UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

MAURÍCIO FONTANA DE VARGAS

PROPOSTA DE UM SISTEMA DE ASSISTÊNCIA PERSONALIZADA PARA AMBIENTES INTELIGENTES

Porto Alegre
2016

MAURÍCIO FONTANA DE VARGAS

PROPOSTA DE UM SISTEMA DE ASSISTÊNCIA PERSONALIZADA PARA AMBIENTES INTELIGENTES

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.
Área de concentração: Controle e Automação

ORIENTADOR: Prof. Dr. Carlos Eduardo Pereira

Porto Alegre
2016

MAURÍCIO FONTANA DE VARGAS

PROPOSTA DE UM SISTEMA DE ASSISTÊNCIA PERSONALIZADA PARA AMBIENTES INTELIGENTES

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Dr. Carlos Eduardo Pereira, UFRGS

Doutor pela Universidade de Stuttgart – Stuttgart, Alemanha

Banca Examinadora:

Prof. Dr. Leandro Buss Becker, UFSC

Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Prof. Dr. Ivan Müller, UFRGS

Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Prof. Dr. Marcelo Götz, UFRGS

Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil e Universität Paderborn – Alemanha

Coordenador do PPGEE: _____

Prof. Dr. Luís Fernando Alves Pereira

Porto Alegre, abril de 2016.

DEDICATÓRIA

Dedico este trabalho à minha irmã Camila e aos meus pais Neri e Beatriz – minhas maiores inspirações na luta pelos meus sonhos.

AGRADECIMENTOS

Ao Programa de Pós-Graduação em Engenharia Elétrica - PPGEE pela oportunidade de estudo e pesquisa.

Ao orientador Dr. Carlos Eduardo Pereira pelas diversas oportunidades oferecidas, pela confiança depositada e pelas importantes sugestões dadas.

À CAPES pela concessão da bolsa de estudos.

À Homesystems pela disponibilização da infraestrutura necessária para a validação deste trabalho, em especial ao engenheiro de software Miguel dos Santos por todo apoio prestado.

Aos colegas do PPGEE pela parceria, amizade e colaboração, em especial ao Gustavo Rodrigues que sempre esteve disposto a me auxiliar, seja em relação ao LaTeX ou à vida em Porto Alegre.

À minha namorada, Joana Tomaz Tancredo, pela amizade, carinho, incentivo e compreensão nos momentos mais difíceis ao longo desta caminhada. Sou muito grato por todas as palavras e abraços que me ajudaram a manter o foco e equilíbrio quando mais precisei.

Aos meus amigos Gabriel, Júlia, Jacques, Manuela, Giuseppe, Lauren, Rafael e Welter que tornaram minha vida em Porto Alegre muito mais divertida e interessante. Levarei para sempre os momentos que vivemos juntos.

Aos meus sogros, Ramona e Fábio, pelo suporte fundamental para a minha adaptação em Porto Alegre.

Aos meus amigos de Ponta Grossa e Toronto pelo apoio à distância e pelas inesquecíveis visitas e encontros nessa fase da minha vida.

RESUMO

O fenômeno global de envelhecimento populacional está sendo acompanhado pelo rápido crescimento do número de pessoas com algum tipo de deficiência mental ou física, assim como diversas doenças crônicas relacionadas à idade. Devido ao fato de que a maioria dos idosos prefere ficar no conforto de suas residências, é essencial o desenvolvimento de tecnologias que auxiliem os idosos a terem uma vida independente e confortável em seus lares. Em especial, Ambientes Assistidos buscam agir proativamente de forma a auxilar de maneira correta o usuário na execução das suas Atividades de Vida Diária. Devido à grande variabilidade nos níveis de deficiência dos usuários, é fundamental que o Ambiente Assistido seja capaz de fornecer a assistência levando em consideração diversos aspectos relacionados ao usuário, como suas dificuldades, limitações e atividade sendo realizada. Este trabalho propõe um sistema que utiliza uma abordagem baseada nas tecnologias da Web Semântica e no paradigma multiagentes com o objetivo de fornecer assistência personalizada a pessoas idosas e com diversidades funcionais. As características do usuário foram modeladas por uma ontologia OWL, enquanto o mecanismo de *reasoning* responsável por inferir as preferências relacionadas aos serviços do ambiente foi implementada utilizando a linguagem SWRL. Ainda, os componentes presentes na arquitetura proposta são implementados na forma de agentes com funções específicas. Entre as característica inovadoras da proposta estão a incorporação de um modelo padronizado e internacionalmente conhecido para representar as deficiências, dificuldades e limitações dos usuários e a utilização de uma arquitetura distribuída que possui as características inerentes ao paradigma multiagentes, tais como escalabilidade, flexibilidade e dinamismo. A validação do sistema foi realizada através de sua integração com uma Arquitetura Orientada a Serviços que utiliza uma solução comercial de automação residencial e um simulador de Ambientes Inteligentes, e da sua utilização em três casos de usos definidos pelo projeto *AAL Joint Programme*.

Palavras-chave: Ambientes assistidos, tecnologias assistivas, personalização de serviços, automação.

ABSTRACT

The global phenomenon of population aging has been followed by the fast growth in the number of people with some kind of mental or physical disability and chronic illnesses associated with old age. Given the fact that the majority of older adults prefer to stay in the comfort of their own homes, it is crucial to develop technologies that help the elderly to live independently and comfortably in their homes. Specifically, Ambient Assisted Living environments aims to act proactively in order to properly assist the elderly and people with functional diversity to perform their Activities of Daily Living. Since these users tend to be greatly variable in regards to functioning and disability levels, it is essential that the ambient provides personalized assistance based on the user's unique requirements, limitations, preferences, and activity being performed. This work proposes a system that uses an approach based on the Web Semantic technologies and the multiagent paradigm with the goal of providing personalized assistance to the elderly and people with functional diversity. The user's characteristics were modeled using an OWL ontology and a reasoning mechanism responsible to infer the preferences related to the ambient services was implemented using SWRL rules. Furthermore, the proposed architecture components were implemented as agents with specific roles. Among the proposed system innovative features are the incorporation of a standardized and international model used to represent the user's functional and disability information, and the use of a distributed architecture having the multiagent paradigm features, such as scalability, flexibility, and dynamism. The system's validation was accomplished by its integration with a Service Oriented Architecture that uses a commercial automation solution and a smart environment simulator, and by its use in three use cases developed by the *AAL Joint Programme*.

Keywords: Ambient Assisted Living, assistive technologies, service personalization, automation.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

AAL	*Ambient Assisted Living*
AATUM	*Ambient Assistive Technology User Model*
ACC	*Agent Communication Channel*
ACL	*Agent Communication Language*
AMS	*Agent Management System*
API	*Application Programming Interface*
AVD	Atividades de Vida Diária
BPEL	*Business Process Execution Language*
DF	*Directory Facilitator*
DPWS	*Devices Profile for Web Services*
FIPA	*Foundation for Intelligent Physical Agents*
HSNET	*Homesystems Network*
HTTP	*Hypertext Transfer Protocol*
ICF	*International Classification of Functioning, Disability and Health*
JADE	*Java Agent DEvelopment framework*
OMS	Organização Mundial da Saúde
OWL	*Web Ontology Language*
POO	Programação Orientada a Objetos

QR Code	*Quick Response Code*
RDF	*Resource Description Framework*
RFID	*Radio-Frequency Identification*
SMF	*Semantic Matching Framework*
SOA	*Service-Oriented Architecture*
SWRL	*Semantic Web Rule Language*
TCP	*Transmission Control Protocol*
TIC	Tecnologia da Informação e Comunicação
UML	*Unified Modeling Language*
UPnP	*Universal Plug and Play*
URI	*Uniform Resource Identifier*
W3C	*World Wide Web Consortium*
WHO	*World Health Organization*
WHODAS	*WHO Disability Assessment Schedule*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

O envelhecimento da população tem se tornado um fenômeno global devido ao aumento da longevidade e o declínio da taxa de natalidade da sociedade moderna, principalmente em regiões desenvolvidas. Além disso, estudos recentes mostraram que a velocidade do envelhecimento da população mundial tende a aumentar nas próximas décadas. De fato, a estimativa é de que em 2050 a porcentagem da população global com mais de 60 anos seja 22% (LUTZ; SANDERSON; SCHERBOV, 2008). Seguindo a mesma tendência, no Brasil a estimativa é que em 2040 17.5% da população terá mais de 65 anos de idade (LINDA; KENT; MATHER, 2011).

Esse envelhecimento da população será acompanhado pelo rápido crescimento do número de pessoas com algum tipo de deficiência mental ou física, assim como diversas doenças crônicas relacionadas à idade, resultando em diversos desafios ao sistema de saúde para que as infraestruturas físicas possam ser capazes de fornecer saúde e bem estar aos idosos e portadores de necessidades especiais (WORLD HEALTH ORGANIZATION, 2011). Ainda, devido ao fato de que pelo menos 80% das pessoas com mais de 60 anos estão vivendo com algum tipo de doença crônica e 64% dos idosos preferem ficar no conforto de suas residências, é essencial o desenvolvimento de tecnologias que auxiliem os idosos a terem uma vida independente e confortável em seus lares (Centers for Disease Control and Prevention (CDC), 2013).

Durante as ultimas décadas, diversas pesquisas relacionadas ao uso das Tecnologias da Informação e Comunicação (TIC) para o suporte a uma vida independente aos idosos vem chamando a atenção de comunidades e governos ao redor do mundo. Na União Europeia, por exemplo, o *Europe Action Plan for Aging Well* e o *Europe Ambient Assisted Living Joint Program* foram lançados para incentivar o desenvolvimento de produtos, serviços e soluções inovadoras baseados nas TICs para auxiliarem no envelhecimento com

qualidade, aumentando a qualidade de vida, a participação na sociedade e reduzindo os custos relacionados à assistência social e de saúde. Surge assim, o conceito de Ambiente Assistido, também conhecido como Ambiente de Vida Assistida (AAL, do inglês *Ambient Assisted Living*) que pode ser definido como o uso das TICs no cotidiano da população a fim de garantir uma vida independente, confortável e saudável a pessoas idosas e com necessidades especiais - especialmente com problemas cognitivos e que residem sozinhas (LI; LU; MCDONALD-MAIER, 2015).

Indo ao encontro do conceito de Ambiente Assistido, estão os Ambientes Inteligentes - ambientes que são sensitivos, adaptativos e responsivos às necessidades do usuário. Em outras palavras, os Ambientes Inteligentes são capazes de identificar as necessidades do usuário e, com isso, agir proativamente de forma a auxilar de maneira correta o usuário na execução das suas Atividades de Vida Diária (AVD) (DASIOS et al., 2015). Para que isso seja possível, os Ambientes Inteligentes utilizam recursos relacionados a diversas áreas, tais como Redes de Comunicação, Sensores, Interface Homem-Máquina e Inteligência Artificial, além do paradigma conhecido como Computação Ubíqua, termo introduzido por (WEISER, 1991) que refere-se à onipresença (ubiquidade) e transparência da tecnologia no cotidiano das pessoas.

Em seu trabalho, Weiser descreveu a proliferação de dispositivos em diversas escalas, desde dispositivos pessoais portáteis até grandes *displays* compartilhados. Essa proliferação de fato ocorreu e pode ser observada em diversos dispositivos como eletrodomésticos, *smartphones* e sistemas de navegação por satélite. Porém, para que a computação torne-se totalmente transparente ao usuário e os ambientes possam agir de forma proativa em situações específicas de forma a fornecer a assistência necessária, é fundamental fornecer a informação/serviço “certa” no momento “certo” e da maneira “certa” (FISCHER, 2001).

Em relação aos Ambientes Assistidos, essa questão é ainda mais importante devido à grande variedade nos tipos e níveis de deficiências de seus usuários. Assim, o fornecimento de serviços personalizados capazes de proverem assistência adaptadas de acordo com as preferências, necessidades e desejos do usuário é essencial (KADOUCHE et al., 2008).

1.1 Objetivos

O principal objetivo deste trabalho é a criação de um sistema capaz de prover assistência personalizada para usuários de Ambientes Inteligentes - com um foco especial em idosos

e pessoas com diversidades funcionais - agindo de forma proativa e de acordo com as preferências e necessidades do usuário em situações específicas.

Dentre as principais características que esse sistema deve suportar estão:

Sensibilidade ao contexto: capacidade de interpretar e armazenar os dados provenientes dos sensores do ambiente com a finalidade de deduzir as informações que caracterizam a situação do usuário ou do ambiente.

Personalização: os serviços disponíveis no ambiente são adaptados de forma a atender às necessidades particulares do usuário.

Independência de tecnologia: o sistema independe da maneira em que as informações do ambiente são coletadas, possibilitando a utilização de diferentes tecnologias e dispositivos de acordo com a necessidade de cada ambiente.

Escalabilidade e flexibilidade: auto-adaptação em relação à inserção e remoção de dispositivos, serviços e funcionalidades do sistema em tempo de execução.

Integração: sistemas externos podem ser utilizados para a realização de funcionalidades específicas, como o aprendizado automático de preferências e a dedução de contexto do usuário utilizando diferentes técnicas de Inteligência Artificial.

Para que o objetivo principal do trabalho seja alcançado, foram definidos os seguintes objetivos específicos:

1. Desenvolver um modelo de usuário capaz de representar corretamente as características relacionadas às limitações, condições de saúde e preferências de um indivíduo. Ainda, esse modelo deve ser capaz de representar preferências relacionadas ao contexto do usuário, isto é, sua localização e atividade sendo realizada.

2. Criar um mecanismo capaz de inferir automaticamente um conjunto de preferências relacionadas a serviços que tem um impacto significativo na realização das AVD do usuário. Para que as preferências sejam corretamente inferidas, esse mecanismo deve levar em consideração as limitações funcionais e o contexto para cada usuário.

3. Propor e implementar uma arquitetura distribuída, flexível e escalável que defina a interação entre o modelo de usuário, o mecanismo de inferência e a infraestrutura orientada a serviços presente no ambiente.

4. Integrar o sistema proposto com a Arquitetura Orientada a Serviços (SOA, do inglês *Service-Oriented Architecture*) presente em um Ambiente Inteligente, onde essa arquitetura é capaz de receber requisições e gerenciar a forma em que os serviços serão executado de acordo com os dispositivos presentes no momento da requisição.[1]

5. Demonstrar o funcionamento do sistema através de sua aplicação em um conjunto de casos de uso padronizado definido pelo *AAL Joint Programme* (EICHELBERG; RöLKER-DENKER; HELMER, 2014)

Como grande parte dos usuários que necessitam de um ambiente assistido residem sozinhos, este trabalho não leva em consideração a possibilidade de múltiplos usuários interagindo simultaneamente com o ambiente. Ainda, as informações pessoais de cada usuário devem ser inseridas no sistema anteriormente a sua utilização de fato.

1.2 Organização da dissertação

Esta dissertação está estruturada da seguinte maneira: o capítulo 2 apresenta os principais trabalhos no âmbito de personalização de serviços em Ambientes Assistidos. O capítulo 3 apresenta os principais conceitos relacionados a este trabalho, como as tecnologias da Web Semântica e sistemas multiagentes. Em seguida, o capítulo 4 descreve com detalhes o Sistema de Assistência Personalizada proposto enquanto o capítulo 5 discute como a proposta foi implementada. Finalmente, o capítulo 6 apresenta a validação da proposta através de sua utilização em três casos de uso e o Capitulo 7 expõe as conclusões obtidas e discute algumas possibilidades de continuação deste trabalho.

[1] A forma em que os serviços são descritos e a estratégia de busca de serviços não fazem parte do escopo desse trabalho. Maiores informações relacionadas à Arquitetura Orientada a Serviços podem ser encontradas em (PEDROSO, 2016)

2 ANÁLISE DO ESTADO DA ARTE

Este capítulo apresenta uma análise dos principais trabalhos relacionados à personalização de serviços em Ambientes Assistidos. Inicialmente é apresentado um modelo de referência para Ambientes Assistidos que define os principais conceitos utilizados neste trabalho e em seguida são apresentados e comparados diversos modelos de usuário utilizados como forma de representar as informações pertinentes aos usuários de Ambientes Inteligentes. Por fim, são mostrados as principais arquiteturas utilizadas em plataformas e sistemas que possibilitam a disponibilização de serviços personalizados ao usuário.

2.1 O modelo de referência *universAAL*

Os autores em (TAZARI et al., 2012) propuseram o modelo de referência chamado *universAAL* para ser utilizado como base no entendimento das principais questões relacionadas aos Ambientes Assistidos e assim facilitar o consenso no processo de construção de infraestruturas ligadas aos Ambientes Assistidos. Esse modelo é uma combinação consolidada de trabalhos anteriores e não propõe uma abordagem completamente nova, mas sim a integração de conceitos já empregados em diversos projetos da área (MARINC et al., 2011). O modelo *universAAL* é apresentado graficamente através uma série de mapas de conceitos: em um primeiro momento a definição de Serviço AAL é apresentada; em seguida o conceito de Ambiente Assistido é definido; por fim, o conceito de plataforma e *middleware* para Ambientes Assistidos é apresentado.

2.1.1 Serviços AAL

A Figura 1 ilustra a definição de serviço em um Ambiente Assistido e mostra os diferentes interesses e questões que os indivíduos ou organizações podem ter ao fornecer ou ao utilizar esses serviços.

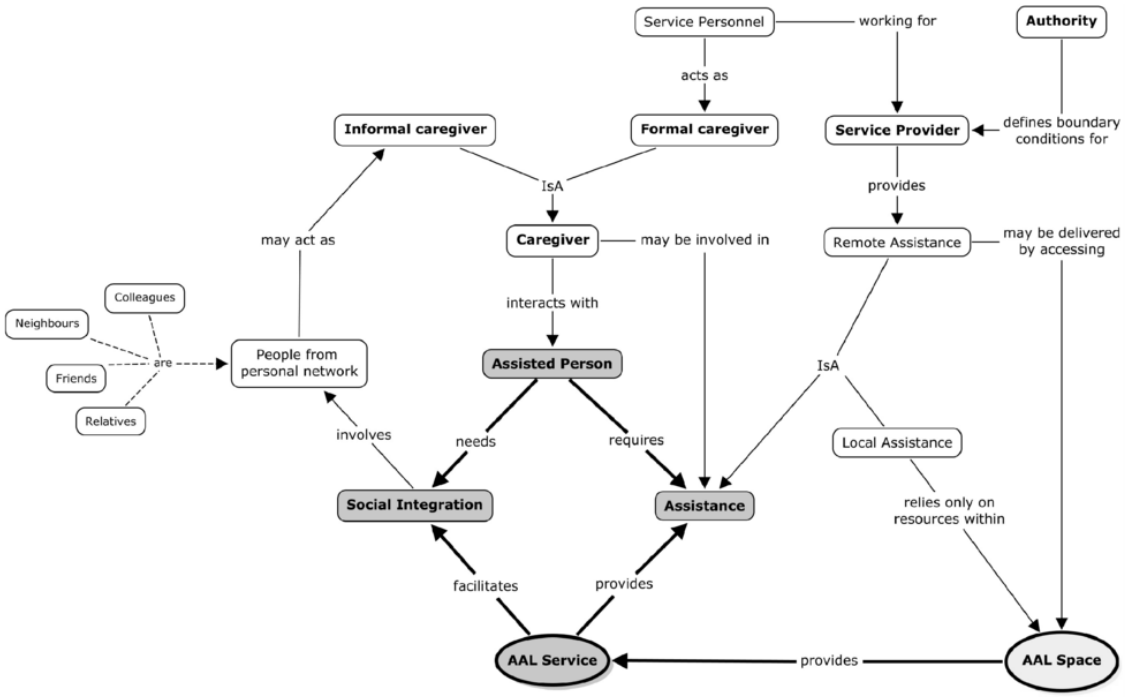


Figura 1: Definição do conceito de serviço no contexto dos Ambientes Assistidos. Fonte: (TAZARI et al., 2012)

Um Serviço AAL (*AAL Service*) foca no bem-estar e no conforto de seus usuários, como por exemplo no fornecimento de assistência necessária em situações específicas e garantindo a integração social em situações de solidão e isolamento. A Pessoa Assistida (*Assisted Person*) é portanto o beneficiário final dos Serviços AAL e suas necessidades são os principais conceitos considerados durante a etapa de projeto desses serviços. Assistência (*Assistance*) no contexto dos ambientes assistidos é definido como o fornecimento de ajuda para uma dificuldade encontrada, seja através da utilização de recursos locais do ambiente assistido (*Local Assistance*) ou envolvendo fornecedores de serviços externos (*Remote Assistance*). Integração social (*Social Integration*) envolve as pessoas relacionadas à pessoa assistida, como familiares, amigos, vizinhos e colegas, que podem agir como Cuidadores Informais (*Informal Caregiver*) durante o fornecimento de assistência à pessoa assistida. Integração social pode ser entendida como o sentimento de pertencer e de ser útil à sociedade. Um Cuidador (*Caregiver*) é uma pessoa que aceitou envolver-se na prestação de serviços AAL, na interação com a pessoa assistida e que pode necessitar acessar os recursos presentes no ambiente assistido. O Cuidador Formal (*Formal Caregiver*) é definido como um cuidador que é pago por estar envolvido na prestação de serviços ou então estar envolvido profissionalmente (voluntário especializado, por exemplo). Assistên-

cia Local baseia-se apenas nos recursos disponíveis no interior do Ambiente Asssistido, normalmente relacionados à automação fornecida pelos componentes/dispositivos instalados no ambiente. Assistência remota (*Remote Assistance*) refere-se a um fornecedor de serviços. Um Fornecedor de Serviços (*Service Provider*) oferece seus recursos humanos e tecnológicos que, do ponto de vista do Ambiente Assistido, são considerados recursos externos. Finalmente, as fronteiras para o desenvolvimento, implantação e fornecimento dos Serviços AAL pelos dos Fornecedores de Serviços são definidas pelas Autoridades (*Authority*).

2.1.2 Ambiente Assistido

Uma vez que o conceito de Serviço AAL foi definido, é possível definir um Ambiente Assistido - Ambiente Inteligente centrado nos seus usuários humanos (*Person*) que necessitam de assistência (Figura 2). O desenvolvimento de Ambientes Inteligentes requer uma abordagem multidisciplinar envolvendo técnicas e métodos de diversos campos, como a Computação Sensível ao Contexto, Interação Homem-Máquina, Inteligência Artificial e Computação Ubíqua/Pervasiva. Esta ultima refere-se à existência de dispositivos com capacidades computacionais e de comunicação incorporados de forma transparente nos objetos físicos do cotidiano. Esses dispositivos (*Networked Artifacts*) podem comunicar entre si a fim de trocarem informações e serviços e especialmente para fornecerem canais (*Channel*) que ligam o mundo físico (*Physical World*) e o mundo virtual/lógico (*Virtual Realm*). Esses canais podem ser classificados como canais sensitivos (fornecidos pelos sensores), canais atuadores (fornecidos pelos atuadores), canais de entrada (fornecidos por microfones, teclados, etc.) ou canais de saída (fornecidos por *displays*, alto-falantes, etc.). O mundo virtual amplia o escopo em termos de sensitividade e responsividade para que a noção de um Ambiente Inteligente faça sentido e seja capaz de servir aos humanos presentes no ambiente. Esse arranjo resulta em características como sensibilidade ao contexto (*Context-awareness*), personalização (*Personalization*), reatividade (*Reactivity*) e proatividade (*Proactivity*) em Ambientes Inteligentes. Em outras palavras, um Ambiente Inteligente deve fornecer suporte para agir de uma maneira personalizada e de acordo com o contexto para que as respostas do ambiente possam ser consideradas adaptativas. A qualidade dessas quatro características centrais podem ser aperfeiçoadas se algum mecanismo de raciocínio (*Reasoning*) for utilizado.

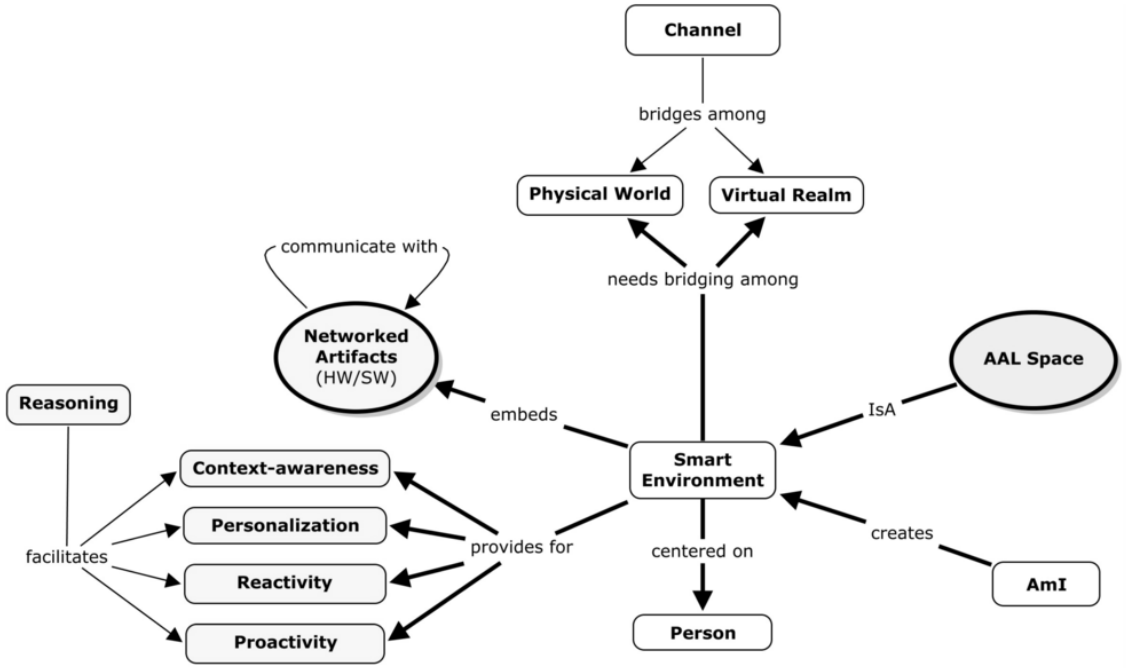


Figura 2: Definição do conceito de Ambiente Assistido. Fonte: (TAZARI et al., 2012)

2.1.3 Plataforma e *middleware* AAL

A Figura 3 descreve o conceito de um *middleware*, que é o conceito central no desenvolvimento de plataformas para Ambientes Assistidos (*AAL Platform*). O *middleware* pode ser entendido como um componente de software (*Software Component*) que idealmente está presente em todos os nós encontrados na rede (*Networking-enabled node*). O *middleware* fornece uma interface comum que facilita a integração (*Integration*) de outros componentes de software e a comunicação (*Communication*) entre eles. Como o *middleware* esconde a distribuição (*Distribution*) e a heterogeneidade (*Heterogeneity*) dos nós presentes na rede, os desenvolvedores dos componentes de software não precisam se preocupar com questões relacionados à localização dos recursos necessários. Em outras palavras, um *middleware* é um software que encontra-se no centro de plataformas de sistemas distribuídos e é responsável pela integração dos componentes de software distribuídos em diferentes nós da rede, além de facilitar a comunicação entre eles. Para isso, um *middleware* encobre os aspectos relacionados à distribuição do sistemas e qualquer heterogeneidade de seus nós.

2.2 Modelos de usuário

Estar ciente das diferentes caraterísticas dos usuários participando de situações onde a personalização de serviços e recursos se faz necessária é essencial para alcançar o objetivo

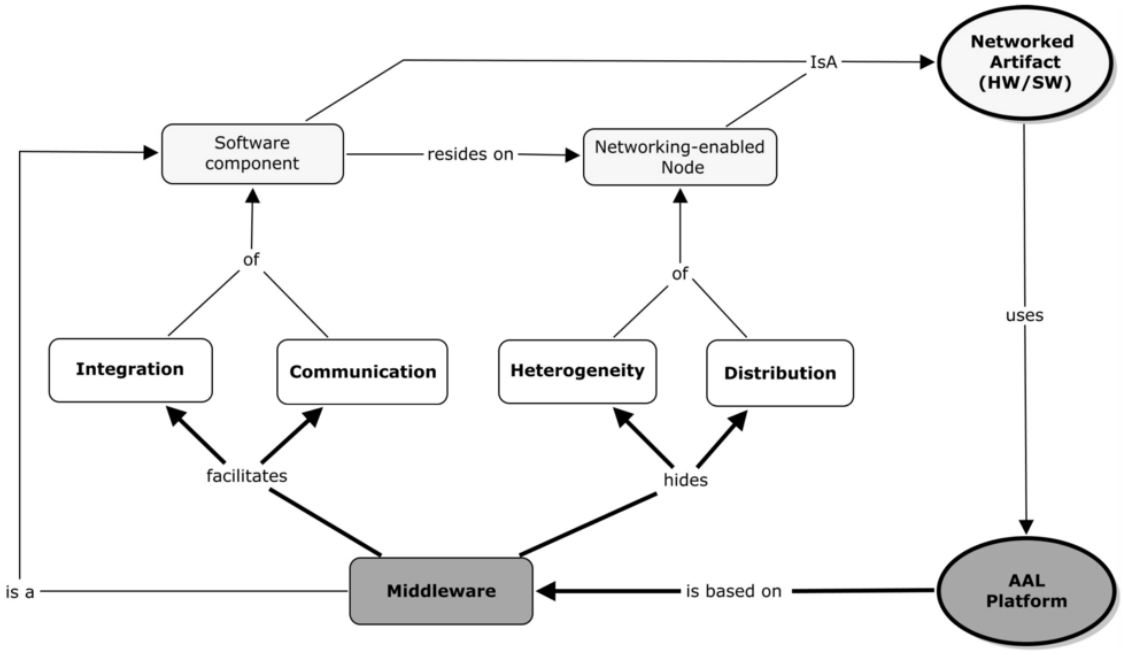


Figura 3: Definição do conceito de plataforma e *middleware* para Ambientes Assistidos. Fonte: (TAZARI et al., 2012)

principal dos Ambientes Assistidos de forma eficiente. Para armazenar diferentes tipos de informação do usuário e de seu contexto, é necessário desenvolver uma série de modelos formais que auxiliam os desenvolvedores a organizar e a associar significados às informações coletadas. Devido aos diversos benefícios inerentes à Web Semântica (discutidos no capítulo 3), a comunidade acadêmica tem proposto diversos modelos de usuário baseados em ontologias. Nesta seção são analisadas algumas das soluções encontradas na literatura para modelar usuários e contextos considerando diferentes abordagens no domínio dos Ambientes Inteligentes.

2.2.1 Modelo de usuário genérico

Os autores em (GOLEMATI et al., 2007) propuseram um modelo de usuário genérico, compreensível e expansível, projetado utilizando a literatura existente, aplicações e ontologias relacionadas com o domínio de criação de perfis de usuários. O modelo proposto, implementado na forma de uma ontologia, pode ser estendido através da herança e adição de novas classes, assim como a instanciação de novas entidades dependendo das necessidades da aplicação em específico. Como resultado, ela pode ser utilizada para a representação de perfis esteriótipos (também conhecido como “Personas”), ou seja, perfis que representam uma categoria especifica de usuário como “mulher” ou “especialista

em computação" e perfis individuais. Porém, a ontologia apresenta apenas informações estáticas e permanentes. Características dinâmicas como a posição atual do usuário não estão incluídas.

