificadas para receber as notificações desse *smart object*. Desta forma, se houver uma notificação "`TemperatureMeasurementNotificationFunctionality`", os dados dessa notificação são impressos pela linha 16. Já as linhas 21–27 definem as ações a serem tomadas quando um evento realizado por alguma interação multimodal do telespectador com a aplicação é recebida. A linha 21 mostra um exemplo de função para receber eventos de interações gerados pelos telespectadores, o `receiveEvent` recebendo dois parâmetros: `Portable` e `Events`. Devido haver uma relação definida em IoTTV-Ont entre os dispositivos portáteis e os telespectadores é possível identificar qual telespectador realizou a interação com a aplicação de TV. Assim, a linha 24 imprime o nome do telespectador que realizou a interação, bem como o nome do evento gerado. A linha 28 submete ao CASMOC os *listeners* definidos nesta aplicação de TV para receber dados dos *smart objects* (padrão *observer*). Já a linha 29 submete o *listerner* que deve receber os eventos de interação realizadas pelos telespectadores com a aplicação de TV. Por fim, a linha 30 inicializa o componente CASMOC.

5.3.2 Atuação no Espaço Físico

Para uma aplicação de TV enviar comandos de atuação para um *smart object*, a mesma deve realizar uma busca no armazenamento local do CASMOC pelos serviços desejados, conforme apresentado na Seção 5.2.1. Posteriormente, caso haja o serviço desejado no ambiente físico, deve-se enviar os comandos conforme as funcionalidades de atuação do *smart object* expressas na IoTTV-Ont.

A Codificação 5.2 apresenta uma implementação, também escrita em Lua, que exemplifica a abstração de programação para o envio de comandos que altera a intensidade de iluminação do ambiente físico por meio do ajuste da intensidade de

todos os *smart objects* reguladores de tal aspecto (*DimmerLamp*) e o acionamento de todos os ventiladores (*Ventilators*) também presentes no ambiente.

Código 5.2: Exemplo de código em Lua utilizando o CASMOC para utilizar serviços de atuação dos *smart objects*

```
1  local CASMOC = require ("CASMOC")
2
3  local dimmerlamps = smartObjects:filterByType('DimmerLamp')
4  for k, dm in pairs(dimmerlamps) do
5    dm.Functionality.ControlFunctionality.LightRegulationFunctionality
6          .Command.SetCommand.realStateValue =  "15%"
7    CASMOC.postSmartObject(dm)
8  end
9
10 local ventilators = smartObjects:filterByType('Ventilator')
11 for k, v in pairs(ventilators) do
12   v.Functionality.ControlFunctionality.OnOffFunctionality
13         .Command.OnCommand.realStateValue = "on"
14   CASMOC.postSmartObject(v)
15 end
```

Esse fragmento de código especifica na linha 1 a importação do componente CASMOC à aplicação de TV, necessário para fazer uso de sua API. As linhas 3–8 especificam o gerenciamento do *smart object* que regula a intensidade da iluminação do ambiente físico (*DimmerLamp*). A linha 3 exibe uma função para realizar um filtro para buscar, dentre as instâncias armazenadas em memória pelo CASMOC, todas as instâncias de *smart objects* do tipo *DimmerLamp* presente no ambiente físico de apresentação. A linha 4 apresenta o início de um laço de repetição que tem finalidade de percorrer todas as instâncias de *smart objects* retornadas pelo filtro codificado na linha 3. As linhas 5–7 definem o novo valor de intensidade de iluminação de cada *smart object* que devem ser publicados no *micro broker*. Mais precisamente, a linha 7 realiza a chamada de um método `postSmartObject(SmartObject obj)` destinado a realizar esta publicação. Já as linhas 10–15 apresentam o acionamento de todos os *smart objects* do tipo *Ventilators* presentes no ambiente físico. Sendo a linha 10 responsável por realizar o filtro de todos os *Ventilators* detectados e armazenados no CASMOC. Um

laço de repetição especificado para percorrer todos os *smart objects* que satisfizeram o filtro é especificado na linha 11. Já as linhas 12–13 inserem novos valores para esses *smart objects*. Por fim, a linha 14 chama o método para publicar os dados de atuação do *Ventilator* no *micro broker*.

5.4 Conclusão

Neste capítulo foi apresentada uma arquitetura de *software*, baseada na perspectiva dos *middleware* de TV digital e IoT, que permite a integração e interoperação entre os mesmos. Foi descrito cada componente presente na arquitetura, bem como apresentada as interações entre esses componentes por meio de diagramas de sequência.

Essa arquitetura permite uma comunicação bidirecional entre as aplicações de TV e os *smart objects* por meio de dispositivos portáteis. Com isso, tanto dados de contexto do ambiente físico podem ser exportados para as aplicações de TV, quanto comandos para atuação de *smart objects* podem ser enviados pela a mesma.

A arquitetura proposta utiliza uma abordagem que especifica dispositivos portáteis presentes no ambiente físico como *master* ou *slaves*. Nessa abordagem diversos dispositivos portáteis *slaves* podem enviar dados de interações dos telespectadores para as aplicações de TV por meio do dispositivo portátil *master*. Como também, várias aplicações de TV podem utilizar os serviços de *smart objects* presentes no ambiente físico por meio do dispositivo portátil *master*.

Por fim, foram apresentadas abstrações de programação para aplicações de TV. As abstrações consistem na utilização dos termos e suas relações especificadas na ontologia adotada na arquitetura proposta, a IoTTV-Ont, para escrever as aplicações de TV. Com isso as aplicações de TV podem receber e enviar dados semânticos para o *middleware* de IoT.

6 Protótipo Baseado na Arquitetura Proposta

Neste capítulo é apresentado um protótipo que utiliza uma implementação concreta da arquitetura de *software* proposta. Especificidades quanto às ferramentas tecnológicas utilizadas também são apresentadas nesse texto. Algumas ferramentas tecnológicas foram desenvolvidas e também são retratadas no decorrer do capítulo.

6.1 Tecnologias Utilizadas

A Figura 6.1 mostra uma visão geral do protótipo desenvolvido em que é possível visualizar as tecnologias empregadas e as relações que as mesmas realizam entre si. Devido às especificidades do SDPEU e do CASMOC, os mesmos foram implementados no contexto deste trabalho de mestrado. Sendo o primeiro implementado[1] como módulo para aplicações Android e o segundo implementado[2] utilizando a linguagem Lua. A implementação do SDPEU utiliza os componentes nos quais se relaciona nesta infraestrutura de *software* e fornece uma API para as aplicações de dispositivos móveis poderem fazer uso de seus recursos.

Uma das características dessa arquitetura de *software* é o fato de permitir a utilização de diversos *middleware* de IoT. Para tanto, padrões de projeto, como *adapter*, foram utilizados para possibilitar a substituição desse *middleware* de maneira simples e com esforços reduzido, uma vez que esse padrão de projeto permite adaptar a interface fornecida pelo SDPEU para outras esperadas por diversos *middleware* de IoT [16]. Contudo, nessa materialização da arquitetura de *software*, foi utilizado o *middleware* de IoT M-Hub (apresentado no Capítulo 3) por atender aos requisitos para *middleware* de IoT exigidos na arquitetura.

Quanto ao *framework* para ontologias foi utilizado o Jena [7] para Android (Jena-Android[3]). Enquanto a base de conhecimento, como já mencionada, foi composta pelo esquema da ontologia escrito em OWL e suas instâncias, mais precisamente pela

[1] `www.github.com/makleystonlsdi/SDPEU`

[2] `www.github.com/makleystonlsdi/iDTVModules`

[3] `www.github.com/sbrunk/jena-android`

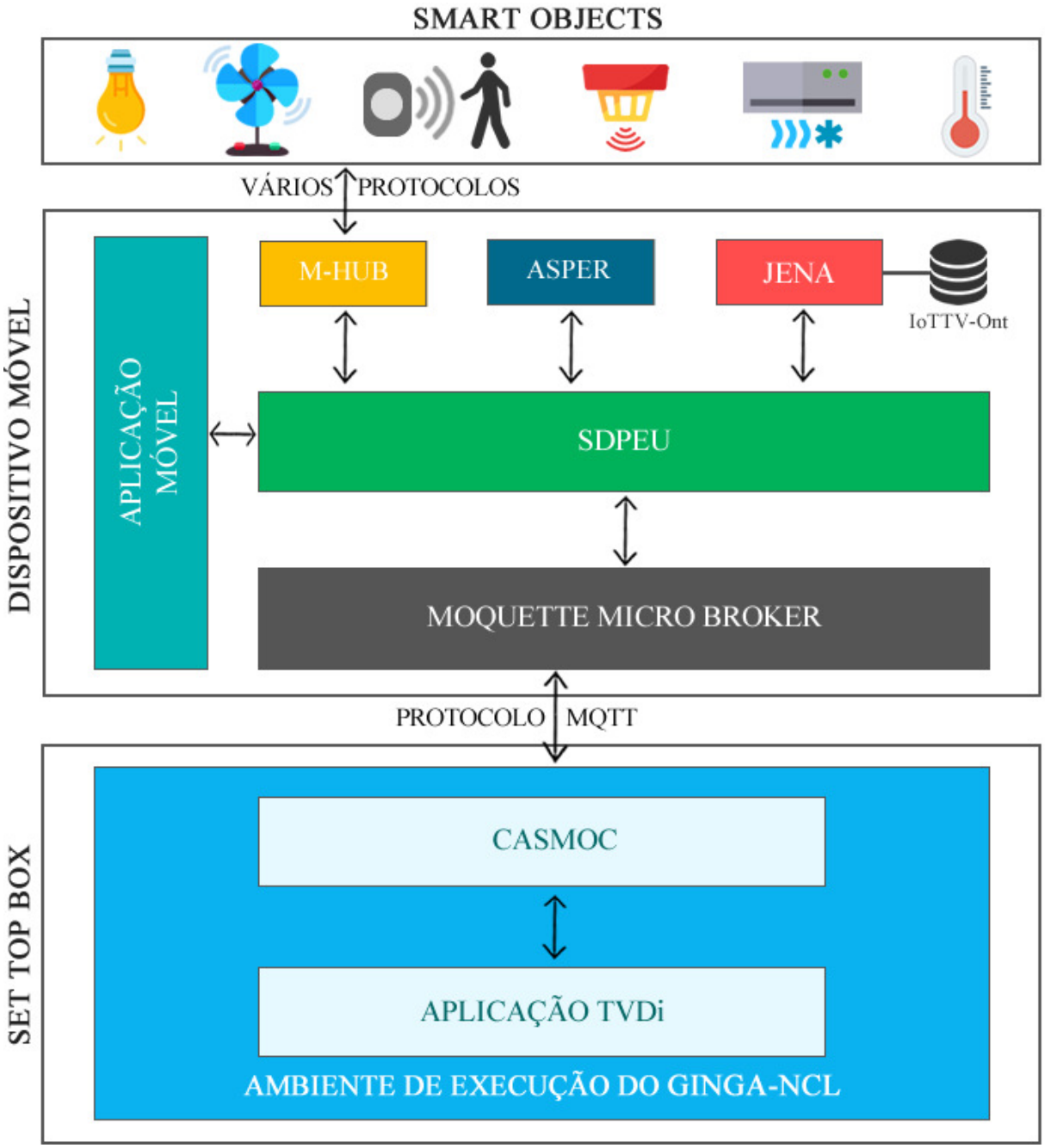


Figura 6.1: Visão geral do protótipo implementado

IoTTV-Ont povoada com instâncias que modelam uma residência contendo alguns *smart objects*, como sensores de temperatura e umidade, e alguns atuadores, como regulador da intensidade da iluminação e sistemas de ventilação.

