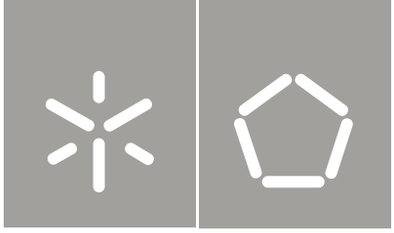
Universidade do Minho
Escola de Engenharia

Diogo Aires Gonçalves Ribeiro

Modelação em contextos de Ambient-Assisted Living

Modelação em contextos de Ambient-Assisted Living
Diogo Aires Gonçalves Ribeiro
UMinho | 2013

dezembro de 2013



Universidade do Minho
Escola de Engenharia

Diogo Aires Gonçalves Ribeiro

Modelação em contextos de Ambient-Assisted Living

Tese de Mestrado
Ciclo de Estudos Integrados Conducentes ao Grau de
Mestre em Engenharia Eletrónica Industrial e de Computadores

Trabalho efetuado sob a orientação do
Doutor Paulo Francisco Silva Cardoso

dezembro de 2013

DECLARAÇÃO

Diogo Aires Gonçalves Ribeiro
Endereço eletrónico: a54018@alunos.uminho.pt Telefone:
Número do Bilhete de Identidade:
Título da Tese:

Modelação em contextos de *Ambient-Assisted Living*

Orientador:

Doutor Paulo Francisco Silva Cardoso

Ano de conclusão: 2013

Tese submetida na Universidade do Minho para a obtenção do grau de

Mestre em Engenharia Eletrónica Industrial e de Computadores

Universidade do Minho, 12/12/2013

Assinatura: _____

A todos que contribuíram para este trabalho...

Agradecimentos

A realização do trabalho aqui apresentado não teria sido possível sem o apoio e a colaboração de algumas pessoas, às quais transmito os meus mais sinceros agradecimentos:

Ao meu orientador pela disponibilidade e dedicação sempre demonstradas. A sua colaboração foi fundamental nos momentos de maior desânimo e desespero. Não teria sido possível completar este trabalho sem a sua colaboração. Agradeço por todos os momentos em que me ajudou a não desistir. Agracio ainda por todo o conhecimento que me transmitiu ao longo do curso. Obrigado por não ser apenas "mais um professor".

Ao professor Adriano Tavares por todo o conhecimento transmitido através da sua forma muito particular. O meu agradecimento prolonga-se não só por este ano mas pelos últimos dois anos. Foi uma experiência extremamente enriquecedora que não vou jamais esquecer. Devo-lhe grande parte da minha capacidade de resolução de problemas e resiliência em momentos de elevada pressão.

À minha namorada, Ana Margarida Trigo, pelo apoio incondicional durante todo o meu percurso académico. A ela um muito obrigado pois não seria capaz de estar aqui hoje sem o seu suporte.

À minha família pelo suporte e paciência nos momentos de maior ausência, irritação e *stress*.

Aos meus colegas e amigos pelos momentos de diversão, de estudo e paciência nos momentos mais difíceis do curso.

Aos funcionários Carlos Torres, Joel Almeida e Ângela Macedo que, na qualidade de técnicos das oficinas do Departamento de Eletrónica Industrial, foram sempre incansáveis em todos os anos do mestrado.

Resumo

Neste relatório de dissertação está descrito o desenvolvimento de uma ontologia para a aplicação em sistemas de assistência à vida humana e todas as etapas adjacentes a este projeto. Está ainda presente uma vista geral da construção deste tipo de soluções, desde a arquitetura do *hardware* às técnicas de *software*. As soluções de *Ambient-Assited Living* (AAL*)*, surgem na necessidade da sociedade em manter pessoas idosas ou com necessidades especiais mais tempo em suas casas, não descurando a sua saúde, mantendo o seu nível de vida. No entanto, devido à sensibilidade do tema ainda não existe nenhuma solução totalmente testada e de fácil implementação nas habitações dos utilizadores. É um tema que requer especial cuidado pois foca-se na interação homem-máquina, sendo que a atuação pode contribuir para um melhoramento do estado físico do utilizador. Existem assim diversos *trade-offs* a estudar e equacionar ao longo do processo. Por outro lado, o surgimento de técnicas de modelação de dados baseadas em ontologias, promete facilitar a modelação destes complexos ambientes e heterogéneos, facultando uma descrição robusta do domínio em estudo. Neste trabalho, foram estudadas as diversas técnicas a nível de modelação de dados e as arquiteturas em que estas podem assentar. A ontologia é uma destas técnicas e permite a descrição do conhecimento através de axiomas e imposição de relações entre estes. Esta tornou-se uma proposta efetiva para este tipo de sistemas graças ao seu provado valor na *Semantic Web*. Proporcionam ainda diversas funcionalidades extras face à tradicional implementação baseada em bases de dados, sendo um tema discutido nesta dissertação. Como forma de provar o valor das ontologias para modelação de conhecimento de um domínio, foi também desenvolvida uma ontologia de raiz, a Ont4AAL, onde são modelados os vários aspetos de um ambiente habitacional e a sua interação com os seus utilizadores. Por fim, foram retiradas as principais conclusões e aferido o valor real da utilização de ontologias em AAL.

Palavras-Chave: *AAL*, Modelação, Ontologia, *Software*, *Semantic Web*

Abstract

In this thesis report it is described the development of an Ontology for the application on ambient assisted living systems and all stages required to this project. It is also presented an overview of all techniques concerning this kind of solutions, from hardware techniques to software constrains. AAL solutions came from the social need to keep elderly and special people with special needs at their home, for as long as possible, keeping their health and maintaining their life quality. On the other hand, due to the research social impact there isn't yet a fully tested solution with easy home installation. This theme requires special attention because it is based on man-machine interactions, making actuation crucial to improve user's quality of life. Therefore, there are several trade-offs yet to be studied during the process. The emergence of some techniques based on ontologies, promise to help model complex and heterogeneous environments, providing a robust description of a certain domain. In this report were studied several modelling techniques and some architectures on which they can be based. Ontologies are one of those modelling techniques and they allow the description of knowledge, based on axioms and rule assertion. They became a proficient approach to AAL solutions due to its proven value with the Semantic Web. Compared to traditional data based solutions, they outcome in several aspects, bringing some new and interesting features, also discussed on this dissertation. To prove ontologies real capabilities to describe knowledge concerning a certain domain, it was created an ontology made from scratch, the Ont4AAL, where it's modelled the numerous aspects of a home building and possible interactions with its occupants.

To finish the work, were taken the principal conclusions and inferred the real benefits of using ontologies for AAL systems.

Keywords: *AAL*, Modelling, Ontology, *Software*, *Semantic Web*

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

AAL	*Ambient-Assisted Living*
AIO	*All in one*
API	*Application Programming Interface*
EDA	*Event-driven architecture*
ICT	*Information and communication technologies*
IT	*Information Technology*
OWL	*Web Ontology Language*
P2P	*Peer-to-Peer*
RDF	*Resource Description Framework*
SOA	*Service-Oriented Architecture*
SODA	*Service-Oriented Device Architecture*
SOPRANO	*Service-oriented Programmable Smart Environments for Older Europeans*
SWRL	*Semantic Web Rules Language*
WWW	*World Wide Web*
W3C	*World Wide Web Consortium*
XML	*eXtensible Markup Language*

CAPÍTULO 1

Introdução

Neste capítulo é traçado um enquadramento sobre o tema da presente dissertação. São também abordadas as principais motivações do surgimento deste trabalho de dissertação, qual a sua importância e de que forma poderá ajudar a resolver o problema em estudo. Por fim, é apresentada uma breve organização dos próximos capítulos e qual o seu conteúdo de forma geral.

1.1. Enquadramento

Os primeiros edifícios inteligentes surgiram em meados dos anos 80 e têm vindo a aumentar de número a cada ano. Segundo a *Abi Reasearch* em 2017 [1], 90 milhões das casas abarcarão sistemas de automação doméstica. Tal representa um crescimento de 60% entre 2012 e 2017 face aos correntes números. Ainda no mesmo artigo, estima-se que 1.5 milhões de sistemas de automação doméstica tenham sido instalados em habitações dos Estados Unidos da América. Desta forma, vários esforços têm sido levados a cabo para que estes sistemas possam evoluir em diversas vertentes. Uma destas evoluções dos tradicionais sistemas de domótica são as vocacionadas para o auxílio de pessoas idosas ou com necessidades especiais/específicas. A estes programas, têm sido providos inúmeros incentivos provenientes da União Europeia. Um destes exemplos é o "*Ambient Assisted Living Joint Programme*" [2], um programa que tem como objetivo criar melhores condições de vida para pessoas idosas. Este, vem desta forma auxiliar o combate aos problemas inerentes ao envelhecimento da população e a sua qualidade de vida pessoal e social. No mesmo ano, em paralelo com a fundação deste programa (meados de 2008), foi criado o projeto *SOPRANO* (*Service-oriented Programmable Smart Environments for Older Europeans*) [3], não estando diretamente ligado ao programa mencionado, este visava identificar quais as particularidades presentes nos sistemas de AAL que os utilizadores finais mais apreciavam e consideravam mais *user-friendly* [4][5][6]. Neste estudo foram recolhidos dados interessantes e pouco expectáveis. Por exemplo, os desenvolvedores acreditavam que cada ícone nos painéis de controlo deveria ter uma cor específica consoante a sua

função. No entanto, os resultados provaram que, contrariamente a esta situação, os utilizadores preferiam a substituição desses ícones por números, semelhantes aos dos comandos de televisão.

A utilização de sistemas de automação doméstica requer um avultado volume de dados a serem manipulados todos os segundos, pelo que a forma como se lida com a informação tem-se provado um verdadeiro desafio [7]. De facto, é crucial construir uma metodologia que suporte o reconhecimento e tratamento de dados e conheça o processo de construção do sistema. Esta necessidade é também importante para que se evolua no sentido de haverem ambientes mais eficientes e capazes, assentando assim numa melhor gestão de dados realizada.

À parte do volume e difícil gestão de dados, existem ainda outros problemas inerentes à integração de sistemas inteligentes na casa dos diversos utilizadores. Vivendo habitualmente em família e havendo uma vasta diversidade etária, de género, de complexidade física, entre outros fatores, a construção de um sistema adaptável para todos, representa outro problema na fase de desenho das soluções de *AAL*. No entanto, para combater tal problema, na última década tem sido utilizado o conceito de *design* universal [8]. Representa a construção de produtos, ambientes e sistemas para todas as idades, capacidades e tamanhos, e tem-se tornado um assunto de maior importância, tendo em conta o crescimento da população idosa ou com limitações.

Postos alguns dos desafios inerentes ao desenvolvimento de soluções de *AAL*, existem ainda certos problemas que, de modo geral, afetam todas os sistemas atuais de domótica [9] e [10] :

1. São produzidos e distribuídos por vários produtores de componentes, tendo cada um os seus objetivos e políticas de mercado;
2. Representam pequenas evoluções dos componentes elétricos já existentes (por exemplo relés). Isto implica a impossibilidade de acrescentar inteligência nativa, para além de simples cenários de automação.

O primeiro problema pode acarretar ainda problemas de interoperabilidade. Isto é, cada fornecedor pode desenvolver um protocolo proprietário, não sendo possível, maior parte das vezes, conectar dispositivos de várias fontes a não ser através de *gateways* ou adaptadores específicos. Este problema não existirá caso optem pela utilização de protocolos *standard*.

Verifica-se desta forma dois tipos de problemas de interoperabilidade: a nível dos dados e a nível físico. No que diz respeito aos problemas de interoperabilidade a nível

dos dados, a classificação da informação e sua acessibilidade tem sido alvo de avultados estudos.

Na demanda pela resolução do problema de interoperabilidade a nível dos dados, têm sido introduzidas várias soluções [11][12]. A par de muitas outras soluções, adaptou-se o conceito de ontologia, até então pouco relacionado com engenharia. As ontologias representam uma forma estruturada de organizar os dados de um sistema descrevendo relações entre os vários atores dentro do mesmo [13]. Juntando as funcionalidades já presentes na tecnologia, à otimização que as ontologias podem providenciar, criam-se sistemas novos e mais eficientes [14].Tal como será referenciado em 3.3.1, a eficiência da utilização de ontologias deve-se a vários fatores: as linguagens ontológicas são de fácil compreensão e leitura; os modelos ontológicos representam explicitamente apenas o necessário podem ser utilizadas como uma parte substancial de código executável.

O maior contributo da utilização de ontologias ocorreu recentemente no domínio da "*Semantic Web*". Esta representa um movimento colaborativo conduzido pelo *Word Wide Web Consortium* (W3C) [15]. Este *standard* promove uma formatação de dados semelhantes pela *Word Wide Web* (WWW). Tem como princípios a construção de uma linguagem descritiva utilizando *standards* semânticos, bem como permitir que esta seja processada pela máquina [16]. A ideia deste movimento é transportar os princípios já utilizados nos documentos da Web para os dados. Assim, estes poderiam ser acedidos usando a arquitetura Web, estando ao mesmo tempo relacionados entre si. Desta forma, a *Semantic Web* assenta, fortemente na estrutura formal das ontologias e na forma como lida com o objetivo de uma melhor análise e fácil portabilidade. Assim, para o sucesso deste *standard*, depende-se da proliferação das ontologias, da fácil a construção destas, evitando desta forma as dificuldades de acesso aos dados.

Além da *Semantic Web*, existem ainda inúmeros sistemas que fornecem serviços já construídos com base em ontologias. Um exemplo desta abordagem é [17], um recomendador de dietas que tem em atenção o estado físico do utilizador. Assim, uma pessoa diabética pode controlar as suas refeições tendo em conta os seus valores de açúcar. Além destes cuidados, é ainda possível que o utilizador escolha quais os seus alimentos favoritos, criando dinamicamente refeições que os contemplem.

Na temática da indústria militar, o mesmo conjunto de conceitos foram utilizados em [18] para o cálculo de uma rota dinâmica em tempo real. Esta serve um pelotão em movimento que ultrapassa uma zona em guerra. Mediante a presença de fatores externos, tais como clima ou movimentação de tropas inimigas, a rota é recalculada

autonomamente. Desta forma, o risco de ataque aos militares é reduzida, auxiliando-os nas mais variadas tarefas.

Os exemplos apresentados representam modelos testados e que podem ser reutilizados para o corrente projeto**.** Além dos já mencionados, existem outros exemplos importantes como é o caso de [19][20].

1.2. Motivações

Os sistemas de domótica envolvem cada vez mais um volumoso número de parâmetros a serem ponderados a cada instante. O crescente volume de informação a tratar requer uma classificação e organização de forma estruturada e lógica. A utilização de ontologias tenta solucionar o problema, fornecendo uma abordagem diferente para a representação dos dados. Assim, a possibilidade de utilizar um conceito pouco proliferado nos sistemas de domótica, como forma de obter um escalonamento organizado da informação, é de especial importância. A aplicação de ontologias a sistemas de automação é um conceito com poucos anos de utilização, tendo sido os primeiros artigos publicados datados de meados de 1990. No entanto, este conceito já era conhecido noutros ramos da ciência, havendo referências a artigos de 1980. Desta forma, prova-se um tema de elevada importância, visto a sua gama aplicacional e o impacto que a sua integração em sistemas de AAL poderá representar. Os avanços conseguidos nesta área representam ainda melhorias significativas no quotidiano de diversos humanos, que vivem em condições desfavoráveis ou que moram em lares devido à falta de condições na sua habitual casa. Surge também como forma de resolução de alguns problemas sociais, tais como os avultados gastos na área da saúde e cuidados ao domicílio.

1.3. Objetivos e Contribuições

A tese proposta contém um aprofundado estudo sobre os componentes integrantes de um sistema de *Ambient Assisted Living*. Inclui no seu conteúdo as soluções tecnológicas a nível de arquitetura para estes sistemas, bem como de modelação e implementação ao nível do *software*. Não tendo sido objetivo final implementar uma solução totalmente integrada, *hardware* mais *software*, é requisito que se modele um ambiente habitacional utilizando uma técnica recente na área da domótica, denominada de ontologia. No final, esta é ainda validada através da instanciação direta de vários componentes inteligentes, bens não controláveis e utilizadores.

A contribuição deste trabalho assenta fundamentalmente na exploração das diversas técnicas de implementação de *hardware* e *software* para sistemas de *AAL*, servindo ainda de ponto de partida para uma solução integrada completa. No final do trabalho, poder-se-á utilizar a ontologia construída para testes num *middleware* apropriado e testar em situações reais.

1.4. Organização e Estrutura da Tese

A tese apresentada encontra-se dividida em seis capítulos, incluindo introdução e conclusão. No capítulo 1, a introdução, é efetuado um levantamento das necessidades empíricas de soluções de *AAL*, as propostas já existentes e quais as suas lacunas. No segundo capitulo é caracterizada a vivência assistida, qual o seu domínio, desafios e qualidades necessárias. O seguinte capítulo trata das diversas técnicas de modelação conhecidas para este tipo de soluções. Esta secção encontra-se ainda dividida em estilos arquiteturais e modelos semânticos. Enquanto o primeiro trata dos aspetos físicos das soluções, tal como a sua arquitetura, o segundo lida com a modelação do ambiente habitacional onde se encontram inseridos. É ainda efetuada uma breve comparação entre as técnicas tradicionais de modelação (baseadas em bases de dados) e a técnica imergente de ontologias. No quarto capítulo é apresentada a ontologia construída, Ont4AAL. São abordados todos os aspetos fundamentais desta, desde as suas classes às suas propriedades. No capítulo da implementação e resultados encontra-se o resultado do trabalho efetuado. É instanciada uma habitação modelo e discutidos todos os seus aspetos fundamentais. Por fim, no capítulo das conclusões são apresentadas todas as ilações finais, fazendo um ponto da situação e sugerindo possíveis melhorias.

CAPÍTULO 2

Ambient Assisted Living

2.1. Introdução

A população europeia está a envelhecer e graças aos avanços médicos conseguidos nas últimas décadas, a esperança média de vida aumentou de 55 anos em 1920 para 80, atualmente. Passado o "*baby boom*", acredita-se que o número de pessoas entre 65 e 80 anos crescerá aproximadamente 40% entre 2010 e 2030 [21]. Esta alteração demográfica implica um grande desafio para a sociedade e para a economia europeia. O envelhecimento da população pressiona ainda a sustentabilidade dos sistemas de saúde dos seus países, comprometendo assim a capacidade de fornecimento de um serviço de saúde equivalente para todos os seus habitantes. A necessidade de reajuste do sistema de saúde representa custos avultados que poderão ser contornados, mas não negligenciados pelo estado, através da aplicação de tecnologias emergentes, em prol de sistemas de AAL. Estas permitem que se consiga cuidar e tratar pessoas a um custo menor e, fundamentalmente, nas suas casas. Assim, começam a surgir soluções *All-in-One* (AIO) tais como o *Aware Home* [22] ou o *I-Living* [23], construídas para combater problemas relacionados com falhas de memória, visão, audição, mobilidade e, essencialmente, perda de autonomia mais vincada em idades mais avançadas. Estes sistemas são de elevada importância, pois possibilitam que os seus utilizadores mantenham a sua integração na comunidade ativa. Os principais objetivos destas soluções, segundo Becker [10] e o *Ambient Assisted Living Joint Programme* são: (i) aumentar a qualidade de vida e cuidados a pessoas com necessidades especiais, (ii) reduzir a necessidade de assistência externa, (iii) reduzir os custos de saúde e cuidados associados ao individuo e à sociedade, (iv) evitar a estigmatização da pessoa. Estes assentam, essencialmente, na filosofia por trás do *AAL* que é fortemente influenciada pelo paradigma "*Ambient Intelligence*" [24] que descreve a colocação de um individuo num ambiente inteligente e automatizado e que garante uma assistência proactiva e possibilita uma interação totalmente natural. Estes ambientes são arquitetados de forma a consistirem em grupos de dispositivos interconectados na mesma rede, seguindo as

mesmas normas de ação, podendo reagir ativamente perante a condição do utilizador ou a sua interação com o sistema.

Apesar de todos os protótipos, aproximações e tecnologias promissoras, o AAL é mais uma visão de o que uma realidade atual. O desafio tecnológico que representam a par com o impacto social que desempenham, requer que se tenha especial cuidado com a integração de várias soluções no mercado. De facto, o surgimento de várias soluções provenientes de diversas áreas e com objetivos de qualidade diferente terão de se unir como forma a cumprir os objetivos deste projeto. Este fenómeno pode ser considerado o maior problema/inibidor à sua proliferação num futuro próximo.

Nos próximos subcapítulos, será apresentado o domínio ao nível de serviços das soluções de AAL, as suas possíveis classificações, qualidades e desafios.

2.2. Domínio do *Ambient-Assisted Living*

Nesta secção é caracterizado o domínio ao nível de serviços do *Ambient-Assisted Living*. Inicialmente será descrito o esquema de classificação para serviços de assistência à vida, sendo que de seguida são identificados alguns desafios quer para o utilizador quer para o ambiente a serem tratados por qualquer solução *ICT-based* de AAL. Por fim, é aferida uma lista das qualidades essenciais que um sistema deverá ter, tendo em conta os desafios que serão relatados.

2.2.1. Esquema de classificação de AAL

O leque de serviços que poderão ser associados ao domínio de assistência à vida é enorme. Este engloba qualquer tipo de serviço de assistência que facilitará o quotidiano do utilizador. Na Tabela 2.1, encontra-se estruturado o domínio do *AAL* em três subdomínios estereotipados [10]. A análise desta, implica que se divida o esquema em duas grandes áreas de atuação - interior e exterior. Os serviços de assistência no interior serão sempre confinados a áreas bem definidas: casas, hospitais, etc.

Tabela 2.1 - Esquema de classificação de AAL.

	Tratamento de Emergência	Melhorias ao nível da autonomia	Conforto
Interior	Previsão / Deteção / Prevenção	Beber / Comer / Cozinhar / Limpar / Vestir / Medicação	Serviços de logística / Serviços de procura de objetos / Serviços de info-entretenimento

Exterior	Previsão / Deteção / Prevenção	Compras / Viagens	Serviços de transporte / Serviços de navegação

Os serviços de assistência no interior serão sempre confinados a áreas bem definidas: casas, hospitais, etc. Posto isto, a assistência no interior pode ser baseada na instalação de *software / hardware* em locais pré-definidos, providenciando um ambiente com poucas variações. Por outro lado, a assistência no exterior é mais variável e imprevisível. Esta auxilia o utilizador enquanto este executa tarefas como ir às compras, viajar ou desempenhar outro tipo de atividade social. Estes serviços têm de lidar com condições ambientais altamente instáveis onde, tecnologicamente poderá ser complicado prestar suporte ao utilizador.

Da tabela apresentada pode-se ainda retirar uma organização mais vocacionada para o tipo de serviços que são prestados, em vez das áreas em que estes atuam. Desta forma, tal como ilustrado, foi levado a cabo uma divisão em três tipos de serviços [10]:

I. Serviços de tratamento de emergência – Este tipo de funcionalidades tem como objetivo uma previsão, recuperação e alerta rápido de condições críticas que possam resultar em situações de emergência;

II. Serviços de otimização de autonomia – Provêm funcionalidades de suporte à total autonomia ao utilizador;

III. Serviços de conforto - Comporta todas as atividades não contempladas em (I) e (II) e melhoram o quotidiano do utilizador não sendo estritamente necessárias para o seu bem-estar físico.

É seguro afirmar que, dos três tipos de serviços disponibilizados, os dois primeiros detêm uma maior importância e impacto social, sendo uma prioridade a nível de implementação. É importante ressalvar que a classificação do serviço depende, essencialmente, da capacidade mental e motora da pessoa a ser assistida, bem como do ambiente em que a solução se encontra inserida. Isto é, o sistema de AAL terá de se adaptar à necessidade específica de cada utilizador, não sendo homogénea a todos os seus benificiários. Esta classificação pode variar com o tempo, devido às mudanças que podem ocorrer na saúde do utilizador. Idealmente, as três caracterizações deveriam ser implementadas no sistema final, visto o seu provado benefício no quotidiano do humano [25].

2.3. Desafios às soluções de *Ambient Assisted Living*

Após a classificação do sistema de AAL consoante o meio em que o suporte é necessário, é importante compreender quais os desafios que estas soluções têm de lidar quer internamente (problemas tecnológicos), quer externamente (ambiente onde se encontram inseridos, especificidades do utilizador) no que diz respeito à sua integração. Estes, encontram-se referenciados em vários artigos, tais como em [26][27] e são referenciados de seguida**:**

I. Diminuídas e/ou decrescentes capacidades – Representa não só um desafio para as soluções de *AAL*, como também uma das principais razões da necessidade destas. Alguns problemas de saúde associados à pessoa assistida proporcionam-lhe fracas capacidades motoras implicando uma assistência externa. A acrescentar a este problema, há ainda o facto de estas poderem piorar com o tempo.

II. Necessidades específicas para cada utilizador – As debilidades, objetivos e hábitos de cada utilizador são bastante díspares quer a nível de utilização quer a nível de aceitação à intrusão no seu quotidiano. Desta forma, não há uma solução única e homogénea para todos os utilizadores.

III. Recursos limitados – Os recursos humanos e financeiros para apoio a pessoas mais debilitadas estão cada vez mais limitados e prometem escassear ainda mais nos próximos anos.

IV. Baixa tolerância para problemas técnicos – Existe uma baixa aceitação de soluções de AAL caso a pessoa assistida encontre uma dificuldade em a utilizar.

V. Envolvimento na vida social ativa – Apesar das necessidades especiais que necessitam, as pessoas assistidas desejam um envolvimento ativo na comunidade o maior tempo possível. Não é aceitável a substituição do contato humano por uma assistência totalmente técnica e automatizada.

VI. Noção de manter o controlo – Apesar da necessidade de assistência, estas pessoas gostam de manter o controlo sobre a assistência que lhes é prestada.

VII. Evitar a estigmatização – Deverão ser evitados os dispositivos externos visíveis para evitar a estigmatização e aumentar a aceitabilidade das soluções de AAL.

VIII. Manter a privacidade – A informação referente ao estado de saúde passado e presente da pessoa assistida não deverá ser pública, limitando a sua divulgação apenas às pessoas e instituições que lhe prestam assistência ativa.

Os pontos apresentados representarem uma vista geral dos problemas que os desenvolvedores de soluções de AAL terão de enfrentar, sendo agrupá-los e distingui-los em apenas três desafios chave a encontrar neste tipo de sistemas [28]: (i) dinâmica da disponibilidade de serviços; (ii) Mapeamento dos serviços; (iii) aceitação do utilizador. Estes pontos serão discutidos de seguida separadamente.

(i) Apesar do emprego de profissionais especializados em cuidados a idosos ajudar à redução das necessidades sociais existentes, a variabilidade de necessidades a nível temporal e funcional complica a sua organização sendo necessário um complemento adicional, como por exemplo a utilização de soluções de AAL. Desta forma, gerir esta dinâmica de necessidades representa um grande desafio. As arquiteturas orientadas a serviços, que serão discutidas no seguinte capítulo, podem representar uma solução para lidar com o problema mencionado. A sua flexibilidade e arquitetura *standardizada* que suporta a ligação a vários serviços, mostra elevado potencial no combate a esta dinâmica. A aplicação desta arquitetura em plataformas como o *OSGi* [29], pode ajudar a estabelecer um *framework* onde os dispositivos inteligentes podem ser integrados em conjunto, automaticamente chamados, ligados ou parados. A utilização desta plataforma para a construção de ambientes seguros estão também reportados em [30].