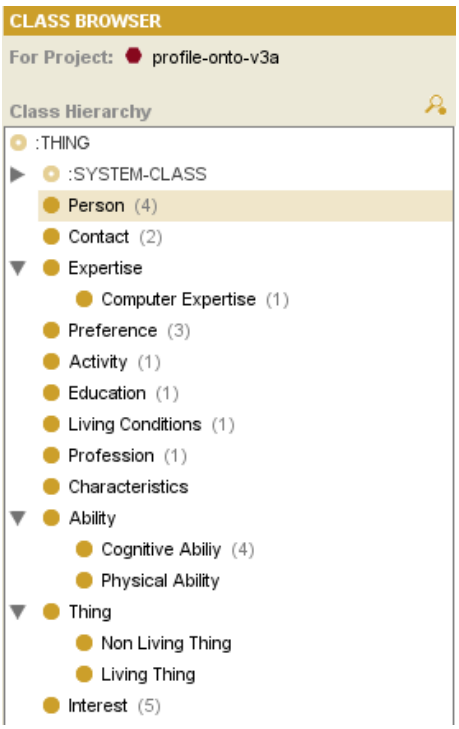


Figura 4: Estrutura hierárquica da ontologia proposta por (GOLEMATI et al., 2007)

A Figura 4 apresenta a hierarquia das principais classes dessa ontologia. A classe *Person* é o centro da ontologia, uma vez que ela contém todas as características do perfil do usuário como o nome e data de nascimento. O restante das classes são utilizadas para descrever as características complexas do usuário. *Living conditions*, *Contact*, *Education*, *Expertise*, *Activity* e *Profession* incluem um conjunto de propriedades descrevendo os respectivos aspectos da vida do usuário assim como o período de tempo em que aquele aspecto é válido. *Interest*, *Preference*, *Ability*, *Characteristic* e *Thing* contém apenas três propriedades: *type*, *name* e *score*. No caso de *Interest*, a propriedade *interest type* foi adicionada para permitir a criação de hierarquias de interesses.

Para representar o nível de habilidade do usuário em uma determinada atividade, foi criada a classe *Expertise*, onde a escolha das características que podem ser utilizadas como fatores ou indicadores da habilidade, assim como a definição e as medidas dos níveis de habilidade são especificas da aplicação em que a ontologia será utilizada.

2.2.2 Modelo do usuário e seu ambiente

Em (KADOUCHE et al., 2008), foi proposto o SMF *(Semantic Matching Framework)* para ser utilizado em ambientes onde a principal função é fornecer serviços de assistência de acordo com as capacidades e preferências do usuário. Para isso, foi utilizado o conceito de *"handicap situation"*, que indica que a incapacidade não é intrínseco do usuário, mas representa as inconveniências do ambiente em relação às capacidades do usuário.

A principal funcionalidade do SMF é baseada na correspondência semântica entre os modelos do usuário e do ambiente, definidos através da ontologia SMF (Figura 5). O modelo do usuário é caracterizado por suas informações, preferências e capacidades enquanto o modelo do ambiente descreve os dispositivos existentes e as suas características. Por exemplo, o dispositivo "porta" poderia ter os atributos "força necessária para abrir" e "tamanho". Esses fatores são quantificados a fim de formalizar a relação entre os atributos do usuário e do ambiente, o que por sua vez define o conceito de *"handicap situation"* para cada usuário em suas atividades cotidianas. Através de um processo de inferência o SMF analisa as capacidades do usuário e as características do ambiente e então deduz todas as *"handicap situations"*. Para isso o motor de inferência utiliza um conjunto de regras definidas através de lógica de primeira ordem envolvendo o usuário e o ambiente.

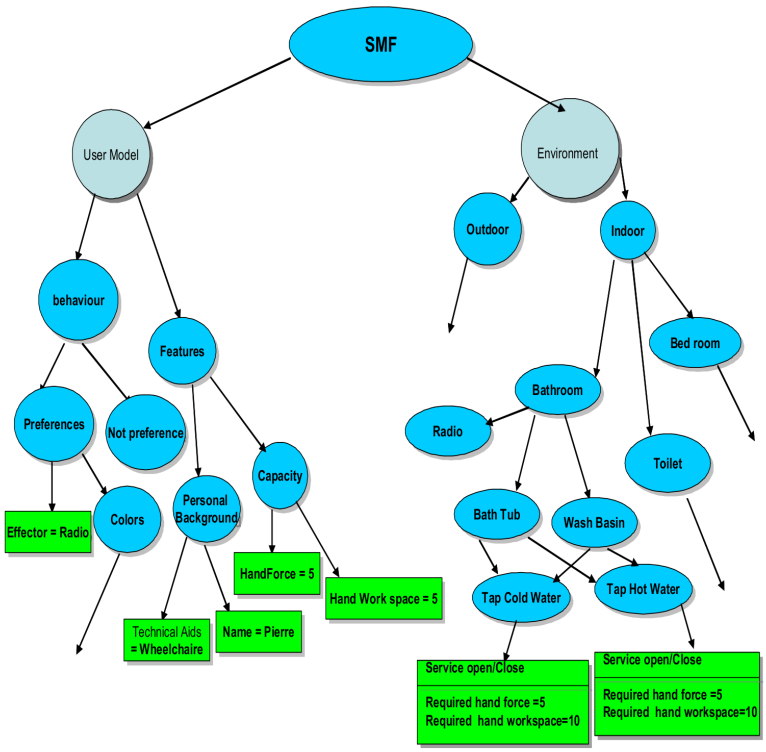


Figura 5: Ontologia SMF proposta por (KADOUCHE et al., 2008)

2.2.3 Utilização de preferências relacionadas às dificuldades do usuário

Outra abordagem para a modelagem de usuários foi apresentada em (CASAS et al., 2008), onde foi utilizado conceito de "Persona" que refere-se a uma representação genérica de um grupo de pessoas. Os autores estabeleceram um conjunto de 10 "Personas", cada um contendo uma série características como idade, profissão, educação e necessidades do usuário.

O diferencial na estratégia utilizada é o fato de não considerar as capacidades do usuário, mas sim suas necessidades decorrentes de suas deficiências físicas e cognitivas. Para isso, o perfil de usuário é composto pelo (i) nível de facilidade do usuário em utilizar o sistema, (ii) tipo de interface de entrada e saída a ser utilizada, (iii) características relacionadas ao áudio e (iv) características relacionadas à exibição, descritos em detalhe a seguir:

- *User Level*: são utilizados quatro níveis: não-capaz (0), indicando que o usuário está temporariamente incapaz de utilizar o sistema; fácil (1), padrão (2) e avançado (3) indicando o nível de entendimento do sistema. Esse entendimento inclui diferentes aspectos como o conhecimento da funcionalidades e possibilidades do sistema, habilidades tecnológicas, perda de memória, entre outros.

- *Interface*: indica a maneira em que o usuário irá interagir com o sistema. A entrada pode ser na forma de voz, que é a maneira de comunicação mais natural para pessoas com capacidades cognitivas reduzidas ou baixa afinidade com a tecnologia, na forma de tato (telas sensíveis ao toque, teclados, etc.) e interfaces sensoriais como biosensores e rastreamento dos olhos. Em relação a saída de dados, as opções disponíveis são na forma gráfica, textual e sonora.

- *Audio*: contém características ligadas ao áudio reproduzido pelo sistema, como volume e frequência, aspectos fundamentais que podem auxiliar usuários com problemas auditivos e visuais.

- *Display*: inclui os ajustes mais comuns utilizados nas telas: contraste, brilho e configuração de cores. Essas configurações podem ajudar pessoas com deficiências visuais a interagir com as telas dos dispositivos presentes no ambiente.

2.2.4 Ontologia com preferências dependentes de contexto

De maneira similar, o UPOS *(User-Profile Ontology with Situation-Dependent Preferences Support)* (SUTTERER; DROEGEHORN; DAVID, 2008) é uma ontologia que aborda aspectos estáticos e sensíveis ao contexto, onde são representadas informações sobre o usuário, preferências relacionadas a diferentes situações, assim como sua localização e suas atividades. Essa ontologia permite a criação de um subconjunto de perfis dependentes do contexto. De acordo com uma condição, que inclui o usuário (Bob), um operador (e.g. igual) e um valor do contexto (e.g. Escritório), o processo de correspondência procura encontrar um sub perfil contendo todas as indicações para a personalização dos serviços relacionados a essa condição. Assim, quando o sistema inferir que o usuário Bob encontra-se em seu escritório, o perfil contendo o serviço personalizado “bloquear SMS” será utilizado, por exemplo.

2.2.5 Modelo de usuário para computação móvel

Em (SKILLEN et al., 2012) foi apresentado um método para a criação de perfis de usuário através do uso de uma ontologia que considera os aspectos estáticos e dinâmicos do usuário, além das suas capacidades, a fim de modelar as suas mudanças comportamentais. O foco nos aspectos dinâmicos torna o modelo flexível, podendo ser utilizado em diversos ambientes e aplicações, garantido assim a versatilidade do modelo do usuário. A Figura 6 apresenta a ontologia proposta.

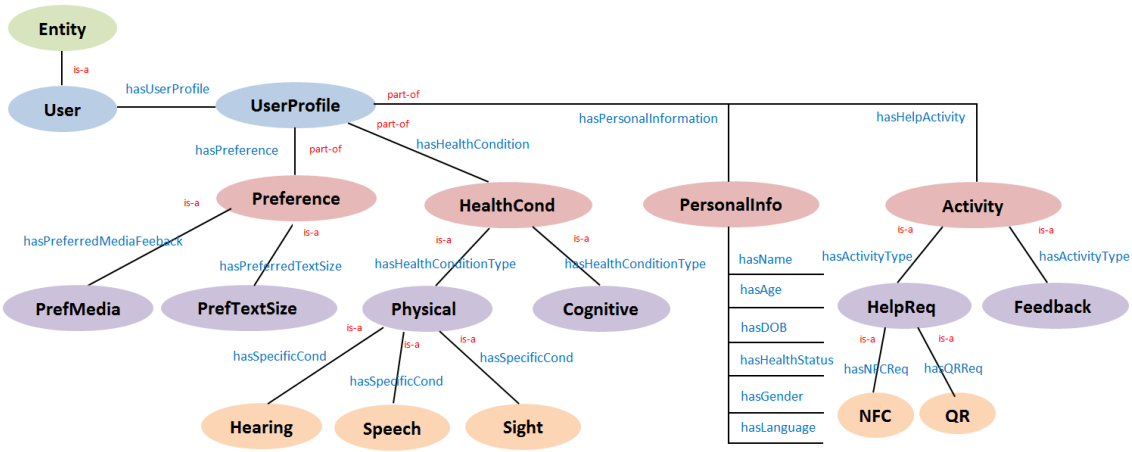


Figura 6: Ontologia de modelo de usuário proposta por (SKILLEN et al., 2014)

Assim como nos outros trabalhos apresentadas até o momento, esta ontologia contém aspectos para representar as informações básicas do usuário, como nome, idade e preferên-

cias. As capacidades do usuário, definidas através dos níveis no qual o usuário tem uma habilidade física, emocional ou cognitiva a fim de realizar uma tarefa ou atividade, tem um papel fundamental na ontologia definida e são utilizadas para determinar o melhor plano de ação em diferentes cenários. Um exemplo é a substituição da interface gráfica de um *smartphone* por comandos de voz ou gestos a fim de adequar-se as condições do usuário portador de deficiência visual. Além disso, a atividade sendo requisitada pelo usuário e a forma em que isso acontece - por meio de um sensor NFC ou QR Code - é representada na ontologia

Dando continuidade a esse trabalho, os autores em (SKILLEN et al., 2014) propuseram um método combinando a modelagem do perfil do usuário, mecanismos de personalização baseados em regras e arquitetura orientada a serviços para disponibilizar serviços *Help on-Demand* personalizados para usuários móveis em ambientes pervasivos. A flexibilidade do método é alcançada através da utilização da personalização inteligente dos serviços utilizando um mecanismo de inferência baseado em regras.

2.2.6 Ontologia para Ambientes Assistidos

Outro trabalho focado na criação de perfis de usuários é o de (FREDRICH; KUIJS; REICH, 2014), parte do projeto PCEICL (*Person Centered Environment for Information, Communication and Learning*). Novamente o usuário é modelado através de uma ontologia, onde o usuário é o conceito central da plataforma e é descrito por suas propriedades como condição de saúde, capacidades, preferências e ambiente social, como visto na Figura 7. Ainda, através da ontologia criada, é possível obter um histórico das condições do usuário, tornando possível o acompanhamento e a análise da evolução da aprendizagem ou das condições de saúde do usuário, por exemplo.

A maior limitação neste trabalho é a não adaptação dinâmica da ontologia em tempo de execução. Assim, uma vez criado, o perfil do usuário não será alterado de acordo com as informações de contexto capturadas, o que seria interessante em diversos casos. Um exemplo é a mudança dos interesses do usuário de acordo com as solicitações de busca em seu navegador ou o horário em que deseja configurar seu despertador de acordo com o horário em que foi dormir na noite anterior.

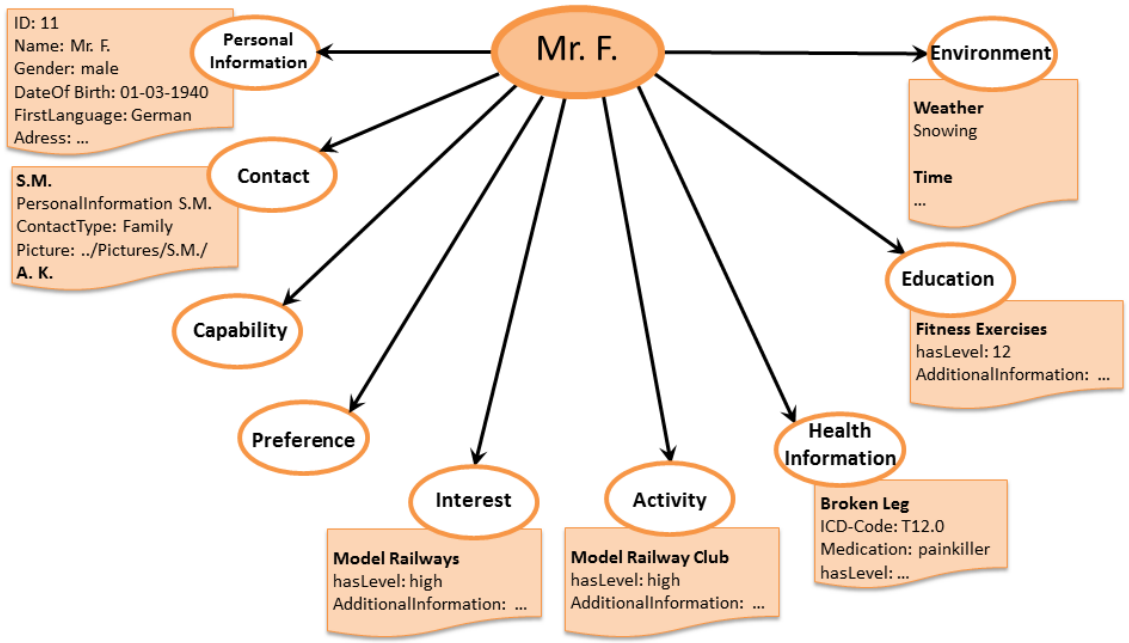


Figura 7: Exemplo de um perfil de usuário representado na ontologia proposta por (FREDRICH; KUIJS; REICH, 2014)

2.2.7 Comparação entre os modelos de usuário para ambientes inteligentes

Dois grandes problemas foram identificados após a análise do estado da arte sobre os modelos de usuário ter sido realizada: primeiramente, nenhum dos trabalhos buscou a utilização de uma representação uniforme e consistente das condições de saúde, limitações e deficiências do usuário. Essas informações são normalmente representadas utilizando um modelo projetado para atender às necessidades de uma aplicação em específico, dificultando o seu reuso em outros sistemas. Como o fornecimento de assistência personalizada em um Ambiente Assistido deve ser realizado de acordo com as limitações e capacidades do usuário, a utilização de um modelo adequado e padronizado internacionalmente mostra-se fundamental. O segundo problema é que nenhum dos modelos encontrados é capaz de representar a informação necessária para responder a perguntas extremamente importantes no âmbito dos Ambientes Assistidos, como:

- *Como o usuário com dificuldades de audição deve ser notificado enquanto ele está dormindo?*

- *Como a luz do ambiente deve ser ajustada quando um usuário com dificuldades de visão está assistindo a um filme?*

Isso acontece devido ao fato de que nenhum dos modelos apresenta maneiras de representar informações relacionadas às dificuldades e limitações do usuário em um contexto específico ao mesmo tempo que é capaz de representar as preferências relacionadas àquele mesmo contexto. Para que isso seja possível, as características hachuradas na Tabela 1 devem ser unificadas em uma única ontologia - que também deve utilizar um modelo padronizado para modelar as condições de saúde, dificuldades e limitações, e assim solucionar o primeiro problema supracitado.

Tabela 1: Comparação entre os modelos de usuário para Ambientes Inteligentes encontrados na literatura

Proposta	Modelo do contexto	Preferência do usuário	Condições de saúde
2.2.1	-	Pessoais	Cognitivas e físicas (não padronizadas)
2.2.2	Ambiente e seus dispositivos	Não especificado	Na forma de capacidades
2.2.3	-	Relacionadas às necessidades	-
2.2.4	Localização e atividade do usuário	Relacionadas ao contexto	-
2.2.5	Localização do usuário	Relacionadas às necessidades	Cognitivas e físicas (não padronizadas)
2.2.6	Informações do ambiente	Pessoais e relacionadas aos serviços do sistema	ICD *(International Classification of Diseases)*

2.3 Arquiteturas e *middlewares* para Ambientes Assistidos

Esta seção apresenta algumas das arquiteturas para Ambientes Assistidos voltadas à disponibilização de serviços personalizados aos seus usuários.

2.3.1 INHOME

O projeto INHOME (VERGADOS, 2010) tem como meta principal proporcionar maneiras de melhorar a qualidade de vida das pessoas idosas através de tecnologias genéricas para o gerenciamento dos ambientes domésticos, incluindo eletrodomésticos, equipamentos voltados ao entretenimento e sistemas de automação residencial, aumentando assim a autonomia e a segurança de seus usuários. Uma vasta gama de serviços são oferecidas pela plataforma INHOME, como monitoramento médico e controle dos eletrodomésticos.

A arquitetura da plataforma INHOME, apresentada na Figura 8, é composta por

diversos sub-sistemas, cada um realizando funções complementares para o gerenciamento de informação de maneira inteligente e transparente. Nas camadas inferiores encontram-se os canais de comunicação físicos, o *gateway* para os dispositivos da residência e o ambiente de execução de serviços. Logo acima estão as interfaces M2M (*machine-to-machine*), as políticas de acesso, as interfaces homem-máquina, os perfis de usuário e dos dispositivos e o mecanismo de personalização de serviços. No penúltimo nível estão os componentes responsáveis pelo monitoramento de atividades, operações implícitas da rede de comunicações, monitoramento e manutenção remoto e assistência médica remota. Finalmente, no nível superior encontram-se as aplicações AAL que utilizam os serviços oferecidos pela camada inferior. Os números de 1 a 8 presentes na Figura 8 mostram as dependências entre os componentes da arquitetura.

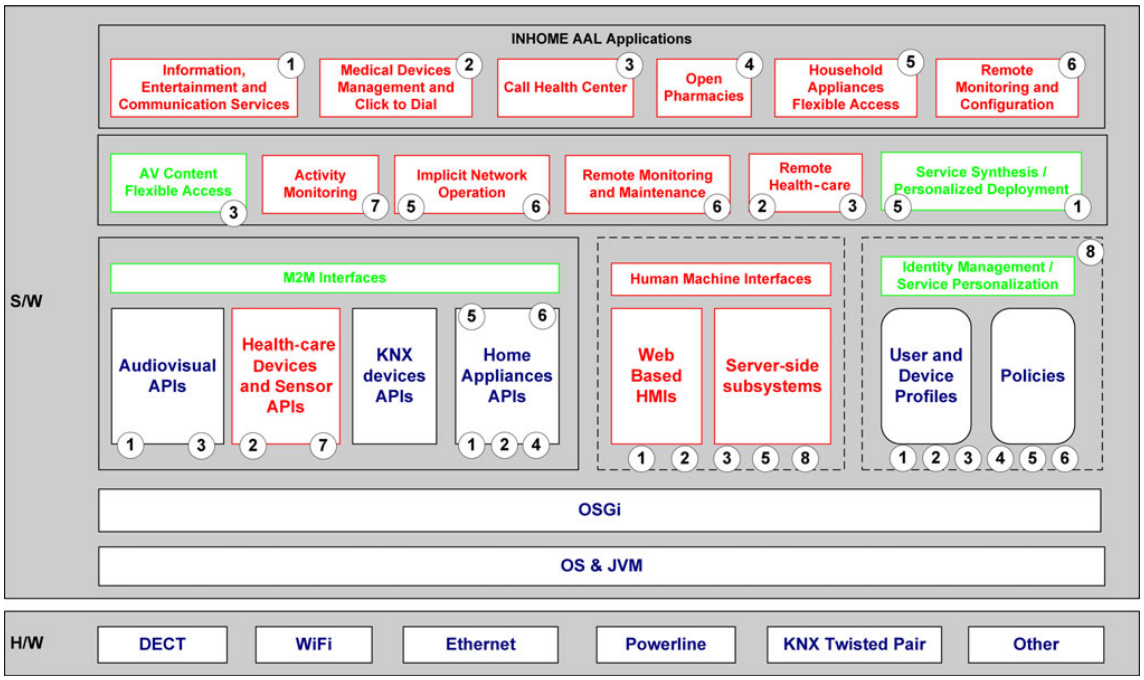


Figura 8: Arquitetura da plataforma INHOME. Fonte: (VERGADOS, 2010)

2.3.2 Plataforma *openAAL*

O *openALL* (WOLF et al., 2010) é um flexível e poderoso *middleware* para Ambientes Assistidos, criado em uma iniciativa conjunta entre centros de pesquisa e empresas de software alemãs. A plataforma openAAL permite a simples implementação, configuração e fornecimento de serviços personalizados e dependentes de contexto através de uma arquitetura baseada na tecnologia OSGi que permite a fácil integração e comunicação entre os serviços. Acima da camada base OSGi encontram-se os três principais componentes

da plataforma: o *Context Manager*, *Procedural Manager* e o *Composer*, como ilustra a Figura 9. A seguir esses componentes são descritos

***Context Manager*:** Fornece uma ontologia para representar as informações dos sensores e entradas do usuário. Sua infraestrutura interna facilita o uso de diferentes algoritmos para a inferência de situações de interesse. Especificamente, o *framework openAAL* suporta o processo de inferência de contexto desde um modelo de baixo nível baseado em sensores até um modelo de alto nível orientado a serviços. O modelo de baixo nível representa os estados dos dispositivos do ambiente, enquanto o modelo de alto nível foca no ambiente onde o sistema é utilizado e fornece um vocabulário para definir as preferências, localização e atividades do usuário.

***Procedural Manager*:** gerencia e executa as rotinas para reagir em situações de interesse. Essas rotinas são especificadas em termos de processos BPEL (*Business Process Execution Language*) estendidos com construtores que permitem a comunicação síncrona e assíncrona com o *Context Manager*. De maneira geral, essas rotinas definem a reação do sistema à certas situações identificadas pelo *Context Manager* e podem ser utilizadas para resolver situações criticas ou informar alguma pessoa. A requisição de serviços são especificadas de maneira abstrata e independe dos softwares e hardwares disponíveis no ambiente.

***Composer*:** O *Composer* analisa os serviços disponíveis no ambiente para então selecioná-los e combiná-los de forma a atingir as requisições de serviço realizadas pelo *Procedural Manager*. Internamente, os mecanismos para composição dos serviços são baseados no *framework* DSD (DIANE Service Description) e utilizam a noção de serviços virtuais para realizarem a ligação entre as requisições abstratas de serviços e os objetivos concretos do serviço requisitado.

2.3.3 Arquitetura para Ambientes e objetos inteligentes (S^{3}OiA)

Os autores em (VEGA-BARBAS et al., 2012) propuseram uma arquitetura composta por três níveis chamada S^{3}OiA (*Smart Spaces and Smart Objects interoperability Architecture*). O principal objetivo desta arquitetura é a integração de dispositivos e aplicativos heterogêneos, além da composição e adaptação dinâmica de aplicativos voltados a Ambi-

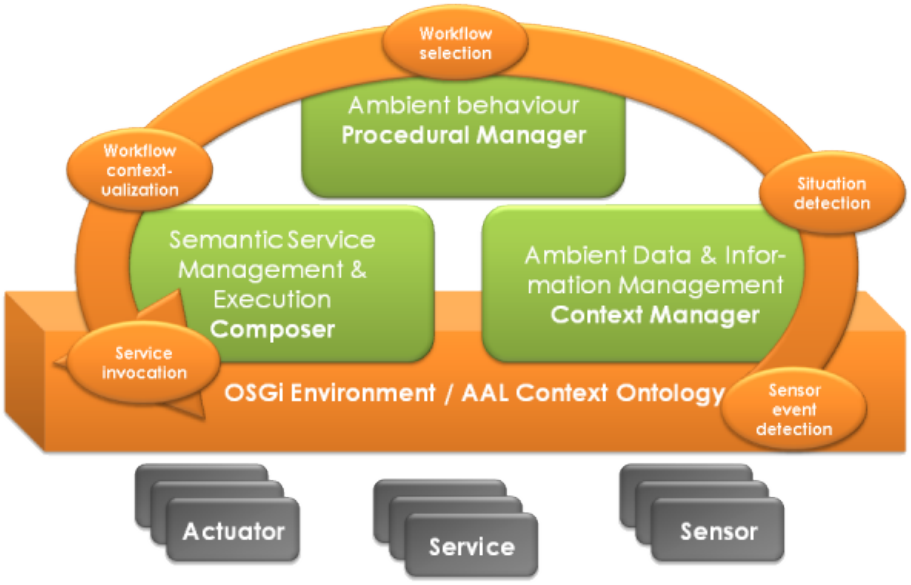


Figura 9: Arquitetura da plataforma *openAAL*. Fonte: (WOLF et al., 2010)

entes Assistidos de acordo com o contexto onde esses aplicativos são executados. A Figura 10 apresenta a arquitetura proposta, descrita a seguir.

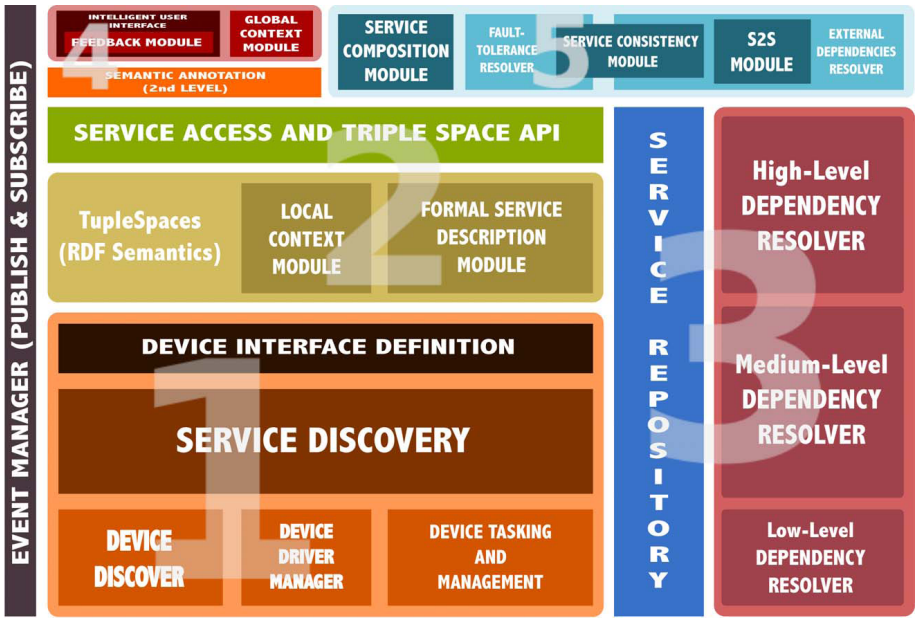


Figura 10: Arquitetura S^{3}OiA. Fonte: (VEGA-BARBAS et al., 2012)

1. Descobrimento de dispositivos e serviços (*Device and Service Discovery*): grupo de módulos responsáveis por integrar e abstrair todos os dispositivos, independentemente de seus protocolos de comunicação. Por exemplo, é possível que um dispositivo DPWS comunique-se com um dispositivo UPnP através de um barramento em comum oferecido por ambos *gateways*.

2. Descobrimento de dispositivos e serviços (*Semantic Triple Spaces and Web Service Exposition*): esse conjunto de módulos fornece a capacidade de comunicação utilizando triplas RDF, onde a informação trocada é representada por um sujeito, um predicado e um objeto. Utilizando esse paradigma é possível disponibilizar as informações conhecidas pelo sistema para grupos de nós similares e sensíveis ao contexto através de uma API compatível com os padrões Web.

3. Repositório de serviços e resolução de dependências (*Service Repository and Dependencies Resolution*): esse grupo de módulos gerencia os serviços disponíveis dentro de um Ambiente Inteligente e resolve as dependências extraídas da composição das aplicações no contexto local. Além disso, é responsável pelo gerenciamento de eventos utilizando o paradigma publicador/assinante. Dessa maneira, quando um novo serviço torna-se disponível ou deixa de existir, os outros módulos são notificados. Esse repositório deve ser concebido como uma instância de armazenamento acessível por todos os módulos da arquitetura.

4. Interface de interação (*Interaction Interface*): grupo de módulos responsável por gerenciar a interação entre o Ambiente Inteligente e o usuário. A principal função desses módulos é a obtenção de informações sobre as intenções do usuário capazes de serem convertidas em uma sequencia de passos a fim de compor aplicações complexas ou adequar as aplicações já existentes às necessidades do usuário.

5. Composição, tolerância a falhas e dependências distantes (*Composition, Fault Tolerance and Distant Dependencies*): presente no ultimo nível da arquitetura, assim como a interface de interação, estão os módulos responsáveis pela composição e orquestração de serviços. Ainda, esses módulos gerenciam o acesso aos recursos do Ambiente Inteligente quando dois ou mais nós externos à rede os requisitam simultaneamente.

3 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo apresentar alguns conceitos fundamentais necessários para o entendimento desse trabalho. Inicialmente é feita uma breve apresentação sobre a Web Semântica e então dois de seus principais conceitos são explanados: as ontologias e o processo de inferência utilizando regras. Por fim, o paradigma multiagentes é apresentado e a plataforma utilizada para a sua implementação é brevemente descrita.

3.1 Web Semântica

O termo Web Semântica, introduzido em (BERNERS-LEE et al., 2001) refere-se à extensão da Web atual onde toda informação é representada através de seu significado (semântica) bem definido, permitindo assim que computadores e pessoas possam trabalhar em cooperação. Uma vez que todo conteúdo presente na Web Semântica possui significados atribuídos, a busca por dados para serem usados em contextos específicos pode ser realizada de maneira simples e direta (SHADBOLT; HALL; BERNERS-LEE, 2006). Atualmente, a Web Semântica é padronizada pela World Wide Web Consortium (W3C) e é definida por uma arquitetura baseada em uma pilha, como pode ser observado na Figura 11.

Nas camadas inferiores estão as tecnologias atualmente presentes e conhecidas na Web como URI (*Uniform Resource Identifier*), que permite a identificação única de um recurso e o padrão UNICODE que garante uma representação padronizada de textos. Ainda, a linguagem XML permite a criação de documentos estruturados em um formato que pode ser compreendido tanto por máquinas quanto por humanos.

Nas camadas intermediárias estão as tecnologias padronizadas pela W3C e que permitem o desenvolvimento de aplicações, como o caso deste trabalho. O RDF permite a representação de modelos de dados através de declarações chamadas de “triplas RDF”, na

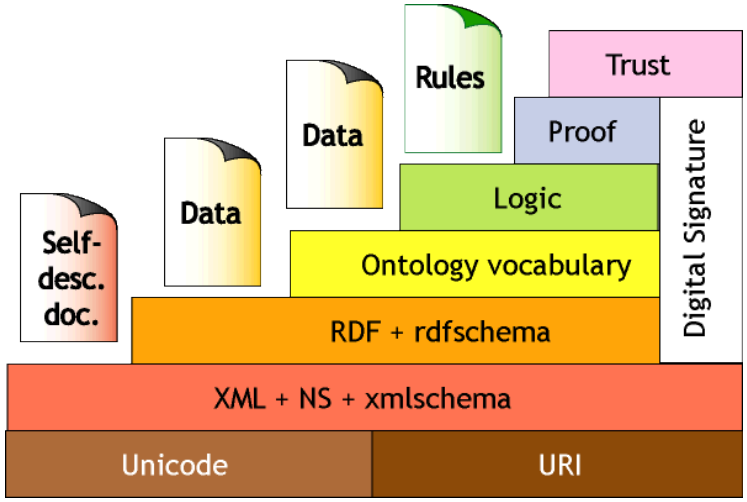


Figura 11: Arquitetura da Web Semântica. Fonte: (BERNERS-LEE, 2000)

forma sujeito-predicado- objeto (LASSILA; SWICK, 1999). Por exemplo, a sentença "o autor do Documento1 é John Smith" é representada como o sujeito sendo "Documento1", o predicado sendo "tem autor" e o objeto "John Smith".