O motor CEP utilizado na infraestrutura de *software* deste protótipo foi uma variação do Esper [12] desenvolvido para o Android, denominado Asper[4], que além de atender às especificações expostas na arquitetura, é um projeto de código aberto (*open source*). Esse motor CEP baseia-se em inferência e cláusulas de Evento-Condição-Ação

[4] `www.github.com/mobile-event-processing/Asper`

(*Event-Condition-Action* - ECA), ou seja, inicialmente, o motor CEP recebe os eventos de um fluxo, posteriormente são analisadas condições sobre esses eventos e, caso atenda alguma condição especificada, uma ação é acionada [55]. Essas condições são escritas em forma de regras CEP, que por sua vez são escritas sob uma linguagem declarativa *Event Processing Languages* (EPL) (EPL Esper) que estende todos os operadores da *Structured Query Language* (SQL) (*SQL-like extended*).

Já ao *micro broker* optou-se por uma versão destinada aos dispositivos móveis e que utiliza o protocolo de comunicação *Message Queuing Telemetry Transport* (MQTT)[5] [27], devido este protocolo ser simples e leve, além de ter sido projetado para dispositivos com recursos limitados e redes de baixa largura de banda, alta latência ou não confiáveis [25]. Desta forma, optou-se pela ferramenta denominada *Micro Broker Moquette*[6], em virtude de atender as necessidades exigidas pela arquitetura e de ser leve e livre de direitos autorais. Para o SDPEU publicar e subscrever no *micro broker Moquette* foi utilizado o cliente para *brokers* MQTT denominado MQTT Eclipse Paho Android Service 1.0.2[7] (versão para Android). Já para permitir o CASMOC publicar e subscrever no *micro broker* utilizou-se o Lua MQTT Client Library 0.2[8].

A implementação do CASMOC compreende todas as especificações desse componente exposto neste trabalho, incluindo as abstrações para comunicação com o *micro broker* e as abstrações de programação para o desenvolvimento de aplicações de TV expressos na Seção 5.3. Essas abstrações de programação são retratadas por meio de uma API que reflete os termos e relações presentes na ontologia IoTTV-Ont.

Por fim, foram desenvolvidos: um *software* para *smartphones* que executam o sistema operacional Android no qual foram importadas as tecnologias que compuseram essa materialização da arquitetura de *software*; aplicativos para TV digital brasileira, consequentemente utilizando o *middleware* Ginga-NCL; e os *smart objects* para sensoriar e atuar no ambiente. Para desenvolver os *smart objects* foi utilizada a plataforma de prototipagem eletrônica Arduíno[9] e componentes eletrônicos. Mais precisamente, foram desenvolvidos dois *smart objects*, denominados de *obj_of_action* e *obj_of_sensing*. O *obj_of_action* foi desenvolvido com dois serviços de

[5] `www.mqtt.org`

[6] `www.github.com/technocreatives/moquette`

[7] `www.eclipse.org/paho/`

[8] `www.github.com/geekscape/mqtt_lua`

[9] `www.arduino.cc`

atuação: regulador da iluminação e do sistema de ventilação do ambiente físico, e o *obj_of_sensing* foi desenvolvido para coletar dados de contexto do ambiente físico, tais como: dados de temperatura, umidade e presença. Em ambos os *smart objects* foram utilizados módulos de comunicação Bluetooth Classic.

A aplicação móvel presente no *smartphone* faz uso da API do SDPEU para, dentre outras atividades, registrar os padrões de movimentos de interação desejados pelos telespectadores, bem como registrar os dados relativos aos perfis dos mesmos. Nesse caso, a aplicação móvel permite aos telespectadores realizarem esses registros por meio de uma interface gráfica – *Graphical User Interface* (GUI). Também foi possível projetar na tela do *smartphone* dados sobre o conteúdo transmitido, como o guia de programação eletrônico – EPG. Por outro lado, foram desenvolvidas várias aplicações de TV digital nas quais importaram o CASMOC para fazer uso de sua API e, com isso, enviar comandos de atuação para os *smart objects* e receber dados de contexto do ambiente e dos telespectadores presentes.

6.2 Estrutura de Tópicos

A comunicação entre o SDPEU e o CASMOC ocorre por meio do *micro broker*, no qual são publicados em determinados tópicos dados referentes aos *smart objects* e aos telespectadores presentes no ambiente. Dessa forma, para permitir a troca de dados entre esses componentes, uma estrutura de tópicos capaz de expressar dados relativos aos serviços e estados dos *smart objects*, como também capaz de expressar dados relativos aos dispositivos portáteis, como dados de perfil do seu proprietário e as interações multimodais geradas pelo mesmo foi desenvolvida. Essas estruturas de tópicos refletem o modelo conceitual IoTTV-Ont, utilizando os termos expressos em seus conceitos para compor o tópico no qual os dados devem ser publicados.

Um tópico, no protocolo MQTT, é a "chave que identifica o canal de informações para o qual os dados de carga útil são publicados" [26]. Essa chave é identificada por uma cadeia de caracteres UTF-8 que por meio dela permite a filtragem de mensagens trafegadas no *micro broker*. Um tópico pode conter diversos níveis nos quais utilizam-se o caractere de barra inclinada para direita ("/") para especificá-los. Por exemplo:

`/primeiro_nivel/segundo_nivel/.../n_nivel`.

Desse modo, é possível especificar com maior detalhamento um tópico no qual deve ser publicada uma mensagem, como também permite que os subscritores possam realizar filtros mais específicos sobre as mensagens publicadas no *micro broker*.

Para um subscritor receber os dados do sensor de temperatura do exemplo citado é preciso expressar no *micro broker* o interesse a esse tópico. No entanto, o protocolo MQTT permite o uso de caracteres "curingas" para auxiliar nas filtragens de mensagens trafegadas no *micro broker*. Esses caracteres podem substituir níveis na estrutura de tópicos permitindo que os subscritores possam receber quaisquer mensagens cujo nível que possui algum desses caracteres curingas não seja restringido a nenhuma cadeia de caracteres. Os curingas permitidos no protocolo MQTT são: curinga de nível único ("+") e curinga multinível ("#"). O primeiro permite substituir um único nível dentro da estrutura de tópicos possibilitando o recebimento das mensagens trafegadas no *micro broker* independentemente da cadeia de caracteres presentes no nível em que é utilizado o "+". Já o segundo permite que o subscritor receba todas as mensagens trafegadas no *micro broker* independente dos níveis posteriores ao curinga "#". A geração desses tópicos são realizados em tempo de execução, ou seja, não é necessário predefini-los, visto que podem ser especificados ao passo em que se necessite.

Diante dessas características do MQTT, foram desenvolvidas estruturas de tópicos que permitem a troca de dados entre o CASMOC e SDPEU. Essas estruturas de tópicos permitem a aplicação de TV filtrar dados de *smart objects* publicados pelo SDPEU e filtrar dados dos dispositivos portáteis presentes no ambiente físico, além de permitir o SDPEU filtrar dados publicados pela aplicação de TV para atuação no ambiente físico. As mensagens transmitidas em qualquer uma dessas estruturas de tópicos é uma *string* formatada no tipo *JavaScript Object Notation* (JSON), e tanto o CASMOC quanto o SDPEU conhecem as estruturas de mensagens formatadas.

6.2.1 Estrutura de Tópicos dos Dados Coletados Através de *Smart Objects*

O modelo de tópico desenvolvido para o envio de mensagens referentes aos *smart objects* para as aplicações de TV expressa: o ambiente físico (*Environment*) em que se encontra o *smart object* (*e. g. Bedroom* e *DiningRoom*), a categoria em que se enquadra o *smart objects* (*Controllable*), que pode ser *Actuator* ou *Sensor*, e o tipo (*Type*) do *smart object* (*e. g. TemperatureSensor*). O modelo dessa estrutura de tópicos pode ser representado da seguinte forma:

`/Environment/Controllable/Type`.

Desse modo, um exemplo de tópico capaz de filtrar mensagens trafegadas no *micro broker* em busca de dados de um sensor de presença presentes em qualquer cômodo de uma casa, seria:

`/+/Sensor/PresenceSensor`.

Nessa estrutura de tópicos, a mensagem enviada para o CASMOC descreve os *smart objects* já anotado semanticamente, contendo: as funcionalidades e as notificações que o mesmo possui, seu estado, a descrição do ambiente físico no qual esse *smart object* se encontra, o identificador único desse *smart object*, seu tipo e a sua categoria, se é um atuador ou sensor.

6.2.2 Estrutura de Tópicos Relativos à Interação Multimodal e Perfil de Usuários

Para possibilitar o envio de dados sobre os dispositivos portáteis presentes no ambiente físico para as aplicações de TV foi utilizada uma estrutura de tópicos contendo apenas dois níveis. O primeiro nível contém a especificação do ambiente físico (*Environment*) em que se encontra o dispositivo portátil (*e. g. Bedroom*). Já o segundo nível contém a informação referente ao tipo do dispositivo portátil (*TypePortable*) que pode ser um *smartphone* ou *tablet*, por exemplo. Desta forma, essa estrutura de tópicos pode ser entendida por este modelo:

`/Environment/TypePortable`.

Em um cenário em que se deseja filtrar apenas os dados provindos dos *smartphones* presentes no quarto da casa, deve-se utilizar:

`/Bedroom/Smartphone.`

A *string* enviada na mensagem possui dados anotados semanticamente. A mensagem é compreendida por: o identificador único do dispositivo portátil, a informação sobre o ambiente físico no qual esse dispositivo portátil se encontra, dados do perfil de seu proprietário e eventos gerados por alguma interação multimodal realizada pelo telespectador, quando houver.

6.2.3 Estrutura de Tópicos Relativos à Atuação no Espaço Físico

O envio de dados das aplicações de TV para o SDPEU relativos à atuação dos *smart objects* presentes no espaço físico ocorre por meio de publicações em um tópico constituído por apenas um nível. A *string* que representa esse nível é denominado *Actuator* (expresso na IoTTV-Ont). Desta forma, um exemplo dessa estrutura de tópicos é:

`/Actuator`

A mensagem publicada expressa todo um *smart object* no estado desejável pela aplicação de TV. Em outras palavras, essa mensagem reporta qual o estado do *smart object* que o mesmo deve ficar após sua atuação. Por exemplo, se um regulador de intensidade presente no ambiente físico estiver iluminando apenas 25% de sua capacidade, uma possível mensagem pode expressar que o mesmo *smart object* ilumine 40% de sua capacidade. Um outro exemplo é a publicação de uma mensagem, pela aplicação de TV, para um *smart object* que gerencia o sistema de ventilação do ambiente físico. Nesse caso, a mensagem pode conter *off* ou *on* como valores para seu estado final após a atuação. Para compor o restante da mensagem, as propriedades especificadas pela IoTTV-Ont que descreve um *smart object* também devem ser inseridas, tais como: seu identificador, a funcionalidade de atuação que deseja utilizar e o tipo do *smart object* (*e. g. DimmerLamp* e *Ventilator*).

