LETICIA RAMOS

UMA PROPOSTA DE ONTOLOGIA PARA RESIDÊNCIAS INTELIGENTES BUSCANDO A INTEGRAÇÃO DE DISPOSITIVOS

Dissertação submetida ao Programa de Pós-Graduação em Computação Aplicada da Universidade Tecnológica Federal do Paraná como requisito parcial para a obtenção do título de Mestre em Computação Aplicada.

Curitiba PR

Agosto 2014

LETICIA RAMOS

UMA PROPOSTA DE ONTOLOGIA PARA RESIDÊNCIAS INTELIGENTES BUSCANDO A INTEGRAÇÃO DE DISPOSITIVOS

Dissertação submetida ao Programa de Pós-Graduação em Computação Aplicada da Universidade Tecnológica Federal do Paraná como requisito parcial para a obtenção do título de Mestre em Computação Aplicada.

Área de concentração: Engenharia de Sistemas Computacionais
Linha de Pesquisa: Sistemas de Informação

Orientador: Prof. Dr. Roberto Cesar Betini

Curitiba PR

Agosto 2014

Ramos, Leticia
R175p Uma proposta de ontologia para residências inteligentes
2014 buscando a integração de dispositivos / Leticia Ramos.--
2014.
xii, 94 f: il.; 30 cm

Texto em português, com resumo em inglês
Dissertação (Mestrado) - Universidade Tecnológica
Federal do Paraná. Programa de Pós-Graduação em Computação
Aplicada, Curitiba, 2014
Bibliografia: f. 89-94

1. Ontologia. 2. Computação ubíqua. 3. Automação residencial
4. Redes de sensores sem fio. 5. Computação - Dissertações.
I.Betini, Roberto Cesar. II.Universidade Tecnológica
Federal do Paraná - Programa de Pós-graduação em Computação
Aplicada. III. Título.

ATA DA DEFESA DE DISSERTAÇÃO DE MESTRADO 20

DISSERTAÇÃO PARA OBTENÇÃO DO GRAU DE MESTRE EM **COMPUTAÇÃO APLICADA**

No dia 28 de agosto de 2014, às 9:00 horas, reuniu-se na Sala B-202 - bloco C - 2º andar do Câmpus Curitiba, a banca examinadora composta pelos professores doutores:

Prof. Roberto Cesar Betini, Dr. (Presidente)	UTFPR - CT
Prof. João Alberto Fabro, Dr.	UTFPR - CT
Prof. Antonio Morais da Silveira, Dr.	UFPA
Prof. Marco Aurélio Wehrmeister, Dr.	UTFPR - CT

sob Presidência de **Roberto Cesar Betini** para examinar a dissertação da candidata **LETÍCIA RAMOS**, intitulada**:** "Uma proposta de Ontologia para Residências Inteligentes buscando a Integração de Dispositivos"**.** Após a apresentação, a candidata foi arguida pelos examinadores e foi dada a palavra aos presentes para formularem perguntas à candidata. Os examinadores reunidos deliberaram pela _____ da dissertação.

A candidata foi informada que a concessão do referido grau, na área de concentração Engenharia de Sistemas Computacionais, está condicionada à (i) satisfação dos requisitos solicitados pela Banca Examinadora e lavrados na documentação entregue à Candidata; (ii) entrega da dissertação em conformidade com as normas exigidas pela UTFPR; e (iii) entrega da documentação necessária para elaboração do Diploma. A Banca Examinadora determina um **prazo de _____ dias** para o cumprimento dos requisitos (desconsiderar esse parágrafo caso a dissertação seja reprovada).

_____

_____

_____

Nada mais havendo a tratar, a sessão foi encerrada às _____, dela sendo lavrada a presente ata que segue assinada pela Banca Examinadora e pela Candidata.

_____
Prof. **Roberto Cesar Betini, Dr.**
presidente - (UTFPR - CT)

_____
Prof. **João Alberto Fabro, Dr.**
(UTFPR - CT)

_____
Prof. **Antonio Morais da Silveira, Dr.**
(UFPA)

_____
Prof. **Marco Aurélio Wehrmeister**
(UTFPR - CT)

Candidata: _____

DECLARAÇÃO PARA A OBTENÇÃO DO GRAU DE MESTRE

A coordenação do Programa declara que foram cumpridos todos os requisitos exigidos pelo Programa de Pós-Graduação para a obtenção do grau de mestre.

"

Curitiba, ____de _____de 20____.

A Ata de Defesa original está arquivada na Secretaria do PPGCA".

Este trabalho é dedicado à minha mãe, Maria Inês Marin.

Agradecimentos

Agradeço primeiramente à Deus por me dar forças e sabedoria para completar meu mestrado.

Agradeço à minha família pelo apoio. Em especial minha mãe Maria Inês e meu esposo Rodrigo por todo carinho, paciência, incentivo e orações. Sem vocês eu não chegaria ao fim deste caminho.

Agradeço também ao meu orientador Roberto Cesar Betini por todo apoio, paciência e ensinamentos recebidos, permitindo completar este trabalho.

Aos professores Marco Aurélio Wehrmeister, João Alberto Fabro, Adolfo Gustavo Serra Seca Neto e Gustavo Alberto Giménez Lugo, por todas as revisões e sugestões durante os seminários.

Resumo

Uma residência inteligente é capaz de adquirir e aplicar conhecimentos sobre um ambiente e, de forma autônoma, adaptar-se aos seus habitantes. Estes espaços contêm uma diversidade de dispositivos que precisam interagir, porém, também existem no mercado diversos padrões para automação. Esta heterogeneidade dificulta a interoperabilidade entre dispositivos tornando a aplicação dependente de um único fornecedor para que a comunicação seja efetiva. Da mesma forma, sistemas domóticos possuem serviços dinâmicos e a adição ou mobilidade de novos dispositivos ao longo do tempo requer não somente que aplicações possam comunicar-se, mas também identificar que novos serviços foram adicionados e, quando necessário, estabelecer uma comunicação pertinente ao novo contexto. Finalmente, sistemas de automação também precisam de alguma forma perceber o mundo real e interpretá-lo. Os sinais recebidos por sensores não possuem qualquer valor se não forem interpretados em relação ao contexto da residência. Ontologias podem auxiliar a resolver estes três problemas representando formalmente o domínio. Ela permite uma compreensão comum das informações através de um modelo semântico e possibilita a execução de suposições explícitas para o contexto da residência. Assim, esta dissertação propõe uma ontologia para integração em uma residência inteligente que representa o ambiente, seus dispositivos, indivíduos e os agentes de software permitindo que sejam criadas as relações para a interação. A consistência da ontologia foi avaliada em relação às questões de competência e um protótipo foi criado para validar a aplicação prática do modelo.

Palavras chaves: Ambientes inteligentes, residências inteligentes, ontologias, computação ubíqua.

Abstract

A smart home is able to acquire and apply knowledge about the environment and autonomously adapt to its inhabitants. These spaces contain a diversity of devices that need to interact and there are also many commercial protocols for automation. This heterogeneity hinders the interoperability between devices turning the application dependent on a single technology for an effective communication. In the same way, domotic systems have dynamic services and the mobility of new devices from time to time requires the ability to discovery when a new service have been added and, when necessary, establish a new communication related to this new context. Finally, automation systems also need to perceive the real world and interpret it. The signals received from sensors have no value unless they are interpreted against the context of the residence. Ontologies can help to solve these three problems formally representing the domain. It enables a common understanding of information through a semantic model and enables the execution of explicit assumptions to the context of residence. Thus, this dissertation proposes integration ontology for a smart house that represents the environment, devices, people and software agents creating the relationships and reasoning about rules for interaction. The consistency of the ontology was assessed in relation to its competence and a prototype was created to validate the practical application of the model.

Keywords: Ontology, Smart Home, Pervasive Computing, Context-Aware Computing

Sumário

Lista De Ilustrações

Lista De Tabelas

Lista De Abreviaturas

ACL	*Agent Communication Language*
AID	*Agent Identifier*
AMS	Sistema Gerenciador de Agentes
AP	Plataforma de Agentes
BDI	*Beliefs, Desires, Intentions*
CIF	Classificação Internacional de Funcionalidade, Incapacidade e Saúde
DF	*Directory Facilitator*
DL	*Description Logic*
DDO	*Device Description Ontology*
EIA	*Electronic Industries Association*
EIB	*European Installation Bus*
EHS	*European Home Systems Protocol*
FHSS	*Frequency-hopping spread spectrum*
FIPA	*Foundation for Intelligent Physical Agents.*
HTTP	*Hypertext Transfer Protocol*
HVAC	*Heating, Ventilation, and Air Conditioning*
IHC	Interface Humano Computador.
IIOP	*Internet Inter-Orb Protocol*
ISO	*International Organization for Standardization*
KQML	*Knowledge Query and Manipulation Language*
MTS	*Message Transport Service*
OMS	Organização Mundial da Saúde
OSI	*Open Systems Interconnection*
OWL	*Web Ontology Language*
RDF	*Resource Description Framework*
SL	*Semantic Language*
TCP/IP	*Transmission Control Protocol / Internet Protocol*
UPnP	*Universal Plug and Play*
USB	*Universal Serial Bus*

W3C	*World Wide Web Consortium*
WAP	*Wireless Application Protocol*
WEP	*Wired Equivalent Privacy*
WPA	*Wi-Fi Protected Access*
XML	*eXtensible Markup Language*

Capítulo 1

Introdução

A automação de residências, também denominada domótica, é considerada uma das áreas mais promissoras em relação ao desenvolvimento de tecnologias eletrônicas [Chan, 2008]. A redução do custo de hardware aliada a softwares embarcados em minúsculos chips permite que novas aplicações sejam desenvolvidas e que a tecnologia esteja acoplada em diversos dispositivos do dia a dia das pessoas [Augusto, 2010].

Weiser (1991) afirma que um dia a tecnologia estará integrada e acoplada tão fortemente em objetos e ambientes que não será mais percebida pelo usuário. A esta visão, deu o nome de computação ubíqua. Assim, torna-se um dos requisitos de sistemas inteligentes, o dever de agir proativamente, antecipando as necessidades do usuário e suas preferências. O ambiente deve de alguma forma obter conhecimento das necessidades e preferências do usuário, porém liberando-o do ônus da prestação de informação ou programação de qualquer dispositivo, tanto quanto possível.

Cook (2005) considera inteligente uma residência capaz de adquirir e aplicar conhecimentos sobre um ambiente e, de forma autônoma, adaptar-se aos seus habitantes. Estes espaços contêm uma diversidade de sensores, atuadores e dispositivos que precisam interagir, porém, também existem no mercado diversos padrões para automação, como por exemplo, os protocolos de domótica: X10, KNX, LonWorks, UPnP. Esta heterogeneidade dificulta a interoperabilidade entre dispositivos tornando a aplicação dependente de um único fornecedor para que a comunicação seja efetiva [Miori et al., 2010].

Assim, Residências Inteligentes ainda estão longe de serem alcançadas. Apesar do grande número de pesquisas sobre esse contexto, apenas alguns aplicativos foram disponibilizados para o ambiente doméstico e público em geral. Isto ocorre principalmente devido à segmentação de padrões e soluções proprietárias. Embora as casas modernas possam ser equipadas com aparelhos tecnológicos ditos “inteligentes”, ainda poucos destes aparelhos podem ser facilmente conectados uns aos outros [Miori et al.. 2010].

Em sistemas domóticos os serviços são dinâmicos e a adição de novos dispositivos ao longo do tempo requer não somente que aplicações possam comunicar-se, mas também identificar/descobrir quais novos serviços foram adicionados e, quando necessário, estabelecer uma comunicação pertinente ao contexto desta nova aplicação. Além disso, se uma determinada aplicação não exerce o seu papel corretamente no sistema ela poderá afetar de forma inadequada os demais serviços. Ou seja, para que todos os serviços e tecnologias de uma residência possam interagir é necessária uma interoperabilidade tanto no nível sintático como semântico [Edwards, 2001].

Para interagir com o ambiente um sistema de automação também precisa de alguma forma de perceber o mundo real e interpretá-lo. O processo de percepção ocorre através de sensores espalhados pela residência, porém os sinais recebidos não possuem qualquer valor se não forem interpretados em relação ao contexto da residência. Por exemplo, um sensor de presença envia um sinal quando percebe uma variação brusca na temperatura do ambiente.

Mas a aplicação precisa entender o que este sinal significa em relação à automação da residência e informações como: a posição do sensor na residência, o horário em que ocorreu a ação e as funcionalidades que utilizam os dados deste sensor podem ajudar a entender e contextualizar o valor em questão. Uma maneira de representar estas informações em relação à residência é através de instâncias de uma ontologia.

Uma ontologia, para o contexto computacional, é um tipo de formalismo para a representação do conhecimento. Ela permite uma compreensão comum das informações, possibilita o reuso de domínios de conhecimento e faz suposições explícitas de um domínio [Noy, 2001]. A aplicação de ontologias para sistemas domóticos inteligentes podem auxiliar na construção do raciocínio que pode ser utilizado para diagnóstico, controle e otimização de dispositivos. Segundo Strang et al. (2004) ontologias correspondem a técnica mais promissora para modelagem de contexto.

Esta dissertação pretende solucionar o problema de integração entre aplicações de automação residencial que precisam compartilhar de um mesmo modelo semântico. Uma ontologia comum define o vocabulário para consultas, solicitações e afirmações entre agentes de software e garante a consistência das informações. Porém o problema é definir um modelo que descreva o contexto da residência em relação a quem, o que, onde, quando e por que [Abowd et al., 2000]. Desta forma, uma ontologia denominada AssistiveHome é proposta com o objetivo de permitir a interoperabilidade entre diversos sensores e aplicações. Por último a ontologia será aplicada de forma prática em um arcabouço para validar o modelo.

1.1 Motivação

O aspecto multidisciplinar do domínio de residências inteligentes compreende: tecnologia de sensores e seu hardware; a rede de comunicação, topologia e protocolos; algoritmos de coleta e processamento dos dados; algoritmos para tomada de decisão e contexto [Cook et al., 2005]. Desta forma, uma ontologia que represente todo o contexto de uma residência em relação a quem, o que, onde, quando e por que pode auxiliar na consistência do significado dos dados entre as diversas camadas que compõe uma aplicação de automação inteligente.

A modelagem semântica de uma residência também está presente no trabalho de diversos autores. Dibowski (2010) e Ploennigst (2012) utilizam uma ontologia como modelo de integração para dispositivos heterogêneos. O arcabouço DomoML [Sommaruga, 2005] e DogOnt [Bonino, 2008] propõe ontologias que descrevem o ambiente, os dispositivos e funcionalidades. Abdulrazak (2010) inclui o conceito de tempo e localização absoluta no seu modelo ontológico de residência. A ontologia SOUPA [Chen, 2004] também é utilizada em aplicações de ambientes inteligentes através de classes mais generalistas.

Uma ontologia é uma visão de um domínio do ponto de vista dos seus desenvolvedores, para esta dissertação acredita-se que o usuário e os softwares também fazem parte de uma residência inteligente. Apesar de existirem ontologias para descrever os conceitos acima citados, a criação de um modelo unificado e robusto permite a atualização dos relacionamentos entre as classes, auxiliando não somente na interoperabilidade, mas também oferecendo um modelo de inferência consistente para modelagem de contexto e regras.

1.2 Objetivos

O objetivo desta dissertação é propor um modelo semântico unificado de residências inteligentes que auxilie na interoperabilidade entre as diversas aplicações, através de uma ontologia

1.2.1 Objetivos Específicos

Os objetivos específicos da dissertação são:

- Criar uma ontologia referente ao contexto de uma residência inteligente.
- Aplicar a ontologia em um arcabouço para interoperabilidade entre as aplicações de um ambiente inteligente.
- Validar o modelo por meio de uma aplicação prática.
- Comparar a ontologia com o estado da arte.

1.3 Organização do Trabalho

Este documento está dividido em sete capítulos. O capítulo dois apresenta uma revisão de conceitos relacionados a residências inteligentes, um referencial teórico para Agentes além de descrever conceitos referentes a ontologias. O capítulo três faz uma análise dos trabalhos correlatos à proposta desta dissertação. O capítulo quatro apresenta o modelo semântico proposto. O capítulo cinco demonstra a aplicação prática para validação da ontologia. O capítulo cinco discute sobre os resultados alcançados. Por fim, o capítulo seis expõe as conclusões obtidas através da validação da proposta.

Capítulo 2

Materiais e Métodos

Este capítulo descreve os conceitos teóricos pertinentes a dissertação. Primeiramente são apresentados conceitos relativos a uma residência inteligente, que serão utilizados para o desenvolvimento da ontologia *AssistiveHome*. Em seguida, são apresentados conceitos referentes a ontologias e métodos de desenvolvimento. Finalmente, são abordados conceitos relativos a sistemas de agentes, pois serão utilizados na construção do arcabouço de validação da ontologia.

2.1 Residências Inteligentes

O conceito de Residência Inteligente pode ser considerado um subdomínio da área de Ambientes Inteligentes. Segundo Alam (2012), em uma definição recente, residência inteligente é uma aplicação da computação ubíqua [Weiser, 1991] que é capaz de prover serviços automatizados sensíveis ao contexto ou assistivos na forma de inteligência ambiental, controle remoto ou automação residencial. Os serviços oferecidos têm por objetivo o conforto, economia de energia, saúde, ou segurança dos usuários.

Para que um ambiente possa ser considerado inteligente os sistemas devem ser projetados para perceber as características dos usuários, raciocinar sobre os dados armazenados e então selecionar a ação que melhor beneficiará o usuário neste ambiente [Cook, 2007]. Expandindo-se esse conceito, conforme Figura 1, é possível identificar diversos fatores que compõe um sistema de automação. Na base da organização encontram-se os dispositivos que correspondem a sensores, atuadores ou demais eletrônicos que possam de alguma forma perceber ou atuar sobre o ambiente e os usuários. Então, os dispositivos interconectados por uma rede de comunicação, com ou sem fio, enviam os dados para armazenamento e processamento através de algum software com objetivo de prestar serviços para o usuário. Os principais serviços oferecidos tem a finalidade de aumentar o conforto, auxiliar na reabilitação e monitoramento da saúde e oferecer segurança ao usuário [Chan et al., 2008].

Através da análise de todos os itens que compõe uma residência inteligente é possível construir um modelo de domínio que suporte a interoperabilidade entre os sistemas que compõe o ambiente. Dessa forma, serão analisados a seguir conceitos e diversas tecnologias pertinentes ao tema.

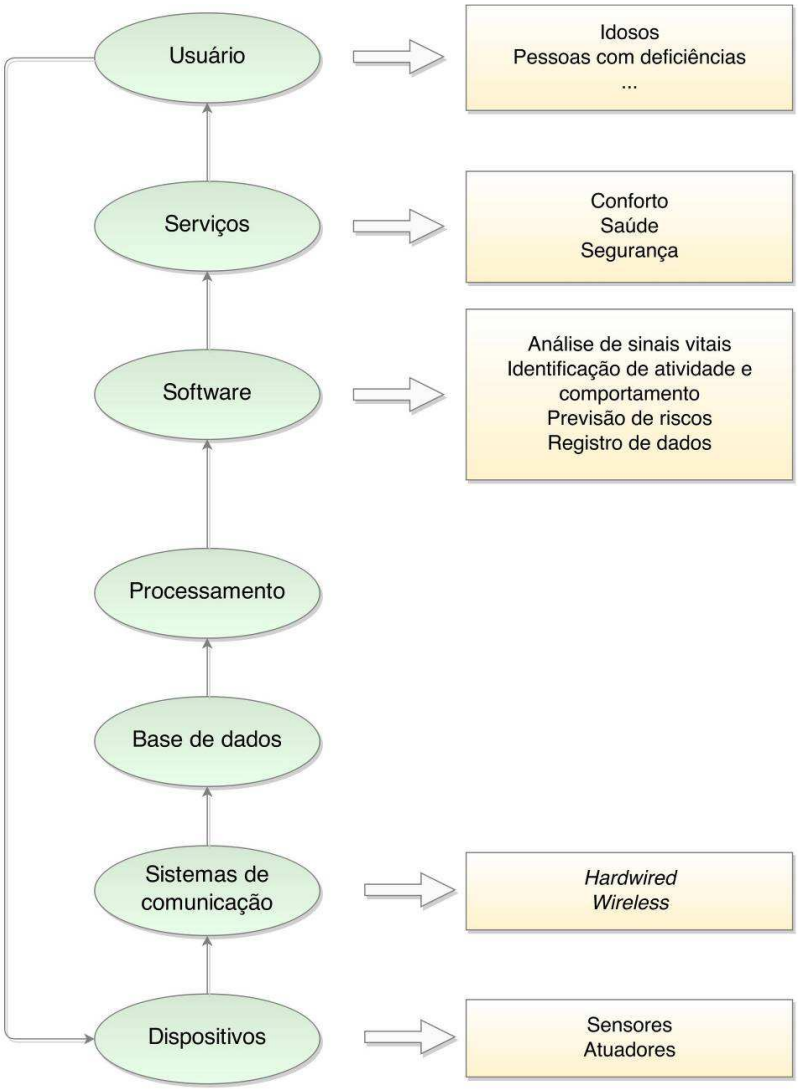

Figura 1: Organização de um ambiente inteligente adaptado de Chan (2008).

2.1.1 Funcionalidades

As funcionalidades de uma residência inteligente devem suprir as necessidades específicas de cada usuário e podem variar de acordo com o público alvo do sistema. Os sistemas domóticos comerciais oferecem suporte às funções mais comumente requisitadas divididas em grupos de controles [Sommaruga et al., 2011]. Apesar de não ser um fator limitante, para a criação da ontologia, é importante classificar as funcionalidades existentes tanto para agrupamento das classes quanto para mapeamento de dispositivos e tecnologias utilizados para suporte as funções. Assim, são listadas as principais funções de acordo com [Bolzani, 2004]:

- Energia Elétrica: existem diversas aplicações para monitoramento, balanceamento e estabilização da rede.

- Ar-condicionado: controla a temperatura e ventilação da residência.

- Iluminação: atua no acionamento de lâmpadas e também oferece suporte para as funções de consumo de energia.
- Segurança: corresponde aos sistemas de detecção de intrusos, alarmes e controles de acesso.
- Combate a incêndio: monitora a presença de fumaça e fogo.
- Multimídia: controla funções de áudio e vídeo.
- Água e dejetos: oferece controle para válvulas, abastecimento de água, e escoamento de dejetos
- Comunicação: permite a comunicação com entidades externas através de diferentes meios.

2.1.2 Sensibilidade ao contexto

A computação sensível ao contexto permite que as informações obtidas no mundo físico sejam processadas com a finalidade de fornecer serviços para o usuário de acordo com uma determinada situação [Hoareau et al,. 2009]. Aplicações sensíveis ao contexto devem adaptar-se constantemente às mudanças no ambiente, que podem ser referentes a atributos físicos, fisiológicos, psicológicos, comportamentais, de atividades ao longo do tempo, entre outros.

Para que a adaptação ocorra o agente de software deve preocupar-se com a atividade de aquisição, abstração e compreensão dos dados sensoriais, reconhecendo o contexto para um determinado momento. Abowd et al. (2000) apresenta uma classificação para a modelagem de contexto por meio de cinco questões: *Who, What, Where, When* e *Why* (Quem, O que, Onde, Quando e Por que).

- *Who* (Quem): aquilo que realiza uma ação. Pode ser tanto uma pessoa quanto um dispositivo ou aplicação.
- *What* (O que): informações das atividades ou ações executadas por uma pessoa ou dispositivo.
- *Where* (Onde): refere-se à localização dos objetos ou pessoas no ambiente.
- *When* (Quando): toda informação de tempo em relação a atividades, como início, fim e tempo de duração.
- *Why* (Por que): relaciona-se ao motivo pelo qual uma ação foi executada.

A aquisição de contexto através de sensores é uma tarefa complexa. Com a heterogeneidade dos dados adquiridos por diferentes dispositivos, a aplicação necessita de uma etapa para interpretação dos dados de forma a ser útil para a aplicação. A informação contextual é dinâmica por natureza e as alterações necessitam ser capturadas ao longo do

tempo. Um dos maiores desafios da computação ubíqua é executar a ação correta no momento correto.

Um sistema de automação inteligente é um grande desafio para a computação sensível ao contexto, ela deve levar em consideração o contexto do usuário em relação à localização, atividade e estado. Da mesma forma, os dispositivos também podem ser móveis e apresentar diferentes estados e funções dependendo do momento.

2.1.3 Perfil de Usuários

O desenvolvimento de ambientes inteligentes não deve levar em consideração apenas o ponto de vista tecnológico. O perfil do usuário é importante para determinar os seus papéis, necessidades, preferências, ou limitações, de forma a proporcionarem respostas personalizadas e adequadas no espaço e no tempo, antecipando a vontade do usuário sem que para tal tenha que existir uma mediação consciente [Röcker et al., 2014].

Qualquer indivíduo pode beneficiar-se de produtos de automação, seja para conforto, segurança ou saúde. Porém para Chan (2008) os maiores beneficiários da tecnologia são idosos, pessoas com deficiência ou doenças crônicas. A Classificação Internacional de Funcionalidade, Incapacidade e Saúde (CIF) [WHO, 2001], desenvolvida pela Organização Mundial da Saúde (OMS) descreve as funcionalidades e incapacidades relacionadas às condições de saúde, identificando o impacto também no ambiente. Neste sentido, a automação residencial assistiva pode auxiliar o indivíduo na execução de atividades diárias de forma independente, diminuindo as barreiras relacionadas aos fatores ambientais e proporcionando certo grau de independência [Ramos et al., 2014a].

A descrição do perfil do usuário e barreiras funcionais na ontologia associadas a dispositivos e atividades permite a execução de ações automatizadas. A relação entre as classes respeitando a classificação de contexto (Quem, O que, Onde, Quando e Por que) permite que as mensagens comunicadas entre as aplicações da residência sejam também pertinentes ao usuário e não somente aos dispositivos envolvidos.

2.1.4 Dispositivos, sensores e atuadores

Sensores e atuadores representam os componentes mais importantes de um ambiente inteligente, eles correspondem aos dispositivos de entrada e saída do sistema, permitindo a interligação entre a computação e o ambiente físico. Existe uma variedade muito grande de atuadores e sensores que possibilitam a execução de ações e o monitoramento de inúmeras grandezas físicas e eventos.

Porém, transformar os dados de sensores em informação é uma tarefa complexa. Eles possuem características únicas que desafiam as técnicas de análise de dados convencionais. Muitas vezes os dados precisam ser tratados em tempo real e de forma contínua incorporando informações temporais e espaciais para obter significado. Algumas vezes os valores também são imprecisos, gerando ruídos [Cook, 2007].

Da mesma forma, os atuadores interligam-se com outros dispositivos da residência para executar uma ação. Por exemplo, alterar a temperatura do sistema de ar condicionado ou acionar uma lâmpada.

As características dos sensores e atuadores são relevantes para a dissertação, uma vez que deles dependem diretamente as capacidades de uma residência inteligente. Em Bolzani

(2004) e Chan (2008) é possível encontrar uma lista de diversos dispositivos que atuam em um ambiente.

2.1.5 Padrões e protocolos utilizados em Automação Residencial

Para suporte ao modelo semântico proposto nesta dissertação é necessário conhecer os modelos e protocolos existentes para a área de automação residencial. Assim é possível criar uma ontologia que atenda as características de cada tecnologia, principalmente quanto à especificação de funções e comandos de dispositivos. Os protocolos mais utilizados são os seguintes:

ZigBee: É um protocolo de radiofrequência baseado no padrão IEEE 802.15.4. Ele possui baixas taxas de transmissão de dados (250kb/s), porém com baixo consumo de energia. O alcance padrão entre os dispositivos é de 30 m para ambientes internos e 100 m para externos [Rathnayaka et al., 2011].

CEBus: O Protocolo Consumer Electronic Bus foi criado pela EIA (Associação de Indústrias Eletrônicas). O CEBus é uma arquitetura aberta que define protocolos para as comunicações de aparelhos em diferentes meios físicos como: linhas de força, par trançado de baixa voltagem, cabo coaxial, infravermelho, RF e fibra ótica. O endereçamento do dispositivo é feito por hardware e tem 4 bilhões de possibilidades. O CEBus permite que a rede opere ponto-a-ponto sem a necessidade de um nó controlador. O padrão também oferece uma linguagem para controle de objetos que inclui comandos tais como aumentar volume, avançar rápido, voltar, pausar e pular [Desbonnet, 1997]. O padrão CEBus desenvolveu uma linguagem universal para comunicação entre dispositivos de automação denominada CAL (*Common Application Language*). A linguagem é orientada a objeto e define classes, instâncias e métodos para comandos binários, contínuos, discretos entre outros [Khawand, 1991]. Por exemplo a classe Binary Sensor modela a operação de um sensor que possui estados binários, verdadeiro ou falso, como ligado e desligado. Outro exemplo é a classe Analog Sensor que modela uma quantidade física e contínua como temperatura.

Protocolo X-10 O sistema X-10 foi originalmente desenvolvido nos anos 70 pela empresa Pico Electronics, na Escócia. Porém a patente original expirou em dezembro de 1997 e diversos fabricantes iniciaram a fabricação e expansão de dispositivos seguindo este protocolo. O X-10 utiliza comunicação por rede elétrica utilizando um pulso de sinal na freqüência de 60hz AC, quando o sinal cruza o ponto zero da curva de freqüência. O padrão permite até 256 endereços. Uma mensagem básica em X-10 usa 13 bits: 4 para o código de entrada, 4 para o código do ambiente, 4 para função ou unidade e o último bit indica se os 4 anteriores correspondem a função ou unidade. Uma das limitações do padrão é a de operar apenas funções simples tipo liga/desliga e dimerização de luzes [Kovatsch et al., 2010; Alam et al., 2012].

***Universal Plug and Play*:** O protocolo de Automação Residencial UPnP foi criado em 1999 pelo Fórum UPnP. Este protocolo foi desenvolvido a partir do protocolo PnP (*Plug and Play*) para suportar configuração automática entre computadores e dispositivos utilizando padrões existentes para Internet como HTML e XML. Cada novo dispositivo conectado deve compartilhar uma descrição em XML contendo informações do fabricante e serviços

oferecidos, incluindo uma lista de comandos e ações. O Fórum UPnP especifica modelos de controle para alguns dispositivos de automação, entre eles dispositivos para segurança, controle de iluminação e HVAC [Alam et al., 2012].