Logo acima estão as camadas de Ontologia, utilizada para representar as informações semânticas de uma maneira estruturada e de Lógica, utilizada para inferir novas relações a partir das informações representadas na ontologia. Como essas duas camadas possuem grande importância neste trabalho, elas são explicadas detalhadamente nas próximas seções.

3.1.1 Ontologias

Originalmente, o termo "Ontologia" vem da filosofia, onde era utilizado para descrever a natureza da existência. No contexto da gestão de conhecimento, diversas definições foram dadas nas ultimas décadas, destacando-se a de (GRUBER, 1993): uma ontologia é uma especificação formal e explícita de uma conceitualização compartilhada.

"Conceitualização" refere-se ao modelo abstrato de algum fenômeno no mundo a partir da identificação de conceitos relevantes desse fenômeno. O termo " explícito" significa que o tipo do conceito usado e as restrições no seu uso são definidas explicitamente. Por exemplo, no domínio médico, dois possíveis conceitos são doenças e sintomas, as relações entre eles são causais e uma restrição é que uma doença não pode ser causada por ela mesmo. "Formal" refere-se ao fato de que a ontologia deve ser legível por máquinas, o que exclui a utilização de linguagens naturais. "Compartilhado" reflete a ideia de que uma ontologia captura conhecimentos consensuais, ou seja, que não é privada a algum

individuo mas sim aceita por um grupo (STUDER; BENJAMINS; FENSEL, 1998).

Ainda, de acordo com (NOY; MCGUINNESS et al., 2001) uma ontologia define um vocabulário comum para pesquisadores e desenvolvedores que necessitam compartilhar informação em um determinado domínio, incluindo definições interpretáveis por máquinas de conceitos básicos do domínio e as relações entre eles.

Entre os diversos motivos para o desenvolvimento de modelos baseados em ontologia, pode-se citar (WANG et al., 2004):

- Compartilhamento de conhecimento: o uso de ontologias permite entidades computacionais, como agentes e serviços em ambientes de computação pervasiva, terem um conjunto comum de conceitos sobre o contexto enquanto interagem entre si.

- Inferência Lógica: Baseado em uma ontologia, a computação sensível a contexto pode explorar vários mecanismos de inferência lógica existentes para deduzir contextos com alto nível de abstração a partir de contextos de baixo nível, i.e, dados fornecidos pelos sensores, e verificar e corrigir inconsistências no contexto devido a falhas na aquisição desses dados.

- Reuso de conhecimento: Através do reuso de ontologias de diferentes domínios, é possível a composição de uma ontologia mais extensa. Por exemplo, modelos para diversos domínios necessitam representar a noção de tempo. Essa representação inclui a noção de intervalos de tempo, pontos no tempo, medidas relativas de tempo, e assim por diante. Se um grupo de pesquisadores desenvolver essa ontologia em detalhe, outros pesquisadores podem simplesmente reusa-la para os seus respectivos domínios. Ainda, é possível a criação de uma ontologia mais complexa a partir da integração de diversas ontologias existentes descrevendo partes do domínio (NOY; MCGUINNESS et al., 2001).

Frequentemente a criação da ontologia de um domínio não é o objetivo final a ser alcançado. O desenvolvimento de uma ontologia é semelhante a definir a estrutura de um conjunto de dados a ser utilizado por outros sistemas. Quando dados são inseridos utilizando a estruturada definida, forma-se o que é conhecido como Base de Conhecimento (*Knowledge Base*) (GIARETTA; GUARINO, 1995). Essas Bases de Conhecimento são utilizadas por software para resolução de problemas, aplicações independentes de domínio e agentes de software com objetivos específicos. Por exemplo, os autores em (NOY;

MCGUINNESS et al., 2001) desenvolveram uma ontologia sobre vinhos, alimentos e combinações apropriadas de vinhos com refeições. Essa ontologia pode ser utilizada como base para aplicativos relacionadas a área de gestão em restaurantes, como um aplicativo que sugere um tipo de vinho de acordo com o menu do dia ou então responde consultas dos garçons e clientes do restaurante. Ainda, um segundo aplicativo poderia analisar o inventário de vinhos na adega e sugerir quais categorias de vinho devem ser expandidas e quais vinhos devem ser comprados para os próximos menus a serem utilizados.

3.1.2 OWL - *Web Ontology Language*

De acordo com (ANTONIOU; HARMELEN, 2009), o uso de ontologias no contexto da Web Semântica requer uma linguagem contendo alguns requisitos básicos: sintaxe e semântica bem definidas; suporte a *reasoning*; poder expressivo suficiente; e conveniência de expressão.

Diferentes linguagens de ontologia foram propostas e utilizadas nas ultimas décadas, incluindo a SHOE (*Simple HTML Ontology Extensions*) (HEFLIN; HENDLER; LUKE, 1999), a OIL (*Ontology Inference Layer*) (FENSEL et al., 2001), a DAML (*DARPA agent markup language*) (HENDLER; MCGUINNESS, 2000) e a DAML+OIL (HORROCKS et al., 2002). Buscando a definição de um padrão, a W3C desenvolveu, a partir das linguagens existentes e de padrões da web como XML, RDF e URIs, a linguagem OWL (*Web Ontology Language*) (MCGUINNESS; VAN HARMELEN et al., 2004). Nas próximas subseções são apresentados os elementos básicos para a construção de uma ontologia utilizando a OWL: as classes, os indivíduos (instâncias das classes) e as propriedades.

3.1.2.1 Classes

O conceito de classe ontológica é relacionado ao conceito de classe no âmbito de programação orientada a objetos e a tabelas em sistemas de banco de dados relacionais. Objetos no mundo real podem ser categorizados em grupos ou conjuntos com características similares (LACY, 2005).

Da maneira similar, indivíduos podem ser classificados em diferentes classes. Classes representam um grupo de instâncias de objetos (indivíduos) com propriedades similares, sejam elas implícitas ou explicitas. Através do agrupamento de objetos em classes ontológicas, afirmações podem ser feitas de maneira generalizada, incluindo todos os objetos membros da mesma classe.

3.1.2.2 Propriedades

Uma propriedade é uma associação binária que relaciona um indivíduo a um valor. Esse valor pode ser um simples dado numérico (*datatype properties*) ou um outro indivíduo (*object properties*) (MCGUINNESS; VAN HARMELEN et al., 2004).

As propriedades fornecem atributos para os indivíduos, semelhante aos métodos de acesso na programação orientada a objetos (*getters*) que fornecem dados a partir dos objetos da classe. Porém, diferentemente dos métodos do paradigma de orientação a objetos, uma propriedade pode ser associada a outras múltiplas classes ao invés de apenas uma, o que leva a sua reusabilidade (LACY, 2005).

3.1.2.3 Indivíduos

Os indivíduos representam instâncias das classes no domínio descrito pela ontologia. Novamente, essas instâncias são similares aos objetos (instâncias das classes) na POO, com a diferença de serem meramente representações da informação, não tendo nenhuma funcionalidade associada. Indivíduos podem representar ambos conceitos virtuais e objetos físicos.

3.1.3 Inferência lógica baseada em regras

Embora as ontologias sejam capazes de representar os conceitos sobre os indivíduos e como eles são relacionados, a utilização do processo de inferência traz a importante vantagem de possibilitar o descobrimento automático de novas informações baseado nos dados existentes na ontologia. O método de inferência da Web Semântica é baseado em regras do tipo Horn onde termos de conceitos ontológicos são expressados a fim de inferir novas informações sobre indivíduos da ontologia.

Essas regras são escritas utilizando a linguagem SWRL (*Semantic Web Rule Language*) que foi projetada pela W3C para ser a linguagem baseada em regras da Web Semântica. A SWRL é baseada na combinação das sublinguagens OWL DL e OWL Lite da OWL com as sublinguagens *Unary/Binary Datalog* da *Rule Markup Language* (O'CONNOR et al., 2005)

Assim como em outras diversas linguagens baseadas em regras, as regras SWRL são escritas na forma de uma implicação lógica entre um antecedente (corpo) e um consequente (cabeça). Toda vez em que as condições especificadas no antecedente forem verdadeiras,

então a regra é ativada e as condições do consequente também tornam-se verdadeiras (PRENTZAS; HATZILYGEROUDIS, 2007). Ambos antecedente e consequente consistem na conjunção de zero ou mais átomos que podem ser relacionados à conceitos ou propriedades dos indivíduos da ontologia. Formalmente, os átomos encontram-se em um dos seguintes formatos: C(x) ; P(x,y) ; sameAs(x,y) ; differentFrom(x,y) e builtIn(r,x,...), onde C é uma classe OWL, P é uma propriedade OWL, r é uma relação *built-in* (explanada a seguir), x e y são variáveis, indivíduos OWL ou valores de data OWL. Portanto, o átomo do tipo C(x) será verdadeiro se x é uma instância da classe C ; o átomo P(x,y) será verdadeiro se x está relacionado a y pela propriedade P ; o átomo sameAs(x,y) será verdadeiro se x é interpretado como sendo o mesmo objeto que y ; o átomo differentFrom(x,y) será verdadeiro se x e y são interpretados como sendo diferentes objetos ; e builtIn(r,x,...) será verdadeiro se o built-in r é verdadeiro de acordo com a interpretação de seus argumentos (HORROCKS et al., 2004).

Dessa forma, uma regra expressando o conceito familiar "Tio" pode ser escrita como:

```
Pai(?x,?y), Irmão(?x,?z) -> Tio(?z,?y)
```

Os predicados *built-in* possuem grande importância na linguagem SWRL pois ampliam de forma significativa o poder de expressividade da linguagem. Como dito anteriormente, os átomos *built-in* podem aceitar diversos argumentos de acordo com sua função. Entre os tipos de *built-ins* suportados pela SWRL estão os de comparação, os matemáticos, para *strings*, datas e listas[1]. Por exemplo, a regra a seguir utiliza o *built-in* `greaterThan` para expressar que um individuo possui um irmão mais velho:

```
temIrmao(?x1,?x2), temIdade(?x1,?idade1), temIdade(?x2,?idade2),
swrlb:greaterThan(?idade2,?idade1) -> temIrmaoMaisVelho(?x1,?x2)
```

Vale ressaltar que devido a forma natural em que as relações de causa-efeito são representadas pelas regras SWRL, qualquer alteração no conjunto de regras pode ser realizada de maneira simples e direta, o que é mais um aspecto interessante na utilização dessa técnica de inferência.

[1]Uma descrição detalhada dos *built-in* da linguagem SWRL pode ser encontrada em http://www.daml.org/rules/proposal/builtins.html

3.2 O paradigma multiagentes

O paradigma multiagentes, também conhecido como Programação Orientada a Agentes (AOP, do inglês *Agent-Oriented Programming*) é um paradigma de software relativamente novo que traz conceitos da teoria de Inteligência Artificial ao encontro do domínio dos sistemas distribuídos. O paradigma multiagentes essencialmente modela uma aplicação como uma coleção de componentes chamados agentes que são caracterizados por, entre várias coisas, autonomia, proatividade e capacidade de comunicação.

Através da autonomia os agentes podem tomar a iniciativa de realizar uma determinada tarefa mesmo sem uma requisição explícita do usuário. Sendo comunicativos, os agentes podem interagir com outras entidades para auxiliar no cumprimento dos objetivos próprios e dessas entidades. O modelo da arquitetura de uma aplicação orientada a agentes é intrinsecamente ponto a ponto, já que qualquer agente é capaz de iniciar a comunicação com qualquer outro agente ou é sujeito a receber uma comunicação a qualquer momento.

3.2.1 A plataforma JADE

A plataforma JADE (*Java Agent DEvelopment framework*) é provavelmente o *middleware* orientado a agentes em uso mais conhecido em atualmente. JADE é um *middleware* completamente distribuído com uma infraestrutura flexível que facilita o desenvolvimento de aplicações complexas baseadas no paradigma multiagentes por meio de um ambiente de execução (run-time environment) que por sua vez implementa os recursos necessários a um agente durante seu ciclo de vida, a lógica central dos agentes, e uma série de ferramentas gráficas que auxiliar na implementação e no monitoramento das aplicações. Ainda, como o JADE é escrito inteiramente em JAVA, ele se beneficia da enorme quantidade de bibliotecas disponíveis e, com isso, oferece uma série de abstrações na programação que permitem aos desenvolvedores a construção de sistemas multiagentes mesmo que esses possuam pouco conhecimento sobre a teoria de agentes (BELLIFEMINE; CAIRE; GREENWOOD, 2007).

Os recursos da plataforma são implementados conforme as especificações FIPA (*Foundation for Intelligent Physical Agents*), que descrevem um modelo de referência para as plataformas multiagentes por meio da especificação da linguagem utilizada na comunicação entre os agentes e na identificação de agentes cruciais para o gerenciamento da plataforma (BELLIFEMINE; POGGI; RIMASSA, 1999). Esses agentes são descritos a seguir:

- *Agent Management System* (AMS): agente que exerce o controle a supervisão sobre o acesso e uso da plataforma. Também é responsável por manter uma lista dos identificadores dos agentes (AID) que estão na plataforma;

- *Agent Communication Channel* (ACC): agente que fornece o caminho para o contato básico entre os agentes de dentro e fora da plataforma. É o método de comunicação padrão que oferece o serviço de roteamento das mensagens de maneira confiável, ordenada e precisa. Além disso, promove a interoperabilidade entre diferentes plataformas de agentes;

- *Directory Facilitator* (DF): fornece o serviço de páginas amarelas da plataforma utilizada pelos agentes para registrar seus serviços e buscar por agentes que oferecem serviços específicos.

Uma plataforma JADE é formada por um ou mais *containers*. Esses *containers* podem ser executados em um ou mais *hosts* conectados por uma na rede de comunicação, formando assim um sistema distribuído. Por sua vez, cada *container* pode conter zero ou mais agentes, exceto o *container* principal que sempre conterá ao menos os agentes AMS e DF. Ainda, o *container* principal é o primeiro a ser inicializado e é onde todos os outros *containers* devem se registrar durante a inicialização do sistema. A Figura 12 exemplifica um sistema multi agentes distribuído formado por três *hosts*, três *containers* e quatro agentes (A1,A2,A3 e A4), além do AMS e DF (CAIRE, 2003)

A estrutura das mensagens trocadas entre os agentes obedece a linguagem ACL (*Agent Communication Language*), também definida pela FIPA. A ACL é baseada na teoria dos atos de fala que afirma que mensagens representam ações ou atos comunicativos. Por exemplo, a sentença “Meu nome é João”, quando emitida, informa o destinatário com um item de informação. A FIPA-ACL define um conjunto de 22 atos de comunicação, também chamado de performativas, que capturam a essência da maioria das formas de comunicação, onde os mais utilizados são *inform*, *request*, *agree*, *not understood* e *refuse*. Além dos atos de comunicação, a FIPA-ACL utiliza diversos outros parâmetros para caracterizar as mensagens trocadas entre os agentes. A Tabela 2 apresenta esses parâmetros.

Uma das propriedades fundamentais e distintivas de um agente é a sua autonomia: um agente não é limitado a reagir a estímulos externos, também sendo capaz de iniciar novos atos comunicativos autonomamente. Isso requere que cada agente tenha sua própria

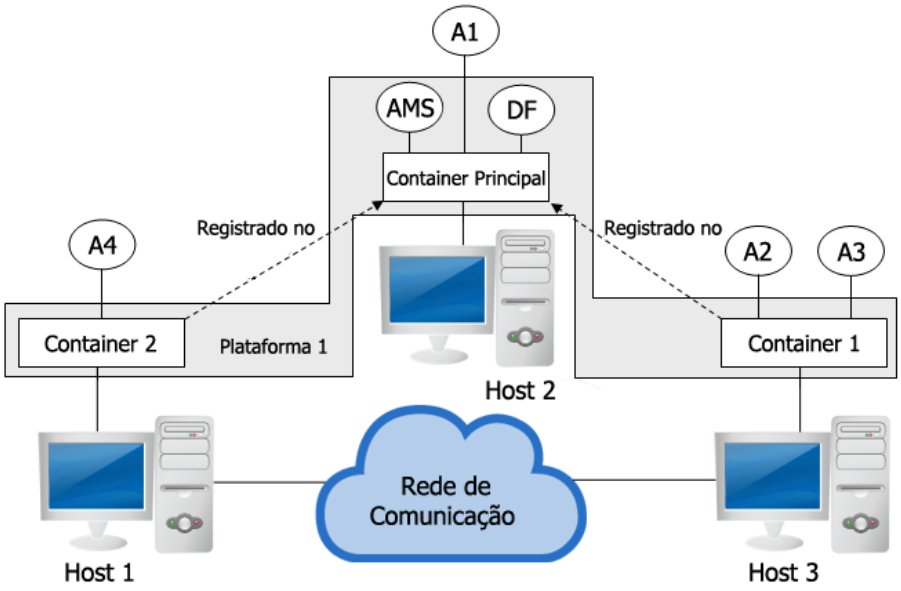


Figura 12: Demonstração da arquitetura JADE. Fonte: o autor (2016)

Tabela 2: Parâmetros utilizados nas mensagens FIPA-ACL

Parâmetro	Descrição
`performative`	Tipo do ato de comunicação da mensagem
`sender`	Identidade do agente remetente
`receiver`	Identidade do agente destinatário
`reply-to`	Identidade do agente cujas próximas mensagens dentro de uma conversação devem ser enviadas
`content`	Conteúdo da mensagem
`language`	Linguagem utilizada par expressar o conteúdo da mensagem
`encoding`	Codificação específica do conteúdo da mensagem
`ontology`	Referência a uma ontologia que expressa o significado dos símbolos do conteúdo da mensagem
`protocol`	Protocolo de interação usado na estrutura da conversação
`conversation-id`	Identificador único de um tópico uma conversação
`reply-with`	Expressão a ser usada pelo agente na resposta a fim de identificar a mensagem
`in-reply-to`	Referência a uma ação anterior na qual a mensagem é uma resposta
`reply-by`	Data e horário indicando até quando uma resposta deve ser recebida

thread de controle; entretanto, um agente pode participar de múltiplas conversações simultaneamente, além de desempenhar outras atividades que não envolvam troca de mensagens. Assim, JADE usa a abstração do Comportamento (*Behaviour*) para modelar as tarefas que um agente é capaz de realizar e então os agentes instanciam seus *behaviour* de acordo com suas necessidades e capacidades.

Do ponto de vista da programação concorrente, um agente é um objeto ativo contendo uma *thread* de controle interna. O JADE utiliza um modelo contendo uma *thread* por agente ao invés de uma *thread* por comportamento a fim de manter um menor número de *threads* sendo executadas na plataforma. Isso implica que, enquanto diferentes agentes são executados em um ambiente *multithread* preemptivo, dois comportamentos de um mesmo agente são escalonados cooperativamente. Um escalonador, implementado pela classe base `Agent` e transparente ao programador, possui uma política de escalonamento do tipo *round-robin* não-preemptivo entre todos os comportamentos disponíveis na fila das *threads* prontas (*ready queue*), permitindo a execução de um comportamento até que ele mesmo libere o controle de execução. Caso a tarefa que abriu mão do controle não tenha sido completada, ela é reescalonada no próxima rodada, a não ser que seja bloqueada.

Portanto, o programador deve estender a classe `Agent`, implementar as tarefas especificas àquele agente através de uma ou mais classes `Behaviour`, e então instanciar e adicioná-las ao agente. A classe `Agent` representa uma superclasse genérica para que o usuário possa definir seus agentes. Assim, do ponto de vista do programador, um agente JADE é simplesmente uma classe JAVA que estende a classe base `Agent`. Quando isto é feito, é herdado um comportamento básico escondido responsável por tratar as tarefas relacionadas às plataforma, assim como um conjunto básico de métodos que podem ser utilizados para implementar as tarefas de uma aplicação no agente, tais como enviar/receber mensagens e utilizar protocolos de interação padrões.

4 PROPOSTA DO SISTEMA DE ASSISTÊNCIA PERSONALIZADA

Este capítulo apresenta a proposta de um sistema de assistência personalizada para ambientes inteligentes, onde sua principal função é fornecer assistência personalizada na realização das Atividades de Vida Diária dos seus usuários, com um foco especial em idosos e pessoas com diversidades funcionais.

O sistema proposto baseia-se na utilização de duas diferentes tecnologias: as ferramentas da Web Semântica e o paradigma multiagentes. As ferramentas da Web Semântica são utilizadas na construção de uma ontologia que estabelece uma representação das principais características do usuário e de seu ambiente e no processo de inferência de conceitos como o contexto do usuário e suas preferências em relação aos serviços disponíveis no ambiente. O paradigma multiagentes é utilizado para a criação de uma arquitetura distribuída contendo componentes que utilizam as informações descritas na ontologia e interagem entre si para determinar a forma e o momento em que os serviços são requisitados, além de estabelecerem a comunicação entre o sistema de assistência personalizada e a arquitetura orientada a serviços presente no ambiente.

4.1 Ontologia do modelo de usuário AATUM

Nessa seção é apresentada a etapa da modelagem conceitual da ontologia proposta que estabelece um modelo formal dos usuários de ambientes assistidos, chamada AATUM (*Ambient Assistive Technology User Model*) (VARGAS; PEREIRA, 2015). Com o objetivo de tomar as melhores decisões e de prevenir os erros normalmente cometidos durante a etapa de modelagem conceitual, a ontologia proposta nesse trabalho foi desenvolvida baseando-se no método de design proposto por (NOY; MCGUINNESS et al., 2001), onde

são estabelecidas sete etapas, descritas em seguir.

4.1.1 Definição do domínio e escopo da ontologia

Os autores em (NOY; MCGUINNESS et al., 2001) sugerem que o processo de criação de ontologia inicie com a definição de seu domínio e escopo, ou seja, o que a ontologia irá representar, para que ela será utilizada e para quais perguntas as informações presentes na ontologia devem fornecer resposta. Para que essas definições fossem estabelecidas, foi considerada a abordagem inspiracional sugerida por (HOLSAPPLE; JOSHI, 2002), onde o desenvolvedor utiliza sua imaginação, criatividade e conhecimentos pessoais sobre o domínio de interesse.

A ontologia proposta nesse trabalho tem como principal objetivo a correta representação das características do usuário e seu contexto de forma que os serviços em uma casa inteligente possam ser corretamente personalizados de acordo com as reais necessidades do usuário, garantindo uma maior qualidade de vida aos seus habitantes. Não são considerados aspectos como a organização dos serviços no ambiente, assim como informações do usuário não relevantes para o objetivo em questão, como sua personalidade, estado emocional ou grupo étnico.

Dentre as principais questões que as informações presentes na ontologia devem ser capazes de responder, encontram-se as seguintes:

1. Em caso de emergência, para onde a equipe médica de socorro deve ser enviada? Quem mais deve ser contatado?

2. Em qual cômodo da residência o usuário encontra-se e qual é a atividade sendo realizada no momento?

3. Quais dispositivos estão presentes no cômodo onde o usuário encontra-se? Quais são as funções desses dispositivos e seus estados?

4. Quais as preferências do usuário em relação a diferentes atividades?

5. Quais os principais interesses do usuário em relação às atividades de lazer ou trabalho?

6. Qual é a condição de saúde do usuário e quais remédios devem ser administrados? Quais são os horários e doses desses medicamentos?

7. Quais deficiências o usuário possui e quais são os seus níveis?

8. Quais são as dificuldades que o usuário possui ao realizar suas Atividades de Vida Diária, como preparar uma refeição, cuidar de sua higiene pessoal e administrar os seus medicamentos?

4.1.2 Consideração sobre o reuso de ontologias existentes

Conforme descrito no Capítulo 2, a criação de modelos de usuário baseados em ontologias tem sido amplamente discutido em diversos domínios de aplicação nos últimos anos. Aproveitando um dos grandes benefícios inerentes à Web Semântica, isto é, a facilidade no reuso de informação, o modelo de usuário proposto neste trabalho baseia-se em algumas ontologias previamente propostas na literatura. Após a análise do estado da arte ter sido realizada, foram selecionados os principais aspectos que poderiam auxiliar no desenvolvimento do modelo de usuário voltado à personalização de serviços em ambientes assistidos. Essa estratégia vai ao encontro da abordagem sintética descrita por (HOLSAPPLE; JOSHI, 2002), onde um conjunto de ontologias existentes relacionadas ao domínio de interesse é selecionado de forma a construir uma ontologia unificada contendo os conceitos necessários.

A estratégia em estabelecer o usuário como conceito central da ontologia e a forma em que os interesses do usuário são representados são baseados na ontologia PCEICL (FREDRICH; KUIJS; REICH, 2014). A modelagem do contexto e das preferências do usuário baseia-se no trabalhos (SUTTERER; DROEGEHORN; DAVID, 2008) e (STAN et al., 2008), onde são definidos perfis de preferências específicos para os possíveis contextos, além de um perfil padrão a ser utilizado em situações onde o contexto do usuário não é relevante.

O maior problema encontrado nos modelos de usuário existentes é a falta de uma representação uniforme e consistente das condições de saúde, limitações e deficiências do usuário. A representação dessa informação é normalmente projetada para atender às necessidades de uma aplicação em específico, dificultando o seu reuso em outros domínios. No âmbito da personalização de serviços em ambientes assistidos essa questão é ainda mais delicada, pois a forma em que o serviços devem ser personalizados está diretamente relacionado aos níveis e tipos de deficiência do usuário.

Buscando resolver essa questão, esse trabalho propõe a incorporação da Classificação

Internacional de Funcionalidade, Incapacidade e Saúde (ICF, do inglês *International Classification of Functioning, Disability and Health*) no modelo de usuário. Essa classificação foi implementada na forma de uma ontologia pela própria Organização Mundial de Saúde (OMS) e encontra-se disponível online[1], o que facilita sua utilização.

4.1.3 Enumeração de termos importantes da ontologia

A terceira etapa da metodologia utilizada consiste na criação de uma ampla lista contendo termos considerados importantes no domínio em questão, sem a preocupação de como esses termos serão representados na ontologia, como eles são relacionados ou se existe alguma sobreposição entre esses termos. Essa lista é então utilizada nas próximas duas etapas, onde a ontologia é de fato construída.

Como o objetivo geral desse trabalho é aumentar a qualidade de vida de pessoas com necessidades especiais através da disponibilização de assistência personalizada, é necessário que a ontologia contenha um conjunto de preferências relacionadas à serviços que possuem um impacto significativo na realização das AVD e que possam garantir um maior conforto aos usuários. Essas preferências e a forma em que elas podem influenciar na vida dos usuários do ambiente assistido são apresentados a seguir. O restante dos termos da lista[2] são apresentados na próxima subseção, quando as classes e suas propriedades são definidos.

`AmbientLighting`: Nível de iluminação do ambiente. O uso de uma iluminação correta tem influência direta na qualidade do sono e no conforto durante atividades de lazer e de trabalho, como assistir a um filme ou ler um livro. Além disso, o ajuste correto da iluminação pode auxiliar usuários com dificuldade de visão na locomoção entre os cômodos da residência.

`AmbientTemperature`: Temperatura do ambiente. A utilização de uma temperatura adequada garante o conforto térmico do usuário e pode auxiliar na qualidade do sono (VALHAM et al., 2012), além de possibilitar a economia de energia.

`ScreenBrightness`: Nível de brilho utilizado em dispositivos de imagem, como computadores e aparelhos de televisão. A utilização de um nível adequado faci-

[1] http://apps.who.int/classifications/icfbrowser/

[2] Todos os termos utilizados na ontologia encontram-se em Inglês com a finalidade de facilitar a sua publicação e o seu reuso

lita a visualização de imagens e textos, podendo auxiliar usuários com problemas relacionados à visão.

`ScreenContrast:` Nível de contraste utilizado em dispositivos de imagem, como computadores e aparelhos de televisão. Assim como no caso anterior, o uso de um nível de contraste adequado facilita a visualização de imagens e textos.

`ColorScheme:` Esquema de cores utilizados em dispositivos de imagem. Tem grande importância para usuários com dificuldade em diferenciar cores semelhantes e imagens com pouca diferença de contraste.

`MediaType:` Tipo de mídia utilizado para fornecer informações ao usuário, como lembretes ou alarmes. Garante que usuários com deficiências relacionadas à audição e visão, ou em situações especificas como durante o sono ou o banho, consigam receber a informação necessária.

`MediaLanguage:` Linguagem utilizada em textos e áudios apresentados ao usuário. A utilização da linguagem correta é fundamental para que o usuário possa entender as informações apresentadas a ele.

`MediaVolume:` Volume utilizado em dispositivos de mídia como aparelhos de televisão e *smartphones*. A utilização de um nível adequado proporciona um maior conforto durante atividades de lazer, como assistir à televisão, e garante que avisos ou alarmes sejam ouvidos claramente pelo usuário.

`TextSize:` Tamanho do texto exibido em dispositivos de imagem. Tem impacto direto em diversas Atividades de Vida Diária e permitem que usuários com problemas relacionados a visão possam ler confortavelmente.

`MobilityDeviceType:` Tipo de dispositivo utilizado para auxiliar na mobilidade do usuário. Essa preferência pode ser utilizada em sistemas que possuam múltiplos usuários, como em casas de repouso para idosos, de forma a permitir um controle dos dispositivos sendo utilizados, ou na sugestão do tipo do dispositivo adequado para os usuários.

`MobilityDeviceSpeed:` Velocidade do dispositivo utilizado para auxiliar na mobilidade do usuário, em especial cadeiras de roda elétricas. Permite que usuários com

problemas cognitivos ou que possuam problemas de coordenação motora possam conduzir suas cadeiras de rodas com conforto e segurança.

`FurnitureHeight:` Altura utilizada nos móveis da casa. Pode ser utilizado em móveis inteligentes que possuam controle de altura automatizado, como estantes e mesas, auxiliando usuários de cadeiras de rodas ou que possuam problemas motores.

`ReminderLevel:` Frequência com que lembretes devem ser apresentados ao usuário. Permite que usuários com problemas cognitivos possam ser alertados de maneira adequada sobre as Atividades de Vida Diária a serem realizadas ou remédios a serem administrados.

4.1.4 Definição das classes e suas propriedades

As próximas duas etapas da metodologia utilizada consistem em definir as classes que compõe a ontologia e as propriedades que descrevem a estrutura interna e as relações entre as classes. De acordo com (NOY; MCGUINNESS et al., 2001), essas duas etapas estão fortemente entrelaçadas e normalmente são realizadas simultaneamente.

Para a definição das classes, foi utilizada uma abordagem *top-down*, iniciando com a definição das classes mais gerais da ontologia e em seguida especializando essas classes através da criação de suas sub-classes. Para isso, foram selecionados da lista criada na etapa anterior os termos que são suficientemente capazes de descreverem um conceito sem a utilização de outros termos, ou seja, que são predicados unários. Um exemplo é a classe `User`, que refere-se ao usuário do sistema. Por outro lado, o termo `Has Impairment` não é capaz de representar um conceito, necessitando de termos adicionais como a pessoa que possui a debilitação e a debilitação em si.

A estrutura interna das classes da ontologia, isto é, suas propriedades de dados, é construída a partir dos termos restantes da lista definida anteriormente. De maneira análoga, as relações entre as classes são estabelecidas através das propriedades de objeto, também utilizando os termos encontrados na lista definida na etapa anterior.

Para facilitar a leitura desse trabalho, primeiramente é apresentada a estrutura hierárquica e as relações entre as classes da ontologia através da Figura 13 e em seguida, as classes e suas estruturas internas são detalhadas. O conjunto de classes que representam o contexto do usuário (elipses em azul) é chamado de **Modelo de Contexto**.