6.3 Características de Implementação do SDPEU

A Figura 6.2 exibe o diagrama de classes da *Unified Modeling Language* (UML) correspondente à implementação do SDPEU contendo as principais classes e métodos. O Diagrama de Classes está organizado em pacotes, os quais concentram as classes responsáveis por gerenciar cada componente da arquitetura de *software* proposta. Uma aplicação móvel para fazer uso desse protótipo deve importar o SDPEU. Dessa forma, a aplicação móvel poderá utilizar os métodos construídos nessa classe e poderá utilizar seus recursos.

Uma aplicação cliente que importa o componente SDPEU visualiza apenas os métodos públicos presentes na classe SDPEU. Essa classe implementa os métodos presentes em sua interface, os quais permitem utilizar recursos implementados em outros pacotes, tais como: *setIoTMiddleware*, que permite a inserção de uma classe adaptadora para o *middleware* de IoT especificado pelo desenvolvedor; *setFrameworkForOntology*, que possibilita a inserção de a classe adaptadora da tecnologia que gerencia uma base de conhecimento; e *setIPMaster*, que permite inserir o endereço de IP do dispositivo portátil *master*.

Como mencionado, o SDPEU permite a substituição de alguns componentes, como o *middleware* de IoT e o *framework* para ontologias, de maneira simples e com esforço reduzido. Para isso, o padrão *adapter* foi empregado para esta finalidade. Assim, o pacote *IoTMiddlewarePackage* possui as entidades: *IoTMiddlewareAdapter*, que é uma classe abstrata na qual o desenvolvedor deve estender e implementar os métodos expressos nela para adaptar a interface do SDPEU à interface do *middleware* de IoT adotado por ele; *IoTMiddlewareTechnology*, que é a classe que recebe a implementação de uma *IoTMiddlewareAdapter* e utiliza os recursos do *middleware* especificado pelo desenvolvedor; *IoTMiddlewareListenerImpl*, que implementa *IoTMiddlewareListener*, a qual é utilizada para monitorar os dados recebidos pelo *middleware* de IoT e fornecê-los à classe SDPEU.

Por outro lado, o pacote *OntologyPackage* possui a classe abstrata *FrameworkForOntologyAdapter* que permite ao desenvolvedor substituir a tecnologia utilizada para consultas sobre a base de conhecimento. Essa classe deve ser estendida e implementada pelo desenvolvedor, adaptando a interface oferecida pelo SDPEU à interface da tecnologia adotada. A classe *FrameworkForOntologyTechnology*

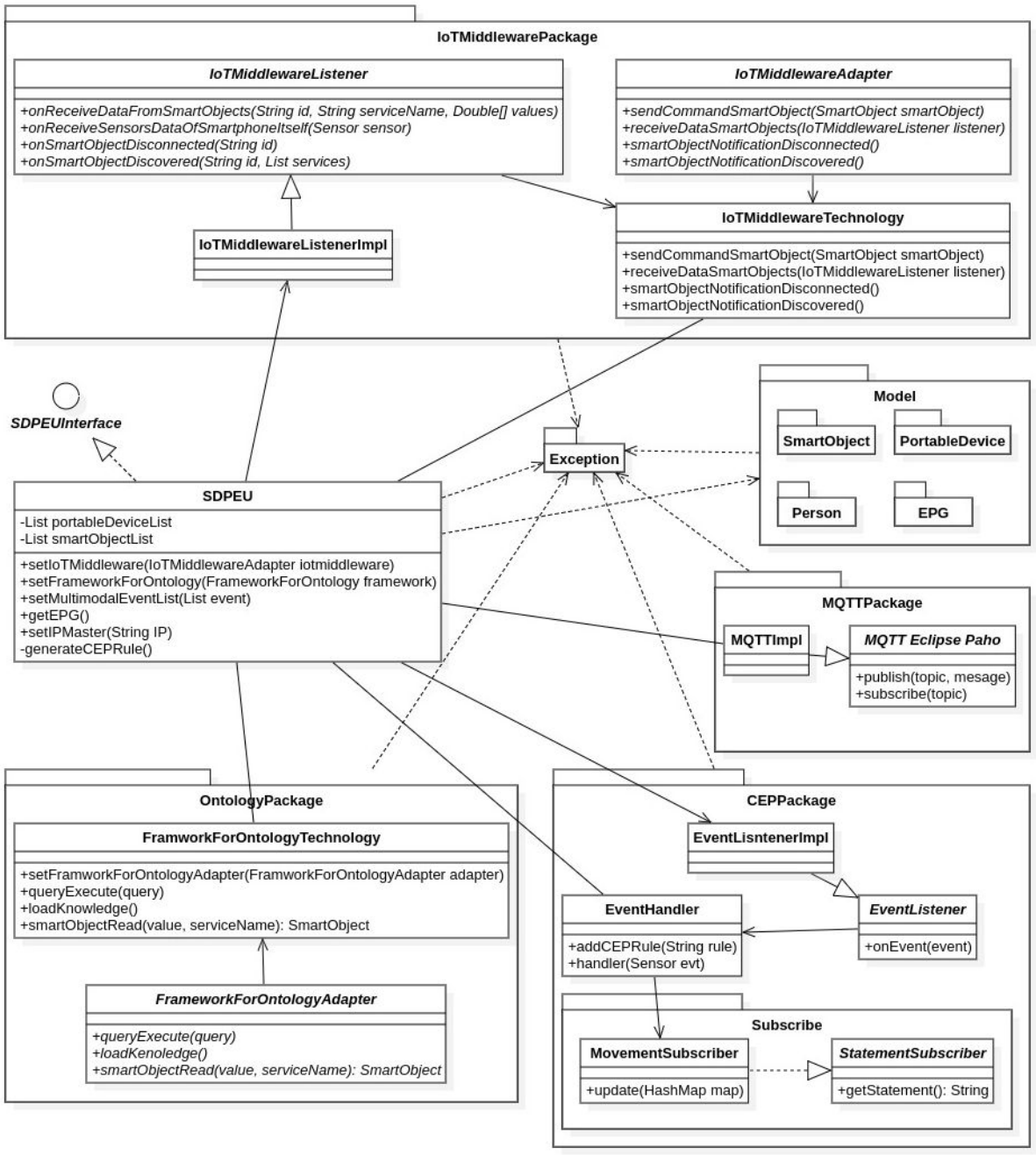


Figura 6.2: Diagrama de Classes relacionado ao SDPEU

possui métodos que recebem a implementação do *FrameworkForOntologyAdapter* pelo desenvolvedor e utiliza os recursos da tecnologia especificada por ele.

O pacote *Model* possui classes organizadas em subdiretórios nas quais especificam objetos que são instanciados pelas demais classes presentes no SDPEU. As classes contidas nesse pacote descrevem os *smart objects*, os dispositivos portáteis, as pessoas e o EPG. Já o pacote *MQTTPackage* contém uma classe que implementa os métodos pelos quais são publicadas as mensagens no *micro broker*, como também são realizadas as subscrições no mesmo.

As entidades contidas no pacote *CEPPackage* permitem utilizar os recursos do motor CEP para identificar padrões de interações realizadas pelos telespectadores por meio de algum sensor interno ao dispositivo portátil. A classe *EventHandler* permite, dentre outras tarefas, adicionar regras CEP para serem utilizadas em análises sobre esses fluxos de dados dos sensores que o mesmo recebe por meio do método *handler*. Essas regras são inseridas por meio do SDPEU, pois sempre que é inserida em *setMultimodalEventList* uma lista de eventos que definem as interações multimodais do telespectador, o método *generateCEPRule* é acionado para gerar as regras referentes aos eventos listados. Esse método executa o método *addCEPRule* presente em *EventHandler* para registrar tais regras no motor CEP.

O subdiretório *Subscribe* possui as classes que implementam as regras CEP de acordo com os meios de interação do telespectador com a aplicação de TV. Por exemplo, o *MovementSubscriber* é a classe que define uma regra cuja interação foi realizada pelo movimento do dispositivo portátil. Alguns dos sensores possíveis para essa interação do telespectador são os sensores de acelerômetro e giroscópio.

Uma exemplo de uma regra CEP utilizada para identificar um padrão de interação do telespectador que usa os sensores de acelerômetro do dispositivo portátil pode ser escrito da seguinte maneira:

```
select * from AcceleromenterSensor(
x between [X1Interval] and [X2Interval]
, y between [Y1Interval] and [Y2Interval]
, z between [Z1Interval] and [Z2Interval])
```

Nesse exemplo, os intervalos presentes em x, y e z representam uma margem de tolerância para detectar um movimento com o *smartphone*. Dessa forma, quanto menor o intervalo, maior deve ser a precisão na execução do movimento para que os valores dos três eixos do sensor de acelerômetro se alinhem dentro desse intervalo especificado na regra.

Por fim, o pacote *Exception* possui as implementações das exceções que são tratadas pelo SDPEU. Essas exceções são dependências de diversas classes do componente.

6.4 Características de Implementação do CASMOC

A Figura 6.3 apresenta o Diagrama de Classes da UML correspondente ao componente CASMOC desenvolvido sob a linguagem Lua. Nesse diagrama é apresentada a classe CASMOC e seus principais métodos, além das bibliotecas que a mesma utiliza. Para uma aplicação de TV utilizar os recursos do CASMOC a mesma deve importá-lo em sua implementação.

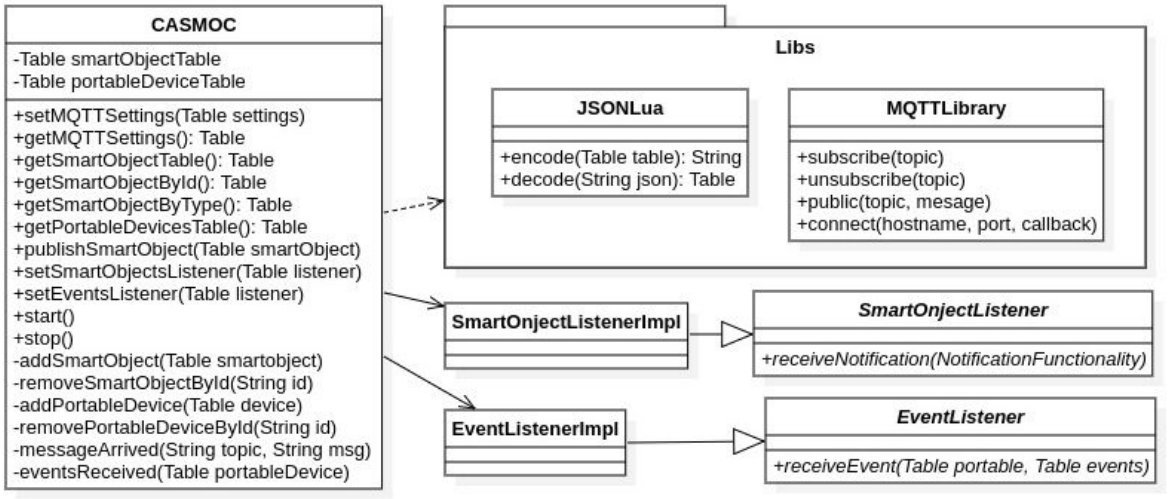


Figura 6.3: Diagrama de Classes relacionado ao CASMOC

A classe CASMOC possui como principais atributos os *smartObjectTable* e *portableDeviceTable* que armazena todos os *smart objects* e dispositivos portáteis descobertos no ambiente físico. Esses atributos também são atualizado sempre que novos valores são lidos pelo *middleware* de IoT, bem como são removidos quando perdem conexão com o mesmo. Para armazenar e remover esses *smart objects*, os métodos privados *addSmartObject* e *removeSmartObject* são utilizados. Já para adicionar e remover o dispositivos portáteis utiliza-se os métodos privados *addPortableDevice* e *removePortableDevice*.