(ii) Um dos grandes desafios de sistemas de *AAL* é o mapeamento dos serviços disponíveis/requisitados. A base do mapeamento de serviços é a descrição dos serviços em si. É necessária a utilização de uma base de dados semântica para descrever precisamente os serviços possíveis, sendo necessário o desenvolvimento de livrarias ontológicas que retratem o domínio do conhecimento sobre um ambiente de cuidados especiais. Após a modelação do conhecimento neste domínio de um ponto de vista conceptual, podem ser aplicadas técnicas de mapeamento de serviços aos modelos semânticos. O *OWL-S* [31] é, atualmente, a tecnologia mais utilizada neste domínio. É capaz de facultar um *framework* capaz de semanticamente descrever *Web Services* de várias perspetivas, por exemplo, procura de serviços, invocação e composição. Existem algumas ferramentas de mapeamento de serviços para mapear serviços *OWL-S*, tais como o *OWL-S Matcher* [32], *OWL-S UDDI/Matchmaker* [33] e *OWLS-MX Matchmaker* [34]. O problema das duas primeiras ferramentas é que o processo de mapeamento demora muito tempo enquanto a desvantagem do último é o uso de memória intensivo [35]. Estas ferramentas servem como base

de investigação no campo do mapeamento de serviços de *Web Services*, enquanto outros mecanismos de mapeamento se encontram em desenvolvimento.

(iii) O último desafio chave para as soluções de AAL é o problema de aceitação que estes tipos de sistemas normalmente se deparam no ceio da comunidade necessitada. É importante aferir a melhor metodologia de sensibilização para a necessidade destes sistemas, estudando quais os estímulos necessários para aumentar a sua taxa de aceitação. Assim, um sistema verdadeiramente eficiente de AAL não poderá negligenciar as contribuições provenientes da sociedade, de todas as formas, com a participação informal de profissionais de cuidados especializados (amadores ou profissionais) e os próprios afetados em si. No entanto, atualmente este ideal tem sido contrariado pela baixa taxa de aceitação deste tipo de sistemas sendo que tal adversidade deve-se essencialmente a dois fatores [28] – psicológicos (a.) e tecnológicos (b.).

a. Com o avanço da idade surge alguma frustração relativa à perda progressiva de algumas capacidades físicas mas o maior problema prende-se com o fator psicológico: tornam-se utilizadores dependentes de serviços sociais em vez de continuarem como membros ativos. Desta forma, baixa também a sua autoestima. Grande parte dos sistemas de AAL para pessoas idosas, consideram os seus utilizadores como pessoas fracas e assistidas passivamente por outras, negligenciando os contributos que possam dar. Para os criadores destes sistemas, ser capaz de manter certo grau de independência dos utilizadores sem trazer muita carga para a sociedade é um verdadeiro desafio. Assim, um sistema que recorra à partição humana encoraja os idosos a participar ativamente em atividades de grupo e possivelmente partilhar a sua experiência com gerações mais jovens, p.e, alguns problemas no trabalho ou escola [36].
b. As pessoas idosas são normalmente receosas no que diz respeito às novas tecnologias. Como forma de os adaptar aos sistemas de *AAL*, é necessário que se criem interfaces *user-friendly* e fornecer formação apropriada para os seus utilizadores. Desenvolver interfaces adaptativas, naturais e multimodais são o maior desafio ao nível da interação para sistemas de *AAL* [37]. É também aconselhável que se envolvam os

utilizadores na construção destas interfaces antes destes necessitarem delas [38].

2.4. Qualidades da solução e dos serviços

Acabada a identificação das classificações dos sistemas de AAL consoante o ambiente onde se encontram inseridos e quais os desafios com que estes terão de lidar quer a nível do utilizador quer ao nível ambiente de ação, é possível aferir assim uma lista de características fundamentais a reter em todas as soluções presentes e futuras de assistência a utilizadores idosos ou com necessidades especiais. Desta forma, destacam-se as seguintes, referindo ainda quais os pontos da secção anterior que contribuem para o seu realce:

I. Custo (2.3.III) **–** A construção de soluções de tão grande dimensão como as discutidas até este ponto do trabalho, implicam custos avançados de investigação e desenvolvimento, logísticos, inerentes à produção em grande escala e instalação dos sistemas na casa do utilizador final. Serão ainda acrescidos valores de manutenção e de sustentação dos próprios serviços (possível mensalidade da utilização de serviços). Apesar destes fatores, é importante que estes sistemas possuam valores finais de venda ao público o mais baixo possível, não só pelas diferenças económicas e culturais dos diversos países e dos diversos utilizadores, mas também pelo papel vital que podem vir a representar para o quotidiano de alguns seres humanos.

II. Fácil uso e boa acessibilidade para o utilizador (2.3.I, 2.3.IV, 2.3.VI, 2.3.VII) **–** Se os serviços de assistência requerem algum tipo de interação do/com o utilizador, então este deverá ser capaz de demonstrar a sua vontade, fornecendo-lhe uma experiência agradável e simples. Isto é, o sistema final não poderá ser complicado à utilização, idealmente sem qualquer contacto direto do utilizador, realizando grande parte das suas tarefas de forma transparente, aumentando assim a sua probabilidade de adoção e aceitação.

III. Adequabilidade (2.3.I, 2.3.II, 2.3.III,2.3.IV, 2.3.V, 2.3.VI, 2.3.VII) – Os serviços deverão ser adequados à necessidade da pessoa a ser assistida. A pessoa deverá possuir um benefício direto inerente à sua utilização. Caso contrário, a sua aceitação diminuirá com o tempo. De facto, se o utilizador não identificar *à priori* vantagem na utilização do sistema, a sua tolerância e conforto para com

este ao longo do tempo diminuirá, aumentando o grau de desconfiança relativo ao seu valor real.

IV. Confiança / Segurança (2.3.I, 2.3.IV, 2.3.VI, 2.3.VIII) – As soluções deverão ser robustas contra todo o tipo de mau uso ou erros e os serviços deverão manter-se ativos e funcionais mesmo na presença de falhas de *hardware* ou escassez de recursos. Em todas as situações a segurança da pessoa assistida deverá ser prioridade máxima. Não deverá assim ser descuidado todo o ambiente em que se engloba o sistema, tendo em conta todos os potenciais danos para o utilizador, garantindo que estes sejam totalmente suprimidos, mesmo que isso ponha em causa a segurança do próprio sistema.

V. Adaptabilidade (2.3.I, 2.3.II, 2.3.III, 2.3.IV) – Os sistemas deverão ser capazes de se adaptar mesmo quando em execução. Este desafio é das características mais importantes a deter neste tipo de soluções. Para suportar tal requisito, os sistemas terão de não só monitorizar o utilizador mas também o ambiente e a si mesmo. Assim, terão de conseguir efetuar tarefas elementares tais como: autoconfiguração (dispensando interferência do utilizador); autorreparação (dispensando interferência de técnicos especializados); auto-otimizaçao e autoproteção (não descuidando o ponto III).

VI. Expansibilidade (2.3.I, 2.3.II, 2.3.III) – O sistema deverá suportar de forma simples e "*plug-and-play*" a adição de novos dispositivos ou serviços quando em execução, como forma de adaptação, resultado da alteração dos requisitos ao longo do tempo.

VII. Eficiência (2.3.I, 2.3.II, 2.3.III) – Todos os recursos são escassos ou merecedores de uma utilização responsável. Desta forma, nestas soluções deverão estar presentes diversas metodologias de gestão de recursos eficiente, quer seja a nível de energia, de rede ou qualquer outro tipo de recursos.

VIII. Heterogeneidade (2.3.I, 2.3.II, 2.3.III, 2.3.V, 2.3.VII) – O sistema é geralmente composto por vários subsistemas provenientes de diferentes fabricantes.

Estas representam grande parte das qualidades que terão de estar presentes nos sistemas de *AAL*. A sua aplicação poderá depender do contexto e dos utilizadores presentes no ambiente.

2.5. Conclusão

Naturalmente, por em prática tantos conceitos pode representar alguma dificuldade técnica pelo que é necessário estabelecer um compromisso, levando a cabo *tradeoffs* que sejam considerados essenciais. Isto é, nem sempre é possível tecnicamente juntar no mesmo sistema todas as qualidades mencionadas anteriormente. Tal facto complica a construção de soluções de *AAL* de forma substancial, sendo necessário compreender quais as qualidades que mais contribuem para a eficiência do sistema final. Um modelo de relações entre as qualidades (2.4) e suas dependências (2.2 e 2.3) será útil, mas tal requer um estudo intenso e interdisciplinar que deverá ser levado a cabo nos próximos anos. Até ao momento, os primeiros protótipos utilizam a filosofia "*just enough quality*", satisfazendo necessidades complementares dos utilizadores. Assim e visto que os serviços de assistência providenciados por humanos não representam 100% do seu tempo, pequenos problemas de a qualidade nas soluções de AAL deverão ser tolerados, desde que compensados pelos assistentes humanos, não comprometendo a qualidade de vida das pessoas assistidas.

CAPÍTULO 3

Técnicas de modelação em AAL

3.1. Introdução

Nesta secção é descrita a forma como as características referidas no capítulo 2 são endereçadas pelas técnicas mencionadas neste capítulo. Primeiro é apresentada uma vista geral das várias propostas/estilos arquiteturais existentes e depois é feita uma discussão de alguns padrões para endereçar adaptabilidade e a capacidade computacional dos sistemas.

As arquiteturas são os artefactos centrais que tratam de requisitos como o preço, performance e essencialmente as funcionalidades da solução final. Em qualquer sistema de *software*, a arquitetura representa o papel fundamental à qualidade do sistema. Esta representa a estrutura ou estruturas de um sistema, que compreende elementos de *software*, as suas propriedades externas visíveis e as relações entre elas.

Atualmente, não existe nenhum modelo arquitetural que se possa considerar referência e que seja amplamente aceite para sistemas de AAL. Na realidade, são utilizadas várias configurações para atingir as metas e requisitos necessários ao sistema. Assim, nos seguintes subcapitulos serão apresentadas algumas das soluções. De notar que o tópico mais aprofundado será o das ontologias devido às suas mais recentes contribuições no AAL e na *Web Semantics* [39][40]. Inicialmente serão abordados os estilos arquiteturais existentes para soluções de AAL e posteriormente modelos semânticos para a modelação do conhecimento do ambiente habitacional. Por fim, serão apresentados algumas integrações de estilos arquiteturais com modelos semânticos, como é o caso da implementação *SOA + OWL*.

3.2. Estilos arquiteturais

É possível construir várias soluções para realizar a decomposição conceptual das funcionalidades já descritas, presentes num sistema de AAL. De uma perspetiva física, ao nível do *hardware*, a topologia dos sistemas de AAL consistem num número imenso de nós de interação, desde pequenos sensores a telemóveis ou de um leque de sistemas

embebidos com pouco poder computacional até grandes máquinas de computação. Tal encontra-se ilustrado na Figura 3.1.

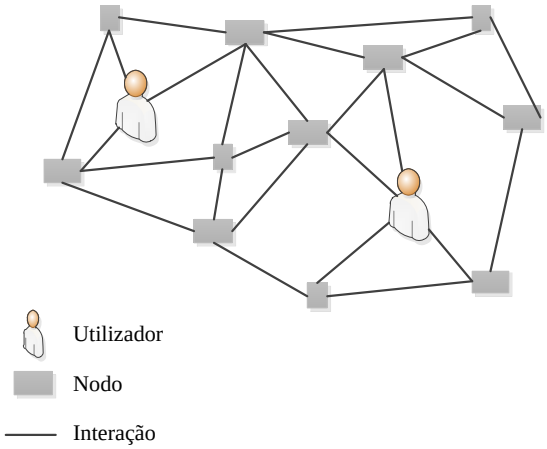


Figura 3.1 - Estrutura de sistemas de AAL.

O maior desafio dos desenvolvedores de soluções para AAL é a integração da diversidade no tempo e no espaço onde estes conceitos assentam. Definir os estilos arquiteturais a aplicar é uma tarefa complicada visto que este influencia diretamente as qualidades do sistema.

3.2.1. *Service-Oriented Architecture*

A arquitetura orientada a serviços é considerada o estilo arquitetural da próxima década [41], especialmente nos sistemas da informação. Esta foi concebida para a integração de aplicações empresariais. Ultimamente tem sido bastante utilizada em sistemas de AAL [2]. A qualidade mais notável das *Service-Oriented Architecture* (SOA)[28] é o suporte à configuração (adição e remoção) de serviços ao sistema sem afetar os demais. Tal contribui para a expansibilidade e modificabilidade do sistema mesmo depois de instalado. É efetuada uma separação entre a implementação (componentes) e o chamado contrato (serviços). Através desta configuração, a complexidade inerente aos sensores, atores e subsistemas é mais controlável. Ainda, o princípio de *loose coupling* facilita a integração de novos componentes que podem ser adicionados ao sistema de acordo com as necessidades do utilizador. Este princípio dita que cada componente, também denominado de elemento, deverá depender dos outros o mínimo possível. Possui também características inerentes à descoberta de serviços que que possibilitam uma autoconfiguração do sistema, requisito fundamental de qualquer solução de AAL (2.4 - V).

A Everware-*CBDI* [42] define as *SOA* como sendo um estilo resultante do uso de certas politicas, práticas e *frameworks* que facultam serviços respeitando certas

normas. Esta definição destaca que qualquer forma de serviço pode ser publicado através de uma interface presente num *Web Service*. Apesar deste facto, atributos de alto nível como a reusabilidade e a independência da implementação, são apenas atingidas recorrendo a alguma ciência no desenho e no processo de construção que está explicitamente direcionado ao aumento de objetivos, por detrás das operações básicas fornecidas pela utilização de *Web Services*. Assim, é possível aferir que o aparecimento dos *Web Services*, há cerca de 3 anos atrás, começou com o surgimento das técnicas pró *Service Oriented*. No entanto, estes serviços representam um pequeno passo num processo bastante extenso. A noção de serviço, é uma parte integral da constituição de componentes e, é claro que arquiteturas distribuídas, foram tentativas de implementação de arquiteturas orientadas a serviços (*SOA*). Desta forma os *Web Services* são parte de um conceito muito mais abrangente que a arquitetura orientada a serviços. De facto, os *Web Services* não são componentes obrigatórios das *SOA*, apesar de serem cada vez mais utilizados em conjunto. Por outro lado, as *SOA* são um conceito potencialmente mais abrangente no que diz respeito ao seu raio de ação, ao contrário dos *Web Services* que são uma simples definição de uma implementação de um serviço. Esta arquitetura não trata apenas da qualidade do serviço do ponto de vista do fornecedor e consumidor do serviço sendo que os *Web Services* representam meramente a sua implementação. Não é apenas uma arquitetura de serviços visto pela perspetiva tecnológica, mas sim pelas suas normas, práticas e *framework* que garante que os serviços corretos são consumidos e disponibilizados. Desta forma, é necessário um *framework* capaz de compreender o que é que constitui um serviço e as suas características fundamentais. Considerando que se necessita de vários níveis de utilização, é requerido um conjunto de princípios no que diz respeito à orientação de serviços. É então possível discernir dois princípios [43]:

- Princípios vocacionados à interface – Neutralidade da tecnologia, *standardização* e consumibilidade;

- Princípios vocacionados ao desenvolvimento – Estes representam serviços de qualidade, com mais objetivos, respeitando necessidades comerciais, fazendo com que estes sejam de fácil utilização, adaptáveis e de fácil gestão.

O segundo princípio tem vindo a ser tratado por alguns estudos, tendo conseguidos resultados promissores neste campo [43]. No entanto, estima-se que grande parte das empresas dificilmente conseguirá justificar tamanho grau de exigência e disciplina inerentes à conversão das suas tradicionais soluções, para uma arquitetura baseada em

serviços. Enquanto alguns componentes de alto nível foram criados para certas aplicações, onde é claro o nível de reutilização e partilha, na generalidade tem sido difícil incorrer num investimento de tal ordem. É importante salientar que nem todos os serviços necessitam as características já mencionadas (caso da reutilização ou partilha). No entanto, se é importante que um serviço possa ser utilizado por múltiplos utilizadores a sua especificação necessita de ser generalizada e terá de se abstrair da sua implementação, não tendo os desenvolvedores de saber qual o modelo ou regras por detrás dos mesmos.

A especificação de regras que as aplicações precisam de cumprir, necessitam de ser formalmente definidas, precisas e os seus serviços devem estar disponíveis sob um nível relevante de granularidade, que combine uma flexibilidade apropriada com facilidade de integração nos processos empresariais.

Sendo que grande parte das soluções baseadas em SOA se materializam através dos *Web Services*, é importante aferir quais as características que cada uma proporciona para a qualidade da solução final. Assim, na Tabela 3.1 encontram-se categorizadas as características da arquitetura (SOA) e da implementação (Web Services) e que papeis representam na solução construída.

Tabela 3.1 - *Web Services* e *SOA*.

Proporcionado por *Web Services*	*Tecnologia Neutra*	Independência de plataforma.
	Standardizado	Baseado em protocolos *standardizados*.
	Utilizável	Permite a descoberta e utilização automática.
Proporcionado por SOA	*Reutilizável*	Reutilização do serviço e não da sua implementação.
	Abstraível	Publicação das especificações do serviço e não da sua implementação.
	Publicado	Publicação das especificações do serviço e não da sua implementação.
	Formal	Contrato de obrigações formais entre o fornecedor do serviço e o consumidor.
	Relevante	Granularidade reconhecida pelo utilizador como um serviço importante.

Sumarizando as características apresentadas na tabela, é possível retirar três benefícios desta arquitetura:

- Sincronização entre o negócio (pessoas) e a perspetiva da implementação IT;

- Um serviço bem construído fornece uma unidade de gerenciamento deste relativo à sua utilização no negócio;
- Quando se abstrai da implementação é possível considerar várias alternativas para os modelos de entrega e colaboração.

Estas características representam um sumário breve das potencialidades desta arquitetura, sendo importante não esquecer outras virtudes mencionadas anteriormente, como o caso do *loose coupling*, da reutilização e partilha.

3.2.2. Service-Oriented Device Architecture (SODA)

A arquitetura de dispositivos orientados a serviços é uma adaptação da já mencionada SOA. O objetivo desta configuração é garantir a interoperabilidade entre todos os seus dispositivos, possibilitando um alto nível de abstração a nível físico, para o sistema final. Desta forma, o acesso ao infindável número de dispositivos presentes num AAL poderá ser feito de forma transparente ao *design* do sistema e sua implementação, abstraindo a interação entre sistemas de informação e o mundo físico.

A metodologia SODA [44] para desenhar e construir *software* distribuído implica integrar um vasto leque de dispositivos físicos em sistemas de informação. No seu nível mais simples, SODA permite que os programadores lidem com sensores e atuadores tal como os serviços são usados nas arquiteturas SOA atuais. Esta arquitetura foca-se assim, na camada intermédia que separa o mundo físico do mundo real, possibilitando um acesso transparente entre estes.

Uma análise simplificada da SODA, certamente colocará poucas restrições no tipo de dispositivo a usar. O grau de complexidade destes dispositivos pode variar e ir do mais simples sensor aos mais complexos equipamentos de diagnóstico. Seja a fornecer a localização de uma carga crítica, o nível de açúcar no sangue ou o estado do sistema de distribuição elétrico, os dispositivos partilham em comum é os seus dados, as suas funções e os eventos são serviços críticos para o sistema onde são usados. Assim, as SODA têm como objetivo [45]:

- Fornecer um alto nível de abstração do mundo real;
- Estabelecer uma relação entre o mundo físico e o digital através de um ou mais serviços conhecidos.

Apesar de representarem uma extensão da arquitetura SOA, as implementações SODA podem utilizar *standards* novos, em detrimento dos tradicionais standards SOA. (Figura 3.2). A sua implementação deverá ser direta e simples, mostrando o seu potencial ao nível da ativação de novos serviços baseados em redes, como a internet. No

entanto, apesar de poderem diferir em alguns aspetos já mencionados, as arquiteturas SODA devem ir de encontro aos princípios básicos das SOA. Estes permitem mapear processos e eventos ao longo de uma infraestrutura de comunicação aberta, partilhando dados e funcionalidades de uma forma livre e simples.

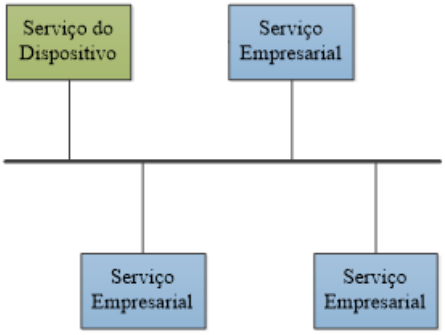


Figura 3.2 - Dispositivos modelados como serviços numa implementação SOA.

As tradicionais abordagens de integração de dispositivos centram-se, geralmente, num *software* de interface proprietário, a comunicar com aplicações comerciais através de uma variedade de *middlewares* e *API's*.

Apesar desta arquitetura se ter adequado à maioria das empresas atuais, as *SOA*, os *standards* e iniciativas de software livres estão-se a direcionar para além desta. Apesar das aplicações comerciais se estarem a adaptar à *SOA*, *standards* para a definição das interfaces dos dispositivos de baixo nível continuam a emergir. No entanto, existe tecnologia suficiente atualmente para a implementação de uma *SOA* num ambiente de eventos críticos e dados originados de dispositivos. Mecanismos para o desenvolvimento e partilha de interfaces de dispositivos, capacidades para manutenção remota de *software*, e modelos de troca de informação baseados em *loosely coupling* revelam-se como técnicas muito eficientes para a implementação de *SODA*. Desta forma as SODA contemplam alguns requisitos, tais como [46]:

- Usar um modelo de adaptação de dispositivos para englobar programação específica ao dispositivo;
- Empregar sistemas *loosely coupled* para a troca de informação dentro do servidor, capaz de suportar múltiplos serviços de partilha de informação, como os utilizados nos sistemas SOA, dos quais o *Enterprise Service Bus* faz parte da lista de exemplos;
- Utilizar *standards* livres quando possível, quer a nível do dispositivo quer ao nível da interface com os seus serviços;

- Fornecer métodos (adaptadores) de apresentação de *standards* ou serviços livres, a dispositivos com protocolos proprietários ou onde não é realista aplicar outros *standards* a interfaces de dispositivos de baixo nível;
- Suportar a implementação de uma variedade de métodos de apresentação de *standards* ou serviços livres – desde simples, sensores de baixo custo a complexos dispositivos;
- Suportar a configuração de componentes lógicos remotamente para manutenção, atualização e extensão de adaptadores de dispositivos.
- Adaptar mecanismos de segurança tal como requisito na área de *IT*.

Posto isto, uma implementação baseada em SODA contempla três constituintes básicos. Os já mencionados adaptadores para a comunicação entre interfaces, protocolos e conexões de um lado e providenciarem um modelo de abstração de serviços de um dispositivo para outro. O *"bus adapter"* que serve como canal de comunicação entre dispositivos entrega os dados através de diversos protocolos de rede, mapeando o modelo de abstração especifico ao dispositivo ao mecanismo *SOA* utilizado no sistema. Por fim, o registo de serviços dos dispositivos é necessário para a descoberta e acesso dos serviços fornecidos pelas *SODA*.

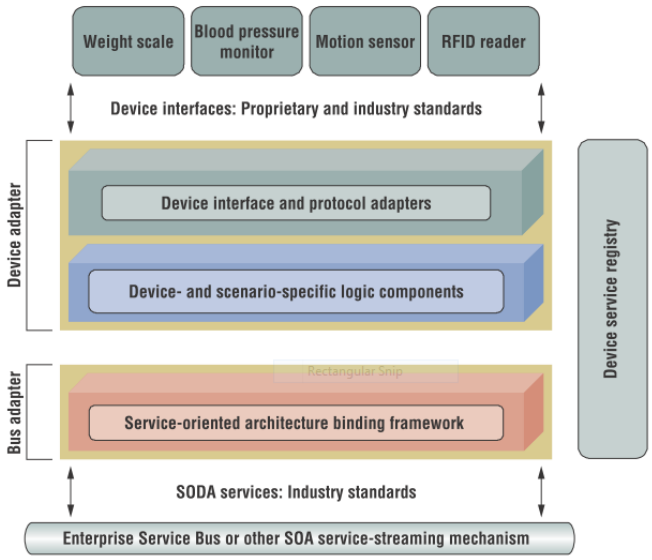


Figura 3.3 - Implementação das camadas SODA [46].

Quando não está presente um *standard* ao nível da interface do dispositivo, o adaptador presente nas implementações baseadas em SODA, faculta um modelo comum ao *software* utilizado para a criação de interfaces para os seus serviços.

3.2.3. Peer-to-Peer Architecture (P2P)

A arquitetura *Peer-to-Peer* (P2P) tendo visto a sua popularidade a aumentar nos últimos anos [47]. O seu aparecimento coincidiu com o de três sistemas importantes, em 1999: (1) *Napster* - Sistema de partilha de músicas; (2) *Freenet* – Armazenamento de dados de forma anónima; (3) *SETI@home* – Computação de projetos científicos através da entreajuda de voluntários. Apesar deste tipo de computação ter ficado intrinsecamente associado à partilha de aplicações entre utilizadores, atualmente o cenário é bem diferente. Não pondo de parte esta utilização, é agora possível utilizá-la para a distribuição de dados, *software*, estando presente em diversas aplicações comerciais (caso do *Skype*).

Estes sistemas possuem diversas características, sendo bastante conhecidos devido à sua robustez, performance e extensibilidade [47]. Esta arquitetura utiliza comunicações diversas entre os participantes numa rede e o *bandwith* total da soma de todos os participantes, em vez dos tradicionais recursos centralizados, baseados num único servidor (que representa a base do conhecimento). Na Figura 3.4 e Figura 3.5, encontra-se ilustrada a diferença entre os sistemas tradicionais de partilha de informação e a configuração de um sistema P2P. Num sistema tradicional, os clientes utilizam a informação fornecida por um servidor central, onde estão ligados todos os outros clientes. Por outro lado, utilizando a arquitetura P2P, a troca de informação ocorre entre *peers*, dispensando a utilização de um recurso centralizado, o servidor.

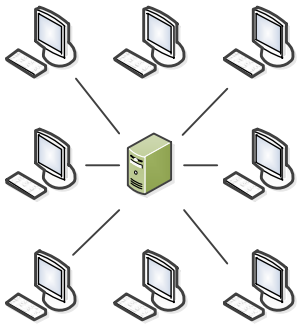


Figura 3.4 – Sistema Tradicional.

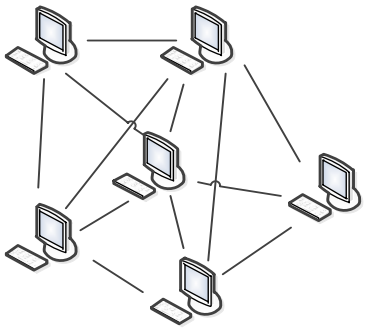


Figura 3.5 - Sistema P2P.

Desta forma são é o habitual estrangulamento dos canais de comunicação, fenómeno este denominado de *bottlenecks*. Devem-se essencialmente ao número excessivo de pedidos ao servidor, não sendo este capaz de responder em tempo útil. O modelo tradicional de comunicação possui ainda uma grande desvantagem face a este tipo de arquitetura: possui apenas um único ponto de falha, comprometendo todo o processo em caso de erro. Assim, um sistema de P2P puro consiste num número de

nodos, denominados de *peernodes*, igual aos necessários para cumprir o processo de partilha de informação, quer para os clientes quer para os servidores. No entanto, esta configuração só é eficiente se todos os nodos tiverem a mesma capacidade (poder de processamento, mesma comunicação, etc.). Apesar das capacidades já mencionadas, esta técnica não poderá constituir a arquitetura fundamental onde assentaria uma solução de AAL. Em sistemas heterogéneos como os AAL poder-se-ia aplicar esta técnica a subsistemas da solução, onde os dispositivos afetados pertenceriam à mesma classe computacional. Além destes fatores, as vantagens inerentes a estes sistemas representam também algumas das suas falhas. Não havendo dependência dedicada de uma estrutura onde o controlo é centralizado, estes sistemas encontram-se expostos a alguns problemas não existentes no sistema tradicional: são fáceis alvos de ataques, não há controlo de participantes e conteúdo é pouco ou nada filtrado.