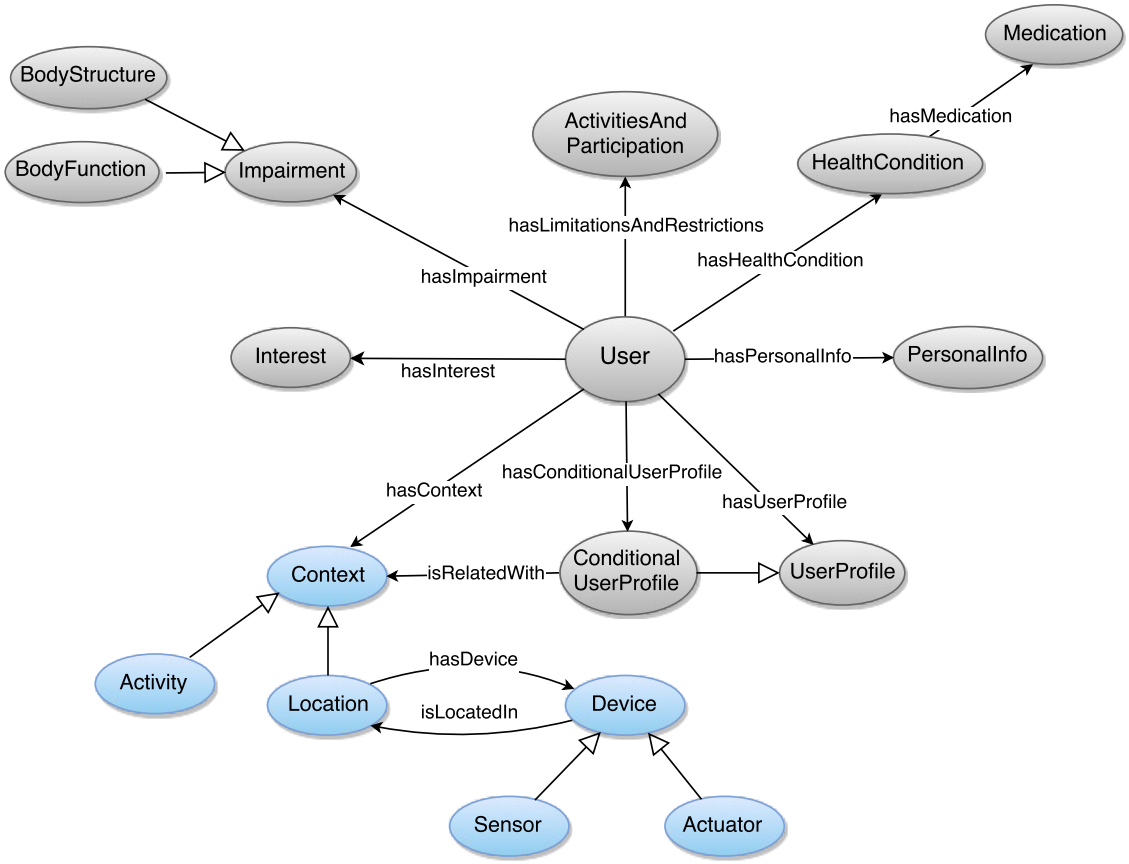


Figura 13: Classes da ontologia proposta e suas relações. Fonte: o autor (2016)

`User`: conceito central da ontologia, representa o usuário do ambiente assistido. Todas as outras classes da ontologia estão, de alguma forma, relacionadas a essa. A sua propriedade `hasUserID` permite a associação de um número de identificação único ao usuário, o que pode ser útil para sistemas de identificação como sensores RFID e sistemas de visão computacional, por exemplo.

`PersonalInfo`: apresenta um conjunto de informações básicas do usuário através das propriedades `hasDateOfBirth`, `hasName`, `hasAddress`, `speaksLanguage` e `hasContactNumber`. Embora sejam informações básicas, elas possuem grande importância em ambientes assistidos, como em casos de emergência onde é necessário enviar uma ambulância à casa do usuário, ou então para a construção de um relatório estatístico documentando as condições de saúde de um bairro ou região.

`HealthCondition`: classe responsável por armazenar informações relacionadas às condições de saúde do usuário, isto é, desordens, doenças ou ferimentos diagnos-

ticados por um profissional de saúde. A propriedade `hasICDcode` permite que o código da Classificação Internacional de Doenças (ICD, do inglês *International Classification of Diseases*) relacionado a uma condição de saúde específica seja armazenado, garantindo que esse informação seja representada de uma maneira padronizada e internacionalmente conhecida. As informações contidas nessa classe podem ser utilizadas, por exemplo, para uma análise geral das condições de saúde de um grupo da população.

`Medication`: reúne as informações dos medicamentos utilizados no tratamento de uma condição de saúde específica. A propriedade `hasPurpose` é utilizada para indicar o propósito do medicamento, enquanto as propriedades `hasProprietaryName` e `hasNonProprietaryName` são utilizadas para armazenar os nomes comerciais e o nome científico do medicamento, respectivamente. Outra importante informação que poderia ser representada nessa classe é o código do medicamento. Porém, devido à falta de um modelo internacionalmente aceito e ao grande número de padrões de códigos existentes, tais como o WHO-DDE (*World Health Organisation Drug Dictionary Enhanced*), MedDRA (*Medical Dictionary for Regulatory Activities*), COSTART (*Coding Symbols for Thesaurus of Adverse Reaction Terms*), CHNM (Código Hospitalar Nacional do Medicamento), optou-se por não representar essa informação. Além disso, a propriedade `hasSchedule` permite que os horários e doses do medicamento sejam armazenados, o que é de fundamental importância para uma correta assistência ao usuário que utiliza esse medicamento.

`ActivitiesAndParticipation`: classe ICF utilizada para representar as dificuldades que o usuário possui ao realizar as Atividades de Vida Diária e os problemas que o indivíduo pode encontrar ao se envolver em situações do cotidiano, como relacionar-se com outras pessoas. A propriedade `hasCapacityLevel` é utilizada para quantificar essas dificuldades de acordo com o modelo proposto pela OMS.

`Impairment`: debilitação relacionada às estruturas anatômicas do corpo-humano, como órgãos e membros, e às funções fisiológicas dos sistemas corpóreos. A propriedade `hasExtentOfImpairment` é utilizada para indicar a severidade da debilitação de acordo com o modelo proposto pela OMS.

`Interest`: interesses do usuário em relação às atividades de lazer ou trabalho. Cada

interesse é associado a um peso através da propriedade `hasLevelOfInterest` a fim de estabelecer o quanto o usuário é interessado naquele assunto em específico. Ainda, informações adicionais como descrições ou programações podem ser armazenadas utilizando a propriedade `hasAdditionalInformation`. Essa classe pode ser útil para aplicativos de recomendação ou filtragem de conteúdo que, embora possam parecer irrelevantes no âmbito dos ambientes assistidos, podem ajudar os usuários a terem uma maior qualidade de vida. Encontrar um filme dentre as dezenas de canais existentes, por exemplo, pode ser uma tarefa difícil para usuários idosos ou com problemas cognitivos.

`Context:` informação utilizada para caracterizar a situação do usuário no ambiente. Nesse trabalho optou-se por definir contexto como o cômodo da residência (`Location`) onde o usuário encontra-se e a Atividade de Vida Diária sendo realizada pelo usuário (`Activity`). Essa classe é de fundamental importância para que a personalização de serviços seja feita de modo adequado, visto que as necessidades do usuário estão fortemente relacionados às atividades e a localização do usuário. Ainda, a sub-classe `Location` possui algumas propriedades utilizadas para caracterizar o do cômodo da residência: `LocationType` refere-se ao tipo de cômodo (quarto, banheiro, etc.); `hasAmbientNoiseLevel` indica o nível sonoro do ambiente; `hasAmbientLightingLevel` representa o nível de iluminação; `hasAmbientTemperature` fornece a temperatura atual do cômodo.

`Device:` em contraste às classes apresentadas até o momento, o contexto do usuário apresenta um comportamento dinâmico e sua determinação é realizada apenas pelo processo de inferência, ou seja, o seu valor não é informado explicitamente. Para que esse processo de inferência possa ser realizado, é necessário que a ontologia contenha as informações dos dispositivos presentes no ambiente. Por exemplo, uma das possíveis estratégias para determinar que o usuário está dormindo é utilizando sensores vestíveis (*wearable sensors*) que monitoram os batimentos cardíacos do indivíduo. Outra possível estratégia - mais simples, porém ainda funcional e utilizada em trabalhos relacionados - poderia utilizar o estado de sensores que identificam que o usuário está deitado sobre sua cama ou sofá. Deve-se notar que essas informações podem ser entregues de diversas maneiras: os batimentos cardíacos podem ser medidos por um *smartwatch*, uma pulseira médica ou uma cinta; o usuário

deitado sobre a cama pode ser itendificado por um sensor de pressão colocado sob a cama, por um sistema de visão computacional ou até mesmo por um sensor infravermelho. Para que não haja nenhum tipo de limitação em relação ao tipo de dispositivo utilizado no sistema, a propriedade `hasDeviceType` é utilizada para informar a função do dispositivo, enquanto as propriedades `hasSensorValue` e `hasActuatorState` informam o valor fornecido pelos sensores e o estados dos atuadores, respectivamente. Dessa forma, o processo de inferência é realizado baseado na função do dispositivo, não interessando a tecnologia, modelo ou outras características do sensor. Por exemplo, para inferir que o usuário está dormindo, bastaria saber se o valor dos batimentos cardíacos fornecido por um sensor cujo `hasDeviceType` é igual a `HeartRateMonitor` é inferior a um certo valor. A estratégia mais simples utilizaria o valor do sensor que identifica que o usuário está deitado sobre a cama (`hasDeviceType` igual a `BedOccupancy`).

`UserProfile`: perfil de usuário que descreve o conjunto de preferências apresentados anteriormente para situações independentes do contexto. Cada preferência é estabelecida através de uma propriedade de dados específica. Por exemplo,a temperatura ambiente preferida é representada pela propriedade `hasAmbientTemperature-Preference`, enquanto o volume utilizado em dispositivos de mídia é representado pela propriedade `hasMediaVolumePreference`.

`ConditionalUserProfile`: assim como sua superclasse `UserProfile`, descreve o conjunto de preferências relacionadas aos serviços do ambiente inteligente, porém relacionados a um contexto específico. Dessa maneira, para cada possível contexto do usuário, um `ConditionalUserProfile` contendo as preferências para aquele contexto é associado.

4.1.5 Definição das restrições das propriedades

Uma vez definidas as propriedades das classes, é necessário estabelecer suas restrições quanto à quantidade (cardinalidade) e aos tipos de valores que podem ser associados. A Figura 14 apresenta a ontologia contendo as restrições impostas às propriedades. Todo usuário necessita de um, e apenas um `PersonalInfo`, `UserID` e `UserProfile`. Em relação ao contexto, o usuário poderá ter até dois valores associados: uma localização e uma atividade sendo realizada. Não há limitações em relação ao número de

`ConditionalUserProfile` associado a um usuário, já que para cada possível contexto um perfil condicional pode ser criado. Naturalmente, os interesses, debilitações, condições de saúde e dificuldades podem ter múltiplos valores ou estarem ausentes. Ainda, cada preferência presente no perfil do usuário pode ter no máximo um valor.

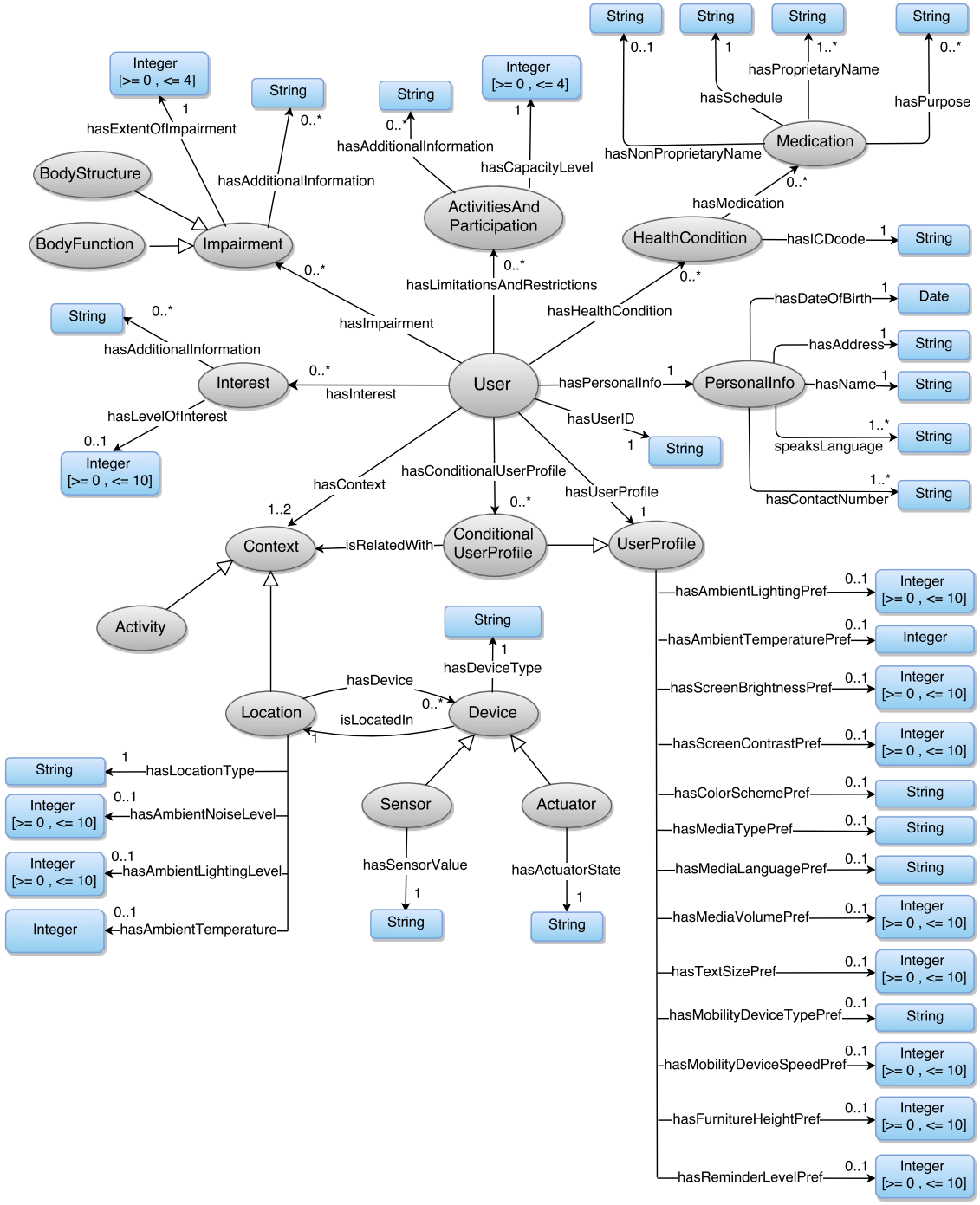


Figura 14: A ontologia do modelo de usuário completa. Fonte: o autor (2016)

Em relação ao tipo de dado que pode ser associado às propriedades de dados, diferentes restrições são utilizadas. As propriedades `hasDeviceType`, `hasLocationType`, `hasColorSchemePreference`, `hasMediaTypePreference` e `hasMobilityDevicePreference` são restringidas para permitirem apenas *strings*. Para preferências que indicam níveis, como o brilho da tela e volume, foi estabelecida uma escala que varia de 0 a 10 devido a facilidade de representar os valores utilizados. Outra possível estratégia seria utilizar os valores absolutos para cada grandeza: a luminância do ambiente poderia ser representada em lux; o nível sonoro em decibéis (dB), e assim por diante. Porém, a requisição de serviços utilizando esses valores como parâmetros não aconteceria de forma natural e necessitaria - no mínimo - de sensores adicionais. Por exemplo, é muito mais natural e simples a requisição para ajustar o volume da televisão para o nível 4 (representando 40% do volume máximo) do que para 70 dB.

4.1.6 Criação das instâncias

A ultima etapa de desenvolvimento de ontologias consiste na criação dos indivíduos, também chamados de instâncias das classes. Nesse trabalho essa criação é realizada em dois momentos distintos: os indivíduos que representam as possíveis AVD são criados anteriormente à utilização do sistema, enquanto o restante dos indivíduos é criado em tempo de execução, de acordo com o que acontece no ambiente. O processo de instanciação em tempo de execução pode ser dividida em dois grupos:

- Indivíduos relacionados às informações dos usuários: as instâncias das classes que representam informações pertinentes ao usuário, como seus interesses, condições de saúde e informações pessoais são criadas no momento em que um novo usuário é cadastrado no sistema.

- Indivíduos relacionados ao ambiente: as instâncias das classes que representam os cômodos da residência, os sensores e atuadores são criadas e removidas de forma dinâmica de acordo com o ambiente onde o sistema está sendo utilizado.

Além disso, informações relacionadas ao contexto e às preferências do usuário não são inseridas explicitamente na ontologia na forma de instâncias das classes. Ao invés disso, essas informações são representadas por axiomas inferidos, conforme o processo de inferência apresentado na próxima seção.

4.2 Processo de inferência

Entre as principais técnicas de inferência utilizadas pelos sistemas de personalização encontrados na literatura, como em (RICQUEBOURG et al., 2007), (SKILLEN et al., 2014), (WANG et al., 2004) e (TIBERGHIEN et al., 2012), o método baseado em regras e implementado utilizando as ferramentas da Web Semântica tem destacado-se como uma tendência. Assim, o sistema proposto nesse trabalho utiliza as tecnologias da Web Semântica para o processo de inferência do contexto e das preferências do usuário em relação aos serviços disponíveis no ambiente. Portanto, é necessário definir as regras (cláusulas de Horn) estabelecendo relações de causa-efeito a serem utilizadas pelo *reasoner* semântico.

Nesse trabalho as regras pertencentes ao processo de inferência são divididas em três conjuntos, cada um contendo um objetivo específico: estabelecer as preferências padrões do usuário de forma automática de acordo com as suas deficiências e dificuldades; deduzir as atividades (com baixa granularidade de detalhamento) sendo desempenhadas pelo usuário baseado nos estados dos dispositivos instalados no ambiente; inferir as preferências dependentes de contexto baseado nos valores padrões previamente definidos. A seguir os diferentes conjuntos de regras são detalhados.

4.2.1 Inferência das preferências padrões

Esse conjunto de regras é utilizado para determinar de forma automática os valores para as preferências relacionadas aos serviços disponíveis no ambiente de acordo com as informações do usuário descritas através das propriedades `hasImpairment` e `hasLimitationsAndRestrictions`. Essas preferências são independentes de contexto e só irão mudar caso as informações pessoais do usuário sejam atualizadas.

Para a construção das regras, em um primeiro momento buscou-se na literatura trabalhos que relacionassem os níveis das limitações e deficiências descritos no modelo ICF com valores de preferências a serem utilizados nos serviços relacionados às AVD. Embora o ICF seja um modelo internacional e amplamente conhecido, não foi encontrado nenhum trabalho que apresentasse um estudo relacionando esses valores. Portanto, as regras propostas nesse trabalho foram construídas de forma empírica, baseando-se na seguinte metodologia:

1. Busca-se no ICF as classes representando as deficiências e dificuldades que possuem relação direta com a preferência em questão;

2. Analisa-se de que maneira a preferência pode contribuir na superação das dificuldades impostas pela deficiência ou no aumento do conforto do usuário;

3. A relação identificada no passo anterior é formalizada na forma de uma equação matemática ou uma árvore de decisão, de acordo com a necessidade.

A seguir são detalhadas as relações entre as deficiências, dificuldades e preferências a serem representadas pelas regras. Embora regras da lógica de primeira ordem sejam simples e sempre estabeleçam uma relação de causa-efeito, ambas formas de representação utilizadas nesse trabalho (função matemática e árvore de decisão) podem ser facilmente implementadas utilizando uma combinação de regras, o que mostra mais uma vantagem na utilização desse método de inferência.

`AmbientLighting` Dentre as classes do ICF que representam as deficiências e dificuldades do indivíduo, `b21020.Light sensitivity` e `b2100.Visual acuity functions` são as que possuem relação direta com o nível de iluminação de ambiente. Para que o usuário possa desempenhar suas AVD com maior facilidade, quanto maior a sensibilidade a luz, menor deve ser o nível de iluminação no ambiente. Por outro lado, quanto pior for a acuidade de visão, maior deve ser o nível de iluminação do ambiente. Assim, neste trabalho o nível de iluminação no ambiente é inferido utilizando a seguinte função:

$$AmbientLighting(LightSensivity, VisualAcuity) = 5 + VisualAcuity - LightSensivity$$

`ScreenBrightness` O nível de brilho dos dispositivos de imagem obedece a mesma lógica apresentada para o nível de iluminação no ambiente. Assim, a função utilizada neste trabalho que estabelece o nível de brilho é representada por:

$$ScreenBrightness(LightSensivity, VisualAcuity) = 5 + VisualAcuity - LightSensivity$$

ScreenContrast O correto ajuste do nível de contraste nos dispositivos de imagem pode auxiliar os usuários que possuem sensibilidade ao contraste (`b21022.Contrast sensitivity`) e problemas relacionados a acuidade de visão (`b2100.Visual acuity functions`). Quanto maior o nível dos dois problemas, maior deve ser o contraste utilizado nos dispositivos. Essa relação pode ser expressa de diversas maneiras, como por exemplo através da soma dos níveis de deficiência ou então através da média dos valores. Neste trabalho optou-se pela soma entre os dois níveis e a constante 5, limitando o resultado ao valor 10. Assim:

$$ScreenContrast(ContrastSensitivity, VisualAcuity) = min(10, 5 + ContrastSensitivity + VisualAcuity)$$

ColorScheme Para a definição do esquema de cores a ser utilizado, é necessário relacionar as informações sobre problemas ligados à diferenciação de cores (`b21021.Colour vision`) e a sensibilidade ao contraste (`b21022.Contrast sensitivity`). Caso o usuário possua alguma deficiência relacionada a diferenciação das cores, é natural que seja utilizado um esquema de cores adaptado ao tipo específico de deficiência descrito através da propriedade `hasAdditionalInformation`. Caso o usuário não possua nenhum problema de visão de cores e apresente um alto nível de sensibilidade ao contraste, um esquema de alto contraste deve ser utilizado. A Figura 15 ilustra o processo de decisão do esquema de cores.

MediaLanguage A linguagem utilizada em textos e áudios apresentados ao usuário é estabelecida diretamente de acordo com a informação representada pela propriedade `speaksLanguage` associada às informações pessoais usuário.

MediaVolume O volume utilizado em dispositivos de mídia depende apenas do nível dos problemas relacionados à audição (`b230.Hearing`), conforme a função:

$$MediaVolume(Hearing) = 5 + Hearing$$

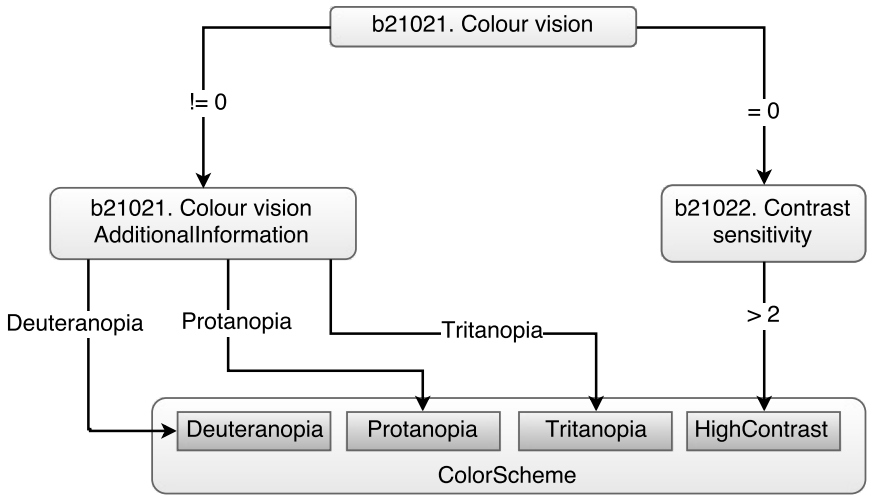


Figura 15: Árvore de decisão para a regra da preferência `ColorScheme`. Fonte: o autor (2016)

`MediaType` O tipo de mídia a ser utilizado tem relação direta com problemas relacionados à audição (`b230.Hearing`) e a visão (`b210.Seeing`). A estratégia de decisão proposta nesse trabalho, apresentada na Figura 16, leva em consideração o fato de que ambos problemas podem ocorrer de forma simultânea. Para que uma preferência seja estabelecida, a diferença entre os níveis das deficiências de audição e visão deve ser maior que um. Caso isso não ocorra, a preferência não conterá nenhum valor.

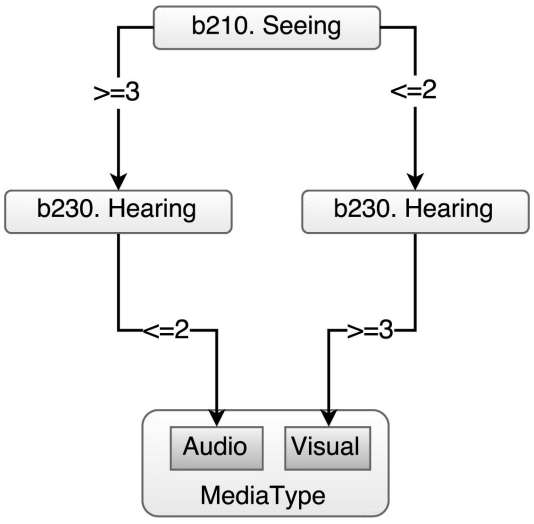


Figura 16: Árvore de decisão para a regra da preferência `MediaType`. Fonte: o autor (2016)

`TextSize` O tamanho do texto apresentado ao usuário depende apenas do nível dos problemas relacionados à acuidade de visão (b2100. Visual acuity functions), como ilustra a seguinte função:

$$TextSize(VisualAcuity) = 5 + VisualAcuity$$

`MobilityDeviceType` O tipo de dispositivo de auxilio de mobilidade deve ser escolhido de acordo com a dificuldade em caminhar (`d450.Walking`) e em mover-se utilizando equipamentos de assistência (`d465.Moving around using equipment`). A relação entre os níveis de deficiências com o valor da preferência, obedece a seguinte lógica: quanto maior a dificuldade em utilizar um equipamento de assistência, menor é a influência da dificuldade em caminhar sobre o nível de automação necessário no equipamento. A Figura 17 elucida essa relação.

`MobilityDeviceSpeed` Caso o usuário utilize uma cadeira de rodas elétrica, sua segurança e conforto podem estar comprometidos caso exista alguma dificuldade em utilizar o *joystick* controlador da cadeira da maneira correta. Uma das maneiras de contornar esse problema é limitando a velocidade da cadeira de rodas de acordo com o nível da dificuldade em realizar momentos finos com as mãos (`d440.Fine hand use`). Neste trabalho optou-se por utilizar uma função linear que limita a velocidade a 20% da velocidade máxima da cadeira para usuários que possuam um alto nível de dificuldade (4), e que não limita a velocidade para usuários que não possuam dificuldade em controlar a cadeira. Essa função é apresentada a seguir:

$$MobilityDeviceSpeed(FineHandUse) = 10 - 2 * FineHandUse$$

`FurnitureHeight` A altura dos moveis da residência está relacionada com o tipo de dispositivo de mobilidade utilizado e pelo nível de dificuldade em carregar e mover objetos (`d430.Lifting and carrying objects`). Caso o usuário utilize qualquer tipo de cadeira de rodas ou tenha um alto nível de dificuldade em carregar e mover objetos, o nível dos móveis deve ficar em uma altura próxima a um metro (Prefeitura da Cidade de São Paulo, 2014). Caso o usuário não tenha nenhuma das limitações supracitadas, nenhuma preferência é estabelecida.

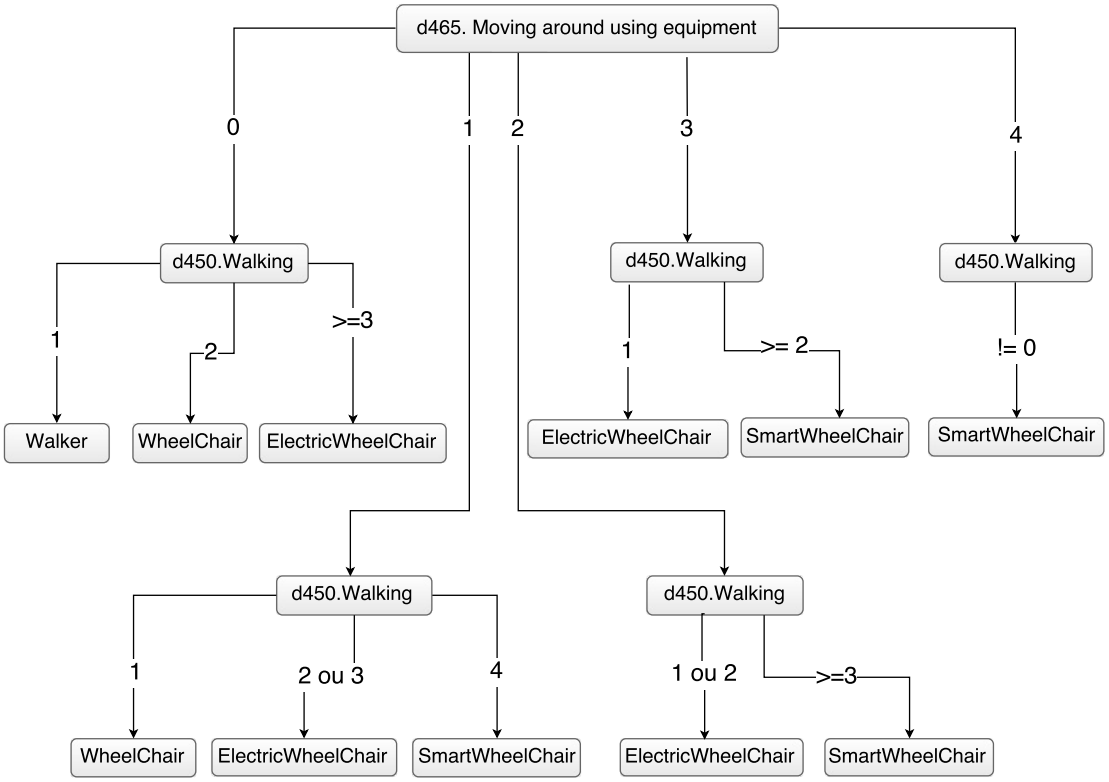


Figura 17: Árvore de decisão para a regra da preferência `MobilityDeviceType`. Fonte: o autor (2016)

`ReminderLevel` A frequência em que lembretes devem ser apresentados ao usuário dependem do nível dos problemas relacionados às funções mentais (`b1.Mental functions`). Neste trabalho a frequência dos lembretes é estabelecida obedecendo uma relação linear, como mostra a equação:

$$ReminderLevel(MentalFuntions) = 5 + MentalFunctions$$

4.2.2 Inferência do contexto

Uma das principais características no sistema proposto é a capacidade de personalizar a assistência de acordo com o contexto do usuário. Dessa maneira, é necessário estabelecer a maneira em que a localização do usuário e a atividade sendo realizada devem ser inferidas a partir das informações coletadas pelos sensores no ambiente através de um conjunto de regras.

Embora o método de inferência baseado em regras seja simples e possa apresentar resultados equivocados devido à sua natureza de representar apenas relações de causa-efeito, sua performance em ambientes inteligentes e sistemas de personalização tem sido considerada

satisfatória, como demonstrado nos trabalhos (WANG et al., 2004), (RICQUEBOURG et al., 2007) e (HONG et al., 2009). As regras utilizadas neste trabalho foram construídas utilizando uma abordagem empírica, baseando-se nos trabalhos supracitados, e não visam a detecção de atividades complexas ou que fujam de um padrão considerado normal para os usuários finais do sistema proposto - pessoas idosas e/ou com diversidades funcionais morando sozinhas em suas residências.