Por outro lado, os métodos visíveis às aplicações de TV permitem as mesmas consultarem a lista de *smart objects* e dispositivos portáteis presentes no ambiente físico (*getSmartObjectTable* e *getPortableDeviceTable*, respectivamente). Ainda, essa classe oferece métodos para as aplicações de TV publicarem no *micro broker* comandos para atuadores (*publishSmartObject*), além de oferecer meios para receber notificações dos *smart objects* nas quais possuem interesse. Nesse último caso, o padrão de projeto *observer* foi implementado para a aplicação de TV poder ser

notificada instantaneamente após o recebimento das notificações geradas pelos *smart objects*. Utilizando o mesmo padrão de projeto, foi implementado o mecanismo para notificar as aplicações de TV sobre eventos de interações geradas pelos telespectadores por meio do seus dispositivos portáteis. Nesse caso, as aplicações de TV devem implementar as classes abstratas *SmartObjectListener* e *EventListener* e atribui-las aos métodos *setSmartObjectListener* e *setPortableListener* do CASMOC, respectivamente, para serem notificadas imediatamente quando novos valores forem recebidos do *micro broker*.

A classe *MQTTLibrary* é utilizada para realizar a comunicação do CASMOC com o *micro broker*. Seus métodos permitem se conectar ao *micro broker*, além de publicar e subscrever em determinados tópicos no mesmo. Já a classe *JSONLua* transforma as tabelas Lua em *Strings* formatadas em JSON e vice-versa. Ambas as classes são dependências do CASMOC.

6.5 Conclusão

Este capítulo exibiu o desenvolvimento de um protótipo que faz uso de uma implementação concreta da arquitetura de *software* proposta. Neste texto foi apresentada as tecnologias que compreenderam a infraestrutura de *software* utilizada no protótipo.

Os componentes SDPEU e CASMOC também foram desenvolvidos neste trabalho, no qual o primeiro foi implementado para aplicações móveis executados em sistemas operacionais Android, e o segundo desenvolvido em Lua, executado no ambiente de execução de aplicações do *middleware* Ginga-NCL. Ambos estão disponíveis para *download* no site do projeto IoTTV[10] do Laboratório de Sistemas Distribuídos e Inteligentes (LSDi) da Universidade Federal do Maranhão - MA.

Para viabilizar a troca de dados entre o SDPEU e o CASMOC pelo *micro broker*, foram desenvolvidas estruturas de tópicos nas quais permitiram: enviar dados dos *smart objects* para as aplicações de TV; enviar dados dos dispositivos portáteis para as aplicações de TV; e enviar dados referentes aos comandos de atuação dos *smart objects* das aplicações de TV para o SDPEU.

[10] `www.lsdi.ufma.br/~iottv`

7 Avaliação

Este capítulo apresenta a avaliação realizada para validar a proposta deste trabalho. Demonstrações de casos de uso foram adotadas para realizar tal avaliação. Desse modo, diversos cenários são apresentados, os quais exploram os recursos oferecidos por uma implementação concreta da arquitetura proposta. Por fim, uma discussão é apresentada ao término do capítulo.

7.1 Casos de Uso

O produto deste trabalho é uma arquitetura de *software* capaz de permitir que aplicações de TV possam trocar dados com um *middleware* de IoT. Deste modo, as aplicações de TV podem enviar comandos de atuação para os *smart objects* e receber dados de sensores do ambiente e dos dispositivos portáteis dos telespectadores presentes no espaço físico de apresentação. Não somente, foi produzido neste trabalho o modelo conceitual IoTTV-Ont, capaz de representar uma residência com os *smart objects* e as pessoas presentes nela, incluindo o espaço físico de percepção audiovisual no qual a TV se encontra. Diante disso, optou-se por avaliar essa arquitetura de *software* por meio de casos de uso, utilizando o protótipo apresentado no Capítulo 6, uma vez que o mesmo utiliza uma implementação concreta da arquitetura proposta.

Este protótipo foi exposto a diversos cenários, nos quais foram filmados e disponibilizados no site do projeto IoTTV[1]. Em cada cenário desenvolvido foi implementada uma aplicação de TV que faz uso desse protótipo. E em todas elas utilizou-se os *middleware* M-Hub, como *middleware* de IoT, e Ginga-NCL, como *middleware* de TV digital. O *middleware* Ginga-NCL foi executado em uma máquina virtual Linux para a ferramenta de execução de máquinas virtuais VMWare[2]. As configurações de parâmetros de *hardware* da máquina virtual do Ginga-NCL foram mantidas conforme sua imagem[3] para *download*. A VMWare foi executada em um

[1] `www.lsdi.ufma.br/~iottv`

[2] `www.vmware.com/products/workstation-player.html`

[3] `www.gingancl.org.br/pt-br/ferramentas`

laptop Intel i7 7ª geração com 16 GB de memória RAM e possuindo 8 núcleos de processamento de frequência 2,7 GHz e com sistema operacional Linux 16.04.

Em todos os casos de uso utilizou-se um *smartphone* de modelo Mi5X da fabricante Xiaomi para intermediar a comunicação entre os *smart objects* e a aplicação de TV (dispositivo portátil *master*). Esse *smartphone* possui as seguintes especificações: processador Qualcomm Snapdragon 625 de 2.0 GHz Octa Core e memória RAM de 4 GB, além de possuir o sistema operacional Android 7.2. Outros *smartphones* (dispositivos portáteis *slaves*), também de sistema operacional Android, foram utilizados para demonstrar as interações multimodais realizadas por diversos telespectadores com as aplicações de TV. São eles: Moto G2 da fabricante Motorola com 1 GB de memória RAM e processador Qualcomm Snapdragon 400 Quad-Core de 1.2 GHz; ZenFone 2 da fabricante ASUS que possui 4GB de memória RAM e processador Z3580 Intel Atom de 2.3 GHz Quad Core.

7.1.1 Caso de Uso 1: Clipe Musical

Este caso de uso teve como objetivo exportar para a aplicação de TV os dados de contexto do ambiente físico no qual a TV se encontra, como: temperatura e umidade do ambiente e a presença de movimentos no mesmo. Não somente, este caso de uso teve a finalidade de demonstrar o suporte à diversos telespectadores presentes no espaço físico de apresentação e distinguir as interações multimodais realizadas pelos mesmos com a aplicação de TV, possibilitando a essas interações refletirem no conteúdo apresentado. Adicionalmente, foi implementado na aplicação de TV a possibilidade da mesma alterar aspectos do ambiente conforme a narrativa do conteúdo apresentado.

O caso de uso mostra a reprodução de um conteúdo multimídia no qual sua narrativa consiste na apresentação de um clipe musical em que o vocalista de uma banda musical caminha no litoral de uma praia durante o amanhecer. Durante a reprodução da mídia, a aplicação de TV recebe os dados de sensores descobertos no ambiente e apresenta essas informações na tela da TV, informando quais tipos de sensores coletaram os dados e quais valores foram lidos. A apresentação na tela desses dados dos sensores são alterados sempre que novos valores são lidos. Além disso, foi implementada na aplicação de TV a tolerância de dez segundos sem a detecção de

movimentos no ambiente físico pelo sensor de presença. Passados esse período surge na tela da TV a notificação de ausência de movimentos na sala.

Ainda durante a execução da mídia, telespectadores utilizaram seus respectivos *smartphones* para interagir com a aplicação de TV. Mais precisamente, eles realizaram movimentos com seus *smartphones*, fazendo uso dos sensores internos ao dispositivo e, a partir disso, os eventos gerados pelos movimentos foram publicados no *micro broker* do dispositivo portátil *master*. Consequentemente, a aplicação de TV pôde receber esses eventos de interações e em seguida refleti-los sobre o conteúdo em reprodução, apresentando na tela da TV o nome do telespectador que realizou a interação e qual comando foi gerado. No entanto, para demonstrar outras aplicabilidades com o uso desses comandos gerados pelos telespectadores, optou-se por definir regras que interrompessem ou retomassem a reprodução do conteúdo em apresentação. Sendo os movimentos laterais e sequenciais para direita e esquerda, nessa ordem, para interromper a reprodução do vídeo, e os movimentos direcionais para cima e para baixo, também nessa ordem, para retomar a reprodução da mídia interrompida. Por fim, no decorrer da música, a intensidade da iluminação no ambiente em que se encontra o cantor da banda é alterada conforme o nascer do sol. Então, foi implementado para a aplicação de TV enviar comandos para intensificar, de maneira gradativa, a iluminação no espaço físico de apresentação.

As Figuras 7.1a, 7.1b e 7.1c apresentam instantes durante o desenvolvimento desse caso de uso. Sendo que a Figura 7.1a mostra o instante em que os dados de contexto do ambiente físico foram exportados para aplicação de TV, nos quais foram apresentados na tela da TV durante a apresentação do clipe musical. Já a Figura 7.1b exibe o momento em que um telespectador acabara de realizar os movimentos laterais e sequenciais para direita e esquerda com seu *smartphone*, interrompendo assim a apresentação do conteúdo apresentado. Por fim, a Figura 7.1c apresenta o instante em que a iluminação do ambiente físico está em 40% do máximo possível pelo *smart object* que o gerencia.

Esse caso de uso permitiu explorar o objetivo presente no escopo deste trabalho que tange a possibilidade da aplicação de TV estar ciente do contexto no qual a TV se encontra. Nesse caso a ciência do contexto ocorreu por meio dos dados coletados pelos sensores dispostos no espaço físico de apresentação, bem como na identificação dos telespectadores presentes.

(a) (b) (c)

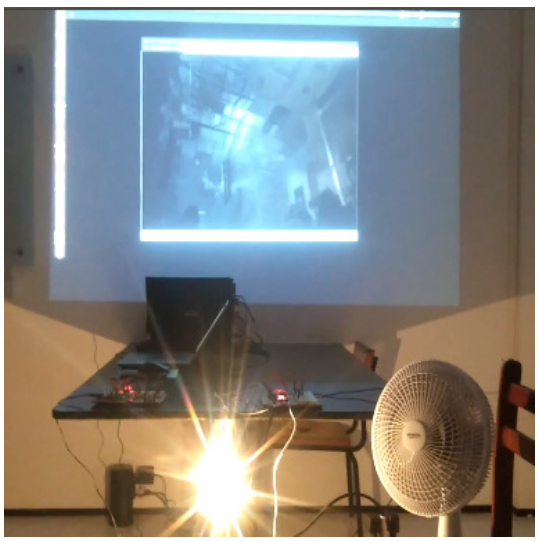
Figura 7.1: Capturas de imagens durante a execução do caso de uso 1

A utilização da abordagem em que utiliza um dispositivo portátil *master* e os demais como dispositivos portáteis *slaves* permitiu que diversos telespectadores pudessem ser detectados e permitiu a interação dos mesmos com a aplicação de TV, uma vez que tanto a aplicação de TV quanto os dispositivos portáteis *slaves* precisaram conhecer apenas o endereço de IP do dispositivo portátil *master* pelo qual houve o envio desses dados dos telespectadores para a aplicação de TV. Não somente, a aplicação de TV desenvolvida para esse caso de uso utilizou, além dos recursos do protótipo que permitem o recebimento dos dados de contexto do ambiente físico e os dados sobre os telespectadores, o recurso que permite a mesma alterar aspectos do ambiente físico, visto que foram enviados comandos para intensificar a iluminação do espaço físico de apresentação.