EIB/KNX: É um padrão ISO e aberto de automação baseado no modelo OSI e dominante na Europa. Ele foi criado através da convergência de outros três padrões, o *European Installation Bus* (EIB), *BatiBUS* e *European Home Systems Protocol* (EHS). O protocolo permite a comunicação por cabo, rádio-frequência, linha de potência ou IP/Ethernet. Existem inúmeros dispositivos disponíveis para a tecnologia que oferecem suporte para serviços de iluminação, segurança, gestão de energia, aquecimento, monitoramento e sinalização.

2.2 Ontologias

Ontologia é um tipo de formalismo utilizado para a representação do conhecimento que objetiva: prover uma compreensão comum de uma estrutura de informação entre pessoas ou agentes de software, possibilitar o reuso de domínios de conhecimento, fazer suposições explícitas de um domínio e separar o domínio de conhecimento do domínio operacional. Para atingir este fim a ontologia define entidades, classes, propriedades, predicados e funções e as relações entre componentes [Noy et al., 2001].

Guarino (1998) apresenta uma definição próxima do objetivo da ontologia nesta dissertação conceituando-a como “um artefato constituído por um vocabulário usado para descrever uma dada realidade, mais um conjunto de fatos explícitos em relação ao sentido pretendido para as palavras do vocabulário. Este conjunto de fatos tem a forma da teoria da lógica de primeira ordem, onde as palavras do vocabulário aparecem como predicados unários ou binários”.

Posteriormente, Gruber (2009) delimita o conceito de ontologia para o contexto da computação afirmando que “uma ontologia define um conjunto de representações primitivas com os quais é modelado o domínio do conhecimento ou discurso. A representação de primitivas são tipicamente classes, atributos e relações. A definição da representação primitiva inclui a informação sobre o seu significado e restrições de aplicações lógicas consistentes”.

Por estarem próximas da lógica de primeira ordem ontologias atuam no nível semântico enquanto bases de dados atuam no nível lógico ou físico. Devido a sua independência dos níveis de modelos de dados, ontologias trabalham como interfaces e são utilizadas para integrar bases heterogêneas, permitindo interoperabilidade entre sistemas distintos. Atualmente existem diversos padrões de linguagem e ferramentas, comerciais e gratuitas para criação e manipulação de ontologias.

2.2.1 Principais componentes de uma ontologia

Os componentes das ontologias variam conforme a linguagem utilizada, mas existem alguns conceitos comuns a qualquer ontologia:

- Classes: Modelam os conceitos de domínio ou tarefa. Usualmente estão organizadas em taxonomias e também podem ser aplicadas heranças. A taxonomia de uma classe é

representada através de uma estrutura em árvore. As classes podem ser concretas ou abstratas. Em contraste com as classes abstratas, as classes concretas podem ter instâncias diretas.

- Atributos: Representam as características dos conceitos e também podem ser chamados de slots, roles ou propriedades. Os atributos representam tipos de dados como números, strings, booleanos, etc.

- Relações: Modelam associações entre conceitos. Relações binárias podem ser utilizadas para expressar atributos de conceitos.

- Instâncias: Representam elementos específicos ou objetos de uma dada classe. Novas instâncias podem ser criadas e valores podem ser atribuídos aos atributos e relações.

- Funções: Representam casos especiais de relações, no qual o enésimo elemento da relação é único para n-1 elementos predecessores, ou seja, é uma relação em que um dado elemento tem uma relação única com um conjunto de outros elementos.

- Axiomas: Modelam frases sempre verdadeiras. Os axiomas são usados para a verificar a consistência da ontologia ou da consistência do conhecimento armazenado.

Existem diversas linguagens para representação de ontologias. Para a construção da ontologia AssitiveHome utilizou-se o modelo OWL (*Web Ontology Language*) recomendado pelo W3C (*World Wide Web Consortium*) [McGuinness et al., 2004]. Uma Ontologia OWL serve para descrever um domínio em termos de classes, propriedades e indivíduo, bem como suas respectivas propriedades e seus relacionamentos. Ela foi projetada para ser usada por aplicações que precisam processar o conteúdo da informação, facilitando assim a possibilidade de interpretação do conteúdo. A OWL é fundamentado na linguagem RDF (Resource Description Framework) [Lassila and Swick, 1999] que é um modelo de dados baseados em triplas de sujeito - predicado - objeto, também conhecido como metadados. O objeto de uma declaração RDF pode ser outro objeto ou um recurso literal com um identificador único chamado de URI.

2.2.2 Classificação de ontologias

Guarino (1998) classifica as ontologias de acordo com sua dependência em relação a uma tarefa específica ou a um ponto de vista do conteúdo representado:

- Ontologias de alto nível: descrevem conceitos de forma generalista como conhecimentos de senso comum independentes de um domínio.

- Ontologias de domínio: descrevem um vocabulário relacionado a um domínio genérico, porém introduz termos específicos em uma ontologia de alto nível. Por exemplo uma ontologia para ambientes inteligentes pode ser considerada como de domínio.

- Ontologias de tarefa: descrevem uma tarefa ou atividade, especializando também os termos da ontologia de alto nível.

- Ontologia de aplicação: descreve conceitos dependentes de um domínio ou tarefa particular, sendo que estas ontologias geralmente estendem ou especializam as ontologias de domínio e de tarefa. A ontologia proposta pode ser considerada uma ontologia de aplicação, pois apresenta conceitos específicos para a comunicação entre sistemas para automação residencial. Ou seja, AssitiveHome é uma especialização do domínio de ambientes inteligentes.

2.2.3 Metodologia para Modelagem de Ontologias

Existem diversas metodologias para desenvolvimento de ontologias como, por exemplo, O4IS (Ontology for Information Systems) [Kabilan, 2007], METHONTOLOGY [Gómez-Perez et al., 2004], e Ontology Development 101 [Noy et al., 2001]. Para a escolha da metodologia a ser utilizada primeiramente é preciso definir os objetivos que se deseja atingir com a ontologia e o tipo de conhecimento que ela irá representar. Assim, entre as diversas alternativas para a modelagem, é possível determinar qual alternativa será a mais adequada para o domínio proposto [Noy et al., 2001; Gómez-Perez et al., 2004].

A O4IS é uma metodologia orientada para a conceituação, concepção e manutenção de uma ontologia. É adequada para pessoas com pouca experiência. A metodologia O4IS tem como núcleo um conjunto de dez etapas, que são orientadas para a definição e desenvolvimento de ontologias de domínio.

A METHONTOLOGY propõe uma metodologia inspirada no ciclo de vida de software para o refinamento do protótipo da ontologia. A evolução do processo de desenvolvimento possui os seguintes passos: especificação, conceitualização, formalização, implementação e manutenção.

A metodologia *Ontology Development 101* consiste de um guia interativo desenvolvido para os utilizadores da ferramenta Protégé [Stanford, 2014] composto de sete passos:

1. Identificação do domínio e escopo da ontologia.
2. Reutilização de ontologias existentes.
3. Enumeração dos termos importantes da ontologia.
4. Definição de classes e sua hierarquia.
5. Definição das propriedades das classes.
6. Definição das restrições.
7. Criação de instâncias das classes.

A metodologia proposta por Noy et. al. (2001) foi utilizada para o desenvolvimento da ontologia AssitiveHome. A escolha deve-se ao formato simplificado da metodologia que permite a construção de ontologias em um tempo reduzido e principalmente ao número de recursos envolvidos no projeto, no caso somente o autor.

Além das metodologias de desenvolvimento de ontologias Gruber (1993) propôs alguns critérios necessários para uma boa ontologia e que podem ser utilizados no processo de validação. São eles:

- Clareza: uma ontologia deve ser capaz de comunicar de forma eficaz o seu significado para os seus utilizadores.
- Coerência: a ontologia deve suportar inferências que sejam consistentes com as suas definições.
- Extensibilidade: uma ontologia deve ser capaz de definir novos termos baseado em definições já existentes.
- Codificação Minimizada: os conceitos devem ser especificados sem depender de nenhum símbolo ou codificação linguística.
- Compromisso Ontológico Mínimo: uma ontologia não deve se restringir ao domínio a ser modelado, dando liberdade aos utilizadores de especializarem e instanciarem a ontologia da forma que necessitarem.

Além dos critérios propostos por Gruber (1993), Brank et al. (2005) sugere entre as possíveis abordagens para avaliar uma ontologia a comparação com um padrão, que pode ser outra ontologia e a utilização de uma aplicação para avaliar os resultados. Desta forma, a ontologia AssitiveHome será avaliada em relação a outras ontologias para verificar a completude do modelo em relação ao contexto de residências inteligentes e também será utilizada em uma arcabouço para avaliar se a ontologia permite a integração entre dispositivos e softwares.

2.3 Sistemas de Agentes

Um agente é uma entidade que executa um conjunto de operações com algum grau de independência ou autonomia e, executando estas operações, emprega algum conhecimento dos objetivos ou desejos do usuário. A sua principal característica está no fato de que não é necessário codificá-lo passo a passo a respeito de como alcançar este objetivo. Ele é capaz de perceber as preferências do usuário, e agir baseado neste conhecimento obtido, tornando a aplicação uma entidade ativa, com certo grau de autonomia e capaz de realizar tarefas que auxiliem o usuário no desempenho de suas atividades de acordo com seus interesses [Wooldridge, 2001].

Wooldridge (2001) apresenta um conjunto de propriedades desejáveis a um agente:

- Sociabilidade: de modo a interagir com outros agentes ou humanos através de algum tipo de linguagem de comunicação;
- Reatividade: de modo a perceber alterações em seu ambiente, reagindo a tempo;
- Pro-atividade: não só reagindo ao ambiente, mas tomando iniciativas quando conveniente;
- Autonomia: permitindo a execução de ações sem intervenção humana.

Em determinadas aplicações, algumas características são mais importantes que outras, portanto, um agente não precisa ter necessariamente todas elas. O comportamento do agente é dado pelo ambiente no qual ele está inserido e modelado através do seu respectivo projeto.

Sistemas multiagentes correspondem, comumente, a qualquer sistema composto por agentes que interagem entre si. Contudo, a área de inteligência artificial distribuída utiliza o termo apenas para um subconjunto de sistemas, dividindo-os em relação à forma de interação através de duas classes principais: Resolução Distribuída de Problemas e Sistemas Multiagentes [Durfee, et al. 1994].

- Resolução Distribuída de Problemas: corresponde a uma abordagem descendente, a solução é estruturada previamente e os agentes comportam-se conforme proposto inicialmente. Isto é, grande parte do raciocínio sobre a solução é inserido no sistema no momento do seu desenvolvimento, levando a controles geralmente hierárquicos e centralizados.

- Sistemas Multiagentes: corresponde a uma abordagem ascendente. A preocupação é desenvolver arquiteturas de agentes que interajam de forma autônoma e social, bem como desenvolver sistemas de comunicação e coordenação independentes do problema a ser resolvido.

O arcabouço para validação da ontologia utiliza a abordagem de resolução distribuída de problemas, sendo que esta escolha deve-se a confiabilidade necessária em um sistema de automação. Os agentes têm papéis bem definidos para controle de sensores, atuadores e processamento dos dados, e podem ser criados na medida em que novos serviços sejam necessários, seguindo o padrão de mensagens. Os agentes desenvolvidos são descritos no capítulo 5.

2.3.1 FIPA

Em 1996, diversas empresas e centros de pesquisas com atividades no campo de Sistemas Multiagentes fundaram a FIPA (*Foundation for Intelligent Physical Agents*). A FIPA é uma fundação internacional exclusivamente para a criação de padrões que tornem possível a implementação de agentes abertos e interoperáveis.

As especificações do FIPA são divididas em cinco áreas distintas: Aplicações de Sistemas Multiagentes, Arquiteturas Abstratas, Comunicação entre Agentes, Gerenciamento de Sistemas Multiagentes e Transporte de Mensagens entre Agentes. A seguir são apresentadas as especificações relevantes para esta dissertação.

Modelo de gerenciamento de agentes

O modelo de gerenciamento de agentes provê o ambiente onde um agente FIPA existe e opera, assim, ele estabelece um modelo lógico de referência para criação, registro, localização, comunicação, migração e desativação de agentes na plataforma FIPA. O desenvolvimento de cada um destes componentes não é objeto de padronização da FIPA, porém diversas plataformas de desenvolvimento disponibilizam estes serviços. Abaixo é apresentada uma descrição detalhada de cada um dos elementos do modelo:

- Plataforma de Agentes (*Agent Plataform* - AP): provê a estrutura física na qual os agentes podem ser executados. Uma AP consiste no computador, o sistema operacional, as aplicações de suporte aos agentes, os componentes de gerenciamento de agentes FIPA e os agentes.

- Agente: combina uma ou mais capacidades de serviços e pode incluir acesso a aplicativos externos, humanos e meios de comunicação. Um agente deve ter pelo menos um proprietário e possuir um identificador único.

- Facilitador de Diretórios (*Directory Facilitator* - DF): é um componente obrigatório da plataforma de agentes. Ele provê um serviço de páginas amarelas para os outros agentes. Os agentes podem registrar seus serviços com o DF ou requisitar ao DF que ele encontre serviços oferecidos pelos outros agentes. Múltiplos DF podem existir dentro de uma mesma plataforma de agentes.

- Sistema Gerenciador de Agentes (*Agent Management System* - AMS): é um componente obrigatório da plataforma de agentes. O AMS exerce o controle de acesso e uso da plataforma de agentes. Só pode existir um AMS numa única plataforma de agentes. O AMS apresenta um diretório de identificadores de agentes AID (*Agent Identifier*) que contém endereços de transporte para agentes registrados com a plataforma. O AMS oferece páginas brancas para os outros agentes. Cada agente deve se registrar com o AMS de forma a receber uma AID válida.

- Serviço de Transporte de Mensagens (*Message transport System* - MTS): fornece o mecanismo de transporte de mensagens entre agentes de uma mesma plataforma ou de plataformas diferentes. Todos os agentes FIPA têm acesso a, no mínimo, um MTS e somente mensagens endereçadas a agentes podem ser enviadas através do MTS. O MTS é disponibilizado pelo Canal de Comunicação de Agentes (ACC - *Agent Communication Channel*).

Os serviços de DF, AMS e MTS são normalmente oferecidos pelas plataformas de agentes, como por exemplo, JADE [Bellifemine et al., 2007], Jadex [Braubach et al., 2003], FIPA-OS [Poslad et al., 2000]. O registro de um novo agente na plataforma é obrigatório e necessário para a comunicação entre os agentes, porém, no caso desta dissertação, o serviço do DF será desenvolvido à parte para a melhor interação. Isso significa que além de registrar-se no DF e AMS padrão e obrigatório, o agente também deverá registrar-se como um indivíduo na ontologia através do agente com função de gerência. Já o canal de comunicação entre os agentes limita-se também em relação ao protocolo disponibilizado pelas plataformas e especificado pela FIPA.

Comunicação entre agentes

A forma de comunicação entre os agentes também possui um papel importante no arcabouço, pois definirá a maneira de interação entre os componentes do sistema. Necessita-se, portanto da definição de um protocolo de comunicação que especifique o exato processo de comunicação, um formato ou sintaxe para as mensagens e uma linguagem de comunicação que especifique o significado ou a semântica da informação. A semântica não se refere apenas ao conteúdo da mensagem, mas também ao seu tipo. Além disso, existem três aspectos para um estado formal de comunicação: a sintaxe, que é como os símbolos da comunicação estão

estruturados; a semântica, que é o que os símbolos significam; e a pragmática, que é como os símbolos são interpretados. [Huhns, 1999].

A FIPA-ACL é uma linguagem de comunicação de agentes. Cada mensagem da FIPA-ACL é composta por um ato comunicativo obrigatório e um conjunto de parâmetros. O mais comum é cada mensagem conter um emissor, um receptor e um campo de conteúdo. Os parâmetros podem ser alocados em qualquer posição dentro da mensagem. Os atos comunicativos das mensagens FIPA-ACL foram projetados para, dentro do possível, representar os atos da fala. O padrão FIPA está dividido em sete subcamadas responsáveis por uma parte do processo de comunicação. [Bellifemine et al., 2001]:

- Transportes: responsável pelo protocolo de transporte. A FIPA definiu protocolos de transporte de mensagens para IIOP (*Internet Inter-ORB Protocol*), WAP (*Wireless Application Protocol*) e HTTP (*Hypertext Transfer Protocol*).

- Codificação: define representações de mensagens para o uso de estruturas de nível mais elevado de dados.

- Estrutura de mensagens: a estrutura da mensagem é independente da codificação particular para incentivar a flexibilidade. Cada mensagem contém um conjunto de parâmetros variáveis de acordo com a situação e o único parâmetro de fato obrigatório pela FIPA é a *performative* ou ato comunicativo. A Figura 2 apresenta um exemplo de mensagem para a *performative Request.* Ela inclui as informações de identificação e endereço do remetente (:*sender*) e do destinatário (:*receiver*), o conteúdo da mengagem (:content), a linguagem do conteúdo (:language), a ontologia que provê significado também ao conteúdo (:ontology) e o protocolo de comunicação (:protocol), neste caso o fipa-request.

```
(REQUEST
:sender  (agent-identifier
:name sensor@192.168.0.19:1099/JADE
:addresses (sequence http://leticia:7778/acc
))
:receiver  (set (agent-identifier
:name LampadaQ@192.168.0.19:1099/JADE ))
:content  "((action (agent-identifier
:name @192.168.0.19:1099/JADE) (OnCommand
:functionality (Functionality
:functionality_inst OnOffFunctionality)
:controllable (Controllable
:controllable_inst SimpleLampBedroom)
:command_value On)))"
:language  fipa-sl
:ontology  AssistiveHome
:protocol  fipa-request
)
```

Figura 2: Exemplo de mensagem [Fonte: O Autor]

- Ontologia: o termo individual contido no conteúdo de uma mensagem FIPA pode ser explicitamente referenciado a um modelo conceitual ou ontologia. Para a automação residencial definiu-se a ontologia AssitiveHome que permite a comunicação das ações e estados de sensores, atuadores, usuários e o ambiente da residência.

- Conteúdo: o teor real de mensagens FIPA. A linguagem mais usada para expressar conteúdo é FIPA-SL. Para a comunicação dos agentes do arcabouço proposto será utilizada tanto a FIPA-SL quanto a linguagem SPARQL, suportados pela ontologia AssitiveHome. A linguagem de conteúdo FIPA-SL baseia-se na lógica de primeira ordem para representar afirmações, condições, razões e ações que atendam os requisitos dos atos comunicativos da FIPA-ACL. O significado do conteúdo das mensagens depende da ontologia compartilhada entre os agentes.

- Ato comunicativo: a classificação da ação ou ato do agente em relação ao conteúdo de uma mensagem. Por exemplo, o ato *inform* especifica que o conteúdo da mensagem é informativo e o ato *request* especifica que o conteúdo da mensagem é uma solicitação.

- Protocolo de interação: raramente as mensagens são trocadas de forma isolada, mas sim fazem parte de alguma sequência de interação. A FIPA define protocolos de interação, especificando as sequências de mensagens. Alguns destes protocolos são utilizados no arcabouço proposto, como, por exemplo:

    - *Request Interaction* [FIPA 2002a] foi desenvolvido para um agente solicitar a outro agente a execução de uma determinada ação. O agente participante pode aceitar ou recusar a solicitação, porém a mensagem de aceite não é obrigatória para ações processadas rapidamente. Após finalizar a ação o agente participante possui três opções de resposta: indicar uma falha, confirmar a execução e confirmar a execução e enviar os dados do resultado.

    - *Query Interaction* [FIPA 2002a], protocolo da FIPA presente no arcabouço desta dissertação é o FIPA *Query,* ele é utilizado quando um agente precisa solicitar alguma informação de outro agente. A mensagem inicial do protocolo pode ser para verificar se uma sentença é verdadeira, neste caso, o requisitante utiliza o ato comunicativo *query-if* ou para solicitar alguma informação, através do ato comunicativo *query-ref*. O agente receptor pode aceitar ou recusar a solicitação enviando uma mensagem. Caso aceite, ao termino do processamento o envia a resposta com os dados solicitados ou indicando falha.

2.4 Ferramentas e Plataformas de Agentes

Para suporte ao arcabouço proposto foram utilizadas algumas ferramentas. A primeira delas é o editor de ontologias Protégé [Stanford, 2014], utilizado para criar o modelo ontológico do domínio da residência e construir as propriedades e restrições entre as classes. Ela também permite à visualização gráfica da ontologia que para esta dissertação foi desenvolvida na linguagem OWL.

Os agentes do protótipo foram desenvolvidos utilizando o arcabouço JADE [Bellifemine et al., 2007] versão 4.3. O JADE é uma plataforma Java para desenvolvimento de agentes que segue os padrões da FIPA de arquitetura, protocolos de transporte e comunicação, linguagens de conteúdo e ontologias. Dessa forma, cada plataforma de agentes também possui as propriedades registro de agentes, descoberta de agentes e transferência de mensagens. Em conformidade com a especificação da FIPA, o JADE possui uma plataforma

de agentes principal e também permite que existam múltiplos hosts e múltiplas plataformas. Para o arcabouço proposto o número de plataformas depende da implementação do sistema na residência e somente a plataforma principal é mandatória, porém a possibilidade de utilização de múltiplos hosts permite que o processamento seja distribuído e que a gerência do sistema não se torne um gargalo na descoberta de serviços.

A FIPA prevê que uma plataforma de agentes tenha pelo menos um Sistema Gerenciador (AMS) e um Facilitador de Diretórios (DF). O arcabouço Jade segue o padrão e possui o DF e MAS nativos que são inicializados junto com a plataforma. Agentes localizados em diferentes hosts deverão registrar-se no facilitador de diretório e sistema de gerência principal. O protocolo padrão para comunicação entre os hosts é o Java *Remote Method Invocation* (RMI) e caso existam plataformas desenvolvidas em outras linguagens de programação o protocolo recomendado é *o* IIOP. Internamente, um agente JADE é desenvolvido baseado em uma lista de comportamentos que um agente deve executar para atingir os seus objetivos.

Além do JADE, o arcabouço JENA [Apache, 2014] é utilizado para manipulação da ontologia que representa o contexto da residência. O JENA oferece suporte para bases de dados RDF permitindo navegar e criar classes, propriedades e instâncias em uma ontologia através do Jena2 Ontology API. Os agentes da aplicação utilizam esta biblioteca para criar instâncias na ontologia e executar queries SPARQL. O SPARQL é uma linguagem inspirada na Structured Query Language (SQL), que possibilita a busca de informações em grafos RDF.

2.5 Discussão

Neste capítulo foi apresentada primeiramente uma visão geral sobre os conceitos que envolvem a automação residencial inteligente. Esta revisão é importante para criação da ontologia para o domínio. A pesquisa das tecnologias existentes no mercado e a classificação dos sensores permitem que o modelo semântico seja adequado aos sistemas reais. A interoperabilidade proposta na dissertação também está diretamente ligada às características destas tecnologias.

Este capítulo também abordou um referencial teórico sobre ontologias, as principais formas de representação e alguns exemplos de metodologias para o seu desenvolvimento. Estes conceitos são aplicáveis a qualquer ontologia, porém a revisão permite um embasamento para os próximos dois capítulos que fazem uma análise do estado da arte em relação ao domínio e abordam a criação da ontologia para residência inteligente. Uma ontologia pode ser utilizada como ferramenta para análise de contexto, dessa forma o entendimento deste conceito também é fundamental para a criação de uma ontologia que possibilite a inferência do contexto dos usuários e dispositivos em relação ao ambiente.

Por último, apresentou-se um referencial teórico sobre agentes e os conceitos básicos dos protocolos definidos pela FIPA. Estes conceitos são utilizados na modelagem do arcabouço para validação da ontologia. A linguagem de conteúdo FIPA-SL utiliza a ontologia para dar significado às palavras presentes no conteúdo da mensagem e permitir a comunicação entre os agentes. A integração dos dispositivos que possuem interfaces de software representadas por agentes é realizada através das mensagens compartilhadas que utilizam os atos comunicativos e protocolos da FIPA para informar ou requisitar uma alteração no ambiente. De forma breve foram também apresentadas as ferramentas necessárias para a reprodução da solução proposta.

Capítulo 3

Trabalhos Correlatos

Na ultima década surgiram diversos projetos para desenvolvimento de residências inteligentes e que abordam problemas em diferentes áreas, como redes de sensores, processamento de contexto, *middlewares* de comunicação e interfaces. Porém esta revisão aborda principalmente soluções que utilizaram ontologias para o domínio de residências. O estudo inclui ontologias para interoperabilidade, sensibilidade ao contexto, residências assistivas, descrição de dispositivos e ambientes inteligente. O objetivo principal é identificar a organização das ontologias, as vantagens e desvantagens de cada abordagem.

Alguns autores descrevem o arcabouço de automação além da ontologia, desta forma, quando disponível, também foi incluída uma breve descrição do seu funcionamento.

3.1 Ontologia *Smart Space Context*

Qin et al. (2007) apresenta um *middleware* multiagente para suporte a ambientes inteligentes sensíveis ao contexto. Ele utiliza uma ontologia combinada com lógica de primeira ordem probabilística para prover um entendimento comum do contexto, auxiliar no raciocínio sobre informações ambíguas e permitir um conhecimento compartilhado e reutilizável. A ontologia divide-se em um modelo central para conceitos sobre o ambiente e uma extensão para um domínio específico, um estudo de caso representando uma sala de aula.

A ontologia central, Figura 3, descreve sete conceitos:

- Usuário (classe *user*): descreve o perfil do usuário, informações de contato, preferências e humor.

- Localização (classe *location*): inclui as propriedades de longitude, latitude e altitude além de importar as propriedades da ontologia GeoSpatial do W3C [W3C, 2013].

- Tempo (classe *time*): corresponde à ontologia OWL-Time [W3C, 2006].

- Atividade (classe *activity*): o autor não descreve as propriedades da classe, porém ela é importante para validar as informações inconsistentes de contexto descrevendo as relações entre a atividade, tempo, localização e a probabilidade de ocorrência.

- Serviço (classe *service*): especifica vários níveis de serviços que a plataforma oferece para apoiar a descoberta e composição destes serviços. Esta classe é baseada na ontologia OWL-S [W3C, 2006] para descrição de *web services*.

- Ambiente (classe *environment*): descreve as condições do ambiente, como nível de ruído, condição de iluminação, umidade, temperatura.

- Plataforma (classe *platform*): descreve o hardware, como por exemplo, sensores e o software do ambiente

A ontologia foi desenvolvida para ambientes inteligentes sensíveis ao contexto de forma geral. Com objetivo de reduzir o tamanho da base de conhecimento, o autor sugere que novas classes sejam incluídas de acordo com as necessidades do domínio. Porém, para interoperabilidade entre aplicações é preciso descrever os conceitos de forma detalhada possibilitando a comunicação de forma efetiva. Por exemplo, não é possível solicitar a automação de um dispositivo apenas com a inclusão da classe principal, é preciso que a ontologia descreva qual o tipo de dispositivo, as propriedades de dados específicas e as funções oferecidas. Apesar de não contemplar conceitos específicos para uma residência, a ontologia proposta por Qin foi utilizada como uma das bases para o desenvolvimento das classes principais da ontologia AssistiveHome.

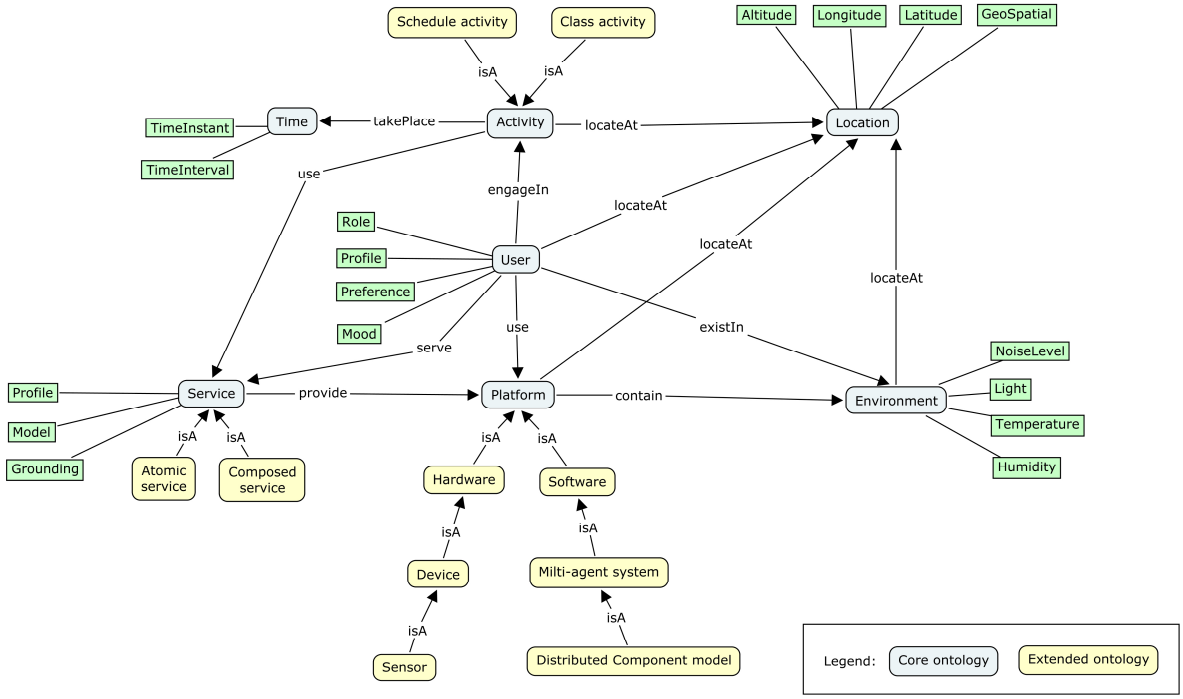

Figura 3: Ontologia Smart space context [Qin et al., 2007]

3.2 DomoML

DomoML [Sommaruga et al., 2011] é um arcabouço baseado em *Web Services* para integração e gerenciamento de diversos recursos em uma residência. Ele utiliza princípios da Web Semântica através de uma ontologia e um protocolo de comunicação para prover interoperabilidade e inteligência para a automação. Os dispositivos heterogêneos, como sensores de diferentes tecnologias são abstraídos através de um driver que traduz os dados capturados para um formato que pode ser interpretado pelo arcabouço através de descrições

RDF/XML baseados na ontologia proposta. A plataforma foi validada em uma residência para auxílio de pessoas idosas. O arcabouço possui também um sistema de gerência que tem por objetivo principal registrar e coordenar os diversos dispositivos da residência, além de acioná-los quando determinada condição é verificada através da análise de regras simples descritas em XML.