A seguir são apresentadas as regras que determinam a localização e as Atividades de Vida Diária dos usuários. Com o objetivo de melhorar a compreensão do texto, primeiramente as regras são introduzidas utilizando linguagem textual e então as cláusulas de Horn correspondentes são apresentadas.

Regra 1 (`UserLocation`)**.** *Se um sensor capaz de identificar usuários identifica um usuário, então o usuário identificado encontra-se no mesmo cômodo onde o sensor está.*

$$\texttt{User(u)} \wedge \texttt{Location(l)} \wedge \texttt{hasUserID(u,id)} \wedge \texttt{Sensor(s)} \wedge \texttt{hasDeviceType(s,UserIdentification)} \wedge \texttt{isLocatedIn(s,l)} \wedge \texttt{hasSensorValue(s,id)} \implies \texttt{hasLocation(u,l)}$$

Regra 2 (`Cooking`)**.** *Se um usuário encontra-se em uma cozinha que contenha um fogão ligado, então esse usuário está cozinhando.*

$$\texttt{User(u)} \wedge \texttt{Location(l)} \wedge \texttt{hasLocationType(l,Kitchen)} \wedge \texttt{hasLocation(u,l)} \wedge \texttt{Actuator(a)} \wedge \texttt{hasDeviceType(a,Stove)} \wedge \texttt{isLocatedIn(a,l)} \wedge \texttt{hasActuatorState(a, true)} \implies \texttt{hasActivity(u,Cooking)}$$

A determinação da atividade `Sleeping` pode ser realizada de diversas maneiras. Caso o usuário esteja utilizando algum dispositivo capaz de detectar o seu estado de acordo com os sinais vitais coletados[3], basta conhecer o valor informado pelo dispositivo. Outra opção é a simples comparação entre os valores dos batimentos cardíacos informados por um sensor vestível com um valor pré-determinado. Neste trabalho optou-se por utilizar uma abordagem baseada na combinação de diversos sensores presentes no ambiente, como mostra a regra a seguir:

[3] A técnica utilizada para a detecção do estado do usuário varia de acordo com o dispositivo vestível, podendo ser baseada nos batimentos cardíacos, na movimentação do usuário ou na combinação de ambas informações.

Regra 3 (`Sleeping `)**.** *Se um usuário encontra-se deitado em uma cama em um quarto cuja porta está fechada, então esse usuário está dormindo.*

$$\texttt{User(u)} \wedge \texttt{Location(l)} \wedge \texttt{hasLocationType(l,Bedroom)} \wedge \texttt{hasLocation(u,l)} \wedge \texttt{Sensor(s1)} \wedge \texttt{hasDeviceType(s1,BedOccupancy)} \wedge \texttt{isLocatedIn(s1,l)} \wedge \texttt{hasSensorValue(s1, true)} \wedge \texttt{Sensor(s2)} \wedge \texttt{hasDeviceType(s2,ClosedDoor)} \wedge \texttt{isLocatedIn(s2,l)} \wedge \texttt{hasSensorValue(v2,true)} \implies \texttt{hasActivity(u,Sleeping)}$$

Embora alguns trabalhos presentes na literatura utilizem a informação do nível de luminosidade no ambiente para detectar que o usuário está dormindo, neste trabalho optou-se por não utilizá-la, já que o controle da luminosidade pode ser realizado de maneira automática e de acordo com as preferências do usuário. Caso o nível de luminosidade fosse utilizado como uma condição para que o usuário esteja dormindo, esse controle automático não poderia ser efetuado como o desejado.

Regra 4 (`Working`)**.** *Se um usuário encontra-se sentado dentro de um quarto ou um escritório cuja porta está fechada, então esse usuário está trabalhando*

Como as regras do tipo Horn possuem no máximo um termo positivo, apenas conjunções de cláusulas são permitidas. Assim, a regra acima precisa ser dividida em duas cláusulas: a primeira referente ao cômodo do tipo `Bedroom` e a segunda cláusula referente ao cômodo do tipo `Office`.

$$\texttt{User(u)} \wedge \texttt{Location(l)} \wedge \texttt{hasLocationType(l,Bedroom)} \wedge \texttt{hasLocation(u,l)} \wedge \texttt{Sensor(s1)} \wedge \texttt{hasDeviceType(s1,SeatOccupancy)} \wedge \texttt{isLocatedIn(s1,l)} \wedge \texttt{hasSensorValue(s1, true)} \wedge \texttt{Sensor(s2)} \wedge \texttt{hasDeviceType(s2,ClosedDoor)} \wedge \texttt{isLocatedIn(s2,l)} \wedge \texttt{hasSensorValue(v2,true)} \implies \texttt{hasActivity(u,Working)}$$

$$\texttt{User(u)} \wedge \texttt{Location(l)} \wedge \texttt{hasLocationType(l,Office)} \wedge \texttt{hasLocation(u,l)} \wedge \texttt{Sensor(s1)} \wedge \texttt{hasDeviceType(s1,SeatOccupancy)} \wedge \texttt{isLocatedIn(s1,l)} \wedge \texttt{hasSensorValue(s1, true)} \wedge \texttt{Sensor(s2)} \wedge \texttt{hasDeviceType(s2,ClosedDoor)} \wedge \texttt{isLocatedIn(s2,l)} \wedge \texttt{hasSensorValue(v2,true)} \implies \texttt{hasActivity(u,Working)}$$

Regra 5 (`WatchingTV`)**.** *Se um usuário encontra-se sentado dentro de algum cômodo que contenha um aparelho de televisão ligado, então esse usuário está assistindo TV.*

$$\texttt{User(u)} \wedge \texttt{Location(l)} \wedge \texttt{hasLocation(u,l)} \wedge \texttt{Sensor(s)} \wedge \texttt{hasDeviceType(s,SeatOccupancy)} \wedge \texttt{isLocatedIn(s,l)} \wedge \texttt{hasSensorValue(s, true)} \wedge \texttt{Actuator(a)} \wedge \texttt{hasDeviceType(a,SmartTV)} \wedge \texttt{isLocatedIn(a,l)} \wedge \texttt{hasActuatorState(a,true)} \implies \texttt{hasActivity(u,WatchingTV)}$$

Regra 6 (`Showering`)**.** *Se um usuário encontra-se em um banheiro que esteja com a porta fechada e o chuveiro ligado, então esse usuário está tomando banho.*

$$\texttt{User(u)} \wedge \texttt{Location(l)} \wedge \texttt{hasLocationType(l,Bathroom)} \wedge \texttt{hasLocation(u,l)} \wedge \texttt{Sensor(s1)} \wedge \texttt{hasDeviceType(s1,ClosedDoor)} \wedge \texttt{isLocatedIn(s1,l)} \wedge \texttt{hasSensorValue(s1, true)} \wedge \texttt{Sensor(s2)} \wedge \texttt{hasDeviceType(s2,ShowerFlowMeter)} \wedge \texttt{isLocatedIn(s2,l)} \wedge \texttt{hasSensorValue(s2,true)} \implies \texttt{hasActivity(u,Showering)}$$

Regra 7 (`Eating`)**.** *Se um usuário encontra-se em uma sala de jantar e está sentado à mesa, então esse usuário está comendo.*

$$\texttt{User(u)} \wedge \texttt{Location(l)} \wedge \texttt{hasLocationType(l,DiningRoom)} \wedge \texttt{hasLocation(u,l)} \wedge \texttt{Sensor(s)} \wedge \texttt{hasDeviceType(s,TableOccupancy)} \wedge \texttt{isLocatedIn(s,l)} \wedge \texttt{hasSensorValue(s, true)} \implies \texttt{hasActivity(u,Eating)}$$

Essa regra poderia ser melhorada utilizando uma estratégia baseada na contagem de tempo entre eventos consecutivos. Por exemplo, a determinação de que o usuário está realmente comendo poderia usar como condição o fato do usuário ter aberto a porta da geladeira ou dos armários utilizados para armazenar os alimentos alguns minutos antes do usuário sentar-se à mesa. Como a detecção das AVD em sistemas sensíveis ao contexto é, por si só, uma grande e desafiadora área de pesquisa e não é o foco principal deste trabalho, optou-se por utilizar uma versão simplificada da regra.

Regra 8 (`Dressing`)**.** *Se um usuário encontra-se em um quarto que esteja com a porta fechada e com as portas do guarda-roupas abertas, então esse usuário está vestindo-se.*

$$\texttt{User(u)} \wedge \texttt{Location(l)} \wedge \texttt{hasLocationType(l,Bedroom)} \wedge \texttt{hasLocation(u,l)} \wedge \texttt{Sensor(s1)} \wedge \texttt{hasDeviceType(s1,WardrobeClosedDoor)} \wedge \texttt{isLocatedIn(s1,l)} \wedge \texttt{hasSensorValue(s1, false)} \wedge \texttt{Sensor(s2)} \wedge \texttt{hasDeviceType(s2,ClosedDoor)} \wedge \texttt{isLocatedIn(s2,l)} \wedge \texttt{hasSensorValue(s2,true)} \implies \texttt{hasActivity(u,Dressing)}$$

Novamente, esta regra pode ser melhorada utilizando a contagem de tempo entre eventos consecutivos, mas optou-se por adotar uma versão simplificada.

Regra 9 (`TakingMedicine`)**.** *Se um usuário encontra-se em um cômodo que possui um sensor do tipo DrugDispenser e esse sensor está ativado, então esse usuário está tomando remédio.*

$$\texttt{User(u)} \wedge \texttt{Location(l)} \wedge \texttt{hasLocation(u,l)} \wedge \texttt{Sensor(s)} \wedge \texttt{hasDeviceType(s,DrugDispenser)} \wedge \texttt{isLocatedIn(s,l)} \wedge \texttt{hasSensorValue(s, true)} \implies \texttt{hasActivity(u,TakingMedicine)}$$

Pode-se observar que devido a maneira em que a ontologia foi projetada, as regras tornam-se independentes do ambiente onde o sistema é utilizado, tornando possível sua utilização em residências com diferentes número de cômodos e dispositivos. Vale ressaltar que o método de inferência escolhido permite que as informações inferidas por uma regra sejam utilizadas no corpo de outras regras, permitindo uma melhor organização das relações de causa-efeito. Um exemplo é o termo `hasLocation(User,Location)`, inferido através da Regra 1 e utilizado em todas as regras que inferem a atividade do usuário.

4.2.3 Inferência das preferências dependentes de contexto

Algumas preferências tendem a mudar de acordo com o ambiente em que o usuário encontra-se e com a atividade sendo realizada. Assim, um terceiro conjunto de regras é estabelecido a fim de ajustar corretamente os valores padrões previamente inferidos, obtendo valores adequados e específicos para o contexto atual do usuário.

O ajuste das preferências representadas por valores numéricos é realizado através da multiplicação do valor da preferência padrão por um coeficiente específico. Em outros

Tabela 3: Coeficientes utilizados no ajuste das preferências dependentes de contexto

Preferência	Atividade					Ambiente
	Cooking	Sleeping	Working	Watching	Bathing	LightingLevel
AmbientLight.	-	0	-	0.5	-	-
AmbientTemp.	-	0.8	-	-	-	-
MediaType	-	audio	-	visual	-	-
MediaVolume	1.5	0.5	0.5	-	2	-
ScreenBrightness	-	-	-	-	-	0.5 + Level / 10
MobilityDevSpd.	-	-	-	-	-	0.5 + Level / 10

casos, como na preferência `MediaType`, o valor é simplesmente trocado por um outro literal. Ainda, algumas preferências podem ser ajustadas de acordo com o estado do ambiente onde o usuário encontra-se. Nesses casos, o coeficiente é representado através de uma função matemática que utiliza as informações do ambiente. A Tabela 3 apresenta os valores utilizados no ajuste das preferências dependentes de contexto, onde a ausência do valor indica que a preferência não sofre ajuste para aquele contexto.

Para garantir uma melhor qualidade de sono do usuário, a luminosidade do ambiente deve ser reduzida ao máximo e a temperatura levemente diminuída. Ainda, é natural que os avisos sejam apresentados na forma de áudio com um volume reduzido.

Em atividades onde o usuário tem sua capacidade de audição reduzida, como durante o banho ou enquanto cozinha, o volume utilizados nos aparelhos de mídia deve ser aumentado consideravelmente. Por outro lado, algumas atividades, como a leitura de um livro, demandam uma maior concentração do usuário. Assim, é natural que o volume do ambiente seja reduzido.

Enquanto o usuário assiste à televisão, a exibição de avisos torna-se mais eficiente caso seja feita de maneira visual e um maior conforto pode ser garantido reduzindo a luminosidade do ambiente. Ainda, utilizando a função apresentada na Tabela 3, o brilho dos dispositivos de imagem é ajustado de acordo com o nível de iluminação no ambiente, facilitando a visualização do conteúdo sendo mostrado no dispositivo. De maneira similar, a velocidade máxima dos dispositivos de auxilio de mobilidade é ajustada de forma a garantir segurança ao usuário.

4.3 Arquitetura do sistema

Essa seção apresenta uma proposta de arquitetura distribuída a ser utilizada em ambientes inteligentes que possuem como principal objetivo o aumento da qualidade de vida de seus usuários através da disponibilização de assistência personalizada. Para facilitar a leitura do trabalho, em um primeiro momento é apresentada uma breve descrição dos componentes dessa arquitetura, conforme mostra a Figura 18.

Gerenciador de Perfis: mantém os perfis dos usuários. É responsável por coletar os dados do usuário, inferir as preferências a serem utilizadas nos serviços do ambiente e fornecer as informações do usuário;

Gerenciador de Contexto: responsável por gerenciar as informações referentes ao contexto do usuário. Mantém os estados dos dispositivos do ambiente atualizados, além de inferir e disponibilizar o contexto do usuário;

Aplicativo de Assistência: utiliza as informações disponibilizadas pelo Gerenciador de Perfil e pelo Gerenciador de Contexto de forma a fornecer algum tipo de assistência ao usuário, como controlar a temperatura ambiente ou informar situações de perigo;

Interface SOA: realiza a comunicação entre os componentes do sistema de assistência personalizada com a arquitetura orientada a serviços.

Cada componente presente na arquitetura proposta é implementado na forma de um agente e as funções a serem realizadas pelo componente são implementadas como comportamentos (*behaviours*) desses agentes. A seguir maiores detalhes referentes aos agentes e seus comportamentos são apresentados.

4.3.1 Gerenciador de Contexto

A principal função do Gerenciador de Contexto é disponibilizar o contexto atual do usuário e do ambiente de maneira adequada. Como dito anteriormente, parte do modelo de contexto é inferido a partir dos dados dos dispositivos presentes no ambiente. Assim, é de extrema importância que os indivíduos da ontologia que representam os dispositivos estejam sempre atualizados, refletindo o estado do dispositivo físico. Isto é feito através do recebimento e interpretação das mensagens enviadas pelo Agente de Interface SOA, que informa o Gerenciador de Contexto a cada mudança de estado dos dispositivos. Ainda,

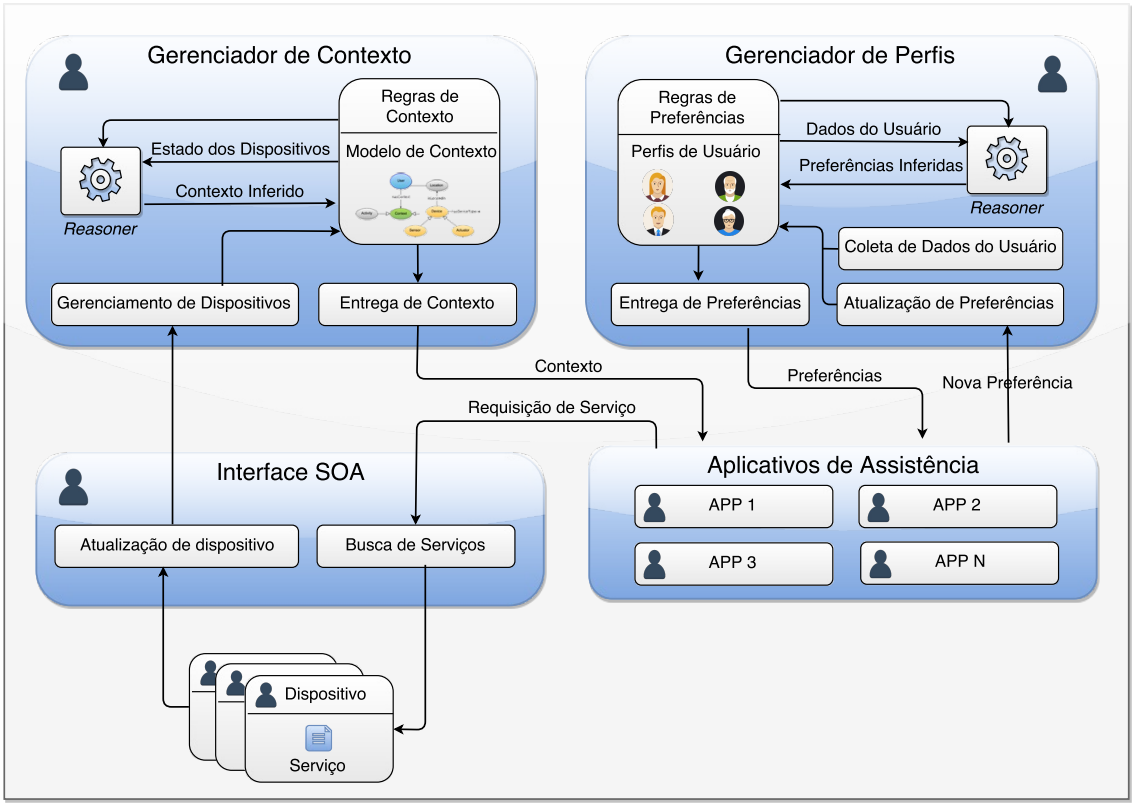


Figura 18: Arquitetura do sistema assistência personalizada. Fonte: o autor (2016)

caso um dispositivo seja inserido ou removido do ambiente, o Gerenciador de Contexto insere ou remove da ontologia o individuo que representa o dispositivo e as propriedades que o descrevem, garantindo que o Modelo de Contexto mantenha-se sempre consistente.

Uma vez que todos os dispositivos do ambiente encontram-se atualizados no modelo de contexto, o *reasoner* semântico é capaz de inferir corretamente as informações relacionadas ao contexto do ambiente e do usuário a cada requisição de contexto realizada por outro agente do sistema. As informações inferidas neste processo são cruciais no sistema proposto, pois grande parte das tomadas de decisão em relação a requisição de serviços dependem da localização e/ou atividade do usuário.

4.3.2 Gerenciador de Perfis

O Gerenciador de Perfis é o principal componente da arquitetura proposta. Sua principal responsabilidade é disponibilizar as informações armazenadas nos perfis de usuário, que podem ser vistos como instâncias do Modelo de Usuário para um usuário em específico.

Para que isso seja possível, em um primeiro momento o Gerenciador de Perfis deve

coletar os dados do usuário necessários para popular os perfis de usário. Esses dados podem ser adquiridos de diversas maneiras: informações básicas como nome, data de nascimento e endereço de residência podem ser inseridos pelo próprio usuário ou cuidador através de um formulário eletrônico ou ainda serem coletadas automaticamente através de um aplicativo para smartphone integrado a alguma rede social em que o usuário seja cadastrado. Além disso, esses dados podem ser obtidos de uma fonte externa como o sistema de gestão utilizado pela equipe médica do usuário que utilize a mesma ontologia ou uma ontologia alinhada[4]. Isso mostra, mais uma vez, a importância do uso de tecnologias da Web Semântica em conjunto com o paradigma multiagentes: enquanto o primeiro garante a interoperabilidade e o reuso de informação, o segundo traz grande facilidade em termos de comunicação, visto que qualquer agente pertencente a outros sistemas é capaz de enviar os dados necessários de maneira simples, bastando conhecer os parâmetros da mensagem.

Uma estratégia interessante para a coleta dos dados relacionados às condições de saúde e limitações físicas e cognitivas é a utilização de um dos instrumentos propostos pela OMS: o *ICF checklist* (World Health Organization, 2001) e o WHODAS (ÜSTÜN, 2010). A primeira opção é um questionário relativamente simples que permite a identificação e quantificação do perfil funcional do individuo utilizando as categorias do ICF mais relevantes na prática clínica (EWERT et al., 2004). A segunda é um instrumento genérico de avaliação baseado em um amplo conjunto de itens do ICF que são suficientemente confiáveis e precisos para fornecer uma medida global em termos de condição de saúde e deficiência (GARIN et al., 2010). Uma característica importante do WHODAS que o difere de outros métodos é a sua direta associação com o ICF, tornando-o uma estratégia bastante interessante para a coleta de todos as informações relacionadas às condições de saúde e limitações em um curto espaço de tempo[5].

Outra importante função desempenhada pelo Gerenciador de Perfis é a atualização das preferências do usuário, que podem ser informadas pelos Aplicativos de Assistência ou por sistemas externos que utilizem técnicas específicas para o aprendizado das preferências e necessidades de usuário. Os valores informados possuem uma prioridade mais alta comparada aos valores inferidos. Assim, quando um Aplicativo de Assistência solicita

[4]Ontologias alinhadas são semanticamente equivalentes e portanto, capazes de interoperar

[5]De acordo com a OMS, a versão contendo 12 itens necessita de um tempo médio de entrevista de apenas cinco minutos

uma certa preferência, o Gerenciador de Perfis inicialmente verifica se algum valor para aquela preferência foi previamente informado; se positivo, esse valor é entregue; se não, o mecanismo de inferência é chamado e o valor inferido é entregue.

4.3.3 Aplicativos de Assistência

Enquanto o Gerenciador de Perfis e o Gerenciador de Contexto coletam, geram e fornecem informações sobre o usuário e seu contexto, eles não são responsáveis por controlar o ambiente. Todas as tomadas de decisão referentes ao controle do ambiente são realizadas pelos Aplicativos de Assistência, que solicitam as a informações referentes ao usuário e seu contexto, realizam o processamento dessas informações e requisitam os serviços do ambiente da maneira necessária a fim de cumprir uma função específica, como mostra a Figura 19.

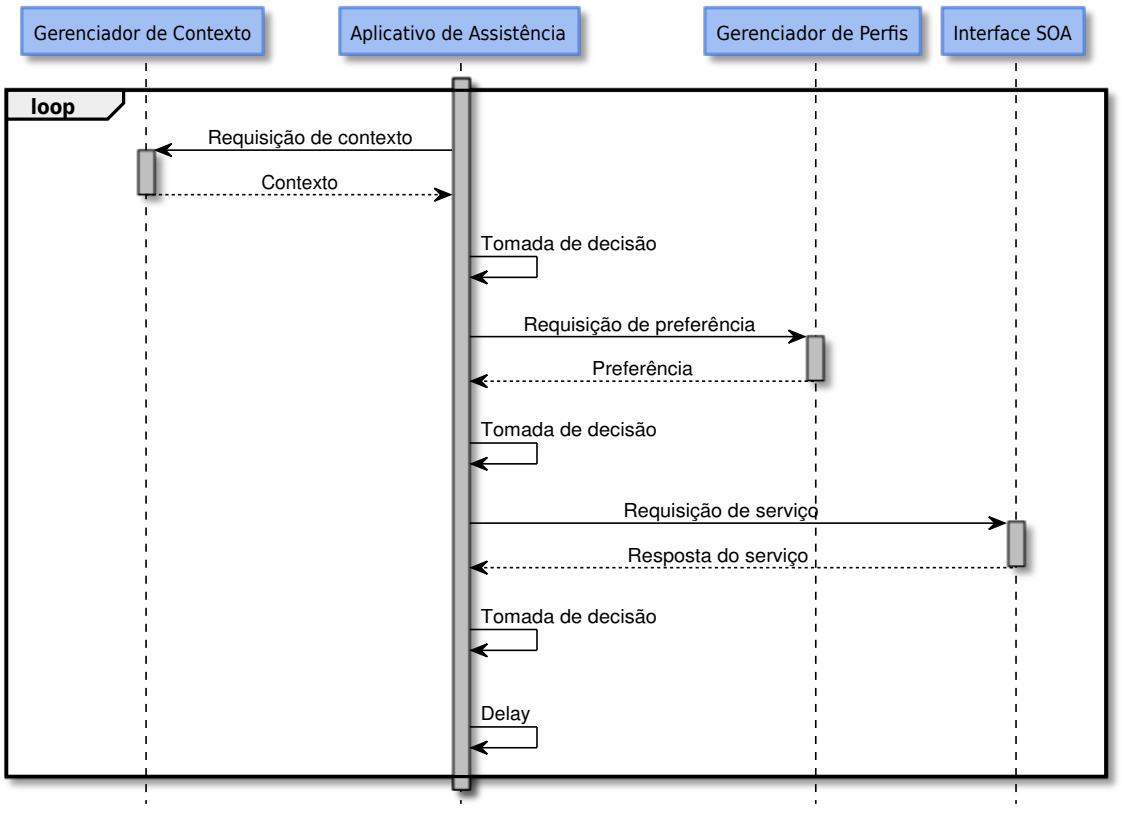


Figura 19: Diagrama de sequência da interação básica entre os agentes do sistema. Fonte: o autor (2016)

Essa separação entre o fornecimento das informações do usuário e seu contexto e a tomada de decisões fornece uma série de vantagens. Primeiramente, como cada funcionalidade é implementada através de um Aplicativo de Assistência específico, a adição ou remoção de funcionalidades pode ser realizada de forma dinâmica em tempo de execução.

Além disso, novos aplicativos podem ser criados por outros desenvolvedores sem a necessidade de conhecer o funcionamento interno do sistema. Outra característica interessante é que um sistema supervisório pode ser implementado na forma de um Aplicativo de Assistência, uma vez que todos os aplicativos são capazes de coletar as informações do ambiente.

4.3.4 Integração com Arquitetura Orientada a Serviços

O agente Interface SOA é responsável pela integração entre o sistema de assistência personalizada e a arquitetura orientada a serviços do ambiente, possuindo grande importância na arquitetura proposta. A principal função do agente Interface SOA é interpretar as requisições recebidas dos Aplicativos de Assistência e realizar a busca dos serviços necessários entre os dispositivos presentes no ambiente[6]. Caso o serviço requisitado encontre-se disponível, o serviço é executado de acordo com os parâmetros enviados pelo Aplicativo de Assistência. A segunda função do agente Interface SOA é verificar os estados dos dispositivos presentes no ambiente e informar qualquer mudança ao Gerenciador de Contexto, incluindo a adição ou remoção de dispositivos.

[6] A forma em que os serviços são descritos e a estratégia de busca de serviços não fazem parte do escopo desse trabalho. Maiores informações relacionadas à arquitetura orientada a serviços podem ser encontradas em (PEDROSO, 2016)

5 IMPLEMENTAÇÃO

Este capítulo apresenta a etapa de implementação do Sistema de Assistência Personalizado proposto, realizada em duas etapas. Primeiramente, a ontologia e as regras de inferência foram implementadas no *software* Protégé a partir da modelagem conceitual descrita no capítulo 4. Em seguida, o sistema foi implementado utilizando a linguagem Java, com o auxilio de três bibliotecas: (1) a plataforma Jade, utilizada para implementar os agentes do sistema; (2) o *reasoner* semântico Pellet, utilizado no processo de inferência; (3) a OWL API, utilizada para permitir que a ontologia desenvolvida possa ser manipulada no código em Java. A Figura 20 ilustra os principais componentes utilizados na implementação.

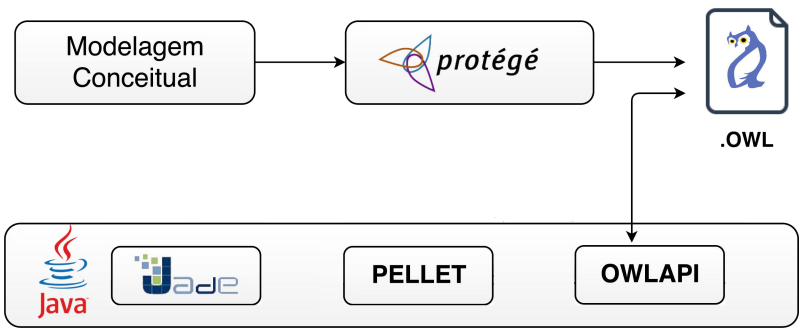


Figura 20: Componentes de software utilizados na implementação. Fonte: o autor (2016)

5.1 Implementação da ontologia

Esta seção apresenta a etapa de implementação da ontologia proposta no Capítulo 4 utilizando a linguagem OWL. Primeiramente a ferramenta de desenvolvimento utilizada nesta etapa é apresentada.

5.1.1 Ferramenta de desenvolvimento

Para auxiliar no desenvolvimento da ontologia proposta no Capítulo 4 foi utilizada a ferramenta Protégé versão 4.3 (STANFORD, 2015), desenvolvida pelo grupo de pesquisa *Stanford Medical Informatics* da escola de medicina da Universidade de Stanford. É um *software* livre, *open-source*, desenvolvido em JAVA e suportado por uma extensa comunidade de desenvolvedores.

Através da sua interface gráfica é possível implementar todos os aspectos da ontologia sem a necessidade de escrever códigos na linguagem OWL. A Figura 21 apresenta a interface gráfica customizada do *software* Protégé contendo as janelas utilizadas para editar e visualizar (1) a hierarquia das classes, (2) as propriedades de objeto, (3) as propriedades de dados, (4) os indivíduos e (5) as regras SWRL. Ainda, a partir do uso de *plugins* é possível adicionar novos recursos, tais como a visualização gráfica da ontologia e a checagem de consistência através de um *reasoner* semântico.