7.1.2 Caso de Uso 2: *Trailer* de Filme

O objetivo deste caso de uso foi apresentar a alternância dos aspectos do ambiente físico de apresentação por meio do gerenciamento da intensidade da iluminação e do sistema de ventilação do ambiente. Neste caso de uso os aspectos do ambiente físico são alterados mediante a narrativa do conteúdo em reprodução pela aplicação de TV. Mais precisamente, o *smart object* capaz de atuar no ambiente físico é acionado diversas vezes pela aplicação de TV para reproduzir efeitos que busquem aumentar o grau de imersão do telespectador diante do conteúdo em reprodução.

A mídia em reprodução apresenta cenas de um filme em que algumas dessas cenas possuem uma alternância da iluminação no ambiente em que os personagens se encontram, como também os mesmos em vezes se encontram em situações em que são impactados por rajadas de ventos. As Figuras 7.2a e 7.2b apresentam instantes em que se desenvolvia este caso de uso. As capturas dessas imagens ilustram momentos em que o sistema de ventilação e iluminação foram utilizados devido aos envios de comandos para acioná-los por parte da aplicação de TV ao *smart object* responsável. Assim, a Figura 7.2a mostra o instante em que fora ajustada a iluminação do ambiente físico em 75% da intensidade máxima do regulador de iluminação e o sistema de ventilação fora acionado. Enquanto que a Figura 7.2b apresenta o instante em que a intensidade da iluminação do ambiente físico fora ajustado para 25% e desligado o sistema de ventilação presente no espaço de apresentação.



(a)

(b)

Figura 7.2: Alteração dos aspectos físicos do ambiente a partir da aplicação de TV

Com este caso de uso é possível visualizar que o protótipo, que faz uso de uma implementação concreta da arquitetura proposta, permite alterar aspectos do ambiente físico no qual a TV se encontra a partir de aplicações de TV. Desta forma, o objetivo deste trabalho que tange à necessidade da aplicação de TV alterar aspectos físico do ambiente de apresentação é contemplado. Neste caso de uso, o protótipo teve que realizar a descoberta dos *smart objects* e dos seus serviços ofertados no ambiente físico e estabelecer uma comunicação com os mesmos, utilizando a tecnologia Bluetooth, para então poder utilizá-los quando requisitados pela aplicação de TV.

7.1.3 Caso de Uso 3: O Jogo *Hungry Monkey*

Este caso de uso teve como objetivo apresentar a utilização do protótipo para permitir que o telespectador interagisse com a aplicação de TV por meio de diversos modos. Para tanto, utilizou como aplicação de TV um jogo virtual no qual consiste na apresentação de um personagem (em forma de um animal macaco) e uma banana. O objetivo desse jogo é mover o personagem até a fruta e ao encontrá-los o jogo finaliza. A princípio, se o jogo fosse desenvolvido para métodos tradicionais de interação do usuário com a aplicação de TV, ou seja, por meio do controle remoto, a navegabilidade se limitaria às teclas de direção presentes no controle. No entanto, aplicando nosso protótipo, os meios para mover o personagem puderam ser ampliados a inúmeras outras formas, como por exemplo realizando a movimentação do *smartphone* para os lados esquerdo e direito e para cima e para baixo, fazendo uso dos sensores de acelerômetro. O caso de uso realizado neste cenário consiste exatamente como o exemplo citado, possibilitando ao telespectador movimentar o personagem por meio desses movimentos direcionais com o *smartphone*. A Figura 7.3 apresenta uma imagem capturada durante a execução desse caso de uso. No momento da captura da imagem o telespectador acabara de realizar o movimento de declinar o *smatphone*, e com isso o personagem do jogo se deslocara para baixo em relação a sua disposição anterior na tela da TV.

Com essa demonstração de uso é possível perceber que o telespectador pôde interagir com a aplicação em execução na TV por meio de diversas formas para gerar o comando de movimentação do personagem, atendendo assim um dos

Figura 7.3: Captura de imagem durante a execução do caso de uso 3

objetivos dispostos neste trabalho: possibilidade de interação multimodal por parte do telespectador com a aplicação de TV.

Mesmo diante da vasta possibilidade de utilização do nosso protótipo, foi decido realizar uma comparação entre a aplicação de TV deste caso de uso com uma aplicação de TV similar, mas que utiliza os meios convencionais para interagir com uma aplicação de TV, ou seja, por meio do controle remoto. O objetivo é visualizar a alteração do esforço por parte dos desenvolvedores quando estes utilizam nosso protótipo. Para tanto, foi realizado uma comparação quanto ao número de linhas implementadas – *Non-Comment Lines of Code* (NCLOC) – em cada aplicação. Adicionalmente, observamos a complexidade ciclomática – *Cyclomatic Complexity* (MVG) – e sua média (MVG AVG) em relação à quantidade de métodos utilizados em toda a aplicação de TV. Por fim comparamos o tamanho resultante das aplicações de TV, em Kbytes, visto que um aplicativo em produção consumirá uma fração da largura de banda em uma transmissão de TV.

A coleta desses dados foi realizada por meio do *plugin* MetricsReloaded[4] do Ambiente de Desenvolvimento Integrado (IDE) IntelliJ[5] e não foram consideradas as extensões do próprio *middleware* de TV (Ginga-NCL) sobre a linguagem Lua, visto que essas extensões são uma constante para qualquer aplicação de TV. Esse *plugin* define o NCLOC como sendo todas linhas de código ignorando as linhas em branco e as linhas comentadas. Já a métrica de complexidade ciclomática (MVG) é uma medida do número máximo de caminhos lineares independentes de um programa [17].

Todos os pontos comparados entre as duas aplicações são apresentados na Tabela 7.1. Nessa tabela é possível observar que foi necessário implementar um número relativamente pequeno de linhas no documento principal da aplicação para a mesma utilizar nosso protótipo, visto que foram adicionadas apenas 13 linhas de codificação. Essas inserções compreenderam a importação do CASMOC e a implementação do *listener* para o recebimento das novas interações dos telespectadores. Na aplicação de TV que faz uso do protótipo percebe-se o incremento de 2222 linhas, além das presentes no documento principal da aplicação, que são oriundas da importação realizada.

Tabela 7.1: Tabela comparativa das aplicações de TV com e sem o uso do nosso protótipo

Aplicações de TV	NCLOC Documento Principal	NCLOC Projeto	MVG (AVG)	MVG Total	Tamanho
Sem a infraestrutura de *software*	59	59	5,00	15	13,6 KB
Com a infraestrutura de *software*	72	2294	4,38	445	123,0 KB

Já quanto ao MVG é possível perceber um incremento de 430 ramificações na aplicação que faz uso do nosso protótipo quando comparado com a aplicação de TV que não o utiliza, que também foram oriundas da importação realizada. No entanto, a média da complexidade ciclomática em relação à quantidade de métodos presentes

[4] `www.plugins.jetbrains.com/plugin/93-metricsreloaded`

[5] `www.jetbrains.com/idea/`

em toda a aplicação de TV (incluindo as presentes no CASMOC) totalizou 4,38. Este resultado é levemente melhor que a aplicação de TV que não utiliza o nosso protótipo.

Devido à importação do CASMOC, a aplicação de TV sofreu um aumento de tamanho em seu projeto final de 109,4 KB. Contudo, provedores de serviços de TV digital possuem uma alta taxa de transmissão, como, por exemplo, nas alcançadas por rádio difusão de sistemas de televisão terrestre em que a taxa é de 19,3 Mbps [46], o que não impactaria significadamente no consumo de banda para o envio dessa aplicação às casas dos telespectadores.

Portanto, mesmo a aplicação após estendida para suportar nossa infraestrutura de *software* sofrer alterações quanto ao número de linhas e tamanho, a mesma continua relativamente leve para os meios de transmissões disponíveis para as TV digitais. Além disso, as aplicações de TV iriam passar a ter uma vasta possibilidade de recursos, como ter ciência do contexto no qual a TV está inserida, por exemplo.

7.1.4 Discussão

Por meio dos casos de uso descritos, foi explorado como este protótipo, que utiliza uma implementação concreta da arquitetura de *software* proposta para a integração dos *middleware* de TV digital e de IoT, permite atingir os requisitos funcionais e não funcionais especificados no Capítulo 5. Diante disso, é possível observar que o caso de uso 3 permitiu explorar novos meios de interação do telespectador com a aplicação de TV. Foi utilizado, nesse caso de uso, movimentações realizadas no próprio *smartphone* que culminaram na geração de comandos baseados na interpretação dos dados do sensor de acelerômetro desse dispositivo. Um outro caso de uso que também permite explorar esses dados dos sensores embutidos ao dispositivo portátil para a geração de comandos à aplicação de TV é o caso de uso 1. Nesse caso de uso, apresentou-se na tela da TV os comandos gerados pelos telespectadores e informações sobre quem os gerou, além de permitir interromper e reiniciar a apresentação do conteúdo em reprodução mediante esses comandos. Em ambos os casos de uso, houve a leitura dos dados dos sensores internos ao dispositivo portátil para gerar esses comandos de interação com a aplicação de TV, assim sendo, o requisito funcional que tange a necessidade de adquirir os dados dos sensores interno ao dispositivo portátil (RF5) foi atendido.

Os casos de uso 1 e 2 permitiram explorar a descoberta dinâmica de *smart objects* e os serviços por eles disponibilizados (RF2), visto que ambos cenários necessitaram identificar no ambiente físico quais *smart objects* possuíam capacidade de atuar no ambiente, para assim intervir no espaço físico de apresentação alterando os seus aspectos, como regulando a intensidade da iluminação e gerando rajadas de vento (RF3). O caso de uso 1 utiliza apenas o serviço de regular a intensidade de iluminação do ambiente presente no *smart object obj_of_action*, enquanto na demonstração de uso 2 é exibida a utilização dos dois serviços oferecidos pelo mesmo *smart object*. Já a comunicação entre estes *smart objects* e a aplicação em execução no *smartphone* ocorreu por meio da tecnologia Bluetooth. No entanto, o *middleware* de IoT utilizado (M-Hub) possui um serviço, denominado S2PA, que abstrai a tecnologia de comunicação utilizada pelos *smart objects* por meio de uma interface padrão para a interação com os mesmos. Assim o requisito funcional que refere-se à necessidade da infraestrutura de *software* oferecer suporte à diversos protocolos de comunicação (RF1) é contemplado. Ainda, diante do caso de uso 1 observa-se que a aplicação de TV pôde se adequar diante dos dados do contexto recebidos, posto que os dados recebidos pelos serviços de sensoriamento do *smart object obj_of_sensing* foram exibidos na tela da TV ao passo em que se coletavam os mesmos, contemplando assim o requisito funcional que tange à necessidade da aplicação de TV se adequar diante dos dados de contexto do ambiente físico (RF4).