DomoML é baseado em três taxonomias: DomoML-env, DomoML-fun e DomoML-com. DomoML-env e DomoML-fun são formalizados como uma ontologia e o DomoML-com consiste em um modelo XML para troca de mensagens.

A ontologia DomoMl-fun descreve as ações e funcionalidades dos recursos, como ligar/desligar ou obter temperatura, porém o autor não descreve a classe em detalhes.

A classe DomoML-env foi elaborada através da análise da taxonomia do *European Home System Protocol* (EHS) [Zahariadis, 2003]. A classificação inclui além dos dispositivos, elementos estáticos como paredes, portas, móveis e também classifica as funções de automação possíveis. As principais classes desta ontologia estão apresentadas na Figura 4 e são descritas a seguir [Sommaruga, 2005]:

- *Building-Equipment:* representa eletrodomésticos, eletrônicos e outros dispositivos possíveis de automatização em uma residência.

- *Component:* inclui os dispositvos simples de automação, como *switchers* válvulas e sensores.

- *Core-Foundation:* agrupa os requisitos técnicos necessários a um componente

- *Building-Environment:* representa a infraestrutura da residência, como cozinha, sala, etc.

- *Location:* Define a localização de cada objeto no ambiente permitindo relações espaciais

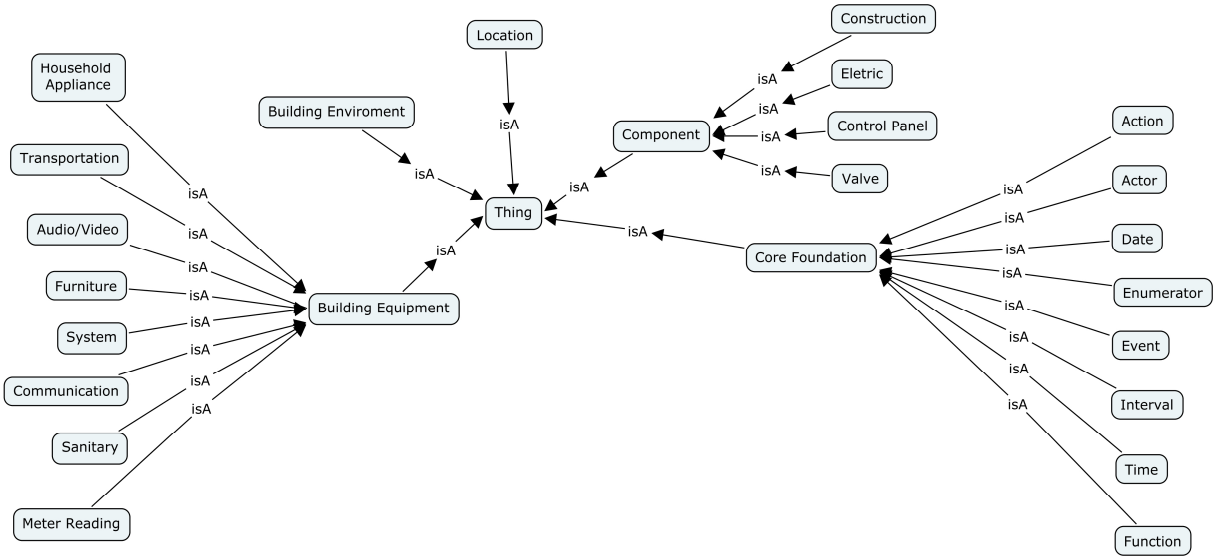

Figura 4: Ontologia DomoML-Env [Sommaruga et al., 2005]

DomoML é uma ontologia modular e abrangente para representação do domínio da residência. Possui classes para representar diversos espaços da residência, além das características físicas como paredes e móveis. As classes de dispositivos baseadas na taxonomia do EHS permitem a representação de um grande número de eletrodomésticos, sensores e atuadores para automação. A ontologia, assim como a proposta da dissertação, tem por objetivo a interoperabilidade entre dispositivos. Porém, as propriedades de relações entre as classes não são apresentadas. A classe Core Foundation também não parece oferecer suporte para análise de contexto do usuário em relação ao ambiente e aos dispositivos.

3.3 DogOnt

Bonino et al. (2008) propõe uma ontologia para o domínio de residências inteligentes com objetivo de prover interoperabilidade entre as diversas soluções de domótica, integrando soluções comerciais com computadores pessoais de baixo custo. A utilização da Web Semântica permite a associação do estado e funcionalidades dos dispositivos através das relações entre as classes da ontologia. A ontologia é utilizada como suporte para diversas soluções do grupo de pesquisa, entre eles um arcabouço orientado a serviços baseado em OSGi (*Open Services Gateway Initiative*).

O conceito principal da ontologia é modelar os dispositivos domóticos suas capacidades e funcionalidades, independente da tecnologia. A abstração da tecnologia permite interligar diferentes dispositivos traduzindo informações de baixo nível em uma linguagem comum e compartilhada. A ontologia também fornece suporte para identificar onde um dispositivo está localizado, estabelecer as capacidades, configurações, e características específicas dos dispositivos e descrever o ambiente e sua composição em relação aos aspectos arquiteturais e outros objetos.

O DogOnt é composto de uma ontologia expressa em linguagem OWL para formalizar os aspectos do ambiente. Através de regras o processo de modelagem é facilitado, gerando automaticamente as propriedades de estados e funcionalidades através de suas relações semânticas. As principais classes da ontologia (Figura 5) são:

- *Building Environment*: descreve o ambiente físico de acordo com suas características, como jardim, sala, etc.

- *Building Thing:* a classe é dividida em outras duas subclasses para descrever objetos controláveis e não controláveis. Os objetos controláveis (classe *Controllable*) correspondem aos dispositivos eletronicamente controlados como sensores, atuadores e demais eletrodomésticos e eletrônicos. Os objetos não controláveis (classe *UnControllable*) correspondem os objetos que fazem parte de um ambiente mas não são controláveis como os móveis.

- *Functionality*: descreve as funções que um dispositivo da classe *Controllable* é capaz de executar. As funcionalidades podem estar relacionadas com controles, estados ou notificações.

- *Command*: indica os comandos necessários para acionar uma função de controle.

- *Notification:* consiste nas notificações enviadas por dispositivos após determinado comando.

- *State e State Values*: representam os estados possíveis de um dispositivo, os valores podem ser discretos ou contínuos, como por exemplo On/Off para discreto e o volume do som da TV com valor contínuo.

- *Domotic Network Component:* esta classe inclui os protocolos de rede utilizados para comunicação por determinado dispositivo. Por exemplo Konnex, ModBus, ZigBee. A modelagem da informação relacionada ao fabricante do dispositivo, as características do protocolo e o modelo de endereçamento são descritos nesta categoria.

A ontologia não inclui classes para representação de uma pessoa, atividades e localização absoluta através de coordenadas. Porém, em comparação as demais ontologias pesquisadas, a DogOnt é a que descreve o maior número de dispositivos, sensores e atuadores para automação bem como suas propriedades e estados. Além disso, ela auxilia na integração através de classes de notificações e funções para cada dispositivo. Diversas classes da ontologia DogOnt foram importadas para a ontologia AssistiveHome.

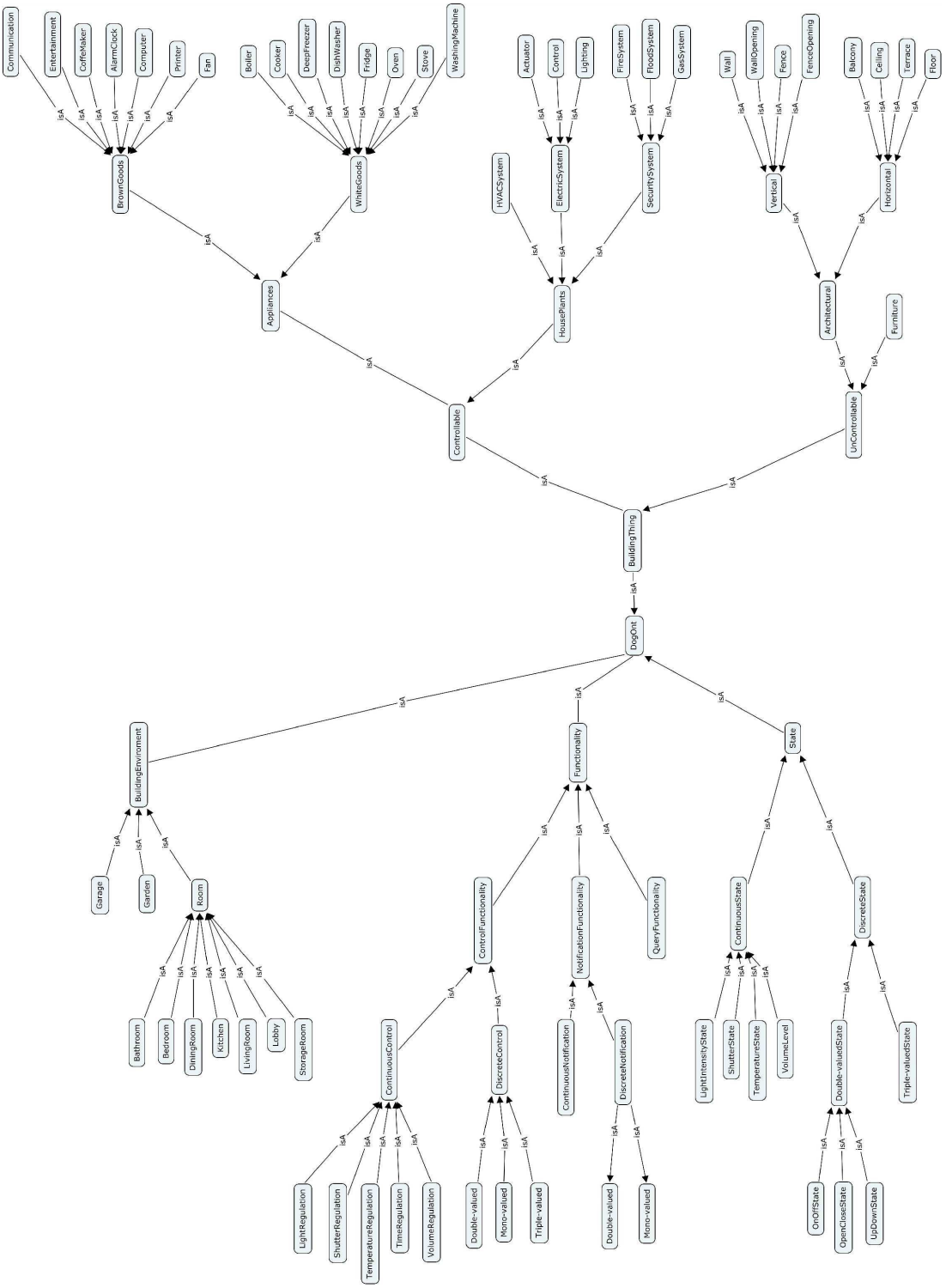

Figura 5: Ontologia DogOnt [Bonino et al., 2008]

3.4 *Context-Based Digital Personality*

O projeto CDBP (*Context-Based Digital Personality*) [Jacquet et al.,2012] propõe um arcabouço baseado em ontologias que permitem expressar as preferências dos usuários com o propósito de personalizar o comportamento dos sistemas. A ontologia também é utilizada para descoberta e interconexão entre dispositivos, base de dados e aplicações. Assim o comportamento do sistema pode ser expresso utilizando regras lógicas. Outra contribuição é um serviço de diagnóstico que monitora o comportamento ao longo da execução do sistema através da descoberta de sensores e o conhecimento sobre o meio físico.

A ontologia é expressa em linguagem OWL e utiliza parte da ontologia DogOnt [Bonino et al., 2008]. A Figura 6 mostra a representação da ontologia e suas principais classes são descritas a seguir:

- *Device*: baseada na ontologia DogOnt, foi simplificada para a nova proposta, porém mantendo as relações entre as classes.

- *Digital Personality*: contém duas subclasses, *Person* e *Digital Personality*. A classe *Person* representa o indivíduo e a classe *Digital Personality* descreve as preferências do usuário para personalização dos serviços.

- *Location*: contém a posição de uma pessoa ou dispositivo em relação ao cômodo da casa.

- *Time*: corresponde a classe *Time* do W3C [W3C, 2006].

- *Diagnosis:* introduz o conceito de efeito físico para inferir sobre a ação esperada de um atuador ou um sensor.

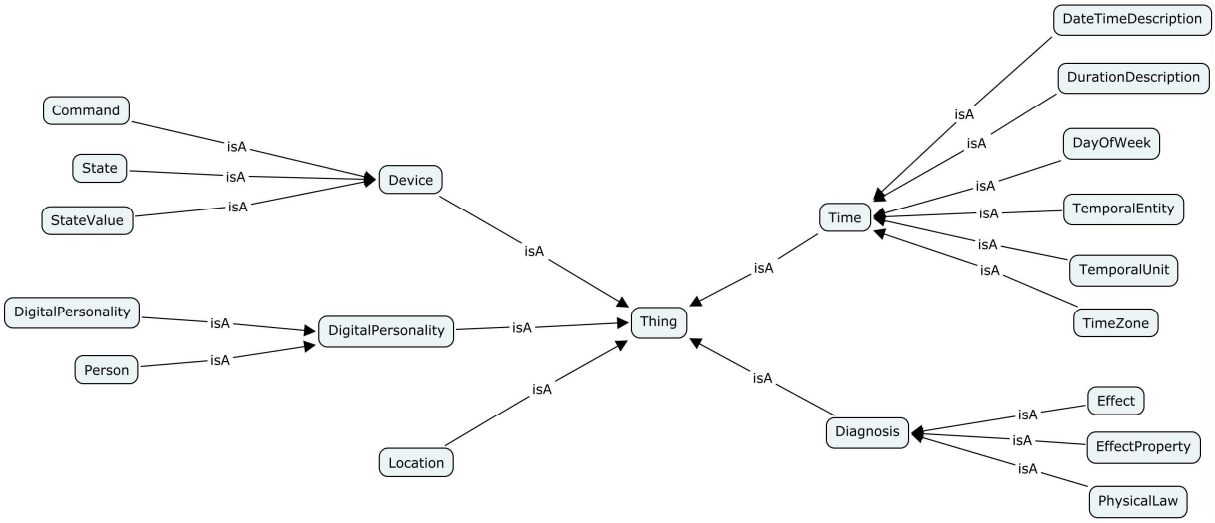

Figura 6: Ontologia CDBP [Jacquet et al.,2012]

O CBDP é composto de *reasoners* que definem o comportamento das aplicações de acordo com as preferências dos usuários representados por regras. Os dados percebidos por sensores são armazenados na ontologia e dependendo das preferências do usuário

representados na classe *Digital Personality* um comando pode ser instanciado. Quando uma nova instância é criada o arcabouço aciona o atuador para que o comando seja de fato executado. Os dispositivos são conectados a uma rede ZigBee e a aplicação utilizada um driver para permitir a troca de dados entre o meio físico e a aplicação.

A ontologia CBDP utiliza a ontologia DogOnt, porém também importa uma classe para representação de pessoa e uma para representação de tempo expandindo a capacidade de sensibilidade ao contexto do modelo anterior. A ontologia não possui modelagem de localização absoluta através de coordenadas, não descreve as funcionalidades da residência e também não apresenta agentes de software.

3.5 *Standard ontology for smart spaces*

A idéia central da ontologia proposta por [Abdulrazak et al., 2010] é a de que um ser vive e interage em um ambiente com uma certa dinâmica. Desta forma, a ontologia é composta de três classes principais: *Being*, *Environment* e *Dynamic* que serão descritas em detalhes abaixo:

- *Environment:* representa o ambiente em que o ser vivo está envolvido. Esta classe inclui conceitos referentes às necessidades do ser como habitação, comida, lazer. O conceito de ambiente está subdividido em duas subclasses *Tangible*, para o que pode ser tocado e *Intangible* que se refere a algo que não possui uma existência física. O conceito de localização também está presente na classe. Ela descreve a localização de um objeto em relação a outros objetos e ao ambiente. Além disso, a classe inclui o referencial de localização absoluto que representa as coordenadas no espaço (*x, y, z)* e a rotação em relação a cada eixo (*yaw, pitch, roll*).

- *Being:* a classe está dividida em duas subclasses principais. *Physical* representa a existência física de um ser que pode ser classificado em *Animal, Plant, Insect* e *Person. NonPhysical* delimita o perfil do ser, permitindo a caracterização do estado em termos de comportamento, preferências, deficiências entre outros. O perfil é baseado no CIF [WHO, 2001] e é composto de *personal profile, health profile* e *organic profile*. *Personal profile* refere-se às características pessoais tais como idade, gênero, estilo de vida, hábitos, entre outros. O *health profile* refere-se ao estado de saúde do indivíduo incluindo doenças, prescrições, alergias e histórico médico. O o*rganic profile* apresenta as funções do corpo e suas estruturas.

- *Dynamic:* descreve as atividades que podem existir em um ambiente. Não está limitada a atividades relacionadas ao Ser, como dormindo, caminhando, limpando, mas também a atividades mecânicas e computacionais. Desta forma a classe divide-se em *Virtual* para atividades computacionais e *Non Virtual* para atividades que podem ser realizadas por um Ser. Para a ontologia, agentes e entidades computacionais podem se comunicar, negociar e tomar decisões para prover os serviços apropriados quando e onde forem necessários.

Um dos diferenciais da ontologia é o conceito de localização relativa entre dois objetos. Outro ponto positivo é a descrição de uma pessoa em relação à saúde e barreiras funcionais. Porém a ontologia não favorece a interoperabilidade, já que possui apenas a classe principal referente aos dispositivos e os softwares, além de representar qualquer ambiente e não apenas

uma residência inteligente. Algumas classes, como *Insect*, não possuem significado para a maioria das aplicações de automação enquanto outras importantes para o contexto, como a representação de tempo, não estão presentes.

3.6 *Ontology-Based Home Service Model*

Wei et al. (2012) propõe um modelo de ontologia para busca e invocação de serviços de acordo com as necessidades do usuário. Primeiramente como referencial teórico o autor classifica o domínio de uma residência em seis classes: *Need, Context, Device, Function, Content, Service*. Através da taxonomia de alto nível é então proposta uma ontologia para funcionalidades e uma ontologia para análise de contexto.

A ontologia de funcionalidade baseia-se na hierarquia de necessidades de Maslow [Maslow, 1943] e as cinco necessidades correspondem às cinco classes principais da ontologia: *Physiological function, Safety function, Belongingness function, Esteem function, Self-actualization function*. Para cada classe foram extraídas as funções da respectiva necessidade, porém o autor não descreve todas as subclasses, somente uma visão geral conforme a Figura 7. Uma função representa a intenção mais básica e é independente do contexto, diferente do conceito de necessidade que é motivada pelo contexto. A função representa a ação que é executada em um dispositivo.

A ontologia de contexto foi elaborada para extrair os conceitos de dispositivos, usuários e o ambiente. As subclasses baseiam-se em outras ontologias presentes na literatura. Na invocação de serviços no ambiente o sistema deve verificar através de regras se existe um serviço que oferece o conteúdo necessário para atender determinada requisição automaticamente.

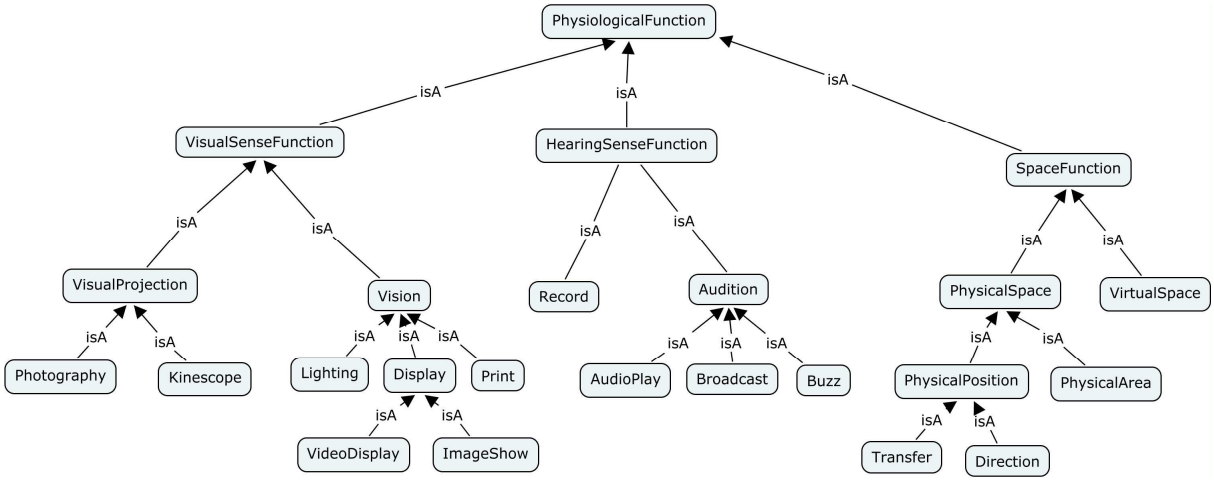

Figura 7: Ontologia para *PhysiologicalFuncion* [Wei et al,. 2012]

A abordagem da ontologia baseada na pirâmide de Maslow não está presente em nenhum dos outros trabalhos analisados e apresenta-se como uma solução bem estruturada para representar as funcionalidades e a relação entre os dispositivos e o usuário. Porém, a hierarquia de classes não é adequada para interoperabilidade entre dispositivos, pois apesar de descrevê-los de acordo com a função, os controles de estados e notificações não existem na ontologia. Também não possui referência para classe de tempo.

3.7 SOPRANO

O projeto SOPRANO (*Service-oriented Programmable Smart Environments for Older Europeans*) [Klein, 2007] tem por objetivo proporcionar independência para idosos no ambiente doméstico através de uma residência inteligente. A proposta utiliza um *middleware* que recebe os comandos do usuário e de sensores e através de regras semânticas aciona a ação apropriada através de atuadores. A ontologia é utilizada como referência para abstração dos sensores e atuadores.

O ponto central da ontologia proposta é o conceito de contexto. A linguagem de representação escolhida é a OWL-Lite. Para modelagem do contexto, o conceito de estado é compartilhado entre diversos componentes como uma propriedade para relacionamento entre as classes. As instancias das propriedades são representadas com dados relativos ao intervalo de tempo. A estrutura geral da ontologia é apresentada na Figura 8. As principais propriedades são: *has-person-state* para todo estado que depende de uma pessoa, *has-environmental-parameter* para toda informação de estado dependente da localização e *has-device-status* para toda informação dependente do dispositivo.

Os sensores enviam mensagens de atualização de estados para o gerente de contexto. O gerente de contexto atualiza a base do conhecimento com a nova informação e, quando possível, atualiza as informações pertinentes para outros estados.

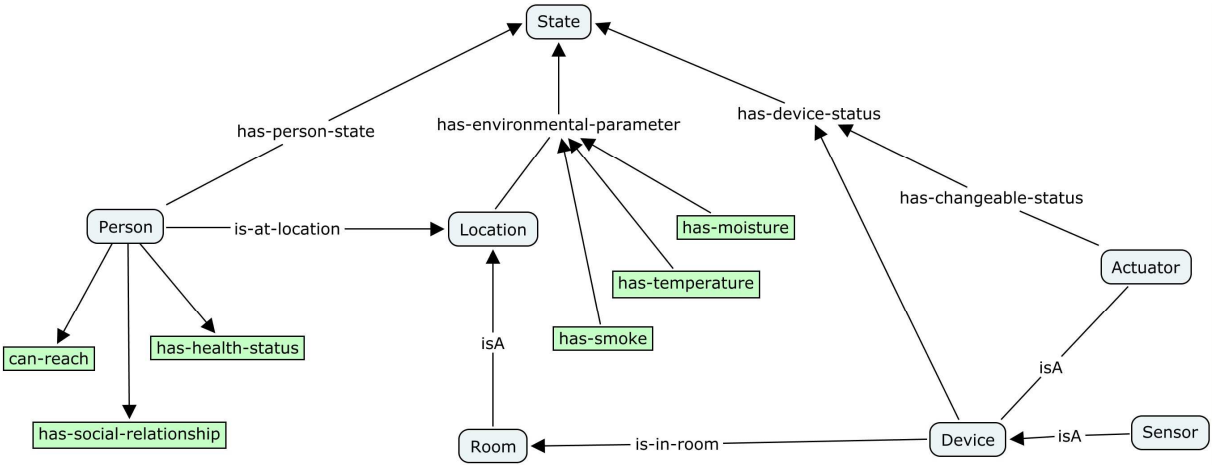

Figura 8: Ontologia SOPRANO [Klein, 2007]

A definição das propriedades entre as classes permite a inferência de contexto entre as diferentes instâncias que compõe o ambiente através da associação entre essas instâncias. Estas propriedades auxiliam na descoberta de serviços e seleção automática de contexto. Apesar da ontologia possuir propriedades entre as classes bem definidas, ela não apresenta uma descrição detalhada dos dispositivos, do ambiente e do indivíduo. Desta forma, não pode ser utilizada como vocabulário para comunicação entre os sistemas de automação.

3.8 BASont

A proposta da ontologia BASont [Ploennigst et al., 2012] é oferecer um modelo para suporte à automação modular, permitindo o reuso. O elemento central da ontologia é a descrição dos dispositivos.

A classe *DeviceInstance*, Figura 9, corresponde aos dispositivos instalados no ambiente. As subclasses da ontologia modelam a lógica de comunicação entre os dispositivos conforme a Figura 8. Os dados e funções são encapsulados em blocos de funções chamados de perfil funcional. Cada instância da *DeviceInstance* pode definir um ou mais *FunctionalProfilelnstances* com diversos *DatapointlnstancesS*. A comunicação entre os dispositivos pode ser definida por *Bindinglnstances* que relacionam duas instancias da classe *Datapointlnstances*. Os dispositivos de fabricantes específicos são então modelados com o DDO (*Device Description Ontology*) que define o vocabulário da ontologia para a especificação de dispositivos de automação. DDO utiliza um modelo hierárquico que separa os conceitos independentes da tecnologia do dispositivo na camada mais alta enquanto os dados específicos de um fabricante localizam-se nos níveis mais baixos. Para cada perfil funcional uma ou mais funções semânticas podem ser instanciadas baseadas na norma ISO 16484-4. O modelo de um dispositivo é representado através da classe *DetailedDesignTemplates* para modelagem das interligações.

A ontologia *Building structure* representa a localização física de um dispositivo. A estrutura da classe foi importada da ontologia proposta por [Beetz et al., 2009].

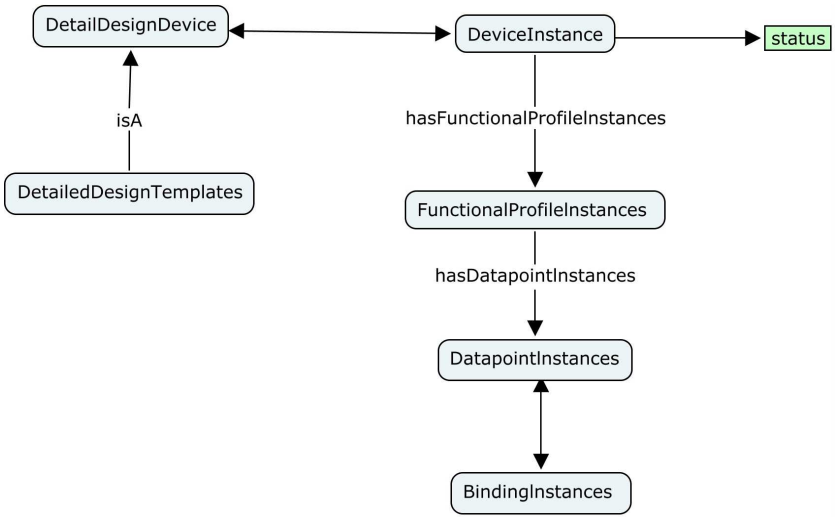

Figura 9: Ontologia para comunicação entre dispositivos [Ploennigst et al., 2012]

A ontologia BASont é utilizada para comunicação entre dispositivos heterogêneos através da especificação das classes e subclasses. O projeto corresponde a uma ontologia para automação predial e é baseada na norma ISO para tecnologia BACnet que não é tão utilizada para automação residencial. BASont representa apenas o domínio dos dispositivos, porém ela aborda o problema de integração e comunicação, mesmo problema da presente dissertação. O modelo completo da ontologia não foi encontrado, apenas é descrita em um artigo, desta forma torna-se difícil comparar a solução em relação aos dispositivos suportados.

3.9 SOUPA

SOUPA [Chen et al,. 2004] é uma ontologia expressa em OWL para o domínio da computação ubíqua e pervasiva. Ela foi projetada em componentes modulares que representam agentes, tempo, espaço, eventos, perfil de usuários, ações, políticas de segurança e privacidade.

SOUPA consiste de duas ontologias distintas: SOUPA *Core* e SOUPA *Extension*. O SOUPA *Core* define conceitos genéricos e universais para qualquer ambiente pervasivo. O SOUPA *Extension* estende a ontologia principal definindo vocabulários adicionais para suporte a aplicações específicas. As relações entre as ontologias são apresentadas na Figura 10.