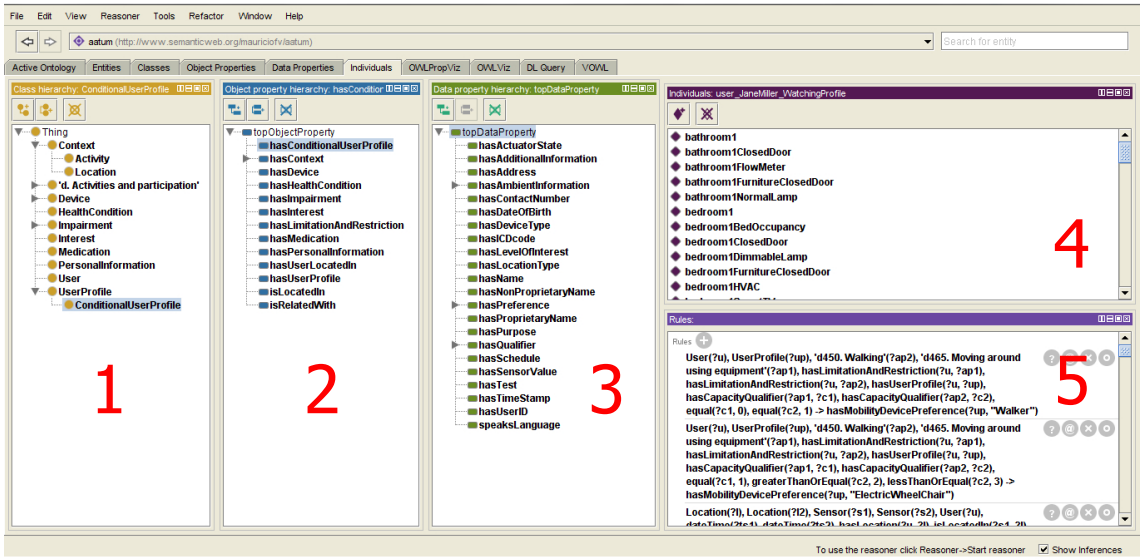


Figura 21: Interface do software Protégé 4.3. Fonte: o autor (2016)

5.1.2 Criação das classes

A hierarquia de classes da ontologia do modelo de usuário foi criada utilizando a interface gráfica apresentada anteriormente. Após todas as informações pertinentes às classes serem inseridas, a ferramenta gera o código OWL de maneira automática, como pode ser observado no código que define a classe `ConditionalUserProfile`, mostrado na Figura 22.

```
<Class rdf:about="#ConditionalUserProfile">
    <rdfs:subClassOf rdf:resource="#UserProfile"/>
</Class>
```

Figura 22: Código XML para a definição da classe `ConditionalUserProfile`. Fonte: o autor (2016)

5.1.3 Criação das propriedades

Assim como no processo de criação das classes, a implementação das propriedades de dados e de objetos é feita através da interface gráfica. O trecho de código gerado pelo software Protégé definindo a propriedade de objeto `hasPersonalInformation` e suas restrições em relação ao domínio, escopo e cardinalidade é apresentado na Figura 23.

```
<ObjectProperty rdf:about="#hasPersonalInformation">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:range rdf:resource="#PersonalInformation"/>
    <rdfs:domain rdf:resource="#User"/>
</ObjectProperty>
```

Figura 23: Código XML para a definição da propriedade `hasPersonalInformation`. Fonte: o autor (2016)

De maneira análoga, a Figura 24 apresenta a implementação da propriedade de dados `hasMediaVolumePreference`, incluindo suas restrições quanto ao domínio (`UserProfile`) e ao escopo (`integer[>= 0 , <= 10]`).

```
<DatatypeProperty rdf:about="#hasMediaVolumePreference">
  <rdfs:domain rdf:resource="#UserProfile"/>
  <rdfs:subPropertyOf rdf:resource="#hasPreference"/>
  <rdfs:range>
     <rdfs:Datatype>
        <onDatatype rdf:resource="&xsd;integer"/>
        <withRestrictions rdf:parseType="Collection">
           <rdf:Description>
              <xsd:maxInclusive rdf:datatype="&xsd;integer">10</xsd:maxInclusive>
           </rdf:Description>
           <rdf:Description>
              <xsd:minInclusive rdf:datatype="&xsd;integer">0</xsd:minInclusive>
           </rdf:Description>
        </withRestrictions>
     </rdfs:Datatype>
  </rdfs:range>
</DatatypeProperty>
```

Figura 24: Código XML para a definição da propriedade `hasMediaVolumePreference`. Fonte: o autor (2016)

5.1.4 Criação dos indivíduos

A interface gráfica do software Protégé também permite a criação dos indivíduos e a associação de suas propriedades. Como explicado no Capítulo 4, os indivíduos que representam as Atividades de Vida Diária devem ser criados durante o desenvolvimento da ontologia. A Figura 25 apresenta o trecho de código que representa a atividade `Cooking`.

```
<NamedIndividual rdf:about="#cooking">
  <rdf:type rdf:resource="#Activity"/>
</NamedIndividual>
```

Figura 25: Código XML para a criação do indivíduo representando a atividade `Cooking`. Fonte: o autor (2016)

5.2 Implementação das regras de inferência

Esta seção apresenta como as regras utilizadas no processo de inferência foram implementadas de acordo com as recomendações do W3C. Em um primeiro momento as funções matemáticas, árvores de decisão e cláusulas de Horn propostas na Seção 4.2 são transformadas em regras escritas na linguagem SWRL. Essas regras são então inseridas em uma interface específica do software Protégé que, após verificar a sintaxe das regras, gera o código OWL correspondente. Esse processo é ilustrado pela Figura 26.

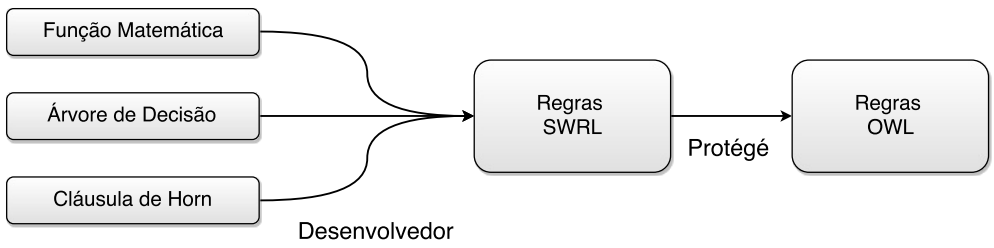


Figura 26: Processo de implementação das regras de inferência. Fonte: o autor (2016)

O processo de implementação das relações definidas através das cláusulas de Horn é realizado de maneira simples e direta, uma vez que a linguagem SWRL baseia-se nesse tipo de cláusula. A Figura 27 exemplifica como a cláusula de Horn que determina a localização do usuário pode ser escrita em SWRL, enquanto a Figura 28 ilustra a representação em OWL dessa regra, gerada automaticamente pelo software Protégé.

As relações descritas por funções matemáticas podem ser implementadas utilizando

```
User(?u), Location(?l), hasUserID(?u,?id), Sensor(?s),
hasDeviceType(?s,"UserIdentification"), isLocatedIn(?s,?l),
hasSensorValue(?s,?v), equal(?v,?id) -> hasLocation(?u,?l)
```

Figura 27: Regra em SWRL para a determinação da localização do usuário. Fonte: o autor (2016)

```
<DLSafeRule>
  <Body>
    <ClassAtom>
      <Class IRI="#Location"/>
      <Variable IRI="urn:swrl#l"/>
    </ClassAtom>
    <ClassAtom>
      <Class IRI="#Sensor"/>
      <Variable IRI="urn:swrl#s"/>
    </ClassAtom>
    <ClassAtom>
      <Class IRI="#User"/>
      <Variable IRI="urn:swrl#u"/>
    </ClassAtom>
    <ObjectPropertyAtom>
      <ObjectProperty IRI="#isLocatedIn"/>
      <Variable IRI="urn:swrl#s"/>
      <Variable IRI="urn:swrl#l"/>
    </ObjectPropertyAtom>
    <DataPropertyAtom>
      <DataProperty IRI="#hasDeviceType"/>
      <Variable IRI="urn:swrl#s"/>
      <Literal datatypeIRI="&rdf;PlainLiteral">UserIdentification</Literal>
    </DataPropertyAtom>
    <DataPropertyAtom>
      <DataProperty IRI="#hasSensorValue"/>
      <Variable IRI="urn:swrl#s"/>
      <Variable IRI="urn:swrl#v"/>
    </DataPropertyAtom>
    <DataPropertyAtom>
      <DataProperty IRI="#hasUserID"/>
      <Variable IRI="urn:swrl#u"/>
      <Variable IRI="urn:swrl#id"/>
    </DataPropertyAtom>
    <BuiltInAtom IRI="http://www.w3.org/2003/11/swrlb#equal">
      <Variable IRI="urn:swrl#v"/>
      <Variable IRI="urn:swrl#id"/>
    </BuiltInAtom>
  </Body>
  <Head>
    <ObjectPropertyAtom>
      <ObjectProperty IRI="#hasLocation"/>
      <Variable IRI="urn:swrl#u"/>
      <Variable IRI="urn:swrl#l"/>
    </ObjectPropertyAtom>
  </Head>
</DLSafeRule>
```

Figura 28: Representação em OWL da regra utilizada para a determinação da localização do usuário. Fonte: o autor (2016)

os *built-ins* matemáticos definidos na linguagem SWRL que permitem operações como adição, multiplicação e exponenciação. A Figura 29 exemplifica como a linguagem SWRL é utilizada para estabelecer a preferência em relação ao volume dos dispositivos de acordo

com o nível de dificuldade de audição do usuário, obedecendo da função matemática:

$$MediaVolume(Hearing) = 5 + Hearing$$

```
User(?u), UserProfile(?up), hasUserProfile(?u, ?up),
   'b230.  Hearing'(?i1), hasImpairment(?u, ?i1),
hasExtentOfImpairmentQualifier(?i1, ?e1), add(?s,?e1,5)->
            hasMediaVolumePreference(?up, ?s)
```

Figura 29: Regra SWRL para a estabelecer a preferência relacionada ao volume dos dispositivos. Fonte: o autor (2016)

As relações expressas por árvores de decisão necessitam de múltiplas regras para serem descritas corretamente. De maneira geral, para cada folha da árvore, uma regra SWRL é necessária. Por exemplo, para a preferência `ColorScheme`, quatro regras são necessárias: as três primeiras associam um esquema de cores específico de acordo com o tipo de daltonismo (expresso pela propriedade `hasAdditionalInformation`) presente no usuário, enquanto a ultima associa um esquema de alto contraste caso o usuário não possua daltonismo (`HasExtentOfImpairment = 0`) e possua um alto nível de deterioração da sensibilidade ao contraste. As Figuras 30 e 31 apresentam duas das regras utilizadas na árvore de decisão em questão.

```
User(?u), UserProfile(?up),hasUserProfile(?u,?up),
'b21021.  Colour vision'(?i), hasImpairment(?u,?i),
hasAdditionalInformation(?i,?ai),equal(?ai,"Deuteranopia") ->
hasColorSchemePreference(?up,"DeuteranopiaScheme")
```

Figura 30: Regra SWRL para a estabelecer o esquema de cores para usuários com daltonismo. Fonte: o autor (2016)

```
User(?u), UserProfile(?up),hasUserProfile(?u,?up),
'b21021.  Colour vision'(?i1), hasImpairment(?u,?i1),
hasExtentOfImpairmentQualifier(?i1,?e1),equal(?e1,0),
'b21022.  Contrast sensitivity'(?i2), hasImpairment(?u,?i2),
hasExtentOfImpairmentQualifier(?i2,?e2), greaterThanOrEqual(?e2,3)
-> hasColorSchemePreference(?up,"HighContrastScheme")
```

Figura 31: Regra SWRL para a estabelecer o esquema de cores para usuários com deterioração da sensibilidade ao contraste. Fonte: o autor (2016)

5.3 Manipulação da ontologia

Após a etapa de desenvolvimento da ontologia e a criação do arquivo .OWL utilizando o software Protégé, foi necessário selecionar uma API de alto nível para inserir, editar e consultar as informações ontológicas através de código escrito na linguagem Java. Dentre as opções de API disponíveis, foi escolhida a OWL API 4.0.2 (HORRIDGE; BECHHOFER, 2011) devido a fatores como: (i) extensa documentação, (ii) facilidade na utilização e (iii) suporte a *reasoning* utilizando SWRL.

Simultaneamente à definição da API para a manipulação da ontologia, foi realizada a escolha do *reasoner* semântico a ser utilizado no processo de inferência baseando-se na comparação realizada em (ABBURU, 2012). Devido a características como (i) extensa documentação (ii) compatibilidade com OWL API e (iii) código aberto, o *reasoner* Pellet 2.4 (SIRIN et al., 2007) foi selecionado.

Todas as operações relacionadas à manipulação da ontologia foram implementadas na forma de métodos da classe `OntologyInterface` que por sua vez utiliza os métodos da OWL API a fim de consultar ou alterar os axiomas da ontologia. Essas operações podem ser classificadas em dois grandes grupos: (1) as operações que inserem, removem e alteram os axiomas na base de conhecimento; (2) as operações que consultam as informações presentes na base de conhecimento. As próximas subseções detalham como os dois tipos de operação são realizados no código desenvolvido nesse trabalho.

5.3.1 Manipulação de axiomas

Durante a execução do sistema, informações são inseridas, modificadas e removidas da ontologia. Essas operações são realizadas através da manipulação dos axiomas que representam os indivíduos, suas estruturas taxonômicas e suas relações com outros indivíduos ou dados. A execução dessas etapas é definida por uma ordem de chamadas dos métodos da OWL API. Por exemplo, o processo de adição de uma novo indivíduo obedece a seguinte sequência:

1. Busca-se a referência da classe a qual o novo indivíduo pertence;

2. Cria-se o novo indivíduo;

3. Cria-se o axioma associando o individuo à sua classe;

4. Busca-se as referências dos indivíduos que possuem alguma relação com indivíduo sendo adicionado;

5. Cria-se o(s) axioma(s) relacionando o indivíduo recém criado com o(s) encontrado(s) no passo anterior;

Esse processo é exemplificado através do código apresentado na Figura 32, onde um novo sensor é adicionado à ontologia.

```
// Referência da classe Sensor
OWLClass sensor = dataFactory.getOWLClass("Sensor", prefixManager);
// Criação do indivíduo representando novo sensor
OWLNamedIndividual newSensor = dataFactory.getOWLNamedIndividual(sensorName,
    prefixManager);
// Axioma indicando que newSensor é da classe Sensor
OWLClassAssertionAxiom classAssertion = dataFactory.getOWLClassAssertionAxiom(sensor,
    newSensor);
// Novo axioma é inserido na ontologia
ontologyManager.addAxiom(aatumOntology, classAssertion);
// Referência do indivíduo representando a localização do novo sensor
OWLNamedIndividual locationIndividual =
    dataFactory.getOWLNamedIndividual(sensorLocation, prefixManager);
// Referência da propriedade isLocatedIn
OWLObjectProperty isLocatedIn = dataFactory.getOWLObjectProperty("isLocatedIn",
    prefixManager);
// Novo axioma indicando onde o novo sensor está localizado
OWLObjectPropertyAssertionAxiom propertyAssertion =
    dataFactory.getOWLObjectPropertyAssertionAxiom(isLocatedIn, newSensor,
    locationIndividual);
// Novo axioma é inserido na ontologia
ontologyManager.addAxiom(aatumOntology, propertyAssertion);
```

Figura 32: Manipulação de axiomas utilizando a OWLAPI. Fonte: o autor (2016)

5.3.2 Consulta de axiomas

O outro tipo de operação realizada pela classe `OntologyInterface` é a consulta de informações presentes na ontologia. Essa consulta é realizada a partir de um conjunto de métodos presentes na OWL API que utilizam o *reasoner* semântico para buscar e inferir axiomas na ontologia. O trecho de código apresentado na Figura 33 mostra como a consulta do valor padrão para a preferência `MediaVolume` de um usuário específico é realizada. Utilizando as referências dos indivíduos e propriedades da ontologia, o *reasoner* é acionado a fim de buscar pelos valores e objetos associados às propriedades indicadas.

É importante destacar que o resultado dessa busca é sempre formado pelos valores declarados e pelos valores inferidos. Caso seja necessário diferenciar entre valores declaradas e inferidos, como na consulta de uma preferência do usuário onde valores declarados possuem prioridade sobre valores inferidos, um pós-processamento é necessário, como ilustrado na Figura 34.

```
// Referência do indivíduo representando o usuário
OWLNamedIndividual userIndividual = dataFactory.getOWLNamedIndividual(userID,
    prefixManager);
// Referência da propriedade de objeto hasUserProfile
OWLObjectProperty hasUserProfile = dataFactory.getOWLObjectProperty("hasUserProfile",
    prefixManager);
// Referência da propriedade hasMediaVolumePreference
OWLDataProperty hasMediaVolumePref =
    dataFactory.getOWLDataProperty("hasMediaVolumePreference", prefixManager);
// O reasoner é utilizado para pesquisar qual é o userProfile associado ao usuário em
    questão
OWLNamedIndividual userProfile = reasoner.getObjectPropertyValues(userIndividual,
    hasUserProfile);
// O reasoner é utilizado para consultar os valores associados ao userProfile através
    da propriedade hasMediaVolumePreference
Set<OWLLiteral> preferenceValues = reasoner.getDataPropertyValues(userProfile,
    hasMediaVolumePref);
```

Figura 33: Consulta de axiomas utilizando a OWLAPI. Fonte: o autor (2016)

```
Iterator it = preferenceValues.iterator();
while (it.hasNext()) {
   // Valor retornado pelo reasoner
   preferenceValue = ((OWLLiteral) it.next());
   // Axioma utilizando esse valor é construido
   OWLDataPropertyAssertionAxiom dataPropertyAxiom =
        dataFactory.getOWLDataPropertyAssertionAxiom(hasMediaVolumePref, userProfile,
        preferenceValue);
   // Caso a ontologia contenha o axioma, o valor é utilizado
   if (aatumOntology.containsAxiom(dataPropertyAxiom)) {
     return preferenceValue.getLiteral();
   }
}
```

Figura 34: Verificação da existência de um axioma utilizando a OWLAPI. Fonte: o autor (2016)

Para cada valor da preferência retornado pelo *reasoner*, um axioma é construído e consultado na ontologia. Se o axioma existir, então o valor retornado pelo *reasoner* é um valor declarado e deve ser utilizado. Caso o axioma não exista, então o valor foi inferido pelo *reasoner* e só deve ser utilizado caso não sejam encontrados valores declarados.

5.4 Implementação da arquitetura multiagentes

Essa seção descreve os aspectos relacionados à implementação da arquitetura multiagentes proposta utilizando a plataforma JADE. Em um primeiro momento apresenta-se as implementações dos agentes propostos pela arquitetura, incluindo as questões ligadas à comunicação e a dinâmica dos agentes. Por fim é discutido como a interação entre os agentes do sistema proposto é estabelecida com os agentes presentes na Arquitetura Orientada a Serviços utilizada neste trabalho.

5.4.1 Componentes da arquitetura

As Figuras 35 e 36 apresentam os diagramas de classes dos componentes da arquitetura multiagentes proposta. Cada componente (agente) do sistema é implementado em uma classe específica a partir da superclasse base da plataforma Jade (`Agent`). De maneira similar, os comportamentos dos agentes encarregados de realizarem tarefas específicas relacionadas a busca, adição ou modificação das informações presentes na Base de Conhecimento, são implementados a partir da classe `Behaviour` utilizando os métodos da classe `OntologyInterface`.

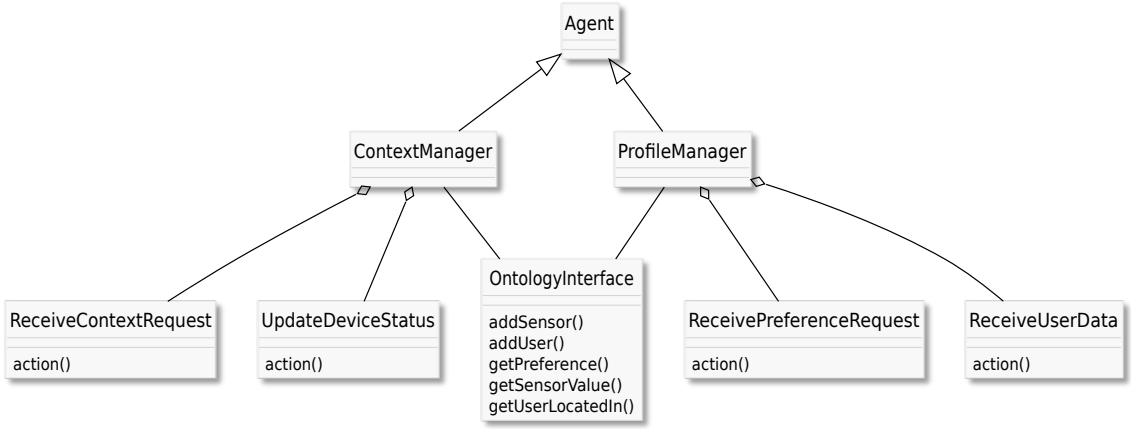


Figura 35: Diagrama de classes dos componentes da arquitetura proposta. Fonte: o autor (2016)

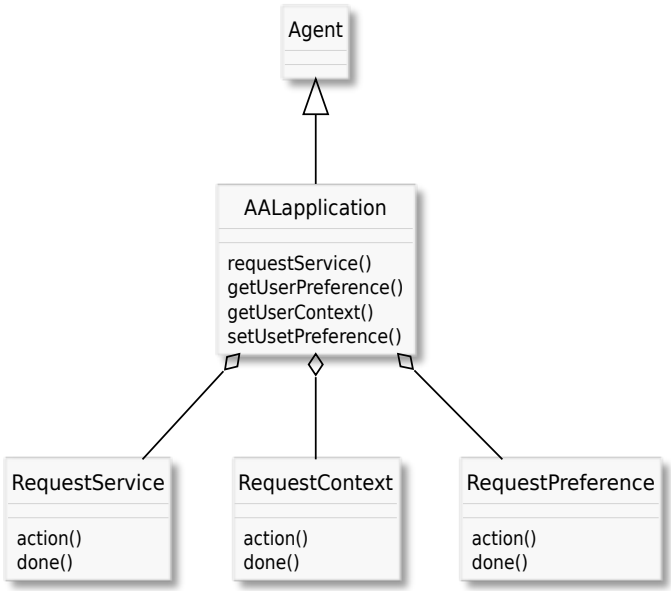


Figura 36: Diagrama de classes dos componentes da arquitetura proposta (continuação). Fonte: o autor (2016)

5.4.2 Dinâmica do sistema

Inicialmente os agentes `ContextManager` e `ProfileManager` realizam seus cadastros no DF para poderem ser posteriormente encontrados pelos agentes `AALapplication` e `SOAInterface`. Isto é realizado através dos métodos disponíveis nas classes `DFAgentDescription` e `ServiceDescription`.

Após a fase inicial de registro no DF, a comunicação entre os agentes do sistema é realizada utilizando o protocolo FIPA-ACL. Esse protocolo permite a configuração de um conjunto de parâmetros da mensagem de forma a garantir uma comunicação funcional e efetiva. Nesse trabalho, além do parâmetro básico `performative` que define a ação realizada pela comunicação, foi utilizado o parâmetro `conversation-id` a fim de estabelecer o tipo da mensagem sendo trocada. Dessa forma, para cada tipo de mensagem recebida pelo agente, um comportamento específico é executado para realizar o processamento necessário. A Figura 37 apresenta os possíveis tipos de mensagens trocados entre os agentes, no formato PERFORMATIVE(conversation-id).

Como pode ser observado, a Interface SOA informa o gerenciador de contexto sobre qualquer evento relacionado com os dispositivos do ambiente por meio de mensagens com performativa INFORM. Ainda, o campo `conversation-id` é utilizado para indicar o tipo de evento ocorrido, isto é, se algum dispositivo foi inserido, removido ou teve seu estado alterado.

Após os Aplicativos de Assistência terem encontrado os agentes Gerenciador de Contexto e Gerenciador de Perfis, o controle do ambiente é realizado utilizando mensagens com performativa REQUEST, que podem ser enviadas tanto para a requisição do contexto `(context-request)`, para a requisição de preferências do usuário `(preference-request)` e para a requisição de serviços no ambiente `(service-request)`. Finalmente, as informações requisitadas são retornadas utilizando mensagens com performativa INFORM e com `conversation-id` igual ao da mensagem recebida.

A identificação desses parâmetros que acompanham as mensagens é realizada através da classe `MessageTemplate` que fornece métodos para ler e comparar os parâmetros presentes na mensagem recebida com um padrão pré estabelecido. Esses padrões são definidos nos construtores das classes que implementam os comportamentos encarregados de realizar o processamento necessário indicado pelo tipo e conteúdo da mensagem. Uma vez

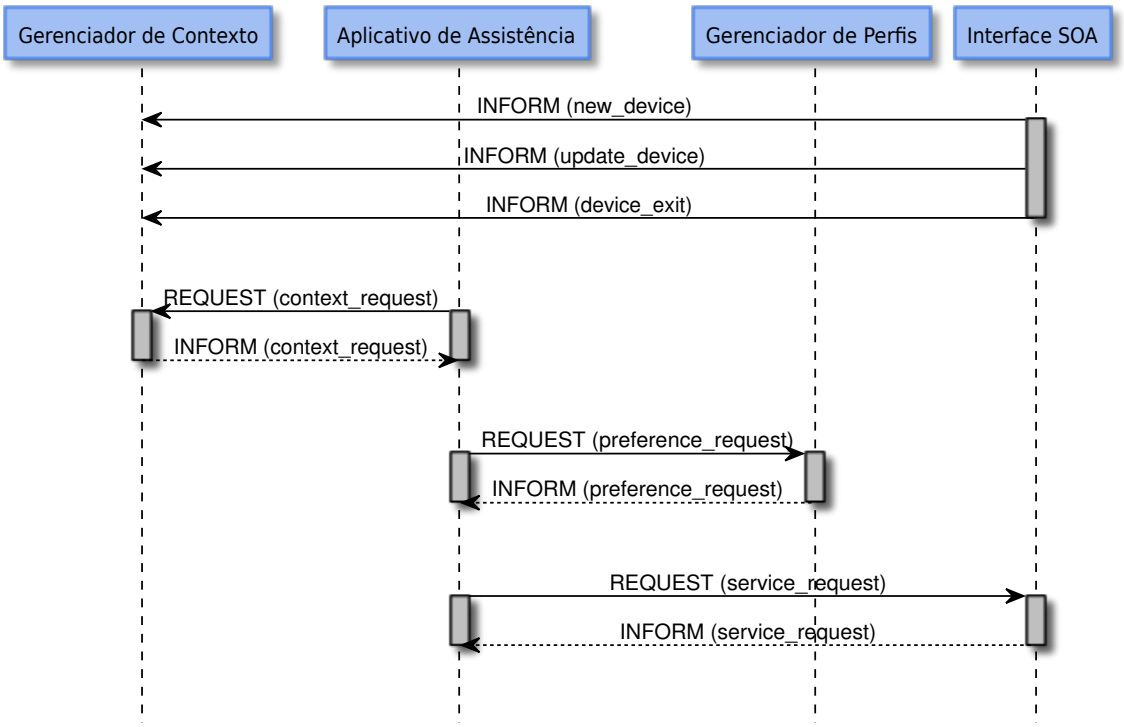


Figura 37: Tipos de mensagens trocados entre os agentes da arquitetura proposta. Fonte: o autor (2016)

que uma mensagem contendo os parâmetros corretos (para um comportamento em específico) foi recebida, o conteúdo da mensagem - representada por uma String - é interpretado de acordo com um padrão pre-definido. Por exemplo, uma das possíveis requisições tratadas pelo comportamento `ReceiveContextRequest` é em relação a qual usuário encontra-se em um determinado cômodo. Essa requisição é representada pela String "getUserLocatedIn,nomeDoComodo". Uma vez que o conteúdo da mensagem é identificado, o processamento é realizado através dos métodos da classe `OntologyInterface`, que realiza a busca ou modificação dos axiomas da Base de Conhecimento. Por fim, a resposta apropriada é enviada para o Aplicativo de Assistência que realizou a requisição.

O trecho de código apresentado na Figura 38 exemplifica a etapa de identificação e tratamento das mensagens para o comportamento `ReceiveContextRequest`, que utiliza como padrão o parâmetro `performative` igual a `REQUEST` e o parâmetro `conversation-id` igual a `context_request`.

5.4.3 Aplicativos de assistência

Para que o Sistema de Assistência Personalizada proposto neste trabalho possa ser utilizado nos mais diferentes cenários dentro do âmbito dos Ambientes Assistidos, as

```
private class ReceiveContextRequest extends CyclicBehaviour {
    MessageTemplate mt;
    String request;
    String userID = null;
    String locationName = null;
    String response = null;
    .
    .

    public ReceiveContextRequest() {
       mt =
            MessageTemplate.and(MessageTemplate.MatchPerformative(ACLMessage.REQUEST),
            MessageTemplate.MatchConversationId("context_request"));

    }

    @Override
    public void action() {
       ACLMessage msg = myAgent.receive(mt);
       if (msg != null) {
          request = msg.getContent().split(",");
          // Identifica o conteúdo da mensagem
          switch (request[0]) {
             case "getUserLocatedIn":
                     locationName = request[1];
                     userID = ontologyInterface.getUserLocatedIn(locationName);
                     response = userID;
                     break;
             .
             .
          }
          // Após ação ter sido executada, resposta é enviada
          ACLMessage reply = msg.createReply();
          reply.setPerformative(ACLMessage.INFORM);
          reply.setContent(response);
          send(reply);
       }
    }
}
```

Figura 38: Tratamento das mensagens para o comportamento `ReceiveContextRequest`. Fonte: o autor (2016)

funcionalidades fornecidas pelo sistema são implementadas através dos Aplicativos de Assistência. A estratégia de implementação em termos do número de aplicativos é flexível, podendo ser escolhida pelo desenvolvedor do aplicativo de acordo com o caso de uso. Por exemplo, cada aplicativo pode implementar uma ou mais funcionalidades, ou ainda múltiplos aplicativos podem ser combinados para oferecerem uma única funcionalidade.

Esses Aplicativos de Assistência são agentes implementados a partir do agente `AALapplication`, como mostra a Figura 39. A classe `AALapplication` apresenta um conjunto de métodos que implementa a comunicação entre os aplicativos e os outros componentes do sistema de acordo com o a estratégia descrita anteriormente. Dessa forma, as classes que representam os Aplicativos de Assistência podem utilizar esses métodos para implementar suas funcionalidades de acordo com o desejado de forma transparente em relação ao funcionamento interno do sistema. O trecho de código apresentado na Figura 40

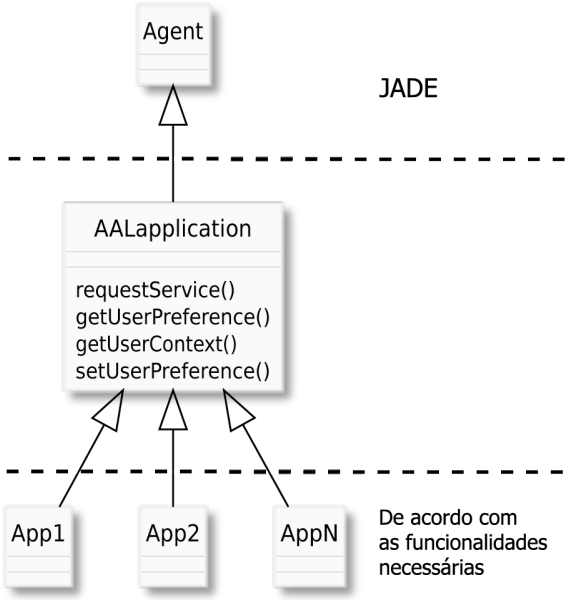


Figura 39: Diagrama de classe referente aos Aplicativos de Assistência. Fonte: o autor (2016)

```
// Busca o nome dos cômodos da residência
String locations[] = getLocations();
// Percorre os cômodos
for (int i = 0; i < locations.length; i++) {
  // Obtém o usuário presente no cômodo e seu contexto
  String user = getUserLocatedIn(locations[i]);
  String userContext = getUserContext(user);
  .
  .
  // Obtém as preferências do usuário
  lightingPreference = getAmbientLightingPreference(user, userContext);
  temperaturePreference = getAmbientTemperaturePreference(user, userContext);
  // Requisita os serviços para ajustar a luminosidade e a temperatura do cômodo
  requestService("setAmbientLighting",locations[i], lightingPreference);
  requestService("setAmbientTemperature",locations[i], temperaturePreference);
  .
  .
}
```

Figura 40: Implementação do Aplicativo de Assistência para controle de luminosidade e temperatura. Fonte: o autor (2016)

exemplifica como um Aplicativo de Assistência com a finalidade de ajustar a luminosidade e a temperatura do ambiente de acordo com o usuário e seu contexto é implementado.

5.4.4 Integração com a Arquitetura SOA

A interação entre o Sistema de Assistência Personalizada proposto e a Arquitetura Orientada a Serviços do ambiente acontece pelas requisições realizadas pelos Aplicativos de Assistência na forma de mensagens que obedecem o padrão comentado anteriormente e são recebidas e executadas pelo agente `Interface SOA` presente na Arquitetura Orientada a Serviços proposta em (PEDROSO, 2016). Nesta arquitetura o agente `Interface`

SOA é responsável por realizar a comunicação com os componentes consumidores dos serviços e por orquestrar, compor e buscar os serviços fornecidos pelo grupo de agentes que representam os dispositivos encontrados no ambiente e seus serviços. A implementação destes agente não faz parte desse trabalho e, portanto, maiores informações devem ser buscadas em (PEDROSO, 2016).

6 VALIDAÇÃO DA PROPOSTA

Esse capítulo apresenta a validação do sistema desenvolvido através de sua utilização no fornecimento de assistência personalizada nas AVD de usuários de uma casa inteligente. Inicialmente o ambiente onde o sistema foi utilizado é detalhado.

6.1 Infraestrutura utilizada

Idealmente, para que todas as funcionalidades do sistema pudessem ser testadas e demonstradas, seria necessário uma casa inteligente completa, contendo uma infraestrutura composta por uma ampla gama de sensores e atuadores, uma estrutura de rede e um *middleware* que possibilitasse a comunicação com os componentes físicos do sistema de forma transparente.