O suporte à semântica especificado no requisito funcional RF6 é contemplado pelos casos de uso quando o protótipo, ao descobrir os *smart objects* presentes no ambiente (ambos sem valores semânticos em suas identificações – *obj_of_action* e *obj_of_sensing*), realiza consultas em uma base de conhecimento em busca de informações semânticas sobre os mesmos. Isso possibilita que os componentes que fazem parte da infraestrutura de *software* pudessem utilizar as informações não mais pelos dados físicos dos *smart objects*, mas pela sua semântica. Com isso, evitou-se a ambiguidade sobre os dados trafegados pela infraestrutura de *software*, permitindo que as aplicações de TV pudessem trocar dados com os *smart objects*, visto que os dados trabalhados por cada componente possuíam os mesmos significados em cada um deles.

Finalmente, com relação ao requisito não funcional, todas as comunicações entre a TV e os *smart object* é realizada no âmbito de redes locais, que possuem baixa

latência. A comunicação entre as aplicações de TV e a aplicação em execução no *smartphone* é realizada através de uma rede local Wi-Fi, enquanto que a comunicação entre o *middleware* de IoT e os *smart objects* varia de acordo com a tecnologia de comunicação de curto alcance utilizada pelo *smart object*.

7.2 Conclusão

Este capítulo apresentou uma avaliação por casos de uso sobre o protótipo desenvolvido no Capítulo 6. A avaliação sobre esse protótipo ocorreu devido o mesmo fazer uso de uma implementação concreta da arquitetura proposta, permitindo então, averiguar os requisitos especificados no desenvolvimento da arquitetura.

Foram apresentados três casos de uso que ilustraram a adoção do protótipo para atender os objetivos especificados neste trabalho, a saber: permitir interações multimodais dos telespectadores com a aplicação TV; permitir que a aplicação de TV possa alterar aspectos dos ambientes físicos de percepção audiovisual; e permitir à aplicação de TV a ciência do contexto no qual a TV se encontra.

Uma discussão apontando como foi realizada a completude dos requisitos funcionais e não funcionais expressos no Capítulo 5 também foi exibida. Ainda, foi apresentada uma comparação entre implementações de aplicações de TV com e sem o uso do nosso protótipo considerando os seguintes aspectos: a quantidade de linhas implementada; a complexidade ciclomática; e o tamanho resultante da aplicação.

8 Trabalhos Relacionados

Outros trabalhos também compartilham de nossos objetivos de permitir a integração entre aplicações de TV e *smart objects*, além de permitir interações dos telespectadores com essas aplicações. Esses trabalhos foram organizados em três contextos, a saber: "Mecanismos de Interação do Usuário com a Aplicação em Execução na TV Digital" em que são apresentados trabalhos que propõem formas de permitir ao telespectador a possibilidade de interagir com o conteúdo apresentado na TV; "Uso da TV Digital como Interface para o Uso de *Smart Objects* em *Smart Homes*" em que são apresentados trabalhos cujo objetivo é viabilizar a comunicação entre aplicações de TV e os *smart objects* presentes no ambiente físico de modo que se possa utilizá-los por meio da TV; e "Favorecimento do Processo de Imersão do Usuário ao Conteúdo Apresentado na TV Digital Através de *Smart Objects*" em que são mostrados trabalhos cujo foco é aumentar a sensação de imersão ao conteúdo apresentado na TV por meio da atuação de *smart objects* para gerar efeitos sensoriais no ambiente físico.

8.1 Mecanismos de Interação do Usuário com a Aplicação em Execução na TV Digital

Punt el al. em [40] visualizam a TV Digital Interativa (iDTV) como um dispositivo que possibilita a apresentação de *games multiplayers* em que os usuários desses *games* podem estar geograficamente próximos ou distantes. O objetivo desse trabalho é possibilitar que os usuários de TV possam interagir com diversos *games* por meio de dispositivos móveis, como *smartphones* e *tablets*, fazendo uso dos sensores internos aos mesmos para promover diversas formas de interação. Para tanto, foi desenvolvido um *framework* que permite que *games* possam ser desenvolvidos fazendo uso da TV como ambiente de apresentação do mesmo, além de oferecer suporte aos dispositivos móveis para apresentação de dados complementares ao *game*. Nesse trabalho o *game* pode estar alocado tanto em servidores nas nuvens ou em servidores

locais, e a comunicação entre os dispositivos móveis e o *game* em execução é realizada fazendo uso do protocolo TCP/IP.

O trabalho de Rosa et al. [42] almeja permitir novos modos de interações dos telespectadores com um conteúdo televiso e ainda identificar quem os realizou. Com isso, é possível que as aplicação de TV possam direcionar conteúdo adaptativo a cada telespectador, uma vez que se armazena o histórico de interações juntamente com dados contextuais do conteúdo apresentado, como data, hora, geolocalização, canal apresentado, dentre outros. As interações multimodais geradas pelos telespectadores são realizadas por meio de um dispositivo móvel, como *smartphone* e *tablets*. Nesse trabalho é pressuposto que esses dispositivos são utilizados, geralmente, por um individuo por vez, desta forma é possível identificar o usuário que está gerando comandos para aplicação de TV. Ainda, utilizou-se a Internet para que tanto as aplicações de TV quanto as aplicações dos dispositivos móveis pudessem consumir serviços web, que também foram desenvolvidos no trabalho, para gerenciar acesso dos telespectadores e para armazenar dados de interações dos mesmos. O envio de comandos dos dispositivos móveis para as aplicações de TV ocorre por meio do Wi-Fi. Por fim, nesse trabalho não se utiliza semântica para identificar os telespectadores e as interações realizadas com a aplicação de TV.

Em [31], Lima et al. propuseram uma arquitetura para convergir dispositivos de TV, a Internet e dispositivos móveis (*e. g. smartphones*). Essa arquitetura consiste na utilização de aplicações em dispositivos móveis e nas TVs, permitindo realizar a troca de dados entre os mesmos por meio de um rede local. As aplicações de TVs lançam mensagens de *broadcast* na rede, enquanto que as aplicações dos dispositivos móveis o respondem quando os recebem, estabelecendo uma comunicação entre eles. A ideia é permitir que os telespectadores possam interagir com as aplicações de TV por meio de aplicações de dispositivos móveis, e ainda utilizar os dados de contexto do conteúdo apresentado para uso em redes sociais *online* dos mesmos.

Os trabalhos de Punt et al. [40], Rosa et al. [42] e Lima et al. [31] possibilitam novos modos do usuário interagir com o conteúdo em apresentação na TV. No entanto, o trabalho [40] permite comandos de interações dos usuários do *game* por meio de uma associação entre interfaces dos *players*, oferecidos pelo *framweork* proposto, com o dispositivo móvel que acessa o *game*. Desse modo, não se sabe os perfis dos

usuários que estão interagindo com a aplicação. Já o trabalho [42] identifica os usuários assumindo que os dispositivos móveis são acessados por um único indivíduo por vez juntamente com um sistema de login para identificar o indivíduo. Porém, não utilizam um modelo conceitual para representar os perfís dos usuários, deixando para a aplicação do dispositivo móvel definir esse modelo de representação. Por fim, a arquiteutura apresentada em [31] permite interações multimodais dos usuários por meio de *smartphones*, mas também não identifica os usuários que realizam as interações multimodais com a aplicação de TV. A nossa proposta pressupõe que a presença de um dispositivo portátil no ambiente de apresentação implica na presença do seu respectivo proprietário, essa abordagem é semelhante à apresentada em [42]. Com isso, nossa proposta permite identificar todos os telespectadores presentes no ambiente físico. Para enviar os comandos de interações multimodais de todos os telespectadores para a aplicação de TV é utilizado uma abordagem descentralizada de análise de dados dos sensores em que os dispositivos portáteis *slaves* publicam apenas o evento gerado, já anotado semanticamente, no dispositivo *master*. Essa notação semântica é realizada utilizando o modelo conceitual IoTTV-Ont, que também conceitua uma pessoa e seu perfil. A abordagem que utiliza os dispositivos portáteis *slaves* para analisar seus fluxos de dados de sensores internos permite a adoção de modos distintos para a realização de uma interação com a aplicação de TV por parte dos telespectadores, visto que essas interações são relativos a cada indivíduo.

8.2 Uso da TV Digital como Interface para o Uso de *Smart Objects* em *Smart Homes*

Lucena et al. apresentam em [44] uma arquitetura que permite a TV receber dados provindos de sensores médicos, tais como sensor de temperatura, medidor de pressão arterial e oxímetro de pulso. O objetivo é permitir que os usuários possam visualizar, ou receber notificações, na tela da TV sobre dados extraídos de sensores médicos atrelados a si ou dispostos no ambiente físico. Para sua proposta, utilizou-se um *gateway* de serviços, denominado *Open Services Gateway Initiative* (OSGi). O OSGi permite que o sensores se conectem a ele (outros equipamentos domésticos também podem se conectar) e forneçam seus dados de medição. A arquitetura proposta

prevê o recebimento, a análise e a entrega desses dados para as aplicações de TV. Ainda, os autores destacaram o fato de não possuir um padrão para a leitura dos dados de sensores, uma vez que cada fabricante implementa seus próprios padrões de propriedade pela rede, sendo um ponto problemático.

Algumas empresas comerciais também vem se dedicando a desenvolver soluções para convergir dispositivos de TV com o gerenciamento de *smart objects*. Kapri et al. patentearam em [30] um dispositivo de TV que permite o mesmo participar de uma rede da IoT. Mais precisamente, a TV pode adotar ações que alterem seu estado mediante mensagens recebidas de *smart objects* conectados a ela, como também pode enviar mensagens para um controlador de IoT poder gerenciar tais *smart objects*. Para tanto, a TV possui um centro de processo de dispositivos da IoT, que tem entre outras finalidades a de enviar e receber mensagens de um controlador de IoT. Esse serviço de controle de IoT pode ser implementado em diversas formas, incluindo a utilização de *hub* separado ou integrado à TV. Este por sua vez se conecta com os *smart objects* presentes no ambiente físico. Nesse trabalho, as alterações realizadas na TV se limitam as gerenciadas pelo sistema operacional da mesma, como intensidade do volume de som, por exemplo.

No trabalho de Lucena et al. [44] os autores destacaram o fato de não possuir um padrão para a leitura dos dados de sensores médicos, devido serem produzidos por fabricantes distintos, um problema a ser enfrentando em futuros trabalhos. Já o trabalho de Kapri et al. [30] sugere a utilização de um *hub* de IoT para gerenciar os *smart objects* presentes no ambiente físico. Esse *hub* deve trocar mensagens com alguns módulos funcionais presentes na TV. As mensagens que a TV pode receber permite alterar apenas parâmetros do sistema operacional da TV, como o ajuste do nível de volume do som e do brilho. Não sendo possível alterações sobre a aplicação em execução na TV. Por outro lado, nossa proposta permite que as aplicações de TV possam estar cientes dos serviços ofertados pelos *smart objects* presentes no espaço físico, independentemente dos fabricantes dos *smart objects*. Pois, é utilizado a IoTTV-Ont para permitir que os identificadores dos *smart objects*, bem como os identificadores dos serviços ofertados pelos mesmos, possam ser armazenados com valores semânticos. Nossa proposta oferece ainda uma API para que as aplicações de TV possam consumir os dados de contexto do ambiente físico e com isso realizar alterações sobre a mesma. Além do mais, ao utilizar o ambiente de execução do

middleware de TV digital para reproduzir a aplicação de TV, esta pode utilizar recursos dessa para alterar parâmetros do sistema operacional da TV.