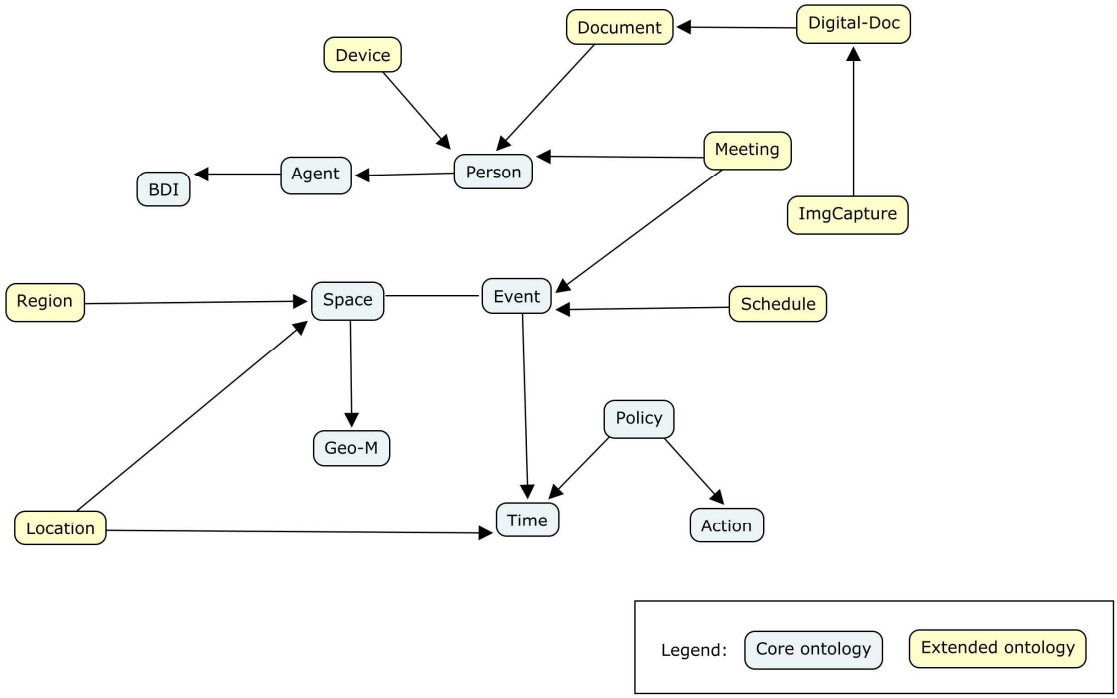

Figura 10: Ontologias SOUPA Core e SOUPA Extension [Chen et al,. 2004]

Classe *Person*: descreve as informações de contato e perfil de uma pessoa. Esta classe é equivalente a classe *Person* da ontologia FOAF [Brickley, 2012]. A classe inclui diversas propriedades como nome, idade, data de nascimento, além de propriedades sociais e profissionais.

Classe *Action*: define o vocabulário para representar segurança e privacidade. O vocabulário é baseado na linguagem Rei [Kagal et al., 2003]. Uma ação (classe *Action*) possui as propriedades *actor, recipient, target, location, time* e *instrument*.

Class *Policy* : uma permissão consiste de regras que permitem ou proíbem a execução de uma ação. A classe *Policy* está associada com as propriedades *permit* ou *forbiden*. A extensão dessas duas propriedades é a classe *PermittedAction* e *ForbiddenAction*. A classe também define o vocabulário para descrever informações sobre permissões individuais. Essas informações incluem o autor de uma permissão (*creator*) a entidade que aplica a regra (*enforcer*), a data de criação da regra (*createdOn*) e o modo padrão de raciocínio *defaultPolicyMode*.

Classe *Agent & BDI*: permitem que agentes compartilhem um entendimento comum de um estado mental, auxiliando na cooperação e colaboração. Entidades computacionais e humanas podem ser modeladas como agentes. A classe *Agent* possui as propriedades *believes, desires e intends* que se relacionam respectivamente com as subclasses *Fact*, *Desire e Intention* da classe BDI. Os objetivos de um agente são expressos através propriedade *hasGoal* que é uma sub-propriedade de *desires*. A subclasse *Desire* define um conjunto de estados que um agente deseja produzir. A classe *Intention* representa os planos que um agente

deseja executar. Planos são definidos em termos de ações, pré-condições e efeitos. A classe *Plan* é definida como uma subclasse de *Action* com propriedades adicionais como *precondition* e *effect*.

Classe Time: ontologia para especificar tempo e relações temporais. Ela adota parte do vocabulário da ontologia DAML-time. Uma representação básica de tempo consiste de *TimeInstant e TimeInterval. InstantThing e IntervalThing* são duas classes disjuntas para um instante ou intervalo de tempo. Para descrever a relação de ordem entre dois eventos no tempo a classe apresenta as propriedades before, after, beforeOrAt, afterOrAt e sameTimeAs.

Classe Space: oferece suporte para raciocínio relativo ao espaço, seja geográfico, regional ou coordenadas geo-espaciais. Parte da ontologia deriva das ontologias OpenCyc e OpenGIS. A classe G*eo-measurement* define longitude, latitude, altitude, distância e superfície. Individuos da classe *SpatialThing* são descritos em relação às coordenadas e as relações são expressas através das propriedades *hasCoordinates* para a classe *LocationCoordinates*.

Classe *Event:* um evento é uma atividade que possui uma relação de tempo e espaço.

A ontologia foi criada para representar contexto em ambientes pervasivos. Cobra é uma ontologia de alto nível que pode ser reutilizada por outras aplicações, porém, não apresenta as classes pertinentes para representar uma residência inteligente.

3.10 Discussão

Cada ontologia representa a visão de um desenvolvedor para o domínio de residência com a finalidade de suprir as necessidades da sua aplicação. Algumas aplicações preocupam-se com a interoperabilidade entre dispositivo, como o DomoML [Sommaruga et al., 2011] e DogOnt [Bonino et al., 2008] e BasOnt [Ploennigst et al., 2012], porém não levam em consideração os demais aspectos do ambiente como pessoa e atividades. Outras se preocupam em representar um modelo generalista, possibilitando extensões futuras, por exemplo SOUPA [Chen et al,. 2005] e Smart Space Context [Qin et al. 2007]. Porém, modelos genéricos não resolvem o problema de interoperabilidade, pois a aplicação precisa saber não somente que existe um dispositivo na residência, mas também qual o tipo deste sensor, quais os valores permitidos na interação e a qual funcionalidade este sensor atende.

Em relação ao usuário, a Standard Ontology [Abdulrazak et al., 2010] e SOPRANO [Klein et al,. 2007] possuem classes detalhadas para representar informações pessoais e questões relacionadas à saúde e funcionalidade do indivíduo, enquanto Wei et al. (2012) classifica a residência de acordo com as necessidades do usuário.

Uma grande dificuldade para a análise das ontologias para o domínio de residências é que nem todas estão disponíveis integralmente, desta forma não é possível analisar o modelo por completo. Muitos artigos apresentam apenas as classes principais e suas relações que podem ser utilizadas por qualquer ambiente inteligente.

Através da análise dos trabalhos correlatos foi possível identificar os conceitos para representação do domínio de uma residência que serão utilizadas como referencial para a criação da nova ontologia, são elas: Dispositivo, Ambiente, Localização, Pessoa, Função, Atividade, Agente de Software e Tempo. A Tabela 1 apresenta as ontologias que foram utilizadas na composição da ontologia AssistiveHome. Cada um dos conceitos resultou em classes, subclasses e propriedades que serão descritos no próximo capítulo. No capítulo de

Resultados será apresentado também um comparativo entre as ontologias do estado da arte e a ontologia proposta nesta dissertação.

Tabela 1: Conceitos reutilizados na ontologia AssistiveHome [O autor]

Conceito	**Conceitos reutilizados**
Dispositivos	DogOnt [Bonino et al., 2008]: as classes de dispositivos, funções e estados além das propriedades que relacionam estas classes foram importadas.
Ambientes	DogOnt [Bonino et al., 2008]: a maioria dos termos da classe de Ambientes da ontologia AssitiveHome foi baseada na classe BuildingEnviroment de DogOnt. DomoML [Sommaruga et al., 2011]: esta ontologia também possui uma classe BuildingEnviroment e desta forma esse foi o termo mantido em AssistiveHome. Cobra-Ont [Chen, 2004]: esta ontologia possui os conceitos de ambiente interno e externo não apresentado nas ontologias interiores e que foi adicionado ao novo modelo.
Localização	Standard ontology [Abdulrazak et al., 2010]: os conceitos de localização absoluta em um ambiente e as propriedades para a localização relativa entre dois objetos foram utilizados. GeoSpatial do W3C [W3C, 2013]: possui o mesmo conceito de relação absoluta no espaço que foi utilizado como as propriedades longitude, latitude e altura.
Pessoas	Standard ontology [Abdulrazak et al., 2010]: o modelo baseado no perfil da pessoa foi utilizado principalmente em relação a propriedades de dados, como altura e nome. Além disso. o perfil em relação as barreiras funcionais também foi considerado no reuso.
Funcionalidades	Home Service Model [Wei et al,. 2012]: as funcionalidades de domótica foram baseados no conceitos de necessidades dessa ontologia.
Atividades	Nenhuma das ontologias atendeu o modelo e não foram importados novos conceitos.
Agentes de Software	SOUPA [Chen et al,. 2005]: apenas a existência da classe principal e as relações foi considerada já que o conceito de agente nessa ontologia refere-se diretamente ao modelo BDI.
Tempo	OWL-Time [W3C, 2006]: esta ontologia foi importada por completo para representação de tempo.

Capítulo 4

Ontologia Proposta

Neste capítulo são descritas as etapas de desenvolvimento da ontologia AssistiveHome. Existem diversas metodologias propostas para o desenvolvimento de ontologias, nesta dissertação foi utilizada a metodologia descrita por Noy et al. (2001) denominada *Ontology Development 101*. A construção da ontologia é apresentada em detalhes a seguir. A lista completa de classes e propriedades encontra-se em formato RDF/OWL e pode ser importado para um visualizador como, por exemplo, o Protege.

4.1 Domínio e escopo da ontologia

Antes de iniciar a criação da ontologia é preciso determinar o domínio, a finalidade e o público-alvo que irá manter a ontologia. O domínio é a representação de uma residência inteligente. A finalidade da ontologia é permitir a interoperabilidade entre os dispositivos e aplicações através de suporte semântico para a comunicação entre os agentes. A AssistiveHome representa o contexto do usuário, ambiente, dispositivos e aplicações.

Em relação ao público alvo, a ontologia poderá ser utilizada por aplicações de domótica em geral ou voltadas para automação assistiva.

Noy et al. (2001) sugere a criação de questões de competência, que são perguntas que a ontologia deve responder com objetivo de avaliar se a estrutura atende as expectativas. Para a ontologia AssistiveHome as questões de competência foram efetuadas baseadas na classificação para a modelagem de contexto de Abowd et al. (2000) composta de cinco questões: Quem, O que, Onde, Quando e Por que.

O contexto de “Quem” refere-se a uma pessoa, agente de software ou dispositivo em uma residência. O “Onde” descreve a localização do quem através de sua posição absoluta ou relativa. O conceito de “O que” está relacionada à atividade tanto de uma pessoa quanto de um dispositivo. O quando se refere ao tempo em relação a uma atividade ou ação. Já o “Por que” compreende a motivação para efetuar determinada ação e não é parte da ontologia, mas sim parte da aplicação presente na residência que seleciona a melhor ação identificando o motivo de determinado contexto. Desta forma foram criadas as seguintes questões de competência que a ontologia deve responder através da troca de mensagens no arcabouço proposto:

1. Quais agentes (Quem) controlam um determinado tipo de dispositivos (Quem) em um ambiente (Onde)?
2. Qual a atividade do usuário (O que) em determinado instante (Quando)?
3. Quais as ações (O que) possíveis de um dispositivo (Quem)?

4.2 Classes e hierarquias

Através da avaliação das ontologias existentes, bem como da análise do domínio e escopo foram enumerados os termos considerados importantes. Conforme apresentado no capítulo de trabalhos correlatos, existem diversas ontologias que descrevem uma residência ou, de forma mais generalista, descreveram funcionalidades em ambientes pervasivos. Conforme sugerido na metodologia *Ontology Development 101* [Noy et al.,2001], diversos termos existentes em outras ontologias foram utilizados na ontologia AssistiveHome.

Para cada uma das classes foi considerado o reuso de termos de outras ontologias, de forma a reutilizar o conceito, hierarquia e relações. Algumas ontologias possuem o mesmo conceito apenas com palavras diferentes, como a ontologia será utilizada para comunicação entre agentes através de um modelo compartilhado, não é necessária a inclusão de sinônimos e somente um dos termos foi adotado para a criação das classes, propriedades de objetos e de dados. O nome das classes e suas propriedades foram mantidos em inglês para facilitar o mapeamento dos termos em relação às demais ontologias. As figuras de representação da ontologia deste capítulo baseiam-se em mapas conceituais, os conceitos correspondem às classes e as relações correspondem as propriedades de objetos. A Figura 11 apresenta uma visão geral da ontologia AssitiveHome. A seguir será descrito o processo de criação das classes e a justificativa para o modelo.

4.2.1 Classes de Localização

Existem diversas formas de contextualizar a posição de um objeto no ambiente. As classes de localização devem permitir que agentes informem a posição de uma pessoa ou dispositivo em relação ao ambiente, espaço ou a outro objeto, incluindo dispositivos e pessoas. Bettini (2010) apresenta dois tipos de localização presentes na maioria dos projetos para modelagem de contexto:

- Coordenadas geométricas: representam pontos ou áreas no espaço métrico, como as coordenadas WGS84 [W3C, 2006] para latitude, longitude e elevação. Através da sobreposição das figuras geométricas é possível efetuar diversas consultas sobre a relação espacial entre dois objetos.

- Coordenadas Simbólicas: referem-se à posição relativa de um objeto representada por um identificador, por exemplo, os cômodos da residência. Este conceito pode ser expandido também para indicar proximidade ou outras relações qualitativas, como em frente, atrás, acima, ao lado.

Desta forma, para a ontologia AssistiveHome foram descritas duas classes principais referentes a localização: *BuildingEnviroment* e *GeoLocation.* Além das classes também foi inserida a propriedade de objeto *ReferencialLocation, composta de diversas* sub-propriedades para representar as relações qualitativas. Estes itens serão descritos em detalhes a seguir.

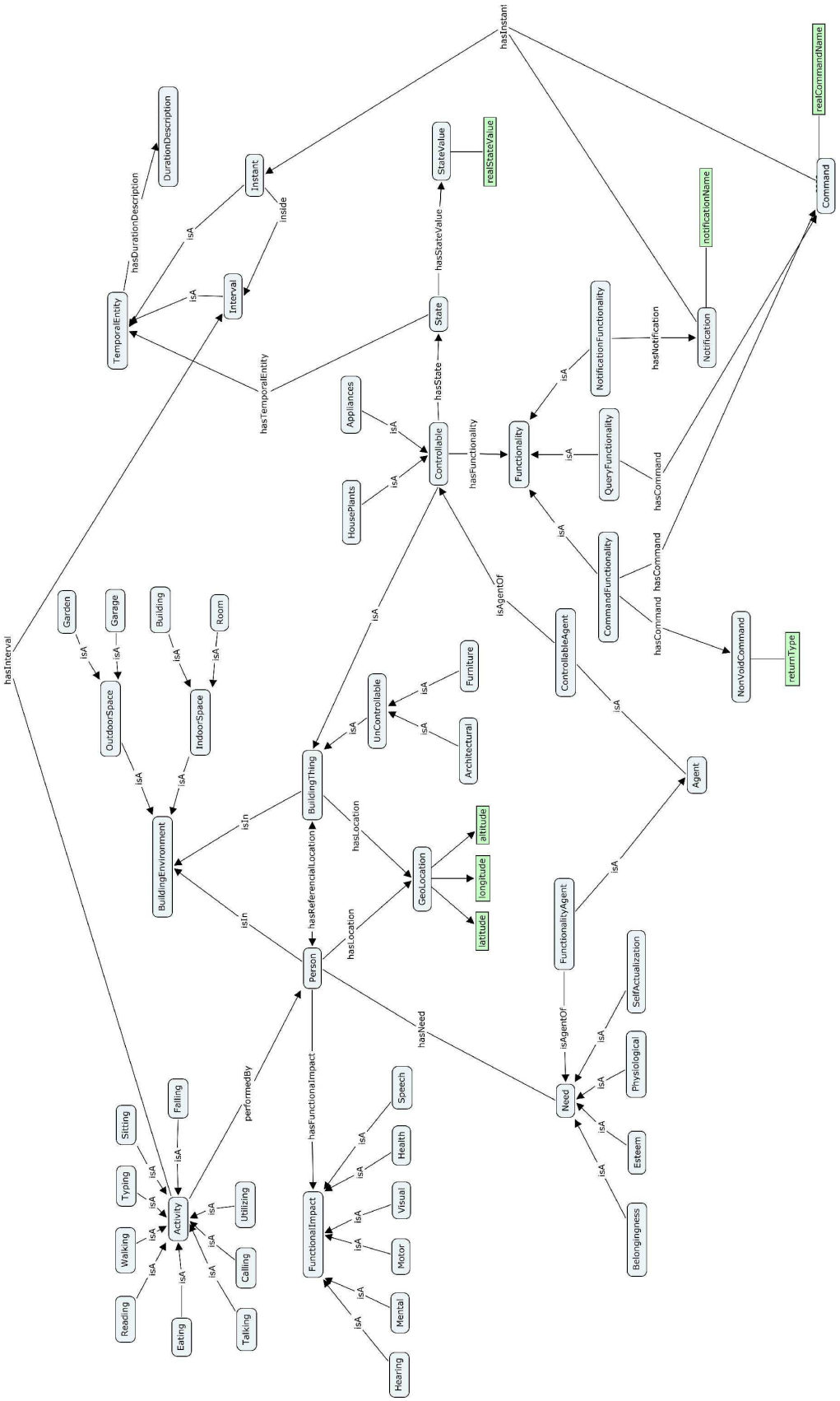

Figura 11 – Ontologia AssistiveHome [Fonte: O Autor]

Classe BuildingEnviroment

A classe *BuildingEnviroment* representa os espaços delimitados de uma residência. Grande parte das ontologias analisadas possui classes para representar o ambiente, por exemplo, DomoML-env [Sommaruga, 2005] e DogOnt [Bonino, 2008] possuem a classe *BuildingEnvironment*, Cobra-Ont [Chen, 2004] possui a classe *Place*. As classes possuem subclasses para classificar o ambiente. Cobra-Ont [Chen, 2004] possui as subclases *AtomicPlaceInBuilding* e *AtomicPlaceNotInBuilding* para identificar se um local está localizado internamente ou externamente à uma residência, o mesmo ocorre com a ontologia CONON [Wang et al., 2004] através das subclasses *OutdoorSpace* e *IndoorSpace*.

Na ontologia AssistiveHome, conforme Figura 12, optou-se por manter o nome da classe *BuildingEnvironment*, adicionando duas subclasses disjuntas: *OutdoorSpace e IndoorSpace.* A classe *OutdoorSpace* inclui o jardim (*Garden*) e a garagem (*Garage*). A classe *IndoorSpace* divide o ambiente interno entre comôdos (*Room)* e contrução (*Building*). A classe *Room* classifica o tipo de cômodo através de subclasses como, por exemplo, *Kitchen*, *Bathroom* e *DiningRoom*. A propriedade de objeto *isIn* permite identificar a posição de uma instância de objeto da classe *BuildingThing* ou pessoa da classe *Person* em uma instância da Classe *BuildingEnvironment*. Ou seja, a propriedade *isIn* permite expressar que a pessoa chamada Leticia está no quarto ou que o dispositivo lâmpada está na cozinha.

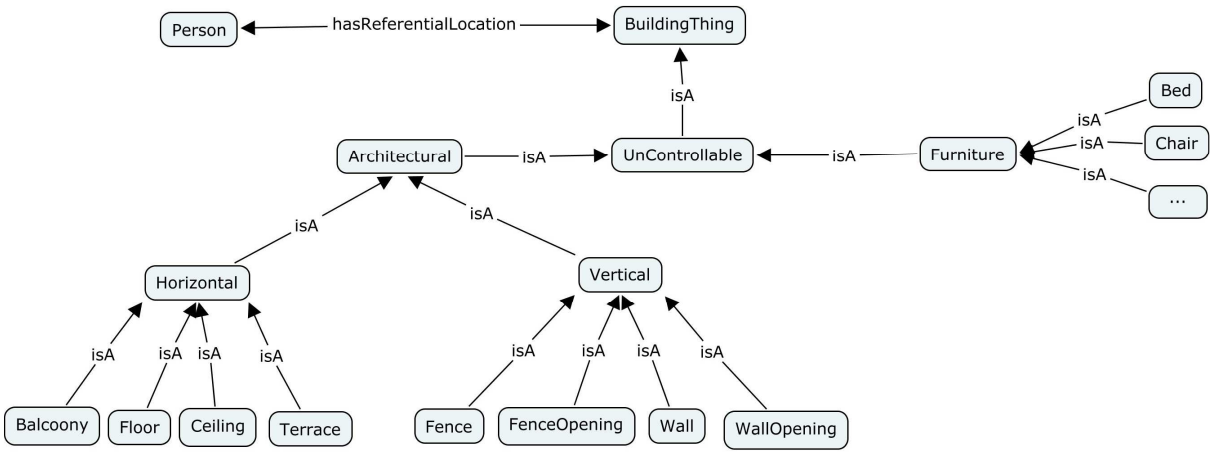

Figura 12 - Classe *BuildingEnvironment* [Fonte: O Autor]

Classe *GeoLocation*

A classe Geo*Location*, Figura 13, permite estabelecer a posição referencial de um objeto ou pessoa em relação ao ambiente. Baseado em WGS84 [W3C, 2006] ela representa as coordenadas longitude, latitude e altitude em relação ao nível do mar através das propriedades de dados com o mesmo nome. A propriedade de objeto *haslocation* relaciona uma instância da classe *GeoLocation* com individuos da classe *Person* ou *BuildingThing*, desta forma é possível representar a localização espacial de uma pessoa ou dispositivo.

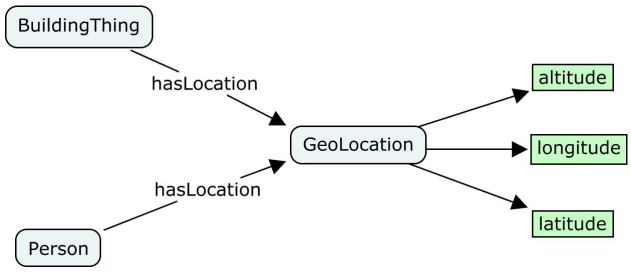

Figura 13 - Classe *GeoLocation* [Fonte: O Autor]

Propriedade de objeto ReferencialLocation

A propriedade de objeto *hasReferencialLocation* permite estabelecer relações entre dois objetos, este tipo de propriedade pode ser utilizado, por exemplo, para identificar qual dispositivo deve ser acionado de acordo com a posição do usuário. Abdulrazak et al. (2010) propõe as relações *upTo* (acima), *downTo* (abaixo), *behindTo* (atrás), *inFrontTo* (a frente), *nearTo* (próximo), *inTo* (dentro) e as proprieddes de dados *Xpositive* (x positivo) e *Xnegative* (x negativo). Estas propriedades foram incluídas na ontologia AssistiveHome, exceto *Xpositive* e *Xnegative* que representavam o ângulo entre dois objetos e foram substituídos por referenciais cardinais *North* (Norte), *South* (Sul), *East* (Leste), *West* (Oeste), *SouthEast* (Sudeste), *SouthWest* (Sudoeste), *NorthEast* (Nordeste), *NorthWest* (Noroeste) como propriedades de objetos. Além disso, também foi adicionado o conceito *farFrom* como referencia oposta à *nearTo*. O domínio (*domain*) e o escopo (range) para *hasReferencialLocation* e sub-propriedades corresponde as classes *BuildingThing* e *Person.*

4.2.2 Classes de Dispositivos

Os dispositivos representam o ponto central da integração em residências inteligentes e fazem parte da ontologia AssitiveHome para permitir a automação. Apesar da necessidade de utilização de um *driver* específico para comunicação com o hardware, a comunicação no nível da aplicação também precisa da informação da função, do estado e dos dados de cada dispositivo. Desta forma, o sistema responsável pelo controle do dispositivo pode ler a requisição e então transformá-la em sinais compreensíveis no nível do hardware.

Buscou-se através das classes de controle desenvolver um modelo compatível com os principais protocolos de automação: ZigBee, CEBus, X-10, UPnP e KNX. Apesar de cada padrão utilizar diferentes modelos de comunicação, os comandos e funcionalidades são comuns, pois não dependem do protocolo e sim do dispositivo.

A construção das classes de dispositivo considerou o reuso de outras ontologias e também da tabela de dispositivos apresentadas por Bolzani (2004) e Chan (2008). A ontologia AssistiveHome importa as classes e subclasses de *BuildingThing*, *State, Functionality* e *Command* da ontologia DogOnt [Bonino et al., 2008]. A escolha deve-se primeiramente ao grande número de dispositivos descritos, além de atender ao objetivo de compatibilidade com diferentes protocolos de automação.

A classe denominada BuildingThing inclui todos os objetos de uma residência dividindo-os em controláveis (Classe *Controllable*) e não controláveis (Classe *UnControllable*). Os objetos eletronicamente controláveis representados através da classe

Controllable possuem funcionalidades descritas através da classe *Functionality*. Por exemplo, a função *OnOffFunctionality* descreve a função de ligar e desligar uma lâmpada. As funcionalidades relacionam-se com comandos ou notificações. A classe *Command* indica os comandos necessários para acionar uma função de controle, no caso, a classe *OnOffFunctionality* relaciona-se com os comandos *OnCommand* e *OffCommand*. Da mesma forma, uma funcionalidade pode representar uma função de notificação relacionando-se com a classe *Notification*, por exemplo, a funcionalidade *OnOffNotifiation* possui as notificações *OnNotification* e *OffNotification.* As classes *State* e *StateValue* representam os estados de um dispositivo através de valores discretos ou contínuos. A seguir encontra-se uma descrição das classes referentes a dispositivos.

Classe Un*Controllable*

A classe *UnControllable* descreve os componentes não controláveis de uma residência e divide-se em elementos de arquitetura e móveis das classes *Architectural* e *Furniture* respectivamente. Conforme a Figura 14, a classe *Architectural* classifica os objetos em horizontal e vertical (classe *Horizontal* e classe *Vertical*). Elementos horizontais são classificados em Varanda, Piso, Teto e Terraço (*Balcony*, *Floor*, *Ceilling* e *Terrace*). Elementos verticais classificam-se em Cerca, Abertura na cerca, Parede e Abertura na Parede (*Fence*, *FenceOppning*, *Wall* e *WallOpening*), permitindo contextualizar o ambiente. Furniture possui 14 subclasses para representar os móveis de uma casa, *Bed* e *Chair* são exemplos e podem ser utilizados para identificar a atividade de um usuário de acordo com a posição através da propriedade de objeto e sub-propriedades de *hasReferentialLocation*.

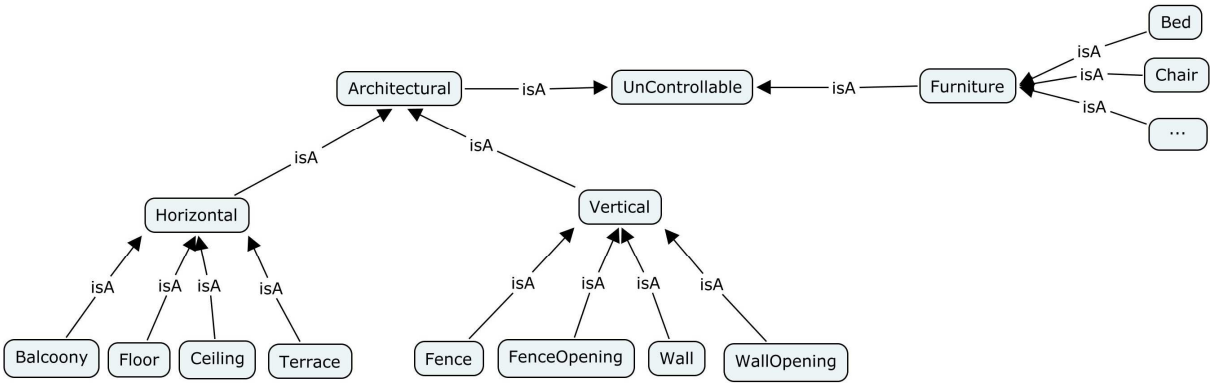

Figura 14: Classe *UnControllable* para representação de elementos estáticos [Fonte: O Autor]

Classe *Controllable*

A classe *Controllable* compreende todos os objetos controláveis da residência e são classificados em *Appliances e HousePlants.* A classe *Appliances* abrange os eletrodomésticos e eletrônicos que são divididos em *BrownGoods e WhiteGoods* para linha branca e marrom conforme o tamanho e função do equipamento. Por exemplo, relógio e computador são considerados parte da linha marrom e desta forma correspondem a subclasses de *BrownGoods.* Já geladeira e fogão classificam-se como *WhiteGoods.*

A classe *HousePlants* descreve os dispositivos de automação e possui as subclasses *HVAC systems* para aquecimento, ventilação e ar condicionado; *SecuritySystems* para sistema de segurança e *ElectricSystems* para os demais dispositivos conectados a rede elétrica. Cada

uma das subclasses possui inúmeros dispositivos. A hierarquia das classes principais está representada na Figura 15.

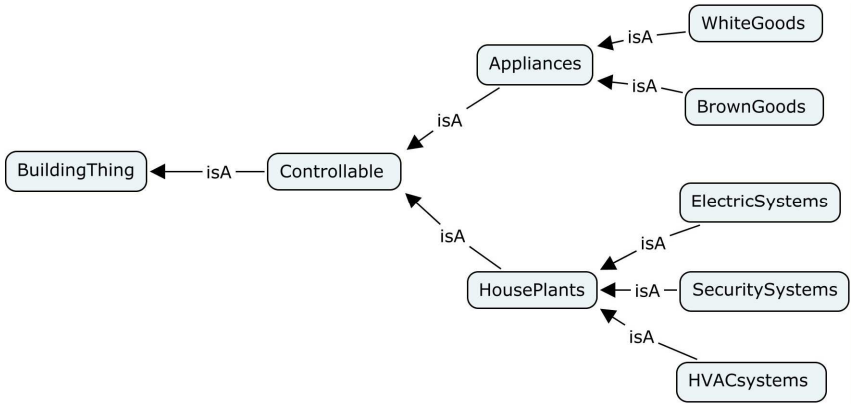

Figura 15: Classe *Controllable* para representação de dispositivos [Fonte: O Autor]

Classe *Functionality*

A classe *Functionality* descreve as funções de automação de acordo com os dispositivos da classe *Controllable* e cada funcionalidade define os comandos possíveis (classe *Command*) para modificar um dispositivo.

As funcionalidades são divididas em diferentes classes de acordo com seus objetivos (Figura 16). A classe *ControlFunctionalities* modela a habilidade de controlar um dispositivo ou parte dele. A classe *NotificationFunctionalities* representam a habilidade de um dispositivo de informar seu estado interno ou sinalizar a mudança de um estado. A classe *QueryFunctionalities* compreende a capacidade de um dispositivo de ser consultado sobre sua condição como falhas ou estados.