3.2.4. Event-driven architecture (EDA)

Esta metodologia segue um padrão de publicação-subscrição. Os sistemas EDA promovem a criação e reação a eventos, em que um evento representa uma mudança significativa em algum componente ou estado [48]. Construir aplicações ou sistemas em torno de arquiteturas baseadas em eventos permite que estas respondam rapidamente a situações não programadas. Tal deve-se ao tipo de arquitetura que representam que, por *design*, são mais normalizados para ambientes imprevisíveis e assíncronos. Este tipo de configuração permite uma fácil idealização e conceptualização de comunicações um para muitos (n:m) em que os produtores de eventos não sabem do número de consumidores. Esta é uma grande diferença no processo face aos sistemas *SOA* [49] já discutidos anteriormente. Esta possibilidade contribui efetivamente para a extensibilidade e modificabilidade dos sistemas e pode representar uma boa arquitetura para soluções de AAL. Na Figura 3.6, encontra-se uma vista geral deste tipo de arquitetura. Cada evento ocorre de forma assíncrona, despoletando uma resposta à necessidade de uma notificação, executando um processo programado no bloco ativado. De notar que o evento encontram-se representado na figura pela numeração presente. Desta forma, a resposta a uma condição especifica (por exemplo, a alteração de um valor de um sensor) forma um evento que ativará uma resposta. Assim que as fontes de notificações publicam um pedido, os recetores de eventos podem escolher entre ouvir ou filtrar eventos específicos e, tomar decisões proactivas e em tempo real sobre como reagir a estes pedidos. Tomando como exemplo um ambiente de domótica, a passagem de uma pessoa por um sensor de presença resultará (se assim for suposto), na luz da

divisão se ligar. Por outro lado, acender as luzes quando o utilizador acorda já resulta da participação de outros sistemas. O sistema que identifica o estado do utilizador notifica que este está acordado, publicando a notificação de que é necessário acender as luzes.

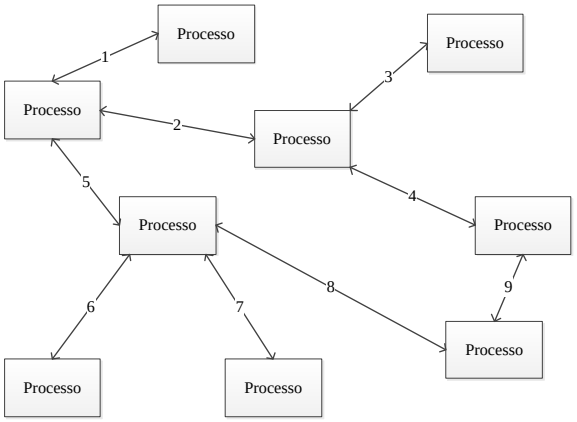


Figura 3.6 - *Event-Driven Architecture*.

Desta forma, o procedimento não é síncrono mas sim resultante da reação a pequenos pedidos (notificações) por parte dos recetores.

Quando comparadas com as arquiteturas orientadas a serviços, de um ponto de vista tecnológico, as EDA podem ser implementadas por cima de uma infraestrutura orientada a serviços, sem aumentar a sua complexidade e o número de dependências. Uma junção destes dois estilos arquitetónicos proporciona um sistema bastante consistente e estabelece uma base sólida para ambientes sempre-ligados e sempre-conectados.

Visto que as EDA requerem uma reformulação dos processos e das regras básicas dos sistemas já existentes, o beneficio desta abordagem tem de ser bastante recompensador para ser utilizado. No entanto, tal reformulação é maioritariamente valoroso e merecedor de utilização, tal como justificado em [50].

Devido aos valores provados desta arquitetura, mencionados em cima, é espectável que esta se funda com as arquiteturas orientadas a serviços (tal como já referido) em constituam a melhor abordagem para as soluções de AAL.

3.2.5. Multi Agent System (MAS)

Os sistemas baseando em múltiplos agentes representam um estilo promissor para a realização de sistemas distribuídos adaptativos. Estes são compostos por vários agentes de *software* – algumas vezes autónomos – coletivamente capazes de atingirem metas que individualmente seriam complexas de cumprir por sistemas monolíticos ou com um único agente [51]. Os *MAS* podem cumprir tarefas complexas e adaptáveis

mesmo quando as estratégias individuais de cada agente é bastante simples. Em [52] retrata-se a utilização deste tipo de sistemas em prol de soluções de AAL. É construída uma equipa de agentes que cooperam na tradução da informação, inserida num contexto, de modo a auxiliar o utilizador necessitado. Na Figura 3.7, é retratado o sistema descrito. Apesar de representar uma situação particular (específica ao estudo), serve de base de comparação e visualização do funcionamento dos sistemas *MAS*.

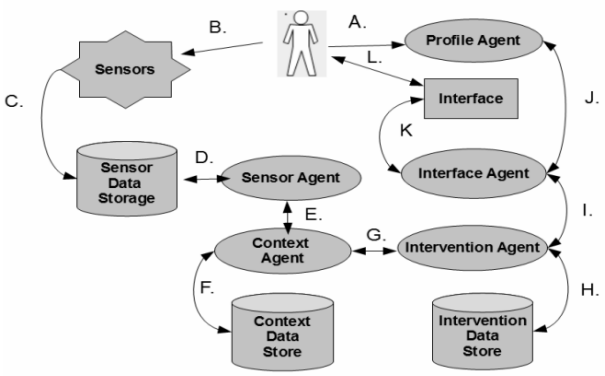


Figura 3.7 - Sistemas Multi-agente.

Desta forma, o sistema é constituído pelo agente dos sensores, agente do contexto, agente de intervenção, agente de interface e o agente de perfis. Cada agente é responsável por responder a tarefas básicas para as quais foram programados. Assim, quando utilizados em conjunto, fornecem uma arquitetura robusta e capaz de grande poder de ação.

3.2.6. Blackboard systems

Os sistemas de *blackboard* não são recentes. O primeiro sistema deste tipo foi o Hearsay-II, uma máquina de reconhecimento e análise de voz [53] foi desenvolvido à cerca de 20 anos atrás. No entanto, algumas características básicas do Hearsay-II ainda se encontrar nos sistemas atuais de *blackboard*. Numerosos avanços e melhoramentos foram realizados através da aplicação destes sistemas numa vasta área aplicacional.

Ao contrário da grande parte das técnicas de resolução de problemas para inteligência artificial que implementam modelos formais, a metodologia deste sistema foi desenhada de forma a lidar com problemas que não esteja empiricamente estabelecidos. Encontram-se ainda independentes de requisitos formais, proporcionando assim a desenvolvedores e investigadores a flexibilidade de inventar e aplicar técnicas avançadas a este tipo de sistemas. Por outro lado, a falta de uma especificação formal

contribui também para a confusão entre os sistemas de *blackboard* e qual o campo que devem atuar no que diz respeito aos problemas de inteligência artificial [54].

Este tipo de sistemas consiste essencialmente em três grandes componentes [55]: (i) os módulos específicos, que formam as fontes de conhecimento, (ii) a *blackboard* que é um repositório de problemas, soluções parciais e sugestões usadas para a troca de informação entre módulos específicos e (iii) um módulo de controlo, que controla o fluxo da atividade de resolução de problemas no sistema. Na Figura 3.8, encontra-se representada uma vista geral do funcionamento dos sistemas de *blackboard*.

Cada fonte de conhecimento é independente das restantes, não necessitando de conhecimento especializado ou sequer da existência de outras fontes. No entanto necessita de compreender o processo de resolução de problemas e qual a representação de informação relevante a apresentar no *blackboard*. Necessita ainda de saber em que condições é que pode contribuir para a solução final e, quando considerar apropriado, tenta contribuir com informação relevante para a resolução do problema. Esta condição de "consciência" é conhecida como condição de *"trigger"*. Comparados com os sistemas *"expert"*, os sistemas *blackboard* são muito mais abrangentes que as regras individuais utilizadas pelo primeiro. Enquanto os sistemas *"expert"* funcionam através da ativação de uma regra em resposta a um estímulo, os sistemas *blackboard* funcionam através da ativação de um módulo de conhecimento completo, uma rotina, ou um procedimento.

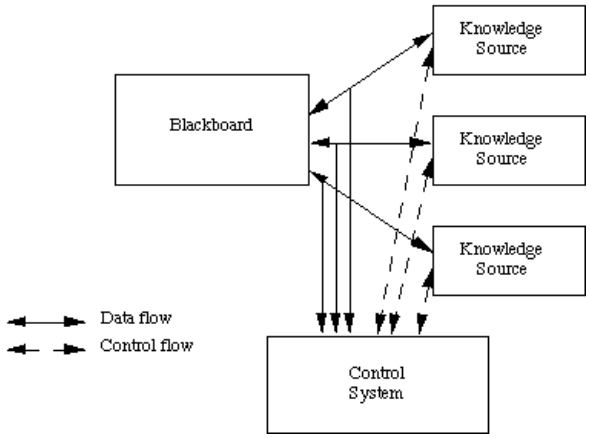


Figura 3.8 - Vista geral de sistemas *blackboard*.

No entanto, as fontes de conhecimento não são agentes ativos nos sistemas de *blackboard*. As entidades responsáveis pela ativação destas fontes de conhecimento (também chamadas de instancias das fontes de conhecimento) são entidades sempre ativas que competem pelos recursos de execução. É o resultado da combinação do conhecimento presente nelas e um contexto que as ativou. Assim, a distinção entre

fontes de conhecimento e a ativação da mesma é importante em aplicações em que numerosos eventos ativam a mesma fonte de conhecimento. Nestes casos, as decisões de controlo envolvem a escolha entre aplicações particulares dentro da mesma fonte de conhecimento (focando no contexto a aplicar), em vez de procurar dentro de fontes diferentes (focando no conhecimento a aplicar). Desta forma, as fontes representam repositórios estáticos de conhecimento enquanto as suas ativações representam processos ativos.

O *blackboard* é uma estrutura global disponível para todas as fontes de conhecimentos e serve como:

- Um registo global de dados de entrada, soluções parciais, alternativas e as soluções finais e as informações de controlo;
- Um meio de comunicação;
- Um mecanismo de ativação das fontes de conhecimento.

Aplicações baseadas em *blackboard* tendem a ter estruturas elaboradas, com múltiplos níveis de análise ou abstração. Ocasionalmente, um sistema contendo subsistemas que comunicam utilizando uma base de dados global é incorretamente apresentado como sendo um sistema de *blackboard*. No entanto, os verdadeiros sistemas deste tipo envolvem uma interação próxima com as fontes de conhecimento e um mecanismo de controlo separado.

O componente de controlo é um mecanismo que direciona o processo de resolução de problemas, permitindo que as fontes de conhecimento respondam oportunamente às alterações na base de dados do *blackboard*. Na base do estado do *blackboard* e no *set* de bases de conhecimento ativados, o mecanismo de controlo é quem decide o fluxo da ação (do processo de resolução de problemas).

Um sistema de blackboard utiliza um sistema de resolução de problemas incremental: a solução ao problema é construída um passo de cada vez. Em cada passo, o sistema pode:

- Executar qualquer fonte de conhecimento ativada;
- Escolher uma fonte de atenção diferente, na base do estado da solução;

Numa abordagem de controlo típico, a fonte de conhecimento ativada em execução gera eventos enquanto contribuir para o *blackboard*. Estes eventos são mantidos até que a ativação dessa fonte termine. Nesse ponto, os componentes de controlo usam eventos para ativar outras fontes. Estas são classificadas segundo a contribuição mais apropriada que cada uma poderá dar e a mais adequada é selecionada para execução. Este ciclo repete-se até o problema se encontrar resolvido. Os sistemas

de *blackboard* suportam vários tipos de mecanismos e algoritmos de controlo proporcionando assim uma escolha mais vasta para o desenvolvedor da aplicação.

O modelo *blackboard* oferece uma arquitetura de resolução de problemas muito forte e que é adequado nas seguintes situações:

- Vasta diversidade e é necessária a representação de conhecimento especializado. As fontes de conhecimento podem ser desenvolvidas sob a representação mais apropriada para a informação que vão utilizar.
- Um *framework* de integração é necessário para permitir a resolução de problemas heterogénea.
- O desenvolvimento de uma aplicação envolve muitos desenvolvedores. A modularidade e independência fornecida pelas fontes de conhecimentos nos sistemas de *blackboard* permite que estas últimas sejam desenvolvidas e testadas individualmente. Estes benefícios são visíveis na fase de *design*, implementação, teste e manutenção da aplicação.
- Conhecimento limitado ou pouco explicito inibe a determinação absoluta de uma solução. A abordagem incremental destes sistemas permite que seja possível algum progresso.

Os sistemas de *blackboard* têm sido aplicados em inúmeras áreas, como interpretação sensorial, visão por computador, aprendizagem simbólica, entre outras. Em todas estas aplicações o âmbito do problema foi o fator primordial à seleção da escolha deste tipo de abordagem. Assim, a decisão de quando usar um sistema de *blackboard* deverá ser baseado na resolução de problemas apresentado, em vez da área específica de aplicação.

3.2.7. Conclusão

Ultimada a apresentação de alguns estilos arquiteturais mais usados para sistemas de *AAL*, é impossível afirmar que existe um que se adequa perfeitamente ao domínio do mesmo. É expectável que ao longo do tempo alguns estilos se fundam ou convirjam (*SOA* e *EDA*, p.e.) de maneira a colmatar diferentes requisitos (mencionados em 2.4). Existem alguns projetos [56][57] onde esta união já ocorre e os resultados têm correspondido às espectativas. No projeto *BelAmI* [58][59] está a ser investigado um estilo hibrido entre *SOA* e *EDA* que seja sensível ao contexto dos serviços e providenciar um suporte dedicado para adaptabilidade (caso da auto-adaptação).

3.3. Modelação de dados

As soluções de domótica atuais contam com inúmeros parâmetros a considerar a cada segundo. As crescentes funcionalidades acrescentadas a este tipo de sistemas implicam obrigatoriamente a organização dos dados de forma rápida e acessível. Assim, surge a necessidade de modelar os dados para que possam ser consultados e alterados como resultado da resposta a um procedimento. Esta modelação é especialmente importante pois é baseado nela que as arquiteturas de *software* baseiam as suas respostas. Assim, algumas técnicas de modelação de dados têm surgido nas últimas décadas de forma a colmatar esta necessidade. O maior avanço neste campo a nível tecnológico para os sistemas de AAL além do *middleware* é a *Semantic Web*. A autoavaliação do sistema e avaliação do utilizador nos sistemas de AAL requerem algum conhecimento conceptual sobre o sistema e o ambiente. Estes modelos necessitam de ser representados pelo menos de uma forma semiformal, para que possibilitem um sistema de processamento inteligente e a *Semantic Web* tem vindo a evoluir neste sentido.

A *Semantic Web* é uma extensão da *World Wide Web* [60] em que o conteúdo da internet poderá ser expresso não só numa representação atual mas também num formato que poderá ser facilmente lido e interpretado por processos computorizados. Isto permite que a pesquisa, partilha e integração da informação seja facilitada e muito mais eficiente (estando melhor catalogada, o seu acesso é mais rápido). Algumas das mais notáveis tecnologias atuais é o *Resource Description Framework* (*RDF*), uma variedade de formatos de dados e a *Web Ontology Language* (*OWL*) [61][62]. Pretendem prover uma descrição formal a nível de conceitos, termos e relações entre o conhecimento sobre um domínio.

O *RDF* é representa uma família do *World Wide Web Consortium* sendo que suas as especificações são originalmente desenhadas segundo um metamodelo mas são utilizadas como um método de modelar informação, através de uma variedade de formatos de sintaxe [63]**.** O metamodelo *RDF* é baseado na ideia de construir frases sobre recursos, sob a forma de sujeito-predicado-objeto, denominados "triplos" na terminologia *RDF*. Neste modelo, o sujeito denota o recurso e o predicado os aspetos do recurso e expressa ainda a relação entre o sujeito e o objeto [60].

Por outro lado, o *OWL* constitui uma recomendação formal do *W3C* para a representação de ontologias [64][65]. Esta é baseada no *RDF*, tendo sido adicionado mais vocabulário para descrever propriedades e classes: entre entidades, relações entre

classes, cardinalidade, igualdade, propriedades mais refinadas, características das propriedades e classes enumeradas. O *standard OWL* define três sub-linguagens, que diferem no seu poder de representação do conhecimento[66][67]. A sub-linguagem mais completa do *OWL* não restringe o uso de recursos *OWL*. As outras sub-linguagens contém restrições de modo a tornar o processamento da informação mais eficiente (a restrição de algumas aferições torna o processamento mais rápido). À parte destas linguagens existem ainda um conjunto de ferramentas (p.e, *Protégé*) para modelar as ontologias e vários *reasoners* (p.e *Pellet*). Nos seguintes subcapítulos serão aprofundados os temas mencionados, sendo no fim comparados com as abordagens mais tradicionais. De notar que não existirá uma secção sobre RDF devido à sua semelhança com o formato OWL.

3.3.1. Ontologias

As ontologias tem merecido enorme discussão na comunidade da inteligência artificial (IA), Estas, tem uma longa história na filosofia, onde se refere à existência do ser. Por outro lado, é usualmente confundida com epistemologia que também trata do ser e do conhecimento. Assim, a definição citada tem sido amplamente aceite na comunidade científica, referindo-se à partilha de conhecimento. Esta definição é consistente com a utilização da palavra como um conjunto de definições de um conceito/domínio tendo sido apenas generalizada. Implica que a ontologia é uma descrição de conceitos e relações que podem existir/ocorrer dentro de um domínio. Na prática, uma ontologia é um acordo para utilização de um vocabulário específico (fazer perguntas ou asserções) de uma forma concisa (não sendo completa) sobre um domínio por si especificado.

Quando um domínio de conhecimento é representado segundo um formato declarativo, o conjunto de objetos que podem ser representados chama-se "universo de discurso". É este conjunto e as suas relações internas, que refletem no vocabulário em que os programas baseados em conhecimento o representam internamente. Assim, definições associam os nomes das entidades no universo de discurso (por exemplo, classes, relações, ou outros) como texto lido por um humano. Este texto, descreve o significado dos nomes e uma interpretação formal dos axiomas.

Usualmente, as ontologias são utilizadas para descrever relações formais entre um conjunto de agentes, fornecendo caminho à comunicação em torno de um domínio de discurso, sem requerer uma operação a nível global. Um agente é aceite numa ontologia se é observável que as suas ações são consistentes com as definições da

ontologia. A ideia deste "*commitment*" baseia-se no *"Knowledge-Level perspective"* [68]. Este nível de descrição do conhecimento de um agente é independente da representação do nível de símbolos, usada internamente pelo agente. Um agente "sabe" algo se se comporta como se tivesse informação e se age conforme os seus objetivos. As suas ações, incluindo os servidores da base do conhecimento e sistemas baseados em conhecimento, podem ser vistas como uma interface pergunta/resposta. Este modelo foi proposto por *Levesque* em 1984 [69]. Nele, um cliente interage com um agente através de "*queries*" (perguntas) e asserções lógicas (respostas). Pragmaticamente, uma ontologia define o vocabulário entre o qual as perguntas e respostas são válidas entre os agentes. Esta situação pode ser comparada com a definição de requisitos de um sistema. Os agentes que partilham o mesmo vocabulário não necessitam implicitamente de partilhar a mesma base de conhecimento. Cada um tem conhecimento de fatores que o outro não tem e, um agente que se faça parte de uma ontologia, não necessita obrigatoriamente de saber as respostas a todas as perguntas num vocabulário partilhado. A Figura 3.9 ilustra os conceitos apresentados anteriormente de uma forma geral.

A construção de uma ontologia não se restringe a uma solução única e de simples modelação, podendo haver várias maneiras de representar o mesmo domínio. À semelhança dos tradicionais sistemas, têm de ser tomado em conta um processo iterativo, começando por um esboço geral da ontologia, refinando e evoluindo a mesma, aumentando os detalhes sobre os dados. Tal como referido em [70] é necessário seguir um conjunto de regras predefinidas mas que auxiliam o processo de tomada de decisões.

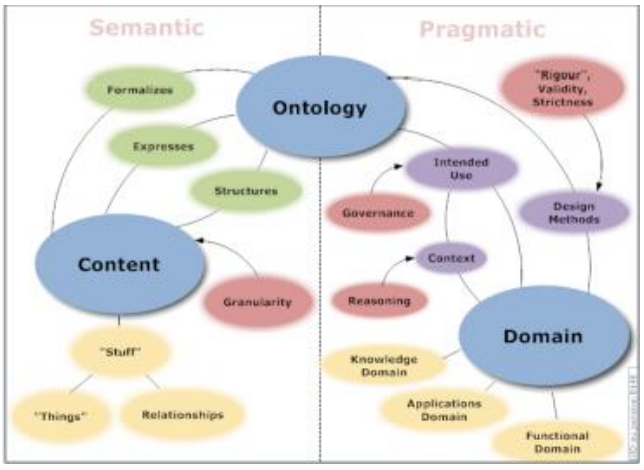


Figura 3.9 - Mapa de dimensão da Ontologia em [70].

Estas são:

- Não existe uma maneira única e correta de modelar um domínio – Existem sempre alternativas viáveis. A melhor solução depende quase sempre da aplicação que se tem em mente e as extensões que suportará.

- O desenvolvimento de ontologias é, obrigatoriamente, um processo iterativo.
- Conceitos numa ontologia devem estar o mais próximo possível dos objetos representam e das suas relações dentro do seu domínio. Estes representam maior parte das vezes nomes(objetos) ou verbos (relações) em frases que representam o domínio onde atuam.

Estas *guidelines* são especialmente úteis para orientar o processo de desenvolvimento de uma ontologia. Além das possíveis alternativas para a construção da solução, é necessário decidir qual a mais viável, mais intuitiva e extensível. É também necessário relembrar que uma ontologia representa um mundo real e que os seus conceitos têm de refletir obrigatoriamente essa realidade. Uma representação ontológica segue-se por vários princípios que auxiliam à construção do modelo. Ainda em [70] é proposto um guião de ajuda à construção do modelo ontológico. Assim, devem-se seguir as seguintes etapas:

1. Determinar o domínio e contexto da ontologia – Nesta etapa são respondidas perguntas sobre o conteúdo da ontologia, para que serve, quem a irá usar e quais as perguntas às quais se querem respostas. Apesar destas poderem ver a sua resposta alterada ao longo do processo de desenvolvimento da ontologia, certamente seguirão de ajuda em qualquer etapa. Uma das formas de identificar os limites/contexto da ontologia é formular questões denominadas de "*Competency Questions*". Estas serão abordadas no fim deste capítulo e baseadas no Anexo 1.
2. Considerar a reutilização de ontologias já existentes – Usualmente, é proveitoso considerar o que alguém já fez e verificar se se pode refinar e estender ontologias já existentes para o domínio que se está a modelar. Esta fase é obrigatória quando o sistema a modelar necessita de interagir com aplicações já associadas a outras ontologias ou possuem um vocabulário específico e controlado. Atualmente, já se encontram disponíveis inúmeras ontologias em formato eletrónico mas poucas vocacionadas para os sistemas de AAL. A maneira como estas estão expressas não é relevante, pois o processo de adaptação é simples e grande parte das vezes, automático. Existem inúmeras bibliotecas que disponibilizam ontologias como é o caso da Ontolingua [71], DALM [72], RosettaNet [73] ou DMOZ [74].
3. Enumerar termos importantes à ontologia a modelar – Este tópico diz respeito a um processo de *brainstorming* formal sendo necessário criar uma lista de conceitos a contemplar na ontologia, necessária à construção de frases ou à

interação com o utilizador. Este processo responde ainda a perguntas relativas ao que ao domínio com que estamos a lidar, o que é necessário dizer sobre os seus termos e quais as suas propriedades. Inicialmente, é importante reter o maior número possível de noções não sendo requisito preocupar-se com termos que possam vir a ser duplicados, sendo a relação entre eles, as suas propriedades ou classificação é relegada para segundo plano.

4. Definir as classes e suas hierarquias – A elaboração desta etapa representa a primeira grande decisão a tomar, existindo várias metodologias para cumprir esta meta. O desenvolvimento de uma hierarquia de classes [75] pode seguir uma aproximação "top-down", "bottom-up" ou uma combinação dos dois.
    a. Top-down - requer que o processo se inicie com a definição dos conceitos mais gerais do domínio e consequente especificação;
    b. Bottom-up - Na abordagem "bottom-up" o processo é inverso. Inicia-se pela discriminação das classes mais específicas, agrupando-as depois em classes mais abrangentes;
    c. Combined - Por fim, na abordagem combinada, definem-se os conceitos mais importantes primeiro, generalizando-os convenientemente. Nenhum dos três conceitos é considerado certo ou indicado, ficando ao critério do desenvolvedor e da sua experiência.
5. Definir as propriedades das classes – As classes per si não possuem informação suficiente para responder às "*competency questions*" mencionadas na etapa 1. Uma vez definidas algumas das classes é necessário descrever/representar a sua estrutura interna. Desta forma, maior parte dos termos em falta serão propriedades das classes sendo que para cada propriedade da lista, é necessário determinar qual classe esta descreve.
6. Definir as características dos *slots* – Os *slots* podem conter diversos aspetos que contemplam o seu tipo de valores, gama de valores aceitável, número de valores (cardinalidade) entre outras propriedades. Por exemplo, o valor de um *slot* "nome" é uma *string*. Assim, "nome" é um *slot* com o tipo de valor *string*. Existem diversos *facets* usualmente utilizados. Estes são:
    a. Cardinalidade do *slot* – define quantos valores o slot pode ter. Esta propriedade é distinguida por alguns sistemas entre cardinalidade simples (só aceita um valor) e múltipla cardinalidade (aceita vários valores). Existem também sistemas que permite balizar a cardinalidade, descrevendo assim o valor dos *slots* de forma mais precisa. Por vezes,

põe-se este valor a '0' de modo a indicar que o *slot* não pode aceitar valores de uma determinada subclasse;

b. Tipo do valor do *slot* – um "value-type facet" define que tipo de valores podem preencher o slot. Os mais usuais são:
    i. *String* – é o tipo mais simples e é utilizado, por exemplo, para *slots* que descrevam o nome;
    ii. Número – Descreve o *slot* com um valor numérico e pode ser especificado utilizando *float* ou *integer*;
    iii. Booleano – Utilizados como *slots* sim/não. Normalmente utilizado para identificar a presença de alguma característica;
    iv. Enumeração – Especifica uma lista de atributos que o *slot* pode tomar;
    v. Instância – Permite a definição e relações entre indivíduos. *Slots* deste tipo devem definir também uma lista de classes cuja instanciação é possível.

7. Instanciação – Esta última etapa diz respeito à criação de instanciações individuais de classes. Defini-las requer (1) escolher uma classe, (2) criar uma instância individual da mesma e (3) preencher os valores dos seus *slots*. É o formato final desta etapa que será utilizado pelos sistemas para se relacionarem com a base de conhecimento.

Referidas na etapa 1, as "*competency questions*", são fundamentais à construção de uma ontologia, servindo não só para determinar qual o contexto desta mas também, posteriormente, a validar. Um conjunto de perguntas bem construídas permite o desenvolvedor identificar quais os pontos fracos e fortes da ontologia aquando da validação sendo que, numa fase inicial a permite refinar e corrigir. A colocação destas perguntas guiará o utilizador num percurso pré-definido pela ontologia, em torno das suas classes sendo que uma resposta de acordo com o estipulado pelo desenvolvedor representa uma validação de parte da ontologia. Desta forma, o número e a diversidade de *"competency questions"* é bastante útil antes, durante e depois da construção da ontologia.

Tal como já referido, a construção de uma ontologia implica um procedimento obrigatório de regras apresentadas em [70]. Para a criação destas é necessário recorrer a ferramentas especializadas para modelação a tão alto nível. Existem inúmeras alternativas tais como (1) *Protégé* [76], (2) *NeOn Toolkit* [77], (3) *Vitro* [78], (4) *IBM Integrated Ontology Development Kit* [79], entre outras. Ao longo do desenvolvimento

do presente trabalho foi utilizada a ferramenta (1) devido à sua excelente documentação, facilidade de utilização e suporte prestado pela entidade desenvolvedora. No entanto, a representação final (sob a forma de um *ficheiro XML – Extensive Markup Language + RDF – Resource Description Framework*) de algumas instâncias encontra-se incompleta devido a alguns problemas na última versão da ferramenta *Protégé*. O modelo RDF é a base da linguagem OWL e é através dele que se estabelecem as relações entre objetos.