8.3 Favorecimento do Processo de Imersão do Usuário ao Conteúdo Apresentado na TV Digital Através de *Smart Objects*

Saleme et al. [43] propõem que aplicações de TV usem efeitos sensoriais definidos no padrão MPEG-V usando os metadados para efeitos sensoriais *Sensory Effect Metadata* (SEM) [37] com marcações de tempo relativos ao vídeo. Sua proposta consistiu em desacoplar os renderizadores de efeitos sensoriais dos reprodutores de vídeos. Desta forma, foi possível reutilizar o renderizador dos efeitos sensoriais em outros dispositivos, como as TV interativas. Assim, aplicações de TV em execução em seu *middleware* de TV digital é capaz de enviar comandos MPEG-V para o renderizador desenvolvido na proposta (SE Renderer) através de um serviço UPnP, que por sua vez aciona um dispositivo Arduíno, uma vez que este está conectado à esse. Em seus experimentos, o dispositivo Arduíno foi capaz de apresentar efeitos sensoriais que controlam uma lâmpada e atuadores de vibração e vento.

Guedes et al. [24] apresentam uma abordagem de interação entre dispositivos IoT em sincronismo ao conteúdo apresentado na TV brasileira. Sua proposta utiliza o ambiente declarativo Ginga-NCL, do *middleware* Ginga, para enviar comandos a um dispositivo Intel Galileo[1] com o kit *Seeed Studio Grove*[2]. O trabalho objetiva gerar imersão ao usuário por meio da atuação dos equipamentos elétricos presentes na residência do usuário. A comunicação entre o Galileo e os equipamentos elétricos ocorre por meio de um contato físico obtido por fios. A proposta faz uso de uma programação de *scripts* em Lua que recebe eventos da linguagem NCL para acionar os objetos com base na reprodução da mídia em execução. Nessa arquitetura, os dispositivos devem ser previamente conhecidos pelo componente gerenciador (Galileo) e também devem ser especificados na linguagem NCL. Desse modo, não é

[1] `www.intel.com/content/www/us/en/do-it-yourself/galileo-maker-board.html`

[2] `www.seeedstudio.com/item_detail.html?p_id=1978`

possível a inserção de um novo dispositivo de geração de efeitos sensoriais durante a execução de uma mídia na TV.

Diante dos trabalhos apresentados, percebe-se que os trabalhos de Guedes et al. [24] e Saleme et al. [43] tem como objetivo gerar efeitos sensoriais com intuito de aumentar o grau de imersão do usuário em relação do conteúdo apresentado. Guedes et al. [24] apresentam uma estrutura limitada aos *smart objects* pré-definidos pela aplicação. Enquanto Saleme et al. [43] utilizam UPnP para se conectar aos renderizadores de efeitos sensoriais, que por sua vez aciona os *smart objects* conectados a ele. As propostas apresentadas em ambos os trabalhos não possuem suporte a diversos protocolos de comunicação dos *smart objects*. Ainda, este último não possui mecanismos para que o conteúdo em reprodução possa ser ajustado diante dos dados de contexto do ambiente físico. Já a nossa proposta utiliza um *middleware* de IoT (M-Hub) que realiza buscas periódicas de *smart objects* próximos, como também realiza uma comunicação com os mesmos independentemente do protocolo de comunicação de curto alcance utilizado. A nossa proposta também permite uma comunicação bidirecional entre as aplicações de TV e os *smart objects* permitindo que as aplicações de TV possam se adequar aos dados contextuais do ambiente físico.

8.4 Discussão

Ao final de cada grupo de trabalhos relacionados apresentado foi feita uma análise comparativa com a nossa proposta em que são analisados à luz dos requisitos especificados no Capítulo 5. A taxonomia adotada para os trabalhos relacionados agrupou trabalhos cujos objetivos se assemelham. Desse modo, um grupo de trabalhos relacionados aborda alguns requisitos, enquanto outros trabalhos abordam outros requisitos. Para elencar os requisitos abordados por cada trabalho relacionado foi desenvolvida a Tabela 8.1 que possui duas colunas: a coluna à esquerda apresenta uma lista dos trabalhos relacionados e a coluna à direita apresenta os requisitos abordados por ele.

As propostas presentes nos trabalhos levantados não possuem recursos semânticos para a descoberta de *smart objects* e seus serviços. A utilização de semântica para a descoberta de *smart objects* permite que a aplicação de TV não

Tabela 8.1: Tabela dos requisitos abordados pelos trabalhos relacionados

Trabalho	Requisitos abordados
Punt et al.	RF2, RF5
Rosa et al.	RF5
Lima et al.	RF2, RF5
Lucena et al.	RF1, RF2, RF4
Kapri et al.	RF1, RF2, RF3, RF4
Saleme et al.	RF2, RF3
Guedes et al.	RF3, RF4

seja dependente de determinados *smart objects*, uma vez que seus fabricantes podem denotar propriedades específicas em seus produtos, incluindo o nome que identifica o *smart object* em uma rede, por exemplo. Também observa-se que não foram utilizados, em nenhum desses trabalhos, recursos semânticos para descrever os perfis dos usuários que interagem com as aplicações de TV.

Em contrapartida, nossa proposta utiliza a IoTTV-Ont, desenvolvida também neste trabalho, para representar o espaço físico de uma casa e das pessoas presentes nela. A utilização desse modelo conceitual permite ainda que os dados trocados entre as aplicações de TV e o *middleware* de IoT possam ter o mesmo significado. Assim nossa proposta também permite uma comunicação bidirecional baseada em valores semânticos.

Por fim, nossa proposta concentra as três categorias apresentadas, permitindo a alteração de aspectos do espaço físico por meio de aplicações da TV, como também possibilita o recebimento de dados de sensores do ambiente para as aplicações de TV. Já a utilização de sensores capazes de captar áudios e movimentos, por exemplo, permitem que o usuário interaja com a aplicação de TV por meio de comandos de voz e gestos, respectivamente. Esses dados podem ser extraídos do próprio *smartphone* no qual se executa uma aplicação que faz uso de nossa infraestrutura de *software*.

8.5 Conclusão

Neste capítulo foi apresentada uma relação de trabalhos relacionados nos quais foram organizados em três contextos: "Mecanismos de Interação do Usuário com a Aplicação em Execução na TV Digital", "Uso da TV Digital como Interface para o Uso de *Smart Objects* em *Smart Homes*" e "Favorecimento do Processo de Imersão do Usuário ao Conteúdo Apresentado na TV Digital Através de *Smart Objects*". Foi realizada uma comparação entre os trabalhos relacionados em cada grupo com a nossa proposta e a mesma foi descrita ao término da apresentação de cada categoria.

Por fim, foi apresentada uma discussão em que observou-se que nenhum dos trabalhos relacionados oferecerem algum recurso semântico para o tratamento dos dados dos *smart objects*, das pessoas e do próprio ambiente no qual se encontra a TV. Em comparação com os demais trabalhos, a nossa proposta oferece diversos suportes, tais como: à descoberta dinâmica dos *smart objects*, à heterogeneidade dos protocolos de comunicação e dos identificadores dos *smart objects*, à semântica, às interações multimodais por parte dos telespectadores, dentre outros.

9 Conclusão e Trabalhos Futuros

Este trabalho objetivou a integração entre aplicações de TV e dispositivos da IoT. O foco foi permitir que as aplicações de TV pudessem consumir dados e serviços ofertados pelos *smart objects* presentes no espaço físico de apresentação, para com isso alterar aspectos físicos do mesmo, bem como perceber o contexto no qual a TV se encontra.

No entanto, muitos desfios que circundam essa convergência foram apontados neste trabalho, a saber: a descoberta dinâmica de *smart objects* e dos serviços que eles disponibilizam; o suporte à heterogeneidade dos protocolos de comunicação dos *smart objects*; o suporte à heterogeneidade dos dados de identificação dos *smart objects*; a representação do ambiente físico de apresentação; a identificação dos telespectadores e seus perfis; a sincronização da mídia em reprodução com os *smart objects* presentes no ambiente físico; e abstrações de programação para o desenvolvimento de aplicações de TV que explorem os recursos da IoT.

Podemos destacar as principais contribuições deste trabalho como segue:

- Definição de um modelo conceitual que descreve o espaço físico de apresentação do conteúdo televisivo através de uma abordagem baseada em ontologia: um dos desafios abordados neste trabalho tange à necessidade de exportar para a aplicação da TV o conhecimento do espaço físico, os *smart objects* e pessoas presentes nele. Para tanto, foi desenvolvida uma ontologia, denominada IoTTV-Ont (Capítulo 4), que permite representar um ambiente residencial, os *smart objects* e as pessoas presentes na mesma. Adicionalmente, esta ontologia permite descrever as interações dos telespectadores com as aplicações de TV.

- Desenvolvimento de uma arquitetura de *software* (Capítulo 5) que possibilita a integração entre os *middleware* de TV e IoT. Esta arquitetura é baseada no modelo conceitual definido neste trabalho e define um conjunto de componentes responsáveis por exportar para a aplicação da TV o contexto do espaço físico de apresentação, permite à aplicação da TV atuar sobre este espaço físico e disponibiliza novas formas de interação do usuário com a aplicação em execução

na TV. Esta arquitetura foi projetada para ser independente de plataformas específicas de *middleware* para TV digital e IoT.

- Um protótipo de uma implementação concreta da arquitetura de *software* proposta baseado no *middleware* M-Hub como plataforma de IoT, e no *middleware* Ginga-NCL como plataforma de TV digital. Este protótipo está disponível para *download* no site do projeto IoTTV juntamente com sua documentação.

A partir do protótipo implementado neste trabalho foi realizada uma avaliação por meio de casos de uso em que pôde-se comprovar que os requisitos funcionais e não funcionais definidos para a arquitetura proposta foram atendidos. Nesses casos de uso as aplicações de TV alteraram aspectos do ambiente físico de apresentação, tais como a intensidade de iluminação e acionamento de sistemas de ventilação de forma sincronizada com a reprodução de um conteúdo multimídia; receberam dados de sensores do ambiente físico, e com isso foi possível ajustar o conteúdo apresentado mediante a entrada desses dados; e puderam receber dados sobre os telespectadores presentes no espaço físico de apresentação, bem como as interações realizadas por diversos modos por meio dos seus respectivos dispositivos portáteis.

Alguns caminhos podem ser seguidos para estender este trabalho, entre os quais destacamos:

- Diversas linguagens de desenvolvimento de aplicação de TV não possuem suporte nativo à troca de dados entre as aplicações de TV e os *smart objects*. Um exemplo disso é a linguagem declarativa NCL utilizada pelo *middleware* Ginga-NCL. Devido à falta de primitivas nessa linguagem que permita a troca de dados entre aplicações em NCL e *smart objects*, pretendemos estender o trabalho de Guedes et al. [23], incorporando a nossa infraestrutura de *softwares* às extensões apresentadas por eles.