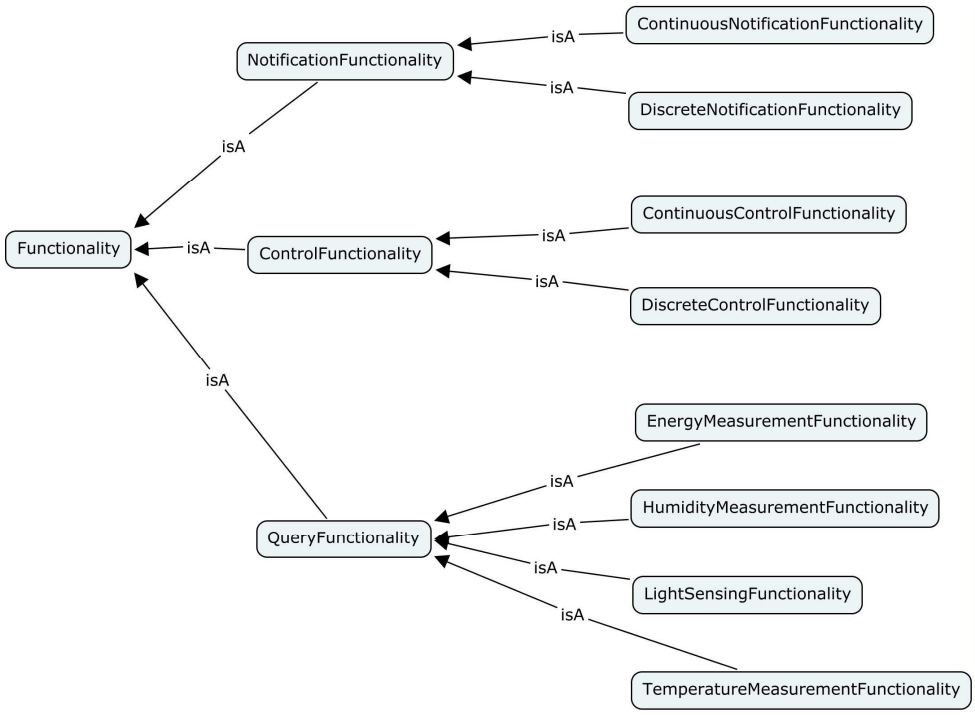

Figura 16 - Classe Functionality [Fonte: O Autor]

As classes acima também são modeladas de acordo com a forma de interação e modificação do estado que pode ser contínuo quando permitem modificar as características de forma contínua entre um valor mínimo e máximo e funções discretas que permitem somente mudanças de valores pré determinados, como *on* e *off*.

Classe *Command*

A classe *Command*, Figura 17, apresenta os possíveis comandos que uma funcionalidade de controle ou *query* pode executar. Cada subclasse possui um tipo de propriedade de dados diferente dependendo do valor a ser representado que pode ser do tipo *string* ou *double* por exemplo. A classe divide-se em duas subclasses: *VoidCommand* e *NonVoidCommand*. A classe *VoidCommand* corresponde as funções que tem apenas uma entrada e a classe *NonVoidCommand* é utilizada para comandos que retornam uma resposta.

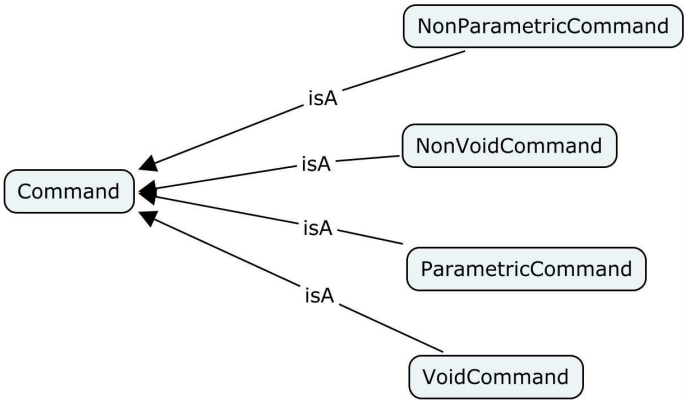

Figura 17 - Classe Command [Fonte: O Autor]

Classe *State e StateValue*

A classe *State, Figura 18*, classifica os possíveis estados de um dispositivo da classe *Controllable.* Estados são classificados em contínuos ou discretos de acordo com o tipo de valor que podem assumir. Os valores estão representados na classe StateValue e os estados da classe State possuem um ou mais estados referentes a classe StateValue através da propriedade de dados hasStateValue. As subclasses da classe *ContinuousValue* possuem uma propriedade de dados do tipo *float* enquanto as subclasses de *DiscreteValue* possuem um conjunto de duas ou três propriedades de dados de acordo com os valores possíveis. Por exemplo, uma lâmpada da classe *Controllable* possui o estado *OnOffState* que é um estado com dois possíveis valores representados na classe *StateValue* (classes *OnStateValue* e *OffStateValue).* Os valores reais dos estados são representados por uma propriedade de dados do tipo *string* "*On*" ou "*Off*".

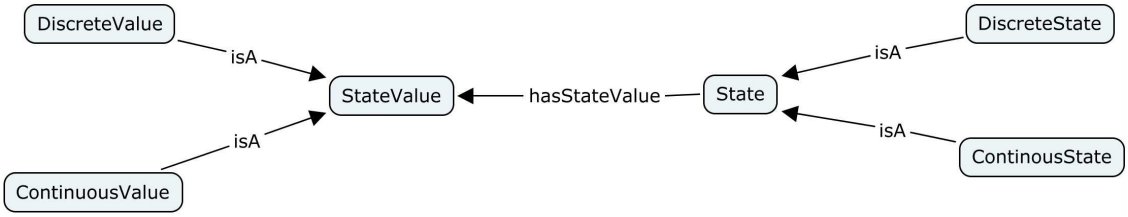

Figura 18 - Classe State [Fonte: O Autor]

Propriedades para classes de dispositivos

As classes para automação de dispositivos relacionam-se através de propriedades de dados conforme a Figura 19. A propriedade *hasFunctionality* informa que um determinado dispositivo possui uma funcionalidade. Já a propriedade *hasState* associa a classe Controllable com os possíveis estados do dispositivo da classe *State*. A classe *State*, então, relaciona-se com uma instância da classe *StateValue* para descrever o valor de um determinado estado através da propriedade de objeto *realStateValue*.

As funcionalidades dividem-se em comandos, notificações e requisições. A propriedade *hasCommand* delimita os possíveis comandos de instâncias da classe *CommandFunctionality* ou *QueryFunctionality e* a propriedade *returnType* descreve o valor de retorno para um comando não nulo. A propriedade *hasNotification* relaciona uma funcionalidade de notificação da classe *NotificationFunctionality* com as possíveis notificações para a função. As propriedade de dados *realCommandName* permite especificar em uma instância o valor real do comando para o padrão de automação ao qual o dispositivo pertence. A propriedade *notificationName* especifica o nome da notificação também referente ao protocolo de automação.

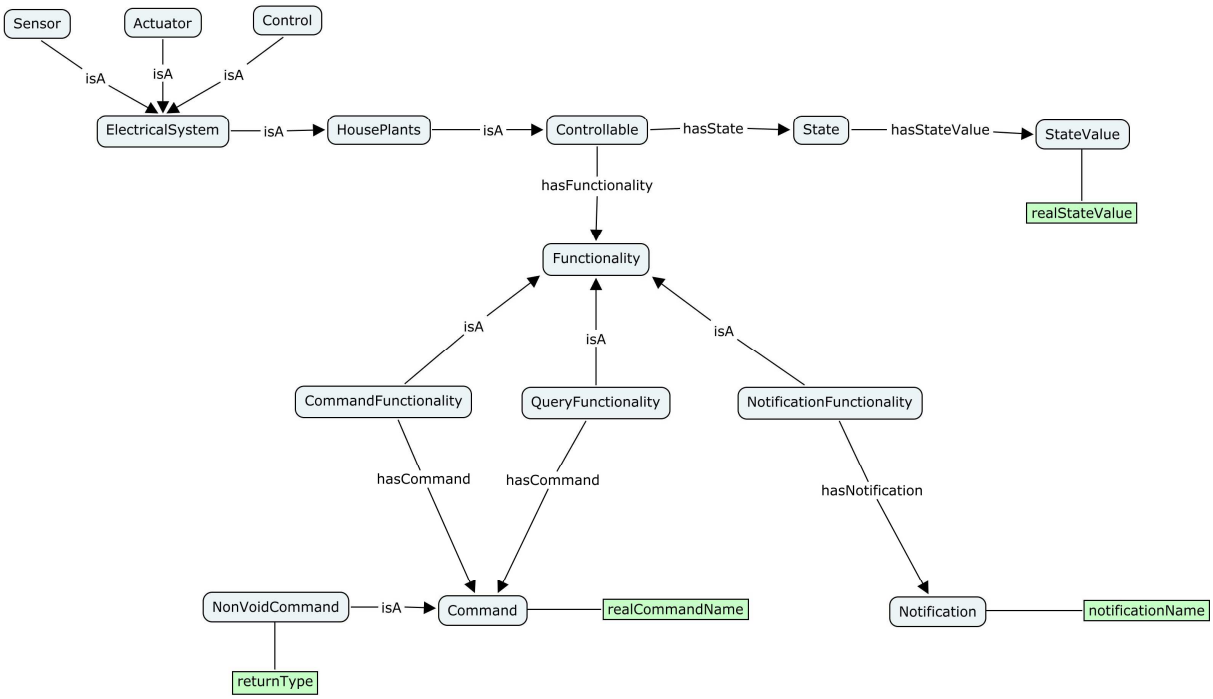

Figura 19 – Propriedades para classes de dispositivos[Fonte: O Autor]

4.2.3 Classe para *Pessoa*

Uma residência também deve levar em consideração os seus moradores, cada pessoa possui características próprias que podem influenciar no resultado da interação. Além das características do indivíduo, as atividades executadas em determinado momento também influenciam no contexto da automação. Por último, um usuário possui preferências ou necessidades que uma aplicação deve suprir. Assim, foram criadas quatro classes referentes à pessoa: *Person, Activity, Need e FunctionalImpact.*.

Classe Person

A classe *Person* (Figura 20) foi baseada parcialmente no conceito proposto por Abdulrazak et al. (2010) que utiliza o modelo do CIF [WHO, 2001] e é composto de *personal profile, health profile* e *organic profile*. As características descritas como *Personal profile* foram criadas como propriedades de dados para a nova ontologi e são: peso (*weight*), altura (*height*), idade (*age*), data de nascimento (*birthDate*), nome (*name*), email (*email*), sexo (*gender*) e número de telefone (*phoneNumber*). Estas propriedades permitem que a automação seja relevante para o perfil do usuário ou até mesmo permite identificar através do peso e altura qual a pessoa presente em um ambiente. Adicionalmente a classe *Person* criou-se subclasses para definir se um morador é Idoso (classe *Senior*), Adulto (classe *Adult*), Criança (classe *Child*) e Bebê (classe *Baby*).

O perfil de funcionalidade originou uma nova classe composta das subclasses *Hearing*, *Mental*, *Motor*, *Visual*, *Health* e *Speech* de acordo com a barreira funcional do indivíduo. O perfil pode ser utilizadas para escolha do dispositivo ou da melhor interface de comunicação com o usuário.

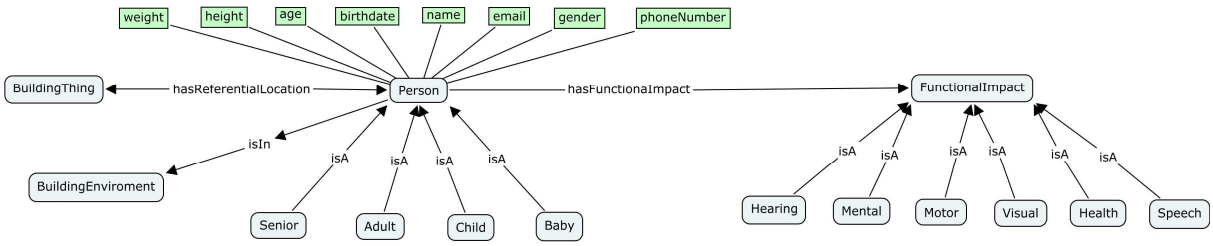

Figura 20 - Classe *Person*, *FunctionalImpact* e propriedades [Fonte: O Autor]

Classe Need

A ontologia Home Service Model [Wei et al,. 2012] descreve as funcionalidades de uma residência em relação ao indivíduo e as necessidades da pirâmide de Maslow [Maslow, 1943] através das subclasses *Belongingness, Esteem, Physiological* e *SelfActualization* . Para a ontologia da dissertação, uma necessidade pode ser utilizada para compor as funcionalidades que o ambiente inteligente possui. Por exemplo, uma aplicação que atenda o a necessidade de segurança, pode controlar a abertura de portas e acionamento de alarme.

A Figura 21 apresenta a classe *Need* e as propriedades que relacionam a classe com o domínio das classes *Person* e *FunctionalityAgent*. Ou seja, um agente de software representado pela classe *FunctionalityAgent* pode relacionar-se com uma necessidade através da propriedade *isAgentOf* enquando uma pessoa da classe *Person* também possui necessidades através da proprieade *hasNeed*.

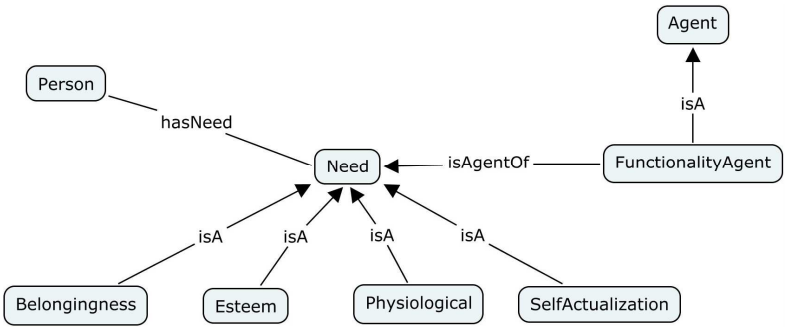

Figura 21 - Classe *Need* e propriedades [Fonte: O Autor]

Classe Activity

A classe *Activity*, representada na Figura 22, descreve as atividades do usuário em determinado momento. O conceito de atividade é importante para determinar o contexto da aplicação, por exemplo, se o habitante da residência está dormindo as luzes podem estar apagadas. Foram listadas diversas atividades: *Reading, Walking, Typing, Sitting, Eating, Talking, Calling, Utilizing* e *Falling* que estão presentes na Figura 21. A propriedade de objeto *performedBy* indica a atividade de um usuário.

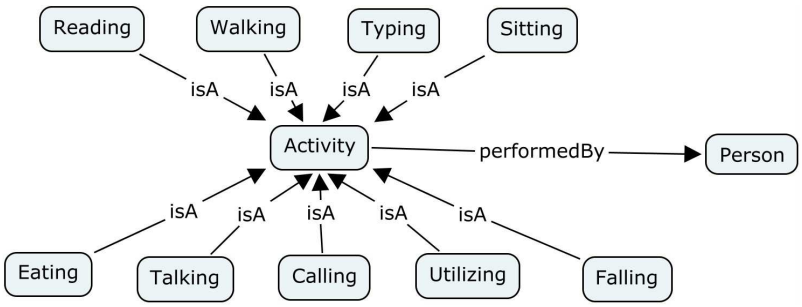

Figura 22: Classes relativas à pessoa [Fonte: O Autor]

4.2.4 Classe *Time*

A ontologia Time importa a ontologia OWL-Time do W3C [W3C, 2006], não foi feita nenhuma alteração ao modelo. Ela representa um intervalo de tempo ou um instante, define as propriedades de duração e a unidade de medida temporal entre outras relações que estão descritas na Figura 23.

As relações temporais na ontologia OWL-Time são representadas por uma classe abstrata *TemporalEntity* que pode ser classificada como um intervalo de tempo (classe *Interval*) ou um instante de tempo(classe *Instant*). As propriedades hasBeggining e hasEnd relacionam uma instância de *Instant* com uma entidade temporal, pode-se dizer que o início ou fim de um instante corresponde ao valor deste mesmo instante.

A propriedade *inside* relaciona um instante (classe *Instant*) a um intervalo (classe *Interval*), enquanto a propriedade *before* é utilizada para indicar que uma entidade temporal ocorre antes de outra entidade temporal. Existem diversas outras propriedades para relacionar

duas instâncias de *TemporalEntity: intervalEquals, intervalBefore, intervalMeets, intervalOverlaps, intervalStarts, intervalDuring, intervalFinishes*.

A duração de um intervalo pode ter diversas descrições, por exemplo, 1 dia e 2 horas ou 26 horas e este tipo de descrição é feita classe *DurationDescription* que possui as propriedades de dados para representar de segundos até anos. Uma instância da classe *DurationDescription* relaciona-se com uma instância de tempo através da propriedade *hasDurationDescription*. Um intervalo pode ter diversas descrições, mas todas com a mesma duração.

A classe de *Interval* relaciona-se com a classe de atividades (*Activity*) de uma pessoa através da propriedade *hasTemporalEntity* quando deseja comunicar a duração de uma atividade. Da mesma forma comandos e notificações de dispositivos relacionam-se com a classe *Instant* para determinar o instante em que ocorreram pela propriedade *hasInstant.* Por último o estado de um dispositivo pode ter a intenção de informar tanto um instante quanto um intervalo, nesse caso a classe State relaciona-se com a classe *TemporalEntity* através da propriedade *hasTemporalEntity*.

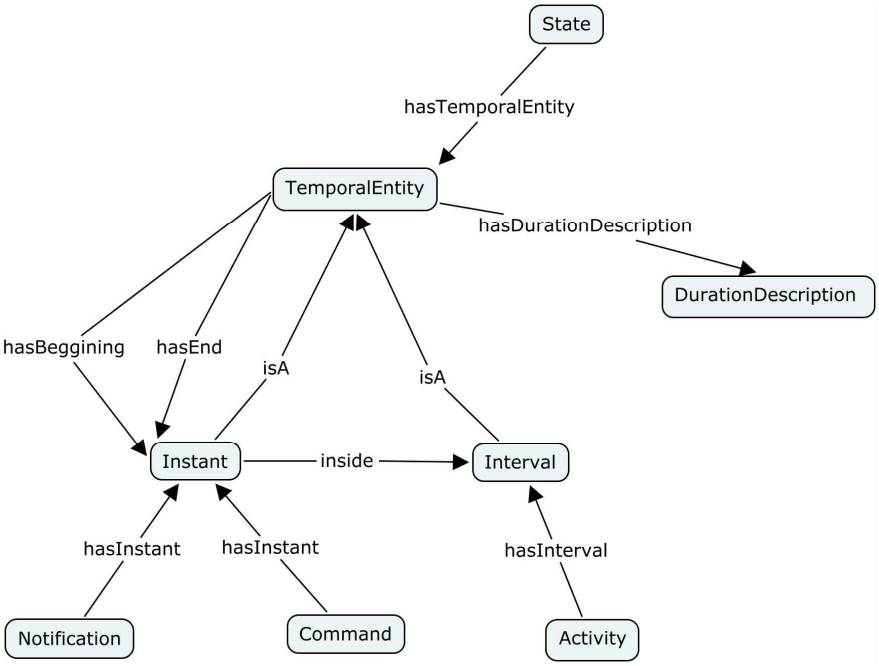

Figura 23: Classe OWL-Time [W3C, 2006]

4.2.5 Classe para Agentes de Software

Os softwares instalados no ambiente são os responsáveis por fornecer os serviços de automação inteligente. A utilização de um único software localizado em um computador central não atende as necessidades da computação pervasiva, pois podem existir inúmeros dispositivos, muitas vezes móveis, que enviam dados a todo o momento.

A utilização de diversas aplicações no ambiente torna necessária a troca de mensagens para comunicar ou solicitar uma ação. A modelagem de uma classe que represente os softwares atuantes da residência permite a busca de aplicações que forneçam um determinado serviço para um determinado contexto. Para o arcabouço de validação, será utilizado o

conceito de sistemas de agentes [Wooldridge, 2001], desta forma a ontologia proposta inclui uma classe denominada *Agent*.

A ontologia SOUPA [Chen et al,. 2005] também possui uma classe para modelagem de agentes. Porém a ontologia para a dissertação não utilizou o mesmo modelo proposto no projeto SOUPA, pois ele descreve a arquitetura interna do agente enquanto a classe Agent da ontologia AssitiveHome prevê que cada agente possui um papel no ambiente e deve apenas seguir as regras para comunicação com outros agentes.

As classes relacionadas à classe de agentes estão disponíveis na Figura 24. Para a residência inteligente um agente pode exercer o papel de um dispositivo ou de uma funcionalidade que atende uma necessidade do usuário. Caso represente um dispositivo, o agente pertence à subclasse *ControllableAgent* e relaciona-se à uma instância da sub-classe *Controllable* e cada instância da classe *Functionality* associada ao dispositivo é considerado um objetivo a ser alcançado do agente apesar de não haver uma relação explícita. Caso o agente exerça o papel de suprir uma necessidade do usuário ele é classificado como uma instância da classe *FunctionalityAgent* e relaciona-se com uma necessidade da classe *Need*, que por sua vez, atende à um usuário e gerencia diversos dispositivos, os agentes que controlam um dispositivo normalmente são somente reativos, enquanto os agentes que atendem uma funcionalidade possuem certo grau de inteligência para decidir sobre a mudança de estado das aplicações e dispositivos.

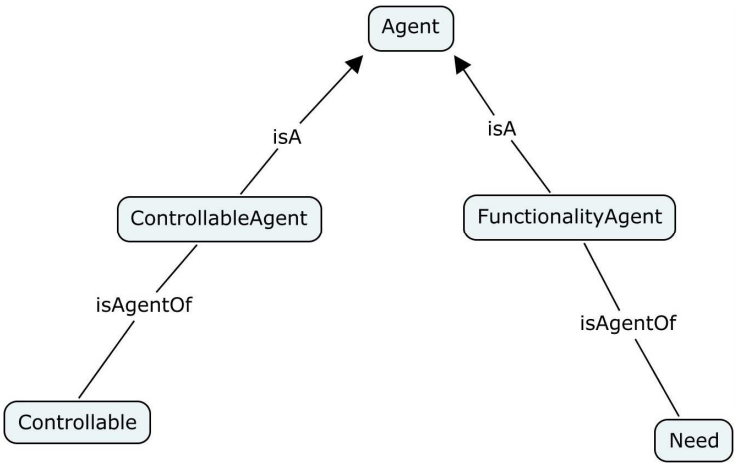

Figura 24: Classes relacionadas ao Agente

4.3 Discussão

Este capítulo descreveu as classes da ontologia e suas propriedades. Para cada classe foi considerado primeiramente o reuso de outras ontologias que foram adaptadas para o contexto da aplicação. As propriedades das classes permitem que sejam feitas associações e inferências sobre o modelo. No próximo capítulo a ontologia será aplicada em um arcabouço como modelo semântico para comunicação entre os agentes.

Capítulo 5

Arcabouço Proposto

O arcabouço descrito a seguir foi proposto para verificar que a ontologia desenvolvida atende os seguintes requisitos:

1. Oferece suporte semântico para comunicação entre os agentes de softwares.
2. Permite a descoberta de serviços.

O suporte semântico permite a comunicação entre diferentes aplicações envolvidas na automação da residência, a descoberta de serviços permite que novas funcionalidades e dispositivos sejam adicionados ao longo do tempo e por último a representação do contexto permite que a comunicação seja pertinente para um determinado ambiente ou indivíduo. Ou seja, o objetivo principal é prover um arcabouço que possibilite a automação de uma residência, fornecendo suporte para comunicação entre os dispositivos e aplicações presentes em uma determinada residência.

O desenvolvimento do arcabouço foi apoiado por uma arquitetura de agentes. A escolha deve-se às características dos agentes de sociabilidade, reatividade, pró-atividade e autonomia, que são adequadas para o desenvolvimento de uma residência inteligente. Aplicações de domótica usualmente possuem diversos aplicativos espalhados pelos ambientes e que devem interagir para que os serviços de automação sejam executados com o mínino de intervenção possível do usuário.

A interoperabilidade entre dispositivos e softwares só pode ser atingida quando a comunicação é pertinente para o nível semântico e sintático. A ontologia oferece suporte semântico, porém também é necessário escolher um protocolo comum para que a comunicação seja efetiva. Na arquitetura proposta as normas de interação seguem os protocolos descritos pela FIPA para comunicação em seus três níveis: protocolos de interação, atos comunicativos e linguagens de conteúdo. O protocolo FIPA é implementado pela maioria dos arcabouços de agentes, evitando também que o desenvolvedor tenha a preocupação com seu desenvolvimento.

A ontologia define uma especificação padrão para comunicação entre diferentes dispositivos independente da tecnologia, porém ela atua na camada de software da aplicação. Para cada novo dispositivo adicionado à residência é necessário que um driver específico seja desenvolvido com a finalidade de traduzir os sinais recebidos em dados computacionais ou vice versa. Na camada abaixo da aplicação o desenvolvedor da solução poderá utilizar a tecnologia de rede que considerar mais adequada, desde que os dados sejam transferidos para um servidor que possua capacidade de comunicação TCP/IP, permitindo que a aplicação comunique-se com os demais elementos da arquitetura.

A Figura 23 apresenta uma visão geral da arquitetura e seus componentes serão descritos a seguir.

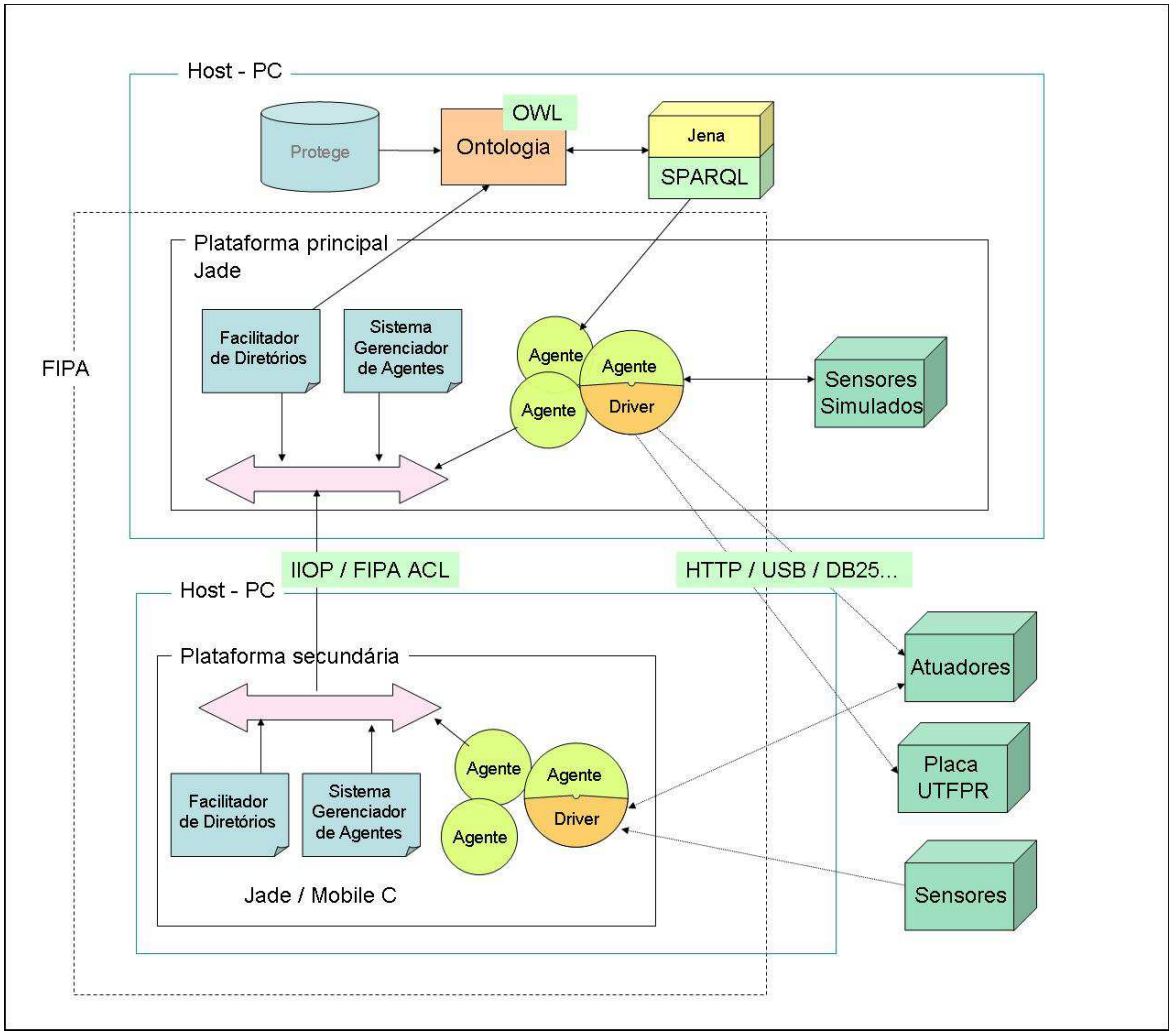

Figura 25: Arquitetura do Arcabouço [Fonte: O Autor]

5.1 Protocolos de Comunicação

A troca de mensagens entre os agentes ocorre através dos protocolos estabelecidos pela FIPA. A maioria das interações dos agentes no arcabouço proposto utiliza o protocolo *Request Interaction* [FIPA 2002a]. A FIPA não especifica uma linguagem do conteúdo mandatória e para o arcabouço foram utilizadas três linguagem de conteúdo: FIPA SL, SPARQL e RDF. As mensagens com conteúdo FIPA SL representam os dados coletados dos sensores, ou os comandos de atuação em relação ao usuário ou à residência. O exemplo da Figura 26 apresenta uma mensagem FIPA SL, uma mensagem do FIPA SL possui um campo para representação da ontologia que oferece suporte semântico para o conteúdo da mensagem [FIPA, 2002a]. O protocolo *Request* determina que o conteúdo da mensagem inicial seja uma ação. Composta de um predicado com dois argumentos. Assim o conteúdo da mensagem inicia com a palavra *action* precedida do agente que executará ação e do termo que indica a ação a ser executada.

```
(REQUEST
:sender  (agent-identifier
:name sensor@192.168.0.19:1099/JADE
:addresses (sequence http://leticia:7778/acc
))
:receiver  (set (agent-identifier
:name LampadaQ@192.168.0.19:1099/JADE ))
:content  "((action (agent-identifier
:name @192.168.0.19:1099/JADE) (OnCommand
:functionality (Functionality
:functionality_inst OnOffFunctionality)
:controllable (Controllable
:controllable_inst SimpleLampBedroom)
:command_value On)))"
:language  fipa-sl
:ontology  AssistiveHome
:protocol  fipa-request
)
```

Figura 26: Exemplo de mensagem [Fonte: O Autor]

No JADE a ontologia é baseada em predicados, conceitos e ações. Para o arcabouço as ações correspondem aos comandos da classe *Command* da ontologia seguido dos conceitos funcionalidade (classe *Functionality*) e do dispositivo (Classe *Controllable*) ao qual o comando será executado. Por último também está incluído o valor do parâmetro que será alterado. A Tabela 2 apresenta o esquema da ação *Command* na classe que define a ontologia no JADE. As especificações da funcionalidade e do dispositivo são necessárias já que cada dispositivo possui diversas funcionalidades e da mesma forma um agente pode ser responsável por mais de um dispositivo. O parâmetro da mensagem para o comando *OnCommand* será sempre “On”, porém existem dispositivos que permitem valores contínuos como por exemplo a ação de alterar a temperatura.