Após a modelação da ontologia, esta pode ser acedida com ajuda de um *reasoner*. O *reasoner* é um código de *software* que permite inferir relações lógicas dos axiomas e asserções presentes na ontologia. Não havendo um único *reasoner* apropriado para todas as ontologias existem diversas alternativas, sendo que os mais usados são o *Pellet* [80] e *FaCT++*[81]. Tal prende-se com ao facto de serem disponibilizados de forma livre e providenciarem boas características tal como referenciado em [82].

A integração de ontologias em sistemas finais implica a sua conversão para formatos mais proliferados e de fácil utilização. Assim, são utilizados conversores para *JAVA* recorrendo a *plugins* que podem ser adicionados às ferramentas de modelação ontológica. Existe um vasto leque de conversores sendo os mais utilizados o *Jena* [83] e *OWL API* [84].

3.3.2. Ontologias vs Modelos tradicionais baseados em bases de dados

Atualmente existem dois grandes modelos para a representação de dados e conhecimento acerca de um domínio em sistemas computorizados. Modelos baseados em bases de dados, especialmente os de bases de dados relacionais, têm sido os líderes das últimas décadas, possibilitando aos dados uma forma eficiente de ser armazenada e questionada. Por outro lado, as ontologias apareceram como uma alternativa a este modelo, servindo melhor a aplicações que necessitam de um ambiente mais rico semântica e descritivamente. No entanto, existe ainda bastante controvérsia sobre qual a técnica de modelação de informação é melhor, dado que ambos os métodos apresentam várias características similares.

Vários modelos de representação de informação legível por computadores têm vindo a emergir desde que os modelos baseados em ficheiros desapareceram e os modelos baseados em bases de dados aumentaram a sua popularidade devido à sua eficiência, flexibilidade e performance a representar e gerir dados. Muitos esforços foram tomados para que se alterasse esta hegemonia desta técnica de modelação através de diversos modelos que complementam as suas falhas (discutidas posteriormente). Alguns destes exemplos são as técnicas orientadas aos objetos, modelos espaciais,

dedutivos e multimédia. Simultaneamente com estes esforços, técnicas de modelação baseadas em conhecimentos sobre um domínio propuseram modelos alternativos [85] para definir a informação de forma mais expressiva mas também em complexidade [86][87]. Como consequência natural estas representações aplicadas maioritariamente em IA, tais como a lógica descritiva (*Description logics* – DL) ou a lógica de primeira ordem (*First Order Logic* – FOL) tornaram-se incrementalmente menos populares que as bases de dados relacionais, especialmente em aplicações comerciais. No entanto, o único modelo merecedor de destaque na última década é o ontológico. Esta representação, compete em âmbitos diferentes, com o crescimento e popularidade do esquema de base de dados devido ao surgimento de tecnologias baseadas em ontologias em desenvolvimento, essencialmente no que diz respeito à *Semantic Web*. Existe atualmente bastante discussão no mundo académico sobre qual a técnica de modelação mais adequada para os seus sistemas e se será compensatório o esforço inerente à sua integração. Esta situação tem levado a inúmeras discussões sobre a abordagem mais correta de representar informação e conhecimento em computadores. Consequentemente, as duas técnicas têm sido comparadas como forma de estabelecer um grau de similaridade entra as duas, bem como os seus prós e contras. No entanto, esta discussão toma diversos aspetos em conta, tal como a assuntos de eficiência, capacidades de representação, tecnologias envolvidas e repercussões para os sistemas atuais.

Tendo já sido discutida a influência e importância das ontologias na secção anterior, nesta apenas será abordada a relevância dos modelos baseados em bases de dados e como se comparam estas com as primeiras.

Existe uma literatura abundante sobre bases de dados desde que estas substituíram modelos baseados em ficheiros há cerca de quarenta anos atrás. O modelo relacional proposto em [88] causou uma revolução na gestão de grandes repositórios de dados e novas técnicas para representação de dados surgiram, tais como bases de dados orientadas a objetos [89], bases de dados multimédia para o armazenamento de ficheiros multimédia [90], bases de dados XML para a interação com a Web [91], bases de dados dedutivas ou lógicas para fazer deduções utilizando a informação lá contida [92][93], etc. Ainda, outros modelos representativos de conhecimento, tais como [87] ou [93], tentaram realçar o poder representacional das bases de dados usando *DL* ou *FOL*. Estas propostas complementaram as falhas existentes nas bases de dados relacionais, especialmente na representação semântica.

Apesar da área de modelação de conhecimentos ter vindo a definir inúmeros modelos representacionais da informação, as ontologias têm tomado especial destaque nas últimas décadas. As ontologias envolveram uma revolução na ciência da computação, especialmente na área de inteligência artificial e disciplinas de bases de dados. Por um lado, desde o início, as bases de dados representam informação sobre o mundo real mas impõe restrições (a nível do formato dos dados, não haver variáveis null, etc.) ao seu acesso, de modo a obter uma organização mais eficiente. Por outro lado, as técnicas ontológicas têm de lidar com os mesmos tipos de problemas que as bases de dados tiveram de resolver anos atrás [94]. Problemas como a heterogeneidade da informação, mapeamento de pesquisas, alinhamento do modelo, resolução de conflitos, etc., presentes no processo de representação ontológica apareceram similarmente à técnica de bases de dados no passado [90-93]. A comunidade científica considera que as ontologias são o melhor método para representar a realidade devido à sua capacidade de modelação semântica de conceitos. As ontologias utilizam classes, propriedades, instâncias, relações e em especial axiomas representados maioritariamente por linguagens lógicas tais como o *DL* ou *FOL* para adicionar semântica aos seus modelos. É bastante fácil encontrar relações diretas entre uma ontologia e uma base de dados: uma classe pode corresponder a uma tabela relacional, uma propriedade a um atributo relacional, relações generalistas ou regras com os axiomas, etc. Por outro lado, esta comparação não e tão trivial como possa parecer. Desta forma é necessário analisar as suas semelhanças e diferenças baseando as suas conclusões em:

- Conceptualização da informação;
- Representação da informação;
- Modelação da informação;
- Eficiência.

Estes pontos serão abordados nas próximas secções.

3.3.2.1.Conceptualização da informação

A comunidade ontológica descreve as bases de dados como ontologias leves devido à forma como estas representam a estrutura da informação e qual o propósito que esta serve. Esta classificação significa que o esquema baseado em ontologias, tal como algumas bases de dados, não é considerada uma verdadeira ontologia [94][96] porque lhe faltam axiomas que descrevem a realidade da sua representação. O modelo de base de dados é um mecanismo de representação concebido essencialmente para

corresponder aos requisitos de uma aplicação específica, sendo necessária a sua alteração, quando estes se alteram [97]. Em contraste, as ontologias resultam de um esforço coletivo e necessitam de ser partilhados por toda a comunidade [98]. Assim, uma base de dados resulta de um trabalho de equipa enquanto a ontologia requer coordenação entre diversos grupos de trabalho de diversas áreas [99]. Resumindo, as ontologias são consideradas independentes da implementação e como tal operam a um nível muito superior de abstração. Os modelos informacionais, em oposição, operam a um nível de abstração mais baixo [100].

Referindo-se à definição de ontologia, diversos autores [100] consideram que apenas são validas as cujo conhecimento e sentido é partilhado e unanimemente aceite e usado na comunidade. Assim, todas as restantes não se consideram ontologias, sendo específicas a determinada aplicação. No entanto, numerosas ontologias estão em desenvolvimento a representar a mesma realidade: atualmente, novas aplicações da *Semantic Web* têm como objetivo o desenvolvimento de motores de busca de ontologias, tal como *Watson* [101], *OntoSearch* [102] ou o *Swoogle* [103].

As ontologias não necessitam de distinguir entre tipos de dados mais complexos ou mais básicos produzindo as suas propriedades mais valor semântico que os tipos de dados existentes nas bases de dados.

As desvantagens apresentadas inerentes à utilização de bases de dados, resultam essencialmente da incapacidade de representar axiomas mas podem ser colmatadas através da utilização de bases de dados logicó-dedutivas e regras lógicas tal como proposto em [104].

3.3.2.2. Representação dos dados

A informação representada pelas ontologias junta a especificação do modelo com os dados reais, as instâncias. A informação do modelo é guardada em *tuples* no dicionário da base de dados e pode posteriormente ser gerido tal como o resto da informação do sistema [105][98]. Graças à falta de uma distinção entre dados e modelo por parte das ontologias, aquando da utilização as instâncias das classes podem ser instanciadas inúmeras vezes. Esta característica que pode representar uma desvantagem, não existe no modelo de dados relacionais (ao contrário do modelo orientado a objetos que também permite tal característica). Consequentemente, uma nova geração de linguagens ontológicas tais como *OWL*, *OWL2* (*W3C OWLWorking Group 2009*), não permitem tal característica.

No que diz respeito ao processo de definição de instâncias, uma ontologia não segue qualquer procedimento [105]. De facto, a definição de uma nova instancia não requer que nenhuma regra seja cumprida – esta é simplesmente adicionada. No entanto, aquando da utilização do *reasoner*, esta será ignorada devido à falta de conteúdo semântico que representa. Alternativamente, a representação de bases de dados necessitam de cumprir todos os requisitos definidos de forma a assegurar a integridade dos dados. Mais especificamente, um dado não pode ser incluído na base de dados se este não satisfizer todas as regras semânticas do modelo, tais como as chaves primárias, chaves estrangeiras ou regras *null*. Apesar de representar uma vantagem a nível da integridade dos dados, prejudica o processamento semântico tal como provado em [98]. Por outro lado, as ontologias utilizam os *reasoners* para resolver tais problemas. Estes determinam quais as instâncias que pertencem à ontologia de acordo com a sua capacidade para cumprir os seus requisitos. Como resultado, cada vez que é necessário efetuar um teste à instância, o *reasoner* é chamado e executado, ao contrário dos sistemas de base de dados cuja consistência se encontra sempre assegurada. Evidentemente, os *reasoners* ontológicos são utlizados para extrair nova informação das ontologias. De forma similar, linguagens de *query* tais como o SQL são utilizadas para executar perguntas à das bases de dados. A grande diferença destas duas tecnologias assenta na capacidade dos *reasoners* descobrirem nova informação independentemente dos dados estarem totalmente corretos e descritos. Nas bases de dados, só é possível obter informação utilizando os *tuples* armazenados no sistema sendo que a informação relativa ao esquema ou à união de várias partes destes com os *tuples* nunca são obtidos.

À parte dos *reasoners*, linguagens de *query* ontológicas são também utilizadas para pesquisar informação na ontologia tais como *SPARQL* ou *SERQL*. Estas linguagens devolvem informação de relações através de predicados e triplos, enquanto a linguagem SQL representa uma estrutura tabular da informação armazenada no dicionário da base de dados.

A utilização de ontologias certamente servirá mais as necessidades dos utilizadores, devido à sua capacidade de resposta baseado num vocabulário mais rico. Estas *queries* podem ser feitas quando a ontologia representam um esquema conceptual de acesso à informação armazenado numa base de dados. No entanto, este processo não é fácil sendo que a *query* é formulada num ambiente muito complexo, envolvendo a dimensão dos dados que é muito maior que a dimensão do conhecimento [106].

3.3.2.3.Técnica de modelação

Ao contrário das ontologias, o modelo conceptual para representar a informação em bases de dados é considerado por vários autores [107][88][102] como sendo semanticamente mais rico do que a lógica descritiva utilizada nas ontologias. Atualmente existem modelos conceptuais para ontologias baseados em *frames* também muito intuitivos. No entanto, as ontologias requerem uma expressividade a um nível muito mais elevado do que os modelos conceptuais conseguem oferecer.

As diferenças existentes nos modelos baseados em bases de dados também necessitam de ser levados em conta. As limitações do modelo de dados escolhido estabelece uma enorme perda semântica no que diz respeito à representação de informação. O mesmo problema ocorre com as ontologias: a sua semântica depende da linguagem que a representa e são ainda representadas utilizando uma linguagem específica, tal como *KIF*, *RDF*, *OWL* ou *LOOM* ou uma ferramenta de geração ontológica como o Protégé, proporcionando um conjunto de limitações e restrições semânticas. Ainda assim, as conversões entre este tipo de linguagens ou representações pode conduzir a várias perdas semânticas ao longo do processo ou até algumas incompatibilidades. Esta desvantagem não ocorre nas bases de dados, especialmente no modelo objecto-relacional, em que teoricamente se usa uma linguagem *standard*, *ANSI SQL* em todas as suas implementações (*Oracle, MySQL, PostgresQL, Access,* etc.). No entanto, é necessário reconhecer que o *OWL* também tem sido bastante utilizada devido à *Semantic Web* e já é considerado um *standard*. Por outro lado, as linguagens ontológicas representam muito maior expressividade semântica do que as linguagens de bases de dados que apenas utilizam constructos para definir ou extrair dados [100]. Assim, as linguagens ontológicas fornecem uma boa forma conceptualizar um domínio, sendo uma abordagem a ter em conta neste tipo de sistemas.

3.3.2.4. Eficiência

No contexto de modelos baseados em bases de dados, a métrica da eficiência representa o maior motivo da sua popularidade e proliferação. Consequentemente, as bases de dados também estão presentes no ambiente ontológico. Mais precisamente, as ontologias podem representar a realidade bastante bem mas esta tecnologia não é suficientemente eficiente para gerar as suas instâncias, visto que são grande parte das vezes representadas em ficheiros *OWL* ou *RDF*. Como consequência natural, quando o número de instâncias aumenta, estas necessitam de ser armazenadas num ambiente de

bases de dados e então a ontologia faculta uma interface que possibilita o acesso a estas. Da mesma forma, o acesso às bases de dados é estabelecido através do uso de ontologias e não através do esquema habitual. Assim, têm sido propostos vários modelos para a comunicação entre bases de dados e ontologias, especialmente para as bases de dados relacionais devido à sua hegemonia. Estas propostas têm sido classificadas de acordo com a forma como a comunicação entre estas duas técnicas é estabelecida.

3.4. Conclusão

Neste capítulo foram abordados os diversos aspetos que contribuem para a qualidade final dos sistemas de *Ambient Assisted Living*. É fundamental estabelecer um modelo arquitetural e de gestão de dados que vá de acordo aos requisitos do sistema. A nível arquitetural existem diversas técnicas que responderão melhor a nível de performance e eficiência mas que falham noutras métricas importantes, como o custo ou a sua adaptabilidade. Não havendo solução perfeita vocacionada para esta área, é fundamental optar por uma solução mais global e que de alguma forma cumpra o maior número de requisitos possíveis. Relativamente à modelação de dados, não existe ainda uma maneira única e efetiva de representar ambientes tão complexos como os do tema em questão. Por um lado existem as ontologias, uma técnica de modelação com recente destaque, com uma vasta capacidade descritiva. Por outro lado existem as tradicionais bases de dados com a sua elevada eficiência e provado valor ao longo dos anos. Neste campo existem ainda novas técnicas a surgir que prometem juntar estes dois campos, retirando as melhores características de cada uma. Desta forma e apesar da domótica já estar presente em algumas habitações, uma aplicação de total suporte à vida (soluções de AAL) faz parte de um futuro bastante próximo. De facto, o surgimento de novas técnicas que prometem colmatar as falhas relatadas ao longo deste capítulo, pode representar o avanço necessário para a sua total proliferação no mercado.

CAPÍTULO 4

Desenvolvimento de uma ontologia para um sistema de AAL

4.1. Introdução

Sendo um dos objetivos do presente trabalho, este capítulo apresenta a elaboração de uma ontologia para a modelação de um sistema de AAL. Apesar de ter sido efetuada uma pesquisa de ontologias que pudessem servir de base ao trabalho, nenhuma delas se adequava totalmente ao tema. A modelação da ontologia foi conseguida utiliza dando a ferramenta *Protégé*, devido ao seu vasto suporte na comunidade e fácil interface de utilização. A ontologia desenvolvida é capaz de fornecer informações importantes tais como:

- Localização de determinado dispositivo;
- Funcionalidades de cada dispositivo;
- Diferentes estados que cada dispositivo pode assumir;
- Composição geral da casa (nível arquitetónico incluído).

Apesar da ontologia criada não descrever com elevado pormenor os aspetos arquitetónicos da habitação onde o sistema está instalado, fornece informação suficiente para uma descrição básica da mesma. Contém na sua composição todas as divisões de uma habitação, os seus bens móveis e imóveis, dispositivos que providenciem alguma forma de controlo, entre outros. No futuro, poderão ser adicionados outros bens ou divisões que possam estar em falta, apenas sendo necessário adicionar à sua caracterização propriedades específicas, de entre as já existentes na ontologia. A ontologia foi denominada de *Ont4AAL* que significa "*Ontology for Ambient Assisted Living*".

No seguimento deste capítulo é apresentada a *Ont4AAL*, expondo as suas classes principais, as suas funcionalidades e as potenciais melhorias. São ainda apresentadas as relações existentes entre as suas classes e propriedades.

4.2. Ont4AAL

A ontologia é composta por 358 classes e 618 subclasses, 1693 axiomas (regras/relações/restrições), sendo 873 destes lógicos (representam relações entre classes, objetos, etc.). Possui ainda 68 propriedades relativas aos objetos e 21 propriedades ao nível dos dados que permitem uma melhor descrição das classes. No entanto, as restantes estatísticas podem ser consultadas na Tabela 4.1.

Tabela 4.1 - Estatísticas da Ont4AAL.

Métrica	**Contagem**
Número de axiomas	1693
Número de axiomas lógicos	873
Número de axiomas "*disjoint*"	51
Número de classes	358
Número de subclasses	618
Número de propriedades de objetos	68
Número de propriedades de objetos inversas	32
Número de subpropriedades de objetos	36
Número de propriedades de objetos com restrições ao nível do domínio	63
Número de propriedades de objetos com restrições ao nível da sua abrangência	62
Número de propriedades de dados	21
Número de subpropriedades de dados	5

A ontologia construída assenta fundamentalmente em 8 classes-mãe (Figura 4.1 e Figura 4.2) que são:

- *Bens*: modela todos os bens presentes dentro de uma habitação;
- *Comando:* traça todos os comandos passiveis de utilização pelos sensores, atuadores ou outros agentes;
- *Configuração:* descreve a habitação desde a sua tipologia às separações existentes na mesma;
- *Controlo:* ilustra todos os órgãos passiveis de controlo, sejam controladores, atuadores ou sensores;

- *Estado:* considera todos estados estáveis que os objetos controláveis podem assumir;
- *Funcionalidade*: traça as funcionalidades que cada objeto controlável pode fornecer;
- *Utilizador:* representa o(s) utilizador*(es)* presente*(s)* dentro de cada divisão, sendo ainda possível adicionar rotinas e preferências associadas a cada individuo.

Figura 4.1 - Vista geral das classes da ontologia.

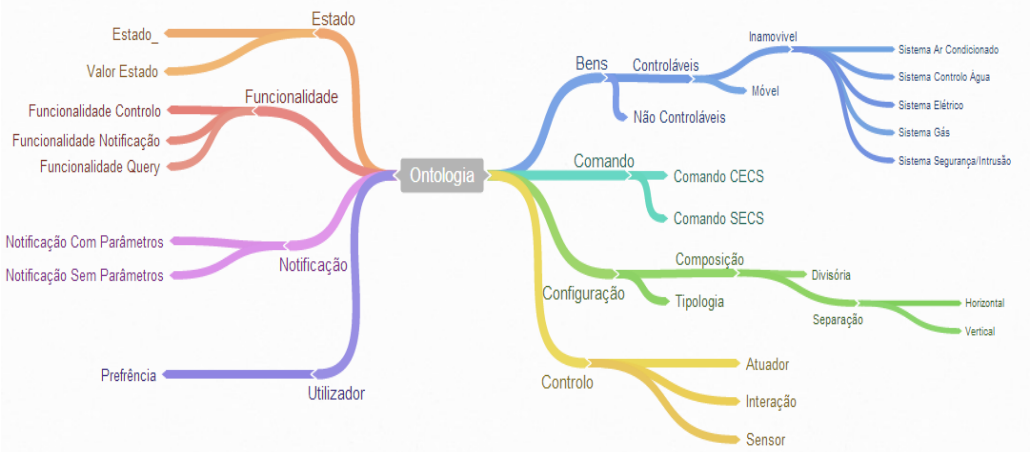


Figura 4.2 - Diagrama das classes da ontologia.

Nos subcapítulos que se sucedem encontram-se descritos os vários aspetos modelados para a representação de uma habitação.

4.2.1. Modelação do ambiente

A modelação do ambiente é atingida através dos conceitos herdados da *Configuração* e dos *Bens*. A descrição deste ambiente rigoroso foi reduzida, restringindo a sua definição ao mínimo de primitivas necessárias para a localização dos dispositivos, da mobília e bens presentes num edifício. Podem ser representados edifícios completos estendendo estas duas classes da ontologia através de subclasses

adicionais e consequente definição de relações apropriadas (a titulo de exemplo, a introdução de modelação e *reasoning* espacial [108][109]**).**

Os *Bens* (Figura 4.3) representam todos os equipamentos que podem existir na habitação, sendo alguns específicos a cada divisão (por exemplo o sistema de ar condicionado). A ontologia separa ainda objetos que podem ser controlados (objetos *Controláveis*) num sistema de domótica dos restantes objetos (objetos *Não Controláveis*).

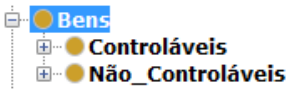


Figura 4.3 - Ontologia Classe de Bens.

Os objetos *Controláveis* encontram-se ainda divididos em *Móvel* e *Inamovível* (Figura 4.4). Através desta distinção, é possível dividir claramente quais os objetos que são afetos à casa (tal como o sistema de ar condicionado) e quais os aparelhos que podem variar a sua localização (por exemplo a televisão). Esta classe representa todos os bens que se encontram fixos, afetos à casa, cuja aplicação é específica e restringida a uma localização.

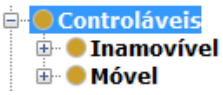


Figura 4.4 - Ontologia Classe de Bens Controláveis.

Os aparelhos móveis encontram-se separados em *Grandes Eletrodomésticos* (Figura 4.5) e *Pequenos Eletrodomésticos* (Figura 4.6), sendo classificados pelo seu consumo energético. Na primeira classe encontram-se alguns eletrodomésticos tais como o forno, frigorífico e máquina de lavar loiça. Na segunda estão aparelhos mais pequenos e de menor consumo tais como o telefone, a batedeira elétrica ou o computador. Nestas duas categorias estão presentes os eletrodomésticos mais utilizados no quotidiano do utilizador comum. Apesar de não estarem representados todos os aparelhos existentes no mercado, passíveis de serem inseridos nestas duas categorias, foram contemplados ainda uma taxa elevada de aparelhos. Tal como à semelhança de outras classes, estas também podem ser facilmente alargadas sendo apenas indispensável associar-lhes as propriedades necessárias.

Quanto ao nível de propriedades (Figura 4.7), os objetos *Controláveis* devem cumprir três requisitos. Estes objetos devem fornecer pelo menos uma funcionalidade (exatamente no sentido de serem considerados *Controláveis*), conter uma

Funcionalidade de Query de modo a responder a certos pedidos de informação e, por último, deverão ser capazes de notificar o sistema central de alterações no seu estado.

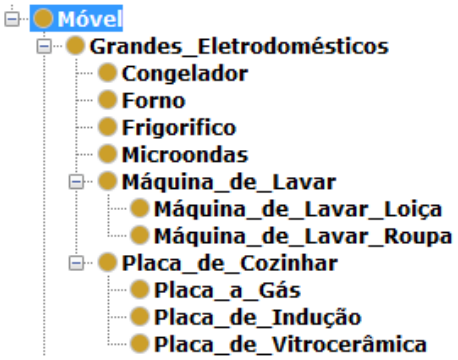


Figura 4.5- Classe de Bens Controláveis Móvel - Grandes Eletrodomésticos.

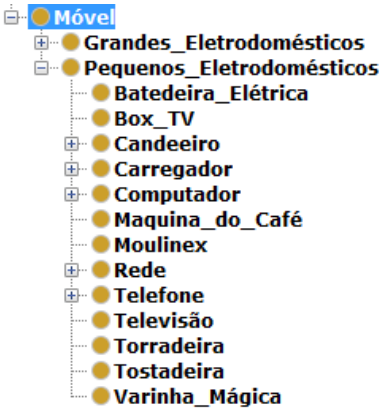


Figura 4.6 - Classe de Bens Controláveis Móvel - Pequenos Eletrodomésticos.

As funcionalidades referidas serão expostas nos subcapítulos seguintes e por isso nesta secção não vai ser apresentada uma descrição detalhada das mesmas.

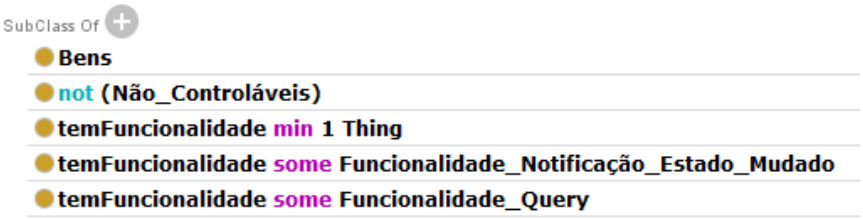


Figura 4.7 - Propriedades da Classe de Bens Controláveis.

É ainda de salientar que os objectos Controláveis se encontram nitidamente distintos dos objetos *"Não Controláveis"* fazendo uso da segunda propriedade "*disjoint*". Esta propriedade indica que nenhuma instância poderá fazer parte das duas classes em simultâneo (*Controláveis* e *Não Controláveis*). Será também possível expandir esta classe adicionando novos dispositivos e consequentes propriedades.

No que diz respeito aos aparelhos afetos à casa (Figura 4.8) encontram-se diversos sistemas modelados: *Sistema de Ar Condicionado*, *Sistema de Controlo de Água*, *Sistema Elétrico*, *Sistema de Gás* e *Sistema de Segurança/Intrusão*. Cada um descreve, naturalmente, todas as particularidades do sistema em questão.

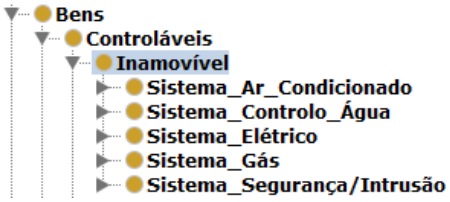


Figura 4.8 - Classe de Bens Controláveis Inamovível.

Aprofundando mais estas categorias, irá analisar-se a primeira classe – *Sistema de Ar Condicionado*. Na Figura 4.9 estão modelados os componentes essenciais a este sistema. Estes foram aferidos através do estudo dos diversos sistemas de ar condicionado. Após este estudo foram avaliados os constituintes essenciais a todas as configurações analisadas, sendo apresentados nesta classe. Pontualmente, poderá faltar algum componente específico a um modo de instalação ou aparelho de ar condicionado. Naturalmente, a cada constituinte desta classe encontram-se associadas inúmeras propriedades que, em conjunto com os restantes dispositivos, compõem o sistema de ar condicionado.

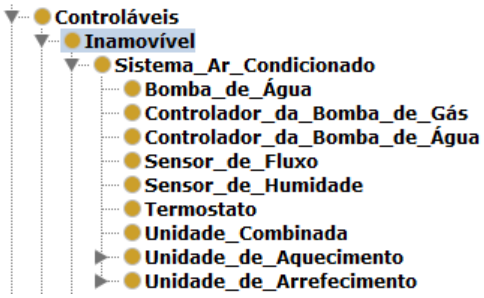


Figura 4.9 - Classe de Bens Controláveis Inamovível - Sistema de Ar Condicionado.

Para demonstrar, analise-se as propriedades da "Bomba de Água", expostas na Figura 4.10. Neste caso, são as mesmas representadas pela classe *Bens Controláveis*, tal como já apresentado na Figura 4.7.

Figura 4.10 - Propriedades da Classe de Sistema de Ar Condicionado - Bomba de Água.

A parte superior da figura representa as propriedades específicas à classe em questão. É possível aferir que esta classe se encontra inserida também em outros sistemas ou classes, tal como a classe *Bomba* (referida nas próximas secções) e *Sistema de Controlo de Água*. É importante referir que, apesar de fazer parte de diversos sistemas, as propriedades são específicas à classe, não mudando consoante o contexto.