Buscando viabilizar a utilização de um ambiente que fosse o mais próximo possível do ideal utilizando os recursos disponíveis, optou-se pela adoção de uma infraestrutura híbrida, onde parte dos dispositivos é simulado por um software e outra parte encontra-se fisicamente vinculada a uma solução comercial de automação residencial. A Figura 41 ilustra a organização em camadas dos componentes da infraestrutura.

Uma vez que a utilização de um simulador permite a customização do cenário onde a demonstração é realizada, neste trabalhou optou-se por criar um cenário baseado no laboratório DOMUS, descrito na próxima subseção.

6.1.1 Laboratório DOMUS

O laboratório DOMUS[1], localizado no departamento de Ciências da Computação da *Université de Sherbrooke*, Canadá, é uma referência mundial em relação a pesquisas sobre casas inteligentes, possuindo mais de cento e quarenta publicações ao longo de quatorze

[1] http://domus.usherbrooke.ca/

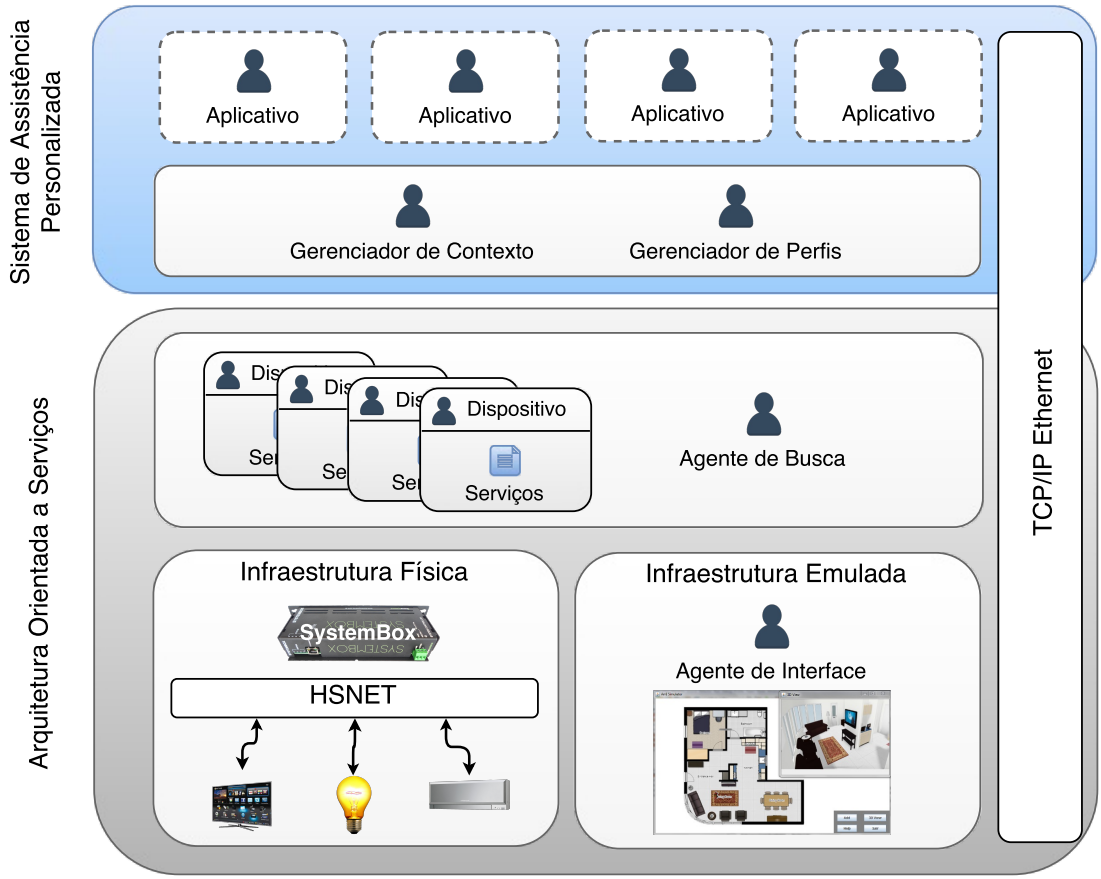


Figura 41: Camadas da infraestrutura utilizada na validação do trabalho. Fonte: o autor (2016)

anos de existência. O laboratório possui um apartamento padrão contendo cozinha, sala de estar, sala de jantar, um dormitório e um banheiro e é equipado com uma série de dispositivos como sensores infravermelho, sensores de pressão, sensores de contato nas portas, sistema de áudio e vídeo e sensores RFID para a obtenção da posição do usuário e dos objetos dentro do apartamento (KADOUCHE et al., 2010). Uma visão parcial da sala de estar, cozinha e sala de jantar é apresentada na Figura 42.

Figura 42: Foto panorâmica da casa inteligente localizada no laboratório DOMUS. Fonte: (Centre de recherche sur le vieillissement, 2015)

6.1.2 Simulador AmI

Para que outros dispositivos além dos disponíveis fisicamente pudessem ser utilizados na validação do sistema, optou-se pelo uso de um software simulador que possibilitasse a customização dos cômodos da residência simulada e permitisse a simulação e o controle dos seus dispositivos através de uma API. Como nenhuma das opções encontradas na literatura (LEI et al., 2010; PARK et al., 2007; VAN NGUYEN; NGUYEN; CHOI, 2010; BOUCHARD et al., 2012) satisfez esses requerimentos, foi desenvolvido, juntamente com o aluno do PPGEE Diego Gimenez Pedroso (PEDROSO, 2016), um software simulador de casas inteligentes customizável e flexível o suficiente para atender as demandas desse trabalho.

Como a Arquitetura Orientada a Serviços utilizada nesse trabalho é implementada utilizando o paradigma multiagentes, ela apresenta a capacidade de auto-organização. Dessa forma, caso um dispositivo seja adicionado ou removido do ambiente, a disponibilização dos serviços é dinamicamente ajustada de acordo com os serviços individuais de cada dispositivo. Para que esse comportamento também fosse observado no ambiente simulado, o software simulador desenvolvido possui a mesma estrutura da Arquitetura Orientada a Serviços, ou seja, um dispositivo simulado é representado na Arquitetura Orientada a Serviços exatamente da mesma forma que um dispositivo real.

Essa homogeneidade na implementação garante outra importante funcionalidade: a possibilidade de integração com o ambiente real. Assim, além da possibilidade de simular dispositivos, o software funciona como um sistema supervisório dos dispositivos reais. Esse espelhamento é realizado por um agente presente no simulador chamado `InterfaceAgent` que registra-se no DF para receber todas as mensagens enviadas

pelos dispositivos do ambiente, sejam eles simulados ou reais (deve-se lembrar que não há distinção na representação entre dispositivos reais ou simulados, o que simplifica a implementação dessa funcionalidade).

Outra característica do simulador desenvolvido é a possibilidade de permitir ao usuário do simulador a completa personalização do ambiente, isto é, o espaço físico, os dispositivos implantados e os usuários (personagens) virtuais utilizados. Esses elementos são descritos através de arquivos XML, que são carregados e interpretados na inicialização do software simulador. A seguir é ilustrado como o simulador foi utilizado para a validação desse trabalho.

6.1.2.1 Definição do ambiente simulado

Os ambiente físicos simulados possuem uma mapa associado cujas características são descritas através de um arquivo XML utilizando a *tag* `<map_data>`. Cada mapa é composto por uma lista de cômodos e possui um nome e um arquivo de imagem associados. Por sua vez, um cômodo é descrito pelo seu nome e por sua localização (coordenadas) na imagem do mapa. Como múltiplos mapas podem ser definidos no mesmo arquivo, a *tag* `<active_map>` é utilizada para indicar qual mapa deve ser carregado e utilizado na simulação. O código apresentado na Figura 43 ilustra o trecho da descrição do mapa referente ao quarto e ao banheiro do laboratório DOMUS, enquanto a Figura 44 mostra o mapa completo desenvolvido a partir da planta baixa publicada em (CHIKHAOUI; WANG; PIGOT, 2010).

6.1.2.2 Definição dos dispositivos no simulador

Cada dispositivo utilizado no simulador, seja ele simulado ou presente fisicamente, é descrito em um arquivo XML através da *tag* `<device_data>` e contém um nome, um tipo (sensor ou atuador), uma lista contendo os possíveis estados e as imagens associadas a esses estados. O código ilustrado na Figura 45 exemplifica a descrição do sensor de pressão enquanto a Figura 46 mostra as imagens que representam os estados `true` e `false` do sensor.

6.1.3 Definição dos personagens

Além dos mapas e dispositivos, é possível personalizar os personagens que representam os usuários da casa inteligente. Esses personagens são relacionados através de um arquivo

```
<?xml version="1.0" encoding="UTF-8" ?>
<map_data>
    <active_map>basic_map</active_map>
    <map name="basic_map">
        <format>png</format>
        <file_path>res/map/ambitecas.png</file_path>
        <room name="bedroom1">
            <x>110</x>
            <y>34</y>
            <width>137</width>
            <height>138</height>
        </room>
        <room name="bathroom1">
            <x>252</x>
            <y>33</y>
            <width>135</width>
            <height>97</height>
        </room>
    </map>
</map_data>
```

Figura 43: Trecho da descrição do mapa referente aos cômodos da residência. Fonte: o autor (2016)

(a) (b)

Figura 44: (a) Mapa utilizado no simulador e desenvolvido a partir da imagem (b), publicada em (CHIKHAOUI; WANG; PIGOT, 2010). Fonte: o autor (2016)

```
<?xml version="1.0" encoding="UTF-8" ?>
<device_data>
    <name>PressureSensor</name>
    <type>sensor</type>
    <width>50</width>
    <height>50</height>
    <state name="true">
        <file_path>res/device/pressureSensor/pressureSensor_true.png</file_path>
    </state>
    <state name="false">
        <file_path>res/device/pressureSensor/pressureSensor_false.png</file_path>
    </state>
</device_data>
```

Figura 45: Descrição do sensor de pressão utilizado no simulador. Fonte: o autor (2016)

(a) (b)

Figura 46: Imagem dos estados do sensor de pressão. Fonte: o autor (2016)

XML utilizando a *tag* `<character_data>`, onde cada personagem é descrito por um identificador único e por uma série de atributos relacionados a sua representação gráfica, tais como tamanho, arquivo de imagem e posição inicial. Esse identificador é utilizado pelos sensores do ambiente para identificar qual é o usuário em questão, permitindo assim a busca das informações necessárias para a entrega dos serviços personalizados.

Assim como na definição dos mapas, é utilizado uma *tag* (`<active_character>`) para definir qual usuário deve ser utilizado no início da simulação. O restante dos usuários descritos no arquivo XML podem ser utilizados a qualquer momento da simulação, bastando selecionar o usuário através da interface gráfica do software. O trecho de código apresentado na Figura 47 exemplifica como um personagem é definido no simulador.

```
<?xml version="1.0" encoding="UTF-8" ?>
<character_data>
    <active_character>John</active_character>
    <character name="John">
        <format>png</format>
        <tilesWidth>60</tilesWidth>
        <tilesHeight>60</tilesHeight>
        <initialPositionX>0</initialPositionX>
        <initialPositionY>0</initialPositionY>
        <circleRadius>10</circleRadius>
        <file_path>res/character/userSpriteSheet.png</file_path>
    </character>
    .
    .
    .

</character_data>
```

Figura 47: Definição de um personagem no simulador. Fonte: o autor (2016)

6.1.4 Sistema de automação residencial

A parte real (física) da Arquitetura Orientada a Serviços utilizada foi implementada utilizando as soluções de automação predial/residencial da empresa Homesystems[2], a qual vem atuando em parceria com o Grupo de Controle, Automação e Robótica da UFRGS em diversos projetos nos últimos anos. O sistema de automação da Homesystems é composto

[2]http://www.homesystems.com.br/

de diversos módulos interligados através da rede proprietária HSNET, que utiliza o padrão RS-485 na sua camada física e é controlada pelo módulo denominado *Systembox*. Além de controlar a rede proprietária, o *Systembox* possibilita a comunicação com todos os dispositivos da rede por meio do protocolo HTTP. Dessa forma, a comunicação entre o agente que representa o dispositivo (pertencente à Arquitetura Orientada a Serviços) e o dispositivo representado pode ser facilmente implementada pelos desenvolvedores utilizando *Sockets* TCP, bastando conhecer a porta de comunicação e o endereço IP do *Systembox*, os endereços associados a cada dispositivo na rede proprietária (chamados de *units*) e os valores que representam os comandos a serem enviados.

A validação do sistema foi realizada na sala *Sunset* na sede da empresa, que pode ser observada na Figura 48. Buscando utilizar a estrutura disponível da melhor maneira possível, os dispositivos presentes na sala foram divididos em dois grupos: o primeiro representando os dispositivos da sala de estar (Figura 49) e o segundo simbolizando os dispositivos do dormitório (Figura 50) da casa DOMUS.

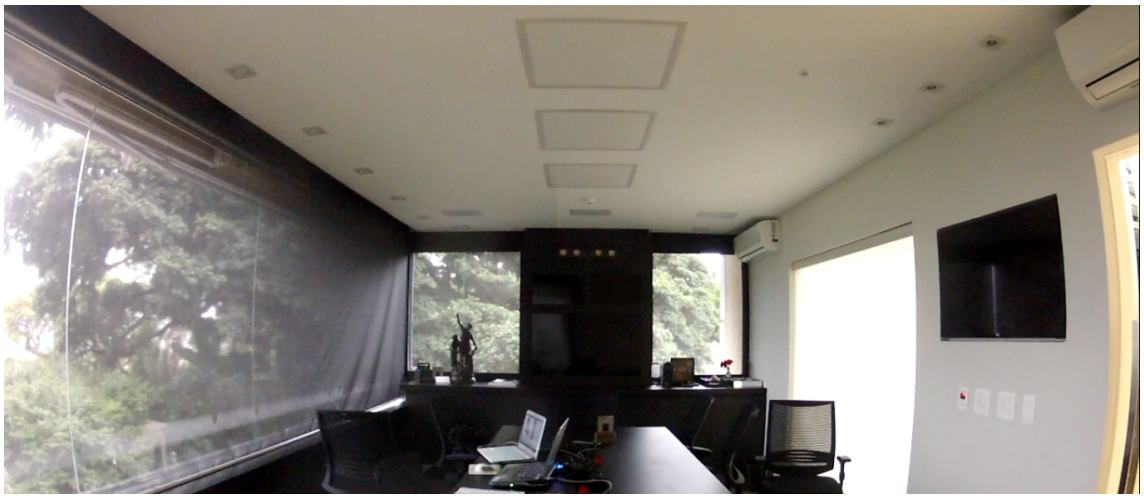


Figura 48: Foto panorâmica da sala *Sunset* na sede da empresa Homesystems. Fonte: o autor (2016)

6.2 Casos de uso

Com o objetivo de demonstrar como o sistema proposto pode ser utilizado para fornecer assistência personalizada ao usuário em uma casa inteligente, foram definidos casos de uso baseados nos "casos de uso representativos" desenvolvidos pelo projeto internacional *Action Aimed at Promoting Standards and Interoperability in the Field of AAL* (EICHELBERG; RöLKER-DENKER; HELMER, 2014), lançado pelo programa *AAL*

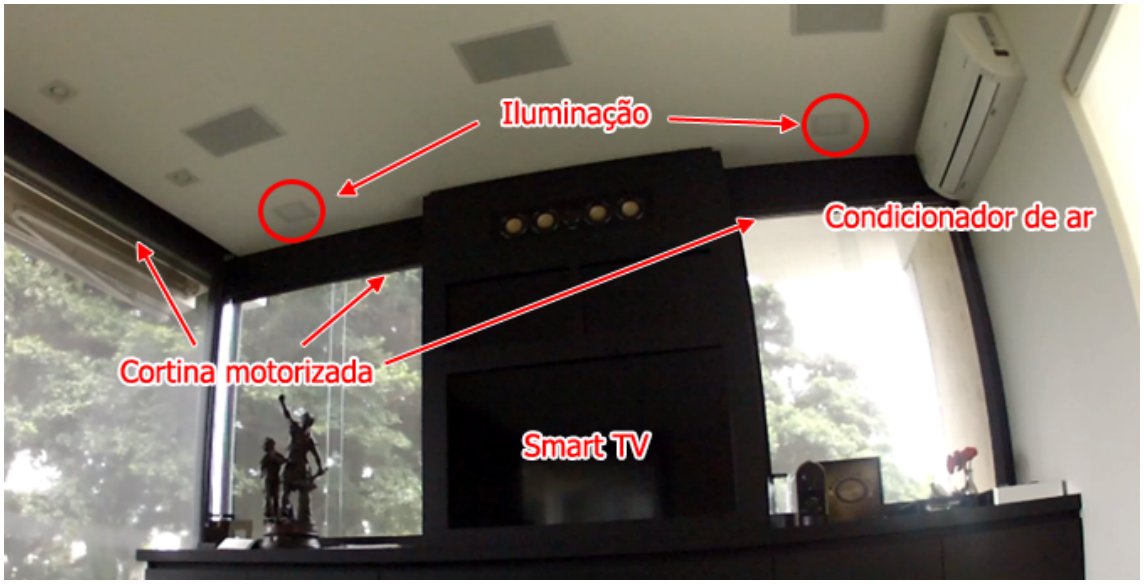


Figura 49: Foto dos dispositivos utilizados na sala de estar. Fonte: o autor (2016)

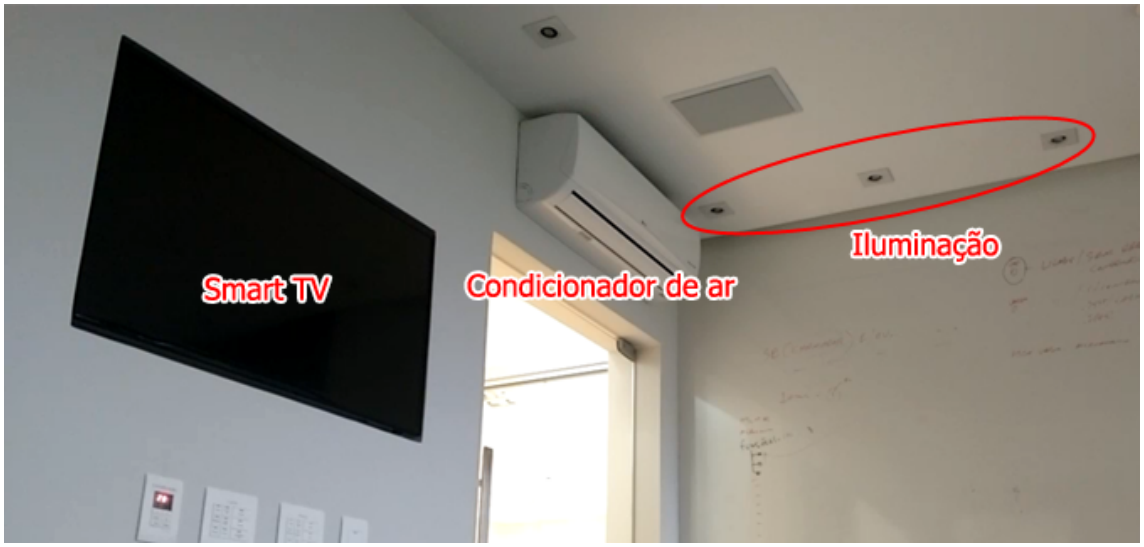


Figura 50: Foto dos dispositivos utilizados no dormitório. Fonte: o autor (2016)

Joint Programme.

O objetivo principal desse projeto é a definição de casos de usos que representam um conjunto de características cobrindo o domínio dos Ambientes Assistidos, em particular os tópicos das seis chamadas publicadas pela AAL até 2013, incluindo (i) Prevenção e Gerenciamento de Condições Crônicas (*Prevention and Management of Chronic Conditions*), (ii) (Auto-) Gerenciamendo das Atividades de Vida Diária no Domícilio (*(Self-)Management of Daily Life Activities at Home*), (iii) Independência e Participação Ativa na Sociedade (*Independence and Participation in the "Self-Serve Society"*), entre outros.

Os resultados desse projeto mostraram que entre os principais propósitos inerentes

aos Ambientes Assistidos está a compensação de perdas relacionadas às funções mentais e o auxílio de atividades relacionadas à comunicação. Assim, o *AAL Joint Programme* identificou as principais funcionalidades que as tecnologias assistivas presentes no ambiente assistido devem oferecer a fim de cumprir com esses propósitos de maneira adequada. Essas funcionalidades foram então compiladas na forma de "casos de uso representativos", cujas descrições técnicas são apresentadas a seguir.

6.2.1 Funcionalidades necessárias

Monitoramento de comportamento: As disfunções cognitivas em um indivíduo normalmente progridem lentamente ao decorrer dos anos. Para preservar o máximo de independência possível ao usuário, enquanto prevenindo acidentes relacionadas às disfunções, é necessário que exista um monitoramento de comportamento que tente identificar as atividades do usuário na residência, fornecer alertas ao usuário em situações perigosas ou notificações aos cuidadores caso seja identificado um aumento da necessidade de ajuda. A localização e as atividades do usuário devem ser monitorados por sensores presentes no ambiente de forma não intrusiva. Dependendo da classificação da situação identificada, o sistema deve notificar o usuário, o cuidador formal/familiar, ou disparar um alarme. Além disso, o sistema pode controlar os atuadores no ambiente (em particular lâmpadas e persianas) para fornecer iluminação para o banheiro quando o usuário acorda a noite, ou então para guiar o usuário até o seu dormitório de maneira não intrusiva, por exemplo.

Serviço de calendário : O serviço de calendário é um sistema focado para usuários que frequentemente esquecem coisas como compromissos, atividades sociais e particularmente, medicamentos. O sistema deve atuar como um calendário inteligente que relembra o usuário sobre compromissos marcados, notifica se alguma AVD básica não foi realizada e avisa os horários em que os medicamentos precisam ser administrados. Ainda, caso o sistema seja acoplado a um dispensador de medicamentos, o sistema pode avisar os cuidadores se o usuário esquecer de tomar os remédios com frequência.

Interação social com Smart TVs : Um dos principais meios utilizados por usuários de ambientes assistidos para a comunicação e interação social são as Smart Tvs. Por meio dessa tecnologia, os usuários podem efetuar compras, navegar na Internet,

participar de jogos online, além de poder realizar vídeo conferências para conversar com familiares e amigos. Além disso, um smartphone pode ser conectado à smart tv de forma a funcionar como um controle remoto.

6.2.2 Definição dos usuários

Após os "casos de uso representativos" formalizados pelo *AAL Joint Programme* terem sido selecionados e descritos, são apresentados os usuários fictícios utilizados na validação desse trabalho, também definidos baseando-se nos resultados encontrados em (EICHELBERG; RöLKER-DENKER; HELMER, 2014).

Jane Miller é uma senhora de 85 anos de idade que vive independentemente em seu próprio apartamento desde que seu marido faleceu, há alguns anos atrás. Sua filha mora a cinquenta quilômetros, perto o bastante para que ela possa visitar uma ou duas vezes por semana, mas não todos os dias. Apesar de sua dificuldade de audição e suas doenças crônicas que necessitam de medicamentes duas vezes ao dia, Jane está relativamente bem. Porém, recentemente ela começou a esquecer algumas coisas e a cometer enganos que até então não aconteciam. O médico da família a diagnosticou com uma leve disfunção cognitiva, isto é, um estágio inicial de demência que pode ou não piorar com o tempo. Há alguns meses atrás ela ligou o fogão e saiu de casa para fazer compras, causando um incêndio na cozinha que poderia ter se alastrado por toda a casa. A Figura 51 apresenta o perfil de usuário ontológico de Jane Miller, contendo todas as informações fornecidas durante o seu cadastro no sistema, além das preferências inferidas pelo Gerenciador de Perfis.

Peter é um senhor de 83 anos vivendo no subúrbio de uma grande cidade brasileira. Sua esposa faleceu quatro anos atrás e seu filho mudou-se para uma cidade a 200 km de distância. No passado Peter não costumava utilizar computadores e dispositivos móveis, mas desde que a geração de dispositivos com *touch screen* tornou-se disponível, ele está mais entusiasmado em relação a utilização dessa tecnologia. Peter ama usar a vídeo conferência durante a noite para conversar com seu filho Michael e sua esposa Júlia, assim como seus três netos. Uma das maiores dificuldades no dia a dia de Peter é a dificuldade em enxergar com clareza os textos e imagens apresentados nos dispositivos de imagem. Além disso, Peter apresenta uma leve dificuldade de audição.

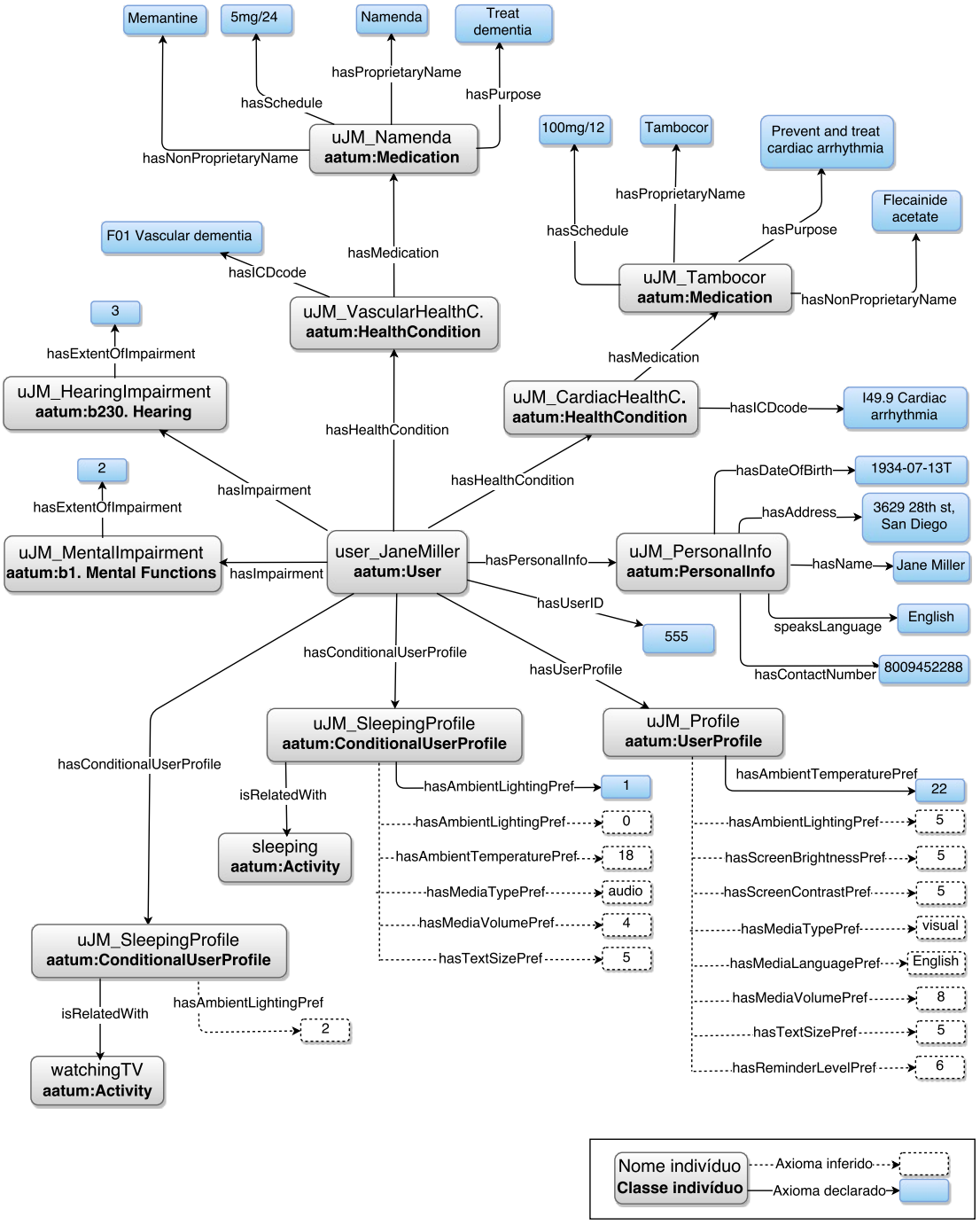


Figura 51: Perfil ontológico da usuária Jane Miller. Fonte: o autor (2016)

Finalmente, foram definidos três casos de uso que demonstram como o sistema proposto é utilizado a fim de atender às funcionalidades identificadas anteriormente. Para isso foram desenvolvidos três aplicativos de assistência: o primeiro responsável por monitorar as atividades do usuário e intervir caso alguma situação de perigo seja detectada; o segundo por controlar a luminosidade e a temperatura do ambiente de modo a garantir conforto e

segurança ao usuário, além de uma possível redução no consumo elétrico; o terceiro por consultar o serviço de calendário e apresentar possíveis lembretes ao usuário.

6.2.3 Caso de uso 1 - Controle do conforto e monitoramento de atividades

Utilizando o simulador de ambientes inteligentes desenvolvido, a personagem simbolizando Jane Miller é movida até a cozinha (Figura 52 (a)). Para simbolizar o acendimento do fogão, o sensor de tensão on/of sobre o fogão é selecionado e tem seu estado alterado. A personagem é então movida em direção seu dormitório. Clicando sobre o sensor de contato na porta do quarto, seu estado é alterado para "false", simbolizando a abertura da porta. Após alguns instantes da entrada de Jane, a luminosidade no cômodo é ajustada para o nível 5, de acordo com a preferência inferida automaticamente pelo Gerenciador de Perfis. Ainda, a temperatura é ajustada para 22 °C, valor este informado por Jane durante seu cadastro no sistema (Figura 52 (b)).

(a) (b)

Figura 52: Visualização do simulador e do ambiente real durante o caso de uso 1 - acendimento do fogão. Fonte: o autor (2016)

A partir desse momento, dois outros possíveis cenários são demonstrados. No primeiro, Jane fecha a porta do dormitório e deita em sua cama para dormir. Quando isso ocorre, a iluminação do quarto é alterada para o nível 1, valor aprendido pelo sistema após Jane requisitar esse valor durante a mesma situação há alguns dias, e o condicionador de ar

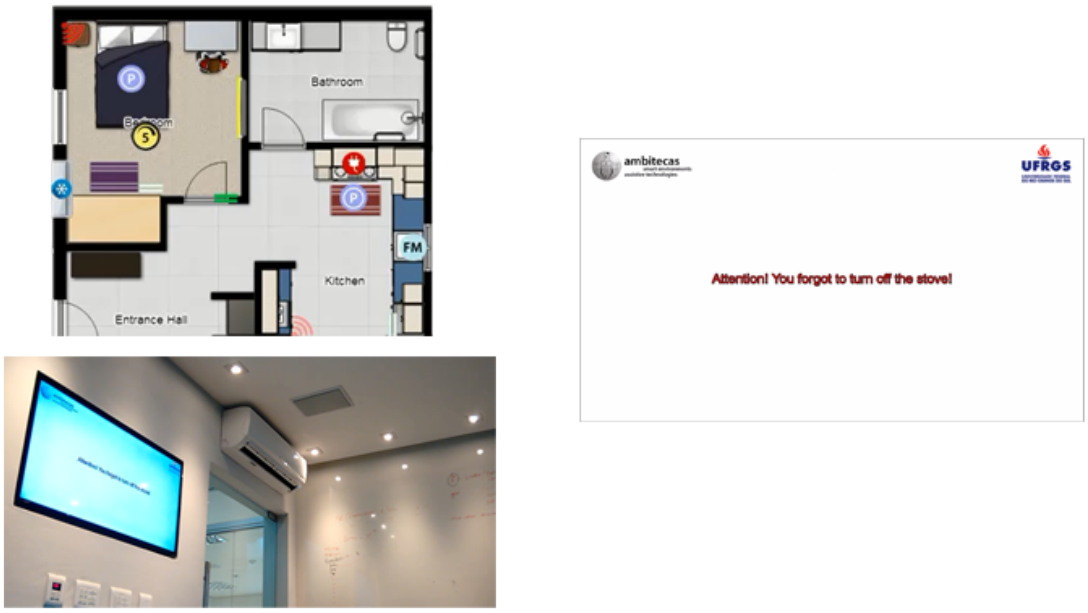
é configurado para a temperatura de 18 °C, valor inferido pelo Gerenciador de Perfis (Figura 53 (a)). Após transcorrido um certo tempo, o aplicativo de monitoramento de comportamento identifica a situação de perigo e atua de forma a garantir a segurança de Jane, ligando as luzes no nível máximo e disparando o aviso sonoro *"Attention! You forgot to turn off the stove!"*, de acordo com o idioma falado por Jane e com o nível de volume 8, inferido de acordo com sua capacidade de audição. (Figura 53 (b)).