Tabela 2: Estrutura da ação no JADE [Fonte: O Autor]

```
// Structure of the schema for the Command agent action
AgentActionSchema as = (AgentActionSchema) getSchema(COMMAND);
as.add(COMMAND_FUNCTIONALITY , (ConceptSchema) getSchema(FUNCTIONALITY));
as.add(COMMAND_CONTROLLABLE, (ConceptSchema) getSchema(CONTROLLABLE));
as.add(COMMAND_VALUE, (PrimitiveSchema) getSchema(BasicOntology.STRING));
```

Cada um dos conceitos que compõe a ontologia possui uma classe Java, assim quando um agente precisa requisitar a ação de um dispositivo no ambiente ele deve primeiramente instanciar os valores para as classes *Command*, *Functionality e Controllable*. Então o JADE automaticamente converte os valores das classes no conteúdo da mensagem FIPA SL. Um exemplo da criação de uma mensagem está presente na Tabela 3.

Tabela 3: Criação das instâncias dos ambientes da residência [Fonte: O Autor]

```
1   // REQUEST message
2   ACLMessage msg = new ACLMessage(ACLMessage.REQUEST);
3   msg.setProtocol(FIPANames.InteractionProtocol.FIPA_REQUEST);
4   // coded = fipa-sl and ontology = HomeOntology
5   msg.setLanguage(codec.getName());
6   msg.setOntology(ontology.getName());
7   ACLMessage acl = msg.createReply();
8   acl.setPerformative(ACLMessage.REQUEST);
```

```
9
10  // Prepare the content.
11  Functionality func = new Functionality();
12  func.setFunctionality_inst("OnOffFunctionality");
13  Controllable cont = new Controllable();
14  cont.setControllable_inst("SimpleLampBedroom");
15  Command on = new Command();
16  on.setCommand_Value("On");
17  on.setFunctionality(func);
18  on.setControllable(cont);
19
20  // set the action with the Command values
21  jade.content.onto.basic.Action act =  new
22  jade.content.onto.basic.Action (new AID( id , AID.ISLOCALNAME),  on);
23        try {
24  // fill the content of the message
25              getContentManager().fillContent(msg, act);
26        } catch (CodecException e) {
27              e.printStackTrace();
28        } catch (OntologyException e) {
29              e.printStackTrace();
30        }
31  // sets the address of the receiver
32  msg.addReceiver( new AID( id , AID.ISLOCALNAME) );
33  // sends the message
34  send(msg);
```

Após executar a ação o agente participante envia uma mensagem do tipo *Inform*, que é outro *performative* especificado pela FIPA, com a mensagem de conclusão da ação ou caso a ação tenha gerado uma saída pode também informar este resultado. As funcionalidades da subclasse *QueryFunctionality* da ontologia normalmente retornam o estado do dispositivos.

Outro protocolo da FIPA presente no arcabouço desta dissertação é o FIPA *Query*, ele é utilizado quando um agente precisa solicitar alguma informação de outro agente para busca de agentes que executam determinada função. A classe *Functionality* da ontologia especifica as funções que podem retornar estados ou outras informações de dispositivos, desta forma é possível um agente identificar previamente quais as funções que outro agente pode retornar e executar a solicitação correta.

O agente de gerência, que será explicado em detalhes no próximo item, é responsável por manter a ontologia atualizada e também por executar as buscas na ontologia. Qualquer agente pode enviar uma mensagem do tipo *query-ref* com uma sentença SPARQL válida para o agente de gerência e após executar a *query* ele envia a resposta para o agente que solicitou em formato XML. A Tabela 4 possui um exemplo de mensagem SPARQL para busca dos agentes que controlam dispositivos na Sala de Estar, que corresponde a instância *LivingRoom1*.

Tabela 4: Exemplo de mensagem SPARQL para busca de agentes [Fonte: O Autor]

```
1  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2  PREFIX owl: <http://www.w3.org/2002/07/owl#>
3  PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
4  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
5  PREFIX pre0: <http://paginapos.utfpr.edu.br/a1252828/assistive-home-
6  ontology/assistivehome.owl#>
7
```

```
SELECT ?z
WHERE{
?x  pre0:isIn pre0:LivingRoom1.
?z pre0:isAgentOf ?x
}
```

Por último, para que as instâncias de um novo agente sejam criadas na ontologia, os agentes enviam para o agente gerente uma mensagem RDF com as instâncias a serem criadas. O conteúdo RDF da Tabela 5 é um exemplo de mensagem enviada pelo agente *AgentLampBedroom*. As instâncias das classes de agente (*Agent*), dispositivo (*Controllable*), estado (*State*) e o valor do estado (*StateValue*) são obrigatórias para os agentes de dispositivos, além das propriedades: *isIn* para referenciar a localização (classe BuildingEnvironment) de um dispositivo (classe *Controllable*), *hastate* para associar o estado (*State*) ao dispositivo (*Controllable*) e *hasStateValue* para criar a relação entre o estado (*State*) e o valor do estado atual (*StateValue*).

Tabela 5: Mensagem RDF para criação das instâncias

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:j.0="http://paginapos.utfpr.edu.br/a1252828/assistive-
home-ontology/assistivehome.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >
  <rdf:Description
rdf:about="http://paginapos.utfpr.edu.br/a1252828/assistive-
home-ontology/assistivehome.owl#SimpleLampBedroom">
    <j.0:hasState
rdf:resource="http://paginapos.utfpr.edu.br/a1252828/assistive-
home-ontology/assistivehome.owl#OnOffStateLampBedroom"/>
    <j.0:isIn
rdf:resource="http://paginapos.utfpr.edu.br/a1252828/assistive-
home-ontology/assistivehome.owl#Bedroom1"/>
    <rdf:type
rdf:resource="http://paginapos.utfpr.edu.br/a1252828/assistive-
home-ontology/assistivehome.owl#SimpleLamp"/>
  </rdf:Description>
  <rdf:Description
rdf:about="http://paginapos.utfpr.edu.br/a1252828/assistive-
home-ontology/assistivehome.owl#AgentLampBedroom">
    <rdf:type
rdf:resource="http://paginapos.utfpr.edu.br/a1252828/assistive-
home-ontology/assistivehome.owl#ControllableAgent"/>
  </rdf:Description>
  <rdf:Description
rdf:about="http://paginapos.utfpr.edu.br/a1252828/assistive-
home-ontology/assistivehome.owl#">
    <owl:imports
rdf:resource="d:/mestrado/workspace/residencia/home.owl"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#Ontology"/>
  </rdf:Description>
  <rdf:Description
rdf:about="http://paginapos.utfpr.edu.br/a1252828/assistive-
home-ontology/assistivehome.owl#OffStateValueLampBedroom">
    <rdf:type
```

```
40 rdf:resource="http://paginapos.utfpr.edu.br/a1252828/assistive-
41 home-ontology/assistivehome.owl#OffStateValue"/>
42   </rdf:Description>
43   <rdf:Description
44 rdf:about="http://paginapos.utfpr.edu.br/a1252828/assistive-
45 home-ontology/assistivehome.owl#Bedroom1">
46     <rdf:type
47 rdf:resource="http://paginapos.utfpr.edu.br/a1252828/assistive-
48 home-ontology/assistivehome.owl#Bedroom"/>
49   </rdf:Description>
50   <rdf:Description
51 rdf:about="http://paginapos.utfpr.edu.br/a1252828/assistive-
52 home-ontology/assistivehome.owl#OnOffStateLampBedroom">
53     <j.0:hasStateValue
54 rdf:resource="http://paginapos.utfpr.edu.br/a1252828/assistive-
55 home-ontology/assistivehome.owl#OffStateValueLampBedroom"/>
56     <rdf:type
57 rdf:resource="http://paginapos.utfpr.edu.br/a1252828/assistive-
58 home-ontology/assistivehome.owl#OnOffState"/>
59   </rdf:Description>
60 </rdf:RDF>
```

5.2 Agentes do Arcabouço

Os agentes seguem a premissa de que um sistema de automação é composto por dispositivos e aplicações que suportam funcionalidades para os usuários da residência. Desta forma, foram identificados os dois tipos básicos de agentes: agentes que suportam dispositivos e agentes que suportam funções, cada um possui uma classe que o representa na ontologia. Podem existir no arcabouço diversos agentes de dispositivos e funções.

Além destes agentes também é proposto um agente de gerência que possui todas as instâncias da ontologia e permite a busca de agentes através de *queries* SPARQL. Os três tipos de agentes serão descritos a seguir.

5.2.1 Agente Gerente

O agente gerente possui três objetivos que são apresentados no diagrama de atividades da Figura 27. O primeiro objetivo é cadastrar as instâncias do ambiente assim que iniciado na plataforma, para o estudo de caso corresponde ao quarto e a sala e as dimensões deste ambiente. Este processo é importante, pois ao incluir dispositivos no ambiente eles pertencerão ou estarão momentaneamente em um cômodo. O Agente gerente utiliza o arcabouço JENA para ler o modelo da ontologia e criar as instâncias diretamente no código do agente conforme demonstra a Tabela 6. O arquivo OWL é carregado na memória do agente e em seguida são criadas instâncias utilizando os métodos da classe *OntoModel* do JENA. O método *getOntClass* cria um objeto que representa a classe na ontologia e então o método *createIndividual* cria a instância do indivíduo na ontologia definindo também a classe à que pertence.

Utilizando a mesma lógica é possível cadastrar dispositivos estáticos da residência nas subclasses de objetos não controláveis da classe *BuildingThing* como por exemplo, um sofá ou uma cama. Dispositivos automatizados também podem ser instanciados manualmente,

porém o objetivo é que eles sejam instanciados em tempo de execução quando um novo agente é iniciado e exerce o papel daquele dispositivo no sistema. Após o cadastro cada instancia é representada na base de conhecimento na memória do agente e em um arquivo OWL.

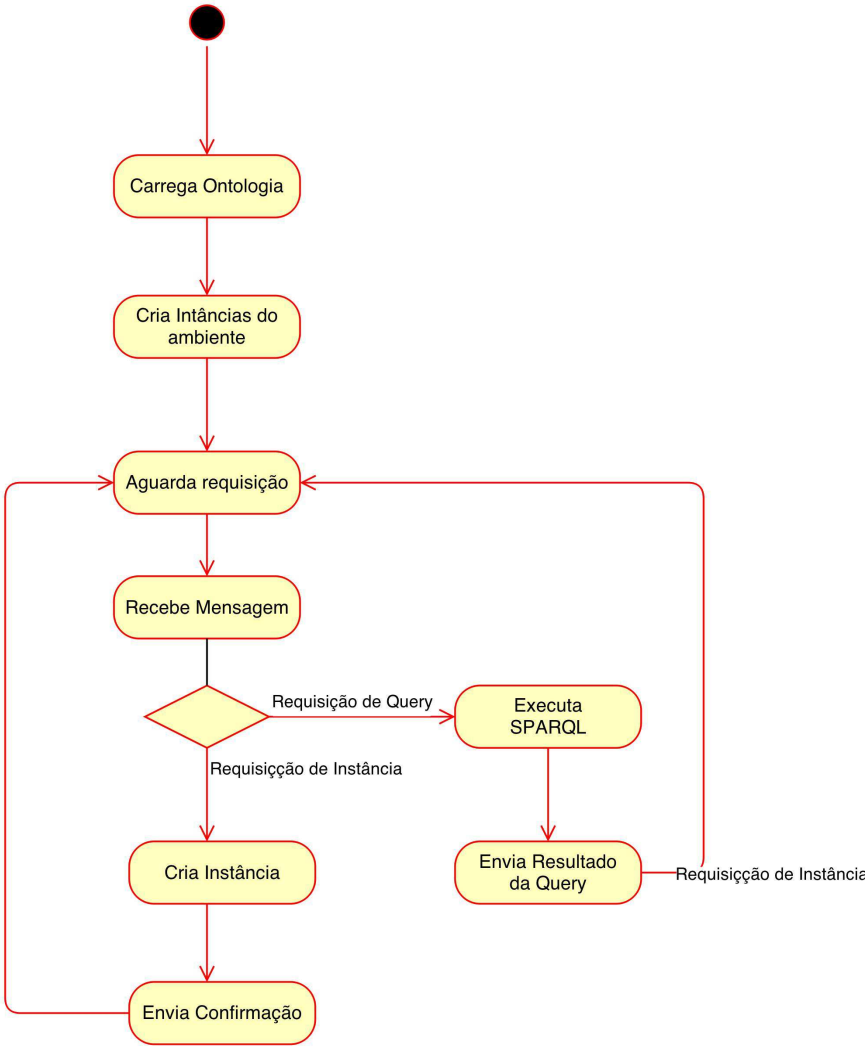

Figura 27: Diagrama de Atividade do Agente Gerente [Fonte: O Autor]

Tabela 6: Criação das instâncias dos ambientes da residência [Fonte: O Autor]

```
1   //creates the ontology model
2   OntModel onto = ModelFactory.createOntologyModel();
3   onto.setDynamicImports(true);
4   InputStream in =
5   FileManager.get().open("d:/mestrado/workspace/residencia/home.owl");
6         if (in == null) {
7             throw new IllegalArgumentException("File: " +
8   "d:/mestrado/workspace/residencia/home.owl" + " not found");
9         }
10  //reads the ontology from file
11  onto.read(in, "");
12
13  //creating home environment instances
14  NS = "http://paginapos.utfpr.edu.br/a1252828/assistive-home-
15  ontology/assistivehome.owl#";
16  OntClass att = onto.getOntClass(NS + "Bedroom");
```

```
17  Individual I1 = onto.createIndividual(NS + "Bedroom1", att);
18  OntClass att2 = onto.getOntClass(NS + "LivingRoom");
19  Individual I2 = onto.createIndividual(NS + "LivingRoom1", att2);
```

Após o cadastro o agente também tem o objetivo de instanciar os demais agentes na ontologia, para que isso ocorra quando um novo agente é iniciado na plataforma ele deve solicitar seu cadastro através do protocolo *FIPA-Request.* Conforme especificado no item anterior, o novo agente inicia a protocolo *FIPA-Request* enviando uma mensagem contendo a classe a que pertence, e os valores para instância do dispositivo ou função que deseja exercer. Neste caso o conteúdo da mensagem é do tipo RDF e número de parâmetros é variável de acordo com a classe herdada pelo dispositivo a ser instanciado. O agente gerente primeiro extrai os valores do grafo RDF e então instancia os valores na ontologia caso o agente envie uma solicitação para o cadastro de instâncias repetidas, por exemplo, o agente gerente retornará para o outro agente uma mensagem de *Failure*, com o nome da instância não localizada. Nos demais casos o agente facilitador envia uma mensagem de *Inform-done* ao final do processo.

Por último, o agente gerente também tem por objetivo executar queries SPARQL na ontologia. Quando outro agente precisa buscar informações na ontologia é possível enviar uma mensagem com a query através do protocolo FIPA Query conforme especificado no item anterior.

5.2.2 Agente de Funcionalidade e de Dispositivos

Os agentes de Funcionalidade e Dispositivos interagem através do protocolo FIPA-*Request* quando a intenção é solicitar alguma ação do outro agente. Neste caso, o protocolo é necessário, pois o agente receptor da mensagem pode aceitar ou recusar a solicitação além de informar por final o resultado desta interação, ou seja, sucesso ou falha.

Existe também a interação apenas para informar um agente interessado sobre o estado de um dispositivo, neste caso não é necessário nenhum protocolo de informação já que o receptor não precisa enviar respostas ao emissor. O agente apenas envia uma mensagem de *inform* [FIPA, 2002d] para os agentes interessados.

A Figura 28 demonstra um exemplo das relações de um agente de dispositivos. Ela foi adaptada de Bonino et al. (2008), ela representa um dispositivo *sample_dimmer_lamp* localizado na sala de estar. Este dispositivo é controlado pelo agente Agent_sample_dimmer_lamp. Na ontologia um dispositivo da classe DimmerLamp pode executar três funções: *QueryFunctionality LightRegulationFunctionality e OnOffFunctionality,* assim o agente deve ter por objetivo estas 3 funções. A ontologia também delimita os comandos de cada função para que os agentes saibam que tipo de dados são permitidos para cada mensagem.

Já os agentes que executam algum tipo de controle para suprir uma necessidade do usuário são instâncias da classe *FunctionalityAgent* e estão relacionados à classe *Need* que por sua vez relaciona-se com o usuário. Este tipo de agente pode gerenciar diversos dispositivos através da propriedade de objetos *managerOf* relacionada à classe *Controllable.* Como se relacionam com os agentes da classe de dispositivo as funcionalidades de um *FunctionalityAgent* são de certa forma opostas as funcionalidades de um agente de dispositivo.

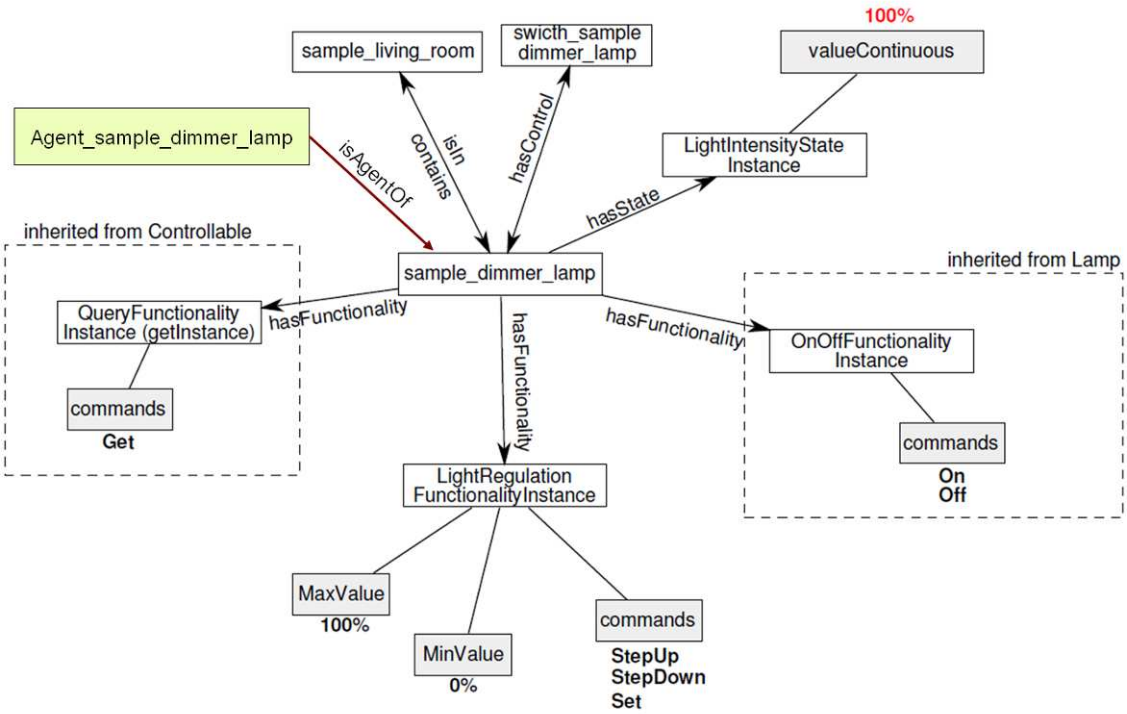

Figura 28 - Exemplos de Notificações e Relações entre Classes [Fonte: O Autor adaptado de Bonino et al., 2008]

5.3 Experimentos Realizados

Os experimentos realizados têm como objetivo validar a viabilidade prática da ontologia para o arcabouço. O experimento realizado é baseado no seguinte contexto:

Em um final de tarde ainda iluminado o morador João adentra sua residência pela sala. Neste momento o sensor de presença detecta a presença do morador e devido a iluminação externa as cortinas são abertas. Com o passar das horas a iluminação diminui e o sensor de iluminação verifica que a luz do ambiente não é suficiente, solicitando que as cortinas sejam fechadas e a lâmpada acesa. Algum tempo depois, a temperatura no ambiente torna-se muito alta e o ar condicionado é acionado. O morador decide então deslocar-se para o quarto e o sensor de presença efetua a ligação da lâmpada deste ambiente enquanto o sensor de temperatura identifica que o ambiente está muito quente e o ar condicionado é ligado. Por fim, como o morador deseja dormir ele aciona o interruptor desligando a lâmpada do quarto.

Em um segundo momento uma nova lâmpada é incluída no ambiente da sala. Quando o agente de funcionalidade de iluminação pesquisa por agentes de lâmpadas na sala, ele receberá a referencia de dois agentes. Para essa situação o agente de iluminação busca a lâmpada que se relaciona com o usuário através da posição relativa entre dois objetos. Por exemplo, uma lâmpada pode estar acima de um indivíduo, através da propriedade de objeto *upTo*.

Através do texto é possível identificar que a residência possui:

- Dois ambientes: Quarto e Sala.
- A sala possui:
  - Três sensores:
    - Sensor de presença
    - Sensor de iluminação
    - Sensor de temperatura
  - Três dispositivos:
    - Lâmpada
    - Persiana
    - Ar condicionado
- O quarto possui:
  - Um sensor:
    - Sensor de presença
  - Dois dispositivos
    - Lâmpada
    - Ar condicionado

Para validação da aplicação foi construído um protótipo para automação dos cinco dispositivos. Toda a aplicação foi desenvolvida em um PC configurado com um processador Intel Pentium 4 de 3.00 GHz e 512 MB de memória RAM. Uma placa para automação desenvolvida pelo departamento de eletrotécnica da Universidade Tecnológica Federal do Paraná foi utilizada para interface entre o computador e os dispositivos da residência. A placa possui 12 relés e 5 entradas lógicas e está conectada a uma fonte de 12 volts. Os dispositivos são representados por lâmpadas em uma planta baixa conforme Figura 29, onde também é possível verificar a placa responsável pela automação.

Nos itens 5.2.1 e 5.2.2 foram descritos os papeis dos agentes no arcabouço e os agentes desenvolvidos no protótipo serão descritos a seguir.

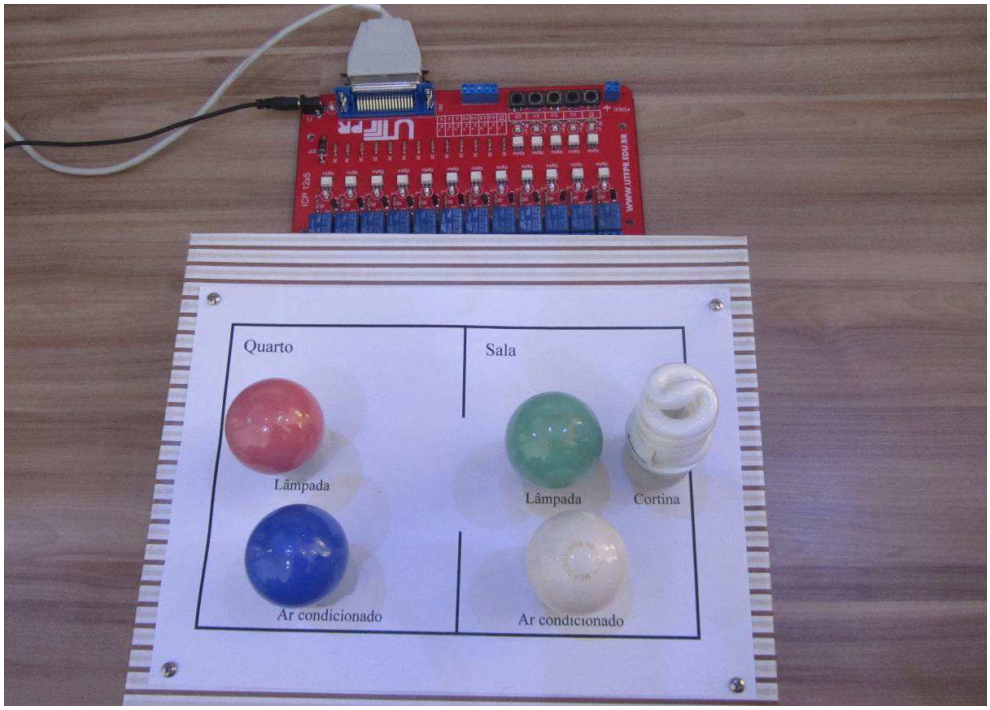

Figura 29: Protótipo e placa de automação [Fonte: O Autor]

Agente Gerente

O agente gerente possui três objetivos. Ele cadastra as instâncias do ambiente que para o estudo de caso corresponde aos dois cômodos do código de exemplo do mesmo item (Tabela 7). Após o cadastro o agente possui dois outros objetivos, executar buscas na ontologia e cadastrar instâncias requisitadas por outros agentes.

Agentes em JADE são desenvolvidos de acordo com os seus comportamentos, no caso do agente gerente ele possui um comportamento cíclico aguardando novas mensagens, na Tabela 6 é possível observar a porção do código do agente que executa esse comportamento. Dependendo do ato comunicativo da mensagem recebida, linha 5, o agente seleciona qual a atividade que deve executar. Caso receba uma mensagem do tipo *Request*, é provável que a mensagem seja de cadastro de instâncias na ontologia e então o agente verifica a linguagem de conteúdo da mensagem (linha 9). Se o conteúdo é tipo RDF o gerente prossegue com a instância, porém se for de outra linguagem a requisição é rejeitada e uma mensagem de *Refuse* é retornada para o agente que iniciou o pedido.

Finalmente, se a mensagem recebida apresentar o ato comunicativo *Query_Ref* o agente verifica a linguagem de conteúdo que deve ser do tipo SPARQL e executa a requisição em caso positivo ou envia a mensagem de rejeição para a mensagem apresente outra linguagem.

Tabela 7: Mensagem RDF para criação das instâncias

```
1  addBehaviour(new CyclicBehaviour(this) {
2    public void action() {
3      ACLMessage msg= receive();
4        if (msg!=null){
5           switch (msg.getPerformative()) {
6             //Requests to update ontology
7             case (ACLMessage.REQUEST):
8               //RDF message to add new instances
9               if (msg.getLanguage()!= null &&
10 msg.getLanguage().equals("RDF")){
11                if (!msg.getContent().equals("")){
12                  OntModel model_inst =
13 ModelFactory.createOntologyModel();
14                  model_inst.read(new
15 StringReader(msg.getContent()), "");
16                  onto.add(model_inst);
17
18                  //Inform done
19                   ACLMessage msg2 = msg.createReply();
20                   msg2.setPerformative(ACLMessage.INFORM);
21                   msg2.setContent("done");
22                   send(msg2);
23                 }
24               }else{
25                   //Refuse mesage
26                   ACLMessage msg2 = msg.createReply();
27                   msg2.setPerformative(ACLMessage.REFUSE);
28                   send(msg2);
29               }
30             break;
31
32             case (ACLMessage.QUERY_REF):
```

```
33                  if (msg.getLanguage()!= null &&
34 msg.getLanguage().equals("SPARQL")){
35                     try {
36                       Query query =
37 QueryFactory.create(msg.getContent());
38                       // Execute the query and obtain results
39                       QueryExecution qe =
40 QueryExecutionFactory.create(query, onto);
41                       com.hp.hpl.jena.query.ResultSet results =
42 qe.execSelect();
43                       String  content =
44 ResultSetFormatter.asXMLString(results);
45
46                       //Send response with the results
47                       ACLMessage msg2 = msg.createReply();
48                       msg2.setPerformative(ACLMessage.INFORM);
49                       msg2.setContent(content);
50                       send(msg2);
51                     } catch (QueryParseException e) {
52                        e.printStackTrace();
53                        //Send response with the results
54                        ACLMessage msg2 = msg.createReply();
55                        msg2.setPerformative(ACLMessage.FAILURE);
56                        send(msg2);
57                     }
58                   }
59                break;
60             }
61             block();
62          }
63       }
64 }
```

Agente Lâmpada Sala e Agente Lâmpada Quarto

O Agente da lâmpada da sala e do quarto tem por objetivo controlar a lâmpada do respectivo ambiente. Ambos os agentes possuem o mesmo comportamento, a única diferença é a relação com a classse *BuildingEnvironment* da ontologia, enquanto a instância de um agente possui a propriedade *isIn Bedroom1* o outro possui a propriedade *isIn LivingRoom1*.

Quando estes agentes são iniciados uma mensagem foi enviada para o agente gerente e criou-se na base de conhecimento uma instância da classe *ControllableAgent* e uma instância para a lâmpada da classe *Lamp* que é uma subclasse de *Controllable*. O estado da classe *State* foi instanciado como *Off* (desligado) um exemplo do processo da instância do quarto é o código da Tabela 8 que cria a mensagem RDF e envia para o gerente.

Tabela 8: Mensagem RDF para criação das instâncias