Desta forma, a bomba de água instanciada terá de estar inserida em apenas um sistema (podem instanciar-se mais bombas de água para outros sistemas) e ser controlada por pelo menos um dispositivo. À parte das características herdadas terá ainda de ter um de dois estados: ligado ou desligado. Deste modo, a classe *Bomba de Água* exposta representa uma exemplificação geral da forma com as características estão distribuídas neste subsistema. À sua semelhança existem numerosos dispositivos espalhados pelos restantes sistemas, tais como se pode verificar na Figura 4.11.

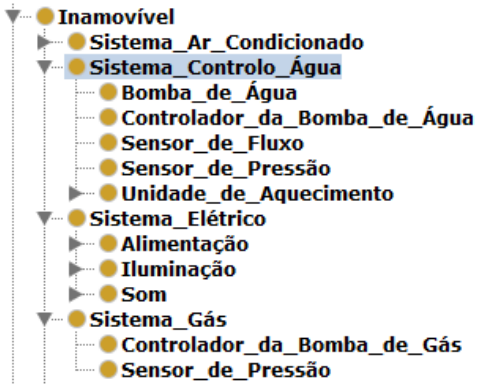


Figura 4.11 - Amostragem de classes presentes na classe de Bens Controláveis Inamovível.

Deixada de parte da figura anterior devido à sua extensão, encontra-se a classe de *Sistema de Segurança/Intrusão* (Figura 4.12), que se encontra dividida em várias subclasses: *Anti-Fugas*, *Anti-Incêndio*, *Anti-Inundação* e *Controlo de Acessos*.

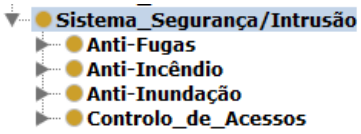


Figura 4.12 - Classe de Sistema de Segurança/Intrusão.

Cada sistema apresentado é composto pelos vários sensores e atuadores que constituem o sistema em questão. Estes estão ilustrados na Figura 4.13. Cada uma das classes detém uma imposição obrigatória de ter pelo menos um sensor ou um atuador para a sua instanciação. As restantes propriedades são herdadas das classes-mãe, tal como já demonstrado.

Os *Bens Não Controláveis* (Figura 4.14) modelam todos os bens que, embora façam parte do ambiente doméstico, não facultam qualquer tipo de forma de controlo. A classe foi implementada para ser possível descrever de forma mais rica o ambiente em que o utilizador se encontra inserido, descrevendo todos os constituintes básicos de uma habitação.

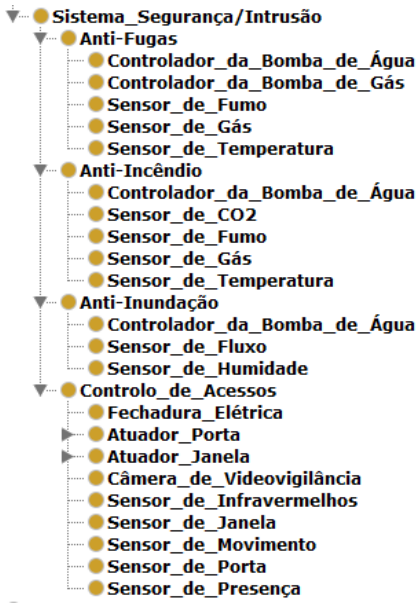


Figura 4.13 - Classe de Sistema de Segurança/Intrusão expandida.

Esta classe serve também para se aferir se o objeto *Não Controlável* poderá influenciar a ação de algum sensor ou atuador.

Figura 4.14 - Classe de Bens Não Controláveis.

Serve ainda de apoio à classificação de situações. De forma a exemplificar, considera-se que um utilizador se encontrar deitado no sofá, nesta situação poderá transmitir ao sistema que este vai dormir. No entanto, se estiver sentado pode simplesmente estar a ler um livro ou a ver televisão.

Outra parte integrante desta classe é a *Configuração* (Figura 4.15) que representa, um ambiente doméstico e todos os aspetos arquiteturais: vários tipos de divisões, garagem e jardim. Encontra-se dividida em duas categorias: *Composição* e *Tipologia*.

Figura 4.15 - Ontologia Classe de Configuração.

A primeira categoria, *Composição*, representa todos os constituintes internos da casa a nível arquitetural. Isto é, indica as divisões, as separações e até as varandas. A segunda descreve que tipo de imóvel se está a representar. A classe *Composição* subdivide-se em *Divisória* (Figura 4.16) e *Separação* (Figura 4.17). Representam assim, as divisórias que a casa pode possuir (quarto, sala, etc.) e as separações *Verticais* e *Horizontais* (balcão ou varanda) presentes na casa, respetivamente.

Existem várias condições aplicadas a estas subclasses. À classe *Divisória* foi aplicada a regra de ter pelo menos uma separação. Tal implica que esta pelo menos tenha alguma parede representada. Poderia ter sido escolhida a obrigatoriedade de ter pelo menos quatro paredes (fechar a divisão) mas implicaria limitações arquitetónicas, o que não é desejável. Desta forma, é possível uma instanciação mais abrangente e personalizável.

Figura 4.16 - Classe de Composição - Divisória.

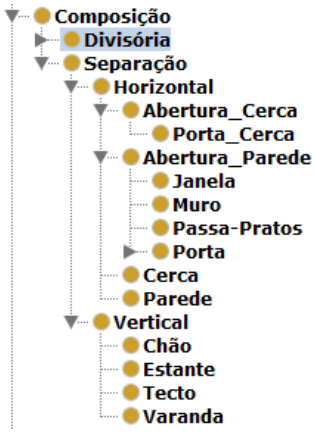


Figura 4.17 - Classe de Composição - Separação.

Foram ainda aplicadas algumas restrições às aberturas de porta, parede e vedações, sendo representadas as duas primeiras pela Figura 4.18 e Figura 4.19, respetivamente. Foi também estabelecido um limite de dois atuadores no máximo por cada abertura na parede. Tal deve-se à existência de atuadores que, apesar de conseguirem abrir a porta, *p.e.*, não a conseguem fechar, sendo imposta a necessidade de serem instalados dois atuadores (regra número um). Por outro lado, foi limitada a utilização de atuadores capazes de atuarem sobre o estado da porta (imposição número dois). No entanto, só é possível instanciar um sensor por porta para aferir se esta se encontra aberta ou fechada (regra número três).

Segue-se a *Tipologia* (Figura 4.20) que diz respeito ao tipo de habitação em questão, classificando-a em *Apartamento*, *Estúdio* ou *Vivenda*.

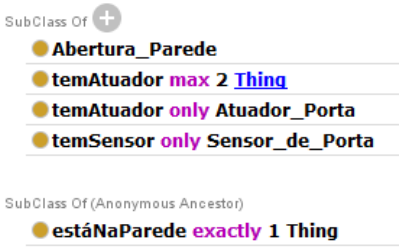


Figura 4.18 - Classe Abertura de Porta.

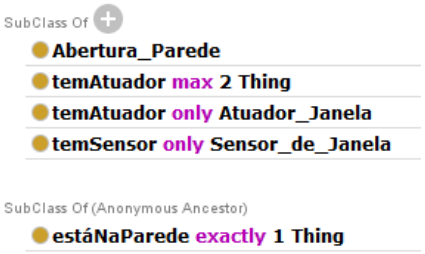


Figura 4.19 - Classe Abertura de Parede.

Em conjunto com os *Bens*, é responsável por representar os aspetos arquiteturais da habitação.

Figura 4.20 - Ontologia Classe de Tipologia.

Esta classe possui apenas duas restrições (Figura 4.21) aplicadas, implicando que a sua instanciação aconteça segundo determinadas normas.

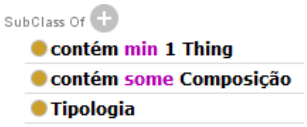


Figura 4.21 - Propriedades das Classes da Tipologia.

Assim, qualquer uma das três subclasses da classe *Tipologia*, deverá possuir alguma composição interna, representando as divisões e separações e pelo menos uma "*Thing*", sendo assim possível adicionar *Bens* ou os restantes sistemas já mencionados.

4.2.2. Modelação de dispositivos

Após a modelação do ambiente habitacional onde estão inseridos os sistemas de apoio ao utilizador , é necessário modelar os dispositivos que os suportam. De modo a alcançar uma representação o mais exata possível destes, foram criadas quatro classes que em conjunto compõem o dispositivo. Estas classes são: *Comando*, *Funcionalidade*, *Notificação* e *Estado*. Representam, respetivamente, os comandos para interação com o dispositivo, suas funcionalidades, suas capacidades ao nível de notificação e quais os estados que este pode assumir. Estas serão explicadas nas seguintes secções seguindo a ordem apresentada.

Modelação de comando

A classe *Comando*, como já referido anteriormente é responsável por contemplar todas as instruções com que é possível interagir com os *Bens Controláveis*. Por forma a distinguir quais os comandos que requerem parâmetros de entrada dos que não necessitam, foram criadas duas subclasses (Figura 4.22): *Comandos CECS* (comandos com entrada e com saída) e *Comandos SECS* (comandos sem entrada e com saída). Assumiu-se à partida que o dispositivo fornece sempre algum tipo de resposta, quer seja física (abrir uma porta, *p.e.*) quer resposta a um dado pedido (medição de temperatura, *p.e.*). É importante ressalvar que estas duas classes encontram-se também visivelmente separadas pela propriedade "*disjoint*".

Figura 4.22 - Ontologia Classe de Comando.

A primeira classe (Figura 4.23) diz respeito aos comandos com parâmetros de entrada e qualquer forma de saída. Esta saída pode ser o valor de uma leitura de um sensor, a resposta a um pedido de fecho de estore ou outra situação semelhante. Foram retratados o maior número de comandos possíveis, passiveis de serem utilizados num sistema de *AAL*. No entanto, tal como as classes anteriores, no futuro poderão ser adicionados novos comandos específicos a protocolos de comunicação, a parametrização de determinados atuadores, entre outros. Pode também ser necessário proceder à alteração de alguns dos comandos implementados para uniformizar a formatação destes com os novos comandos inseridos. De notar que, devido à sua extensão a classe não se encontra totalmente ilustrada na figura que se segue.

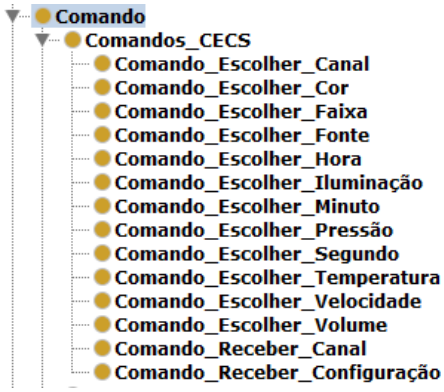


Figura 4.23 - Ontologia Classe de Comandos com Entrada e com Saída.

A modelação desta classe implicou ainda a construção de regras específicas a cada uma das classes referidas. Essencialmente, a grande diferença entre as duas assenta no número de parâmetros de entrada. Enquanto a primeira possui pelo menos um parâmetro de entrada, a segunda não aceita nenhum. Na Figura 4.24 e Figura 4.25 estão assim ilustradas estas diferenças, respetivamente.

Figura 4.24 - Propriedades da Classe de Comando com Entrada e com Saída.

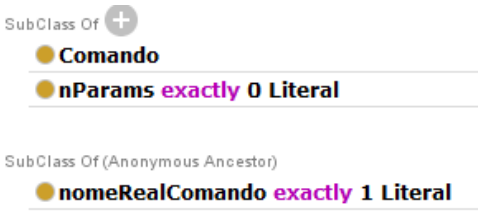


Figura 4.25 - Propriedades da Classe de Comando sem Entrada e com Saída.

Como é possível aferir pelas figuras apresentadas, estão presentes duas propriedades a nível dos dados. A primeira é a "*nParams*", uma "*data propertie*" que representa o número de parâmetros de entrada. No caso dos comandos com entrada, tem de existir pelo menos um parâmetro de entrada para que este seja válido e processável pelo sistema. Já nos comandos sem entrada, o número de parâmetros apresentados têm de ser exatamente zero. Em relação à segunda *"data propertie", "nomeRealComando"*, serve para definir especificamente qual o nome interno do comando para o sistema. Considere-se o *Comando com Entrada e com Saída Comando_Escolher_Canal* ilustrado na Figura 4.26. Apesar do nome da classe ser "*Comando_Escolher_Canal*", internamente este representa um comando do género "*selecionarCanal(canal)*", sendo o canal uma variável do tipo *string*. É desta forma mais simples e natural interagir com o sistema, de forma *standardizada* a todos os comandos.

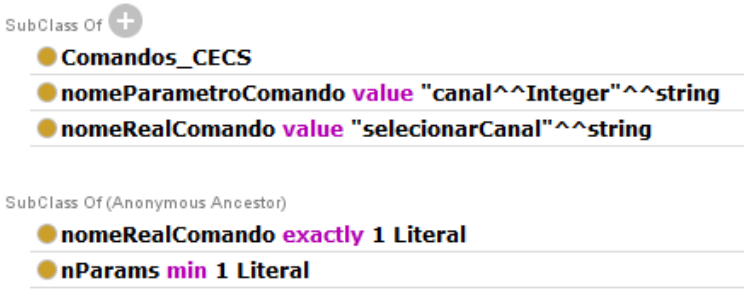


Figura 4.26 - Propriedades do *Comando_Escolher_Canal*.

Observando a figura pode ainda notar-se a existência de mais uma propriedade a nível dos dados que ajuda à composição do comando em si. Assim, o "*nomeParametroComando*" representa o formato do parâmetro de entrada. No caso do comando apresentado, é requerido que o canal seja fornecido segundo a regra imposta.

A seguir ao nome "canal" espera-se encontrar um valor alfanumérico, indicando o número do canal desejado. Juntado as duas "*data properties*" formadas, atinge-se o formato final do comando – "*selecionarCanal(canalX)*", sendo "X" o número do canal. Este formato estende-se aos restantes comandos apresentados na Figura 4.23.

Quanto aos comandos sem entrada e com saída (Figura 4.27), existem algumas particularidades a serem referidas. Apesar de não receberem parâmetros de entrada implicitamente, para o sistema estes existem internamente. Isto é, apesar de não se realizar a operação de abertura de uma porta, por exemplo., seguindo o *standard* anteriormente apresentado (operação *"abrir(portaX)"*), é apenas necessário invocar o comando "*abrir*". Tal particularidade deve-se ao facto de cada objeto, quando instanciado, ser caracterizado através das suas funcionalidades. Desta forma, o sistema sabe internamente que a porta "X" possui a capacidade de ser alvo de atuação para abertura. Assim, o sistema apenas se refere ao objeto e aplica o comando geral de abertura. Não foi também especificado diretamente o âmbito da ação. Não existem assim comandos imediatos de "*abrirPorta*", por exemplo. Existe o comando generalizado de abertura, "*abrir*". Pela mesma razão apresentada anteriormente, dificilmente um dispositivo terá mais que um comando de abertura. Mesmo que seja passível de possuir dois comandos de abertura, no caso da porta, que se pode considerar a abertura da porta e a abertura da fechadura ou fecho elétrico, dizem respeito a atuadores diferentes. Assim, cada um possui uma funcionalidade única de abertura (imposta aquando da instanciação), não complicando a tarefa do sistema.

Figura 4.27 - Ontologia Classe de Comandos sem Entrada e Com Saída.

As propriedades referidas encontram-se ilustradas na Figura 4.28.

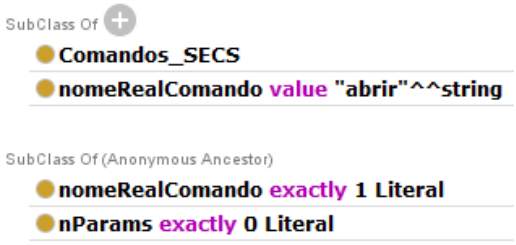


Figura 4.28 - Propriedades do *Comando_Abrir*.

Como seria espectável, a grande diferença nas propriedades deste comando, face aos presentes na classe de comandos com entrada e com saída, está no número de parâmetros de aceita para entrada (zero, neste caso).

São assim abrangidos todos os comandos utilizados pelo sistema para interação com os dispositivos.

Modelação de funcionalidades

As subclasses de *Bens Controláveis* e *Não Controláveis* diferem, essencialmente, na obrigatoriedade que a primeira tem de satisfazer certas restrições ao nível de *Comandos*, *Funcionalidades* e *Estados*. Cada dispositivo na ontologia está associado a um conjunto de funcionalidades únicas, oficializadas através da relação "*temFuncionalidade*". Podem, ser utilizadas várias estratégias de modelação de funcionalidades: a técnica composicional, tal como utilizada no projeto DomoML[110][111], onde as funcionalidades de um dado dispositivo derivam das funcionalidades providenciadas pelos seus constituintes; um modelo descritivo, onde as funcionalidades são descritas separadamente e depois relacionadas de modo a compor um único dispositivo. A ontologia construída assenta nesta última técnica e modela as funcionalidades por objetivos. Isto permite que seja utilizada apenas uma instância por funcionalidade, visto as capacidades do dispositivo serem modeladas independentemente da classe do mesmo. Cada funcionalidade define o comando para atuar sobre uma propriedade de um dado dispositivo (a intensidade da luz, p.e.) e os valores que pode assumir. As funcionalidades encontram-se divididas de acordo com o seu objetivo. Assim foram criadas três subclasses (Figura 4.29): *Funcionalidade_Controlo*, *Funcionalidade_Notificação* e *Funcionalidade_Query*.

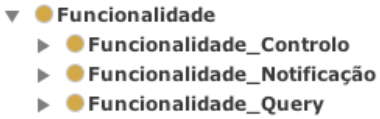


Figura 4.29 - Ontologia Classe de Funcionalidade.

A classe *Funcionalidade_Controlo* (Figura 4.30) modela uma variedade de capacidades que o dispositivo pode fornecer a nível de controlo. Naturalmente, um dispositivo comum utilizará uma ou duas das funcionalidades presentes nesta classe.

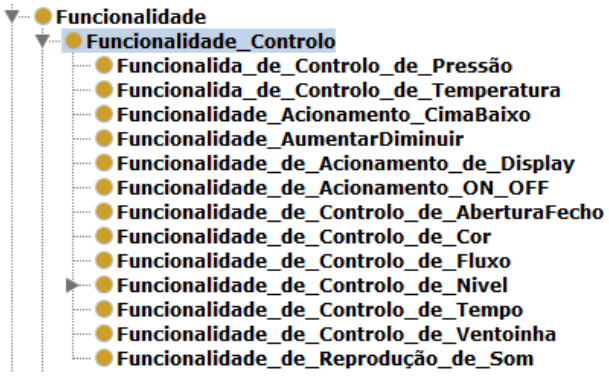


Figura 4.30 - Ontologia Classe de *Funcionalidade_Controlo*.

As subclasses contidas nesta classe-mãe formam a capacidade de controlar um dispositivo ou parte dele, como por exemplo abrir um estore. Estão modeladas nestas classes as funcionalidades básicas de controlo dos diversos dispositivos. A cada uma destas funcionalidades está associado um controlo específico como será apresentado nos subcapítulos seguintes. Esta relação encontra-se implícita pela regra "*temControlo min 1Thing*".

A classe *Funcionalidade Notificação* (Figura 4.31) descreve a funcionalidade do dispositivo autonomamente notificar o seu estado interno e, em particular, a habilidade de detetar e notificar mudanças no seu estado.

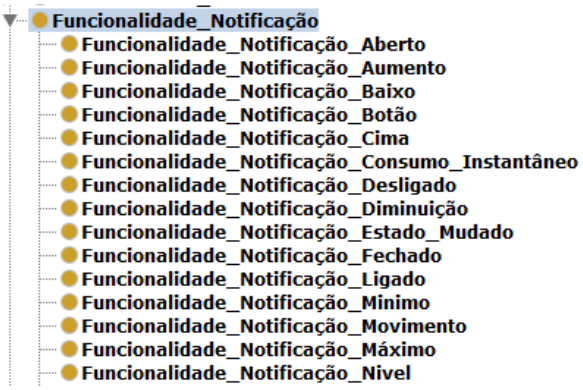


Figura 4.31- Ontologia Classe *Funcionalidade_Notificação*.

Esta classe é de elevada importância pois, será responsável por informar o sistema central das mudanças que ocorrem na habitação. A transmissão da informação (notificação) desencadeará uma resposta apropriada com a modelação da base de conhecimento sobre o domínio. A cada dispositivo está associado um número de funcionalidades de notificação, sendo obrigatório que pelo menos uma destas

funcionalidades esteja presente. Além desta norma foi ainda associada uma notificação correspondente a cada uma das funcionalidades apresentadas. Isto é, considerando a classe "*Funcionalidade_Notificação_Aumento*" (Figura 4.32), encontra-se implicitamente associada uma notificação tipo "*Notificação_Aumento*". Enquanto a primeira representa a formalização da funcionalidade de notificação, a segunda ilustra a formatação da notificação apresentada ao sistema. A modelação distinta destas duas classes permite uma melhor configuração das funcionalidades do dispositivo e uma maior customização da notificação apresentada. Torna ainda mais fácil a alteração da notificação para outro *standard*, não sendo preciso alterar a funcionalidade, apenas o formato da notificação.

A associação mencionada encontra-se vincada pela propriedade "*temNotificação*".

Por fim, a *Funcionalidade de Query* (Figura 4.33), engloba todas as habilidades de um dispositivo ser questionado sobre a sua condição, p.e, de falha, qual o seu valor atual, etc. Esta classe modela todos os pedidos que podem ser efetuados aos dispositivos e é através desta que as rotinas do sistema assentam.

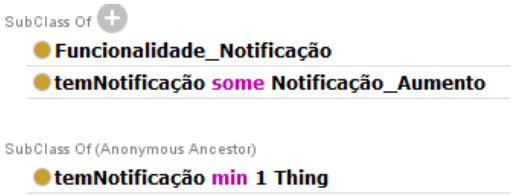


Figura 4.32 - Propriedades da Classe de *Funcionalidade_Notificação_Aumento*.

Isto é, para manter a temperatura ambiente a certo valor, p.e., o sistema terá de questionar o dispositivo ou simplesmente configura-lo para o notificar aquando da mudança do seu valor. No entanto, nem todos os aparelhos facultam o modo de notificação autónoma, sendo assim necessária a inclusão desta classe.

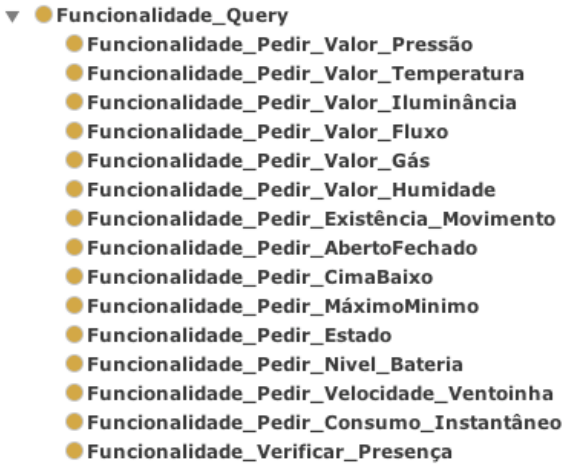


Figura 4.33 - Ontologia Classe de Funcionalidade de *Query*.

O procedimento explicado anteriormente constitui a forma como a *Ont4AAL* lida com as funcionalidades que cada dispositivo deverá possuir para poder ser classificado como bem controlável.

Modelação de notificações

A modelação de notificações surge da necessidade demonstrada na secção anterior. Uma das funcionalidades presentes nos dispositivos inseridos na habitação é a de notificarem o sistema de uma mudança no seu estado ou valor. Ditada esta funcionalidade é necessário modelar o formato que a mensagem será transmitida ao sistema central, sendo útil a criação de uma nova classe, a de notificações. Semelhante à classe de *Comando*, esta nova classe foi dividida de acordo com os parâmetros que transporta: *NotificaçãoComParâmetros* e *NotificaçãoSemParâmetros* como ilustrado na Figura 4.34.

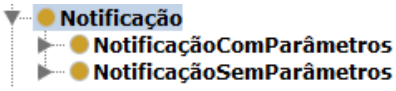


Figura 4.34 - Ontologia Classe *Notificação*.

Associadas à primeira subclasse, a de notificação com parâmetros, encontram-se as notificações que transportam valores. Assim, é possível encontrar nesta classe notificações sobre o valor da temperatura, humidade, entre outras variáveis (Figura 4.35).

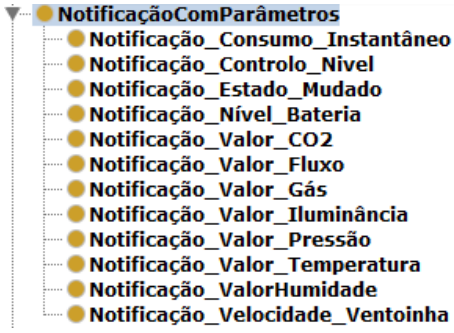


Figura 4.35 - Ontologia Classe de *NotificaçãoComParâmetros*.

Novamente, de forma semelhança à classe *Comandos*, as regras aplicadas são análogas. Utilizando a classe *Notificação_Consumo_Instasntâneo* para explicação, as normas a esta aplicada estão explícitas na Figura 4.36. A ela estão associadas essencialmente três "*data properties*": "*nomeNotificação*", "*nomeParametroNotificação*" e a já utilizada "*nParams*". As duas primeiras representam, respetivamente o formato da notificação interna do sistema e o formato do parâmetro a passar nessa notificação. Estas duas propriedades em conjunto constituem a

notificação final a ser entregue. Uma utilização possível deste comando será a seguinte *string standardizada*: "*novoConsumoInstantâneo(Medição10)*". Devido à natureza da classe (possuir parâmetros de saída), foi aplicada a última propriedade – possuir exatamente um parâmetro de saída.

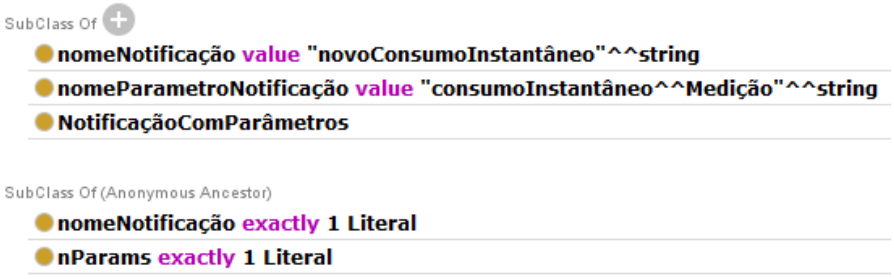


Figura 4.36 - Propriedades da Classe *Notificação_Consumo_Instantâneo*.

Relativamente à segunda classe, a de notificações sem parâmetros de saída, estão contidas nesta classe notificações do tipo "*aberto*", "*fechado*", entre outros (Figura 4.37).

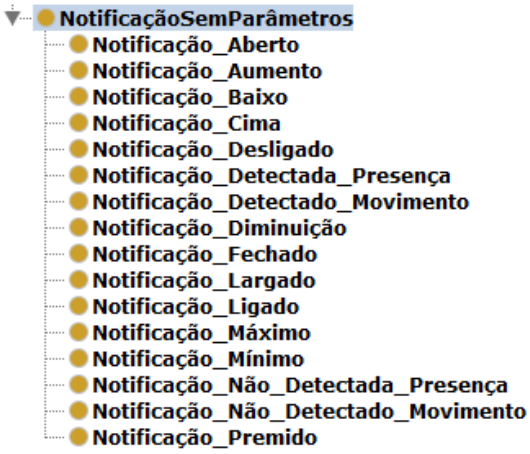


Figura 4.37 - Ontologia Classe *NotificaçãoSemParâmetros*.

Tal como já mencionado em secções anteriores, estas notificações apenas informam o sistema que certa funcionalidade do dispositivo se encontra alterada, sem recorrer à utilização de parâmetros. Assim sendo, as propriedades associadas a estas subclasses são essencialmente a formalização da utilização de zero parâmetros e do nome formato da notificação (Figura 4.38).