- A utilização de dispositivos móveis, como *smartphones* e *tablets*, por telespectadores durante a apresentação de um conteúdo televisivo é uma prática que já vem sendo utilizada. Assim é interessante permitir que o conteúdo apresentado na televisão seja estendido para dispositivos móveis como uma

segunda tela. Portanto, pretende-se estender este trabalho para que a arquitetura de *software* ofereça suporte às aplicações em segunda tela.

- Desenvolvimento de uma ferramenta de autoria para auxiliar os programadores no desenvolvimento de aplicações de TV que explorem recursos de IoT conforme a proposta apresentada neste trabalho.

Referências Bibliográficas

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376, Jun 2015.

[2] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, Jun 2010.

[3] S. Bandyopadhyay, M. Sengupta, S. Maiti, and S. Dutta. Role of middleware for internet of things: A study. *International Journal of Computer Science and Engineering Survey*, 2(3):94–105, Aug 2011.

[4] L. Biljana and V. Kire. A review of internet of things for smart home: Challenges and solutions. *Journal of Cleaner Production*, 140:1454 – 1464, Oct 2017.

[5] D. Bonino and F. Corno. Dogont - ontology modeling for intelligent domotic environments. In *The Semantic Web - ISWC 2008: 7th International Semantic Web Conference, ISWC 2008*, pages 790–803, Berlin, Heidelberg, Oct 2008. Springer.

[6] D. Brickley and L. Miller. Foaf vocabulary specification 0.91, 2007.

[7] J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: Implementing the semantic web recommendations. In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers &Amp; Posters*, WWW Alt. '04, pages 74–83, New York, NY, USA, 2004. ACM.

[8] P. Cesar and K. Chorianopoulos. The evolution of tv systems, content, and users toward interactivity. *Foundations and Trends® in Human–Computer Interaction*, 2(4):279–373, Aug 2009.

[9] M. A. Chaqfeh and N. Mohamed. Challenges in middleware solutions for the internet of things. In *Collaboration Technologies and Systems (CTS), 2012 International Conference on*, pages 21–26, Denver, CO, USA, May 2012. IEEE.

[10] H. Chen, T. Finin, and A. Joshi. The soupa ontology for pervasive computing. In *Ontologies for Agents: Theory and Experiences*, pages 233–258, Basel, Switzerland, 2005. Birkhäuser Basel.

[11] C. De Salles, S. Neto, S. Diniz, S. Diniz Junqueira Barbosa, L. Fernando, L. F. Soares, and R. Ferreira Rodrigues. Capítulo 1: Desenvolvimento de aplicações declarativas para tv digital interativa. 05 2015.

[12] EsperTech. Esper - complex event processing. `http://www.espertech.com/esper/`. Accessed: 20 oct 2017.

[13] O. Etzion and P. Niblett. *Event Processing in Action*. Manning Publications Co., Greenwich, CT, USA, 1st edition, 2011.

[14] P. Eugster, P. Felber, R. Guerraoui, and A. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, Jun 2003.

[15] European Broadcasting Union, Sophia Antipolis, FRANCE. *Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification*, 1.0.3 edition, Aug 2006.

[16] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Padrões de Projetos: Soluções Reutilizáveis de software orientados a objetos*. Bookman editora, Porto Alegre, RS, 2 edition, 2009.

[17] G. Gill and C. Kemerer. Cyclomatic complexity density and software maintenance productivity. *IEEE Transactions on Software Engineering*, 17(12):1284–1288, Dec 1991.

[18] B. Gomes, L. Muniz, F. J. Silva, L. E. T. Ríos, and M. Endler. A comprehensive and scalable middleware for ambient assisted living based on cloud computing and internet of things. *Concurrency and Computation: Practice and Experience*, 29(11):e4043, Dec. e4043 cpe.4043.

[19] L. Görgü, B. Kroon, A. Campbell, and G. O'Hare. Enabling a mobile, dynamic and heterogeneous discovery service in a sensor web by using androsixth. In *Ambient Intelligence: 4th International Joint Conference, AmI 2013, Dublin, Ireland, December 3-5, 2013. Proceedings*, pages 287–292, Cham, 2013. Springer International Publishing.

[20] T. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, Jun 1993.

[21] N. Guarino. *Formal Ontology in Information Systems: Proceedings of the 1st International Conference June 6-8, 1998, Trento, Italy*. IOS Press, Amsterdam, The Netherlands, The Netherlands, 1st edition, 1998.

[22] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.*, 29(7):1645–1660, Sept. 2013.

[23] A. Guedes, R. Azevedo, S. Colcher, and S. Barbosa. Extending ncl to support multiuser and multimodal interactions. In *Proceedings of the 22Nd Brazilian Symposium on Multimedia and the Web*, Webmedia '16, pages 39–46, Teresina, Piauí, Brazil, 2016. ACM.

[24] Á. Guedes, M. Cunha, H. Fuks, S. Colcher, and S. Barbosa. Using ncl to synchronize media objects, sensors and actuators. In *XXII Simpósio Brasileiro de Sistemas Multim'idia e Web (Vol. 2): Workshops e Sessão de Pôsteres.*, pages 184–189, Teresina, Piauí State, Brazil, 2016. Webmedia.

[25] U. Hunkeler, H. Truong, and A. Stanford-Clark. Mqtt-s; a publish/subscribe protocol for wireless sensor networks. In *Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on*, pages 791–798, Jan 2008.

[26] IBM and Eurotech. Mqtt v3.1 protocol specification. `www.public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html`.

[27] I. B. M. C. (IBM). Mqtt v3.1 protocol specification, 2010.

[28] ITU-T. Nested context language (ncl) and ginga-ncl. Recommendation H.761, International Telecommunication Union, Geneva, Nov. 2014.

[29] C. Jacquet, A. Mohamed, and Y. Bellik. An Ambient Assisted Living Framework with Automatic Self-Diagnosis. *International Journal On Advances in Life Sciences*, 5(1):10p., Jun 2013.

[30] A. Kapri, Y. Adalgeirsson, K. Kator, J. Chung, and H. Holtzman. Television (tv) as an internet of things (iot) participant, feb 2017. US Patent App. 15/087, 813.

[31] E. Lima and R. Rabêlo. An architectural model for communication between the idtv and mobile devices. In *Computing, Networking and Communications (ICNC), 2015 International Conference on*, pages 1102–1105, Feb 2015.

[32] D. Luckham. The power of events: An introduction to complex event processing in distributed enterprise systems. In *Rule Representation, Interchange and Reasoning on the Web*, pages 3–3, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[33] D. Miorandi, S. Sicari, F. D. Pellegrini, and I. Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497 – 1516, Apr 2012.

[34] M. Nagy, A. Katasonov, O. Khriyenko, S. Nikitin, M. Szydlowski, and V. Terziyan. Challenges of middleware for the internet of things, automation control - theory and practice. In *Automation Control-Theory and Practice*, pages 247–270. InTech, Dec 2009.

[35] K. Nahrstedt, H. Li, P. Nguyen, S. Chang, and L. Vu. Internet of mobile things: Mobility-driven challenges, designs and implementations. In *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 25–36, April 2016.

[36] N. Noy and D. McGuinness. Ontology development 101: A guide to creating your first ontology. Technical report, Stanford, CA, 94305, march 2001.

[37] O. of Standardization (ISO). *IEC 23005-3 Information technology - Media context and control – Part 3: Sensory information*, Jul 2016.

[38] C. Perera, P. Jayaraman, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Mosden: An internet of things middleware for resource constrained mobile devices. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on*, pages 1053–1062, Waikoloa, HI, USA, Jan 2014. IEEE.

[39] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context aware computing for the internet of things: A survey. *IEEE Communications Surveys & Tutorials*, 16(1):414–454, May 2014.

[40] M. Punt, M. Bjelica, V. Zdravkovic, and N. Teslic. An integrated environment and development framework for social gaming using mobile devices, digital tv and internet. *Multimedia Tools and Applications*, 74(18):8137–8169, Sep 2015.

[41] W. C. Roberto Ierusalimschy, Luiz Henrique de Figueiredo. *Lua Reference Manual*. PUC-Rio, 5.1 edition, aug 2006.

[42] R. Rosa and V. Lucena. Contextualizing and capturing individual user interactions in shared itv environments. *Multimedia Tools and Applications*, 76(6):8573–8595, Mar 2017.

[43] E. Saleme and C. Santos. PlaySEM: A platform for rendering MulSeMedia compatible with MPEG-v. In *Proceedings of the 21st Brazilian Symposium on Multimedia and the Web*, WebMedia '15, pages 145–148, Manaus, Brazil, Oct 2015. ACM.

[44] V. Silva, O. Maia, M. Rodrigues, and V. Lucena. Universal system for integrating commercial medical devices with standardized digital tv system. In *2016 IEEE International Conference on Consumer Electronics (ICCE)*, pages 301–304, Jan 2016.

[45] L. Soares. Sincronismo das coisas não é apenas expressão da moda. In *XX Brazilian Symposium on Multimedia and the Web: Workshops.* Webmedia, 2014. Porto Alegre: Sociedade Brasileira de Computação.

[46] L. Soares and S. Barbosa. *Programando em NCL 3.0 2a. Edição Versão 2.1*. Elsevier Campos, 2 edition, May 2012.

[47] L. Soares, M. Moreno, and C. Neto. Ginga-ncl: Declarative middleware for multimedia iptv services. *IEEE Communications Magazine*, 48(6):74–81, Jun 2010.

[48] L. Soares, R. Rodrigues, and M. Moreno. Ginga-NCL: the declarative environment of the Brazilian digital TV system. *Journal of the Brazilian Computer Society*, 12:37–46, 2007.

[49] L. Sommaruga, T. Formilli, and N. Rizzo. Domoml: An integrating devices framework for ambient intelligence solutions. In *Proceedings of the 6th International Workshop on Enhanced Web Service Technologies*, WEWST '11, pages 9–15, Lugano, Switzerland, Sep 2011. ACM.

[50] A. Standard. *DTV APPLICATION SOFTWARE ENVIRONMENT LEVEL 1 (DASE-1) PART 1: INTRODUCTION, ARCHITECTURE, AND COMMON FACILITIES*. Advanced Television Systems Committee.

[51] I. Strategy and P. Unit. Itu internet reports 2005: The internet of things. *Geneva: International Telecommunication Union (ITU)*, 1:62, 2005.

[52] Y. Sulema. Mulsemedia vs. multimedia: State of the art and future trends. In *2016 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 1–5, Bratislava, Slovakia, Jul 2016. IEEE.

[53] L. Talavera, M. Endler, I. Vasconcelos, R. Vasconcelos, M. Cunha, and F. Silva. The mobile hub concept: Enabling applications for the internet of mobile things. In *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 123–128, March 2015.

[54] W3C OWL Working Group. OWL 2 Web Ontology Language, W3C Recommendation. `www.w3.org/TR/owl-overview/`.

[55] J. Wang. *Encyclopedia of Business Analytics and Optimization*. IGI Global, 71 E. Chocolate Avenue Hershey PA 17033, Feb 2014.

[56] A. Wehbi, A. Cherif, and C. Tadj. Modeling ontology for multimodal interaction in ubiquitous computing systems. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 842–849, Pittsburgh, Pennsylvania, Sep 2012. ACM.

[57] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin. Iot gateway: Bridgingwireless sensor networks into internet of things. In *Embedded and Ubiquitous Computing (EUC), 2010 IEEE/IFIP 8th International Conference on*, pages 347–352, Hong Kong, China, Dec 2010. IEEE.