```
1  OntModel onto = ModelFactory.createOntologyModel();
2        onto.setDynamicImports(true);
3
4  String NS = "http://paginapos.utfpr.edu.br/a1252828/assistive-
5  home-ontology/assistivehome.owl" + "#";
6        com.hp.hpl.jena.ontology.Ontology on =
7  onto.createOntology(NS);
8
9  on.addImport( onto.createResource(
10 "d:/mestrado/workspace/residencia/home.owl" ) );
11
12 //creating lamp instances
13 NS = "http://paginapos.utfpr.edu.br/a1252828/assistive-home-
14 ontology/assistivehome.owl#";
15 OntClass att = onto.getOntClass(NS + "Bedroom");
16 Individual I1 = onto.createIndividual(NS + "Bedroom1", att);
17
18 //Agent Lamp
19 att = onto.getOntClass(NS + "ControllableAgent");
20 Individual I3 = onto.createIndividual(NS + "AgentLampBedroom",
21 att);
22
23 //Lamp
24 att = onto.getOntClass(NS + "SimpleLamp");
25 ObjectProperty ob4 = onto.getObjectProperty(NS + "isIn");
26 Individual I4 = onto.createIndividual(NS + "SimpleLampBedroom",
27 att);
28 onto.add(I4, ob4, I1);
29 ObjectProperty agentof = onto.getObjectProperty(NS +
30 "isAgentOf");
31 nto.add(I3, agentof, I4);
32
33 //State Lamp
34 att = onto.getOntClass(NS + "OnOffState");
35 ObjectProperty ob5 = onto.getObjectProperty(NS + "hasState");
36 Individual I5 = onto.createIndividual(NS +
37 "OnOffStateLampBedroom", att);
38 onto.add(I4, ob5, I5);
39
40 //StateValue Lamp
41 OntClass att6 = onto.getOntClass(NS + "OffStateValue");
42 ObjectProperty ob6 = onto.getObjectProperty(NS +
43 "hasStateValue");
44 Individual I6 = onto.createIndividual(NS +
45 "OffStateValueLampBedroom", att6);
46 onto.add(I5, ob6, I6);
47
48 //Writes ontology
49 StringWriter sw = new StringWriter();
50 onto.write(sw, "RDF/XML");
51
52 //Sends message
53 ACLMessage msgont = new ACLMessage(ACLMessage.REQUEST);
54 msgont.setProtocol(FIPANames.InteractionProtocol.FIPA_REQUEST);
55 msgont.setPerformative(ACLMessage.REQUEST);
56 msgont.addReceiver( new AID( "gerente" , AID.ISLOCALNAME) );
```

```
57 msgont.setContent( sw.toString() );
58 msgont.setLanguage("RDF");
59 send(msgont);
```

Após o cadastro o agente da lâmpada também possui um comportamento de aguardar novas mensagens. Porém este agente aceita somente mensagens de conteúdo FIPA SL e que utilizem a ontologia AssitiveHome como vocabulário. A Tabela 9 extrai o código JADE do agente que bloqueia as mensagens fora do padrão.

Tabela 9: Bloqueio de mensagens de acordo com a ontologia e linguagem [Fonte: O Autor]

```
1 //Only accepts messages with SL language and AssistiveHome
2 MessageTemplate mt = MessageTemplate.and(
3 MessageTemplate.MatchLanguage(codec.getName()),
4 MessageTemplate.MatchOntology(ontology.getName()) );
5 ACLMessage msg = blockingReceive(mt);
```

As mensagens de conteúdo FIPA SL precisam ser decodificadas para que o agente entenda os termos da ontologia. No JADE uma ontologia é representada através de classes Java. Quando confirmado que a mensagem refere-se à linguagem esperada o agente transforma o conteúdo da mensagem enviado em formato de texto para as classes Java da ontologia conforme a Tabela 10. Finalmente o agente verifica se a mensagem refere-se à funcionalidade de ligar e desligar a lâmpada (classe *OnOffFunctionality*) e ao dispositivo comandado pelo agente, no caso a lâmpada do quarto (instância *SimpleLampBedroom*) e atua conforme a ação solicitada e o respectivo parâmetro.

Tabela 10: Extração do conteúdo do texto da mensagem para classes Java [Fonte: O Autor]

```
1  ContentElement ce = getContentManager().extractContent(msg);
2  Concept action = ((Action)ce).getAction();
3
4  if (action instanceof Command) {
5
6  Command command = (Command) action;
7  Functionality f = command.getFunctionality();
8  Controllable contrl = command.getControllable();
9
10 //Verifies the Lamp instance and the requested functionality
11 if   (contrl.getControllable_inst().equals("SimpleLampBedroom")&&
11 f.getFunctionality_inst().equals("OnOffFunctionality") ) {
12   if (!command.getCommand_Value().equals(state)) {
     …
```

Além da funcionalidade *OnOffFunctionality* o agente também implementa as funcionalidades *StateChangeNotificationFunctionality e QueryFunctionality.*

A *StateChangeNotificationFunctionality* envia uma mensagem para o agente de gerência notificando da alteração do estado. O agente gerente por sua vez atualiza a instância na ontologia para a classe *StateValue.*

A funcionalidade *QueryFunctionality* possui uma lógica semelhante a *OnOffFunctionality,* a diferença é que o agente verifica se a mensagem recebida refere-se a f.getFunctionality_inst().equals(*"QueryFunctionality* ") e a mensagem de retorno é o valor do estado da classe StateValue, enquanto que para a funcionalidade *OnOffFunctionality* a mensagem de retorno é apenas de confirmação ou falha.

Agente Ar Condicionado Sala e Agente Ar Condicionado Quarto

Os agentes de ar condicionado controlam o ar condicionado da sala e do quarto respectivamente. A lógica interna dos agentes que controlam os dispositivos é bastante semelhante, principalmente porque a placa permite apenas dispositivos com valores discretos, no caso *On e Off.* Quando iniciados estes agentes criam na base de conhecimento uma instância da classe *ControllableAgent* e uma instância para o ar condicionado, da subclasse de *Controllable, AirConditioningSystem e* com o estado *Off (desligado).* Um ar condicionado possui as funcionalidades *OnOffFunctionality, TemperatureRegulationFunctionality,* e as funcções herdadas *StateChangeNotificationFunctionality e QueryFunctionality.* Porém, a funcionalidade *TemperatureRegulationFunctionality* não foi desenvolvida já que a lâmpada que simula a automação possui apenas o estado de liga e desliga.

Agente Cortina da Sala

O agente que controla a cortina também possui a lógica semelhante aos demais dispositivos, ele criou uma instância da classe *ControllableAgent* e uma instância da subclasse de *Controllable*, *ShutterActuator* na ontologia. Apesar do agente estar definido como cortina ele na verdade é responsável pelo atuador que controla esta cortina. O agente possui as funcionalidades *UpDownRestFunctionality, StateChangeNotificationFunctionality e QueryFunctionality.* A funcionalidade *UpDownRestFunctionality* é semelhante a funcionalidade *OnOffFunctionality* a única diferença entre os códigos é que os valores possíveis para o StateValue são *Up* e *Down.*

Agente de Sensor de Presença e Temperatura do Quarto

O agente de sensores do quarto tem por objetivo simular a detecção de presença e a alteração da temperatura no respectivo ambiente. Ele possui uma interface simples, Figura 30, com um botão para detecção de presença e dois botões para alterar o valor percebido da temperatura no ambiente.

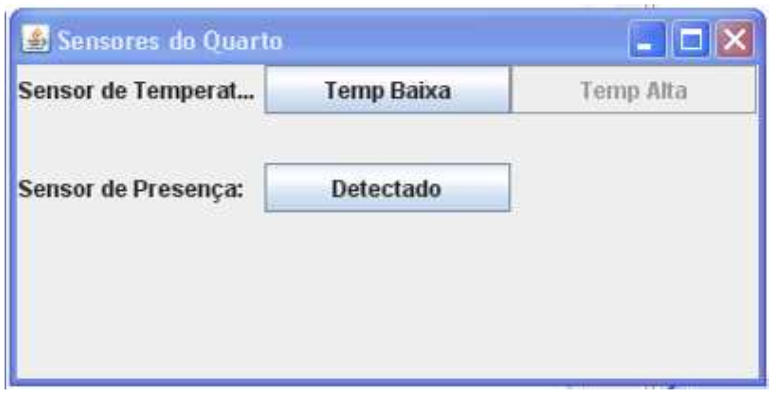

Figura 30: Interface dos sensores do quarto [Fonte: O Autor]

Quando iniciado o agente envia uma mensagem para o gerente com a finalidade de criar as instâncias na base de conhecimento. O agente é do tipo *ControllableAgent* , e possui dois dispositivos o sensor de presença é uma instância da classe *PresenceSensor*, subclasse de *Controllable*, enquanto o sensor de temperatura criou uma instância da classe *TemperatureSensor*, também subclasse de *Controllable*. As instâncias dos sensores estão relacionadas ao agente através da propriedade *isAgentOf*. Este agente possui as funcionalidades *PresenceNotificationFunctionality*, e as funções herdadas *StateChangeNotificationFunctionality e QueryFunctionality* para o sensor de presença e *TemperatureMeasurementNotificationFunctionality StateChangeNotificationFunctionality e QueryFunctionality* para o sensor de temperatura.

Este agente não altera diretamente o estado de nenhum atuador do ambiente, ele apenas envia mensagens referentes às mudanças de estados dos sensores para o agente de controle de funcionalidade pertinente. Quando o botão de presença é pressionado uma mensagem é enviada para o agente de controle de iluminação. A Tabela 11 apresenta o código para envio da mensagem. A mensagem do tipo FIPA *Inform* inclui a funcionalidade, no caso *PresenceNotificationFunctionality* e o dispositivo que foi acionado que corresponde a instância na ontologia *PresenceSensorBedroom*. A mensagem também inclui o valor da notificação que neste caso é *isPresent*.

Tabela 11: Envio de mensagem para detecção de presença [Fonte: O Autor]

```
1  // Create message
2  ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
3  msg.setLanguage(codec.getName());
4  msg.setOntology(ontology.getName());
5
6  // Prepare the content.
7  Functionality func = new Functionality();
8  func.setFunctionality_inst("PresenceNotificationFunctionality");
9
10 Controllable cont = new Controllable();
11 cont.setControllable_inst("PresenceSensorBedroom");
12
13 Notification not = new Notification();
14 not.setNotification_Value("isPresent");
15 not.setFunctionality(func);
16 not.setControllable(cont);
17
18 // Set the content
```

```
19 getContentManager().fillContent(msg, not);
20 msg.setPerformative(ACLMessage.INFORM);
21 msg.addReceiver( new AID( "iluminacaoq" , AID.ISLOCALNAME) );
22 send(msgont);
```

Da mesma forma, quando a temperatura é alterada através dos botões da interface uma mensagem é enviada para o agente de controle de temperatura em um processo semelhante ao apresentado na Tabela 10, porém a funcionalidade da mensagem é *TemperatureMeasurementNotificationFunctionality* bem como o dispositivo foi alterado para *TemperatureSensorBedroom* e o valor da notificação pode ser *High* ou *Low*.

Agente de Sensor de Presença, Temperatura e Iluminação da Sala

Este agente é semelhante ao agente de sensores do quarto, porém a interface possui adicionalmente um sensor de iluminação, conforme Figura 31.

Figura 31: Simulação de sensores para testes [Fonte: O Autor]

Quando iniciado o agente envia uma mensagem para o gerente com a finalidade de criar as instâncias na base de conhecimento. O agente é do tipo *ControllableAgent* , e possui três sensores. A instância adicional criada em relação ao agente de sensores do quarto é o sensor de iluminação da classe *LightSensor* e possui as funcionalidades *LightSensingFunctionality*, *LuminosityNotificationFunctionality*, e *StateChangeNotificationFunctionality.*

Agente de Controle de Iluminação da Sala

O agente de iluminação é um agente responsável pelo controle da iluminação para um usuário. Ao iniciar na plataforma o agente cria uma instancia da classe *FunctionalityAgent* e da classe *Need*. Este agente relaciona-se com os agentes: Lâmpada, Cortina, Sensor de Iluminação e Presença da sala. Caso receba uma mensagem do tipo *inform* do sensor de presença ele inicia o protocolo FIPA-*Request* e envia uma mensagem para o agente pertinente dependendo da iluminação do ambiente, solicitando a mudança do estado On/Off da lâmpada ou Up/Down cortina. Quando é possível alterar o estado o agente que foi solicitado para efetuar a mudança envia a confirmação, caso contrário envia somente a mensagem de falha.

Agente de Controle de Temperatura da Sala

O agente de controle de temperatura também é um agente de funcionalidade e a instância na ontologia segue a mesma lógica dos demais agentes. Este agente espera por mensagens do agente de sensor de temperatura através da função *StateChangeNotificationFunctionality* e caso a temperatura esteja abaixo de 18°C ele envia uma mensagem para o agente do ar condicionado para desligar o aparelho, caso a temperatura esteja acima de 25°C ele envia uma mensagem para ligar o ar novamente. As duas mensagens são solicitadas através do protocolo *Request.*

5.3.1 Resultados da Interação

O estudo de caso relatado no início do item 5.3 foi executado passo a passo observando-se o comportamento dos agentes e dos dispositivos de acordo com os resultados esperados. A plataforma de agentes do JADE possui um *sniffer* que permite identificar a troca de mensagens entre os agentes e capturar o conteúdo das mensagens, a Figura 32 apresenta a interface desta funcionalidade com todos os agentes do protótipo antes de iniciar as interações.

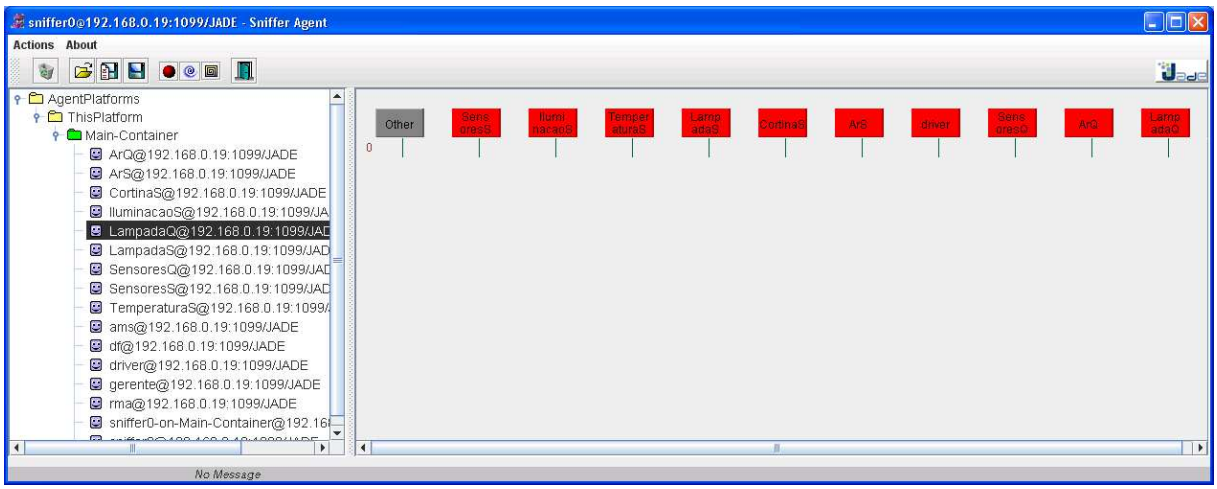

Figura 32: Sniffer do JADE antes de iniciar a interação [Fonte: O Autor]

A seguir, o estudo de caso será verificado passo a passo:

1. Passo 1: Sensor de presença detectado: O botão de sensor de presença da sala é pressionado, conforme Figura 33.

Figura 33: Botão da interface pressionado [Fonte: O Autor]

a. O botão está localizado no agente de sensores da sala e simula o acionamento do sensor de presença. Assim, quando pressionado o agente envia uma mensagem para o agente de iluminação informando da alteração (Figura 34).

```
(INFORM
:sender  ( agent-identifier :name SensoresS@192.168.0.19:1099/JADE
:addresses (sequence http://leticia:7778/acc ))
:receiver  (set ( agent-identifier :name
IluminacaoS@192.168.0.19:1099/JADE ) )
:content  "((action (agent-identifier :name
IluminacaoS@192.168.0.19:1099/JADE) (Notification :functionality
(Functionality :functionality_inst PresenceNotificationFunctionality)
:controllable (Controllable :controllable_inst PresenceSensorLivingRoom)
:notification_value isPresent)))"
:language  fipa-sl  :ontology  AssistiveHome )
```

Figura 34: Mensagem enviada pelo agente de sensores da sala [Fonte: O Autor]

2. Passo 2: Estado padrão do sensor de Iluminação é alto. (Dia)
   a. O agente de iluminação verifica a presença do usuário e como a iluminação é alta, solicita que a cortina seja aberta e a lâmpada apagada conforme Figura 35.

```
(REQUEST
:sender  ( agent-identifier :name IluminacaoS@192.168.0.19:1099/JADE
:addresses (sequence http://leticia:7778/acc ))
:receiver  (set ( agent-identifier :name CortinaS@192.168.0.19:1099/JADE )
)
:content  "((action (agent-identifier :name
CortinaS@192.168.0.19:1099/JADE) (Command :functionality (Functionality
:functionality_inst OnOffFunctionality) :controllable (Controllable
:controllable_inst ShutterActuatorLivingRoom) :command_value On)))"
:language  fipa-sl  :ontology  AssistiveHome  :protocol  fipa-request
)
```

Figura 35: Mensagem enviada pelo agente de Ilumincação para o agente Lâmpada da sala [Fonte: O Autor]

b. Como a lâmpada já estava no estado apagado o agente da lâmpada da sala rejeita a solicitação através da mensagem da figura 36.

```
(REFUSE
:sender  ( agent-identifier :name LampadaS@192.168.0.19:1099/JADE
:addresses (sequence http://leticia:7778/acc ))
:receiver  (set ( agent-identifier :name
IluminacaoS@192.168.0.19:1099/JADE  :addresses (sequence
http://leticia:7778/acc )) )
:reply-with  IluminacaoS@192.168.0.19:1099/JADE1409210232015  :language
fipa-sl  :ontology  AssistiveHome  :protocol  fipa-request
```

Figura 36:Mensagem enviada pelo agente da Lâmpada recusando a solicitação [Fonte: O Autor]

3. Passo 3: Cortina da sala é aberta (Lâmpada para cortina)
   a. A lâmpada correspondente a cortina é acesa conforme Figura 37.

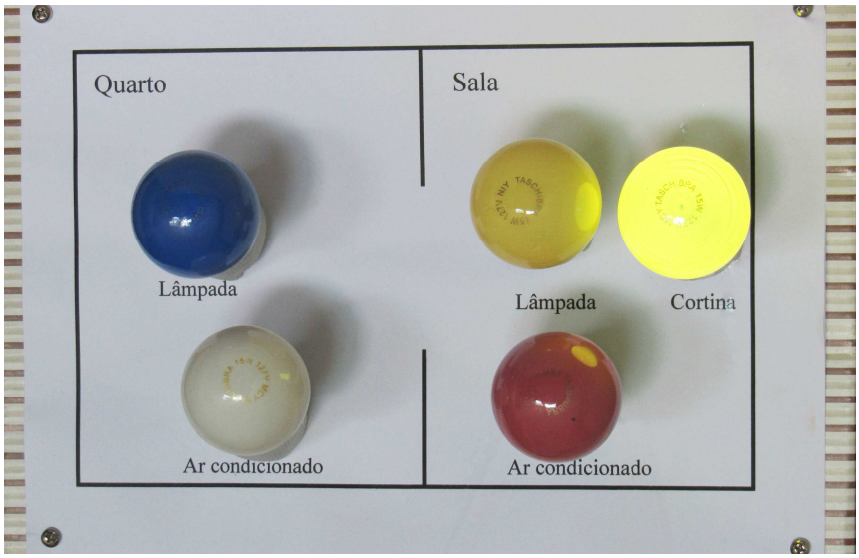

Figura 37:Lâmpada correspondente a cortina é acesa [Fonte: O Autor]

4. Passo 4: Sensor de iluminação Noite é pressionado na sala
   a. Quando o sensor de iluminação é pressionado (Figura 39) o agente de Sensores envia uma mensagem para o agente de iluminação informando da alteração do estado conforme Figura 38.

```
(INFORM
:sender  ( agent-identifier :name SensoresS@192.168.0.19:1099/JADE
:addresses (sequence http://leticia:7778/acc ))
:receiver  (set ( agent-identifier :name
IluminacaoS@192.168.0.19:1099/JADE ) )
:content  "((action (agent-identifier :name
IluminacaoS@192.168.0.19:1099/JADE) (Notification :functionality
(Functionality :functionality_inst StateChangeNotificationFunctionality)
:controllable (Controllable :controllable_inst LightSensorLivingRoom)
:notification_value Noite)))"
:language  fipa-sl  :ontology  AssistiveHome )
```

Figura 38: Mensagem enviada pelo agente de Sensores para o agente de Iluminação[Fonte: O Autor]

Figura 39:Sensor de Iluminação pressionado [Fonte: O Autor]

5. Passo 5: Cortinas fechadas para a sala
    a. Então o agente de iluminação solicita o fechamento da cortina através da mensagem identificada na Figura 40.

```
(REQUEST
:sender  ( agent-identifier :name IluminacaoS@192.168.0.19:1099/JADE
:addresses (sequence http://leticia:7778/acc ))
:receiver  (set ( agent-identifier :name CortinaS@192.168.0.19:1099/JADE )
)
:content  "((action (agent-identifier :name
CortinaS@192.168.0.19:1099/JADE) (Command :functionality (Functionality
:functionality_inst OnOffFunctionality) :controllable (Controllable
:controllable_inst ShutterActuatorLivingRoom) :command_value Off)))"
:language  fipa-sl  :ontology  AssistiveHome  :protocol  fipa-request
)
```

Figura 40: Mensagem enviada pelo agente de Iluminação para o agente que controla a Cortina [Fonte: O Autor]

6. Passo 6: Lâmpada da sala acesa
    a. Da mesma forma que o agente de iluminação solicita o fechamento da cortina ele também solicita ao agente da lâmpada que altera o seu estado para ligado conforme mensagem da Figura 41 e Lâmpada da Figura 42.

```
(REQUEST
:sender  ( agent-identifier :name IluminacaoS@192.168.0.19:1099/JADE
:addresses (sequence http://leticia:7778/acc ))
:receiver  (set ( agent-identifier :name LampadaS@192.168.0.19:1099/JADE )
)
:content  "((action (agent-identifier :name
LampadaS@192.168.0.19:1099/JADE) (Command :functionality (Functionality
:functionality_inst OnOffFunctionality) :controllable (Controllable
:controllable_inst SimpleLampLivingRoom) :command_value On)))"
:language  fipa-sl  :ontology  AssistiveHome  :protocol  fipa-request
)
```

Figura 41: Mensagem enviada pelo agente de Iluminação para o agente da Lâmpada [Fonte: O Autor]

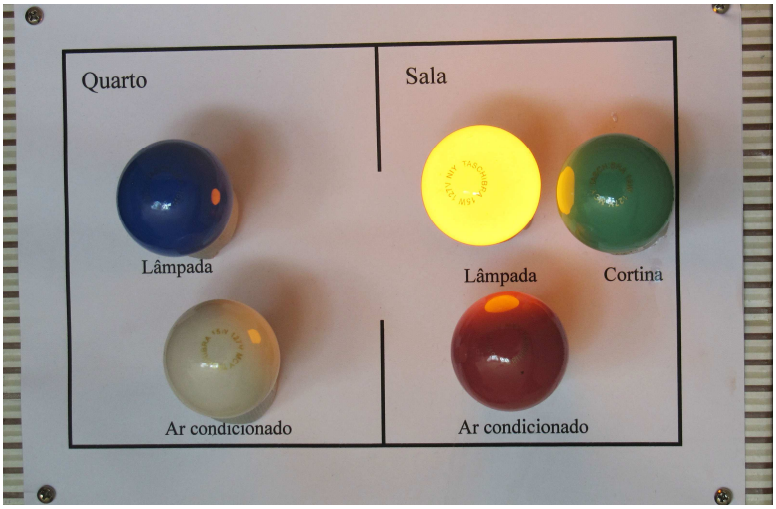

Figura 42:Lâmpada da sala é acessa [Fonte: O Autor]

7. Passo 7: Sensor Temperatura Alta pressionado na sala (Figura 43)
   a. O agente de sensores informa ao agente de controle de temperatura sobre a mudança n ambiente através da mensagem da Figura 44.
   b.

Figura 43: Sensor de temperatura pressionado [Fonte: O Autor]

```
INFORM
:sender  ( agent-identifier :name SensoresS@192.168.0.19:1099/JADE
:addresses (sequence http://leticia:7778/acc ))
:receiver  (set ( agent-identifier :name
TemperaturaS@192.168.0.19:1099/JADE ) )
:content  "((action (agent-identifier :name
TemperaturaS@192.168.0.19:1099/JADE) (Notification :functionality
(Functionality :functionality_inst
TemperatureMeasurementNotificationFunctionality) :controllable
(Controllable :controllable_inst TemperatureSensorLivingRoom)
:notification_value High)))"
:language  fipa-sl  :ontology  AssistiveHome )
```

Figura 44: Mensagem enviada pelo agente de Sensores para o agente de temperatura [Fonte: O Autor]

8. Passo 8: Ar condicionado Ligado para a sala (Figura 45)
    a. Através do protocolo *Request* o agente de controle da temperatura solicita que o agente do ar condicionado altere seu estado através da mensagem da Figura 41.
    b.

```
(REQUEST
:sender  ( agent-identifier :name TemperaturaS@192.168.0.19:1099/JADE
:addresses (sequence http://leticia:7778/acc ))
:receiver  (set ( agent-identifier :name ArS@192.168.0.19:1099/JADE ) )
:content  "((action (agent-identifier :name ArS@192.168.0.19:1099/JADE)
(Command :functionality (Functionality :functionality_inst
OnOffFunctionality) :controllable (Controllable :controllable_inst
AirConditioningSystemLivingRoom) :command_value On)))"
:language  fipa-sl  :ontology  AssistiveHome  :protocol  fipa-request
)
```

Figura 45: Mensagem enviada pelo agente de Sensores para o agente de temperatura [Fonte: O Autor]

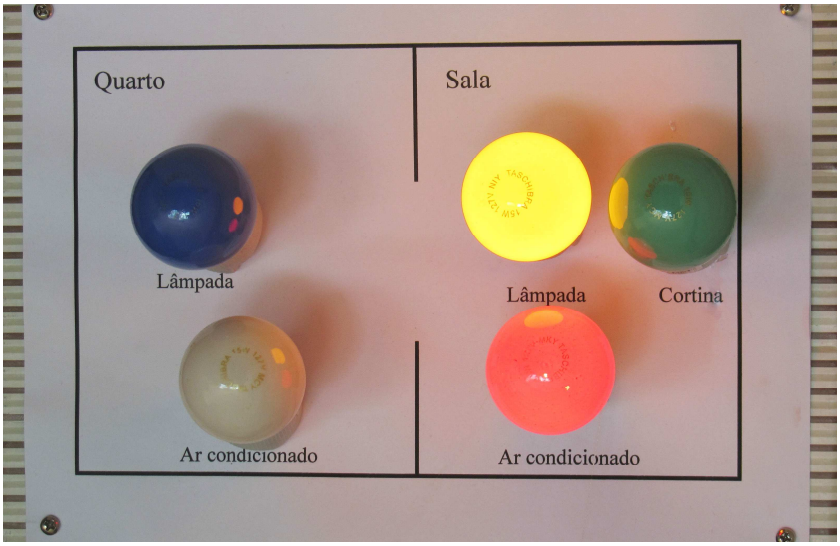

Figura 46:Lâmpada correspondente ao Ar condicionado é acessa [Fonte: O Autor]

9. Passo 9: Sensor de Presença do quarto é ativado (Figura 47)
    a. Quando a presença do individuo é detectada no ambiente do quarto uma mensagem é enviada do agente de sensores para o agente da Lâmpada requisitando a ação de ligar a luz, Figura 48.

Figura 47: Sensor de presença pressionado para o quarto [Fonte: O Autor]

```
(REQUEST
:sender  ( agent-identifier :name SensoresQ@192.168.0.19:1099/JADE
:addresses (sequence http://leticia:7778/acc ))
:receiver  (set ( agent-identifier :name LampadaQ@192.168.0.19:1099/JADE )
)
:content  "((action (agent-identifier :name
LampadaQ@192.168.0.19:1099/JADE) (Command :functionality (Functionality
:functionality_inst OnOffFunctionality) :controllable (Controllable
:controllable_inst SimpleLampBedroom) :command_value On)))"
:language  fipa-sl  :ontology  AssistiveHome  :protocol  fipa-request)
```

Figura 48: Mensagem enviada pelo agente de Sensores para o agente de temperatura [Fonte: O Autor]

10. Passo 10: Lâmpada do quarto é acesa [Figura 49]

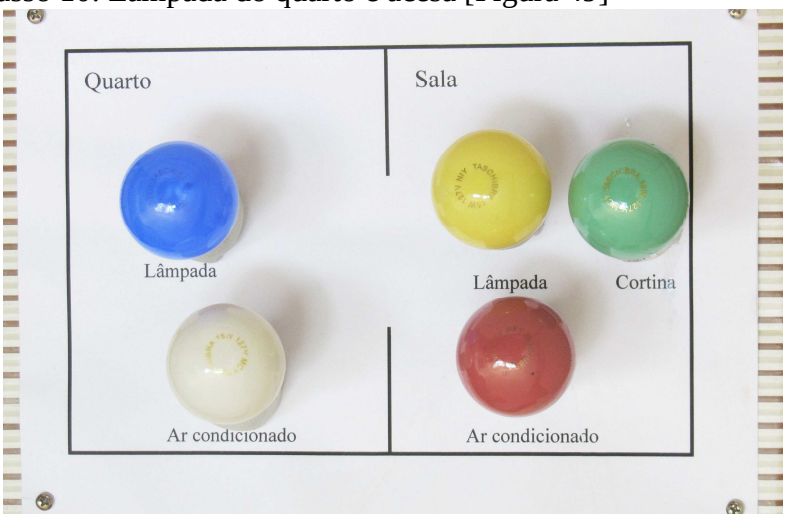

Figura 49: Lâmpada do Quarto acesa [Fonte: O Autor]

11. Passo 11: Sensor de temperatura do quarto altera para temperatura baixa
    a. Quando o sensor de temperatura é alterado o agente de sensores envia uma solicitação para o agente do ar condicionado (Figura 50).
    b.