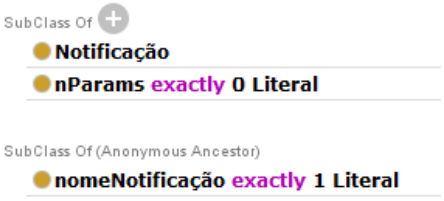


Figura 4.38 - Propriedades Classe *NotificaçãoSemParâmetros*.

Encontra-se assim concluída a descrição de como foram modeladas as notificações na ontologia Ont4AAL.

Modelação de Estados

Os estados são modelados de acordo com a mesma descrição utilizada para as funcionalidades; estes necessitam de ser instanciados para cada dispositivo dentro da habitação, visto que diferentes dispositivos pertencentes à mesma classe conceptual (lâmpadas, p.e), podem estar em diferentes estados (usualmente dois: ligadas ou desligadas). A modelação de estados encontra-se dividida em duas subclasses: *Estado_* e *Valor estado (*Figura 4.39*)*.

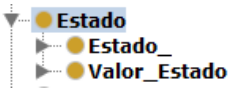


Figura 4.39 - Ontologia Classe Estado.

A primeira engloba todos os estados que os dispositivos podem assumir estando a cada estado desta classe associado um valor correspondente na classe *Valor Estado*. Na Figura 4.40 e Figura 4.41 encontram-se alguns dos estados e respetivos valores destes. De notar que, devido à extensão da classe, apenas algumas destas subclasses se encontram ilustradas no presente documento. A ordem das classes também não é obrigatoriamente coincidente devido aos nomes atribuídos a cada classe.

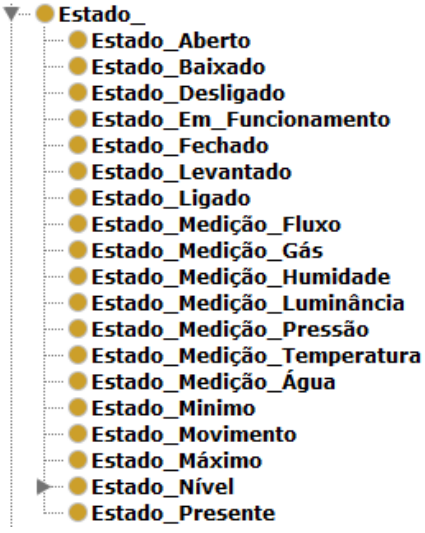


Figura 4.40 - *Ontologia Classe Estado_*.

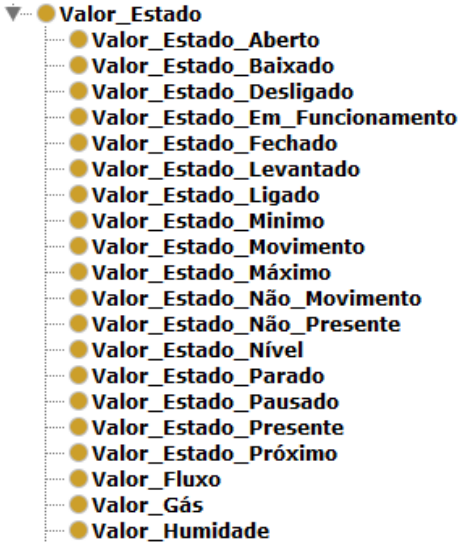


Figura 4.41 - Ontologia Classe *Valor_Estado*.

Considere-se um dispositivo com estado "*Estado_Aberto*". Apesar de nem sempre ser necessário saber qual o valor correspondente a este estado, como é o caso deste exemplo, para outras situações poderá ser proveitoso. Desta forma, a esta subclasse está associada uma subclasse correspondente na classe "*Valor_Estado*",

denominada de "*Valor_Estado_Aberto*". Esta associação foi implementada em todas as subclasses desta categoria, seguindo a norma exemplificada na Figura 4.42.

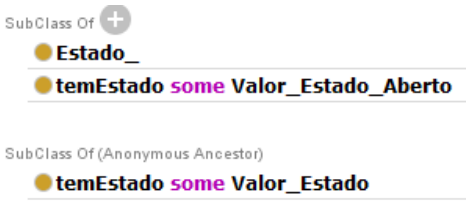


Figura 4.42 - Propriedades da Classe *Estado_Aberto*.

Deste modo, encontra-se associada a "*data propertie*" "*temEstado*" a um valor de estado presente na classe "*Valor_Estado*". A propriedade é ainda reforçada obrigando a que cada estado, para ser instanciado, possua um valor correspondente.

A modelação destas classes pode ser expandida tendo em vista um ponto de vista mais tecnológico. Isto é, poderão ser criados estados (e consequente valores dos estados) específicos para cada protocolo de comunicação (*zigbee*, p.e). Desta forma, a ontologia será mais escalável e abrangente. No entanto, para contemplar este grau de extensão será necessário definir à partida qual o tipo de *middleware* a ser utilizado em conjunto e algumas particularidades que se apliquem. Será também preciso conhecer os vários protocolos de comunicação dos dispositivos, a não ser que o *middleware* faça *query* à ontologia de forma indiferenciada. Isto é, este pode converter todos formatos de dados num formato global, sendo assim o tratamento de dados para a ontologia indiferente do protocolo. Quer seja utilizada uma abordagem mais direcionada ao protocolo ou uma aproximação mais geral, não poderá ser descuidado o formato geral dos dados ou da comunicação. Resumindo, a ontologia implementada fornece uma metodologia global como resposta a esta exigência (a cada estado associar um valor de estado).

4.2.3. Modelação de controlo

A modelação da classe de *Controlo* divide-se em três subclasses que permitem a "humanização" do sistema: *Sensor*, *Interação* e *Atuador*. A primeira classe (Figura 4.43) engloba todos os sensores presentes numa habitação. Estes podem ser incluídos individualmente em situações pontuais ou englobados em subsistemas onde funcionam para um todo, caso do *Sistema de Segurança/Intrusão*, p.e.

Não obstante o âmbito onde estes estão incluídos as suas funcionalidades não se alteram, tendo assim direito a uma classe individual, não sendo apenas parte integrante dos sistemas que formam.

Figura 4.43 - Ontologia Classe de Controlo e Sensor.

Apesar da separação ténue entre a classe *Sensor* e *Interação* (Figura 4.44), esta distinção foi feita de modo a se poder identificar facilmente quais os métodos de interação que o utilizador dispõe dos restantes sensores. Isto é, a classe *Sensor* encontra-se mais dirigida para incorporação num dispositivo, cujo utilizador não tenha interação direta. Por outro lado, a classe *Interação* representa todas as formas de contacto do utilizador com o sistema, desde botões, comutadores a controlos remotos.

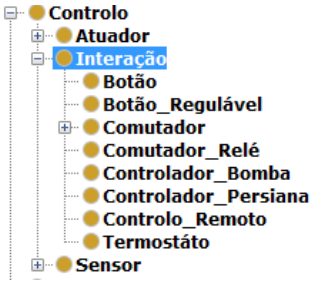


Figura 4.44 - Ontologia Classe de Controlo e Interação.

A classe *Atuador* (Figura 4.45), ilustra a forma de atuação que o sistema poderá ter em seu poder. Esta classe funciona em conjunto com as restantes mencionadas anteriormente. Apesar de ter direito a uma classe separada apenas por questões de referência, as capacidades de atuação estão intrinsecamente ligadas a cada dispositivo. Portanto, os controladores apresentados na secção anterior também poderiam estar inseridos nesta classe. Ao longo da ontologia estão também descritas as atuações específicas a cada dispositivo. Assim, na representação da lâmpada (por exemplo) encontra-se já especificada a capacidade de alterar a sua luminosidade (atuação).

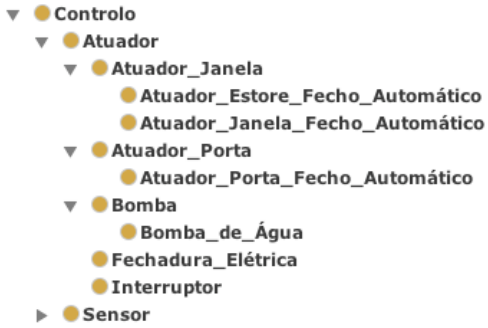


Figura 4.45 - Ontologia Classe de Controlo e Atuador.

Esta classe, encontra-se ainda associada à *Funcionalidade_de_Controlo*. Algumas das funcionalidades apresentadas utilizam recursos desta classe para o *reasoning* em situações convenientes.

4.2.4. Modelação do utilizador

A classe *Utilizador* (Figura 4.46) foi implementada de modo a ser possível aferir rotinas ou fixar preferências individualmente. A nível de subclasses é composta apenas pela classe *Preferência*. Cada utilizador é descrito segundo um conjunto de "*data properties*" que compõe uma descrição básica do mesmo.

Figura 4.46 - Ontologia Classe de Utilizador.

Como é possível visualizar estão englobados alguns dos aspetos gerais do utilizador, tal como o nome, idade e sexo. É ainda representado o seu estado de saúde atual e a sua situação de emprego estando estas relações apresentadas na Figura 4.47.

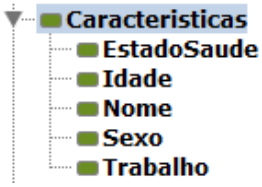


Figura 4.47 - Propriedades Classe Utilizador.

A cada utilizador existente está ainda associado, caso seja aplicável, graus de parentesco face aos restantes. Assim, foi composta uma vasta rede de parentescos, que podem ser associadas aos utilizadores (Figura 4.48). Esta permite que se crie um ambiente muito mais enriquecedor do ponto de vista conceptual.

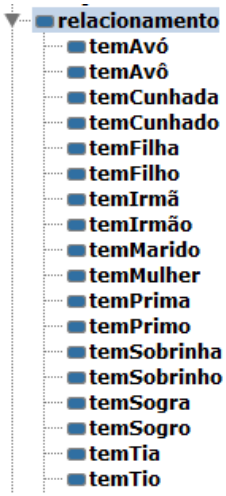


Figura 4.48 - Propriedades de parentesco associadas a cada utilizador.

As propriedades apresentadas têm ainda uma particularidade merecedora de referência, estão demarcadas com a propriedade inversa. Isto é, consideremos o "*utilizador1*" que "*temAvó*" a "*utilizadora1*". Estando ativada a propriedade inversa, na descrição da "*utilizadora1*" irá existir uma relação "*éAvóDe utilizador1*". Desta forma, é desnecessário produzir uma relação aos dois utilizadores. Basta aplicar a propriedade a um dos utilizadores que a ferramenta fará o resto do processamento. Assim, a instanciação de um utilizador criará um perfil geral sobre o mesmo (Figura 4.49).

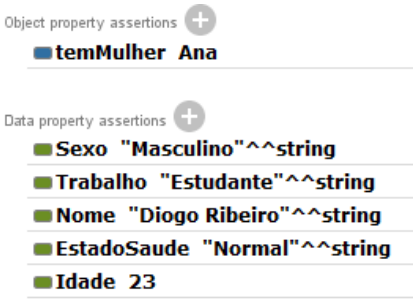


Figura 4.49 - Exemplo de um utilizador instanciado.

Apesar da composição simples a nível de classes, esta classe faculta ainda a possibilidade de criação de preferências por utilizador. A cada preferência está associado um *_Estado* e um *Valor Estado*. Esta preferência é memorizada através da propriedade "*temPreferência*" (Figura 4.50). Esta premissa é importante pois permite que o sistema saiba, em todos os instantes, que o utilizador presente na divisão tem determinada preferência (nível de luminosidade, p.e.). Cada preferência deverá ser vincada aquando da instanciação. Cada preferência tem associado um único utilizador.

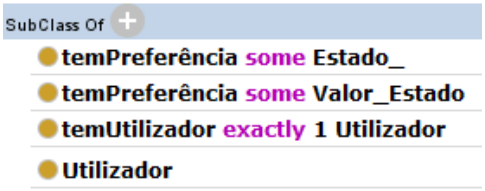


Figura 4.50 - Propriedades Classe *Utilizador*.

A situação de múltiplos utilizadores numa divisão não está contemplada na ontologia. No entanto, poderá ser estabelecida uma estratégia de compromisso, onde a configuração do ambiente na divisão resulta da média dos diferentes níveis de preferência.

4.3. Conclusão

A ontologia construída é composta por vários grupos modelados separadamente. Inicialmente modelou-se o ambiente doméstico, descrevendo os seus aspetos arquiteturais e os seus bens. De seguida, compuseram-se as funcionalidades dos seus dispositivos descrevendo de que forma estes contribuem para o sistema. A modelação do controlo é igualmente importante pois refere-se à forma como se podem controlar os bens, de forma automática ou através de controlos efetuados pelo utilizador. Por fim, foi modelado o utilizador, as suas relações e preferências. Estes grupos em conjunto constituem o ambiente doméstico automatizado.

CAPÍTULO 5

Implementação e resultados

5.1. Introdução

Após a modelação da ontologia *Ont4AAL*, a sua validação requer uma instanciação vocacionada das suas classes. Visto a construção da ontologia ter sido feita tendo em conta o objetivo de utilização, será necessário dividir a sua validação em partes. Desta forma foram construídos alguns cenários de teste que assentam numa habitação do tipo T1 (um único quarto) como modelo. Os elementos constituintes de cada caso de estudo variam, tendo como objetivo visar o maior número de cenários possíveis. A instanciação da ontologia foi divida em três casos de estudo, tendo como base um modelo geral apresentado seguidamente. Inicialmente, importou-se a ontologia já construída e particionada anteriormente para o programa *Protégé*, variando de seguida os elementos instanciados para serem alvo de estudo. De seguida aplicaram-se algumas regras *SWRL* que não são aferidas automaticamente através das restrições impostas na *Ont4AAL*. Estas variam consoante o caso de estudo e serão mencionadas quando for relevante.

5.2. Modelo de estudo

O modelo habitacional desenvolvido (Figura 5.1) assenta numa habitação do tipo apartamento, *T1*, com uma casa de banho, sala e cozinha. Possui ainda uma zona comum de acesso ao quarto, sala e cozinha. A casa é ainda provida de duas varandas na sua zona frontal (a norte) e traseira (a sul), respetivamente. Desta forma, foram estabelecidas as instanciações básicas necessárias à caracterização da casa. Inicialmente é necessário estabelecer os limites das divisões através de paredes, telhado, janelas e portas. Estes representam alguns dos elementos invariáveis a todos os elementos de estudo. A esta categoria há ainda a acrescentar os bens não controláveis. Apesar de poderem interferir na forma com o sistema responde ativamente, não será considerada uma configuração diferente da estipulada inicialmente.

Figura 5.1 - Modelo habitacional de testes.

Através da análise da figura já mencionada, é possível aferir os seguintes números de dispositivos inteligentes em cada divisão.

Tabela 5.1 - Contagem de dispositivos da casa modelo.

Divisão	**Dispositivo**	**Contagem**
Cozinha	Tomada	4
	Iluminação Interior	2
	Iluminação Exterior	1
	Interruptor de estores	1
	Interruptor	2
Hall	Tomada	1
	Iluminação Interior	1
	Iluminação Exterior	1

	Interruptor de estores	0
	Interruptor	2
Quarto	Tomada	3
	Iluminação Interior	2
	Iluminação Exterior	2
	Interruptor de estores	1
	Interruptor	3
	Tomada	3
Sala	Tomada	5
	Iluminação Interior	2
	Iluminação Exterior	1
	Interruptor de estores	1
	Interruptor	2
	Tomada	3
Quarto de Banho	Tomada	1
	Iluminação Interior	1
	Iluminação Exterior	0
	Interruptor de estores	0
	Interruptor	0

Esta contagem de dispositivos será variável em cada caso de estudo em vigor. Parte-se ainda do princípio que cada dispositivo mencionado fornece um conjunto de funcionalidades básicas pelo de número igual ou superior às mencionadas em 4.2.2: *Funcionalidade_Controlo*, *Funcionalidade_Notificação* e *Funcionalidade_Query*.

A utilização de um ambiente habitacional implica a instanciação direta dos diversos constituintes da residência, possibilitando a interação com o meio. Assim, foi criada uma nova ontologia utilizando a ferramenta *Protégé*, para onde se importou a já existente (sem qualquer instanciação). Esta separação é aconselhável pois qualquer alteração na ontologia original se repercutirá na nova ontologia. Qualquer alteração à modelação original não obrigará a tempos de carregamento muito superiores (devido ao avultado número de instanciações). A importação de ontologias é um processo simples e atingível através da própria interface do *software* (Figura 5.2).

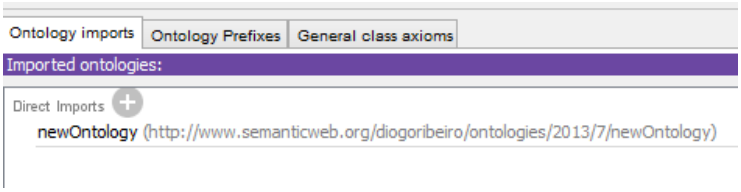


Figura 5.2 - *Protégé* - Importação de Ontologias.

Após a importação da ontologia construída, é possível aceder a todos os campos da mesma. A instanciação da habitação pode assim ser efetuada não afetando diretamente a ontologia original.

O modelo original da habitação instanciado não contempla a habitação e todos os dispositivos mencionados pois a existência destes é variável consoante cada caso de estudo.

5.3. Casos de estudo

A Ont4AALé uma ontologia extensa sendo necessária uma validação baseada em diversos casos de estudo. Foram elaborados três casos de estudo em que se põe à prova as capacidades de instanciação da ontologia, nas diversas divisões existentes na casa. Assim, no primeiro caso de estudo procede-se à instanciação dos componentes constituintes da divisão cozinha. No seguinte caso de estudo objetivo é instanciar a divisão sala. Por fim, falta representar a divisão quarto e os utilizadores presentes na habitação. Não sendo possível validar todas as classes modeladas na ontologia, estes três casos de estudo servirão de amostra da restante representação

5.3.1. Caso de estudo 1

O primeiro caso de estudo refere-se à representação da divisão cozinha, recorrendo-se à aba "*Individuals*" da ferramenta *Protégé*. A análise da Figura 5.1 e Tabela 5.1 permite que se distingam os componentes a instanciar e quais as classes a que pertencem (Tabela 5.2). Esta classificação é importante pois ditará a ordem de instanciação, quais as propriedades a aplicar e respetiva aplicação.

Tabela 5.2 - Caso de Estudo 1 - Composição da Cozinha.

Categoria	Objeto	Quantidade
Bens Não Controláveis	Mesa	1
	Cadeira	4
	Móvel	4

Bens Controláveis	Tomada	4
	Interruptor	2
	Interruptor de Estore	1
	Lâmpada	2

Inicialmente, criou-se uma habitação do tipo apartamento, onde apenas se instanciou a cozinha. A esta habitação deu-se o nome de "*Casa_Teste*". De seguida, aplicaram-se os limites da cozinha estabelecendo as suas paredes, que divisões separam e quais os objetos nelas presentes (caso de janelas ou portas). Criaram-se assim as instâncias: "*Parede_Cozinha_Exterior*", "*Parede_Cozinha_Sala*", "*Parede_Cozinha_VarandaNorte*" e "*Parede_Cozinha_Hall*". Enquanto as duas primeiras não necessitam de qualquer caracterização adicional devido à inexistência de aberturas, iluminação, entre outras, as duas últimas requerem maior especificação. Na "*Parede_Cozinha_VarandaNorte*", está presente uma janela com estore que fornece capacidades de controlo. Na Figura 5.3 está apresentado o trabalho de instanciação executado, representativo da descrição de um estore presente na cozinha.

Figura 5.3 - Instanciação Cozinha com janela e estore controlável.

O número de instâncias apresentadas representam o menor número possível, sendo este ditado pelas limitações impostas aquando da construção da ontologia. É possível aferir a classificação adicional que foi elaborada de modo a representar a porta de acesso ao *hall* e a janela de acesso à varanda. Nesta etapa, tornam-se fundamentais as "*Object_Properties*" definidas anteriormente. A Figura 5.4 ilustra três utilizações destas propriedades em prol do caso de estudo em questão.

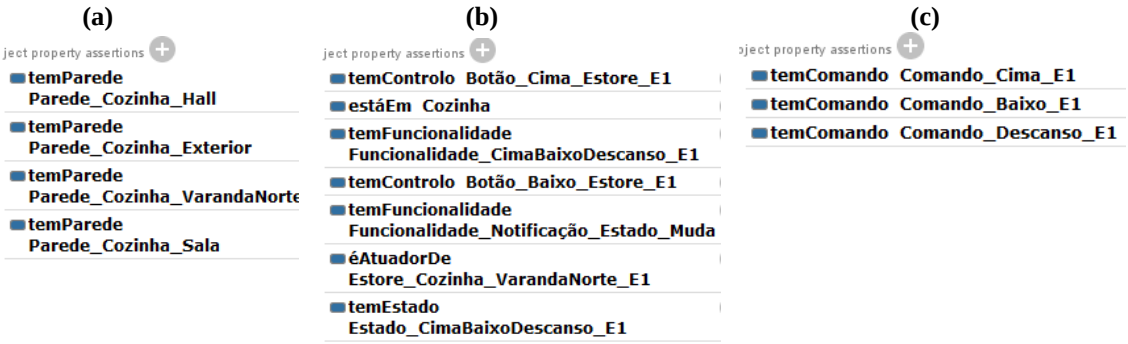


Figura 5.4 - Utilização de *Object_Properties* na Cozinha - (a) Caracterização da cozinha, (b) Caracterização do atuador do estore e (c) Caracterização da funcionalidade cima, baixo e descanso.

A ferramenta *Protégé*, gera automaticamente o código *RDF/XML* necessário nas próximas etapas. Visto ser um processo moroso e repetitivo torna-se propicio ao erro. Desta forma, visto a ferramenta facultar o código gerado, a instanciação dos restantes estores (necessária aos casos de estudo 2 e 3) será executada através da duplicação do código, mudando apenas os nomes a instanciar. Apenas como referência, está apresentado algum do código gerado neste processo. Representam o atuador do estore da cozinha (Figura 5.5) e uma das suas funcionalidades, o controlo através de botões (Figura 5.6).

```
<owl:Thing rdf:about="http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome#Atuador_Estore_Cozinha_Var
    <rdf:type rdf:resource="&newOntology;Atuador_Estore_Fecho_Automático"/>
    <rdf:type rdf:resource="&owl;NamedIndividual"/>
    <newOntology:temControlo rdf:resource="http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome#Botão
    <newOntology:temControlo rdf:resource="http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome#Botão
    <newOntology:estáEm rdf:resource="http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome#Cozinha"/>
    <newOntology:temEstado rdf:resource="http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome#Estado_
    <newOntology:éAtuadorDe rdf:resource="http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome#Estore
    <newOntology:temFuncionalidade rdf:resource="http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome
    <newOntology:temFuncionalidade rdf:resource="http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome
</owl:Thing>
```

Figura 5.5 - Código Gerado para o estore 1 - Atuador do estore.

De notar que a figura apenas representa 41 das 418 linhas de código necessárias para representar tal componente.

```
<!-- http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome#Botão_Baixo_Estore_E1 -->

<owl:Thing rdf:about="http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome#Botão_Baixo_Estore_E1">
    <rdf:type rdf:resource="&newOntology;Botão"/>
    <rdf:type rdf:resource="&owl;NamedIndividual"/>
    <newOntology:objetoControlado rdf:resource="http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome#A
    <newOntology:estáEm rdf:resource="http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome#Cozinha"/>
    <newOntology:temEstado rdf:resource="http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome#Estado_O
    <newOntology:temFuncionalidade rdf:resource="http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome#
    <newOntology:temFuncionalidade rdf:resource="http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome#
</owl:Thing>
```

Figura 5.6 - Código Gerado para o estore 1 - Botão Baixo.

Outro componente existente na divisão são as tomadas elétricas. Tal como referido na Tabela 5.2, existem quatro na cozinha. Desta forma, foi instanciada uma tomada apenas, multiplicando o código para as outras três. A Figura 5.7 representa a

instanciação das quatro tomadas existentes na divisão. Para cada tomada existente na cozinha, determinadas regras necessitam de ser instanciadas e aplicadas.

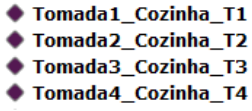


Figura 5.7 - Instanciação das tomadas da cozinha.

A instanciação de cada tomada contém essencialmente cinco normas a aplicar: funcionalidade de notificação de estado mudado, a funcionalidade de ligar e desligar, possibilitar o seu controlo, estar numa parede e estar numa certa localização. A Figura 5.8 representa as propriedades deste dispositivo.

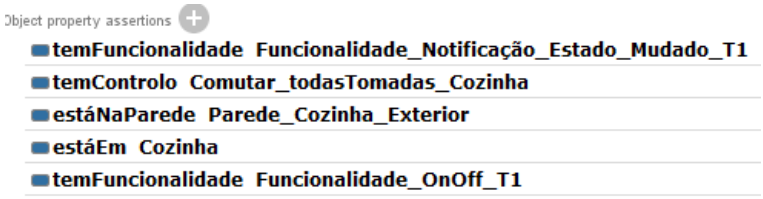


Figura 5.8 - Propriedades da instanciação de uma tomada.

Existem ainda outras subclasses que necessitam de ser instanciadas para compor as propriedades expostas. Um destes exemplos é funcionalidade de ligar e desligar o fornecimento de energia através de uma tomada. Cada uma das tomadas instanciadas possui esta funcionalidade, implicando que na sua composição existe um comando de acionamento e outro de paragem. Desta forma, instanciou-se para cada dispositivo estes dois comandos (Figura 5.9).

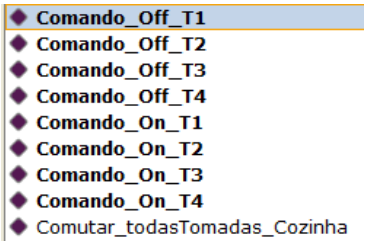


Figura 5.9 - Propriedades associadas à Funcionalidade *OnOff* de uma tomada.

É ainda necessária a instanciação dos diversos valores estado associados a cada tomada. Assim, cada tomada terá um estado (ligado ou desligado) e um valor correspondente a esse estado. Esta situação encontra-se prevista pela ontologia, tal como mencionado no subcapítulo 4.2.2. Relativamente ao código gerado, este representa perto de 500 linhas de código XML, sendo exemplo a Figura 5.10 e Figura 5.11.

```
<!-- http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome#Estado_OnOff_T1 -->

<owl:Thing rdf:about="http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome#Estado_OnOff_T1">
    <rdf:type rdf:resource="&newOntology;Estado_Desligado"/>
    <rdf:type rdf:resource="&newOntology;Estado_Ligado"/>
    <rdf:type rdf:resource="&owl;NamedIndividual"/>
    <newOntology:temValorEstado rdf:resource="http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome#ValorEstado_Off_T1"/>
    <newOntology:temValorEstado rdf:resource="http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome#ValorEstado_On_T1"/>
</owl:Thing>
```

Figura 5.10 - Código Gerado para a tomada 1 - *Estado_OnOff*.

A composição geral da cozinha engloba ainda dois pontos de iluminação que necessitam de ser descritos. Seguiu-se o procedimento optado até este ponto e instanciou-se os componentes necessários seguindo as normas aplicadas à ontologia. Os dois pontos de iluminação foram denominados de "*Iluminação1_Cozinha_I1*" e "*Iluminação2_Cozinha_I2*".

```
    <!-- http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome#Funcionalidade_OnOff_T1 -->

<owl:Thing rdf:about="http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome#Funcionalidade_OnOff_T1">
    <rdf:type rdf:resource="&newOntology;Funcionalidade_de_Acionamento_ON_OFF"/>
    <rdf:type rdf:resource="&owl;NamedIndividual"/>
    <newOntology:temComando rdf:resource="http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome#Comando_Off_T1"/>
    <newOntology:temComando rdf:resource="http://www.semanticweb.org/gerência/ontologies/2013/9/newOntology_ModelHome#Comando_On_T1"/>
</owl:Thing>
```

Figura 5.11 - Código Gerado para a tomada 1 - Funcionalidade *OnOff*.