(a) (b)

Figura 53: Visualização do simulador e do ambiente real durante o caso de uso 1 - dormindo.
Fonte: o autor (2016)

No segundo cenário, Jane fecha a porta e senta-se na cadeira do computador para ler um livro. Assim como no caso anterior, após transcorrido um certo tempo após o acendimento do fogão, um alerta é emitido. Porém, como Jane não está dormindo, o alerta é emitido de acordo com a preferência inferida pelo Gerenciador de Perfis - nesse caso por meio visual, já que Jane possui um alto nível de dificuldade na audição (Figura 54 (a)). Essa mensagem é apresentada no idioma de Jane e com o tamanho de fonte inferido a partir das informações relacionadas a capacidade de visão de Jane. A Figura 54 (b) mostra a captura da tela apresentada a Jane.

Após o alerta ser emitido, o aplicativo aguarda um certo tempo até que o fogão seja desligado. Caso isso não ocorra, o próprio sistema requisita o serviço para desligar o fogão. Como no cenário utilizado não existe nenhum dispositivo capaz de realizar o serviço, o



(a) (b)

Figura 54: Visualização do simulador e do ambiente real durante o caso de uso 1 - outra atividade. Fonte: o autor (2016)

aplicativo decide por alertar a filha de Jane utilizando o contato informado em seu cadastro no sistema. Isso é realizado através de outra requisição de serviço, dessa vez atendida e executada pela Smart TV presente no dormitório.

Quando o mesmo cenário foi reproduzido com o usuário Peter, o alerta informando que o fogão foi esquecido ligado foi apresentado através de um áudio em português, uma vez que esse é o idioma falado por Peter. Assim como anteriormente, o áudio é reproduzido com o volume inferido de acordo com as capacidades auditivas do usuário[3].

6.2.4 Caso de uso 2 - Controle do conforto e monitoramento de atividades

Nesse caso de uso uma situação semelhante a anterior é reproduzida: Jane vai à cozinha, liga o fogão e vai para a sala (Figura 55 (a)). Após alguns instantes após a sua identificação pelo leitor RFID da sala, a luminosidade do cômodo é ajustada para o nível 5, de acordo com a preferência inferida automaticamente pelo Gerenciador de Perfis. Esse valor de luminosidade é alcançado abrindo as cortinas elétricas e mantendo a iluminação elétrica desligada, conforme determinado pelo algoritmo presente na Arquitetura Orientada a Serviços utilizada neste trabalho. Ainda, a temperatura é ajustada para 22 °C, valor este

[3]A demonstração completa pode ser assistida em `https://www.youtube.com/channel/UCD7sGaks-vo2176YBoaVGbg`

(a) (b)

Figura 55: Visualização do simulador e do ambiente real durante o caso de uso 2 - acendimento do fogão. Fonte: o autor (2016)

informado por Jane durante seu cadastro no sistema. Ainda na sala, Jane decide assistir a um filme na Smart TV. Para isso, ela utiliza o aplicativo instalado em seu celular (Figura 56)[4] que apresenta de forma dinâmica os serviços disponíveis no ambiente (Figura 55 (b)). A mídia é então reproduzida utilizando o idioma, volume, contraste e brilho de acordo com as preferências fornecidas pelo Gerenciador de Perfis.

Após solicitar um dos filmes disponíveis na Smart TV, Jane senta-se na poltrona. Nesse momento o Gerenciador de Contexto identifica que Jane está assistindo TV, o que leva o aplicativo responsável pela iluminação a utilizar o valor da luminosidade inferida pelo Gerenciador de Perfis para essa situação, nesse caso igual a 2. Após o aplicativo de assistência responsável pelo controle de luminosidade requisitar o serviço de iluminação com esse valor à Arquitetura Orientada a Serviços, as cortinas elétricas são fechadas e a lâmpada é acionada com a potência adequada (Figura 57 (a)). Após transcorrido um certo tempo, o aplicativo de monitoramento de comportamento identifica a situação de perigo e emite um aviso na forma de texto na mesma Smart TV em que Jane está assistindo. A opção em utilizar texto é inferida pelo Gerenciador de Perfis devido ao alto nível de

[4] Aplicativo desenvolvido pelo aluno do PPGEE Charles Steinmetz a partir da estratégia descrita em 5.4.3

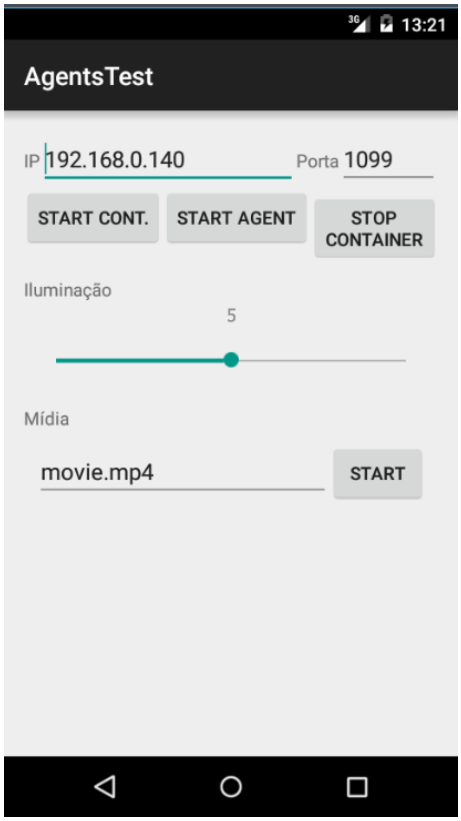
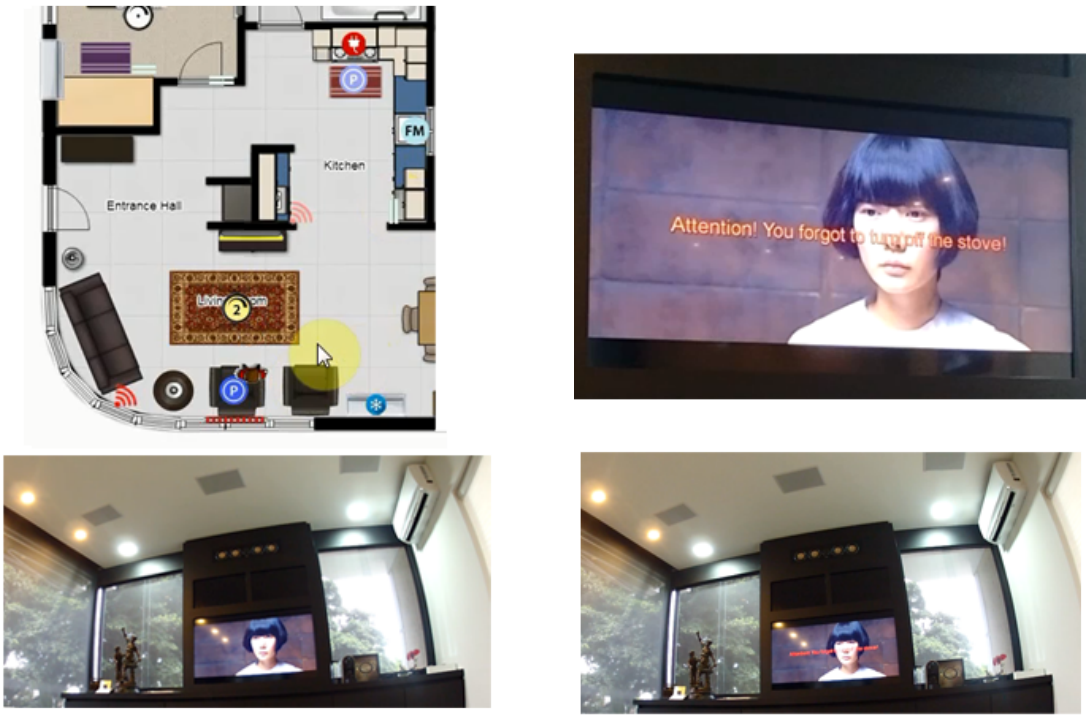


Figura 56: Aplicativo para requisição de serviços no Ambiente Inteligente. Fonte: o autor (2016)

dificuldade na audição. Da mesma maneira, o tamanho do texto e o idioma utilizado são inferidos pelo Gerenciador de Perfis de acordo com as informações pessoais de Jane (Figura 57 (b)).

Assim como no caso de uso anterior, o cenário é reproduzido com o usuário Peter. Dessa vez o condicionador de ar é ajustado para a temperatura preferida por Peter (24 °C) e a luminosidade é alterada para o nível 8, conforme inferido pelo Gerenciador de Perfis. Ao requisitar o filme através do aplicativo em seu smartphone, a Smart TV tem seu brilho e contraste ajustados conforme os valores inferidos e as legendas são apresentadas em português e com tamanho adequado às deficiências visuais de Peter (Figura 58).



(a) (b)

Figura 57: Visualização do simulador e do ambiente real durante o caso de uso 2 - notificação emitida enquanto o usuário assiste a TV. Fonte: o autor (2016)

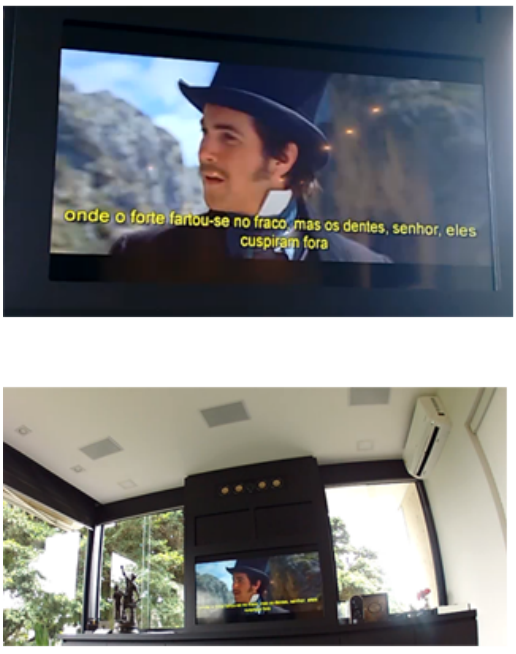


Figura 58: Visualização do ambiente real durante o caso de uso 2 - ajuste da Smart Tv para o usuário Peter. Fonte: o autor (2016)

6.2.5 Caso de uso 3 - Notificação dos medicamentos

No ultimo caso de uso reproduzido, é utilizado um Aplicativo de Assistência responsável por interagir com o o serviço de calendário disponível no ambiente contendo o cronograma dos medicamentos que devem ser administrados pelos usuários. Quando o aplicativo identifica que um medicamento deve ser administrado, uma notificação personalizada é emitida de acordo com o usuário e seu contexto.

A Figura 59 mostra a notificação emitida para Jane Miller quando ela encontra-se em seu dormitório. Novamente, a notificação ocorre através da exibição da mensagem *"It's time to take your medicine: Namenda 5mg"* na SmartTV que obedece o idioma e o tamanho indicados pelo Gerenciador de Perfis, além de possuir um fundo animado para chamar a atenção de Jane.

Quando a situação foi repetida para o usuário Peter, sentado na sala de estar, a notificação aconteceu de maneira totalmente diferente: a SmartTV reproduziu o áudio em português *"Atenção, está na hora de tomar o seu remédio"* com o nível de volume indicado pelo Gerenciador de Perfis.

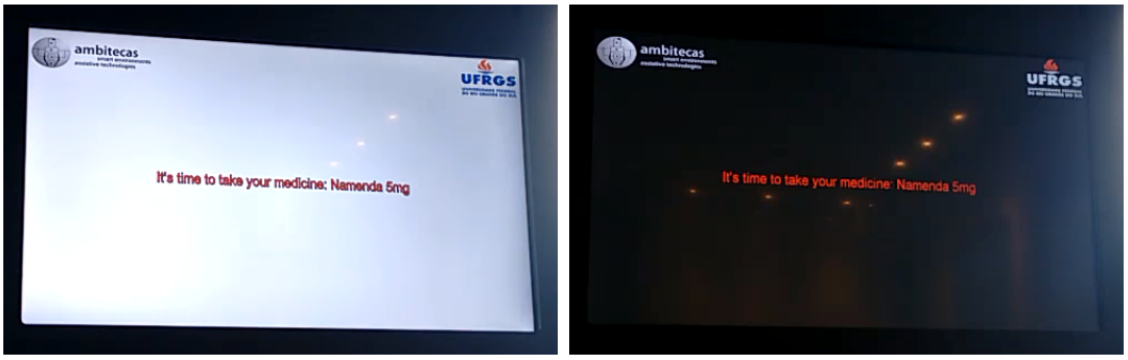


Figura 59: Mensagem personalizada apresentada a Jane Miller alertando-a sobre seu remédio. Fonte: o autor (2016)

7 CONCLUSÕES

Este trabalho apresentou a proposta de um sistema de assistência personalizada para ambientes inteligentes onde o objetivo principal é aumentar a qualidade de vida dos usuários idosos e com diversidades funcionais através da personalização dos serviços encontrados no ambiente. Para que o fornecimento da assistência personalizada seja possível, foi construído um modelo de usuário capaz de representar diversos aspectos relacionados ao usuário e seu contexto de maneira padronizada e consistente. Juntamente a esse modelo, um mecanismo de inferência lógica foi utilizado para determinar como os serviços do ambiente devem ser personalizados de acordo com as informações representadas utilizando o modelo de usuário proposto, como as dificuldades, a localização e a atividade do usuário. Por fim, foi proposta e implementada uma arquitetura distribuída que permite a construção de Aplicativos de Assistência de forma escalável e flexível utilizando as informações fornecidas pelo modelo de usuário e o mecanismo de inferência lógica para requisitar os serviços presentes na Arquitetura Orientada a Serviços de maneira correta, e consequentemente auxiliar o usuário de maneira ideal.

A abordagem utilizada nesse trabalho segue a lógica utilizada nos trabalhos relacionados, porém apresenta algumas características inovadoras que trazem algumas vantagens e buscam resolver problemas identificados nos trabalhos anteriores: (i) a primeira característica inovadora é a incorporação de um modelo padronizado e internacionalmente conhecido para representar as deficiências, dificuldades e limitações dos usuários, o que vai ao encontro de um dos grandes benefícios no uso das tecnologias da Web Semântica: interoperabilidade e reuso do conhecimento em diversos domínios de aplicação; (ii) a segunda característica é a utilização do paradigma multi agentes na implementação de uma arquitetura distribuída, flexível, dinâmica e escalável. **Distribuída** pelo fato de permitir que os componentes sejam executadas em lugares diferentes, inclusive em dispositivos

móveis; **flexível** por permitir que sistemas externos interajam com o sistema proposto de maneira a complementar suas funcionalidades, como por exemplo no fornecimento dos dados de usuário ou na utilização de diferentes técnicas de aprendizagem e de detecção de atividades; **dinâmica** por permitir que os dispositivos e/ou serviços disponíveis no ambiente sejam alterados em tempo de execução; **escalável** pois permite que novas funcionalidades sejam desenvolvidas e inseridas no sistema de acordo com o necessário.

Devido às estratégias utilizadas neste trabalho, diversas oportunidades para trabalhos futuros podem ser observadas:

- Desenvolvimento de outras técnicas para a aprendizagem e adaptação das preferências de acordo com o histórico de requisições de serviço e padrões de comportamento observados ao longo da execução do sistema. Possíveis abordagens incluem a utilização de lógica fuzzy e redes neurais artificiais.

- Desenvolvimento de ferramentas que permitam a criação e modificação das regras de inferência de maneira simples e intuitiva. Como a linguagem SWRL representa relações de causa-efeito de maneira intuitiva, novas ferramentas poderiam converter informações representadas graficamente pelo usuário ou seu cuidador para sua implementação na linguagem de inferência.

- Desenvolvimento de ferramentas que possibilitem o desenvolvimento de novos aplicativos de assistência de maneira intuitiva, sem a necessidade de conhecimento de linguagens de programação. Assim como no item anterior, essas ferramentas poderiam conter uma interface gráfica que permitam ao usuário a seleção de funcionalidades e a escolha de ações a serem realizadas em situações específicas.

- Implementação e avaliação do sistema em uma casa inteligente real. Devido às limitações do tempo de desenvolvimento deste trabalho, não foi possível realizar sua validação com usuários reais, portadores de necessidades especiais e que necessitam de assistência na realização de suas AVD. Uma possibilidade para realizar essa validação seria a instalação dos equipamentos de automação residencial da empresa parceira Homesystems em uma casa de cuidados a idosos, juntamente com o sistema proposto. Através do acompanhamento por uma equipe multidisciplinar, contendo profissionais da área de saúde e de tecnologia, seria possível a identificação e

correção de falhas no sistema proposto que não são possíveis de serem observadas de outra maneira.

REFERÊNCIAS

ABBURU, S. A survey on ontology reasoners and comparison. **International Journal of Computer Applications**, [S.l.], v.57, n.17, p.33–39, 2012.

ANTONIOU, G.; HARMELEN, F. van. Web ontology language: owl. In: STAAB, S.; STUDER, R. (Ed.). **Handbook on ontologies**. New York: Springer, 2009. p.91–110.

BELLIFEMINE, F. L.; CAIRE, G.; GREENWOOD, D. **Developing multi-agent systems with JADE**. [S.l.]: John Wiley & Sons, 2007. v.7.

BELLIFEMINE, F.; POGGI, A.; RIMASSA, G. JADE–A FIPA-compliant agent framework. In: INTERNATIONAL CONFERENCE ON THE PRACTICAL APPLICATION OF INTELLIGENT AGENTS AND MULTI-AGENT TECHNOLOGY, 4., 1999, London, UK. **Proceedings...** Blackpool: The Practical Application Company Ltd, 1999. p.97–108.

BERNERS-LEE, T. **Semantic Web - XML2000**. Disponível em: <http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html>. Acesso em: 26 dez. 2015.

BERNERS-LEE, T. et al. The semantic web. **Scientific American**, [S.l.], v.284, n.5, p.28–37, 2001.

BOUCHARD, K. et al. SIMACT: a 3d open source smart home simulator for activity recognition with open database and visual editor. **International Journal of Hybrid Information Technology**, Hobart, v.5, n.3, p.13–32, July 2012.

CAIRE, G. **JADE tutorial**: jade programming for beginners. Disponível em: <http://jade.tilab.com/doc/tutorials/JADEProgramming-Tutorial-for-beginners.pdf>. Acesso em: 02 jan. 2016.

CASAS, R. et al. **User modelling in ambient intelligence for elderly and disabled people**. [S.l.]: Springer, 2008.

Centers for Disease Control and Prevention (CDC). **The state of aging and health in America 2013**. Disponível em: <http://www.cdc.gov/features/agingandhealth/state_of_aging_and_health_in_america_2013.pdf>. Acesso em: 10 jul. 2015.

Centre de recherche sur le vieillissement. **DOMUS Laboratory**: domotics and mobile computing at the université de sherbrooke. Disponível em: <http://cdrv.csss-iugs.ca/details-des-laboratoires/23-laboratoire-domus-domotique-et-informatique-mobile-a-luniversite-de-sherbrooke >. Acesso em: 02 set. 2015.

CHIKHAOUI, B.; WANG, S.; PIGOT, H. A new algorithm based on sequential pattern mining for person identification in ubiquitous environments. In: INTERNATIONAL WORKSHOP ON KNOWLEDGE DISCOVERY FROM SENSOR DATA (SENSORKDD'10), 4., 2010, Washington DC, USA. **Proceedings...** Oak Ridge: Oak Ridge National Laboratory, 2010. p.19–28.

DASIOS, A. et al. Hands-On Experiences in Deploying Cost-Effective Ambient-Assisted Living Systems. **Sensors**, [S.l.], v.15, n.6, p.14487–14512, 2015.

EICHELBERG, M.; RöLKER-DENKER, L.; HELMER, A. **Action Aimed at Promoting Standards and Interoperability in the Field of AAL Deliverable D2 AAL Use Cases and Integration Profiles**. [S.l.: s.n.], 2014.

EWERT, T. et al. Identification of the most common patient problems in patients with chronic conditions using the ICF checklist. **Journal of Rehabilitation Medicine**, [S.l.], v.36, n.0, p.22–29, 2004.

FENSEL, D. et al. OIL: an ontology infrastructure for the semantic web. **IEEE intelligent systems**, [S.l.], v.16, n.2, p.38–45, 2001.

FISCHER, G. User modeling in human–computer interaction. **User modeling and user-adapted interaction**, [S.l.], v.11, n.1-2, p.65–86, 2001.

FREDRICH, C.; KUIJS, H.; REICH, C. An ontology for user profile modeling in the field of ambient assisted living. In: INTERNATIONAL CONFERENCES ON ADVANCED SERVICE COMPUTING, 6., 2014, Venice, Italy. **Proceedings...** Wilmington: IARIA XPS Press, 2014. p.24–31.

GARIN, O. et al. Research Validation of the"World Health Organization Disability Assessment Schedule, WHODAS-2"in patients with chronic diseases. **Health and quality of life outcomes**, [S.l.], v.8, p.51, 2010.

GIARETTA, P.; GUARINO, N. Ontologies and knowledge bases towards a terminological clarification. **Towards very large knowledge bases:** knowledge building & knowledge sharing, [S.l.], v.25, p.25–32, 1995.

GOLEMATI, M. et al. Creating an ontology for the user profile: method and applications. In: INTERNATIONAL CONFERENCE ON RESEARCH CHALLENGES IN INFORMATION SCIENCE (RCIS), 1., 2007, Ouarzazate, Morocco. **Proceedings...** New York: IEEE, 2007. p.407–412.

GRUBER, T. R. A translation approach to portable ontology specifications. **Knowledge Acquisition**, [S.l.], v.5, n.2, p.199–220, 1993.

HEFLIN, J.; HENDLER, J.; LUKE, S. **SHOE**: a knowledge representation language for internet applications. College Park: University of Maryland, 1999.

HENDLER, J.; MCGUINNESS, D. L. The DARPA agent markup language. **IEEE Intelligent Systems**, [S.l.], v.15, n.6, p.67–73, 2000.

HOLSAPPLE, C. W.; JOSHI, K. D. A collaborative approach to ontology design. **Communications of the ACM**, [S.l.], v.45, n.2, p.42–47, 2002.

HONG, J. et al. Context-aware system for proactive personalized service based on context history. **Expert Systems with Applications**, [S.l.], v.36, n.4, p.7448–7457, 2009.

HORRIDGE, M.; BECHHOFER, S. The OWL API: a java api for owl ontologies. **Semantic Web**, [S.l.], v.2, n.1, p.11–21, 2011.

HORROCKS, I. et al. DAML+OIL: a description logic for the semantic web. **IEEE Data Engineering Bulletin**, [S.l.], v.25, n.1, p.4–9, 2002.

HORROCKS, I. et al. SWRL: a semantic web rule language combining owl and ruleml. **W3C Member Submission**, [S.l.], v.21, p.1–79, 2004.

KADOUCHE, R. et al. Personalization in smart homes for disabled people. In: INTERNATIONAL CONFERENCE ON FUTURE GENERATION COMMUNICATION AND NETWORKING (FGCN'08), 2., 2008, Sanya, China. **Proceedings...** New York: IEEE, 2008. p.411–415.

KADOUCHE, R. et al. Support vector machines for inhabitant identification in smart houses. In: **Ubiquitous Intelligence and Computing**. [S.l.]: Springer, 2010. p.83–95.

LACY, L. W. **OWL**: representing information using the web ontology language. [S.l.]: Trafford Publishing, 2005.

LASSILA, O.; SWICK, R. R. **Resource description framework (RDF) model and syntax specification. W3C Recommendation, 1999**. 1999.

LEI, Z. et al. SHSim: an osgi-based smart home simulator. In: IEEE INTERNATIONAL CONFERENCE ON THE UBI-MEDIA COMPUTING (U-MEDIA), 3., 2010, Jinhua, China. **Proceedings...** New York: IEEE, 2010. p.87–90.

LI, R.; LU, B.; MCDONALD-MAIER, K. D. Cognitive assisted living ambient system: a survey. **Digital Communications and Networks**, [S.l.], v.1, n.4, p.229–252, 2015.

LINDA, A.; KENT, M.; MATHER, M. America's aging population. **Population Bulletin**, [S.l.], v.66, n.1, 2011.

LUTZ, W.; SANDERSON, W.; SCHERBOV, S. The coming acceleration of global population ageing. **Nature**, [S.l.], v.451, n.7179, p.716–719, 2008.

MARINC, A. et al. Interactive personalization of ambient assisted living environments. In: INTERNATIONAL CONFERENCE ON HUMAN-COMPUTER INTERACTION, 14., 2011, Orlando, USA. **Proceedings...** New York: Springer, 2011. p.567–576.

MCGUINNESS, D. L.; VAN HARMELEN, F. et al. **OWL web ontology language overview**. Disponível em: <https://www.w3.org/TR/owl-features/>. Acesso em: 13 jan. 2016.

NOY, N. F.; MCGUINNESS, D. L. et al. **Ontology development 101**: a guide to creating your first ontology. [S.l.]: Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880, 2001.

O'CONNOR, M. et al. Supporting rule system interoperability on the semantic web with SWRL. In: INTERNATIONAL SEMANTIC WEB CONFERENCE, 4., 2005, Galway, Ireland. **Proceedings...** New York: Springer, 2005. p.974–986.

PARK, J. et al. CASS: a context-aware simulation system for smart home. In: ACIS INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING RESEARCH, MANAGEMENT & APPLICATIONS, 5., 2007, Busan, Korea. **Proceedings...** New York: IEEE, 2007. p.461–467.

PEDROSO, D. G. **Arquitetura Orientada a Serviços utilizando SOA e MAS**. 2016. 100 f. Dissertação (Mestrado em Engenharia Elétrica) — Universidade Federal do Rio Grande do Sul, Porto Alegre.

Prefeitura da Cidade de São Paulo. **Manual de Instruções Técnicas de Acessibilidade para Apoio ao Projeto Arquitetônico**. Disponível em: <http://www.prefeitura.sp.gov.br/cidade/secretarias/upload/pessoa_com_deficiencia/manual%20acessibilidade.pdf>. Acesso em: 25 jun. 2015.

PRENTZAS, J.; HATZILYGEROUDIS, I. Categorizing approaches combining rule-based and case-based reasoning. **Expert Systems**, [S.l.], v.24, n.2, p.97–122, 2007.

RICQUEBOURG, V. et al. Context inferring in the Smart Home: an swrl approach. In: INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION NETWORKING AND APPLICATIONS WORKSHOPS, 21., 2007, Niagara Falls, Canada. **Proceedings...** New York: IEEE, 2007. v.2, p.290–295.

SHADBOLT, N.; HALL, W.; BERNERS-LEE, T. The semantic web revisited. **Intelligent Systems, IEEE**, [S.l.], v.21, n.3, p.96–101, 2006.

SIRIN, E. et al. Pellet: a practical owl-dl reasoner. **Web Semantics: science, services and agents on the World Wide Web**, [S.l.], v.5, n.2, p.51–53, 2007.

SKILLEN, K.-L. et al. Ontological user profile modeling for context-aware application personalization. In: INTERNATIONAL CONFERENCE ON UBIQUITOUS COMPUTING AND AMBIENT INTELLIGENCE, 6., 2012, Vitoria-Gasteiz, Spain. **Proceedings...** New York: Springer, 2012. p.261–268.

SKILLEN, K.-L. et al. Ontological user modelling and semantic rule-based reasoning for personalisation of help-on-demand services in pervasive environments. **Future Generation Computer Systems**, [S.l.], v.34, p.97–109, 2014.

STAN, J. et al. A user profile ontology for situation-aware social networking. In: WORKSHOP ON ARTIFICIAL INTELLIGENCE TECHNIQUES FOR AMBIENT INTELLIGENCE (AITAMI2008), 3., 2008, Patras, Greece. **Proceedings...** New York: Springer, 2008. não paginado.

STANFORD. **Protégé Stanford Center for Biomedical Informatics Research (BMIR) at the Stanford University School of Medicine**. Disponível em: <http://protege.stanford.edu>. Acesso em: 11 ago. 2015.

STUDER, R.; BENJAMINS, V. R.; FENSEL, D. Knowledge engineering: principles and methods. **Data & knowledge engineering**, [S.l.], v.25, n.1, p.161–197, 1998.

SUTTERER, M.; DROEGEHORN, O.; DAVID, K. UPOS: user profile ontology with situation-dependent preferences support. In: INTERNATIONAL CONFERENCE ON ADVANCES IN COMPUTER-HUMAN INTERACTION, 1., 2008, Venice, Italy. **Proceedings...** New York: IEEE, 2008. p.230–235.

TAZARI, M.-R. et al. The universAAL Reference Model for AAL. **Handbook of Ambient Assisted Living**, [S.l.], v.11, p.610–625, 2012.

TIBERGHIEN, T. et al. Semantic reasoning in context-aware assistive environments to support ageing with dementia. In: **The Semantic Web–ISWC 2012**. [S.l.]: Springer, 2012. p.212–227.

ÜSTÜN, T. B. **Measuring health and disability**: manual for who disability assessment schedule whodas 2.0. [S.l.]: World Health Organization, 2010.

VALHAM, F. et al. Ambient temperature and obstructive sleep apnea: effects on sleep, sleep apnea, and morning alertness. **Sleep**, [S.l.], v.35, n.4, p.513–517, 2012.

VAN NGUYEN, T.; NGUYEN, H.; CHOI, D. Development of a context aware virtual smart home simulator. **arXiv preprint arXiv:1007.1274**, [S.l.], 2010.

VARGAS, M. F. de; PEREIRA, C. E. Ontological User Modeling for Ambient Assisted Living Service Personalization. In: INTERNATIONAL EMBEDDED SYSTEMS SYMPOSIUM (IESS), 5., 2015, Foz do Iguaçu, Brazil. **Proceedings...** New York: Springer, 2015. To appear.

VEGA-BARBAS, M. et al. Smart spaces and smart objects interoperability architecture (S3OiA). In: INTERNATIONAL CONFERENCE ON INNOVATIVE MOBILE AND INTERNET SERVICES IN UBIQUITOUS COMPUTING (IMIS), 6., 2012, Palermo, Italy. **Proceedings...** New York: IEEE, 2012. p.725–730.

VERGADOS, D. D. Service personalization for assistive living in a mobile ambient healthcare-networked environment. **Personal and Ubiquitous Computing**, [S.l.], v.14, n.6, p.575–590, 2010.

WANG, X. H. et al. Ontology based context modeling and reasoning using OWL. In: IEEE ANNUAL CONFERENCE ON PERVASIVE COMPUTING AND COMMUNICATIONS WORKSHOPS, 2., 2004, Orlando, Florida. **Proceedings...** New York: IEEE, 2004. p.18–22.

WEISER, M. The computer for the 21st century. **Scientific American**, [S.l.], v.265, n.3, p.94–104, 1991.

WOLF, P. et al. openAAL-the open source middleware for ambient-assisted living (AAL). In: AALIANCE CONFERENCE, 2010, Malaga, Spain. **Proceedings...** [S.l.: s.n.], 2010. p.1–5.

World Health Organization. **ICF checklist**. Disponível em: <http://www.who.int/classifications/icf/icfchecklist.pdf>. Acesso em: 4 abr. 2015.

WORLD HEALTH ORGANIZATION. Global Health and Aging. **NIH Publication no 117737**, [S.l.], v.1, n.4, p.273–277, 2011.