```
(REQUEST
:sender  ( agent-identifier :name SensoresQ@192.168.0.19:1099/JADE
:addresses (sequence http://leticia:7778/acc ))
:receiver  (set ( agent-identifier :name ArQ@192.168.0.19:1099/JADE ) )
:content  "((action (agent-identifier :name ArQ@192.168.0.19:1099/JADE)
(Command :functionality (Functionality :functionality_inst
OnOffFunctionality) :controllable (Controllable :controllable_inst
AirConditioningSystemBedroom) :command_value On)))"
:language  fipa-sl  :ontology  AssistiveHome  :protocol  fipa-request)
```

Figura 50: Mensagem enviada pelo agente de Sensores para o agente de temperatura [Fonte: O Autor]

12. Passo 12: Ar Ligado no quarto
    a. Finalmente o ar condicionado representado pela lâmpada é acionado para o quarto, de acordo com a Figura 51.

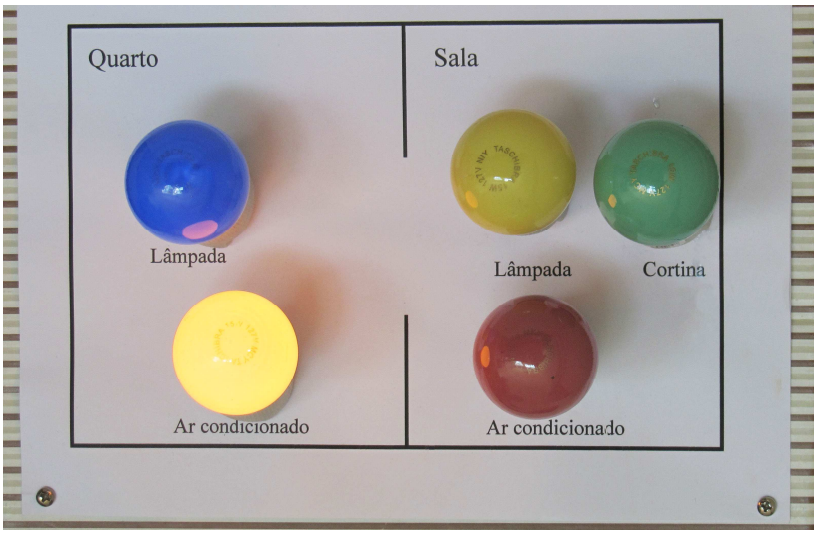

Figura 51: Mensagem enviada pelo agente de Sensores para o agente de temperatura [Fonte: O Autor]

Através da Figura 52 é possível verificar todas as mensagens enviadas durante o caso de teste. Esta figura também permite verificar a ordem das mensagens ao longo do tempo. Assim, verificou-se que o protótipo teve um comportamento adequado. Os dispositivos enviaram as mensagens corretamente e as lâmpadas acenderam e apagaram de acordo com o esperado para cada passo.

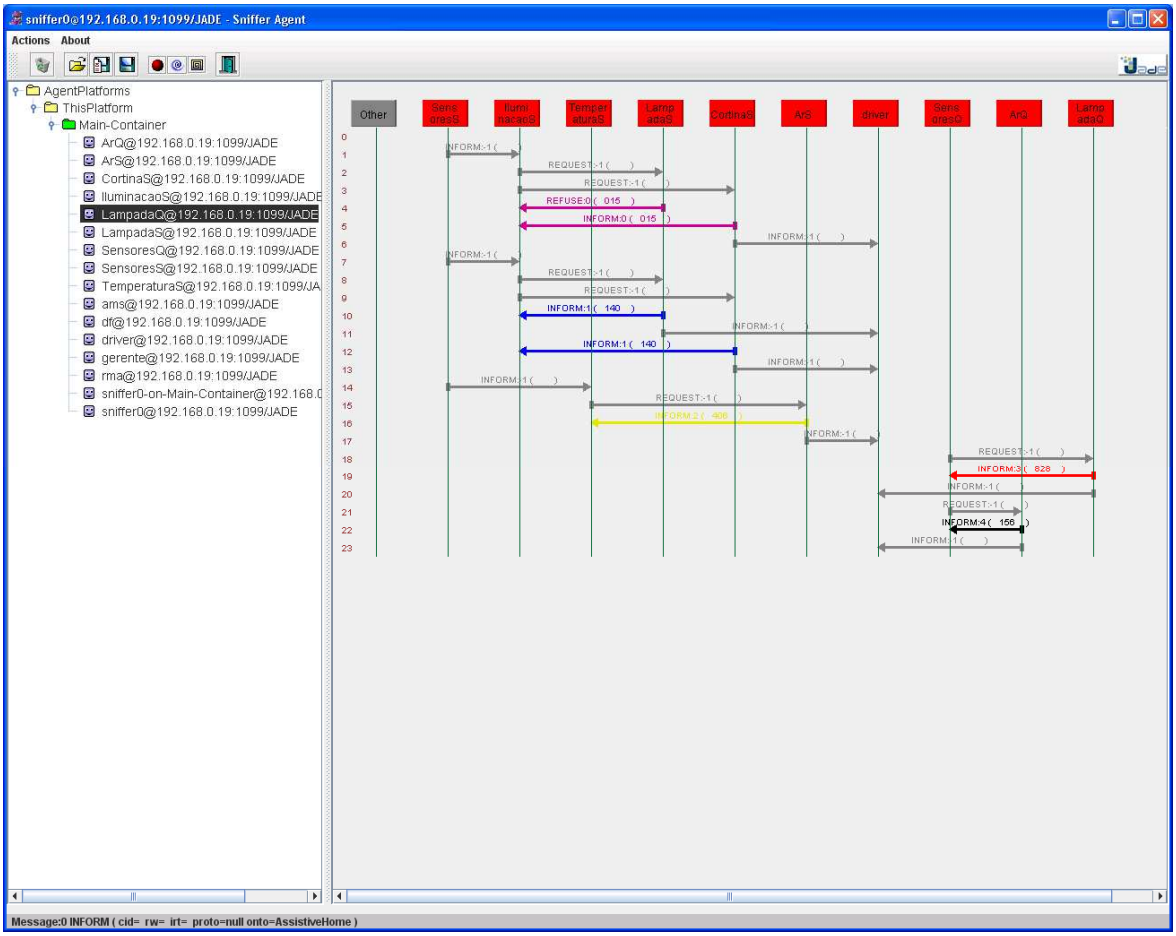

Figura 52: Sniffer ao final da execução do teste com todas as mensagem capturadas [Fonte: O Autor]

Capítulo 6

Resultados

A ontologia criada foi avaliada em três domínios descritos a seguir.

6.1 Consistência semântica

A primeira avaliação verificou a consistência semântica da ontologia e suas propriedades sintáticas. Para esta avaliação foi utilizado um *reasoner* que infere automaticamente sobre as propriedades da ontologia. O *reasoner* escolhido foi o Pellet que é nativo da ferramenta Protégé.

Segundo Parsia (2004), o Pellet suporta os seguintes construtores da linguagem OWL que foi utilizada para representar a ontologia AssitiveHome:

- Restrições de cardinalidade;
- Axiomas complexos;
- Restrições locais de reflexividade;
- Propriedades reflexivas, não-reflexivas, simétricas e assimétricas;
- Propriedades de disjunção;

Uma das tarefas de raciocínio do Pellet é a Classificação que gera, quando possível, relações de subclasse entre as classes da ontologia que não estão explicitas no modelo, inferindo uma hierarquia completa e adequada para responder as consultas de subclasses diretas ou indiretas. A partir dessa nova Classificação o *reasoner* determina as classes as quais uma instância pertence.

Subseqüente a tarefa de Classificação o Pellet realiza duas verificações, sendo uma para avaliação da consistência com objetivo de garantir que a ontologia não possui contradições e uma para verificar a satisfação de conceitos que determina se é possível uma classe possuir instâncias. Como exemplo para a avaliação de consistência, o Pellet pode verificar se as instâncias representam o mesmo indivíduo quando existe uma propriedade funcional. Caso dois indivíduos estejam relacionados por uma propriedade funcional eles não podem pertencer a classes disjuntas.

A Figura 53 apresenta o resultado da execução do *reasoner* Pellet na Ontologia, permitindo visualizar que nenhuma inconsistência foi verificada.

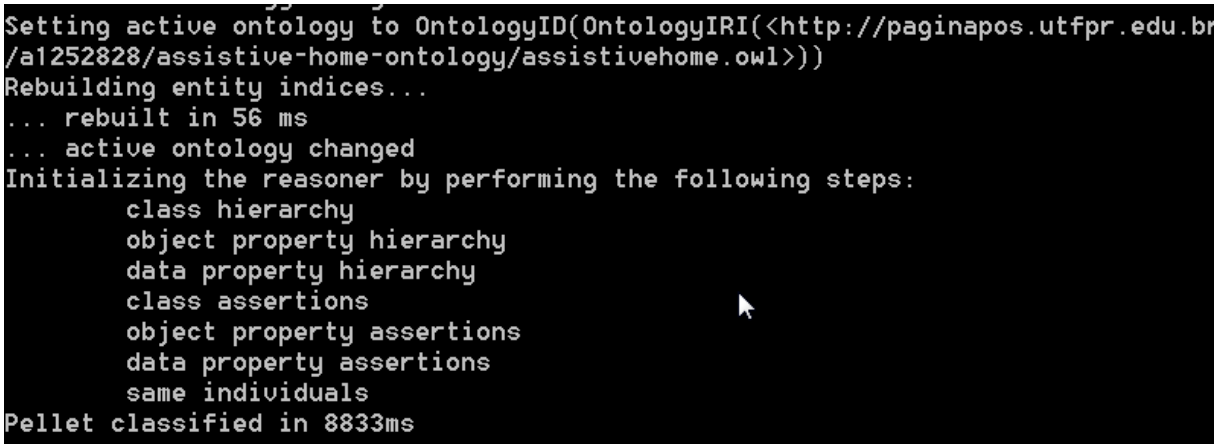

Figura 53: Resoner executado na Ontologia [Fonte: O Autor]

Os motivos de não existirem inconsistências na ontologia advém do fato das classes estarem corretamente ligadas dentro de uma hierarquia pré-determinada e as relações entre as classes, subclasses e indivíduos não possuírem inconsistências em relação ao modelo.

6.2 Avaliação de aplicação prática

A ontologia foi utilizada para modelagem do protótipo presente no capítulo 5 e aplicada nesse contexto da automação dos cinco dispositivos. Durante a execução dos casos de teste foram capturadas as mensagens trocadas entre os agentes, sendo possível verificar que o conteúdo sofreu a mesma interpretação tanto pelo emissor, quanto pelo receptor devido a ontologia compartilhada.

Identificou-se que a ontologia não apresentou inconsistências e efetuou as buscas e relações conforme esperado. Isto permite concluir que a ontologia foi corretamente criada, uma vez que não existiu inconsistência no modelo semântico.

6.3 Comparativo

Para o modelo proposto, seguindo as questões de competência baseadas na classificação de contexto de Abowd et al. (2000), foram definidas oito classes relevantes para representação do domínio de uma residência. São elas: (1) Dispositivos, (2) Ambientes, (3) Localização Absoluta, (4) Pessoas, (5) Funções, (6) Atividades, (7) Agentes de Software, (8) Tempo.

Na Tabela 12, é apresentado uma comparação entre as ontologias descritas no capítulo de trabalhos correlatos, e se elas contemplam ou não as classes determinadas na ontologia desta dissertação.

Através dessa comparação, verifica-se que as únicas ontologias que possuem todas as classes consideradas é a *Smart Space Context* [Qin et al. 2007], e a AssistiveHome.

A ontologia proposta por Qin et al. (2007) possui todas as classes listadas, permite descrever a localização relativa e absoluta de um objeto e contempla as descrições dos perfis de usuário. Porém foi desenvolvida para representar qualquer ambiente inteligente e as classes não possuem subclasses que permitem representar uma residência, todos os dispositivos de automação e as atividades de um habitante. Desta forma ela não pode ser utilizada para interoperabilidade em um ambiente de domótica.

As classes da ontologia AssistiveHome levaram em consideração o reuso e desta forma foi possível integrar domótica em um único modelo as descrições mais adequadas para interoperabilidade em uma residência inteligente.

Tabela 12: Comparação entre as ontologias e as classes contempladas [O autor]

	(1)	**(2)**	**(3)**	**(4)**	**(5)**	**(6)**	**(7)**	**(8)**
Smart Space Context [Qin et al. 2007]	+	+	+	+	+	+	+	+
DomoML [Sommaruga et al., 2011]	+	+	+	+	-	+	-	+
DogOnt [Bonino et al., 2008]	+	+	-	-	+	-	-	-
CDBP [Jacquet et al.,2012]	+	+	-	+	-	-	-	+
Standard ontology [Abdulrazak et al., 2010]	+	+	+	+	+	+	-	+
Ontology-Based Home Service Model [Wei et al,. 2012]	+	+	-	+	+	+	-	-
SOPRANO [Klein et al,. 2007]	+	+	+	+	-	+	-	-
BASont [Ploennigst et al., 2012]	+	+	-	-	+	-	-	-
SOUPA [Chen et al,. 2005]	-	+	-	+	-	+	+	+
Ontologia Proposta	+	+	+	+	+	+	+	+
Legenda: (1) Dispositivos, (2) Ambientes, (3) Localização, (4) Pessoas, (5) Funções, (6) Atividades, (7) Agentes de Software, (8) Tempo								
- não contempla + contempla								

Capítulo 7

Conclusão

O desenvolvimento de ambientes inteligentes é um grande desafio. Analisando-se a Figura 1 (pag 6) percebe-se que o desafio começa logo na primeira camada do modelo que corresponde aos dispositivos. Através da computação móvel e embarcada surgem cada vez mais dispositivos que podem ser utilizados para fornecer algum serviço na residência. Muitos destes serviços consistem de aplicações únicas que exercem apenas uma função sem conectarem-se as demais aplicações do ambiente. Esta característica deve-se principalmente a falta de interoperabilidade entre dispositivos de diversos fabricantes.

A ontologia foi desenvolvida para ser utilizada como um modelo semântico compartilhado entre agentes responsáveis pela automação de dispositivos e controle de funcionalidades. O reuso de trabalhos correlatos garantiu que cada classe incluída no modelo tenha uma grande importância para o domínio e contexto. Diversas ontologias preocupam-se apenas em representar uma parte do contexto da residência e sugerem que os modelos podem ser expansíveis, porém a ontologia AssitiveHome tem como objetivo a especificação de uma ontologia de aplicação que possua significado para a interação. Além disso, a modelagem de classes mais generalistas não agrega valor semântico às aplicações, serve apenas como ponto inicial para modelagem de contexto. Enquanto ontologias completas como a proposta nesta dissertação auxiliam na comunicação tanto do contexto quando do significado semântico.

O modelo também foi aplicado em um protótipo de residência de dois cômodos e um número reduzido de sensores. Através da aplicação foi possível verificar que a ontologia, como forma de representação da residência e agentes, atende ao requisito de um padrão semântico para comunicação e a utilização dos protocolos propostos pela FIPA facilitou o desenvolvimento da solução, já que o JADE oferece suporte para todos os tipos de mensagem.

Finalmente este trabalho contribui para a área de ambientes inteligentes e pervasivos delimitando a ação dos agentes através das funcionalidades e permitindo que o arcabouço comporte-se conforme o esperado, mesmo com a dinâmica destes ambientes.

7.1 Limitações da Pesquisa e Ameaças a Validade do Trabalho

Alguns aspectos não foram abordados pelo arcabouço, como segurança e confiabilidade. Também não foram efetuados testes de eficiência, já que o protótipo inclui um número reduzido de sensores, agentes e atuadores, não sendo possível prever a aplicação em um ambiente com grande número de instâncias.

Outra limitação da validação da ontologia é que os casos de teste não contemplaram todas as classes, pois a AssistiveHome possui 714 classes ao total o que dificulta a aplicação.

Apesar dos agentes de dispositivos não necessitarem de muito processamento, talvez para um ambiente com muitas instâncias da ontologia seja necessária a instalação do agente de gerência em um computador com maior poder computacional.

As instâncias na ontologia são criadas de acordo com a necessidade do ambiente de automação, e a consistência depende do desenvolvedor conhecer a ontologia. Caso contrário o Jena através do *reasoner* Pellet poderá retornar inconsistências em tempo de execução.

7.2 Trabalhos Futuros

Sugere-se futuramente que a ontologia seja estendida para que sejam incluídas regras para inferências mais complexas tanto para as classes de dispositivos quanto para as classes que identificam atividades de uma pessoa.

As classes de pessoa da ontologia leva em consideração apenas alguns aspectos das barreiras funcionais de um indivíduo, um trabalho futuro poderia estender a ontologia para que descreva todas os impactos listados pela CIF [WHO, 2001].

Outra classe que pode ser melhor explorada é a referente ao tempo em relação as demais classes, levando em consideração a concorrência de dispositivos no ambiente.

A dissertação avaliou apenas a ontologia com os trabalhos correlatos, sugere-se que em um trabalho futuro o arcabouço também seja validado e melhorado de acordo com o estado da arte.

A automação foi executada com uma placa desenvolvida pela UTFP conectada a porta paralela, sugere-se que também seja executado testes com dispositivos de automação de outros padrões.

Referências

[Abdulrazak, 2010] Abdulrazak, B., Chikhaoui, B., Gouin-Vallerand, C., & Fraikin, B. (2010). A standard ontology for smart spaces. International Journal of Web and Grid Services, 6(3), 244-268.

[Abowd et al. 2000] Abowd, G. D.; Mynatt E. D. (2000) Charting Past, present and future research in ubiquitous computing. ACM Transactions on Computer-Human Interaction, v. 7, p. 29-58.

[Alam et al., 2012] Alam, M. R., Reaz, M. B. I., & Ali, M. A. M. (2012). A review of smart homes—Past, present, and future. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 42(6), 1190-1203.

[Apache, 2014] Apache Software Foundation. Jena documentation overview.
http://jena.sourceforge.net/. Acessado em: Agosto de 2014.

[Augusto et al., 2010] Augusto, J. C.; Nakashima, H; Aghajan, H. Ambient Intelligence And Smart Environments: (2010) A State Of The Art. In:Handbook Of Ambient Intelligence And Smart Environments. Springer Us,. P. 3-31.

[Beetz et al., 2009] Beetz, J., Van Leeuwen, J., & De Vries, B. (2009). IfcOWL: A case of transforming EXPRESS schemas into ontologies. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 23(01), 89-101.

[Bellifemine et al., 2001] Bellifemine, F., Poggi, A., & Rimassa, G. (2001). Developing multi-agent systems with a FIPA-compliant agent framework. Software-Practice and Experience, 31(2), 103-128.

[Bellifemine et al., 2007] Bellifemine, F. L.; Caire, G.; Greenwood, D. (2007) Developing Multi-Agent Systems With Jade. Wiley.

[Bettini et al,. 2010] Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., & Riboni, D. (2010). A survey of context modelling and reasoning techniques. Pervasive and Mobile Computing, 6(2), 161-180.

[Bluetooth, 2014]. Bluetooth (2014) A Look at the Basics of Bluetooth Technology. http://www.bluetooth.com/Pages/Basics.aspx. Acessado em: Agosto de 2014.

[Bolzani, 2004] Bolzani, C. Residências Inteligentes. São Paulo: Editora Livraria Física, 2004.

[Bonino et al., 2008] Bonino, D.; Corno, F. Dogont-Ontology Modeling For Intelligent Domotic Environments. The Semantic Web-Iswc 2008. Springer Berlin Heidelberg, 2008. P. 790-803.

[Bratman, 1987] Bratman, M. E. (1987). Intention, Plans, and Practical Reason, Cambridge, MA.

[Braubach et al., 2003] Braubach, L., Lamersdorf, W., & Pokahr, A. (2003). Jadex: Implementing a bdi-infrastructure for jade agents. In EXP – In Search of Innovation. Turin, n.3, p.76–85.

[Brickley et al., 2012] Brickley, D.; Miller, L. (2012). FOAF vocabulary specification 0.98. Namespace Document, v. 9,

[Chan et al., 2008] Chan, M.; Estève, D.; Escriba, C.; Campo, E. A Review Of Smart Homes - Present State And Future Challenges. Computer Methods And Programs In Biomedicine, N. 91, P. 55–81, 2008.

[Chen et al., 2004] Chen, H., Perich, F., Finin, T., & Joshi, A. (2004). Soupa: Standard ontology for ubiquitous and pervasive applications. In Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on (pp. 258-267). IEEE.

[Chen et al,. 2008] Chen, D., Vallespir, B., & Daclin, N. (2008). An Approach for Enterprise Interoperability Measurement. In MoDISE-EUS (pp. 1-12).

[Cook et al., 2005] Cook, D. J., Das, S. K. Smart Environments – Technology, Protocol, And Applications. New Jersey: John Wiley & Sons, 2005

[Cook et al., 2007] Cook, D. J. Youngblood, M., Das, S. K. A Multi-Agent Approach To Controlling A Smart Environment. Designing Smart Homes, Lnai 4008, P. 165–182, 2007.

[Dey, 2001] Dey, A. K. (2001). Understanding and using context. Personal and ubiquitous computing, 5(1), 4-7.

[Desbonnet, 1997] Desbonnet, J., & Corcoran, P. M. (1997). System architecture and implementation of a CEBus/Internet gateway. Consumer Electronics, IEEE Transactions on, 43(4), 1057-1062.

[Durfee, et al. 1994]. Durfee, E. H.; Rosenschein, J. S. Distributed Problem Solving And Multi-Agent Systems: Comparisons And Examples. Proceedings Of The International Workshop On Distributed Artificial Intelligence, 1994.

[Dibowski et al., 2010] Dibowski, H., & Kabitzsch, K. (2010). Ontology-based device descriptions and triple store based device repository for automation devices. In Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on (pp. 1-9). IEEE.

[Edwards et al., 2001] Edwards, W. K., & Grinter, R. E. (2001). At home with ubiquitous computing: seven challenges. In Ubicomp 2001: Ubiquitous Computing (pp. 256-272). Springer Berlin Heidelberg.

[Finin et al., 1994] Finin, T., Fritzson, R., McKay, D., & McEntire, R. (1994). KQML as an agent communication language. In Proceedings of the third international conference on Information and knowledge management (pp. 456-463). ACM.

[FIPA, 2002a] Foundation for Intelligent Physical Agents. (2002). FIPA SL Content Language Specification.

[FIPA, 2002b] Foundation for Intelligent Physical Agents. (2002). FIPA Request Interaction Protocol Specification.

[FIPA, 2002c] Foundation for Intelligent Physical Agents. (2002). FIPA Query Interaction Protocol Specification.

[FIPA, 2002d] Foundation for Intelligent Physical Agents. (2002). FIPA Communicative Act Library Specification.

[Guarino, 1998] Guarino, N. (Ed.). (1998). Formal ontology in information systems: Proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy (Vol. 46). IOS press.

[Gómez-Perez et al., 2004] Gomez-Perez, A., Fernández-López, M., & Corcho-Garcia, O. (2004). Ontological engineering. Computing Reviews, 45(8), 478-479.

[Gruber, 1993] Gruber, T. (1993). A translation approach to portable ontology specifications. Knowledge acquisition, 199-220.

[Gruber, 2009] Gruber, T. (2009). Ontology. Encyclopedia of database systems, 1963-1965.

[Hoareau et al,. 2009] Hoareau, C., Satoh I. (2009) "Modeling and processing information for context-aware computing: A survey." New Generation Computing 27.3: 177-196.

[Huhns, 1999]. Huhns, M. N., & Stephens, L. M. (1999). 2 Multiagent Systems and Societies of Agents.

[Jacquet et al.,2012] Jacquet, C., Mohamed, A., & Bellik, Y. (2013). An Ambient Assisted Living Framework with Automatic Self-Diagnosis. International Journal on Advances in Life Sciences, 5(1).

[Jingjing Xu et al. 2009] Xu, J., Lee, Y. H., Tsai, W. T., Li, W., Son, Y. S., Park, J. H., & Moon, K. D. (2009, May). Ontology-based smart home solution and service composition. In Embedded Software and Systems, 2009. ICESS'09. International Conference on (pp. 297-304). IEEE.

[Kagal et al., 2003]. Kagal, L. T. Finin, and A. Joshi. (2003). A Policy Based Approach to Security for the Semantic Web. In 2nd International Semantic Web Conference (ISWC2003).

[Kabilan, 2007] Kabilan, V. (2007). Ontology for information systems (04IS) design methodology: conceptualizing, designing and representing domain ontologies.

[Khawand, 1991] Khawand, C., Douligeris, C., & Khawand, J. (1991). Common Application Language (CAL) and its integration into a home automation system.Consumer Electronics, IEEE Transactions on, 37(2), 157-162.

[Klein et al,. 2007] Klein, M., Schmidt, A., & Lauer, R. (2007). Ontology-centred design of an ambient middleware for assisted living: The case of soprano. InTowards Ambient Intelligence: Methods for Cooperating Ensembles in Ubiquitous Environments (AIM-CU), 30th Annual German Conference on Artificial Intelligence (KI 2007).

[Kovatsch et al., 2010] Kovatsch, M., Weiss, M., & Guinard, D. (2010, September). Embedding internet technology for home automation. In Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on (pp. 1-8). IEEE.

[Labrou, et al., 1999] Labrou, Y.; Finin T.; Peng, Y. (1999). Agent Communication Languages: The Current Landscape. IEEE Intelligent Systems, v.14, n.6, p. 45-52.

[Lassila et al., 1999] Lassila, O., Swick, R. (1999) Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation. http://www.w3.org/TR/REC-rdf-syntax. Acessado em Agosto de 2014.

[Maslow, 1943] Maslow, A. H. (1943). A theory of human motivation. Psychological review,50(4), 370.

[McGuinness et al., 2004] McGuinness, D., van Harmelen, F. OWL - Web Ontology Language Overview, W3C Recommendation. http://www.w3.org/TR/owl-features. Acessado em Agosto de 2014.

[Miori et al., 2010]. Miori, V., Russo, D., & Aliberti, M. (2010). Domotic technologies incompatibility becomes user transparent. Communications of the ACM,53(1), 153-157.

[Motik et al., 2012] Motik, B., Grau, B. C., Horrocks, I., Wu, Z., Fokoue, A., & Lutz, C. (2013). OWL 2 Web Ontology Language Profiles, W3C Recommendation 11 December 2012.

[Noy et al., 2001] Noy, N. F., Mcguinness, D. L. (2001) Ontology Development 101: A Guide To Creating Your First Ontology, Stanford Knowledge Systems Laboratory Technical Report Ksl-01-05 And Stanford Medical Informatics Technical Report Smi-2001-0880.

[OGC, 2011] OGC. (2011) GeoSPARQL - A Geographic Query Language for RDF Data. http://www.opengeospatial.org/standards/geosparql. Acessado em Agosto de 2014.

[Parsia et al., 2004] Parsia, B., & Sirin, E. (2004). Pellet: An owl dl reasoner. In Third International Semantic Web Conference-Poster (Vol. 18).

[Ploennigst et al., 2012] Ploennigs, J., Hensel, B., Dibowski, H., & Kabitzsch, K. (2012, October). BASont-A modular, adaptive building automation system ontology. In IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society (pp. 4827-4833). IEEE.

[Poslad et al., 2000] Poslad, S., Buckle, P., & Hadingham, R. (2000). The FIPA-OS agent platform: Open source for open standards. In Proceedings of the 5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents (Vol. 355, p. 368).

[Qin et al., 2007] Qin, W., Shi, Y., & Suo, Y. (2007). Ontology-based context-aware middleware for smart spaces. Tsinghua Science & Technology, 12(6), 707-713.

[Rathnayaka et al., 2011] Rathnayaka, A. D., Potdar, V. M., & Kuruppu, S. J. (2011). Evaluation of wireless home automation technologies. In Digital Ecosystems and Technologies Conference (DEST), 2011 Proceedings of the 5th IEEE International Conference on (pp. 76-81). IEEE.

[Ramos et al., 2014a] Ramos, L. ; Betini, R. C. (2014). Integration Framework for Assistive Home Based on Ontologies. In: XXIV Congresso Brasileiro de Engenharia Biomédica, 2014, Uberlândia.

[Ramos et al., 2014b] Ramos, L. ; Betini, R. C. (2014). A HCI Vision of Ubiquitous Computing Challenges. In: XIII Simpósio Brasileiro Sobre Fatores Humanos em Sistemas Computacionais, 2014, Foz do Iguaçu.

[Rao, et. al. 1995]. Rao, A.; Georgeff, M. (1995) BDI Agents: from theory to practice. Proceedings of the First International Conference on Multi-Agent Systems, Cambridge, p.312-319.

[Röcker et al., 2014]. Röcker, C., Ziefle, M., & Holzinger, A. (2014). From Computer Innovation to Human Integration: Current Trends and Challenges for Pervasive HealthTechnologies. In Pervasive Health (pp. 1-17). Springer London.

[Stanford, 2014] Stanford. Protégé Desktop User Documentation
http://protegewiki.stanford.edu/wiki/Protege4UserDocs. Acessado em: Agosto de 2014.

[Strang et al., 2004] Strang, T., & Linnhoff-Popien, C. (2004). A context modeling survey. In Workshop Proceedings.

[Sommaruga et al,. 2005] Sommaruga, L.; Perri, A.; Furfari, F. Domoml-Env: An Ontology For Human Home Interaction. In: Swap. 2005.

[Sommaruga et al,. 2011] Sommaruga, L., Formilli, T., & Rizzo, N. (2011, September). DomoML: an integrating devices framework for ambient intelligence solutions. InProceedings of the 6th International Workshop on Enhanced Web Service Technologies (pp. 9-15). ACM.

[W3C, 2004] W3C. (2004) OWL-S: Semantic Markup for Web Services. http://www.w3.org/Submission/OWL-S/. Acessado em: 18/08/2014

[W3C, 2006] W3C. (2006) Time Ontology in OWL. http://www.w3.org/TR/owl-time/ . Acessado em: 18/08/2014.

[W3C, 2013] W3C. (2013) Geolocation API Specification. http://www.w3.org/TR/2013/REC-geolocation-API-20131024/. Acessado em: 18/08/2014.

[Wagh, 2014]. Wagh, S., & Prasad, R. (2014). Maximizing lifetime of wireless sensor networks using genetic approach. In Advance Computing Conference (IACC), 2014 IEEE International (pp. 215-219). IEEE.

[Wang et al., 2004] Wang, X. H., Zhang, D. Q., Gu, T., & Pung, H. K. (2004). Ontology based context modeling and reasoning using OWL. In Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on (pp. 18-22). IEEE.

[Wei et al., 2012] Wei, M., Xu, J., Yun, H., & Xu, L. (2012). Ontology-based home service model. Computer Science and Information Systems, 9(2), 813-838.

[Weiss, 1999] Weiss, G. (1999). Multiagent systems: a modern approach to distributed artificial intelligence. MIT press.

[Weiser, 1991] Weiser, M. The Computer For The 21st Century. Scientific American, V. 265, N. 3, P. 94-104, 1991.

[Wood et al., 2008]. Wood, Stephen R., and Roberto Aiello. Essentials of UWB. New York: Cambridge University Press, 2008.

[Wooldridge, 2001] Wooldridge, M. J. Introduction to Multiagent Systems. New York: John Wiley & Sons Inc, 2001. 347p. 2ed

[WHO, 2001], World Health Organization. International Classification of Functioning, Disability, and Health: ICF. World Health Organization. 2001

[Zahariadis, 2003] Zahariadis, T. B. (2003). Home networking technologies and standards. Artech House. pp. 186