De seguida aplicaram-se as propriedades que descrevem este bem controlável (Figura 5.12). Estas são iguais para os dois pontos de iluminação.

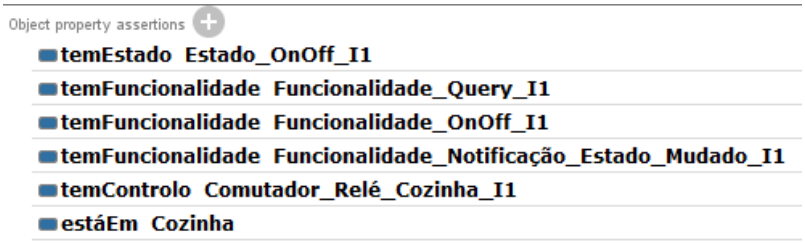


Figura 5.12 - Propriedades do Ponto de Iluminação 1.

À semelhança das tomadas apresentadas anteriormente, o ponto de iluminação também possui funcionalidade de ser ligado ou desligado, um estado associado (ligado ou desligado), a funcionalidade de *query* e notificação do seu estado. É ainda controlado através de um relé que servirá de atuador sobre as lâmpadas. No entanto, a utilização de um relé não implica que a descrição deste componente seja simples. Apesar de se tratar de uma luz simples, não sendo possível ajustar a sua intensidade, o seu controlo depende de muitas variáveis representadas na figura. O relé terá de possuir diversas funcionalidades que trabalhem em prol do sistema geral. Estas encontram-se retratadas na Figura 5.13.

As capacidades deste componente estendem-se face às suas tradicionais aplicações. Não deixando de ser um dispositivo capaz de ativar ou desativar uma lâmpada, deverá facultar informações tais como o seu estado e qual o seu valor. Outro

ponto fundamental é a sua associação à unidade de comando da cozinha relativa à iluminação.

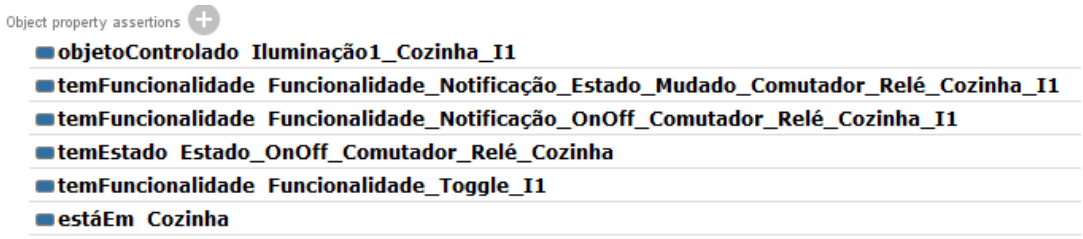


Figura 5.13 - Propriedades do *Comutador_Relé_Cozinha_I1*.

A existência de dois pontos de iluminação levou a que se criasse um controlador único capaz de comandar as iluminações independentemente. Este controlador foi denominado de "*Comutador_Relé_Cozinha*" e a ele estão associados os quatro possíveis estados das duas lâmpadas (ligado ou desligado para cada uma delas). O relé possui desta forma a capacidade de notificar o controlador sobre o seu estado. Na Figura 5.14 encontra-se ilustrada a notificação que o ponto de iluminação dois deverá apresentar a cada mudança de estado. Cada mudança de estado é sinalizada com uma notificação desta forma, contendo como seu argumento o estado da iluminação naquele instante.

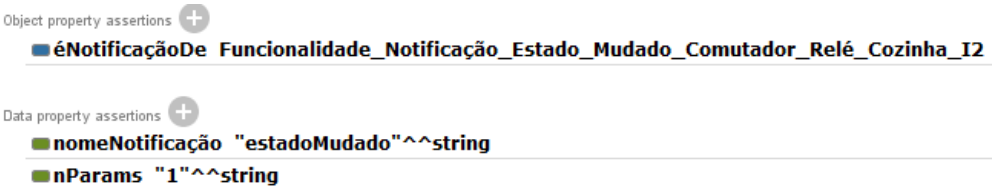


Figura 5.14 - Notificação de estado mudado do ponto de iluminação 2.

O código gerado pela ferramenta é semelhante ao apresentado até este ponto, diferindo apenas no nome e nas classes que compõe a classe mãe. Devido a este facto não será apresentado código relativo a esta instanciação. Para terminar a instanciação da iluminação é necessário ainda descrever o atuador manual desta, o interruptor. As suas funcionalidades são semelhantes aos botões dos estores, diferindo apenas no objeto que controla. Esta modelação encontra-se representada na Figura 5.15.

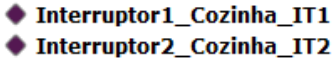


Figura 5.15 - Instanciação dos interruptores da divisão cozinha.

Por fim, é necessário instanciar os restantes bens presentes na divisão cozinha. Os bens não controláveis, tal como mencionado, não interferem diretamente no sistema ontológico no entanto, é necessário representa-los para obter uma descrição do ambiente

mais rica e precisa. Tecnicamente, podem ainda afetar o funcionamento de alguns sensores ou atuadores, sendo assim uma matéria a ter em conta aquando da integração com um *middleware*. Foi adicionado à cozinha uma mesa, quatro cadeiras e quatro móveis. A mesa e as cadeiras encontram-se ligadas através da propriedade "*estáLigado*". Os restantes móveis encontram-se associados às paredes com que fazem costas. A Figura 5.16 é representativa da primeira situação e a Figura 5.17 da segunda.

Figura 5.16 - Instanciação de uma cadeira e respetivas propriedades.

Encontram-se ainda dois móveis na parede para o *Hall* e os restantes na parede da sala.

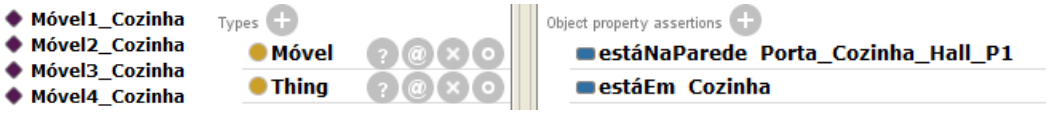


Figura 5.17 - Instanciação de um móvel e respetivas propriedades.

A divisão encontra-se agora totalmente descrita, tendo sido instanciados todos os seus constituintes e respetivas funcionalidades. Podes então validar a construção da ontologia até este ponto. A composição geral da cozinha encontra-se apresentada na Figura 5.18.

Figura 5.18 - Instanciação final da divisão Cozinha.

Findada a instanciação da cozinha, na próxima secção será instanciada a divisão sala.

5.3.2. Caso de estudo 2

O caso de estudo número dois diz respeito à instanciação da divisão sala. Aferiu-se através da análise da Figura 5.1 a Tabela 5.3 que, à semelhança do caso de estudo anterior, foi elaborada de forma a facilitar o processo de identificação de componentes a instanciar.

Tabela 5.3 - Caso de Estudo 2 - Composição da Sala.

Categoria	Objeto	Quantidade
Bens Não Controláveis	Mesa	1
	Cadeira	6
	Móvel	3
	Sofá	3
	Mesa Apoio	1
Bens Controláveis	Tomada	5
	Interruptor	2
	Interruptor de Estore	1
	Lâmpada	2
	Lâmpada Exterior	2

Devido a já ter sido apresentado o procedimento de representação de alguns dos componentes presentes nesta divisão, neste caso de estudo apenas se apresentam as classes mães instanciadas. É o caso da iluminação que foi amplamente retratada no primeiro caso de estudo. Desta forma, nesta secção apenas se encontram representadas as suas classes-mãe, tendo no trabalho sido totalmente representadas, tal como referido. Inicialmente instanciou-se a iluminação interior, existindo duas nesta divisão (Figura 5.19).

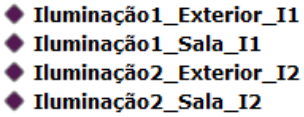


Figura 5.19 - Iluminação interior da divisão sala.

A iluminação necessita ainda de um controlo localizado que se encontra na parede da sala para o quarto e na parede da sala para o exterior. Estes foram adicionados de forma

semelhante aos interruptores presentes na cozinha. Assim foram adicionados dois novos interruptores com o nome "Iluminação1_Sala_I1" e "Iluminação2_Sala_I2" (Figura 5.20).

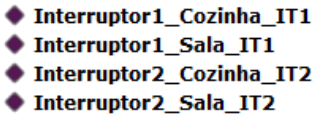


Figura 5.20 - Instanciação dos interruptores da divisão sala.

De notar que à semelhança do caso de estudo 1, estes interruptores são duplos. O interruptor número dois opera a iluminação exterior, enquanto o número um aciona a iluminação interior. Apesar de não haver qualquer objeto controlável à exceção da iluminação exterior na varanda, esta foi também instanciada. Foi denominada de "Varanda_Norte_V1" e possui as características apresentadas na Figura 5.21.

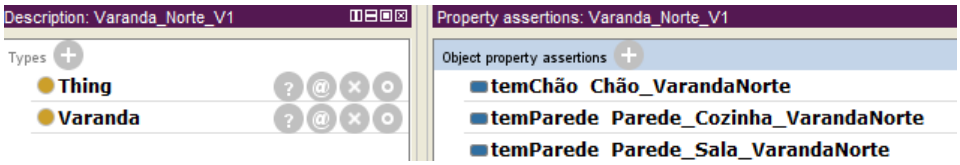


Figura 5.21 - Instanciação da varanda norte.

Naturalmente, através desta descrição é possível aferir que a varanda não se encontra tapada superiormente, não tendo sido assim instanciado o seu teto.

A divisão sala possui ainda cinco tomadas inseridas nas suas paredes. Foram instanciadas as cinco e colocadas nas paredes respetivas tal como mencionado na Figura 5.22. Os elementos de controlo e respetivas propriedades foram também descritos fornecendo assim uma instanciação completa.

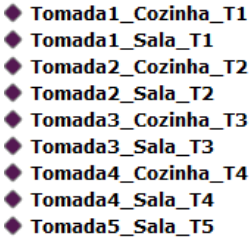


Figura 5.22 - Instanciação das cinco tomadas da divisão sala.

A representação da sala não fica completa sem a instanciação dos bens não controláveis, tendo sido instanciados os seus móveis, as duas mesas, as cadeiras e os sofás. A representação das duas mesas, dos móveis e das cadeiras encontram-se ilustrada na Figura 5.23.

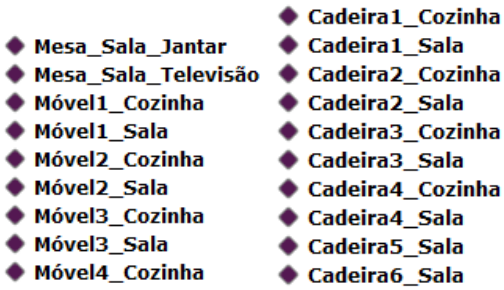


Figura 5.23 - Instanciação dos móveis, mesas e cadeiras da divisão sala.

Por fim, os três sofás presentes na divisão encontram-se na Figura 5.24. À semelhança das mesas e cadeiras, os sofás encontram-se ligados à mesa de apoio à televisão através da propriedade "*estáLigado*".

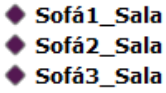


Figura 5.24 - Instanciação dos sofás da divisão sala.

Esta divisão permite que se classifique futuramente a sala em dois tipos: sala de estar e sala de jantar. A propriedade aplicada distinguirá quais os blocos associados a cada tipo de sala. Representadas todas as instanciações necessárias, a divisão encontra-se totalmente classificada, tal como se pode verificar pela Figura 5.25.

Figura 5.25 - Instanciação final da divisão Sala.

Cada componente instanciado tem ainda as subclasses já mencionadas na secção anterior.

5.3.3. Caso de estudo 3

O último caso de estudo apresentado neste documento diz respeito às restantes divisões do ambiente habitacional. Nesta secção será representado o quarto, a casa de banho e a varanda a sul. Irá também conter a instanciação de dois utilizadores que habitam a casa e possuem determinadas preferências. Relativamente aos seus bens, estes encontram-se descritos na Tabela 5.4.

Tabela 5.4 - Caso de Estudo 3 - Composição do Quarto.

Categoria	Objeto	Quantidade
Bens Não Controláveis	Mesa	1
	Mesa Apoio	2
	Móvel	1
	Cama	1
Bens Controláveis	Tomada	3
	Interruptor	3
	Interruptor de Estore	1
	Lâmpada	2
	Lâmpada Exterior	2

A juntar aos bens aferidos para o quarto, há ainda os que necessitam de ser instanciados para descrever corretamente o imóvel. A divisão casa de banho encontra-se associada ao quarto e é composta por alguns bens essenciais. Estes encontram-se agrupados na Tabela 5.5.

Tabela 5.5 - Caso de Estudo 3 - Composição da Casa de Banho.

Categoria	Objeto	Quantidade
Bens Não Controláveis	Sanita	1
	Bidé	1
	Lavatório	1
	Móvel	2
	Chuveiro	1
Bens Controláveis	Tomada	1

Primeiro descreveu-se a iluminação interior e exterior a par dos seus interruptores. De notar que o interruptor número dois do quarto diz respeito ao acionamento das luzes externas e o número três à iluminação da casa de banho. Na Figura 5.26 estão representadas todas as iluminações do imóvel, incluindo as do quarto e casa de banho e os seus interruptores.

Figura 5.26 - Instanciação das diversas iluminações da habitação à esquerda e seus controlos à direita.

Tal como é possível aferir através das zonas luminosas que controlam, alguns interruptores são duplos (caso do interruptor número um da sala) enquanto outros são simples (único interruptor da cozinha). Seguidamente, passou-se à instanciação das três tomadas do quarto e da única tomada presente na casa de banho. Estas seguiram a mesma designação das secções anteriores. Assim, foram denominadas *"Tomada1_Quarto_T1"*, *"Tomada2_Quarto_T2"*, *"Tomada3_Quarto_T3"* e *"Tomada1_CasaBanho_T1"*. Ao nível dos bens controláveis apenas ficava a faltar o interruptor de estores. Este foi denominado de "*Atuador_Estore_Quarto_VarandaSul_E2*" e possui as mesmas propriedades que o estore número um apresentado no primeiro caso de estudo. Relativamente aos bens presentes no imóvel apenas falta descrever os objetos não controláveis presentes nas duas divisões. A Figura 5.27 retrata os móveis e mesas de apoio existentes nas divisões.

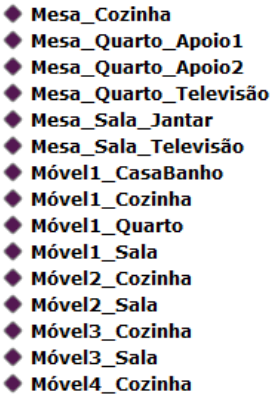


Figura 5.27 - Instanciação dos móveis e mesas existentes na divisão quarto e casa de banho.

Os restantes bens foram também instanciados e estão presentes na descrição global da divisão quarto (Figura 5.28) e casa de banho (Figura 5.29).

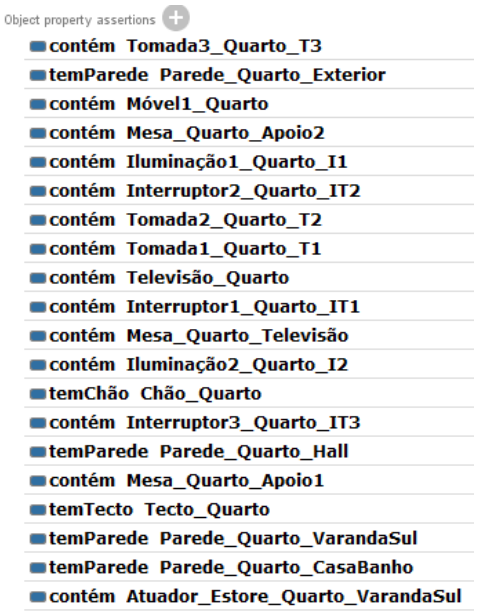


Figura 5.28 - Instanciação final da divisão Quarto.

Figura 5.29 - Instanciação final da divisão Casa de Banho.

Por fim, é necessário instanciar os utilizadores presentes na habitação. Desta forma foram criados dois utilizadores, a "*Ana*" e o "*Diogo*". Estes foram descritos através de várias propriedades a nível dos objetos e dos dados representados pela Figura 5.30.

A caracterização de cada um dos utilizadores poderá ainda ser estendida através

Figura 5.30 - Instanciação dos utilizadores Ana e Diogo.

de várias propriedades adicionadas à ontologia. De ressalvar a propriedade "gosto" onde se podem adicionar *hobbies*, gostos gastronómicos, entre outros. Estes poderão ser utilizados para tornar o ambiente mais rico, fornecendo sugestões a nível de entretenimento ou até de refeições.

5.4. Conclusão

A instanciação da ontologia através dos vários casos de estudo apresentados, veio provar a sua validade e autenticidade, tendo sido possível modelar com sucesso um ambiente habitacional simples. Optou-se por escolher uma habitação do tipo T1, com funcionalidades limitadas, devido à probabilidade de erro que a instanciação manual incute no sistema. Este deve-se à repetição de tarefas monótonas e que exigem total concentração e mentalização do sistema em questão. No entanto, não sendo possível testar a ontologia no seu todo, instanciando todas as situações previstas na mesma, aquando da integração desta num sistema real, a sua instanciação ficará mais simples e automatizada, facilitando assim o teu teste final. A utilização de um *middleware* em conjunto com a ontologia possibilitará que automaticamente se descubram os bens presentes numa habitação e a criação de uma *API* que faça um mapeamento correto entre estes e as instâncias da ontologia. Todos os casos de estudo apresentados foram instanciados na ferramenta *Protégé*, sendo ainda confirmado que o código *RDF/XML* gerado se encontra bem estruturado e claro. Foram apresentados alguns exemplos deste tipo de programação, o que facilitou a quebrar a repetibilidade de algumas instanciações. O acesso ao código gerado provou-se verdadeiramente útil pois através deste foi possível duplicar componentes na habitação. Caso contrário, o tempo dispensado na tarefa seria incomportável e, mais uma vez, monótono e propenso a falhas. O sucesso apresentado nos diversos casos de estudo, vem provar que não só as ontologias são uma forma excecional de representar ambientes complexos, mas também que a ontologia construída é capaz de realizar as tarefas mais triviais de um ambiente de domótica. Certamente necessitará de alguns ajustes para responder efetivamente a questões tecnológicas. No entanto, é uma base muito sólida e com grandes capacidades de evolução.

CAPÍTULO 6

Conclusões

6.1. Conclusões

O domínio dos sistemas de AAL é bastante promissor mas também muito desafiador. De modo a proporcionar uma vida independente a pessoas com necessidades específicas ou até salvar uma vida, estes sistemas têm de lidar com uma série de características do utilizador e do ambiente em que estão inseridos. Estas características resultam num conjunto de serviços e qualidades que todos os sistemas de AAL devem possuir. As características mais notáveis são a disponibilidade, confiabilidade e adaptabilidade. Estabelecer um *tradeoff* entre estas qualidades é complicado e aumenta o nível de dificuldade para a execução destas soluções. Regidas pela necessidade de serem económicas e pelo limite de recursos disponíveis, é claro que estas soluções seguirão a filosofia do "qualidade suficiente" no que diz ao número requisitos a cumprir, de forma racional.

O primeiro artefacto no espaço de soluções que lida com a qualidade explicitamente é a arquitetura. Visto por uma perspetiva funcional, torna-se claro que uma solução de assistência proactiva pode ser considera como uma pequena variação de um controlador em malha fechada. No entanto, a estrutura física das soluções irá variar substancialmente de solução para solução. Atualmente, estão a ser estudados vários estilos arquiteturais que se poderm adequar-se para o domínio de AAL. Infelizmente, nenhum dos estilos tradicionais já existentes podem ser considerados como totalmente adequados. De forma a resolver o exposto, soluções hibridas que combinam diferentes estilos são merecedores de um elevado número de estudos.

No que diz respeito a soluções técnicas que atualmente já se encontram desenvolvidas, pode-se afirmar que as plataformas de *middleware* existentes proporcionam qualidade considerada suficiente para os estilos mais usados, como descrito ao longo do documento. Por outro lado, ainda existe um suporte limitado para

os sistemas híbridos. A qualidade de serviços e a integração de sistemas através das várias plataformas de *middleware* também é um assunto merecedor de grande discussão.

Tal como mencionado no subcaítulo 3.3, um campo promissor da tecnologia para sistemas de AAL é a *Semantic Web*. O *RDF* e o *OWL* pode ajudar à representação da informação sobre o utilizador, sobre o ambiente e o sobre sistema em si de uma forma semiformal. As ferramentas de modelação e *reasoning* possuem o potencial para facilitar soluções baseadas nestes conceitos substancialmente. Já os serviços funcionais, a especificação e a gestão da qualidade de informação continua a ser uma boa temática de pesquisa. Um modelo único condizente com todas as qualidades e interdependências necessárias num sistema deste tipo seria útil. No entanto, ainda não existe presentemente e deverá ser alvo de uma intensiva pesquisa interdisciplinar para os próximos anos. Por outro lado existem os tradicionais sistemas baseados em bases de dados. Apesar das suas vantagens a nível de manutenção, simplicidade e reposta a *queries* sofrem de algumas desvantagens face às ontologias. A maneira lógica como estas lidam com os dados e constroem informação, o seu esquema simples e de fácil manutenção constituem um grande ponto a seu favor. Nas ontologias cada classe pode ter mais do que um nome sendo que nas bases de dados cada nome é único. Outro facto ainda é que se pode considerar que as bases de dados respondem a uma informação concreta tal como quantidades, qualidades ou proprietários. As ontologias podem construir a sua resposta com base em assunções. Podem assim inferir uma solução através dos dados e das regras impostas, sendo mais configuráveis, adaptáveis e uteis em sistemas de AAL.

Nesta dissertação foram apresentados os diversos aspetos tecnológicos que compõe as soluções de *Ambient Assisted Living*. Numa primeira fase do trabalho foi realizado um estudo aprofundado da bibliografia existente, quais os intervenientes nestes sistemas e qual a sua importância. Foi também reconhecida a importância da técnica de modelação mais emergente neste ramo dos últimos anos, as ontologias. A pouca informação sobre esta integração (ontologias em *prol* de sistemas de *AAL*) provou-se um enorme desafio de compreensão e junção de conceitos. Posteriormente, foram apresentadas as diversas técnicas de modelação aferidas ao longo do trabalho de pesquisa. Dividiu-se o capítulo em estilos arquiteturais e estilos semânticos devido à sua separação ténue no domínio aplicacional. A realidade dos sistemas integrados utiliza um estilo arquitetural juntamente com algum modelo capaz de processar e organizar os dados e a informação fornecida. Os modelos semânticos representam uma técnica recente e que tem vindo a ver o seu valor provado em inúmeras aplicações. Em destaque

encontram-se as ontologias, que serviu de objeto aprofundado de estudo neste documento.

No capítulo 4 do documento encontra-se a proposta desenvolvida pelo autor para modelar um ambiente doméstico de automação. Foram apresentados e discutidos todos os constituintes da ontologia, as suas funcionalidades e as suas respetivas restrições.

Por fim, no último capítulo encontram-se simuladas algumas situações reais. Criou-se um ambiente de teste através da utilização de uma habitação T1 recheada com alguns sensores e bens controláveis. Inicialmente começou-se pela descriminação da divisão cozinha, as suas tomadas, iluminação, estores e decoração interna. Posteriormente, avançou-se para a sala, a divisão com mais constituintes. Em último ficou a descrição do quarto e casa de banho. Ao longo destes três casos de estudo foi analisado também o código *RDF/XML* gerado pela ferramenta de modelação ontológica *Protégé*. Através desta funcionalidade da ferramenta foi possível tornar algumas tarefas de instanciação menos sujeitas a erro. Usou-se a duplicação de código (mudando apenas o nome) para instanciar múltiplos componentes semelhantes (caso de tomadas, iluminação, entre outros). A capacidade que esta ontologia provou ter de descrever com elevado grau de detalhe um ambiente habitacional veio provar uma vez mais o valor da tecnologia além do seu valor específico. Desta forma foi possível aferir que a utilização desta técnica deverá continuar a merecer todos os estudos em curso e que finalmente surja uma solução completa de *AAL* baseada em ontologias.

6.2. Sugestões para Trabalho Futuro

Os resultados obtidos neste trabalho permitiram validar a utilização de ontologias em *Ambient Assisted Living*. Contudo, é necessário continuar com os testes efetuados à ontologia criada, não descuidando qualquer classe implementada. Certamente haverão aspetos tecnológicos que, na altura da integração num *middleware*, implicarão alterações em algumas das classes ou propriedades implementadas. Nomeadamente a classe de controlo e de funcionalidades. Esta limitação deve-se ao facto de nem todos os *middlewares* traduzirem de forma transparente os diversos protocolos de comunicação existentes para um formato de dados universal e capaz de interagir com a ontologia.

Em seguida são apresentadas algumas sugestões para desenvolvimentos futuros no sentido de dar continuidade ao trabalho de investigação iniciado nesta Tese:

- Adicionar novos bens não controláveis como forma de enriquecer a descrição do ambiente;

- Adicionar novos sensores/controladores e atuadores que possam fazer parte de um ambiente de automação doméstica;
- Adicionar funcionalidades que por ventura estejam em falta para alguns dispositivos;
- Testar intensivamente, através de instanciações as restantes classes da ontologia;
- Integração da ontologia num *middleware* conhecido, tal como o *DogOnt* o *OSGi*;
- Construir um modelo físico de uma habitação semelhante à apresentada, carregar a ontologia e aferir se as situações previstas funcionam como seria espectável;
- Aplicar mais regras SWRL e a utilização de um *reasoner*.

Como forma de validação completa, seria vantajoso comparar a solução final construída com outras existentes no mercado ou que tenham sido desenvolvidas internamente para o estudo.

Referências

[1] K. Lucero, S., Burden, "Home Automation and Control," *ABI Res.*, 2010.

[2] "Ambient Assisted Living Joint Programme - Catalogue of Projects 2012," *[Online]. Available: http://www.aal-europe.eu/wp-content/uploads/2012/08/AALCatalogue2012_V7.pdf. [Accessed: 21-Set-2013]*, 2012.

[3] "SOPRANO," 2008. [Online]. Available: http://www.soprano-ip.org/. [Accessed: 12-Jan-2013].

[4] P. Wolf, A. Schmidt, and M. Klein, "SOPRANO-An extensible, open AAL platform for elderly people based on semantical contracts," *... Ambient Intell. (AITAmI'08), 18th ...*, no. Ecai 08, 2008.

[5] M. Klein, A. Schmidt, and R. Lauer, "Ontology-centred design of an ambient middleware for assisted living: The case of soprano," *... Ambient Intell. Methods ...*, 2007.

[6] A. Sixsmith, "User requirements for Ambient Assisted Living : Results of the SOPRANO project."

[7] C. D. Nugent, L. Galway, L. Chen, M. P. Donnelly, S. I. Mcclean, S. Zhang, B. W. Scotney, and G. Parr, "Managing Sensor Data in Ambient Assisted Living," *J. Comput. Sci. Eng.*, vol. 5, no. 3, pp. 237–245, Sep. 2011.

[8] M. Story, J. Mueller, and R. Mace, "The universal design file," *Cent. Univers. Des.*, 1998.

[9] and C. D. A. B. Brush, B. Lee, R. Mahajan, S. Agarwal, S. Saroiu, "Home Automation in the Wild: Challenges and Opportunities," *ACM CHI*, 2011.

[10] M. Becker, *Software architecture trends and promising technology for ambient assisted living systems*, no. i. 2008.

[11] E. VDE ASSOCIATION FOR ELECTRICAL and & I. TECHNOLOGIES, "The German AAL Standardization Roadmap," 2012.

[12] R. I. Damas M, Pomares H, Gonzalez S, Olivares A, "Ambient assisted living devices interoperability based on OSGi and the X73 standard," *Telemed J E Heal.*, 2013.

[13] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications by A Translation Approach to Portable Ontology Specifications," *Knowl. Acquis. 5*, no. April, pp. 199–220, 1993.

[14] I. Normann and W. Putz, "Ontologies and reasoning for ambient assisted living," *... Conf. AALiance, Malaga*, 2010.

[15] Tim Berners-Lee, "World Wide Web Consortium," 1994. [Online]. Available: http://www.w3.org/. [Accessed: 13-Jan-2013].

[16] P. Jorge, "The Semantic Web vision : not only human but also machine readable Web."

[17] N. Suksom and M. Buranarach, "A Knowledge-based Framework for Development of Personalized Food Recommender System," *... Support Syst.*, 2010.

[18] S. Stoutenburg, L. Obrst, D. Mccandless, D. Nichols, P. Franklin, M. Prausa, and R. Sward, "Ontologies for Rapid Integration of Heterogeneous Data for Command , Control , & Intelligence 1 Introduction 2 Use Case," pp. 37–42, 2008.

[19] B. O'Flynn and P. Angove, "Wireless biomonitor for ambient assisted living," *Oral Present. ...*, 2006.

[20] C. N. Liming Chen, "Ontology-based activity recognition in intelligent pervasive environments," *Int. J. Web Inf. Syst.*, vol. 5, no. 4, pp. 410–430, 2009.
[21] H. Steg, "Ambient Assisted Living – European overview report," 2005.
[22] A. Home, "Aware Home," 2008. [Online]. Available: http://awarehome.imtc.gatech.edu/. [Accessed: 25-Sep-2013].
[23] U. of Illinois, "I-Living," *Urbana-Champaign, Assisted Living Project*. [Online]. Available: http://lion.cs.uiuc.edu/assistedliving. [Accessed: 23-Sep-2013].
[24] P. Remagnino and G. L. Foresti, "Ambient Intelligence: A New Multidisciplinary Paradigm," *IEEE Trans. Syst. Man, Cybern. - Part A Syst. Humans*, vol. 35, no. 1, pp. 1–6, Jan. 2005.
[25] V. Fuchsberger, "Ambient assisted living," in *Proceeding of the 1st ACM international workshop on Semantic ambient media experiences - SAME '08*, 2008, p. 21.
[26] M. Vacher, F. Portet, A. Fleury, and N. Noury, "Development of Audio Sensing Technology for Ambient Assisted Living," *Int. J. E-Health Med. Commun.*, vol. 2, no. 1, pp. 35–54, Jan. 2011.
[27] A. J. Jara, M. A. Zamora, and A. F. G. Skarmeta, "An internet of things–based personal device for diabetes therapy management in ambient assisted living (AAL)," *Pers. Ubiquitous Comput.*, vol. 15, no. 4, pp. 431–440, Jan. 2011.
[28] H. Sun, V. De Florio, N. Gui, and C. Blondia, "Promises and Challenges of Ambient Assisted Living Systems," in *2009 Sixth International Conference on Information Technology: New Generations*, 2009, pp. 1201–1207.
[29] "Open Service Gateway initiative (OSGI)," 2008. [Online]. Available: http://www.osgi.org. [Accessed: 15-May-2013].
[30] M. Aiello and S. Dustdar, "Are our homes ready for services? A domotic infrastructure based on the Web service stack," *Pervasive Mob. Comput.*, vol. 4, no. 4, pp. 506–525, Aug. 2008.
[31] "Web Ontology Language for Web Services (OWL-S)," *(T.C), OWL-S Technical Committee*, 2002. [Online]. Available: http://www.w3.org/Submission/OWL-S/. [Accessed: 23-May-2013].
[32] S. Tang, "The TUB OWL-S Matcher," 2008. [Online]. Available: http://owlsm.projects.semwebcentral.org/. [Accessed: 23-May-2013].
[33] N. Srinivasan, "OWL-S UDDI Matchmaker." [Online]. Available: http://projects.semwebcentral.org/projects/owl-s-uddi-mm/. [Accessed: 23-May-2013].
[34] K. S. M. Klusch, B. Fries, M Khalid, "OWL-MX Matcher," 2008. [Online]. Available: http://projects.semwebcentral.org/frs/?group_id=90.
[35] N. Georgantas, "Amigo Middleware Core: Prototype Implementation & Documentation," *IST Amigo Proj. Deliv. D3.2*, 2006.
[36] C. B. H. Sun, V. De Florio, N. Gui, "Towards Longer, Better, and More Active Lives - Building Mutual Assisted Living Community for Elder People," *Proc. 47th Eur. FITCE Congr. FITCE, London*, 2008.
[37] P. T. Kleinberger, M. Becker, E. Ras, A. Holzinger and Muller, "Ambient Intelligence in Assisted Living: Enable Elderly People to Handle Future Interfaces," *Univers. Access Human-Computer Interact. Ambient Interact. Part II*, 2007.
[38] L. L. M. Floeck, "Aging in Place: Supporting Senior Citizens' Independence with Ambient Assistive Living Technology," 2007.
[39] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen, "From SHIQ and RDF to OWL: the making of a Web Ontology Language," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 1, no. 1, pp. 7–26, Dec. 2003.

[40] A. Doan, J. Madhavan, P. Domingos, and A. Halevy, "Learning to map between ontologies on the semantic web," in *Proceedings of the eleventh international conference on World Wide Web - WWW '02*, 2002, p. 662.

[41] A. Oracle and W. Paper, "Next Generation Service Integration Platform," no. June, pp. 1–15, 2013.

[42] "Everware-CBDI," 1990. [Online]. Available: http://www.cbdiforum.com/. [Accessed: 24-Apr-2013].

[43] D. S. and L. Wilkes, "Understanding Service-Oriented Architecture," *Gives a concise explanation of service-oriented architecture, what it is, and how it affects what architects, CIOs, project managers, business analysts, and lead developers do*, 2004. [Online]. Available: http://msdn.microsoft.com/en-us/library/aa480021.aspx. [Accessed: 21-Sep-2013].

[44] F. Jammes and H. Smit, "Service-oriented architectures for devices - the SIRENA view," in *INDIN '05. 2005 3rd IEEE International Conference on Industrial Informatics, 2005.*, pp. 140–147.

[45] S. de Deugd, R. Carroll, K. Kelly, B. Millett, and J. Ricker, "SODA: Service Oriented Device Architecture," *IEEE Pervasive Comput.*, vol. 5, no. 3, pp. 94–96, c3, Jul. 2006.

[46] S. de Deugd, R. Carroll, K. Kelly, B. Millett, and J. Ricker, "SODA: Service Oriented Device Architecture," *IEEE Pervasive Comput.*, vol. 5, no. 3, pp. 94–96, c3, Jul. 2006.

[47] R. Rodrigues and P. Druschel, "Peer-to-peer systems," *Commun. ACM*, vol. 53, no. 10, p. 72, Oct. 2010.

[48] U. Dahan, "Event-Driven Architecture: SOA Through the Looking Glass," 2009. [Online]. Available: http://msdn.microsoft.com/en-us/architecture/aa699424.aspx. [Accessed: 22-Sep-2013].

[49] D. Chou, "Using Events in Highly Distributed Architectures," 2008. [Online]. Available: http://msdn.microsoft.com/en-us/library/dd129913.aspx. [Accessed: 22-Sep-2013].

[50] U. Dahan, "Event-Driven Architecture: SOA Through the Looking Glass," 2009. .

[51] G. Neto, "From Single-Agent to Multi-Agent Reinforcement Learning: Foundational Concepts and Methods," *Learn. Theory Course*, no. May, 2005.

[52] J. McNaull, J. C. Augusto, M. Mulvenna, and P. McCullagh, "Multi-agent Interactions for Ambient Assisted Living," in *2011 Seventh International Conference on Intelligent Environments*, 2011, pp. 310–313.

[53] L. D. Erman, F. Hayes-Roth, V. R. Lesser, and D. R. Reddy, "The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty," *ACM Comput. Surv.*, vol. 12, no. 2, pp. 213–253, Jun. 1980.

[54] C. K. Kwong and G. F. Smith, "A computational system for process design of injection moulding: Combining blackboard-based expert system and case-based reasoning approach," *Int. J. Adv. Manuf. Technol.*, vol. 14, no. 4, pp. 239–246, Apr. 1998.

[55] D. Corkill, "Blackboard systems," *AI Expert*, vol. 6, no. September, pp. 40–47, 1991.

[56] J. Maréchaux, "Combining service-oriented architecture and event-driven architecture using an enterprise service bus," *IBM Dev. Work.*, no. April, pp. 1–8, 2006.

[57] F. M. Hugh Taylor, Angela Yochem, Les Phillips, *Event-Driven Architecture: How SOA Enables the Real-Time Enterprise*. Addison-Wesley, 2009.

[58] L. Vajda, "BelAmI JT6: Assisted Living and Working." .

[59] L. V. Ákos Nagy, András Szabó, "BelAmI JT6: Assisted Living and Working Presentation." .

[60] T. I. M. Berners-lee, J. Hendler, and O. R. A. Lassila, "The Semantic Web," no. May, pp. 1–4, 2001.

[61] R. Denaux, L. Aroyo, and V. Dimitrova, "An approach for ontology-based elicitation of user models to enable personalization on the semantic web," in *Special interest tracks and posters of the 14th international conference on World Wide Web - WWW '05*, 2005, p. 1170.

[62] and M. D. Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," 2004. [Online]. Available: http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/. [Accessed: 03-Jun-2013].

[63] S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, and I. Horrocks, "The Semantic Web: the roles of XML and RDF," *IEEE Internet Comput.*, vol. 4, no. 5, pp. 63–73, 2000.

[64] J. Z. Pan, *Handbook on Ontologies*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 71 – 90.

[65] S. Brockmans and R. Colomb, "A model driven approach for building OWL DL and OWL full ontologies," *Semant. Web- …*, 2006.

[66] S. Bechhofer, R. Volz, and P. Lord, "Cooking the Semantic Web with the OWL API," *Semant. Web-ISWC 2003*, 2003.

[67] G. S. Mark van Assem, , Aldo Gangemi, "Conversion of WordNet to a standard RDF/OWL representation," 2006. .

[68] A. Newell, "The Knowledge Level Perspective," *Artif. Intell.*, vol. 18, 1982.

[69] H. J. Levesque, "Foundations of a functional approach to knowledge representation," *Artificial Intell.*, pp. 23:155– 212, 1984.

[70] N. F. Noy and D. L. Mcguinness, "Ontology Development 101 : A Guide to Creating Your First Ontology," pp. 1–22, 2000.

[71] "Ontolingua," *Ontolingua provides a distributed collaborative environment to browse, create, edit, modify, and use ontologies.* [Online]. Available: http://www.ksl.stanford.edu/software/ontolingua/. [Accessed: 05-Dec-2012].

[72] "DAML," *The goal of the DAML effort is to develop a language and tools to facilitate the concept of the Semantic Web*. [Online]. Available: http://www.daml.org/ontologies/. [Accessed: 05-Dec-2012].

[73] "RosettaNet," *RosettaNet develops universal standards for the global supply chain.* [Online]. Available: http://www.rosettanet.org/. [Accessed: 05-Dec-2012].

[74] "DMOZ," *The Open Directory Project is the largest, most comprehensive human-edited directory of the Web. It is constructed and maintained by a vast, global community of volunteer editors.* [Online]. Available: http://www.dmoz.org. [Accessed: 05-Dec-2012].

[75] M. Uschold and M. Gruninger, "Ontologies: Principles, methods and applications," *Knowl. Eng. Rev.*, no. February, 1996.

[76] "Protégé," *Protégé is a free, open source ontology editor and knowledge-base framework.* [Online]. Available: http://protege.stanford.edu/. [Accessed: 14-Dec-2012].

[77] "NeOn Toolkit," *The NeOn Toolkit is a ontology engineering environment*. [Online]. Available: http://neon-toolkit.org/wiki/Main_Page. [Accessed: 14-Dec-2012].

[78] "Vitro," *Vitro is a general-purpose web-based ontology and instance editor with customizable public browsing*. [Online]. Available: http://vitro.mannlib.cornell.edu/. [Accessed: 14-Dec-2012].

[79] "IBM Integrated Ontology Development Kit," *IODK is a toolkit for ontology-driven developmen*. .
[80] "Pellet," *Pellet is an OWL 2 reasoner*. [Online]. Available: http://clarkparsia.com/pellet/. [Accessed: 17-Dec-2012].
[81] "FACT plus plus," *FaCT++ is a DL reasoner*. [Online]. Available: https://code.google.com/p/factplusplus/. [Accessed: 17-Dec-2012].
[82] K. Dentler, R. Cornet, A. ten Teije, and N. de Keizer, "Comparison of reasoners for large ontologies in the OWL 2 EL profile," *Semant. Web*, vol. 2, pp. 71–87, 2011.
[83] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson, "Jena," in *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters - WWW Alt. '04*, 2004, p. 74.
[84] M. Horridge, S. Bechhofer, and O. Noppens, "Igniting the OWL 1.1 Touch Paper: The OWL API.," *OWLED*, 2007.
[85] M. USCHOLD, "Knowledge level modelling: concepts and terminology," *Knowl. Eng. Rev.*, vol. 13, no. 1, pp. 5–29, Mar. 1998.
[86] Y. Wand, V. C. Storey, and R. Weber, "An ontological analysis of the relationship construct in conceptual modeling," *ACM Trans. Database Syst.*, vol. 24, no. 4, pp. 494–528, Dec. 1999.
[87] A. Borgida, "Description logics in data management," *IEEE Trans. Knowl. Data Eng.*, vol. 7, no. 5, pp. 671–682, 1995.
[88] E. F. Codd, "A relational model of data for large shared data banks," *Commun. ACM*, vol. 13, no. 6, pp. 377–387, Jun. 1970.
[89] W. Kim, *Introduction to object-oriented databases*. 1990.
[90] S. Marcus and V. S. Subrahmanian, "Foundations of multimedia database systems," *J. ACM*, vol. 43, no. 3, pp. 474–523, May 1996.
[91] M. Graves, *Designing XML Databases*. Prentice Hall PTR, 2001, p. 688.
[92] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick, "CLASSIC: a structural data model for objects," in *Proceedings of the 1989 ACM SIGMOD international conference on Management of data - SIGMOD '89*, 1989, pp. 58–67.
[93] R. Reiter, *On Conceptual Modelling*. New York, NY: Springer New York, 1984, pp. 191–238.
[94] R. Meersman, "Ontologies and Databases: More than a Fleeting Resemblance."
[95] L. Yan, Z. M. Ma, and J. Liu, "Fuzzy data modeling based on XML schema," in *Proceedings of the 2009 ACM symposium on Applied Computing - SAC '09*, 2009, p. 1563.
[96] N. F. Noy, "Tools for Mapping and Merging Ontologies," *Handb. Ontol.*, 2004.
[97] T. Tran, H. Lewen, and P. Haase, "On the role and application of ontologies in information systems," *... IEEE Int. Conf.*, pp. 14–21, Mar. 2007.
[98] M. G. M. U. M. G. Mike Uschold, "Ontologies: Principles, methods and applications."
[99] N. Konstantinou, D.-E. Spanos, and N. Mitrou, "Ontology and database mapping: a survey of current implementations and future directions," *J. Web Eng.*, vol. 7, no. 1, pp. 1–24, Mar. 2008.
[100] T. Dillon, E. Chang, M. Hadzic, and P. Wongthongtham, "Differentiating conceptual modelling from data modelling, knowledge modelling and ontology modelling and a notation for ontology modelling," pp. 7–17, Jan. 2008.
[101] M. d'Aquin, C. Baldassarre, and L. Gridinoc, "Watson: Supporting next generation semantic web applications," 2007.
[102] Y. Zhang, W. Vasconcelos, and D. Sleeman, "OntoSearch : An Ontology Search Engine 1 IKB : Identify Knowledge Base," 2005.

[103] L. Ding, T. Finin, A. Joshi, and R. Pan, “Swoogle: a search and metadata engine for the semantic web,” *Proc. ...*, 2004.
[104] R. Reiter, “Towards a Logical Reconstruction of Relational Database Theory,” *Concept. Model.*, pp. 191–238, 1982.
[105] N. Cullot, C. Parent, S. Spaccapietra, and C. Vangenot, “Ontologies: A contribution to the DL/DB debate.,” *SWDB*, pp. 1–21, 2003.
[106] E. Franconi, “Ontologies and databases: myths and challenges,” *Proc. VLDB Endow.*, pp. 1518–1519, 2008.
[107] P. Spyns, R. Meersman, and M. Jarrar, “Data modelling versus ontology engineering,” *ACM SIGMOD Rec.*, vol. 31, no. 4, p. 12, Dec. 2002.
[108] J. Renz and B. Nebel, “On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the Region Connection Calculus,” *Artif. Intell.*, vol. 108, no. 1–2, pp. 69–123, Mar. 1999.
[109] G. L. J. Renz, “Weak Composition for Qualitative Spatial and Temporal Reasoning,” *Elev. Int. Conf. Princ. Pract. Constraint Program.*, 2005.
[110] F. Furfari, L. Sommaruga, C. Soria, and R. Fresco, “DomoML,” in *Proceedings of the 2nd European Union symposium on Ambient intelligence - EUSAI ’04*, 2004, p. 41.
[111] L. Sommaruga, A. Perri, and F. Furfari, “DomoML-env : an ontology for Human Home Interaction,” pp. 1–7.

Anexos

Anexo 1 – Cenários de Automação

- **Iluminação:**
  - Desligar luzes quando o utilizador adormece;
  - Ligar/Desligar luzes quando existem pessoas na casa;
  - Ligar/Desligar luzes quando existem pessoas na casa e está de noite;
  - Ligar/Desligar luzes quando passam pessoas nas divisões;
  - Ligar/Desligar luzes quando o utilizador se direciona para uma divisão;
  - Ligar/Desligar luzes recorrendo à utilização de gestos (estilo *Kinect*);
  - Alterar cor da iluminação de acordo com o utilizador na divisão;
  - Alterar intensidade da iluminação de acordo com o utilizador na divisão;
  - Alterar intensidade da iluminação quando o utilizador está a ver um filme (baixar a intensidade da luz);
  - Alterar intensidade da iluminação quando o utilizador está a ler um livro (focar na iluminação perto dele);
  - Alterar intensidade da iluminação quando o utilizador está na cama (se estiver perto da hora de dormir, baixar intensidade);
  - Alterar intensidade da iluminação quando o utilizador está no computador (fornecer boa iluminação na zona – saúde);
  - Alterar intensidade da iluminação conforme o período do dia (manhã, tarde, noite) e época (estação do ano) e força da luz no exterior;
  - Fornecer um modo de iluminação progressiva (intensidade incrementada) para quando o utilizador acorda;
  - Alterar intensidade da iluminação no exterior conforme o período do dia (manhã, tarde, noite), época (estação do ano);
  - Ligar/Desligar luz quando se toca à campainha (luz do exterior);
  - Ligar/Desligar luzes remotamente (aplicação mobile, p.e.);
  - Abrir/Fechar estores conforme a razão de iluminação interior/exterior;
  - Abrir/Fechar estores conforme o utilizador presente na divisão;
  - Abrir/Fechar estores remotamente (aplicação mobile, p.e.);
  - Abrir/Fechar estores recorrendo à utilização de gestos (estilo Kinect);
  - Controlar inclinação dos estores conforme razão de iluminação interior/exterior.

- **Som Ambiente:**
  - Ligar/Desligar o som ambiente ao *home theater* (ver um filme, p.e.);
  - Ligar/Desligar o som ambiente conforme a pessoa presente na divisão;
  - Ligar/Desligar o som ambiente, em modo rádio, conforme a pessoa presente na divisão e quando esta está a acordar;
  - Alterar o som a tocar na divisão conforme a pessoa presente na divisão;
  - Alterar o volume do som a tocar na divisão conforme a pessoa presente na divisão;
  - Alterar o volume e tom do som a tocar pela campainha de entrada conforme a pessoa que está no exterior;
  - Reproduzir o nome da pessoa no exterior quando esta toca à campainha (se for reconhecida);
  - Alterar o volume da TV quando o telefone toca;
  - Alterar o volume do som ambiente quando o telefone toca;
  - Alterar o volume da TV quando a campainha toca;
  - Alterar o volume do som ambiente quando a campainha toca;
  - Reproduzir no som ambiente as tarefas que o utilizador tem de realizar no dia (quando acorda ou quando vai sair de casa, p.e.);
- **Climatização**
  - Ligar/Desligar climatização quando existem pessoas na casa;
  - Ligar/Desligar climatização quando o utilizador adormece ou está prestes a acordar;
  - Ligar/Desligar climatização recorrendo à utilização de gestos (estilo *Kinect*);
  - Alterar o valor da climatização de acordo com o utilizador na divisão;
  - Alterar o valor da climatização conforme o período do dia (manhã, tarde, noite) e época (estação do ano);
  - Ligar/Desligar o aquecimento da cama conforme o período do dia (se o utilizador está prestes a adormecer ou acordar);
  - Ligar/Desligar o aquecimento dos guarda-fatos conforme o período do dia (se o utilizador está prestes a adormecer ou acordar);
  - Controlar inclinação dos estores conforme a temperatura exterior;
  - Abrir/Fechar estores conforme o período do dia de modo a poupar no aquecimento/arrefecimento da casa;

- **Painéis de notificação/controlo:**
  - Fornecer integração com ambiente displays;
  - Mostrar na TV a imagem capturada pelo videoporteiro;
  - Alterar imagens presentes nos painéis conforme o utilizador presente na divisão;
  - Fornecer uma lista de produtos presentes no frigorífico;
  - Criar uma lista de produtos em falta no frigorífico (que usualmente lá estão);
  - Fornecer uma lista de produtos presentes na despensa;
  - Criar uma lista de produtos em falta na despensa (que usualmente lá estão);
  - Fornecer uma lista de tarefas que o utilizador tem de completar para o dia;
  - Fornecer uma lista de contas que o utilizador tem para pagar;
  - Fornecer uma notificação sobre a existência de correspondência na caixa de correio;
  - Fornecer uma lista de aniversários a decorrer num futuro próximo;
  - Fornecer uma agenda virtual para questões profissionais;
  - Fornecer uma lista de programas que irão decorrer na TV que poderão ser do agrado do utilizador;
  - Fornecer uma lista de programas que irão decorrer na rádio que poderão ser do agrado do utilizador;
  - Fornecer uma lista de programas que irão decorrer na internet que poderão ser do agrado do utilizador;
  - Fornecer uma notificação visual sobre o clima que se fará sentir no dia e nos dias seguintes;
  - Fornecer um aviso visual sobre a falta de peças de roupa adequadas ao clima (podem estar para lavar, p.e.);
  - Fornecer uma notificação visual sobre o estado do trânsito. Enfâse nas localizações de interesse do utilizador;
  - Fornecer uma lista de possíveis receitas num painel junto à cozinha;
  - Fornecer método de transferir conteúdos de um monitor ligado numa divisão para outra (computador ligado a um monitor no escritório estar a dar imagem na sala);

- Fornecer uma notificação visual e sonora sobre o estado de bateria dos dispositivos que irão ser necessários para o dia;
- Fornecer um método de sincronização entre os aparelhos móveis e o *core* da casa para calendários, imagens, mensagens, etc.;
- Direcionar automaticamente as chamadas do telefone de casa para o telemóvel do utilizador, quando este não se encontra em casa;
- Habilitar/Desabilitar notificações (telefonemas, mensagens, e-mails, etc.) em situações que o utilizador não quer ser incomodado (está a fazer desporto, p.e.);
- Controlar manualmente, recorrendo ao ecrã táctil, a temperatura da casa;
- Controlar manualmente, recorrendo ao ecrã táctil, a iluminação da casa;
- Controlar manualmente, recorrendo ao ecrã táctil, os estores da casa;
- Controlar manualmente, recorrendo ao ecrã táctil, as janelas da casa;
- Controlar manualmente, recorrendo ao ecrã táctil, alguns eletrodomésticos da casa;
- Controlar manualmente, recorrendo ao ecrã táctil, o sistema de rega da casa;
- Controlar manualmente, recorrendo ao ecrã táctil, a temperatura da água da casa;
- Controlar manualmente, recorrendo ao ecrã táctil, as portas da casa;

- **Miscelâneas:**
  - Fechar os portões quando se deliga o carro;
  - Monitorizar a qualidade do ar e abrir janelas ou ligar purificador. De preferência quando o utilizador não estiver em casa;
  - Monitorizar os parâmetros básicos de saúde para percepcionar o seu estado de saúde. Adaptar ambiente (luz, temperatura e som) conforme essa medição;
  - Ligar/Desligar exaustores ou ventilações quando se detetam odores indesejados (cozinha, WC);

- **Segurança da casa/bens:**
  - Ativar automaticamente o alarme quando o utilizador sai de casa;
  - Fechar automaticamente as janelas de casa quando o utilizador sai de casa;

- Trancar automaticamente portas e garagem quando o utilizador sai de casa;
- Fechar automaticamente os estores quando o utilizador sai de casa;
- Trancar automaticamente as portas quando se deteta uma pessoa estranha à casa no seu exterior;
- Se for detetado movimente no exterior durante a noite, ligar as luzes onde o movimento é detetado;
- Trancar todas as portas e janelas à noite, quando o utilizador for dormir;
- Ativar o alarme automaticamente quando o utilizador for dormir;
- Permitir simular a presença de pessoas em casa (ligar/desligar luzes) quando o utilizador se ausenta para férias;
- Desligar automaticamente a água em situações de inundação;
- Desligar localmente o fornecimento de eletricidade em situações de inundação;
- Desligar automaticamente a água de uma torneira em caso de esquecimento;
- Desligar o fornecimento de gás em caso de fuga;
- Desligar o fornecimento de gás e eletricidade em situação de fumo;
- Notificar o utilizador para a presença de pessoas estranhas à casa;
- Enviar imagens das camaras de vigilância para o telemóvel/portátil do utilizador quando alguém se aproxima de casa (se não estiver ninguém lá);
- Detetar intrusão a partir de janelas e notificar o utilizador;
- Permitir o reconhecimento de impressões digitais para entrar em casa;
- Recolha automática do correio quando o dono se ausenta de casa durante algum tempo;
- Fornecer uma notificação quando existem intrusões na rede wi-fi da casa;
- Notificar quando os níveis de bateria dos detetores de fumo estão baixos;
- Notificar quando os níveis de bateria dos detetores de fugas de água estão baixos;
- Notificar quando os níveis de bateria dos detetores de inundações estão baixos;
- Notificar quando os níveis de bateria do alarme estão baixos;
- Notificar o utilizador em caso de falha dos dispositivos de corte de água;
- Notificar o utilizador em caso de falha dos dispositivos de corte de gás;
- Notificar o utilizador em caso de falha dos dispositivos de corte de energia elétrica;
- Notificar em caso de sobrecargas elétricas nas divisões;

- Em caso de sobrecargas elétricas, o sistema deve desligar alguns dos aparelhos;
- Avisar se houver elementos que possam impedir as portas e janelas de fechar;
- Impedir que as portas e janelas se fechem se existir algum elemento que as possa bloquear;
- Impedir que a porta e/ou janela afeta à divisão da varanda se feche quando o utilizador sai através dela;
- Em caso de situação de incêndio, destrancar todas as portas;
- Em caso de falha de energia elétrica, ligar automaticamente às fontes de energia renováveis;

- **Segurança Pessoal:**
  - Impedir que a temperatura ambiente ultrapasse os 30ºC;
  - Impedir que a temperatura da água no WC seja superior a 38ºC;
  - Monitorizar o estado de saúde dos utilizadores e lançar alertas caso seja detetado algum problema de saúde (ligar a familiares, ligação com médico, etc);
  - Monitorizar utilizador e em caso de queda ligar para números de emergência;
  - Monitorização e alerta em caso de crianças em zonas de risco (varandas, janelas abertas, etc);
  - Fechar cobertura da piscina quando não estiver a ser utilizada;
  - Trancar armário com objetos perigosos (armas, bastões, etc) na presença de crianças;
  - Trancar armário dos produtos químicos na presença de crianças;
  - Impedir ligar/desligar de eletrodomésticos por crianças;
  - Ligar exaustores ou ventilações se forem detetados gases tóxicos;
  - Quando for premido o botão de pânico, ligar para familiares;